

Progressing Level-of-Detail Animation for Volumetric Elastodynamics

JIAYI ERIS ZHANG, Stanford University, USA and Adobe, USA

DOUG L. JAMES, Stanford University, USA

DANNY M. KAUFMAN, Adobe, USA

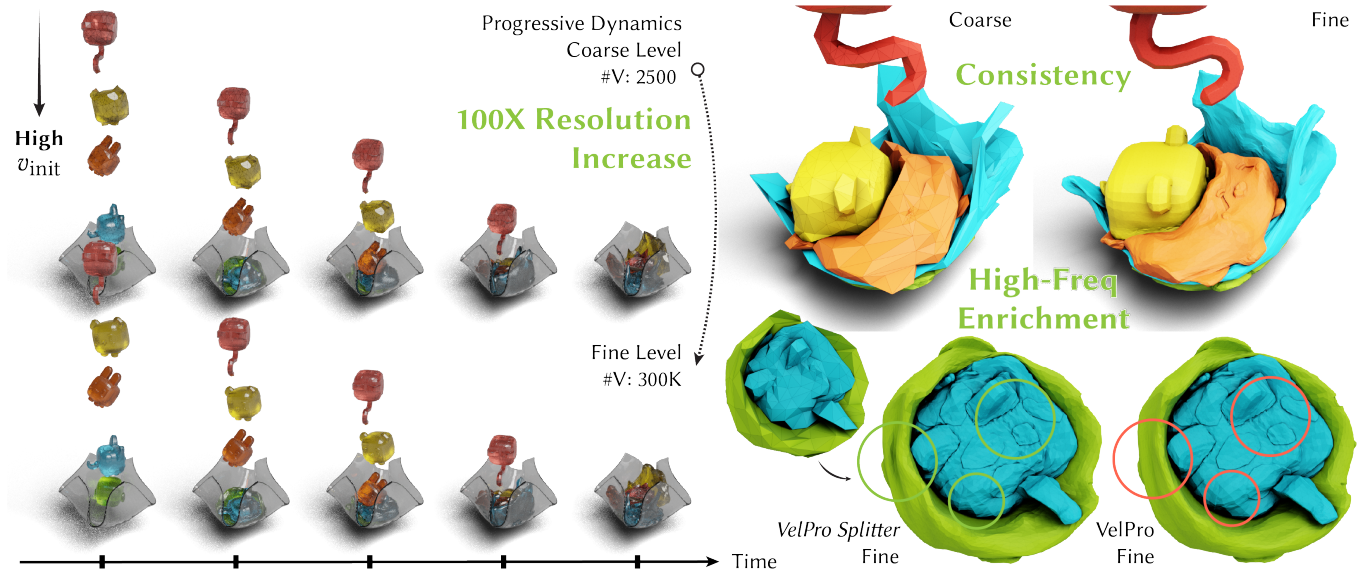


Fig. 1. **Animal Jello:** (Left) We drop a set of animal jello characters into a glass container with high initial downward velocities and simulate the motion with two-level Volumetric Progressive Dynamics (VolPD) under a large coarse-fine resolution gap over 100 $\times$. Despite this substantial resolution gap, VolPD produces stable motion with spatially consistent contact and temporally continuous animation. (Right, top) A zoomed-in view highlights close frame-to-frame consistency (Frame 216). (Right, bottom) Our newly introduced *VelPro Splitter* integrator recovers high-frequency dynamics, substantially enriching motion beyond what geometric refinement and improved contact resolution alone can provide, compared to VelPro [Zhang et al. 2025].

We extend Progressive Dynamics [Zhang et al. 2024, 2025] from cloth and shells to volumetric finite elements, enabling an efficient level-of-detail (LOD) animation-design pipeline with predictive coarse-resolution previews for rapid iteration toward a final high-resolution volumetric elastodynamics animation. To achieve this, we introduce *VelPro Splitter*, a principled and substantially improved VelPro-type integrator [Zhang et al. 2025] that splits the current-level velocity and recombines its high-resolution component with prolonged coarse-level velocity. The resulting VelPro Splitter is general and applies to both shell and volumetric discretizations. This splitting strategy better generates the high-frequency dynamic details that motivate high-resolution elastodynamics animation, going beyond the previously mostly geometric enrichment due to prolonged coarse-level velocity, while maintaining cross-level consistency in bulk deformation across LOD results. As a result, it largely decouples finest-level enrichment quality from the number

of LOD levels, addressing a practical limitation of VelPro at small timestep sizes where many levels are required to obtain sufficient enrichment.

To make this volumetric setting practical for Progressive Dynamics, we add two supporting contributions. First, we construct volumetric hierarchies and introduce a simple and effective topology-aware boundary-binding method that enables reliable prolongation between overlapping, but not necessarily-conforming, meshes using a barycentric-like linear interpolant. Second, instead of applying the subspace approach of Zhang et al. [2024] for reducing coarse model locking, we show that a lightweight, resolution-based stiffness rescaling via a simple Young’s-modulus adjustment [Chen et al. 2017] is both effective and well-suited for progressive volumetric simulation. Together, these contributions jointly realize Volumetric Progressive Dynamics. We demonstrate its high-fidelity LOD matching for volumetric elastodynamics across 1D, 2D, and 3D scenarios with high speeds, large deformations, and frictional contact.

Authors’ Contact Information: Jiayi Eris Zhang, Stanford University, USA and Adobe, USA, eriszhan@stanford.edu; Doug L. James, Stanford University, USA, djames@cs.stanford.edu; Danny M. Kaufman, Adobe, USA, dannykaufman@gmail.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 1557-7368/2026/7-ART150
<https://doi.org/10.1145/3811383>

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Progressive Simulation, LOD Animation, Volumetric Simulation, Linear Finite Elements

ACM Reference Format:

Jiayi Eris Zhang, Doug L. James, and Danny M. Kaufman. 2026. Progressing Level-of-Detail Animation for Volumetric Elastodynamics. *ACM Trans. Graph.* 45, 4, Article 150 (July 2026), 17 pages. <https://doi.org/10.1145/3811383>

1 Introduction

Volumetric elastodynamics is widely used to bring deformable objects and their interactions to life, but high-quality results typically require high-resolution discretizations with many degrees of freedom and thus substantial computational cost. This is especially painful in iterative creative workflows where artists need rapid updates and visual feedback while exploring many variations of a simulation. To accelerate this iterative design process, recent work [Zhang et al. 2024, 2025] introduced Progressive Dynamics for rapid previewing and interactive design of high-fidelity cloth and shell animation, producing predictive coarse previews that progressively refine to high-resolution results through a coarse-to-fine LOD pipeline. Extending this success from shells to volumetric domains, however, is challenging for three main reasons.

First, although in the shell setting, to achieve LOD progressive refinement across levels with increasing fidelity, Zhang et al. [2024, 2025] have explored various progressive integrators for resolution-time integration; so far, this line of work has mainly focused on enriching geometric details such as folds and wrinkle details on a frame-by-frame basis. However, for volumetric elastodynamic animation, while geometric refinement remains important (e.g., built-in surface details from the rest shape reappearing under deformation as it refines), finer resolution is often needed to resolve high-frequency *dynamics*, such as impacts and contact transients, fast waves propagating through an object, and secondary jiggles. This poses a challenge for the VelPro integrator of Zhang et al. [2025]. Its finer-level velocity update is driven by coarse-to-fine prolongation from the coarser level, rather than the finer level’s own previous-timestep velocity as in a direct same-level simulation. As a result, the refined motion can be overly constrained by the coarse level, potentially missing fast dynamics and the high-frequency detail that only the finest discretization can resolve.

Second, although Zhang et al. [2024, 2025] built a fine-to-coarse shell hierarchy for LOD and introduced a novel recursive edge-collapse decimation approach, the same machinery does not directly carry over to tetrahedral meshes. In the shell setting, their approach explicitly tracks bijective patch correspondences for composing and constructing effective prolongation operators that transfer positions and velocities between levels. For tetrahedral meshes, existing tetrahedral decimation and simplification methods typically do not provide comparable local mappings (see Sections 2.3 and 2.4 for a comprehensive discussion). Thus, it remains to be seen what a good strategy is for building such hierarchies and operators in the volumetric setting.

Third, elastic finite element (FE) models at different resolutions can exhibit different effective stiffnesses due to numerical stiffening. Coarse discretizations often behave stiffer than fine ones, leading to mismatched deformation even under the same boundary conditions and material parameters [Chen et al. 2017]. Although prior Progressive Simulation work incorporates coarse-level subspace models [Zhang et al. 2023, 2022, 2024, 2025] to address the locking artifacts (sharp and unnatural creasing and buckling errors), stiffness mismatch remains a challenge, and in particular, the volumetric case remains unexplored.

To address these challenges, we propose Volumetric Progressive Dynamics (VolPD), an end-to-end, practical LOD animation-design pipeline for FE-based volumetric elastodynamics. VolPD enables predictive coarse previews for rapid design iteration, then progressively refines to a final high-resolution animation that preserves the coarse motion while adding high-frequency dynamics. We first diagnose key limitations of the VelPro integrator of Zhang et al. [2025], and show that effective progressive integrator design requires velocity updates that better capture fine-scale detail by explicitly separating coarse and fine scales between the fine-level velocity and the prolonged coarse velocity. Building on this insight, our primary contribution is **VelPro Splitter**, a substantially improved VelPro-type progressive integrator that works effectively for both shells and volumes. It splits and recombines low- and high-resolution velocity content across LOD levels to produce stable, spatially consistent, and temporally continuous animation, while enabling high-frequency dynamic enrichment beyond geometric refinement and contact resolution. It also introduces a principled notion of bulk motion through a spatial projection operator with a clean orthogonality property, algebraically separating coarse- and fine-representable motion so finer levels remain consistent. As a result, *VelPro Splitter* largely decouples finest-level enrichment quality from the number of LOD levels, addressing a practical limitation of VelPro at small timestep sizes where many levels are needed.

Accompanying this, to generate multiresolution tetrahedral mesh hierarchies, we propose a practical pipeline that combines standard surface mesh decimation with tetrahedralization. We assume that the resulting hierarchy comprises tetrahedral meshes that *loosely* approximate the same domain with overlapping interiors, but may exhibit non-conforming boundaries. Building on this setting, we introduce a simple yet effective topology-aware method for constructing linear prolongation operators between overlapping, non-conforming volumetric meshes. Treating the coarse mesh as the embedding shape, the key idea is a boundary-binding strategy that enables barycentric-coordinate evaluation for fine vertices via extrapolation when necessary.

In addition, while the subspace model used in prior Progressive Simulation methods [Zhang et al. 2023, 2022, 2024, 2025] is highly effective at improving coarse preview quality by reducing membrane locking in shells, this shell-specific artifact is absent in the volumetric setting, and the numerical stiffening issue remains unaddressed in previous Progressive Simulation frameworks. We realize that VolPD can instead benefit more from adapting a much cheaper Young’s-modulus rescaling of the coarse model [Chen et al. 2017], which better matches the primary mode across resolutions and mitigates numerical stiffening that would otherwise lead to cross-level inconsistencies in the animation.

