DOI: 10.1111/cgf.13909
EUROGRAPHICS 2020 / U. Assarsson and D. Panozzo
(Guest Editors)

Volume 39 (2020), Number 2

Fast and Scalable Solvers for the Fluid Pressure Equations with Separating Solid Boundary Conditions

Junyu Lai[1], Yangang Chen[2], Yu Gu[1], Christopher Batty[1] and Justin W.L. Wan[1]

[1]David R. Cheriton School of Computer Science, University of Waterloo, Canada
[2]Department of Applied Mathematics, University of Waterloo, Canada

Abstract

In this paper, we propose and evaluate fast, scalable approaches for solving the linear complementarity problems (LCP) arising from the fluid pressure equations with separating solid boundary conditions. Specifically, we present a policy iteration method, a penalty method, and a modified multigrid method, and demonstrate that each is able to properly handle the desired boundary conditions. Moreover, we compare our proposed methods against existing approaches and show that our solvers are more efficient and exhibit better scaling behavior; that is, the number of iterations required for convergence is essentially independent of grid resolution, and thus they are faster at larger grid resolutions. For example, on a 256^3 grid our multigrid method was 30 times faster than the prior multigrid method in the literature.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ *Physical simulation;*

1. Introduction

Liquids are ubiquitous in our daily life: the water from the sea, the coffee in our cups, the blood running through our veins, and more. Motivated by the demand for realistic visual effects in the film industry, liquid animation has been explored in the computer graphics community for many years. The Navier-Stokes equations are commonly used to perform the simulations, using a projection method [Bri08] consisting of of three main steps: advecting the liquid and its velocities through the flow; applying body forces such as gravity; and performing *pressure projection* to enforce incompressibility and boundary conditions. The pressure projection step presents two challenges considered in this work: computation time and visual behavior. First, solving the pressure equation often comprises a significant fraction of the simulation time [DRW15], so it is therefore important to develop a fast and scalable numerical approach. We consider a method to be scalable if the number of iterations is essentially independent of the mesh resolution. Second, standard solid boundary conditions do not allow liquid to naturally separate from a solid boundary; instead, the liquid unnaturally adheres to the top and side walls of a domain. To resolve this issue, Batty et al. [BBB07] proposed a new inequality boundary condition for the liquid-solid wall that allows separation while disallowing penetration. However, while this corrects the behavior, it transforms the pressure equation from a standard linear system into a linear complementarity problem (LCP) which is even more challenging to solve efficiently. As a result, this improved boundary condition has seldom been adopted in practice.

In this paper, we propose to develop and evaluate variants of policy iteration [FL07], a penalty method [dFL04], and full approximation scheme multigrid (FAS-MG) [HW13] applied to the LCP fluid problem, because such schemes are known to be convergent and efficient. While numerical schemes belonging to these families of methods have been explored for problems arising in computational finance, to our knowledge we are the first to consider their use in the context of fluid animation, or computer animation more broadly. Our results show that our proposed methods are both more scalable and more efficient compared with existing approaches.

2. Related Work

The PATH solver [FM00], which is a generalization of the classical Newton method and based on quadratic programming (QP), was used by Batty et al. [BBB07] to solve the LCP problem. However, they point out that it is not scalable to large problems. Narain et al. [NGL10] formulated a pressure equation for granular material simulation into an LCP. Gerszewski et al. [GB13] solved LCPs for pressure and density when animating large-scale splashing liquids. Both Narain et al. [NGL10] and Gerszewski et al. [GB13] used a QP solver called modified proportioning with reduced gradient projections (MPRGP) [DS05] to solve LCPs. MPRGP is an active set method based on preconditioned conjugate gradient (PCG) that interleaves conjugate gradient (CG) steps with expansion and proportioning steps that update the active set. Instead of solving LCPs to achieve the non-sticking effect for fluids, Inglis et al. [IEGT17] proposed a Primal-Dual method to split the problem into two components and solved them with CG and a classification scheme, respectively. Andersen et al. [ANE17a; Erl13] proposed a non-smooth Newton approach for the LCP problem in fluid animation in 2D, which has better convergence than projected Gauss–Seidel (PGS) type methods [Erl07; CA09; GZO10] and is faster than pivoting methods [Bar94; AV11], but their method requires solving a linear system and performing line searches on each Newton iteration. To solve the linear system, they adopted preconditioned conjugate gradient (PCG) but they point out that their overall solver may fail in some cases if the preconditioner is used. This limitation may reduce their solver's potential speed-up. Moreover, PCG itself is not scalable as the number of iterations is known to double when grid resolution is doubled along each dimension, i.e., when the width of each cell is halved. Andersen et al. extend their framework to 3D [ANE17b] and demonstrate convergence for a 100^3 grid. However, it is not clear how their method scales with larger grid sizes. Various multigrid schemes [MCPN08; MST10; FWD14] have been used for liquid simulation with standard solid boundary conditions, but these require solving only linear systems rather than LCPs. Chentanez & Muller [CM12] developed a multigrid method to solve the LCPs from fluid simulation. Their method requires only a few small changes to multigrid for linear systems [CM11]. However, they do not present any scaling tests to demonstrate whether it achieves mesh-independent convergence behavior, which is the major advantage of using a multigrid scheme. It is therefore unclear how their scheme scales with large problems.

3. LCP formulation from pressure equation

We now proceed to give a brief introduction to the pressure projection with separating boundaries and formulate it into an LCP. We denote the fluid velocity field as $\mathbf{u}$, time as t, pressure as p, density as ρ, and acceleration due to body forces, such as gravity, as $\mathbf{f}$. Since we are interested in scenarios in which viscosity is negligible, we omit viscous terms from the Navier-Stokes equations, and adopt the incompressible Euler equations [GDN97]:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \frac{1}{\rho}\nabla p = \mathbf{f} \\ \nabla\cdot\mathbf{u} = 0. \end{cases} \tag{3.1}$$

After splitting off advection and body forces as usual, the pressure projection PDE is given as follows [Bri08]:

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{1}{\rho}\nabla p = 0 \text{ such that } \nabla\cdot\mathbf{u} = 0. \tag{3.2}$$

This is the problem that we are interested in solving efficiently, since it can often be the most time consuming part of a simulation.

The discretization of the domain is illustrated in Figure 3.1. We adopt a standard staggered (MAC) grid layout with velocity normal components on cell faces and pressures at cell centers (black points). We assume there are N_C grid cells in the discretized domain and index them from 0.

Let $\mathbf{u}^n$ be the vector of fluid velocity values and scalar p^n be the vector of pressures at the n-th time step. We obtain an intermediate velocity field $\hat{\mathbf{u}}^n$ after applying advection and body forces. After discretizing the PDE (3.2) in time, we find:

$$\mathbf{u}^{n+1} = \hat{\mathbf{u}}^n - \frac{\Delta t}{\rho}\nabla p^n \text{ such that } \nabla\cdot\mathbf{u}^{n+1} = 0. \tag{3.3}$$

Substituting the first equation in (3.3) into the second gives a Poisson problem,

$$\nabla\cdot\mathbf{u}^{n+1} = -\frac{\Delta t}{\rho}\nabla^2 p^n + \nabla\cdot\hat{\mathbf{u}}^n = 0. \tag{3.4}$$

Henceforth we write $\mathbf{u} = \mathbf{u}^{n+1}$ and $p = p^n$ for convenience.

Next, consider the conditions to be applied at the boundary of the liquid domain. The free surface between liquid and air is modeled as a Dirichlet condition with $p = 0$. The standard solid boundary conditions, $\mathbf{u}\cdot\mathbf{n} = 0$, where $\mathbf{n}$ is the outward normal, cause the liquid to stick to the solid walls (see Figure 3.2). To allow the liquid to separate from the walls, we instead model the solid wall boundary conditions as follows:

$$0 \le p \perp \mathbf{u}\cdot\mathbf{n} \ge 0, \tag{3.5}$$

where the notation $\perp$ means $p \ge 0$ is complementary to $\mathbf{u}\cdot\mathbf{n} \ge 0$. That is, $p > 0$ when $\mathbf{u}\cdot\mathbf{n} = 0$ and $\mathbf{u}\cdot\mathbf{n} > 0$ when $p = 0$.

