Generalized Non-Reflecting Boundaries for Fluid Re-Simulation

Morten Bojsen-Hansen*
IST Austria

Chris Wojtan*
IST Austria

Abstract

When aiming to seamlessly integrate a fluid simulation into a larger scenario (like an open ocean), careful attention must be paid to boundary conditions. In particular, one must implement special "non-reflecting" boundary conditions, which dissipate out-going waves as they exit the simulation. Unfortunately, the state of the art in non-reflecting boundary conditions (perfectly-matched layers, or PMLs) only permits trivially simple inflow/outflow conditions, so there is no reliable way to integrate a fluid simulation into a more complicated environment like a stormy ocean or a turbulent river.

This paper introduces the first method for combining non-reflecting boundary conditions based on PMLs with inflow/outflow boundary conditions that vary arbitrarily throughout space and time. Our algorithm is a generalization of state-of-the-art mean-flow boundary conditions in the computational fluid dynamics literature, and it allows for seamless integration of a fluid simulation into much more complicated environments. Our method also opens the door for previously-unseen post-process effects like retroactively changing the location of solid obstacles, and locally increasing the visual detail of a pre-existing simulation.

Keywords: fluid simulation, non-reflecting boundaries, perfectly matched layers, re-simulation, stay noided

Concepts: •**Computing methodologies** → **Physical simulation;** *Modeling and simulation; Volumetric models;* •**Mathematics of computing** → *Discretization; Partial differential equations;*

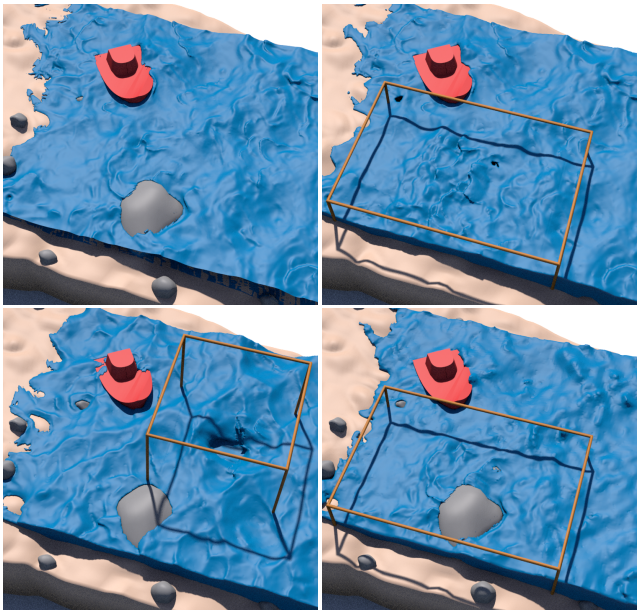

Figure 1: *Given an input fluid simulation (top left), our algorithm can make local changes* retroactively *and seamlessly re-integrate them into the original fluid simulation. Here, we locally edit solid geometry (top right), add a cow splash (bottom left), or re-simulate a specific region at a higher resolution (bottom right). Please see our video.*

1 Introduction

Fluid simulations are indispensable for adding realism to large dynamic environments like open oceans. However, careful attention must be paid to boundary conditions if we wish to seamlessly integrate a fluid simulation into a larger scenario. Instead of making waves spuriously reflect off of invisible walls at the boundary of the simulation domain, we want outgoing waves to quietly dissipate as they exit the simulation; we refer to this desirable behavior as "non-reflecting" boundary conditions.

The state of the art in non-reflecting boundary conditions is known as *perfectly-matched layers* or PMLs [Berenger 1994; Söderström et al. 2010]. At a high level, PMLs work by linearizing the equations of motion, performing a Fourier transform to identify outgoing waves, and exponentially damping the outgoing waves in a thin layer near the boundary of the simulation. Unfortunately, the state of the art in PMLs only permits trivially simple inflow/outflow conditions, like a stationary pool of water or a constantly translating stream (a steady-state mean flow). The state of the art leaves us with no reliable method for integrating a fluid simulation into a non-trivial (i.e., visually interesting) environment like a stormy ocean or a turbulent river.

In this paper, we generalize the state-of-the-art in non-reflecting boundary conditions and present several novel applications. Our contributions are as follows:

- The first PML method with spatially and temporally varying inflow/outflow boundary conditions
- The first derivation of the equations of motion of a perturbation relative to an existing fluid simulation
- The ability to add, remove, and adjust solid boundary obstacles retroactively in a fluid simulation
- The ability to locally re-simulate a fluid animation at a higher resolution (with more visual detail) as a post-process

2 Previous Work

The literature on fluid simulation for computer animation is vast. This discussion will focus primarily on our target problem of developing boundary conditions for fluid simulations, and on our

* (mortenbh|wojtan)@ist.ac.at

SIGGRAPH '16 Technical Paper, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925963

ACM Reference Format
Bojsen-Hansen, M., Wojtan, C. 2016. Generalized Non-Reflecting Boundaries for Fluid Re-Simulation. ACM Trans. Graph. 35, 4, Article 96 (July 2016), 7 pages. DOI = 10.1145/2897824.2925963 http://doi.acm.org/10.1145/2897824.2925963.

target application of editing liquid simulations and efficiently re-simulating simulations.