1.1 Contributions

Together, these components *jointly* enable a practical generalization of Progressive Dynamics to volumetric elastodynamic animation that was not previously possible, which is an important step toward a unified FE-based approach for consistent, high-quality animation design via rapid preview and refinement across diverse object types. We present extensive evaluations showing that VolPD produces high-fidelity volumetric LOD animations that preserve high-frequency

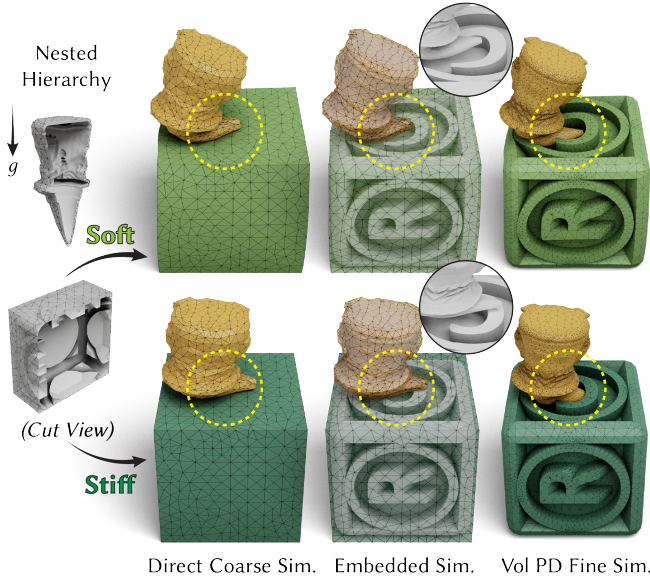


Fig. 2. **Contact Geometry Resolution and Conformity:** We show that VolPD effectively handles contact resolution and geometric conformity for both (Top) soft and (Bottom) stiff materials, overcoming inaccurate contact handling of embedded simulations that can result in unnatural gaps. (Left) Cut-away view of the embedded hierarchy, showing only the boundary surface mesh of the underlying tetrahedral meshes.

detail across resolutions and remain robust in challenging examples with high-speed motion, large deformations, and frictional contact across 1D, 2D, and 3D. Our main technical contributions include the following.

- A key integrator-design insight that motivates *VelPro Splitter*, which separates coarse and fine velocity content and recombines low- and high-frequency components with substantially improved high-frequency dynamics for both shells and volumes and largely decouples refinement quality from the number of levels used.
- A practical volumetric hierarchy and prolongation pipeline that builds multiresolution tetrahedral meshes and defines linear prolongation operators across overlapping, non-conforming LOD meshes via a topology-aware, boundary-binding strategy for barycentric extrapolation.
- A cheap stiffness-matching strategy adapted from prior work [Chen et al. 2017] and *newly* integrated into the VolPD framework to mitigate numerical stiffening and improve cross-level LOD consistency.¹

2 Related Work

2.1 Progressive Simulation

Recent progressive simulation work [Zhang et al. 2023, 2022, 2024, 2025] enables rapid iterative design for high-fidelity cloth and shell

¹By cross-level consistency, we mean that each coarser-level timestep reproduces the bulk deformation of the corresponding finer-level timestep, while finer levels add only high-frequency details without changing the coarse-scale behavior as defined in [Zhang et al. 2025].

simulation by producing predictive coarse previews that progressively refine to the finest-level results in a coarse-to-fine LOD workflow. It has been shown to be effective for both quasistatic simulation [Zhang et al. 2023, 2022] and dynamic animation [Zhang et al. 2024], but remains largely confined to codimension-1 cloth and shell domains.

Volumetric simulation is central to computer animation, yet remains a key missing capability of Progressive Dynamics. We fill this gap by extending the framework to LOD volumetric animation while retaining the core PD pipeline, enabling unified shell–volume co-simulation and substantially broadening the range of physics-based scenarios the Progressive Dynamics framework can support.

2.2 Adaptive, Reduced, and Embedded Simulation

Adaptive and subspace methods accelerate elastodynamics with contact by reducing degrees of freedom or operating in low-dimensional bases [Capell et al. 2002b; Ferguson et al. 2023; Grinspun et al. 2002; Kim and James 2009; Teng et al. 2015; Zheng and James 2011]. While effective in certain regimes, reduced models can limit deformation expressiveness, fine-scale contact handling can become a bottleneck [Trusty et al. 2024], and these approaches are not designed around preview-and-refine workflows or consistent behavior across changing resolutions.

Meanwhile, embedded simulation reduces cost by simulating a coarse domain and interpolating motion to high-resolution geometry [Capell et al. 2002a; Faloutsos et al. 1997; McAdams et al. 2011; Molino et al. 2004; Müller and Gross 2004; Rivers and James 2007; Zhu et al. 2010]. Linear interpolation, while efficient, often introduces artifacts such as deformation-gradient discontinuities, faceting, self-intersections, and inconsistent contact; smoother or higher-order schemes mitigate some visual issues [DeRose et al. 1998; James 2020; James and Pai 2003] but remain limited by the coarse simulation signal. Related skinning-style interpolation and coordinate constructions can provide robust transfer across hierarchies [Brandt et al. 2018; Jacobson et al. 2012; Ruan et al. 2024; Xian et al. 2019], yet without injecting fine-scale physical information they still struggle to reproduce contact-driven nonlinearities and high-frequency dynamics.

2.3 Multiresolution Volumetric Mesh Hierarchy

Multiresolution volumetric hierarchies have been widely studied for efficient storage, processing, and simulation. Examples include adaptive hierarchical tetrahedral decompositions for physics-based refinement [Debunne et al. 2001], progressive tetrahedral representations based on collapse/split operations [Molino et al. 2003; Staadt and Gross 1998], and grid-based adaptive schemes such as octrees for fluid simulation [Losasso et al. 2004]. The progressive tetrahedral representations extend the idea of progressive surface meshes [Hoppe 1996] to volumetric domains. More recently, learning-based methods have also exploited hierarchical volumetric structure for reconstruction [Gao et al. 2020; Wang et al. 2017]. Despite this breadth, constructing scalable volumetric hierarchies that balance geometric fidelity, numerical robustness, and LOD consistency for simulation remains application-specific and challenging.

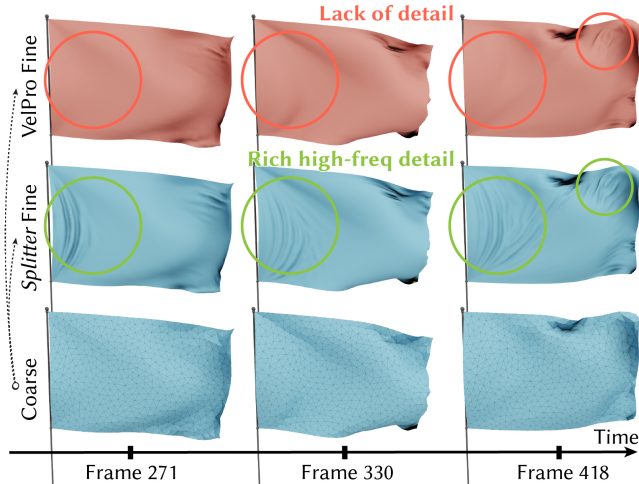


Fig. 3. **Waving Flag:** In this highly oscillatory flag example, when the fine-level dynamics are not coarse-representable, VelPro overly constrains the fine motion toward the coarse level and the result lacks dynamic detail. In contrast, *VelPro Splitter* captures rich, rapidly evolving wrinkling.

2.4 Volumetric Mapping and Multigrid Prolongation

In the shell setting, top-down mesh hierarchies built via recursive edge-collapse decimation can explicitly track bijective patch correspondences [Aksoylu et al. 2005; Garland and Heckbert 1997; Lee et al. 1998; Liu et al. 2021], which can be composed into prolongation operators that respect curved geometry and preserve rest shape [Zhang et al. 2023]. For volumetric meshes, in contrast, existing tetrahedral decimation and simplification methods [Cignoni et al. 2000; Danovaro et al. 2002; Renze and Oliver 1996; Staadt and Gross 1998; Trotts et al. 1999] typically do not provide analogous local mappings. Even if we can produce multiple tetrahedral levels that approximate the same domain, the levels may have non-conforming boundaries and only partially overlapping or mismatched interior regions. In this case, prolongation is no longer a clean composition of tracked correspondences and can become ill-posed without additional structure. On the other hand, volumetric parameterizations could provide cross-resolution correspondences, but robust, general-purpose bijective 3D mapping remains difficult. Optimization-based approaches [Aigerman and Lipman 2013; Kovalsky et al. 2014] are typically nonlinear and computationally expensive, while combinatorial methods [Campen et al. 2016; Cherchi and Livesu 2023; Hinderink et al. 2024; Hinderink and Campen 2023] offer stronger guarantees but often require restrictive topologies such as ball-like or star-shaped domains, limiting applicability to general simulation meshes.

Our volumetric prolongation operator is most closely related to transfer operators in multigrid volumetric solvers (see [Shao et al. 2022]). For efficiency, multigrid often exploits regular grids [Bolz et al. 2003; Trottenberg et al. 2001]. In animation, Zhu et al. [2010] used trilinear interpolation on uniform 3D grids for embedded elasticity, and McAdams et al. [2011] reduced instability near coarse-domain boundaries via gradient-based prolongation. Recent multigrid methods for deformable models in graphics leverage

skinning-style coordinates to define prolongation by interpolating coarse information to fine scales over grid hierarchies [Brandt et al. 2018; Jacobson et al. 2012; Ruan et al. 2024; Xian et al. 2019]. For general unstructured problems, algebraic multigrid can infer transfer operators automatically [Brandt 1986]. Closest to our setting are nonnested tetrahedral FEM transfers [Georgii and Westermann 2006; Kharevych et al. 2009], which use barycentric interpolation for nodes inside coarse elements and use extrapolation otherwise, but practical and robust algorithms for performing this extrapolation have not been detailed. For prolongation definition in our VolPD setting, we show that naive extrapolation can induce binding-related artifacts, propose a topology-aware boundary-binding remedy (Figure 9) with bounded extrapolation effects to enable a barycentric-like linear prolongation operator.

2.5 Volumetric Coarsening and Homogenization

A complementary direction seeks coarse models that reproduce fine-scale mechanical behavior through fitted effective parameters, bases, or element-wise response. Dynamics-Aware Coarsening (DAC) [Chen et al. 2017] calibrates a coarsened elastic model to better match target dynamic response and contact-rich motion for fabrication-oriented design. EigenFit [Chen et al. 2019] further targets cross-resolution consistency by matching leading deformation modes across meshes, correcting the low-energy spectral content that is most sensitive to numerical stiffening under coarsening. It pays off in linear elasticity, but the benefit largely *disappears* for nonlinear elasticity, which is our target setting. Related numerical coarsening and homogenization approaches compute coarse representations that preserve the effective behavior of heterogeneous materials, including numerical coarsening of inhomogeneous linear elasticity [Kharevych et al. 2009] and optimized coarsening via discontinuous, matrix-valued shape functions for heterogeneous and nonlinear models [Chen et al. 2018]. Although these coarsening and homogenization techniques could potentially further improve VolPD’s coarsest-level previews by making them more predictive of fine-scale material response and cross-level consistency, we find that a simple Young’s-modulus rescaling based on eigenanalysis, as in DAC [Chen et al. 2017], is well suited to apply and sufficient to mitigate the inconsistencies that can arise from VelPro-style integration in the VolPD setting.