Integrating $\nabla\cdot u$ over an arbitrarily small control volume V gives

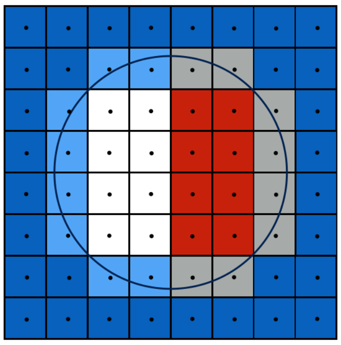


Figure 3.1: *Grid cells after discretization in 2D for a circular domain whose right half contains liquid. Red: cells in the liquid; Blue: cells in the solid; White: cells in the air; Grey: cells in both liquid and solid; Azure: cells in both solid and air.*

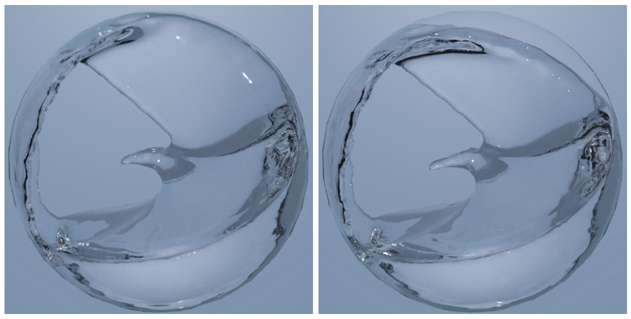


Figure 3.2: *A selected frame from two simulations of a 3D scenario of liquid splashing inside a spherical boundary. Left: Without separating solid wall boundary conditions, the liquid adheres to the top of the sphere. Right: With separating solid wall boundary conditions, the liquid separates naturally.*

$\int_V \nabla \cdot u = \int_S u \cdot n > 0$ where S is the surface of V. After enforcing the condition (3.5) in equation (3.4), the new pressure equation satisfying the separating solid wall boundary conditions becomes:

$$0 \leq p \perp -\frac{\Delta t}{\rho}\nabla^2 p + \nabla \cdot \hat{\mathbf{u}}^n \geq 0. \tag{3.6}$$

Derivation details can be found in the work of Andersen et al. [ANE17a]. The discretization of the pressure p gives an LCP,

$$0 \leq \mathbf{p} \perp \mathbf{A}\mathbf{p} + \mathbf{b} \geq 0, \tag{3.7}$$

where $\mathbf{A}$ comes from the discretization of the operator $-\frac{\Delta t}{\rho}\nabla^2$ and is a symmetric positive definite (SPD) matrix, $\mathbf{b}$ and $\mathbf{p}$ are vectors for values of $\nabla \cdot \hat{\mathbf{u}}$ and pressure p, respectively. Each of their entries corresponds to the value at the center of a grid cell. Separating solid boundary conditions (3.5) only need to be applied to boundary points. Let $\mathbf{p}_i$ be the pressure of the i-th grid cell in the discretization domain. If i corresponds to a cell near the solid boundary (grey), the separating wall boundary conditions are enforced and the i-th row of the discretized pressure equation (3.7) is

$$0 \leq \mathbf{p}_i \perp \sum_{j=0}^{N_C} \mathbf{A}_{i,j}\mathbf{p}_j + \mathbf{b}_i \geq 0, \tag{3.8}$$

which can also be written as an optimal control equation

$$\min_{\lambda_i \in \{0,1\}} \{\lambda_i(\sum_{j=0}^{N_C} \mathbf{A}_{i,j}\mathbf{p}_j + \mathbf{b}_i) + (1-\lambda_i)\mathbf{p}_i\} = 0, \tag{3.9}$$

where λ_i is the control that minimizes the term in the braces on the left hand side. The solution $\mathbf{p}_i$ and the control λ_i are unknown and depend on each other. If the i-th cell is a fully liquid cell (red), the i-th row of the discretized pressure equation simply becomes linear:

$$\sum_{j=0}^{N_C} \mathbf{A}_{i,j}\mathbf{p}_j + \mathbf{b}_i = 0. \tag{3.10}$$

Defining $\mathscr{S} = \{0 \leq i < N_C |$ The i-th cell lie in both liquid and solid$\}$ (grey cells in Figure 3.1), then the LCP problem (3.7) can be converted into the following mixed LCP (or MLCP):

$$\begin{cases} \min_{\lambda_i \in \{0,1\}} \{\lambda_i(\sum_{j=0}^{N_C} \mathbf{A}_{i,j}\mathbf{p}_j + \mathbf{b}_i) + (1-\lambda_i)\mathbf{p}_i\} = 0, \text{ if } i \in \mathscr{S} \\ \sum_{j=0}^{N_C} \mathbf{A}_{i,j}\mathbf{p}_j + \mathbf{b}_i = 0, \text{ otherwise.} \end{cases} \tag{3.11}$$

We define the set of possible control matrices as $\mathscr{M} = \{diag(\lambda_0, ..., \lambda_{N_C-1}) | \lambda_i = 0 \text{ or } 1 \text{ if } i \in \mathscr{S}; \lambda_i = 1 \text{ if } i \notin \mathscr{S}\}$. The nonlinear equation (3.11) can be written as an optimal control equation,

$$\inf_{\Lambda \in \mathscr{M}} \{\Lambda(\mathbf{A}\mathbf{p} + \mathbf{b}) + (I - \Lambda)\mathbf{p}\} = 0, \tag{3.12}$$

where I is the identity matrix, $\Lambda = diag(\{\lambda_i\})$ is a diagonal matrix which serves as the optimal control and inf is performed componentwise on the vector inside the braces.

4. Fast solvers

Solving the LCP equation (3.11) is challenging as the solution and the constraints are coupled with each other. Solving the LCP requires enforcing the constraints while the solution must be known in order to know where to enforce the constraints. The LCP problem is considered to be nonlinear, and therefore efficient solvers for linear systems cannot be used directly.

To solve the LCP fluid problem more efficiently, we propose three methods: policy iteration [FL07], a penalty method [FV02], and full approximation scheme (FAS) multigrid [HW13], which is a multigrid method for nonlinear equations. We choose policy iteration and penalty method because they are fast when the grid size is not very large. To deal with LCPs arising in larger scale fluid simulations, we prefer a multigrid method due to its scalability; that is, its iteration count is expected to be essentially independent of grid resolution. We will show that our proposed methods can outperform the naive multigrid approach proposed by Chentanez et al. [CM12] and the non-smooth Newton method proposed by Andersen et al. [ANE17a].

4.1. Policy iteration

The basic idea of policy iteration for solving a nonlinear problem is to solve one of the unknowns by fixing the other, and iteratively repeat the process until convergence. Compared with Newton's method, the linearization process is simpler and less expensive. Our proposed policy iteration scheme first computes the control Λ using an initial guess of $\mathbf{p}$, and uses the control to linearize the problem. We then solve the linear system to obtain an approximate solution to $\mathbf{p}$ and use it to once again compute a new control. We repeat this process until the residual is sufficiently small. Our policy iteration scheme is presented in Algorithm 1. The policy update in equation (4.1) amounts to explicitly enforcing the LCP constraints, by choosing for each row a λ_i that produces the minimum value. The i-th row becomes trivial if λ_i is chosen as 0. The method is remarkably simple, but we shall see that it is quite effective in practice. We calculate the initial guess using the policy (4.1) from the previous time step since this exhibited the best performance in practice.

Algorithm 1 Policy iteration for solving the LCP problem (3.11)

```
1: Choose an initial guess for $\mathbf{p}^0$ and a tolerance $\varepsilon$.
2: for $k = 0, 1, 2, ...$ until residual $< \varepsilon$ do
3:     Find the optimal control matrix $\Lambda^*$ such that
           $\Lambda^* = \arg\inf_{\Lambda \in \mathscr{M}} \{\Lambda(\mathbf{A}\mathbf{p}^k + \mathbf{b}) + (I - \Lambda)\mathbf{p}^k\}.$   (4.1)
4:     Solve the linear system for $\mathbf{p}^{k+1}$,
           $(\Lambda^*\mathbf{A}\Lambda^* + I - \Lambda^*)\mathbf{p}^{k+1} = -\Lambda^*\mathbf{b}.$   (4.2)
5: end for
6: The approximate solution is given by $\mathbf{p}^{k+1}$ after the residual reaches the tolerance $\varepsilon$.
```

We remark that the matrix $\mathbf{A}$ obtained from the pressure equation has positive diagonal entries and non-positive off-diagonal entries, and is strictly diagonally dominant; hence it is an M-matrix

[Var09]. The matrix in (4.2) essentially replaces some rows of $\mathbf{A}$ by putting one on the diagonal since λ_i is either 0 or 1. Thus the matrix in (4.2) is also an M-matrix. Theorem 6.2 [FL07] showed that the M-matrix property ensures the sequence of approximate solution is monotone and hence our policy iteration is guaranteed to converge to an unique solution.