Boundary conditions for fluid simulations. The problem of non-reflecting boundary conditions originated in computational physics. Berenger [1994] proposed the first method based on perfectly matched layers (PMLs) for the purpose of absorbing electromagnetic waves. PMLs were applied to computational fluid dynamics by Hu et al. [1996], starting with a linearized version of the Euler equations. Researchers subsequently applied PMLs to the non-linear Euler equations [Hu 2006] and Navier-Stokes equations [Hagstrom et al. 2005; Hu et al. 2008].

Hu [2001] and Bécache et al. [2003] showed that PMLs are only guaranteed to properly damp perturbations when the group and phase velocities are in consistent directions. While this disagreement between group and phase velocities never manifests for linear wave equations in a static reference frame, it becomes possible in the presence of a moving background flow, or with more complicated wave dispersion relationships. Their proposed solution to this problem is to apply a coordinate transformation that mathematically guarantees consistency of the group and phase velocities for all wave numbers.

Hu et al. [2008] showed how to create steady state mean flow PML boundary conditions for the Navier-Stokes equations. We generalize their work by allowing mean flows which can vary in time—an essential requirement for realistic computer graphics applications.

Within the field of computer graphics, several researchers proposed simple open boundary conditions for single-phase flows [Stam 1999; Fedkiw et al. 2001], but the wave reflection problem persisted for liquid simulations until the preliminary work of Söderström and Museth [2009] and their thorough follow-up work [Söderström et al. 2010] introduced PMLs for computer animation.

Nielsen and Bridson [2011] re-simulated the surface layer of a low resolution liquid at a higher resolution, using the low resolution simulation as a boundary condition. These guide shapes serve a similar purpose to our time-varying non-reflecting boundary conditions, especially when we inherit our boundary conditions from a lower-resolution simulation. However, their method is not based on PMLs and cannot prevent spurious boundary reflections.

Editing liquid simulations. Our non-reflecting boundary conditions allow a new method for locally editing a fluid animation. While editing a physics simulation is still a challenging problem, researchers in this area have developed several powerful editing techniques already.

Guiding methods [McNamara et al. 2004; Shi and Yu 2005; Thürey et al. 2006] allow more direct control than manipulating initial conditions, which may alleviate the need for going through a large number of iterations. Designing appropriate keyframes can be quite laborious, however, especially if the end result must look physically plausible.

Bhat et al. [2004] present a synthesis approach that allows editing of videos of flows exhibiting roughly stationary dynamics such as waterfalls and rivers. Pighin et al. [2004] go in a different direction and parameterize an Eulerian fluid simulation using advected radial basis functions. Simple edits may then be performed by manipulating flow streamlines. More recently, Pan et al. [2013] developed an interactive sketch-based approach to editing FLIP simulations. User edits are localized in space and time by encoding the edits as deformation fields that are then back-advected and applied with a smooth falloff. The final result is obtained by a guided offline simulation.

Raveendran et al. [2014] introduce a data-driven method for instantly generating new liquid simulations as an interpolation of the inputs using space-time blending. It does not allow arbitrary user edits, however.

Efficient re-simulation. Subspace methods provide a way of significantly reducing the degrees of freedom of the system by leveraging previous simulation data. Recently, Kim and Delaney [2013] addressed the inability of these methods to reproduce the input simulations with a cubature approach. This allowed parameters such as buoyancy, vorticity confinement, and timesteps to be varied and re-simulated efficiently. As with all model reduction methods, continuously changing boundary conditions such as moving solid obstacles or liquid surfaces remain a challenge.

To the best of our knowledge, Srinivasan and Malkawi [2007] provide the only existing method for automatic, localized fluid re-simulation. Their application is indoor airflow visualization for augmented reality. In a pre-computation step, airflow is simulated for a limited number of room topologies (placement and number of openings) using an Eulerian simulator. For each change in boundary condition (*e.g.* adding a window) the user extracts a small number of bounding boxes containing the most significantly changed grid nodes. These bounding boxes are the only areas re-simulated at runtime. Since bounding boxes are determined by running a full simulation, pre-computation time is quite significant, and also suffers from combinatorial explosion as the number of rooms is increased.