2.6 Multiscale Splitting for Integration and Velocity Updates

The construction of our newly proposed *VelPro Splitter* relates closely to fluid-animation super-resolution methods that explicitly mix motion across scales by replacing the bulk (low-frequency) component of a high-resolution velocity with a guided or coarse signal while retaining fine-scale detail. Forootaninia et al. [Forootaninia and Narain 2020] perform this band-splitting operation in the Fourier domain through convolution with a Gaussian kernel. More recent wavelet-domain variants [Lyu et al. 2024] achieve a similar effect via band-local coefficient replacement with improved spatial locality. Variational coupling methods [Nielsen and Christensen 2010; Nielsen et al. 2009] guide smoke by optimizing a coupling objective between simulations at different resolutions, aiming to preserve

high-frequency turbulence while enforcing low-frequency agreement with a guide. Beyond fluids, these “replace low band, keep high band” constructions mirror classical multiscale decompositions in numerical PDEs. This includes variational multiscale (VMS) formulations that split a field into resolved and unresolved components via projection operators [Hughes et al. 1998, 2000], as well as multigrid intuition in which coarse spaces capture smooth, low-frequency content while complementary components represent oscillatory, high-frequency detail [Brandt 1977; Trottenberg et al. 2001]. IMEX time integration splits the dynamics into implicit and explicit parts, often separating stiff from nonstiff contributions [Ascher et al. 1997, 1995]. *VelPro Splitter* brings this spirit into the progressive integrator design by preserving the fine-only residual and replacing only the coarse-representable component. In practice, beyond being particularly effective at enforcing cross-level consistency while maintaining the fine-scale volumetric dynamics essential for contact and wave propagation, it also decouples the refinement effect from the number of levels used in the mesh hierarchy.

3 Background and Core Issues

We aim for predictive coarse-resolution previews that support rapid iterative design while remaining progressively refinable into a final high-resolution volumetric elastodynamics animation with consistency across a hierarchy of models and systematically improving fidelity at each level. Progressive Dynamics [Zhang et al. 2024, 2025] achieves this objective by introducing three key ingredients:

- *A mesh LOD hierarchy with prolongation operators* $P_{l+1}^l(\cdot)$, built during hierarchy construction, that transfer state quantities from a coarse mesh M_l to the next finer mesh M_{l+1} [Zhang et al. 2023].
- *Progressive integrator design*, which keeps coarse previews predictive and allows progressive refinement to the final high-resolution result as geometry is enriched, while maintaining stable timestepping and temporal continuity at each level [Zhang et al. 2024, 2025].
- *Subspace proxy energies*, which enable high-quality coarse previews for cloth and shells by alleviating membrane-locking artifacts [Zhang et al. 2023, 2022].

Before explaining how we extend each ingredient to the volumetric setting, we briefly situate our work in the broader Progressive Simulation context by reviewing the shared Progressive Dynamics scaffold and the core issues that prevent a straightforward adaptation of prior shell-based methods.

3.1 Hierarchy and Energy Construction

Our objective is to construct a hierarchy of volumetric meshes at multiple levels of detail. Similar to the shell case, we index the hierarchy by resolution with $l \in \{0, 1, \dots, L\}$, from finest to coarsest as $M_L, M_{L-1}, \dots, M_0$. At level l , we denote the undeformed (rest) vertex positions by $\bar{x}_l \in \mathbb{R}^{dn_l}$ and the deformed positions by $x_l \in \mathbb{R}^{dn_l}$, where n_l is the number of vertices at level l and $d \in \{1, 2, 3\}$ is the spatial dimension. Given this hierarchy, we also need to correspondingly and properly define a prolongation operator $P_{l+1}^l(\cdot)$ that maps state quantities from a coarse mesh M_l to the next finer mesh M_{l+1} [Zhang et al. 2023]. In the shell setting, hierarchical edge-collapse decimation can track local bijective patch maps, which can

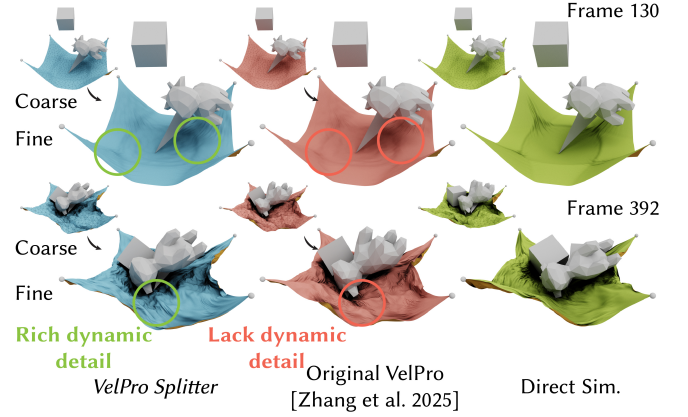


Fig. 4. High-Frequency Dynamic Enrichment vs. Geometric Enrichment: We show a cloth example in which several stiff shell objects are dropped at high speed onto an extremely thin cloth ($d = 0.07$ mm). Thanks to *VelPro Splitter*'s ability to enrich high-frequency dynamics, the fine-level animation exhibits clear wave propagation starting at impact and continuing as traveling waves with high-frequency dynamic details. Qualitatively, these dynamics are closer to the high-frequency behavior produced by direct fine-scale simulation. In contrast, the wrinkles produced across frames by the original VelPro method [Zhang et al. 2025] are relatively static, suggesting that they primarily reflect fine-scale geometric detail rather than dynamically evolving high-frequency motion. Please see our supplemental video for the full motion.

then be composed to induce a geometry-aware prolongation operator. For volumetric meshes, however, existing tetrahedral decimation methods do not naturally provide analogous correspondences, so this coupled hierarchy-and-prolongation strategy does not directly apply.

For the FE-based simulation at each level l , we adopt the IPC formulation [Li et al. 2020] for volumetric elastodynamics, summing elasticity Ψ_l , contact B_l , and friction D_l contributions in a total potential energy E_l , with Ψ_l given by standard volumetric materials and B_l, D_l given by IPC barrier and friction terms. Using the variational incremental potential formulation [Kane et al. 1999; Li et al. 2020], the level- l update is computed as

$$x_l^{t+1} = \underset{x}{\operatorname{argmin}} \frac{1}{2h^2} \|x - \hat{x}_l^t\|_{M_l}^2 + E_l(x), \quad (1)$$

where h is the timestep size, M_l is the mass matrix (with the notation $\|x\|_{M_l}^2 := x^\top M_l x$), and E_l is the total potential energy, also incorporating contributions from body and external forces. Here, direct simulation uses $E_l = \Psi_l + B_l + D_l$ and $\hat{x}_l^t = x_l^t + hv_l^t$, the standard implicit-Euler prediction, where v_l^t is the current-step velocity in a direct single-level simulation. However, prior Progressive Dynamics work further relied on a subspace proxy energy, via an elastic energy parameterization $\Psi_L(P^l(x_l))$, to mitigate membrane locking and produce artifact-free coarse previews. In volumetric elastodynamics, it is unclear whether this helps the dominant failure modes, e.g., numerical stiffening. Hence, we leave the definitions of $\hat{x}_l^t$ and E_l abstract for now.

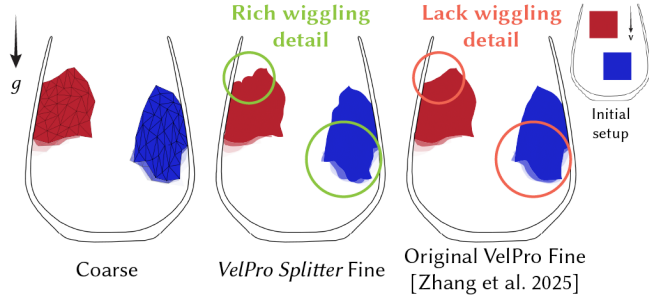


Fig. 5. **High-Speed Soft-Cube Collision:** Two soft cubes collide at high speed. *VelPro Splitter* recovers rich high-frequency detail, while VelPro [Zhang et al. 2025] is overly constrained by the coarse motion and shows little refinement.

3.2 Progressive Advancement and Integration

We next outline the multilevel solver and Zhang et al. [2024; 2025]’s resolution-time integration (progressive advancement). The goal is to fill a space-time multiresolution grid by assigning positions x_l^t and velocities v_l^t to each state (t, l) , for $t \in \{0, \dots, N\}$ and $l \in \{0, \dots, L\}$. Progressive Dynamics advances *horizontally* by stepping from t to $t + 1$ at fixed l , and *vertically* by refining from l to $l + 1$.

Independent per-level simulation via Equation (1) can yield diverging trajectories, so coarse results fail to preview fine behavior. To address this, Zhang et al. [2024] introduce a velocity update that couples refinement with forward dynamics to enable frame-by-frame consistent shell previews. At the coarsest level ($l = 0$), we solve Equation (1) over the full time horizon using $\hat{x}_0^t = x_0^t + h v_0^t$ with $v_0^t = (x_0^t - x_0^{t-1})/h$, producing (x_0^t, v_0^t) for all $t \in \{0, \dots, N\}$ as the base row of the solution grid.

For finer levels $l > 0$, velocity updates become a key design choice, trading off stability, temporal continuity, cross-level consistency, and geometric enrichment (see [Zhang et al. 2025] for detailed definitions). We systematize these tradeoffs by casting Progressive Dynamics as a family of integrators parameterized through

$$\hat{x}_{l+1}^t = \bar{x}_{l+1}^t + h \bar{v}_{l+1}^t, \quad (2)$$

where $\bar{x}_{l+1}^t$ and $\bar{v}_{l+1}^t$ are free design choices that serve as approximations of the converged state (x_{l+1}^t, v_{l+1}^t) at time t and level $l + 1$ in the resolution-time grid. Zhang et al. [2025] particularly advocated the *VelPro* integrator as the most effective option so far, balancing these goals through a prolonged diagonal velocity update at grid points $(t + 1, l + 1)$ of $\hat{x}_{l+1}^t = x_{l+1}^t + h (V_{l+1}^l(x_l^t))v_l^t$, where $V_{l+1}^l(x) = \nabla P_{l+1}^l(x)$ is the velocity prolongator and $v_l^t = (x_l^t - x_l^{t-1})/h$. It is worth noting that in the shell setting, $P_{l+1}^l(\cdot)$ is generally nonlinear by construction (see Equation 7 in [Zhang et al. 2023]), and $V_{l+1}^l(\cdot)$ denotes its Jacobian-based linearization. This operator is likewise fixed by the rest shape, sparse through successive self-parameterization, and precomputable. Finally, timestep advancement for finer levels at grid points $(t + 1, l + 1)$ is achieved by solving the progressive IP problem like Equation (1).

3.2.1 Limitations of Applying VelPro for Volumetric Elastodynamics. However, while the VelPro integrator effectively preserves stability,

temporal continuity, and per-frame geometric consistency across levels in Progressive Dynamics, its design also introduces limitations that are undesirable not only for shells, but also in the volumetric setting. Concretely, when constructing the predicted position $\hat{x}_{l+1}^t$ via Equation (2), VelPro advances from the previous fine-level position through $\bar{x}_{l+1}^t = x_{l+1}^t$, but approximates the fine-level velocity $\bar{v}_{l+1}^t$ with the coarse-prolonged value $(V_{l+1}^l(x_l^t))v_l^t$. While this consistent velocity update is key to enforcing cross-level consistency, it relies on a coarse-level constrained velocity. As a result, it can strongly suppress the high-frequency dynamics that should emerge at fine resolution, because each timestep effectively removes the fine-level high-frequency components encoded in the previous-timestep velocity. Moreover, inspecting the incremental potential in Equation (1), we see that the inertia term $\frac{1}{2h^2} \|x - \hat{x}_l^t\|_{M_l}^2$ becomes increasingly dominant relative to $E_l(x)$ as h decreases. Since $E_l(x)$ effectively acts as a relaxation term, the minimizer becomes more biased toward $\hat{x}_l^t$ for smaller h .² Because VelPro modifies only the inertia term via $\hat{x}_l^t$, this bias shifts toward the coarse prolonged-velocity prediction as h decreases, further amplifying the over-constraining effect and making attainable finest-level enrichment depend on the number of levels (and potentially impractical for sufficiently small h , since many levels may be required [Zhang et al. 2025]).