We will consider two possible iterative methods to solve the inner linear system (4.2) in Algorithm 1. One is to use preconditioned conjugate gradient (PCG) [Eis81] with Incomplete LU factorization (ILU) as the preconditioner. Since the matrix is an M-matrix, it is guaranteed to converge. However, the rate of convergence may depend on the mesh size. Another attractive option is to use multigrid. Multigrid is known for mesh-independent convergence, although special care needs to be taken to capture the irregular geometry and boundary conditions. Note that the multigrid scheme used here is a basic *linear* multigrid method (e.g., [MST10]), distinct from the nonlinear variant we develop in Section 4.3.

4.2. Penalty method

In contrast to policy iteration, our proposed penalty method approach solves the LCP by enforcing the constraints implicitly through large penalties. As a result, formulating the linear system in our penalty method is faster than policy iteration, while the linear system in policy iteration has a smaller size. Considering their tradeoffs, we propose to apply both methods to solve the fluid LCP and explore how they perform.

The idea of penalty methods is to use a large positive penalty term ρ to penalize the violation of the constraint $\mathbf{p} \geq 0$ as follows:

$$\mathbf{Ap} + \mathbf{b} = \rho \max(-\mathbf{p}, 0). \tag{4.3}$$

Note that the penalty only applies to rows corresponding to cells on the fluid-solid boundary. Assuming that $\mathbf{p}$ satisfies the penalized nonlinear equation (4.3), then $\mathbf{Ap}+\mathbf{b} \geq 0$ always holds. If $\mathbf{p}_i \geq 0$, we have $(\mathbf{Ap}+\mathbf{b})_i = 0$. If $\mathbf{p}_i < 0$, $\mathbf{p}_i = -\frac{1}{\rho}(\mathbf{Ap}+\mathbf{b})_i$ is expected to be very small due to the large penalty parameter ρ (10^9 for all our experiments). Given an initial guess, a penalty method formulates a linear system by comparing each entry with the constraint and adding a large penalty to the corresponding diagonal of the matrix $\mathbf{A}$ when there is a constraint violation. Solving the linear system gives a new approximate solution, which is used to formulate a new linear system. This process is repeated until convergence.

Let the diagonal matrix $\Pi(\mathbf{p})$ be defined as:

$$\Pi(\mathbf{p})_{i,i} = \begin{cases} 1, \text{ if } i \in \mathscr{S} \text{ and } \mathbf{p}_i < 0 \\ \\ 0, \text{ otherwise.} \end{cases} \tag{4.4}$$

Our penalty method for the nonlinear problem (3.11) is given by Algorithm 2.

Since $\mathbf{A}$ is an M-matrix as mentioned in Section 4.1, the matrix in (4.5), which is constructed by adding some positive values to the diagonals of $\mathbf{A}$, is also an M-matrix. Therefore, the convergence of penalty iteration to a unique solution is guaranteed [FV02]. As we did for policy iteration, we also propose to solve the inner linear systems using incomplete LU-preconditioned CG [Eis81] or (linear) multigrid. Similar to policy iteration, the initial guess is calculated using the penalty matrix (4.4) from the previous time step to improve the solver's performance.

Algorithm 2 Penalty method for solving the LCP problem (3.11)

```
1: Choose an initial guess for $\mathbf{p}^0$, a tolerance $\varepsilon$, and the discount $\rho \gg \varepsilon$.
2: for $k = 0, 1, 2, \ldots$ until residual $< \varepsilon$ do
3:     Solve the linear system for $\mathbf{p}^{k+1}$,
           $(\mathbf{A} + \rho\Pi(\mathbf{p}^k))\mathbf{p}^{k+1} = -\mathbf{b},$        (4.5)
4: end for
5: The approximate solution is given by $\mathbf{p}^{k+1}$ after the residual reaches the tolerance $\varepsilon$.
```

4.3. Multigrid

Both our policy iteration and penalty method have nested iterations: the outer iteration for updating a linear system and the inner iteration for solving the linear system. Since this can become computationally expensive, especially for larger problems, in this section we propose an efficient multigrid method for the full LCP problem. While Chentanez et al. [CM12] previously proposed a multigrid scheme to treat the fluid LCP, their method is based on standard multigrid for *linear* problems. It is therefore expected to achieve sub-optimal performance and scaling behavior on our nonlinear problem. Instead, we propose to use the full approximation scheme (FAS), which is a multigrid framework designed specifically for nonlinear problems.

Algorithm 3 V-Cycle of the FAS multigrid for solving the LCP problem (3.11)

```
1: $\mathbf{p}^h \leftarrow$ V-Cycle($\mathbf{p}^h, \mathbf{f}^h$):
2: Define the restriction operator $R$ and interpolation operator $P$.
3: if $h$ is the coarsest level then
4:     Solve $\mathcal{N}^h(\mathbf{p}^h) = \mathbf{f}^h$ with PGS.
5: else
6:     Pre-smooth $\mathbf{p}^h$ with PGS.
7:     Compute the residual $\mathbf{r}^h = \mathbf{f}^h - \mathcal{N}^h(\mathbf{p}^h)$.
8:     Restrict $\mathbf{p}^h$: $\mathbf{p}^H = R(\mathbf{p}^h)$, where $H$ is the next coarser level.
9:     Compute $\mathbf{f}^H = \mathcal{N}^H(\mathbf{p}^H) + R(\mathbf{r}^h)$.
10:    $\mathbf{p}^H \leftarrow$ V-Cycle($\mathbf{p}^H, \mathbf{f}^H$)
11:    Interpolate and correct: $\mathbf{p}^h \leftarrow \mathbf{p}^h + P(\mathbf{p}^H - R(\mathbf{p}^h))$.
12:    Post-smooth $\mathbf{p}^h$ with PGS.
13: end if
14: The solution is obtained by iterating V-Cycle($\mathbf{p}, 0$).
```

In this section, we will introduce our multigrid method and describe how it can be applied to solve the LCP (3.11). But first, we will explain the idea of multigrid for solving the pressure equation with the LCP condition. Multigrid has two main components: smoothing, which removes high frequency errors on a fine grid to make the error smooth; and coarse grid correction, which removes the low frequency errors on a coarser grid. It proceeds from the fine grid to the coarse grid and back to correct the fine grid error. This process is called a V-cycle and can be extended to multiple grid levels by applying it recursively. The multigrid method iterates the V-cycle until convergence.

Since the problem we are solving is nonlinear, the multigrid V-cycle for linear problems cannot be used to solve (3.11) directly. To address this issue, we apply the Full Approximation Scheme (FAS) multigrid algorithm [Bra77]. We use $\mathcal{N}$ to define the operator on $\mathbf{p}$ on the left hand side of (3.12) and the equation becomes $\mathcal{N}(\mathbf{p}) = 0$. Let $\mathcal{N}^l$ denote the nonlinear operator $\mathcal{N}$ on the grid at level l. According to (3.12), we have

$$\mathcal{N}^l(\mathbf{p}^l) = \inf_{\Lambda^l \in \mathscr{S}^l} \{\Lambda(\mathbf{A}^l\mathbf{p}^l + \mathbf{b}^l) + (I - \Lambda^l)\mathbf{p}^l\}, \tag{4.6}$$

where the matrix $\mathbf{A}^l$ is constructed at grid level l, vector $\mathbf{p}^l$ is the approximate solution at the l-th level, vector $\mathbf{b}^l$ is restricted from the finer level, and the control set $\mathscr{S}^l$ is defined to be the same as in $\mathscr{S}$ except that the dimension of the matrices is the number of cells on the l-th level. To explain how FAS multigrid works, we let $l = h$ be the fine grid level and $l = H$ be the next coarser grid level. The nonlinear problem at level h is $\mathcal{N}^h(\mathbf{p}^h) = \mathbf{f}^h$, where $\mathbf{f}^h = \mathbf{0}$ and the problem becomes (3.12) if h is the finest grid. The error after pre-smoothing is $\mathbf{e}^h = \mathbf{q}^h - \mathbf{p}^h$ where $\mathbf{q}^h$ is the exact solution. The residual becomes

$$\mathbf{r}^h = \mathbf{f}^h - \mathcal{N}^h(\mathbf{p}^h) = \mathcal{N}^h(\mathbf{q}^h) - \mathcal{N}^h(\mathbf{p}^h). \tag{4.7}$$

Due to the nonlinear operator $\mathcal{N}^h$, we cannot simply find $\mathbf{r}^h = \mathcal{N}(\mathbf{e}^h)$ and restrict it into coarse grid as in the linear case. Instead, we rewrite equation (4.7) as

$$\mathcal{N}^h(\mathbf{q}^h) = \mathcal{N}^h(\mathbf{p}^h) + \mathbf{r}^h \tag{4.8}$$

and restrict it into the coarse level H:

$$\mathcal{N}^H(\mathbf{q}^H) = \mathcal{N}^H(R(\mathbf{p}^h)) + R(\mathbf{r}^h) = \mathbf{f}^H, \tag{4.9}$$

where R is the restriction operator from fine to coarse grid and $\mathbf{q}^H$ is the solution on the coarse grid. Let $\mathbf{p}^H$ be the approximate solution to equation (4.9). The coarse grid error is computed as $\mathbf{e}^H = \mathbf{p}^H - R(\mathbf{p}^h)$. We obtain the fine grid error $\mathbf{e}^h = P(\mathbf{e}^H)$ through interpolation and use it to correct $\mathbf{p}^h$. Algorithm 3 provides the details of FAS multigrid. We use projected Gauss-Seidel (PGS) [CPS92] as the smoother, which is an iterative nonlinear solver based on the Gauss-Seidel method. Besides using FAS, we develop three modifications for our multigrid, which enables fast convergence for the LCP and further distinguishes our method from the multigrid of Chentanez et al. [CM12]: interpolation and restriction operators; boundary handling; and coarse grid matrix construction.