3 Perfectly Matched Layers

In this section, we will review the concept of perfectly matched layers (PMLs). We begin our derivation with the incompressible Navier-Stokes equations in conservation form:

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) = \mathbf{0} \tag{1}$$

$$\mathbf{q} = \begin{pmatrix} 0 \\ \mathbf{u} \end{pmatrix} \tag{2}$$

$$\mathbf{F}(\mathbf{q}) = \begin{pmatrix} \mathbf{u} \\ \mathbf{u} \otimes \mathbf{u} + \frac{1}{\rho} pI \end{pmatrix} \tag{3}$$

Here, $\mathbf{u}$ denotes the usual three-dimensional velocity, ρ is density, p is pressure, and I is a 3×3 identity matrix. Notice that Equation (1) includes both the momentum equation and the zero-divergence constraint. Our exposition neglects viscosity and external forces for clarity, but they can easily be be included as in Söderström et al. [2010].

3.1 Basic PMLs

In the absence of any background flow, the perfectly matched layer will aim to exponentially damp $\mathbf{q}$ toward zero in a thin layer near the boundary. We apply the split variable approach [Berenger 1994] to split $\mathbf{q}$ into separate vectors associated with each spatial dimension $\mathbf{q} = \mathbf{q}_1 + \mathbf{q}_2 + \mathbf{q}_3$. The equation for the time evolution of $\mathbf{q}_1$ ($\mathbf{q}_2$ and $\mathbf{q}_3$ are analogous) is:

$$\frac{\partial \mathbf{q}_1}{\partial t} + \frac{\partial \mathbf{F}_1(\mathbf{q})}{\partial x} = \mathbf{0} \tag{4}$$

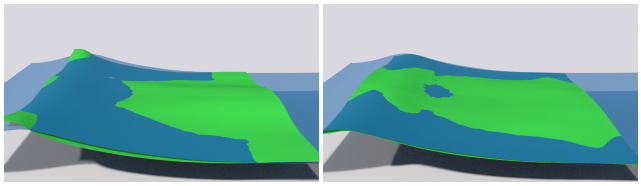

Figure 2: *A simulation without (left) and with (right) perfectly matched layers (PMLs) with time-varying inflow/outflow boundary conditions. The simulation without PMLs exhibits interference patterns from wave reflections. The background flow is visualized in blue. For demonstration purposes this example does not include visual blending.*

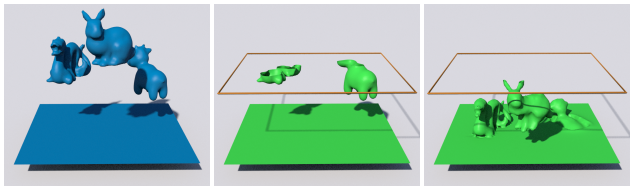

Figure 3: *The background flow (left, blue) includes geometry outside of the simulated flow (middle, green). The upper boundary of the new simulation is indicated by the wire rectangle on the right. Nevertheless, new geometry (the bunny) flows in through the upper boundary as the simulation progresses.*

where

$$\mathbf{F}_1 = \begin{pmatrix} u_x \\ u_x^2 + p/\rho \\ u_x u_y \\ u_x u_z \end{pmatrix}, \mathbf{F}_2 = \begin{pmatrix} u_y \\ u_x u_y \\ u_y^2 + p/\rho \\ u_y u_z \end{pmatrix}, \mathbf{F}_3 = \begin{pmatrix} u_z \\ u_x u_z \\ u_y u_z \\ u_z^2 + p/\rho \end{pmatrix} \tag{5}$$

Our notation uses numerical subscripts to denote quantities and operators associated with split variables, while we use x, y, z subscripts to denote a velocity component in a particular spatial dimension. Note that we recover Equation (1) when we sum the split components defined by Equation (4).

We convert to the frequency domain by applying a Fourier transform $\partial/\partial t \rightarrow -i\omega$ and achieve a spatial stretching in the boundary layer (following Söderström et al. [2010]) with the transformation

$$\frac{\partial}{\partial x} \rightarrow \frac{1}{1 + i\sigma_1/\omega} \frac{\partial}{\partial x} \tag{6}$$

to get

$$-i\omega \widetilde{\mathbf{q}}_1 + \frac{1}{1 + i\sigma_1/\omega} \frac{\partial \widetilde{\mathbf{F}_1(\mathbf{q})}}{\partial x} = \mathbf{0} \tag{7}$$

where the tilde notation indicates a quantity in the frequency domain. Here, $\sigma_1(x)$ is a spatially varying transfer function that depends only on x (not y or z). It will be used to damp waves traveling in the x-direction by setting it to zero in the simulation domain and ramping it up to a large positive value in the boundary layer. We then multiply through by $(1 + i\sigma_1)/\omega$ to get

$$-i\omega \widetilde{\mathbf{q}}_1 + \sigma_1 \widetilde{\mathbf{q}}_1 + \frac{\partial \widetilde{\mathbf{F}_1(\mathbf{q})}}{\partial x} = \mathbf{0} \tag{8}$$

and finally transform back to the original domain with the inverse Fourier transform $-i\omega \rightarrow \partial/\partial t$

$$\frac{\partial \mathbf{q}_1}{\partial t} + \sigma_1 \mathbf{q}_1 + \frac{\partial \mathbf{F}_1(\mathbf{q})}{\partial x} = \mathbf{0} \tag{9}$$

Numerically integrating this equation will exponentially damp $\mathbf{q}_1$ towards zero in the boundary layer. Summing up each of the split components at the end of each time step will recover $\mathbf{q}$.