As a result, fine-level motion can appear overly constrained by the coarse level with limited dynamic enrichment, especially with large resolution gaps between levels, which greatly reduces the benefit of costly fine-resolution solves. While geometric refinement remains important as in the shell case (e.g., rest-shape surface details reappearing under deformation as the hierarchy refines), volumetric elastodynamics often requires finer resolution to resolve high-frequency *dynamics* such as contact transients, impacts, fast waves propagating through an object, and secondary jiggles, which coarse discretizations fundamentally cannot represent.

4 Volumetric Progressive Dynamics

With the framework laid out and the challenges identified, we now dive into the construction of each component and address these issues to enable Volumetric Progressive Dynamics.

4.1 Resolution-Splitting VelPro Integrator

Motivated by these observations, we propose **VelPro Splitter**, a projection-based, resolution-splitting variant of the VelPro integrator that retains stability and temporal continuity at every level from VelPro while substantially increasing high-frequency *dynamic* enrichment. The key idea is to decompose the fine-level velocity into a coarse-representable (low-resolution) component and a fine-only (high-resolution) residual, and then recombine the high-resolution component with coarse-level prolonged velocity so that (i) the low-frequency motion remains consistent with the coarse simulation, while (ii) the fine-level high-frequency content is retained from the fine-level velocity at the previous timestep.

To enable this resolution splitting, we first construct a projection operator onto the coarse subspace that is representable at the coarse level. Let P_{l+1}^l denote the prolongation map from coarse-level

²A compact way to see this is $h \downarrow \Rightarrow \frac{1}{2h^2} \|x - \hat{x}_l^t\|_{M_l}^2 \uparrow$ over $E_l(x) \Rightarrow x_{l+1}^t$ biased toward $\hat{x}_l^t$.

DOFs at level l to fine-level DOFs at level $l + 1$. Unlike the shell setting in prior work [Zhang et al. 2023], where $P_{l+1}^l(\cdot)$ is generally nonlinear and its Jacobian defines the corresponding linearized prolongation operator, our proposed volumetric prolongation $P_{l+1}^l(\cdot)$ is itself linear. Therefore, its Jacobian is constant, and we simply write

$$V_{l+1}^l = \nabla P_{l+1}^l = P_{l+1}^l, \quad (3)$$

where, for brevity, we use V_{l+1}^l to denote its matrix form hereafter, and defer its specific construction in the volumetric setting to Section 4.3. Given any fine-level vector u on level $l + 1$, we define its best coarse-representable approximation on level l by least squares as $V_{l+1}^l w^*$, via the projection

$$w^* := \underset{w}{\operatorname{argmin}} \|V_{l+1}^l w - u\|_{M_{l+1}}^2 = ((V_{l+1}^l)^\top M_{l+1} V_{l+1}^l)^{-1} (V_{l+1}^l)^\top M_{l+1} u,$$

with M_{l+1} being the mass matrix at level $l + 1$. This induces a splitting operator at level $l + 1$ that extracts the low-resolution component,

$$\Pi_{l+1} := V_{l+1}^l ((V_{l+1}^l)^\top M_{l+1} V_{l+1}^l)^{-1} (V_{l+1}^l)^\top M_{l+1}, \quad (4)$$

and its complement $I - \Pi_{l+1}$ that extracts the high-resolution (detail) component, where I is the identity matrix. Given the fine-level velocity v_{l+1}^t , we decompose it into a coarse-representable part and a fine-only residual,

$$v_{l+1}^t = \Pi_{l+1} v_{l+1}^t + (I - \Pi_{l+1}) v_{l+1}^t. \quad (5)$$

We then replace only the low-resolution component with the prolonged coarse velocity while preserving the high-pass detail. This yields the *VelPro Splitter* predicted velocity

$$\hat{v}_{l+1}^t := \underbrace{V_{l+1}^l v_l^t}_{\text{coarse-consistent low-res component}} + \underbrace{(I - \Pi_{l+1}) v_{l+1}^t}_{\text{preserved high-res component}}. \quad (6)$$

Following the prediction form in Equations (1) and (2), we set $\tilde{x}_{l+1}^t = x_{l+1}^t$ and obtain

$$\hat{x}_{l+1}^t = x_{l+1}^t + h \hat{v}_{l+1}^t = x_{l+1}^t + h V_{l+1}^l v_l^t + h (I - \Pi_{l+1}) v_{l+1}^t,$$

defining our *VelPro Splitter* integrator.

Intuitively, the *VelPro Splitter* integrator offers three benefits. First, *VelPro Splitter* leverages a projection operator defined (in a least-squares sense) from the forward prolongation operator between adjacent levels. Projecting the fine-level velocity onto the coarser resolution's subspace, concretely defines coarser-scale “bulk motions” (for the first time in this line of work), towards which finer-level results should be consistent. *VelPro Splitter* then modifies just this coarse-representable component of the fine update, while preserving the fine-only residual. In this way, *VelPro Splitter* retains the coarse level's predicted displacement without sacrificing the richer fine-scale displacements that motivate running the fine simulation.

Second, *VelPro Splitter* retains VelPro's stable resolution-time integration structure by integrating from the previous-step position, $\tilde{x}_{l+1}^t = x_{l+1}^t$ (see [Zhang et al. 2025] for details). This choice is necessary but not sufficient: stability also depends on how the fine-level velocity is split and recombined. An arbitrary splitting operator could corrupt the coarse-representable motion or inject energy and so instability. We therefore use the same projection operator Π_{l+1} to induce a natural mass-weighted orthogonal split,

where the coarse-consistent low-resolution component $V_{l+1}^l v_l^t$ is orthogonal to the retained high-resolution residual $(I - \Pi_{l+1}) v_{l+1}^t$ (see Corollary 1 in Appendix A). This cleanly separates coarse-consistent and fine-residual contributions, which promotes stability while also preventing fine-scale dynamics from being overwritten by coarser levels.

Third, by preserving the high-resolution component explicitly, *VelPro Splitter* reduces the undesirable dependence of enrichment on the number of LOD levels and avoids an otherwise *strong* reliance on nonlinear solves to “recreate” missing detail. With this property, users can jump directly from the coarsest to the finest level and directly obtain rich detail at the finest level, without sequentially advancing through intermediate levels in the hierarchy. This improvement provides a practical advantage over previous progressive integrators: it can reduce computational cost by avoiding unnecessary intermediate-level solves and allowing users to evaluate the final-resolution result directly. To qualitatively and quantitatively validate these benefits, we use a 1D pogo stick (see Figure 6) benchmark that probes *VelPro Splitter*'s ability to recover high-frequency dynamics, showing enrichment comparable to a direct high-resolution simulation. Using the same setup, we further evaluate *VelPro Splitter*'s energy stability, geometric consistency, and temporal continuity. We apply the consistency and continuity metrics proposed by [Zhang et al. 2025] to validate that *VelPro Splitter* remains comparable to the original VelPro; see Figures 7 and 8. To demonstrate the generality of *VelPro Splitter*, we extend it beyond implicit Euler to other time integrators (e.g., explicit Euler and BDF2); see Appendix B for detailed derivations.

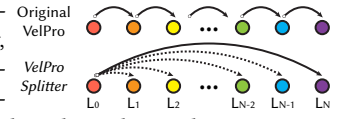
4.2 Volumetric Hierarchy

As outlined in Section 3.1, we now return to constructing a hierarchy of volumetric meshes that represent the same object in 2D or 3D at different LOD. Our approach proceeds in two stages. Given an arbitrarily detailed high-resolution surface geometry S_L , we first decimate it to obtain a surface hierarchy $S_L, S_{L-1}, \dots, S_0$ using off-the-shelf surface decimation, e.g., [Garland and Heckbert 1997]. We then tetrahedralize each S_l using robust tetrahedralization tools [Hu et al. 2020, 2018; Si 2015] to produce the volumetric hierarchy $M_L, M_{L-1}, \dots, M_0$. In our implementation, we use TetWild [Hu et al. 2018] for improved element quality.

Although this provides a principled construction that approximates the same object across LOD as faithfully as possible, it is worth noting that our framework is also well suited and works surprisingly well with almost any reasonable user-defined hierarchy that shares an interior but may have wildly different or even misaligned surfaces. This is largely because we define prolongation independently of tracked decimation correspondences (see Section 4.3 for details).

4.3 Prolongation of Volumetric Data

With the volumetric hierarchy in place, a key remaining question is how to construct prolongation operators that map quantities between non-conforming tetrahedral meshes across levels. As in the shell case, we seek a preferably linear operator P_{l+1}^l that maps from



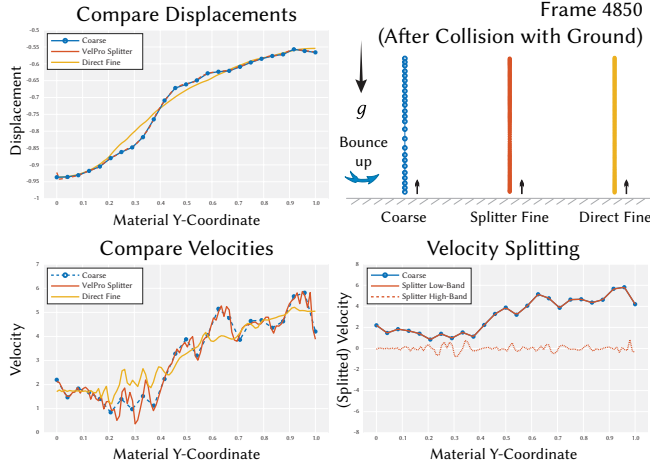
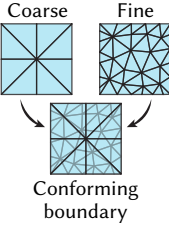


Fig. 6. **Pogo Stick:** To more clearly visualize *VelPro Splitter*'s splitting and enrichment, we simulate a 1D pogo stick (motion constrained to the y direction) dropped under gravity, using a simple stable neo-Hookean material model [Smith et al. 2018] and an IPC-like barrier energy [Li et al. 2020] for ground contact. Across the 1D plots, the x -axis denotes the normalized material y -coordinate of the pogo stick in its rest configuration, ordered from the bottom/contact end to the top end. In the top-right plot (left to right), we show VolPD's coarse result (blue), VolPD's fine result (red), and a direct fine simulation (yellow). From the displacement curves (top-left), *VelPro Splitter* maintains strong coarse-to-fine consistency. The velocity split (bottom-right) shows the fine-level velocity decomposed into low- and high-resolution components, where the low-resolution component is consistent with the coarse velocity (the bulk motion to track). Finally, comparing fine-level velocities (bottom-left), *VelPro Splitter* not only tracks the coarse motion but also recovers higher-frequency details that mimic the direct fine simulation.

$\mathcal{M}_l$ to $\mathcal{M}_{l+1}$. We begin with the simplest two-level setting with conforming boundaries, where aligned meshes admit a straightforward barycentric-coordinate construction [Schneider 2017]. However, this assumption is too restrictive for the complex geometries encountered in practice, where enforcing conforming boundaries is impractical and inefficient. We therefore propose a practical extrapolation algorithm for constructing prolongation operators on general shapes with non-conforming boundaries.