Interpolation and restriction. We consider formulating the restriction matrix R and the interpolation matrix P. To preserve symmetry, the restriction matrix is computed as the transpose of the interpolation matrix scaled by a constant ($\frac{1}{4}$ in 2D and $\frac{1}{8}$ for 3D). In order to achieve good convergence [TOS00], the sum of the orders of interpolation and restriction order should be greater than the order of the differential operator, which is 2 in (3.6). A natural first choice for interpolation is bilinear in 2D and trilinear in 3D, since they are both second order.

Because we have adopted a staggered grid with pressures at cell

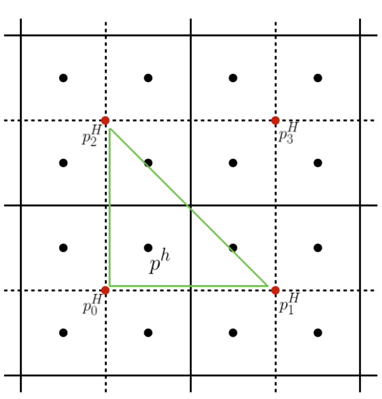


Figure 4.1: *Interpolation of pressure between grid levels in 2D.*

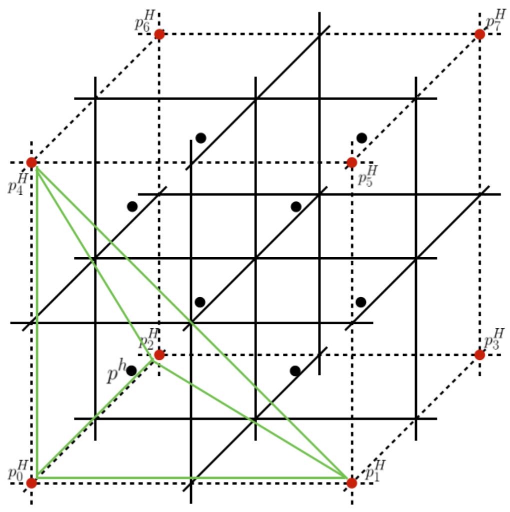


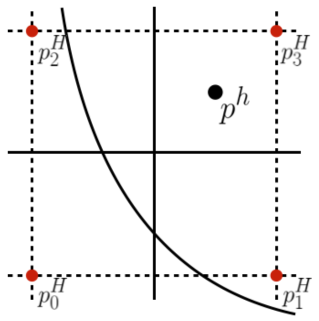
Figure 4.2: *Interpolation of pressure between grid levels in 3D.*

centers, we must take care in interpolating between levels. The layout of two grid levels and their pressure samples is shown in 2D in Figure 4.1. The solid line represents the coarse grid and the dotted line the fine grid. Fine and coarse grid pressure samples are represented by black and red points, respectively. Bilinear interpolation for the fine grid point p^h relies on the four nearest coarse grid points p_0^H, p_1^H, p_2^H, p_3^H. Their interpolation weights are $\frac{9}{16}$, $\frac{3}{16}$, $\frac{3}{16}$, $\frac{1}{16}$, respectively. Trilinear interpolation in 3D is similar: p^h is interpolated from eight surrounding coarse grid points with weights of $\frac{27}{64}$ for p_0^H, $\frac{9}{64}$ for p_1^H, p_2^H, p_4^H, $\frac{3}{64}$ for p_3^H, p_5^H, p_6^H, and $\frac{1}{64}$ for p_7^H.

Unfortunately, using too many nearby coarse grid points yields denser interpolation matrices and makes the computation more expensive. We therefore propose an even simpler alternate solution that uses fewer coarse grid points while preserving the desired second order accuracy. Specifically, we adopt barycentric interpolation. In 2D, we use p_0^H, p_1^H, p_2^H, and the fine grid point is inside the triangle (green lines in Figure 4.1) formed by these coarse grid points. The interpolation becomes $p^h = \frac{1}{2}p_0^H + \frac{1}{4}p_1^H + \frac{1}{4}p_2^H$. In 3D, p^h is inside the tetrahedron (green lines in Figure 4.2) formed by p_0^H, p_1^H, p_2^H, p_4^H. Each coarse grid point is assigned a weight of $\frac{1}{4}$.

These more compact interpolation (and corresponding restriction) operators give interpolation matrices that will be 25 percent sparser in 2D and 50 percent sparser in 3D.

Boundary handling. The solution to the pressure equation (3.11) may have a large jump around the solid boundary due to different conditions being imposed on either side of it (see Figure 4.3). Therefore, we modify our interpolants to accommodate this situation and ensure better convergence of FAS multigrid. Loosely inspired by the work of Wan & Liu [WL04] and Guendelman et al. [GSLF05], we modify our interpolants to exploit knowledge of the solid boundary. Given a fine grid point, we determine the interpolation weights for only the coarse grid points which are on the same side of the solid. We then rescale the weights of those (same side) coarse grid points proportionally so that they sum to 1. For example, consider a 2D coarse grid cell crossing the boundary as shown inset. Points p_0^H and p_2^H are on the opposite side of the solid from the query point p^h. Barycentric interpolation ordinarily assigns weights of $\frac{1}{4}$, $\frac{1}{4}$, $\frac{1}{2}$ to p_1^H, p_2^H, p_3^H, respectively. In our modified one-sided interpolation, only points p_1^H and p_3^H will be used; their weights after rescaling become $\frac{1}{3}$ and $\frac{2}{3}$, so we have $p^H = \frac{1}{3}p_1^H + \frac{2}{3}p_3^H$.



For the simpler linear multigrid used for the inner linear systems in in our policy iteration and penalty method, we likewise use barycentric interpolation/restriction and neglect points across the solid boundary, but found it unnecessary to normalize the weights.

Coarse grid matrix construction. The coarse grid matrix $\mathbf{A}^H$ is often constructed by directly discretizing the problem on the coarse grid. Chentanez et al. [CM12] computed the coarse grid matrix by averaging the the weights of non-solid matter and the liquid level set function from fine to coarse grids. We found that this approach might be insufficiently accurate as it led to the coarse grid error not always matching the fine grid error, in turn causing slow convergence. In general, the complexity of the fluid domain can make

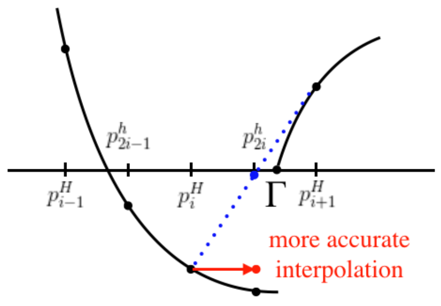


Figure 4.3: *Naive interpolation (dashed blue) across a narrow solid boundary causes large pressure errors (shown in 1D). Our one-sided interpolation (red) yields better behavior.*

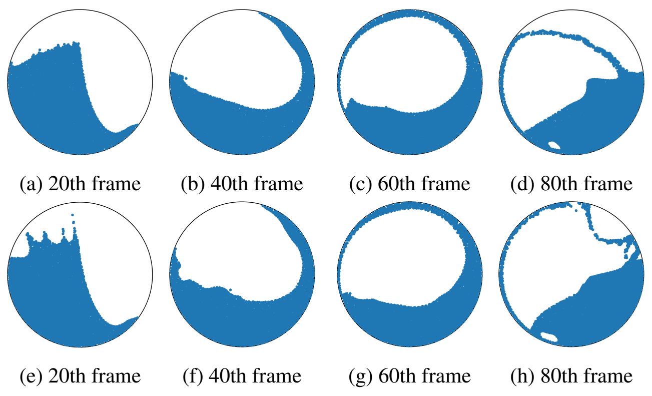


Figure 5.1: *Snapshots from simulating liquid inside a solid circle in 2D using LCP boundaries (top row) and standard boundaries (bottom row).*