3.2 Background Flows

By damping to $\mathbf{q} = \mathbf{0}$ near the boundary, Equation (9) implies that the simulation is located in the middle of a perfectly static fluid. To allow for more interesting background motions, Hu et al. [2008] damp towards a background flow $\overline{\mathbf{q}}$ instead of towards zero. They interpret $\mathbf{q}$ as the summation of the background flow $\overline{\mathbf{q}}$ and a perturbation flow $\mathbf{q}'$:

$$\mathbf{q} = \overline{\mathbf{q}} + \mathbf{q}' \tag{10}$$

where $\mathbf{q}$ and $\overline{\mathbf{q}}$ are solutions to Equation (1). Note, however, that because the Navier-Stokes equations are non-linear, $\mathbf{q}'$ will generally *not* be a solution to Equation (1), making the problem of solving for the motion of $\mathbf{q}'$ substantially more complicated.

Hu [2006] simplifies this problem by assuming that the background flow is already in steady state, i.e., $\partial\overline{\mathbf{q}}/\partial t = \mathbf{0}$, which implies $\partial\mathbf{q}'/\partial t = \partial\mathbf{q}/\partial t$ by Equation (10). Thus, under the steady-state assumption, the dynamics of $\mathbf{q}'$ are just $\partial\mathbf{q}'/\partial t + \nabla \cdot \mathbf{F}(\mathbf{q}) = \mathbf{0}$. Applying the PML transformations above gives us

$$\frac{\partial \mathbf{q}_1'}{\partial t} + \sigma_1 \mathbf{q}_1' + \frac{\partial \mathbf{F}_1(\mathbf{q})}{\partial x} = \mathbf{0} \tag{11}$$

which is essentially the same as Equation (9), except it damps $\mathbf{q}'$ to zero near the boundaries instead of damping the entire $\mathbf{q}$. Intuitively, this drives $\mathbf{q}$ toward the background flow $\overline{\mathbf{q}}$ near the boundary of the simulation domain. This represents the state of the art in non-reflecting boundary conditions, which we will improve upon in the next section.

4 Time-varying Background Flows

While steady-state background flows are convenient for applications like the analysis of airfoils, they are extremely limiting for computer graphics applications. The steady-state assumption is simply insufficient if we wish to have our simulation live within a more natural environment, like an undulating ocean or a turbulent river. In this section we explain our main contribution of achieving time-varying background flows with perfectly matched layers.

We again divide our main simulation variable $\mathbf{q}$ into a background flow $\overline{\mathbf{q}}$ and a perturbation $\mathbf{q}'$, as in Equation (10), but we remove the assumption that $\partial\overline{\mathbf{q}}/\partial t = \mathbf{0}$. As a consequence, we can no longer equate $\partial\mathbf{q}'/\partial t$ to $\partial\mathbf{q}/\partial t$, and we cannot make use of the previous derivation.

However, since both $\mathbf{q}$ and $\overline{\mathbf{q}}$ are solutions to the Navier-Stokes equations, we can write

$$\left[\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q})\right] - \left[\frac{\partial \overline{\mathbf{q}}}{\partial t} + \nabla \cdot \mathbf{F}(\overline{\mathbf{q}})\right] = \mathbf{0} \tag{12}$$

We then regroup the terms

$$\left[\frac{\partial \mathbf{q}}{\partial t} - \frac{\partial \overline{\mathbf{q}}}{\partial t}\right] + [\nabla \cdot \mathbf{F}(\mathbf{q}) - \nabla \cdot \mathbf{F}(\overline{\mathbf{q}})] = \mathbf{0} \tag{13}$$

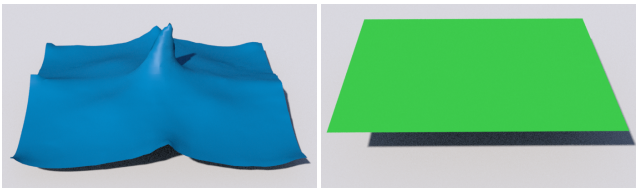

Figure 4: *Even when the background flow (left, blue) is very lively, our algorithm is able to produce a perfectly static pool (right, green).*

and redistribute the linear derivatives.

$$\frac{\partial[\mathbf{q}-\overline{\mathbf{q}}]}{\partial t}+\nabla\cdot[\mathbf{F}(\mathbf{q})-\mathbf{F}(\overline{\mathbf{q}})]=\mathbf{0} \tag{14}$$

We next substitute the definition of $\mathbf{q}'$ to derive the dynamics of the perturbation.