4.3.1 Non-Conforming Boundary Extrapolation. To handle non-conforming boundaries, we need extrapolation, which in turn requires binding. We therefore introduce a simple, topology-aware binding method that enables our volumetric prolongation construction. Concretely, we bind each fine-mesh vertex to a coarse element so barycentric coordinates can be computed as in the conforming case: vertices inside a coarse element are bound directly, while vertices outside are bound to a nearby coarse element and extrapolated using possibly negative barycentric coordinates. These bindings define the prolongation matrix, which may contain negative entries but otherwise behaves identically to the conforming construction. Naive binding methods, such as closest-point binding in Euclidean

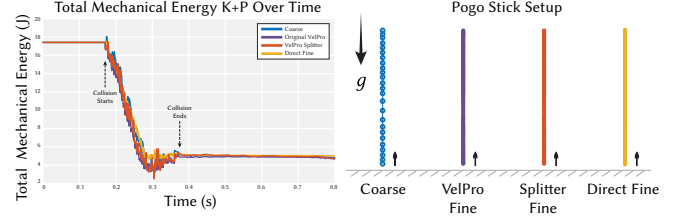


Fig. 7. **Stable and Bounded Energy Evolution:** Left: We report the total mechanical energy $K + P$ for the same 1D pogo-stick example over time as in Figure 6, where K is kinetic energy and P includes elastic and gravitational potentials but excludes the contact barrier energy, so that the plot reflects the physical mechanical energy of the pogo stick. Compared with direct simulations at both coarse and fine resolutions, as well as the original VelPro, *VelPro Splitter* exhibits similarly stable and bounded energy behavior throughout, both before and after collision with the ground, without noticeable long-term drift or instability. Right: We show the corresponding 1D pogo-stick setup used for this energy analysis.

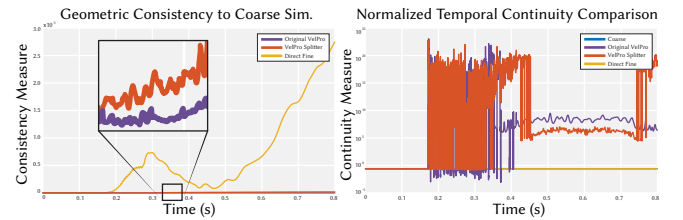


Fig. 8. **Consistency and Continuity Measures:** For the same setup as in Figure 7, we evaluate (left) geometric consistency with respect to the coarse reference simulation using the measure from Zhang et al. [2025]. The fine-level animation results produced by VelPro and *VelPro Splitter* generally remain consistent with the coarse reference, whereas direct fine simulation does not, as expected. The inset further shows that *VelPro Splitter* produces a slightly higher consistency error than the original VelPro. We also evaluate (right) temporal continuity using the continuity measure from Zhang et al. [2025]. The results show that *VelPro Splitter* achieves similarly continuous temporal behavior to VelPro, whose visual continuity was demonstrated in [Zhang et al. 2025].

distance [Georgii and Westermann 2006], ignore topology, so fine vertices can be bound to coarse elements that are nearby in Euclidean space but far in geodesic distance, causing topologically inconsistent bindings and severe artifacts. To avoid this (Figure 9), our algorithm enforces a simple principle: neighboring fine vertices should not map to geodesically distant coarse elements. We therefore leverage local neighborhood information to infer bindings while remaining topology-aware.

4.3.2 Prolongation Construction. Consider a two-level hierarchy with a coarse mesh $\mathcal{M}_C = (\mathcal{V}_C, \mathcal{T}_C)$ and a fine mesh $\mathcal{M}_F = (\mathcal{V}_F, \mathcal{T}_F)$ representing the same shape, with overlapping interiors but not necessarily-conforming boundaries. Our goal is to bind each fine vertex in $\mathcal{V}_F$ to a home coarse element in $\mathcal{T}_C$, from which we then compute barycentric coordinates (or extrapolated barycentric coordinates if needed). We first precompute vertex-vertex adjacency on $\mathcal{M}_F$ and element-element adjacency on $\mathcal{M}_C$. For each fine vertex,

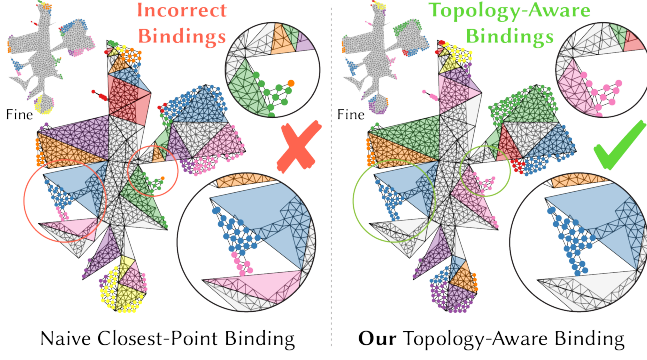
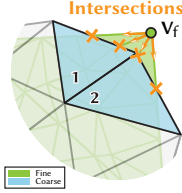


Fig. 9. **Compare Bindings:** We visualize bindings between fine-mesh vertices and coarse-mesh elements. Dots with the same color indicate fine vertices bound to the same coarse element. A naïve closest-point strategy can bind fine vertices to coarse elements that are close in Euclidean distance but far geodesically, introducing simulation artifacts. In contrast, our topology-aware binding avoids these mismatches effectively.

if it already lies inside a coarse element, we bind it directly to that element. This direct-containment case is the primary assumption in all of our examples. Fine vertices that lie outside the coarse mesh are marked as unassigned and processed next.

We then iteratively bind the remaining unassigned fine vertices. At each iteration, we prioritize *one-ring-neighbor consistency*: we select the unassigned vertex with the largest number of already-assigned fine-mesh one-ring neighbors, so each inferred assignment is supported by the strongest immediate local consensus. For the selected vertex v_f , we construct a candidate set of coarse elements as follows: (i) for each incident fine edge segment of v_f , we test that segment against coarse boundary facets and add the adjacent coarse elements of intersected boundary facets; and (ii) we add the home elements already assigned to one-ring neighbors of v_f . After duplicate removal, we restrict the search to this candidate subset. We then run a point-to-mesh distance query for v_f using only this subset (instead of the full coarse mesh), obtain the nearest point on the subset, and bind v_f to the coarse element containing that nearest point. This selected element is used as the home element. Once the home element is determined, we compute the corresponding barycentric coordinates of v_f with respect to that element. Each new assignment updates the one-ring-neighbor support counts of the remaining unassigned vertices, and we repeat this process until every vertex in $\mathcal{V}_f$ is mapped to an element in $\mathcal{T}_c$. Please refer to Algorithm 1 for the pseudocode.

Although we describe the procedure as binding fine vertices to coarse elements, the same binding routine can be applied between any selected pair of levels in the hierarchy, with the fine and coarse roles interchangeable. The extension from a two-level hierarchy to an L -level hierarchy ($L > 2$) is therefore straightforward: depending on the integration path, we simply apply the same procedure successively between the selected level pairs.



4.4 Stiffness Rescaling for Coarse Models

Lastly, prior Progressive Dynamics work for cloth and shells [Zhang et al. 2024, 2025] relied on a subspace proxy energy to mitigate membrane locking and produce high-quality coarse previews by reparameterizing the elasticity term as $\Psi_L(P^l(x_l))$, where P^l prolongs the level- l state to the finest level. In volumetric elastodynamics, membrane locking is absent, but significant discretization artifacts remain, most notably *numerical stiffening*. For instance, volumetric FEM with linear elements can exhibit shear locking, making coarse discretizations overly stiff compared to finer ones. This matters for VelPro-type progressive integrators, including *VelPro Splitter*, since they modify only the velocity prediction; when the system comes to rest ($v_l^t = v_{l+1}^t = 0$), the update reduces to a per-level quasistatic solve with $\hat{x}_{l+1}^t = x_{l+1}^t$, leaving any stiffness mismatch uncorrected and potentially causing cross-level inconsistencies.

To alleviate this, we observe that it is straightforward to adapt the much cheaper Young’s-modulus rescaling of the coarse model from [Chen et al. 2017] and integrate it into coarse-level simulations. This better matches the primary mode across resolutions and mitigates numerical stiffening. Let λ_l^0 denote the principal eigenvalue of the stiffness matrix at rest, evaluated with Young’s modulus E_L at the finest level L , where by principal eigenvalue we mean the smallest nonzero eigenvalue capturing the bulk deformation mode used for cross-resolution stiffness matching. To match the finest-level stiffness response, we set

$$E_l := \frac{\lambda_l^0}{\lambda_l^0} E_L. \quad (7)$$

This inexpensive rescaling improves cross-level consistency without the overhead of fine-scale proxy-energy evaluations, and it proves effective for our practical volumetric examples, especially when the primary deformation mode dominates (e.g., an unconstrained shape settling under gravity). Meanwhile, we acknowledge that it may not suffice for more complex deformations and boundary conditions. We leave nonlinear rescaling, principled subspace design, and a broader study of the tradeoffs to future work, with preliminary results in Section 6.

5 Evaluation

We implement the full VolPD framework in C++, using Apple’s Accelerate framework for linear solves, Eigen for the remaining linear-algebra routines [Guennebaud et al. 2010], and Spectra for robust eigenvalue computations [Qiu 2024]. We report example statistics and timings on an Apple MacBook Pro with an M4 Max chip and 128 GB of RAM. Given *VelPro Splitter*’s strong ability to decouple enrichment quality from the number of levels used (unlike prior progressive integrators that rely heavily on deep hierarchies), we use a two-level hierarchy (coarse and fine) for comparisons throughout the paper unless stated otherwise. In this section, we first run ablations on VolPD’s core components in Section 5.1, and then evaluate VolPD as a full system for producing high-quality LOD animations under a range of conditions in Sections 5.2, 5.3.

5.1 Ablations

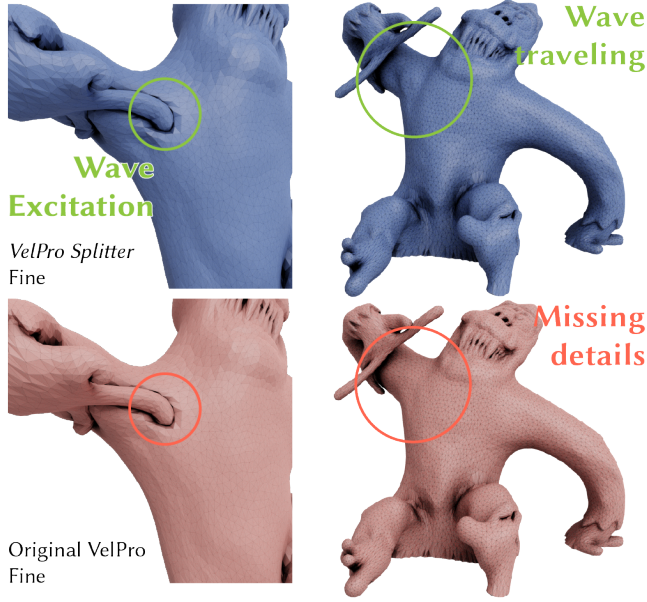


Fig. 10. **Wave Excitation:** In this high-speed, large-deformation setup, a soft, squishy gorilla deforms and self-collides as a fingertip nudges its upper body, triggering a wave excitation. The fine-level animation with *VelPro Splitter* captures the resulting traveling wave, whereas the original VelPro [Zhang et al. 2025] completely misses it.

VelPro Splitter vs. VelPro. To demonstrate *VelPro Splitter*'s ability to improve high-frequency dynamic details over the original VelPro method while remaining stable, we conduct extensive validations across challenging 1D, 2D, and 3D scenarios involving high speeds, large deformations, and frictional contact in both shell and volumetric settings. As one stability check, the inset figure reports the total energy over time for a simple bouncy-ball example using implicit Euler, showing that the energy remains bounded in this case.