Size	FAS-MG	PI-PCG	PI-MG	PE-PCG	PE-MG	FMG	Newton	No LCP
32	7.45	16.9	7.59	16.87	7.99	20.38	92.11	15.99
64	10.39	36.02	9.74	36.07	12.58	35.4	336.17	30.73
128	14.07	70.63	11.89	70.58	17.6	65.37	947.23	55.91
256	18.26	123.61	14.51	122.42	21.98	109.54	2283.65	93.2

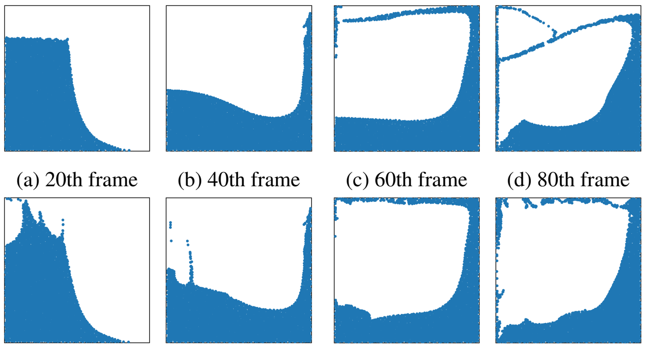
Table 1: *Average number of iterations per pressure equation for solving 100 frames of the circular domain problem in 2D.*

it difficult to determine how to compute a proper coarse grid matrix through direct discretization. Therefore, we propose to construct the coarse grid matrix using the Galerkin method [BM*00]: $\mathbf{A}^H = R \cdot \mathbf{A}^h \cdot P$, where $\mathbf{A}^h$ is the fine grid matrix. This approach ensures that the coarse grid operator is more faithful to its fine grid counterpart, and gives faster convergence. In particular, this ensures that relatively thin boundaries which are only fully resolved at the finest level are still naturally respected at coarser grid levels, without additional treatment. Using the compact interpolation in 3D, the coarse grid matrices will be much sparser and faster to construct since the interpolation matrices are 50 percent sparser. The smoothing process will be much less expensive due to the decrease of nonzeros per row. In our experiments in 3D, both the average number of nonzeros per row of the top level coarse grid matrix and the time for solving the LCP were reduced by about half.

5. Numerical results

To demonstrate that we achieve the desired effect of using the LCP form of pressure projection, we perform tests on several scenarios with and without enforcing separating solid boundary conditions. We use an absolute residual tolerance of 10^{-6} throughout. To ensure convergence of the penalty method, the penalty term ρ must be sufficiently large, which depends on the desired tolerance. In our experiments, we chose ρ to be 10^3/tolerance (i.e., 10^9). Figures 5.1 and 5.2 show frames from two scenarios in 2D which demonstrate the difference between standard and LCP solid boundary conditions. We also present selected frames from a scenario in 3D in Figures 5.3 and 5.4. The liquid more readily separates from the wall in the LCP case.

To evaluate the effectiveness of our proposed solvers, we com-



(a) 20th frame (b) 40th frame (c) 60th frame (d) 80th frame

(e) 20th frame (f) 40th frame (g) 60th frame (h) 80th frame

Figure 5.2: *Snapshots from simulating liquid inside a solid square in 2D using LCP boundaries (top row) and standard boundaries (bottom row).*

Size	FAS-MG	PI-PCG	PI-MG	PE-PCG	PE-MG	FMG	Newton	No LCP
32	9.75	29.9	12.12	30.16	13.93	22.21	1901.69	20.14
64	10.44	70.92	14.66	71.34	19.11	39.81	3890.08	39.57
128	11.94	132.56	17.61	133.53	24.57	61.12	10879.01	62.26
256	14.02	251.88	20.71	252.28	28.84	89.89	NA	100.78

Table 2: *Average number of iterations per pressure equation for solving 100 frames of the spherical domain problem in 3D.*

Size	PI-PCG	PI-MG	PE-PCG	PE-MG	Newton
32	1.12	1.12	1.12	1.23	0.94
64	1.25	1.25	1.25	1.43	1.71
128	1.34	1.34	1.35	1.58	2.08
256	1.43	1.43	1.41	1.63	2.34

Table 3: *Average number of* outer *iterations per pressure equation for policy iteration and penalty method for solving 100 frames of the circular domain problem in 2D.*

Size	PI-PCG	PI-MG	PE-PCG	PE-MG	Newton
32	1.47	1.47	1.49	1.53	2.04
64	1.78	1.76	1.79	1.77	2.57
128	2.1	2.12	2.12	2.19	2.91
256	2.48	2.52	2.48	2.49	NA

Table 4: *Average number of* outer *iterations per pressure equation for policy iteration and penalty method for solving 100 frames of the spherical domain problem in 3D.*

Size	FAS-MG	PI-PCG	PI-MG	PE-PCG	PE-MG	FMG	Newton	No LCP
32	0.0021	0.00054	0.0028	0.00055	0.0025	0.0026	0.002	0.00048
64	0.0082	0.0034	0.0086	0.0033	0.0075	0.022	0.024	0.0029
128	0.029	0.02	0.03	0.021	0.028	0.17	0.24	0.016
256	0.13	0.15	0.12	0.13	0.13	1.25	2.59	0.098

Table 5: *Average time (in seconds) per pressure equation for solving 100 frames of the circular domain problem in 2D.*

Size	FAS-MG	PI-PCG	PI-MG	PE-PCG	PE-MG	FMG	Newton	No LCP
32	0.037	0.014	0.036	0.015	0.036	0.076	0.98	0.0074
64	0.25	0.2	0.33	0.2	0.37	1.68	25.38	0.09
128	2.15	2.64	2.82	2.82	4.28	30.24	637.67	1.15
256	16.78	36.96	24.58	37.82	47.38	510.71	NA	14.24

Table 6: *Average time (in seconds) per pressure equation for solving 100 frames of the spherical domain problem in 3D.*

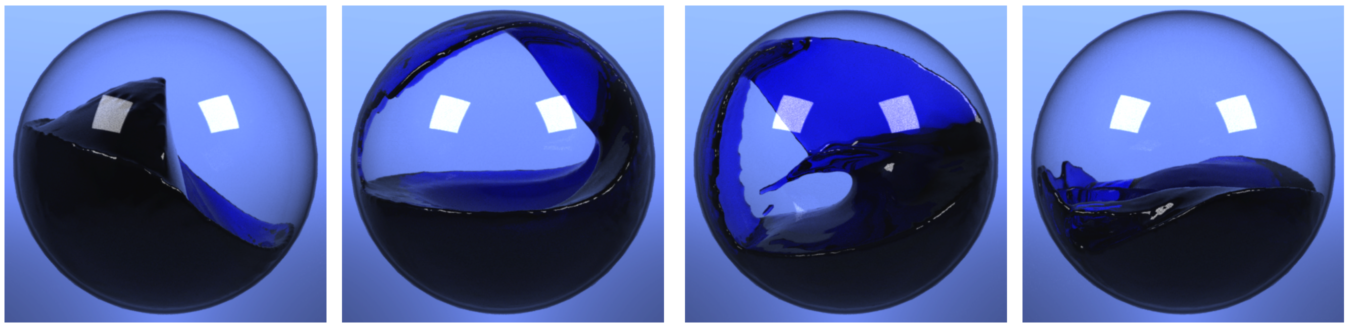
Size	FAS-MG	PI-PCG	PI-MG	PE-PCG	PE-MG	FMG	Newton
32	4.41	9.74	8.11	9.5	7.67	6.89	87.2
64	7.07	15.52	10.55	15.75	10.11	10.71	195.5
128	5.83	31.29	13.52	31.9	12.91	34.13	450.21

Table 7: *Comparison of the number of iterations between our solvers and Newton's method for solving the pressure equations for 100 frames of the maze problem in 2D.*

pare the performance of different solvers, including the standard boundary conditions using PCG ("No LCP"), on CPU in both 2D (Table 1) and 3D (Table 2). For the 256^3 grid in 3D, we did not test Newton's method because it was too slow. The initial scenario in 2D is a circular solid boundary with liquid filling the left half of the circle. In 3D, it is a spherical boundary with its left half filled with liquid. We compare our methods with the multigrid method developed by Chentanez et al. [CM12] and the non-smooth Newton's method [ANE17a]. Chentanez's full multigrid (FMG) approach iterates costly full cycles while our multigrid uses V-cycles, which are simpler and less expensive. Moreover, they pre- and post-smooth the error four times, while we do it only twice. For policy iteration, we tested with two different solvers for the linear system (4.2): PCG and (linear) multigrid, denoted by PI-PCG and PI-MG, respectively. The coarse grid matrix here was also computed using the Galerkin method. We likewise tested penalty method using PCG (PE-PCG) and multigrid (PE-MG). Newton's method is broadly similar in concept to policy iteration, but is much more complicated. In addition to solving a linear system in each Newton iteration, it also requires performing a line search. For the purpose of testing and comparing with Newton's method, we use their public code [Erl11] and ran tests using the CPU (rather than GPU).