$$\frac{\partial\mathbf{q}'}{\partial t}+\nabla\cdot[\mathbf{F}(\mathbf{q})-\mathbf{F}(\overline{\mathbf{q}})]=\mathbf{0} \tag{15}$$

Note that, although the resulting equation is simple, we made no assumptions in its derivation. In particular, we made no linearizations, and $\mathbf{q}'$ is not restricted to be a solution to the Navier-Stokes equations. We then apply the same splitting and coordinate transformations above to recover the PML-specific dynamics.

$$\frac{\partial\mathbf{q}_1'}{\partial t}+\sigma_1\mathbf{q}_1'+\frac{\partial[\mathbf{F}_1(\mathbf{q})-\mathbf{F}_1(\overline{\mathbf{q}})]}{\partial x}=\mathbf{0} \tag{16}$$

Integrating this equation will again damp $\mathbf{q}_1'$ toward zero near the boundaries, but it will also allow $\overline{\mathbf{q}}$ to vary arbitrarily over space and time. Intuitively, $\mathbf{q}$ will continually be damped towards $\overline{\mathbf{q}}$, but now $\overline{\mathbf{q}}$ is a moving target that is free to erratically splash up and down or flow in and out of the domain.

Our derivation of Equation (16) is similar to the one of Hu [2006]. Hu goes on to assume a pseudo mean flow that satisfies steady-state Navier-Stokes, however, our application crucially depends on having a time-varying background flow, which may not be as obviously useful for CFD applications.

4.1 An Addendum for Liquid Surfaces

Equation (16) naturally damps differences in the velocity field near the boundary, but it will not remove differences in the liquid surface geometry. In order to drive the *entire physical state* toward the time-varying background flow, we may also add a small damping term to the free surface geometry. In the case of a level set [Osher and Fedkiw 2006], which is what we used in all of our experiments, this can be expressed

$$\frac{D\phi}{Dt}+\gamma\left(\phi-\overline{\phi}\right)=0 \tag{17}$$

where γ is a transfer function similar to σ that ramps upward in the boundary layer, ϕ is the level set function representing the geometry of the liquid surface, $\overline{\phi}$ is the level set of the background flow and D/Dt is the material derivative. In our examples, we found that the free-surface advection combined with the copying operation described below was accurate enough to safely set $\gamma = 0$, but we document this concept for completeness. We did not experiment with other surface trackers such as triangle meshes or particles.

This damping operation removes continuous deformations of the liquid surface geometry, but it cannot change topology, like when an inflowing velocity field brings entirely new surface geometry through the boundary. In such cases, we must explicitly add the new geometric components to our surface tracker. We do this by copying all liquid geometry from the background flow within a layer with width dependent on a CFL condition (usually 3 cells) beyond the simulation boundary, and we extrapolate the velocity from these regions as well. This way, the liquid surface advection algorithm naturally carries new geometry components in through the boundaries of the simulation. See Figure 3 for a proof of concept.

Algorithm 1 Pseudocode for our one time step of our algorithm.

```
while sub-cycling do
    Compute sub-cycle time step size Δt_sub
    Advect q' with step size Δt_sub
    Damp q' with step size Δt_sub (Equation (18))
end while
Add Body Force
Pressure Projection
Extrapolate q
Advect Level Set
```

5 Implementation Details

We implemented our method as a set of plug-ins for Houdini [Side Effects Software 2016]. The source code for these plug-ins as well as an example Houdini scene file is included in the supplementary material.

Our implementation updates $\mathbf{q}'$ by time-splitting. We analytically integrate the middle (damping) term of Equation (16):

$$\mathbf{q}' \leftarrow \mathbf{q}' e^{-\sigma_1 \Delta t} \tag{18}$$

The rightmost term of Equation (16) encodes advection, the pressure projection, and the pressure update. Like most solvers in computer graphics, we apply time splitting and numerically integrate each term separately. Whenever our solver needs to evaluate an element of $\mathbf{q}$, we sum up each of the split components with $\mathbf{q} = (\mathbf{q}_1' + \mathbf{q}_2' + \mathbf{q}_3') + (\overline{\mathbf{q}}_1 + \overline{\mathbf{q}}_2 + \overline{\mathbf{q}}_3)$.

The order of our time splitting almost exactly follows Söderström et al. [2010], including the advection, the pressure projection, and the addition of conservative body forces like gravity. Because the explicit advection algorithm proposed by Söderström et al. [2010] is conditionally stable, we perform multiple subcycles of the advection and damping routines with the maximum stable time step based on the CFL condition [Bridson 2008]. We track the liquid free-surface using Houdini's particle level set method [Osher and Fedkiw 2006]. Pseudocode for one simulation time step can be seen in Algorithm 1.