For shells, we first run a waving-flag example using a two-level hierarchy with $\Delta t = 0.01$ s and a constant wind force. Since the cloth motion is highly oscillatory with lots of fine-scale wrinkles jumping around, it is extremely hard for a coarse level to represent these finer scales, and VelPro tends to miss those high-frequency details. In contrast, *VelPro Splitter* captures these dynamic details well, showing a distinct difference even with only a two-level hierarchy (Figure 3). As a comparison, with VelPro, even increasing the hierarchy to 4 or even 10 levels still misses some detailed wrinkles that a two-level *VelPro Splitter* can capture (see Figure 17 and our supplemental video for details). Note that the two-level Splitter enriches only frequencies above the coarse resolution, so medium-frequency content that falls within the coarse-representable band is inherited from the coarse simulation rather than generated independently. In the 10-level VelPro, intermediate levels can independently excite those mid-range modes. This is a direct consequence of the two-level design,

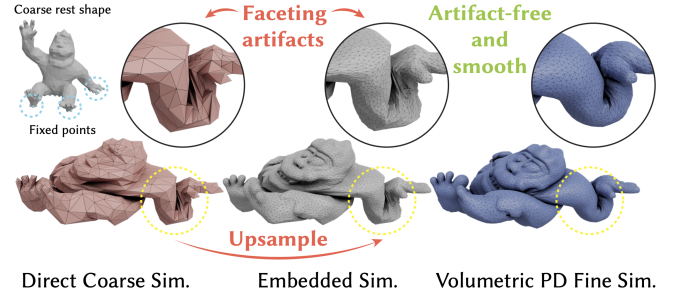
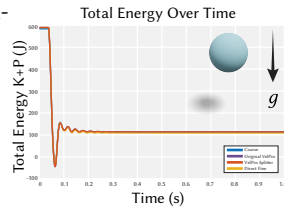


Fig. 11. **Comparison with Embedded Simulation:** (Left) A coarse direct simulation used for (Middle) embedded deformation of a fine render mesh introduces visible “faceting” artifacts. (Right) In contrast, progressive refinement of the coarse simulation using VolPD produces artifact-free smooth deformations (see text for discussion).

not a deficiency: when intermediate levels are added for preview, those frequencies appear as well.

For volumetric elastodynamics, we include two 2D examples with high speed and large deformation. We first show soft deformable cubes with Young’s modulus $E = 1.5 \times 10^4$ and Poisson’s ratio $\nu = 0.3$ using a Neo-Hookean material model, colliding with a rigid cube collider. Fast waves travel through the object with a small timestep size $\Delta t = 0.002$ s. Compared to VelPro, which damps refinement-scale details substantially, *VelPro Splitter* captures these waves and augments them with Neo-Hookean-induced bulges (see Figure 5). Meanwhile, we also include a *Sharkey* character dropped onto the ground, where the object has many thin and detailed structures. Under large deformation and after collision, *VelPro Splitter* captures the natural bends and rotations of these detailed parts while waves travel through it. However, VelPro yields overly constrained motion due to its coarse counterpart, with too little refinement in those parts motion-wise (see Figure 12).

Orthogonal Split. For the same *Sharkey* example, we also conduct an ablation study of *VelPro Splitter* with respect to the choice of projection operator $\Pi_l(\cdot)$ at level l , in contrast to Equation (4). As a comparison, we include a variant that instead applies an alternative Laplacian-based smoothing to define $\Pi_l(\cdot)$, specifically $\Pi_l(\mathbf{v}) = (I_l - \lambda L_l)\mathbf{v}$, where $\mathbf{v}$ denotes the velocity field to be decomposed, I_l is the identity matrix, L_l is the cotangent Laplacian at level l , and λ is a scalar controlling the smoothing strength. This smoothing operator is used only in ablation, in place of our proposed projection operator, to separate the low- and high-resolution components of the velocity field. However, it does not satisfy the same orthogonality property as the projection operator defined algebraically in Section 4.1. Because of this, it is difficult to hand-tune λ so that the projection $\Pi_l(\mathbf{v})$ itself does not alter coarse-level bulk motion. In contrast, the projection-based Splitter preserves coarse-level consistency while still enabling strong high-frequency refinement (see Figure 13).

Time Integrators. Going beyond implicit Euler, we also show that *VelPro Splitter* generalizes well to other time integrators, e.g., BDF2, which is higher-order and can support larger timesteps while still capturing high-frequency motion (Appendix B). To demonstrate this, we first run a cloth example using an extremely thin material

with thickness $d = 0.07$ mm, with three stiff shell objects dropped onto it using $\Delta t = 0.001$ s with a large downward initial speed $v_0 = 10$ m/s. This setup is very good at showing detailed waves traveling through the cloth from the moment the wedge collides with it. While *VelPro Splitter* captures these high-frequency wave-traveling effects, *VelPro* mainly adds geometric enrichment with less pronounced dynamic wrinkling (see Figure 4).

Meanwhile, for volumes, we begin with a simple setup that generates high-speed large deformation: a single gorilla character simulated with a hierarchy using timestep size $\Delta t = 0.002$ s and a relatively soft material ($E = 2 \times 10^4$ Pa). We enforce fixed Dirichlet conditions on the bottom, one hand, and both feet. At the initial frame, we apply a strong upward impulse that stretches the character to an extreme configuration, inducing rapid oscillatory motion. Around Frame 300, the gorilla's fingertip collides with its arm, which induces a subtle wave excitation that travels through its body. Despite being subtle, *VelPro Splitter* captures it while *VelPro* misses it (see Figure 10 and our supplemental video for details).

Stiffness Rescaling. To evaluate stiffness rescaling for mitigating numerical stiffening, we show two setups, one in 2D and one in 3D. As shown in Figure 15, we simulate a 2D cantilever bar bending under gravity, pinned at one end, with a large resolution gap between coarse ($\#V = 33$) and fine ($\#V = 2036$). For the coarse-level simulation, we do a three-way comparison among: (i) direct coarse simulation using the same Young's modulus and physical parameters as the finest level, (ii) a subspace model parameterized by $\Psi_L(P^l(x_l))$ where P^l is the linear prolongation operator (as in [Zhang et al. 2024, 2025]), and (iii) direct coarse simulation with rescaled Young's modulus. It is clear that *VelPro Splitter* helps keep the frequency of the periodic bending motion more consistent. However, *VelPro*-type integrators act only on the velocity update; therefore, once the fine-level solve reaches equilibrium, e.g., $v = 0$ and $a = 0$, numerical stiffening can still lead to inconsistent equilibria. In this setup, this shows up as a mismatch in the lowest point reached by the bar tip under (i), which breaks frame-by-frame consistency. It is also interesting that the subspace reparameterization in (ii) does not help much in this volumetric case. On the contrary, direct coarse simulation with rescaled E (here $0.59\times$) matches the fine equilibrium much better.

Besides this, we also run a 3D twisted-bar example where both ends are fixed and then twisted by a prescribed 300° boundary motion. We compare identical vs. rescaled Young's modulus in direct coarse simulation (factor $0.93\times$ computed by Equation (7)). After the bar reaches equilibrium, the rescaled coarse simulation better matches the fine-level equilibrium than the unscaled coarse simulation (see Figure 16).

5.2 Comparisons

Direct Simulation. For the beach ball example as in Section 5.3, we also run direct simulations at both coarse and fine resolutions

using the same underlying IPC simulator [Li et al. 2020] integrated into our framework. We observe that direct simulation can diverge

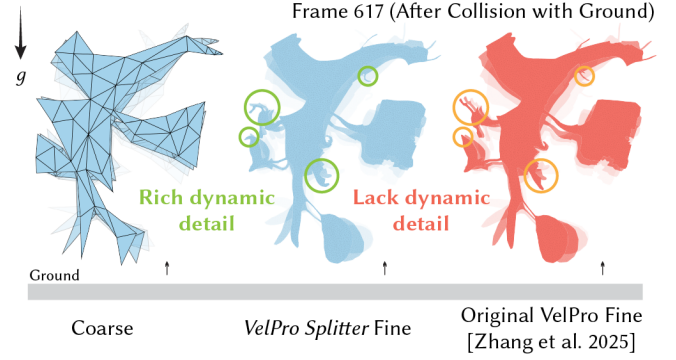
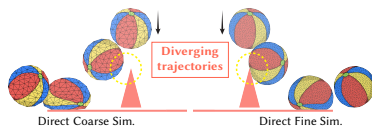


Fig. 12. **Jumping Sharkey:** In this jumping *Sharkey* example with many thin, detailed structures, *VelPro* overly constrains the fine motion toward the coarse level, leading to heavily damped motion. In contrast, *VelPro Splitter* captures the detailed bending and rotation of these parts.

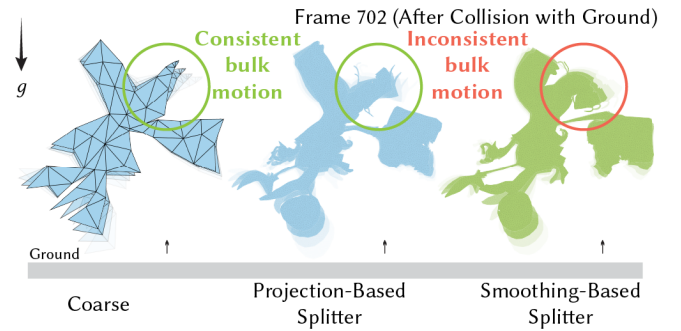


Fig. 13. **Projection-Based Splitter vs. Smoothing-Based Splitter:** To demonstrate the effectiveness and necessity of an orthogonal split when defining the splitting operator for separating low- and high-resolution components, we ablate our projection-based Splitter against a smoothing-based alternative. When used with *VelPro Splitter*, the smoothing-based Splitter does not guarantee an orthogonal split, which can lead to bulk-motion inconsistency relative to the coarse level. In contrast, the projection-based Splitter remains consistent throughout.

substantially across resolutions. The inset figure and our supplemental video highlight visible discrepancies in shape and trajectory between different resolutions that emerge after a short period of simulated time. Despite identical setups, including the same material parameters and initial conditions, these IPC results can exhibit markedly different material behavior and motion. This resolution sensitivity is a known limitation of direct simulation methods and is not unique to high-fidelity finite element schemes [Zhang et al. 2024, 2025]. In contrast, VolPD yields qualitatively consistent animations across levels, with progressively enhanced visual fidelity at higher resolutions. Our figures and videos provide frame-by-frame evidence of consistency in both global motion and fine geometric detail.

TRACKS-Style. To further evaluate VolPD's benefits for producing rich and consistent LOD enrichment, we compare it against a

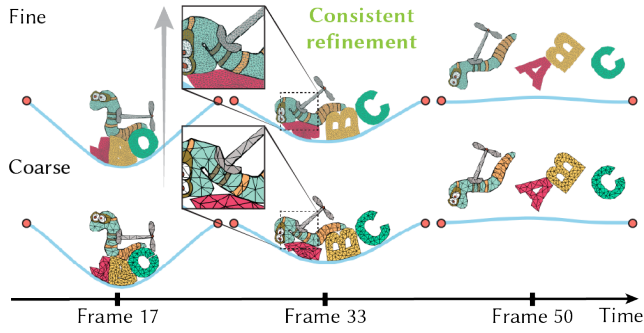


Fig. 14. **Heterogeneous materials:** We show that progressive volumetric dynamics supports heterogeneous materials by simulating a scene that simultaneously involves Neo-Hookean, St. Venant–Kirchhoff, and corotational elastic materials of varying stiffnesses (see text for details).

prolonged-target penalty baseline inspired by the general animation-tracking setting of TRACKS [Bergou et al. 2007]. Here our baseline tests a penalty based TRACKS-style approach in which the fine-level volumetric simulation is penalized (in contrast to TRACK’s hard-constraints) to track the prolonged coarse-level motion.