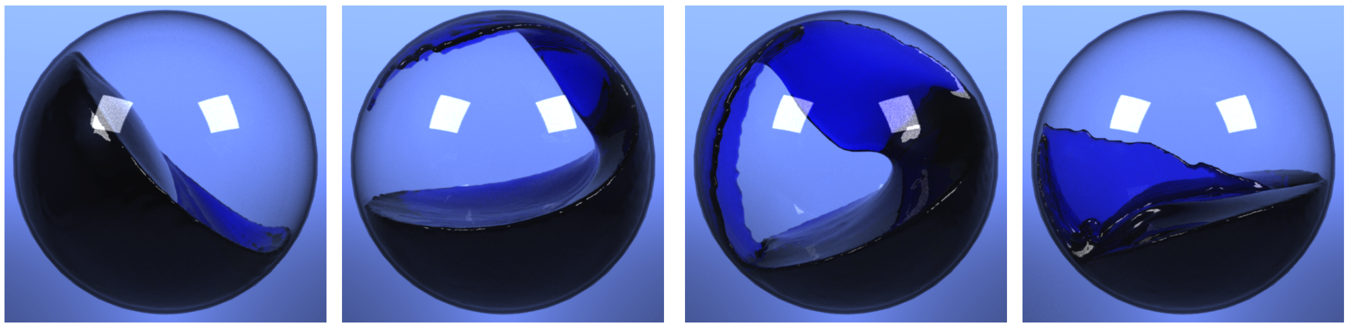
We measured the performance using the average number of iterations per pressure equation. Specifically, this refers to the average number of V-cycles per pressure equation over 100 frames for FAS-MG, and average number of full cycles for FMG. The solution processes of policy iteration, penalty method and Newton's method involve nested iterations, often called outer-inner iteration. The outer iteration updates the linear system whereas the inner iteration solves the linear system by an iterative method. We count all the inner iterations per pressure equation. We measure iterations for policy iteration and penalty method as the total number of PCG/MG iterations per pressure equation. For Newton's method, we add the number of PCG and line search iterations together to count as the number of iterations.

Regarding scalability in terms of number of iterations, our FAS-MG and PI-MG are scalable because the number of iterations increases slowly with increasing problem size. Our PI-MG is scalable due to the use of multigrid method and the good scalability of the number of outer iterations of policy iteration. Tables 3 and



(a) 20th frame (b) 50th frame (c) 70th frame (d) 90th frame

Figure 5.3: *Snapshots from simulating liquid inside a solid sphere in 3D at* 128^3 *using LCP boundaries.*



(a) 20th frame (b) 50th frame (c) 70th frame (d) 90th frame

Figure 5.4: *Snapshots from simulating liquid inside a solid sphere in 3D at* 128^3 *using standard boundaries.*

Size	FAS-MG	PI-PCG	PI-MG	PE-PCG	PE-MG	FMG	Newton
32	0.016	0.0042	0.032	0.004	0.029	0.017	0.41
64	0.092	0.018	0.15	0.018	0.12	0.1	3.5
128	0.38	0.1	0.7	0.11	0.53	1.21	31.35

Table 8: *Comparison of the average time (in seconds) between our solvers and Newton's method for solving the pressure equations for 100 frames of the maze problem in 2D.*

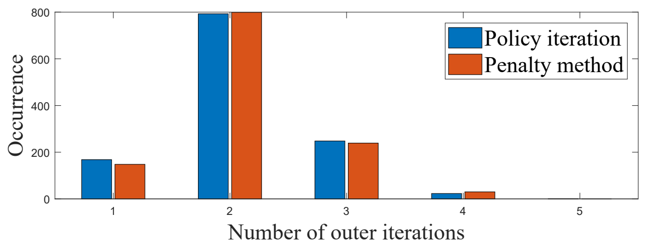
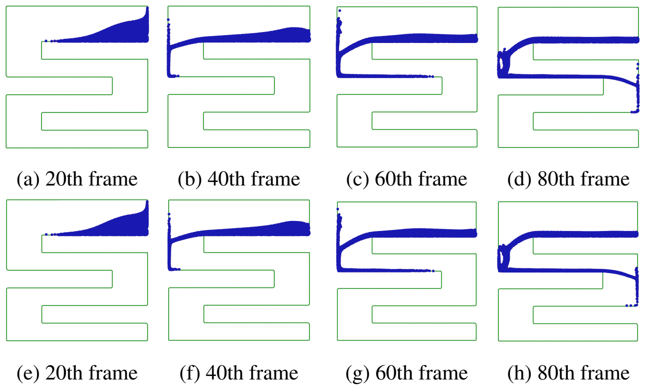


Figure 5.5: *A histogram illustrating how many pressure solves required a given number of outer iterations for policy iteration and penalty method for 100 frames of the spherical domain problem in 3D for grid size 128. About 93 percent of these pressure equations are LCP problems while some of them need only one outer iteration because the first control update leads to the correct linear system to solve the LCP. (Since we use substepping, the number of problems solved exceeds the frame count.)*



(a) 20th frame (b) 40th frame (c) 60th frame (d) 80th frame

(e) 20th frame (f) 40th frame (g) 60th frame (h) 80th frame

Figure 5.6: *Snapshots from simulating liquid inside a maze in 2D using our FAS-MG (top row) vs. the non-smooth Newton's method (bottom row). The results are visually consistent.*

4 show that the number of outer iterations for policy iteration and the penalty method is almost constant with only 2 to 3 linear systems needed per pressure equation. We observed that the pressure equations for most timesteps required solving the LCP equations (i.e., performing multiple outer iterations) when the problem size becomes large. We present a histogram of the number of outer iterations for policy iteration and penalty method over 100 frames

in the 3D test for the size of 128 in Figure 5.5. Policy iteration performs slightly better than the penalty method in terms of average number of outer iterations. Moreover, our policy iteration and penalty method scale slightly better than Newton's method and require fewer outer iterations in general. The scalability of both inner and outer iterations contributes to the scalability of PI-MG. However, for PI-PCG and PE-PCG, the number of iterations doubles as the problem size doubles. This is expected as the number of PCG iterations usually doubles with problem size, while the number of outer iterations remains relatively constant. Newton's method is comparatively slow, especially in 3D, because its inner iteration is computationally expensive although its outer iteration is relatively scalable. FMG also suffers from an increase in the number of iterations for full cycle although the rate is less than 2 (about 1.5). We note that the number of iterations required to converge for FMG is much larger than for our proposed multigrid methods (FAS-MG, PI-MG, and PE-MG) because it is not designed for nonlinear problems. FAS-MG and PI-MG required the smallest number of iterations among all the methods.

Next, we briefly discuss the time complexity per iteration for each method. For the multigrid methods (namely FAS-MG, FMG, and our basic MG for linear systems) each smoothing step takes about the same time on the finest grid. Each PCG iteration has about the same time complexity as one smoothing step on the finest grid in multigrid. Due to our Galerkin construction of the coarse grid matrix, our multigrid methods take more time on the coarse grid for smoothing compared to a direct discretization approach, since the coarse grid matrices have more nonzeros per row. However, this deficiency is outweighed by the good scalability of our resulting method. The size of the linear systems in each outer iteration of policy iteration, penalty method, and Newton's method are about the same. However, our policy iteration and penalty method do not need to perform line searches. Updating the policy (in PI) and adding penalty terms (in PE) are both relatively cheap.