To achieve good damping performance for the PMLs, it is important to pick good parameters. Söderström et al. [2010] performed a rather thorough experimental study of the effect of different choices for $\sigma(x)$, $\sigma_{\max}$ and PML width. Although our setting is slightly different from theirs, we found that basing our parameters on their findings worked well in practice.

In our examples, we set each PML's width to 8% of the simulation domain, and we use the transfer function $\sigma(x) = \sigma_{\max}(x/L)^3$, where x is the distance from the end of the simulation domain, L is the width of the PML and $\sigma_{\max} = 77$. We set the geometry blending coefficient γ to 0.

The PML's exponential damping in the boundary layer is essential for efficiently removing perturbations, but the change is a bit too

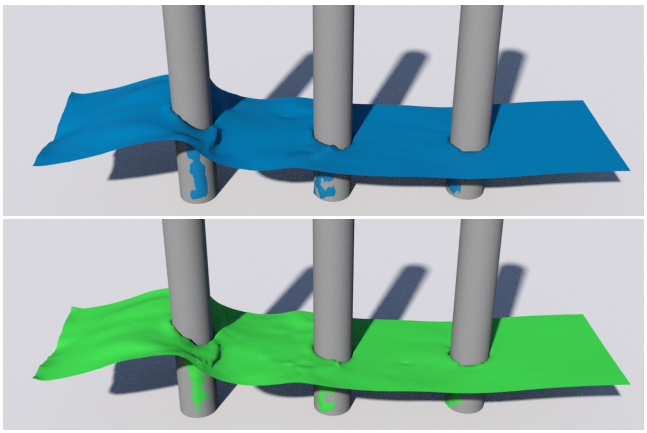

Figure 5: *Our simulation (bottom, green) successfully reproduces the background flow (top, blue) when there are no perturbations.*

sudden for the purpose of compositing a new simulation into an existing one as in Figure 1. For this visual transition, we remove the PML geometry and linearly blend the surface geometry in the outer $6-12\%$ of the simulation domain as a post-process.

During each time step, we fetch part of the background flow $\bar{\mathbf{q}}$ from disk. To make the computational complexity independent of the size of the background simulation, we modified OpenVDB [Museth 2013] to support sparse *out-of-core* grids.

6 Evaluation

We made a small collection of examples to show the robustness of our algorithm. We first verify that our solver can compute the correct behavior even when the correct motion $\mathbf{q}$ deviates far from the background flow $\bar{\mathbf{q}}$. Figure 4 shows that we can recreate a completely static pool even if the background flow is very lively. We do this by effectively deleting a falling sphere from the background flow, and simulating the result as if the original splash never took place ($\mathbf{q}' = -\bar{\mathbf{q}}$). Note how our resulting motion is independent of the background flow, which would not have been possible if we assumed the perturbation $\mathbf{q}'$ was small or if we linearized the Navier-Stokes equations about $\bar{\mathbf{q}}$.

In the other extreme, if we do not make any perturbations at all ($\mathbf{q}' = \mathbf{0}$), then our method reproduces the background flow $\bar{\mathbf{q}}$. Figure 5 shows that our method reproduces the expected motion in the absence of perturbations, and it does not drift away from the original simulation.

Figure 3 shows how our method advects new surface geometry into the simulation when dictated by the background flow. This behavior is important for merging simulations together, like when droplets spray into the simulation domain from somewhere outside.

7 Applications

Our method can create a simulation that appears to be part of a much larger fluid domain. Figure 2 (right) shows how we can use boundary conditions from a gently sloshing pool to create a new simulation set in the middle of a larger simulation. We first notice that ocean waves roll in and out through the domain boundary; this behavior is impossible with previous approaches. Next, we see that even though our simulation creates a large splash, the waves are absorbed by the boundaries. If we do not use our PMLs Figure 2 (left), we see obvious wave reflection artifacts.

Next, we can retroactively change parts of an existing simulation without re-running the entire simulation. We begin with a complicated beach flow that was computed previously. In the top right of Figure 1, we create a small simulation domain around a solid obstacle that we wish to change, using the pre-computed simulation in the top left of Figure 1 as the background flow $\bar{\mathbf{q}}$. We completely remove the obstacle, locally creating a new flow. The combination of non-reflecting boundaries and our novel time-varying background flow allow us to seamlessly merge this simulation together with the larger one. We also perform a similar process in the bottom left of Figure 1, by re-simulating a new splash into the beach simulation as a post-process.

Our method can also retroactively increase the resolution of a portion of a simulation, allowing users to "zoom in" as a post-process. In the bottom right of Figure 1, we create a higher resolution simulation domain where we wish to add more detail to the beach simulation. We then use a lower-resolution flow as the $\bar{\mathbf{q}}$ in our boundary conditions, and we run the new simulation in a small subset of this domain at a much higher resolution. Again, our method allows the two simulations to be seamlessly blended together at the simulation boundaries.