In our experiment (Figure 18), we use a slit-array object and construct a two-level hierarchy with timestep size $\Delta t = 0.01$ s. For the TRACKS-style baseline, the fine-level simulation is constrained to follow the prolonged coarse result via a per-vertex penalty $w_l \|x - P_l^{l-1} x_{l-1}^{t+1}\|_{M_l}^2$, where P_l^{l-1} denotes the prolongation matrix. We find that the effectiveness of this prolonged-target penalty baseline is highly sensitive to the choice of the penalty weight w_l . When w_l is not carefully tuned, the fine-level simulation can diverge from the coarse solution, leading to ghost-force artifacts and degraded motion quality. In the same two-level setup, VolPD does not rely on penalty terms to enforce cross-resolution coherence. It naturally produces consistent fine-level motion, with increasing geometric fidelity and stable behavior as resolution improves. See Figure 18 and our supplemental video for details.

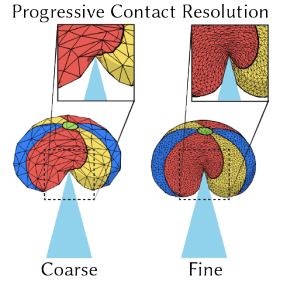
Embedded Simulation. As shown in Figure 11, running a coarse simulation followed by direct embedding often leads to faceting artifacts, where large planar deformations persist even at fine resolutions. This issue reflects a broader limitation of commonly used interpolants in embedded simulation, whether linear, higher-order, or skinning-based, which lack physical awareness and fail to capture fine-scale motion or resolve contact accurately. Our VolPD framework addresses this by treating prolongation as a modular component, with VolPD serving as a physical relaxation stage that enriches embedded results with detailed dynamics and improved contact handling.

5.3 Benchmark Examples

To demonstrate the effectiveness of VolPD, we evaluate it on a range of benchmark examples with varying complexity. Unless otherwise specified, we use barycentric coordinates with our boundary extrapolation introduced in Section 4.3. Overall, VolPD produces high-fidelity, LOD-consistent animations across resolutions, even

under challenging dynamics such as high speeds, large deformations, and frictional contact. We refer to our supplemental video for the full set of animations.

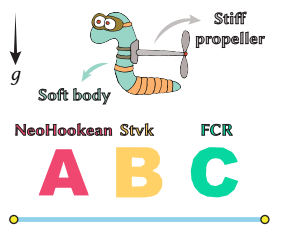
Sharp Spike Contact. As shown in the inset figure, we examine a 2D scenario in which a soft beach ball is dropped onto a sharp spike-shaped collider. At the coarser level, the limited degrees of freedom cannot closely conform to the spike indentation, resulting in an oversimplified deformation. In contrast, finer resolutions resolve boundary geometry that conforms more closely to the collider, while preserving the overall timing of the event. This example illustrates how VolPD supports progressively refined contact interactions with consistent global motion.



Extreme Coarse Shapes. To evaluate VolPD’s support of extremely coarse inputs for fast previewing, Figures 12 and 13 present two 2D examples where the coarsest levels are aggressively decimated. The coarse shapes differ significantly from the fine-level geometry and capture only rough polyhedral approximations of the fine contours. Despite this large discrepancy, VolPD successfully introduces additional physical and geometric details at each level while preserving coherence of motion throughout the animations.

Extreme Geometry Differences Across Levels. Beyond the 2D cases above, Figure 2 demonstrates a challenging 3D test with extreme coarse-to-fine mismatch in the collider geometry. We force the coarse block to be a simple cube, while the finest level includes deep letter engravings that change collision dynamics. We test both a soft and a stiff block. In both cases, the fine-level simulation adapts smoothly to the detailed surface without locking, drifting, or violating motion consistency, highlighting VolPD’s resilience to geometric discrepancies.

Heterogeneous Materials and Multi-Object Interactions. Beyond single-object simulations with homogeneous materials, Figure 14 demonstrates VolPD’s capability in handling heterogeneous materials and complex multi-object interactions. We assign materials to elements at the coarse level and propagate those assignments to finer levels, then define each level’s elastic potential accordingly. As objects fall, collide, and bounce, VolPD maintains consistent global behavior across resolutions. Deformations and contact interactions become increasingly detailed with refinement. See the accompanying video for the full animation.



Timings. Since our only modification in VolPD’s per-level solve is the velocity integration step, each level runs at speeds comparable to direct simulation performed at that resolution. As with Progressive Dynamics for shells [Zhang et al. 2024, 2025], scalability is realized through fast coarse-level previews that accelerate iterative design.

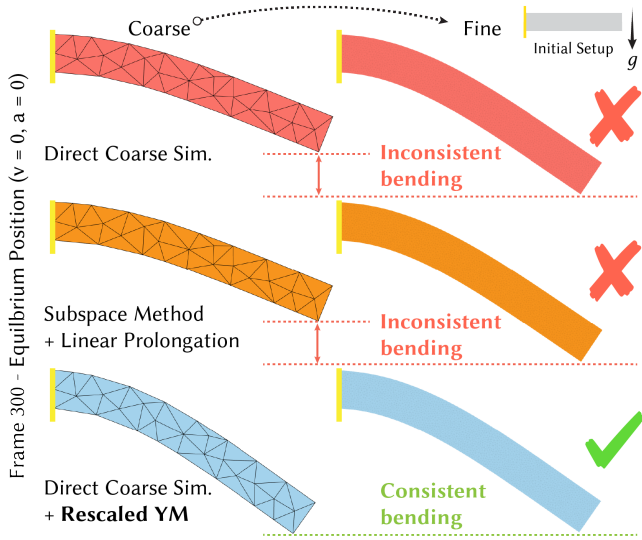


Fig. 15. **Bendy Cantilever Bar:** To mitigate numerical stiffening, which can cause cross-level inconsistency when VolPD reaches equilibrium ($v = 0$, $a = 0$), we run a three-way comparison of coarse-level model choices on a simple cantilever bar that bends under gravity with one end fixed to a wall. With direct coarse simulation using the same physical parameters as the fine model, and with the shell-style subspace energy based on linear prolongation [Zhang et al. 2024, 2025], numerical stiffening leads to inconsistent bending across levels. In contrast, our Young’s-modulus rescaling adapted from [Chen et al. 2017] is well suited for mitigating this issue in VolPD.

Across our examples, we consistently observe one to two orders of magnitude of speedup when using coarse preview simulations compared to direct fine simulation. In one extreme case (the teaser example), we achieve roughly a 120 $\times$ speedup, with the fine simulation taking nearly 20 hours and the coarse preview completing in about 10 minutes. Moreover, because *VelPro Splitter* largely decouples enrichment quality from the number of levels in the VolPD hierarchy, our framework typically pays only the coarsest- and finest-level costs and can skip intermediate levels. For example, on the waving-flag scene, a two-level setup is about 3 $\times$ faster overall than a ten-level hierarchy.

Jello in the Bowl. We simulate a set of soft, jello-like animal characters thrown into a glass bowl at very high speed using a four-level hierarchy. At the coarsest level, each character is represented by a simple blobby approximation for fast previewing. As VolPD refines through the hierarchy, it progressively recovers both geometric detail (surface features) and high-frequency dynamics (fine deformation, waves, and contact), while maintaining consistent overall motion across levels. See Figure 1 and the supplemental video for details.

6 Conclusion and Limitations

We extend Progressive Dynamics [Zhang et al. 2024, 2025] to volumetric finite elements, enabling efficient LOD animation design with predictive coarse previews. Our core contribution is *VelPro Splitter*, which recovers important high-frequency dynamic detail

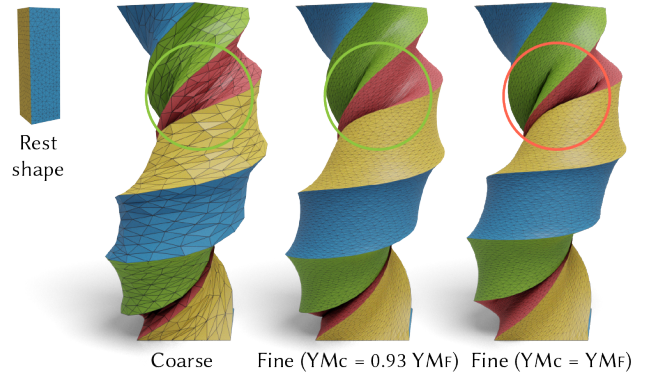


Fig. 16. **Twisted Bar:** We show that rescaling Young’s modulus for direct coarse-level simulation (middle) improves cross-level consistency at equilibrium compared to the unscaled coarse simulation (right), relative to the coarse-simulation reference (left).

that VelPro and other progressive integrators miss. We also integrate two supporting contributions: a topology-aware binding method for volumetric prolongation, and an adapted Young’s-modulus rescaling. Together, these enable VolPD.

Limitations and Future Work. Despite these advances, there remain several limitations and opportunities to further improve both the efficiency and quality of progressive volumetric simulation. For example, as shown in Figure 2, although VolPD generally handles geometric discrepancies across levels well, we can still construct extreme cases—such as using a hollow cube as the fine mesh, but a solid cube for the coarse mesh where the resulting fine animation remains consistent with the coarse preview but fails to fall into the cube (see inset) and so does not produce physically plausible behavior. Most of our implementation is CPU-based; however, most parts of the algorithm would benefit from GPU acceleration.

More importantly, while our adaptive stiffness rescaling helps mitigate numerical stiffening in volumetric elastodynamics, exciting opportunities remain for nonlinear rescaling and more principled subspace proxy-model design. As a preliminary step in this direction, we experimented with an analogous subspace parameterization for volumetric dynamics using biharmonic coordinates [Wang et al. 2015], which act as a higher-order interpolant. Our early results (see inset and the supplemental video) suggest that such proxy-energy constructions have strong potential to further alleviate numerical stiffening and make coarse models more predictive of fine-scale behavior. Important trade-offs remain in efficiency versus effectiveness, as well as robustness issues such as inverted-element handling across different material models. We leave these directions for future work.





Fig. 17. **Increasing Levels:** In the same waving-flag example as in Figure 3, even when increasing the number of levels in the VelPro hierarchy to 4 and 10, respectively, the finest-level animation still fails to recover some fine-scale wrinkling details, compared to a much cheaper two-level *VelPro Splitter*.

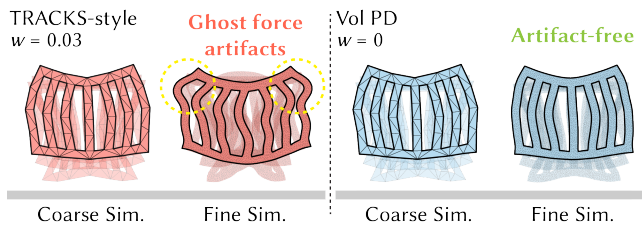


Fig. 18. **Comparison with TRACKS-style position-based tracking:** (Left) TRACKS-style control can produce undesirable deformation artifacts due to the coarse moment-based control forces, whereas (Right) VolPD estimates a fine simulation that closely resembles the coarse preview simulation.