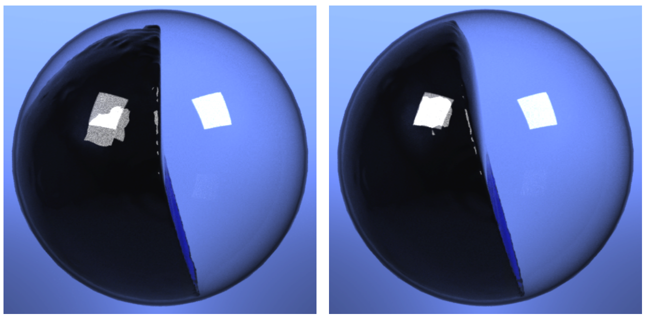


Figure 5.7: *The 10th frame of scenario 1 in 3D, with (left) and without (right) separating solid wall boundary conditions.*

Tables 5 and 6 show the average time per pressure equation for all methods in 2D and 3D, respectively. As an additional comparison point, we included the No-LCP solver, which uses only a single PCG linear solve. Our proposed methods are much faster than FMG and Newton's method in both 2D and 3D. Our fastest method, FAS-MG, is as much as 30 times faster than FMG in 3D at 256^3, and it is only slightly slower than the No-LCP solver. Our methods are also more scalable than FMG and Newton's method in terms of timing.

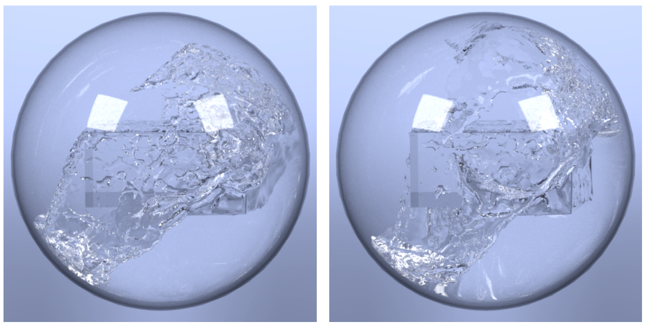


Figure 5.8: *The 70th frame from scenario 2 in 3D, with (left) and without (right) separating solid wall boundary conditions.*

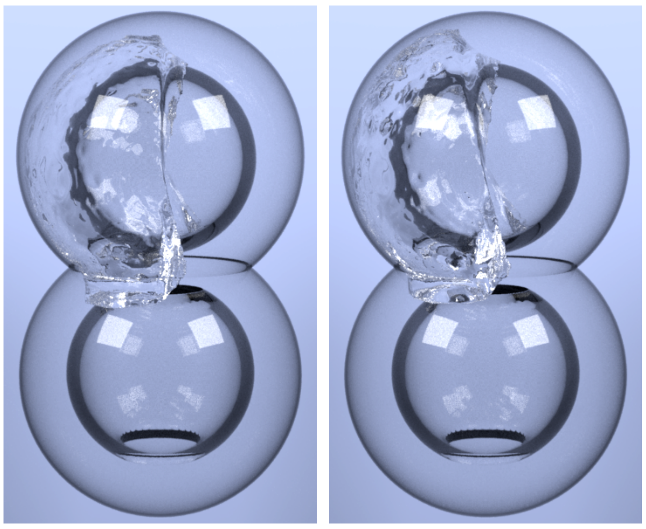


Figure 5.9: *The 10th frame from scenario 3 in 3D, with (left) and without (right) separating solid wall boundary conditions.*

When the problem size is doubled, FAS-MG, PI-MG, and PE-MG take about 5 times longer in 2D and 10 times longer in 3D. PI-PCG and PE-PCG's average time increases by about 7 times in 2D and 14 times in 3D when the problem sizes doubles. However, the average time for FMG increases by about 7 times in 2D and 17 times in 3D when the problem size doubles. Newton's method fares the worst: about 10 times for 2D and 25 times for 3D.

As an additional comparison against Newton's method [ANE17a], we used the authors' code and tested their method and all of our solvers in their maze scenario. The numerical results are shown in Tables 7 and 8 along with frame comparisons in Figure 5.6.

To demonstrate the convergence behavior of our solvers, we look into specific frames with significant handling of liquid solid separation in three different scenarios in 3D (see Figures 5.7, 5.8, 5.9, respectively.) Scenario 1 has already been introduced at the beginning of this section (see Figures 5.3 and 5.4). Scenario 2 is based

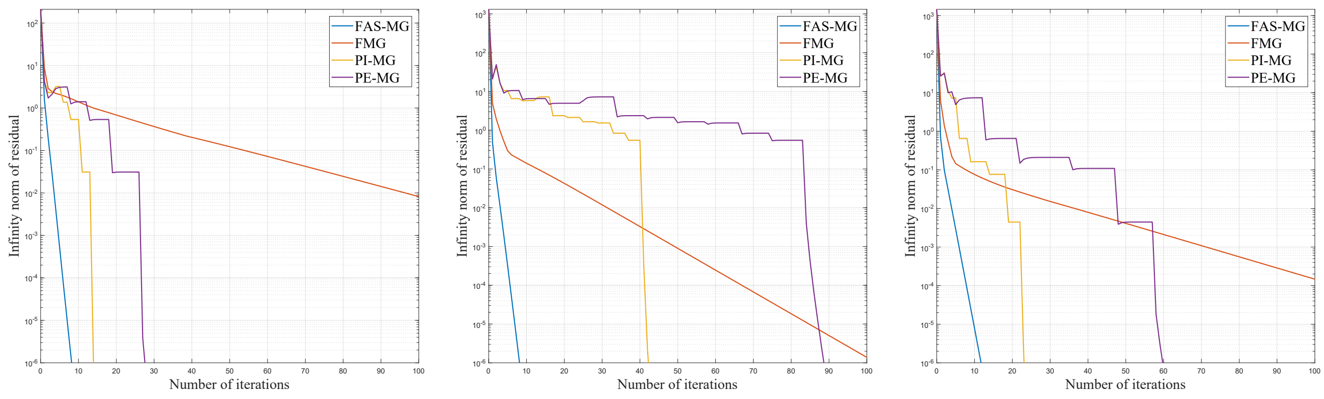


Figure 5.10: *Convergence plots for our methods vs. FMG on a grid of size 256 for scenarios 1 through 3, from left to right.*

Size	FAS-MG	PI-PCG	PI-MG	PE-PCG	PE-MG	FMG	Newton
32	7	139	10	146	21	29	2773
64	8	282	9	301	18	61	5837
128	8	593	10	657	24	128	18680
256	9	1263	14	1255	28	234	38547

Table 9: *Number of iterations for solving the pressure at the 10th frame of scenario 1 in 3D.*

on scenario 1 but has a rectangular cuboid inside so that fluid-solid contacts frequently form and break on the cuboid's bottom face. Scenario 3 has a solid outer boundary created as the union of two spheres. Two hollow solid spheres are also placed inside the domain with the bottom one having holes through it. The number of iterations for solving the pressure for the first substep of a specific frame in the three different scenarios are presented in Tables 9, 10, 11, respectively. For multigrid methods (namely FAS-MG, PI-MG, PE-MG and FMG), we plot their convergence rates for the three different scenarios in Figure 5.10. The tests in scenarios 2 and 3 demonstrate that the FAS-MG is still scalable even for complicated scenarios. The scalability of PI-MG and PE-MG deteriorates a little bit, but they still perform better than the existing methods (FMG and Newton).

Size	FAS-MG	PI-PCG	PI-MG	PE-PCG	PE-MG	FMG	Newton
32	6	149	11	161	18	35	2634
64	7	308	18	336	29	43	11122
128	8	503	22	548	40	63	50162
256	9	1186	43	1313	89	103	95823

Table 10: *Number of iterations for solving the pressure at the 70th frame of scenario 2 in 3D.*

Size	FAS-MG	PI-PCG	PI-MG	PE-PCG	PE-MG	FMG	Newton
32	7	80	14	85	23	15	1430
64	15	182	37	203	57	43	9219
128	9	421	34	446	90	67	46491
256	12	762	24	887	60	175	122080

Table 11: *Number of iterations for solving the pressure at the 10th frame of scenario 3 in 3D.*

Finally, we show how each of our proposed modifications (interpolation and restriction, boundary handling, and coarse grid matrix construction) on FAS-MG contribute to the success of our FAS multigrid solver. We replace each of them with the simpler or standard option, namely, standard trilinear, piecewise constant (PWC) interpolation and restriction, no specialized boundary handling, and direct discretization of coarse levels, respectively, and present the tests (Table 12) on the pressure equation for the first substep at the 10th frame of scenario 1 in 3D. Our method with the barycentric interpolation has similar scalability as the standard trilinear interpolation but is more than two times faster (see Figure 5.11).