Although we view our method as a working prototype and have not optimized it for performance, we list performance figures in Table 1. We believe that the low-resolution examples simulate $\bar{\mathbf{q}}$ significantly faster than $\mathbf{q}'$ because $\mathbf{q}'$ has substantial I/O overhead when reading the boundary conditions from the precomputed $\bar{\mathbf{q}}$. As we localize the flows at higher resolutions, however, the $\mathbf{q}'$ simulations are much faster than the original $\bar{\mathbf{q}}$, allowing many post-processing passes once an initial simulation is computed.

8 Discussion

This work represents a significant generalization of the state of the art in non-reflecting boundary conditions for fluid simulation. Our method opens the door for novel post-process simulation editing and enhancement, and we presented prototypes for retroactively editing solid geometry and increasing simulation resolution.

The theory behind perfectly matched layers guarantees exponential damping of waves in the boundary layer for linear partial differential equations. However, not much is known about guarantees (if any exist) for non-linear equations like ours. The method seems to work exceptionally well in practice, but further theoretical development on this topic should be conducted. We observe that PMLs for the Navier-Stokes equations are dramatically more efficient than the naïve approach of damping the velocity field near the boundary, because they require a far smaller damping region and thus less memory and overall computation time.

Hu [2001] and Bécache et al. [2003] showed that theoretical guarantees on PMLs are only valid when the flow's group and phase velocities are in the same direction. We observed this problem for the scalar wave equation but never for Navier-Stokes. We suspect this is due to the natural frequency dispersion of surface water waves and numerical diffusion in our Navier-Stokes solver. As a result of our observations, we found it unnecessary to implement any of the corrections in the literature for 3D free-surface flows.

In our implementation, highly complex solid obstacles that intersected the boundary layer would occasionally prevent perturba-

Example	Resolution	Run Time
Figure 3 $\bar{\mathbf{q}}$	$100 \times 100 \times 100$	25m 33s
Figure 3 $\mathbf{q}'$	$100 \times 50 \times 100$	45m 03s
Figure 4 $\bar{\mathbf{q}}$	$133 \times 53 \times 33$	10m 28s
Figure 4 $\mathbf{q}'$	$133 \times 53 \times 33$	21m 43s
Figure 5 $\bar{\mathbf{q}}$	$50 \times 50 \times 50$	6m 32s
Figure 5 $\mathbf{q}'$	$50 \times 50 \times 50$	11m 23s
Figure 2 (left) without PML	$53 \times 32 \times 60$	4m
Figure 2 (right) with PML	$53 \times 32 \times 60$	4m
Figure 1 (top left) $\bar{\mathbf{q}}$	$453 \times 100 \times 307$	15h
Figure 1 (top right) $\mathbf{q}'$	$227 \times 67 \times 153$	3h 47m
Figure 1 (bottom left) $\mathbf{q}'$	$133 \times 107 \times 133$	47m 17s
Figure 1 (bottom right) $\mathbf{q}'$	$227 \times 67 \times 153$	3h 30m

Table 1: *Performance figures for the examples in our video. The background flow and perturbation flow (the new simulation) are indicated by* $\bar{\mathbf{q}}$ *and* $\mathbf{q}'$*, respectively.*

tions from quickly damping out. Less complex solid obstacle geometry, however, posed no difficulties (as exhibited by Figure 1). We have yet to conclude whether this is a general theoretical problem or one specific to our implementation.

Our current implementation assumes that the background flow $\bar{\mathbf{q}}$ is a solution to the Navier-Stokes equations discretized by our split-variable solver. We would like to remove this restriction in the future by allowing $\bar{\mathbf{q}}$ to be an arbitrary vector field (*i.e.* from a different fluid solver, or even from an unphysical artist-designed flow). We believe that this can be made possible by adding source terms to the equation of motion and applying the appropriate PML transforms.

The fact that advection is split over three equations prevents an easy extension to semi-Lagragian methods and also to FLIP. This is true even for a non-split variable approach as presented in Hu [2006]. We see it as a very fruitful avenue of future work to investigate how our method could be reconciled with such methods.

A property of our method is that it tends to preserve artifacts present in the input unless they are perturbed. Figure 5 exhibits small bumps in the input animation due to our use of a particle level set. Similarly, the input simulation in Figure 1 contains subtle "tendril"-like artifacts, which we suspect are caused by the wave generation algorithm linearly blending the fluid velocity field with a procedural vector field.

We imagine several extensions to our methods for retroactively improving a simulation. For example, we hope to combine our approach with an adaptively re-sizing simulation domain, allowing us to locally halt the simulation where the perturbation flow has damped out, and to adaptively expand the simulation domain where interesting new flows persist.

Acknowledgements

We thank David Hahn, Stefan Jeschke and Rok Grah for help proofreading our paper, the IST Austria Visual Computing group for helpful feedback throughout the project, and the anonymous reviewers for useful comments on our work. Finally, we thank Side Effects Software for Houdini licences.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 638176.