Size	Ours	Trilinear interp.	PWC interp.	Simple boundaries	Direct discretization
32	7	6	47	6	65
64	8	8	112	32	171
128	8	8	268	Diverged	Diverged
256	9	9	616	Diverged	Diverged

Table 12: *Number of iterations for solving the pressure using variants of our FAS-MG scheme, at frame 10 of scenario 1 in 3D.*

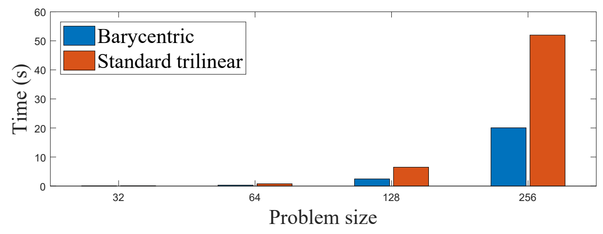


Figure 5.11: *Timing comparison between using barycentric and standard trilinear interpolations for solving the pressure at the 10th frame of scenario 1 in 3D*

6. Conclusion and future work

In summary, we have proposed three methods, namely policy iteration, penalty method, and FAS multigrid, as fast solvers for the pressure equations arising from liquid simulation with separating solid boundary conditions. For our FAS multigrid methodology, we introduced several adaptations to achieve the desired mesh-independent convergence behavior on our LCP fluid problem. We demonstrated the superior efficiency and scalability of our resulting solvers over existing methods. Moreover, our results show that our solvers are able to resolve the liquid sticking issue near the solid boundary without making a major sacrifice in computation time compared with the simpler linear solver case. All of our tests were done on CPU, but it may be interesting to study how these methods perform on GPU. We leave this as future work.

References

[ANE17a] Andersen, Michael, Niebe, Sarah, and Erleben, Kenny. "A fast linear complementarity problem (LCP) solver for separating fluid-solid wall boundary conditions". *Proceedings of the 13th Workshop on Virtual Reality Interactions and Physical Simulations*. Eurographics Association. 2017, 39–48 2, 3, 7, 9.

[ANE17b] Andersen, Michael, Niebe, Sarah, and Erleben, Kenny. "A fast linear complementarity problem solver for fluid animation using high level algebra interfaces for GPU libraries". *Computers & Graphics* 69 (2017), 36–48 2.

[AV11] Adler, Ilan and Verma, Sushil. "The linear complementarity problem, Lemke algorithm, perturbation, and the complexity class PPAD". (2011) 2.

[Bar94] Baraff, David. "Fast contact force computation for nonpenetrating rigid bodies". *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM. 1994, 23–34 2.

[BBB07] Batty, Christopher, Bertails, Florence, and Bridson, Robert. "A fast variational framework for accurate solid-fluid coupling". *ACM Transactions on Graphics (TOG)*. Vol. 26. 3. ACM. 2007, 100 1.

[BM*00] Briggs, William L, McCormick, Steve F, et al. *A multigrid tutorial*. Vol. 72. Siam, 2000 6.

[Bra77] Brandt, Achi. "Multi-level adaptive solutions to boundary-value problems". *Mathematics of computation* 31.138 (1977), 333–390 5.

[Bri08] Bridson, Robert. "Fluid Simulation for Computer Graphics. Ak Peters Series". *Taylor & Francis* 3 (2008), 10 1, 2.

[CA09] Courtecuisse, Hadrien and Allard, Jérémie. "Parallel dense gauss-seidel algorithm on many-core processors". *2009 11th IEEE International Conference on High Performance Computing and Communications*. IEEE. 2009, 139–147 2.

[CM11] Chentanez, Nuttapong and Müller, Matthias. "Real-time Eulerian water simulation using a restricted tall cell grid". *ACM Transactions on Graphics (TOG)*. Vol. 30. 4. ACM. 2011, 82 2.

[CM12] Chentanez, Nuttapong and Mueller-Fischer, Matthias. "A multigrid fluid pressure solver handling separating solid boundary conditions". *IEEE transactions on visualization and computer graphics* 18.8 (2012), 1191–1201 2–7.

[CPS92] Cottle, Richard W, Pang, Jong-Shi, and Stone, Richard E. "The Linear Complementarity Problem. Academic Pr ess". *Inc., Boston, MA* (1992) 5.

[dFL04] d'Halluin, Yann, Forsyth, Peter A, and Labahn, George. "A penalty method for American options with jump diffusion processes". *Numerische Mathematik* 97.2 (2004), 321–352 1.

[DRW15] Dick, Christian, Rogowsky, Marcus, and Westermann, Rüdiger. "Solving the fluid pressure Poisson equation using multigrid—evaluation and improvements". *IEEE transactions on visualization and computer graphics* 22.11 (2015), 2480–2492 1.

[DS05] Dostál, Zdenek and Schoberl, Joachim. "Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination". *Computational Optimization and Applications* 30.1 (2005), 23–43 1.

[Eis81] Eisenstat, Stanley C. "Efficient implementation of a class of preconditioned conjugate gradient methods". *SIAM Journal on Scientific and Statistical Computing* 2.1 (1981), 1–4 4.

[Erl07] Erleben, Kenny. "Velocity-based shock propagation for multibody dynamics animation". *ACM Transactions on Graphics (TOG)* 26.2 (2007), 12 2.

[Erl11] Erleben, Kenny. *Open source project for numerical methods for linear complementarity problems in physics-based animation*. 2011. URL: https://github.com/erleben/num4lcp 7.

[Erl13] Erleben, Kenny. "Numerical methods for linear complementarity problems in physics-based animation". *Acm Siggraph 2013 Courses*. ACM. 2013, 8 2.

[FL07] Forsyth, Peter A and Labahn, George. "Numerical methods for controlled Hamilton-Jacobi-Bellman PDEs in finance". *Journal of Computational Finance* 11.2 (2007), 1 1, 3, 4.

[FM00] Ferris, Michael C and Munson, Todd S. "Complementarity problems in GAMS and the PATH solver". *Journal of Economic Dynamics and Control* 24.2 (2000), 165–188 1.

[FV02] Forsyth, Peter A and Vetzal, Kenneth R. "Quadratic convergence for valuing American options using a penalty method". *SIAM Journal on Scientific Computing* 23.6 (2002), 2095–2122 3, 4.

[FWD14] Ferstl, Florian, Westermann, Rüdiger, and Dick, Christian. "Large-scale liquid simulation on adaptive hexahedral grids". *IEEE transactions on visualization and computer graphics* 20.10 (2014), 1405–1417 2.

[GB13] Gerszewski, Dan and Bargteil, Adam W. "Physics-based animation of large-scale splashing liquids." *ACM Trans. Graph.* 32.6 (2013), 185–1 1.

[GDN97] Griebel, Michael, Dornseifer, Thomas, and Neunhoeffer, Tilman. *Numerical simulation in fluid dynamics: a practical introduction*. Vol. 3. Siam, 1997 2.

[GSLF05] Guendelman, Eran, Selle, Andrew, Losasso, Frank, and Fedkiw, Ronald. "Coupling water and smoke to thin deformable and rigid shells". *ACM Transactions on Graphics (TOG)* 24.3 (2005), 973–981 6.

[GZO10] Gascón, Jorge, Zurdo, Javier S, and Otaduy, Miguel A. "Constraint-based simulation of adhesive contact". *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association. 2010, 39–44 2.

[HW13] Han, Dong and Wan, Justin WL. "Multigrid Methods for Second Order Hamilton–Jacobi–Bellman and Hamilton–Jacobi–Bellman–Isaacs Equations". *SIAM Journal on Scientific Computing* 35.5 (2013), S323–S344 1, 3.

[IEGT17] Inglis, Tiffany, Eckert, M-L, Gregson, James, and Thuerey, Nils. "Primal-Dual Optimization for Fluids". *Computer Graphics Forum*. Vol. 36. 8. Wiley Online Library. 2017, 354–368 1.

[MCPN08] Molemaker, Jeroen, Cohen, Jonathan M, Patel, Sanjit, and Noh, Jonyong. "Low viscosity flow simulations for animation". *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association. 2008, 9–18 2.

[MST10] McAdams, Aleka, Sifakis, Eftychios, and Teran, Joseph. "A parallel multigrid Poisson solver for fluids simulation on large grids". *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association. 2010, 65–74 2, 4.

[NGL10] Narain, Rahul, Golas, Abhinav, and Lin, Ming C. "Free-flowing granular materials with two-way solid coupling". *ACM Transactions on Graphics (TOG)*. Vol. 29. 6. ACM. 2010, 173 1.

[TOS00] Trottenberg, Ulrich, Oosterlee, Cornelius W, and Schuller, Anton. *Multigrid*. Elsevier, 2000 5.

[Var09] Varga, RS. "Matrix Iterative Analysis Second". *Springer Series in Computational Mathematics* 27 (2009) 4.

[WL04] Wan, Justin WL and Liu, Xu-Dong. "A Boundary Condition–Capturing Multigrid Approach to Irregular Boundary Problems". *SIAM Journal on Scientific Computing* 25.6 (2004), 1982–2003 6.