References

BÉCACHE, E., FAUQUEUX, S., and JOLY, P. 2003. Stability of Perfectly Matched Layers, Group Velocities and Anisotropic Waves. In *J. Comput. Phys.* 188.2, pp. 399–433. ISSN: 0021-9991.

BERENGER, J.-P. 1994. A Perfectly Matched Layer for the Absorption of Electromagnetic Waves. In *J. Comput. Phys.* 114.2, pp. 185–200. ISSN: 0021-9991.

BHAT, K. S., SEITZ, S. M., HODGINS, J. K., and KHOSLA, P. K. 2004. Flow-based Video Synthesis and Editing. In *ACM Transactions on Graphics (SIGGRAPH)* 23.3, pp. 360–363.

BRIDSON, R. 2008. *Fluid simulation for computer graphics*. CRC Press.

FEDKIW, R., STAM, J., and JENSEN, H. W. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, pp. 15–22.

HAGSTROM, T., GOODRICH, J., NAZAROV, I., and DODSON, C. 2005. High-order methods and boundary conditions for simulating subsonic flows. In *Proceedings of the 11th AIAA/CEAS Aeroacoustics Conference*.

HU, F. Q. 2001. A Stable, Perfectly Matched Layer for Linearized Euler Equations in Unsplit Physical Variables. In *J. Comput. Phys.* 173.2, pp. 455–480. ISSN: 0021-9991.

HU, F. Q. 2006. On the construction of PML absorbing boundary condition for the non-linear Euler equations. In *AIAA paper* 798, p. 2006.

HU, F. Q., LI, X., and LIN, D. 2008. Absorbing boundary conditions for nonlinear Euler and Navier–Stokes equations based on the perfectly matched layer technique. In *Journal of Computational Physics* 227.9, pp. 4398–4424.

HU, F., HUSSAINI, M., and MANTHEY, J. 1996. Low-Dissipation and Low-Dispersion Runge-Kutta Schemes for Computational Acoustics. In *J. Comput. Phys.* 124.1, pp. 177–191. ISSN: 0021-9991.

KIM, T. and DELANEY, J. 2013. Subspace fluid re-simulation. In *ACM Transactions on Graphics (SIGGRAPH)* 32.4, 62:1–62:9.

MCNAMARA, A., TREUILLE, A., POPOVIĆ, Z., and STAM, J. 2004. Fluid Control Using the Adjoint Method. In *ACM Transactions on Graphics (SIGGRAPH)* 23.3, pp. 449–456.

MUSETH, K. 2013. VDB: High-Resolution Sparse Volumes With Dynamic Topology. In *ACM Transactions on Graphics (to appear)* 32.3.

NIELSEN, M. B. and BRIDSON, R. 2011. Guide Shapes for High Resolution Naturalistic Liquid Simulation. In *ACM Transactions on Graphics (SIGGRAPH)* 30.4, 83:1–83:8.

OSHER, S. and FEDKIW, R. 2006. *Level set methods and dynamic implicit surfaces*. Vol. 153. Springer Science & Business Media.

PAN, Z., HUANG, J., TONG, Y., ZHENG, C., and BAO, H. 2013. Interactive Localized Liquid Motion Editing. In *ACM Transactions on Graphics (SIGGRAPH Asia)* 32.6.

PIGHIN, F., COHEN, J. M., and SHAH, M. 2004. Modeling and editing flows using advected radial basis functions. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pp. 223–232.

RAVEENDRAN, K., WOJTAN, C., THÜREY, N., and TURK, G. 2014. Blending Liquids. In *ACM Transactions on Graphics (SIGGRAPH)* 33.4, 137:1–137:10.

SHI, L. and YU, Y. 2005. Taming Liquids for Rapidly Changing Targets. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pp. 229–236.

SIDE EFFECTS SOFTWARE. 2016. *Houdini*. http://sidefx.com.

SÖDERSTRÖM, A., KARLSSON, M., and MUSETH, K. 2010. A PML-based Nonreflective Boundary for Free Surface Fluid Animation. In *ACM Transactions on Graphics (TOG)* 29.5, 136:1–136:17.

SÖDERSTRÖM, A. and MUSETH, K. 2009. Non-reflective Boundary Conditions for Incompressible Free Surface Fluids. In *SIGGRAPH 2009: Talks*. SIGGRAPH '09. New Orleans, Louisiana: ACM, 4:1–4:1. ISBN: 978-1-60558-834-6.

SRINIVASAN, R. and MALKAWI, A. 2007. Adaptive Localization Method: An Approach to Real Time Airflow Simulation and Immersive Visualization. In *Proceedings of the International Conference on Computer Graphics and Vision (GraphiCon)*.

STAM, J. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., pp. 121–128.

THÜREY, N., KEISER, R., RUEDE, U., and PAULY, M. 2006. Detail-Preserving Fluid Control. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pp. 7–12.