Acknowledgments

Jiayi Eris Zhang is supported by the Stanford Graduate Fellowship, the Roblox Graduate Fellowship, and the NVIDIA Graduate Fellowship. Doug James acknowledges support from Adobe, SideFX, NVIDIA, Activision, and the Department of Energy, National Nuclear Security Administration under Award Number DE-NA0003968; Houdini software courtesy of SideFX.

References

- Noam Aigerman and Yaron Lipman. 2013. Injective and bounded distortion mappings in 3D. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–14.
- Burak Aksoylu, Andrei Khodakovskiy, and Peter Schröder. 2005. Multilevel solvers for unstructured surface meshes. *SIAM Journal on Scientific Computing* 26, 4 (2005), 1146–1165.
- Uri M. Ascher, Steven J. Ruuth, and Raymond J. Spiteri. 1997. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics* 25, 2–3 (1997), 151–167.
- Uri M. Ascher, Steven J. Ruuth, and Brian TR Wetton. 1995. Implicit-explicit methods for time-dependent partial differential equations. *SIAM J. Numer. Anal.* 32, 3 (1995), 797–823.
- Miklós Bergou, Saurabh Mathur, Max Wardetzky, and Eitan Grinspun. 2007. TRACKS: Toward Directable Thin Shells. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 50–es.
- Jeff Bolz, Ian Farmer, Eitan Grinspun, and Peter Schröder. 2003. Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. In *ACM SIGGRAPH 2003 Papers (San Diego, California) (SIGGRAPH '03)*. Association for Computing Machinery, New York, NY, USA, 917–924. doi:10.1145/1201775.882364
- Achi Brandt. 1977. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.* 31, 138 (1977), 333–390. doi:10.1090/S0025-5718-1977-0431719-X
- Achi Brandt. 1986. Algebraic multigrid theory: The symmetric case. *Appl. Math. Comput.* 19, 1–4 (1986), 23–56.
- Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-reduced projective dynamics. *ACM Trans. Graph.* 37, 4, Article 80 (July 2018), 13 pages. doi:10.1145/3197517.3201387

- Marcel Campen, Cláudio T Silva, and Denis Zorin. 2016. Bijective maps from simplicial foliations. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–15.
- Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. 2002a. Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph.* 21, 3 (July 2002), 586–593. doi:10.1145/566654.566622
- Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. 2002b. A multiresolution framework for dynamic deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 41–47.
- Desai Chen, David IW Levin, Wojciech Matusik, and Danny M Kaufman. 2017. Dynamics-aware numerical coarsening for fabrication design. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–15.
- Jiong Chen, Hujun Bao, Tianyu Wang, Mathieu Desbrun, and Jin Huang. 2018. Numerical coarsening using discontinuous shape functions. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.
- Yu Ju Chen, David IW Levin, Danny Kaufman, Uri Ascher, and Dinesh K Pai. 2019. EigenFit for consistent elastodynamic simulation across mesh resolution. In *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 1–13.
- Gianmarco Cherchi and Marco Livesu. 2023. VOLMAP: a Large Scale Benchmark for Volume Mappings to Simple Base Domains. In *Computer Graphics Forum*, Vol. 42. Wiley Online Library, e14915.
- Paolo Cignoni, D Costanza, Claudio Montani, Claudio Rocchini, and Roberto Scopigno. 2000. Simplification of tetrahedral meshes with accurate error evaluation. In *Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145)*. IEEE, 85–92.
- Emanuele Danovaro, Leila De Florian, Michael Lee, and Hanan Samet. 2002. Multiresolution tetrahedral meshes: an analysis and a comparison. In *Proceedings SMI. Shape Modeling International 2002*. IEEE, 83–273.
- Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. 2001. Dynamic Real-Time Deformations using Space and Time Adaptive Sampling. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2001)*. 31–36. doi:10.1145/383259.383262
- Tony DeRose, Michael Kass, and Tien Truong. 1998. Subdivision surfaces in character animation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, 85–94.
- P. Faloutsos, M. van de Panne, and D. Terzopoulos. 1997. Dynamic Free-Form Deformations for Animation Synthesis. *IEEE Trans. on Vis. and Comput. Graph.* 3, 3 (1997), 201–214.
- Zachary Ferguson, Teseo Schneider, Danny M Kaufman, and Daniele Panozzo. 2023. In-Timestep Remeshing for Contacting Elastodynamics. *ACM Trans. Graph.* 42, 4 (2023), 145–1.
- Zahra Forootaninia and Rahul Narain. 2020. Frequency-domain smoke guiding. *ACM Trans. Graph.* 39, 6 (2020), 172:1–172:10. doi:10.1145/3414685.3417842
- Jun Gao, Wenzheng Chen, Tommy Xiang, Clement Fuji Tsang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2020. Learning Deformable Tetrahedral Meshes for 3D Reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*. https://arxiv.org/abs/2011.01437
- Michael Garland and Paul S Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 209–216.
- Joachim Georgii and Rüdiger Westermann. 2006. A multigrid framework for real-time simulation of deformable bodies. *Comput. Graph.* 30, 3 (June 2006), 408–415. doi:10.1016/j.cag.2006.02.016
- Eitan Grinspun, Petr Krysl, and Peter Schröder. 2002. CHARMS: A simple framework for adaptive simulation. *ACM transactions on graphics (TOG)* 21, 3 (2002), 281–290.
- Gael Guennebaud, Benoît Jacob, et al. 2010. Eigen v3.
- Steffen Hinderink, Hendrik Brückler, and Marcel Campen. 2024. Bijective volumetric mapping via star decomposition. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–11.
- Steffen Hinderink and Marcel Campen. 2023. Galaxy maps: Localized foliations for bijective volumetric mapping. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–16.
- Hugues Hoppe. 1996. Progressive Meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. Association for Computing Machinery, New York, NY, USA, 99–108.
- Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast tetrahedral meshing in the wild. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 117–1.
- Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral meshing in the wild. *ACM Trans. Graph.* 37, 4 (2018), 60.
- Thomas J. R. Hughes, Guido R. Feijó, Luca Mazzei, and Jean-Baptiste Quinicy. 1998. The variational multiscale method—a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering* 166, 1–2 (1998), 3–24. doi:10.1016/S0045-7825(98)00079-6
- Thomas J. R. Hughes, Luca Mazzei, and Kenneth E. Jansen. 2000. Large eddy simulation and the variational multiscale method. *Computing and Visualization in Science* 3 (2000), 47–59. doi:10.1007/s007910050051

- Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. 2012. Fast automatic skinning transformations. *ACM Trans. Graph.* 31, 4, Article 77 (July 2012), 10 pages. doi:10.1145/2185520.2185573
- Doug L. James. 2020. Phong Deformation: A Better C0 Interpolant for Embedded Deformation. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 126:1–126:12. doi:10.1145/3386569.3392411
- Doug L. James and Dinesh K. Pai. 2003. Multiresolution Green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Transactions on Graphics (TOG)* 22, 1 (2003), 47–82.
- Conor Kane, Eduardo A. Repetto, Michael Ortiz, and Jerrold E. Marsden. 1999. Finite element analysis of nonsmooth contact. *CMAME* 180, 1-2 (1999).
- Lily Kharevych, Patrick Mullen, Houman Owahdi, and Mathieu Desbrun. 2009. Numerical coarsening of inhomogeneous elastic materials. *ACM Transactions on graphics (TOG)* 28, 3 (2009), 1–8.
- Theodore Kim and Doug L. James. 2009. Skipping steps in deformable simulation with online model reduction. *ACM Transactions on Graphics (TOG)* 28, 5 (2009), 1–9.
- Shahar Z. Kovalsky, Noam Aigerman, Ronen Basri, and Yaron Lipman. 2014. Controlling singular values with semidefinite programming. *ACM Trans. Graph.* 33, 4 (2014), 68–1.
- Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence C. Cowsar, and David P. Dobkin. 1998. MAPS: Multiresolution Adaptive Parameterization of Surfaces. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998, Orlando, FL, USA, July 19-24, 1998*, Steve Cunningham, Walt Bransford, and Michael F. Cohen (Eds.). ACM, 95–104. doi:10.1145/280814.280828
- Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. 2020. Incremental Potential Contact: Intersection- and Inversion-free Large Deformation Dynamics. *ACM Trans. Graph. (SIGGRAPH)* 39, 4, Article 49 (2020).
- Hsueh-Ti Derek Liu, Jiayi Eris Zhang, Mirela Ben-Chen, and Alec Jacobson. 2021. Surface Multigrid via Intrinsic Prolongation. *ACM Trans. Graph.* 40, 4, Article 80 (jul 2021), 13 pages.
- Frank Losasso, Frédéric Gibou, and Ron Fedkiw. 2004. Simulating water and smoke with an octree data structure. In *ACM SIGGRAPH 2004 Papers* (Los Angeles, California) (SIGGRAPH '04). Association for Computing Machinery, New York, NY, USA, 457–462. doi:10.1145/1186562.1015745
- Luan Lyu, Xiaohua Ren, Enhua Wu, and Zhi-Xin Yang. 2024. Fast Wavelet-domain Smoke Guiding. In *Pacific Graphics Conference Papers and Posters*, Renjie Chen, Tobias Ritschel, and Emily Whiting (Eds.). The Eurographics Association. doi:10.2312/pg.20241299
- Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011. Efficient elasticity for character skinning with contact and collisions. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) (SIGGRAPH '11). Association for Computing Machinery, New York, NY, USA, Article 37, 12 pages. doi:10.1145/1964921.1964932
- Neil Molino, Zhaosheng Bao, and Ron Fedkiw. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Transactions on Graphics* 23, 3 (Aug. 2004), 385–392.
- Neil Molino, Robert Bridson, and Ronald Fedkiw. 2003. Tetrahedral Mesh Generation for Deformable Bodies. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 25–35. <https://graphics.stanford.edu/papers/meshing-sig03/>
- M. Müller and M. Gross. 2004. Interactive Virtual Materials. In *Graph. Interface*. 239–246.
- Michael B. Nielsen and Brian B. Christensen. 2010. Improved variational guiding of smoke animations. *Computer Graphics Forum* (2010). Eurographics.
- Michael B. Nielsen, Brian B. Christensen, Nafees Bin Zafar, Doug Roble, and Ken Museth. 2009. Guiding of smoke animations through variational coupling of simulations at different grid resolutions. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*. 217–226.
- Yixuan Qiu. 2024. Spectra: A C++ Header-Only Library for Large Scale Eigenvalue Problems. <https://spectralib.org/>. Version 1.0.1.
- Kevin J Renze and James H Oliver. 1996. Generalized unstructured decimation [computer graphics]. *IEEE Computer Graphics and Applications* 16, 6 (1996), 24–32.
- Alec R. Rivers and Doug L. James. 2007. FastLSM: fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph.* 26, 3 (July 2007), 82–es. doi:10.1145/1276377.1276480 Place: New York, NY, USA Publisher: Association for Computing Machinery.
- Liangwang Ruan, Bin Wang, Tiantian Liu, and Baoquan Chen. 2024. MiNNE: a Mixed Multigrid Method for Real-time Simulation of Nonlinear Near-Incompressible Elastics. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–15.
- Teseo Schneider. 2017. *Theory and Applications of Bijective Barycentric Mappings*. Ph.D. Dissertation. Università della Svizzera italiana.
- Han Shao, Libo Huang, and Dominik L. Michels. 2022. A fast unsmoothed aggregation algebraic multigrid framework for the large-scale simulation of incompressible flow. *ACM Trans. Graph.* 41, 4, Article 49 (July 2022), 18 pages. doi:10.1145/3528223.3530109
- Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2, Article 11 (Feb. 2015), 36 pages. doi:10.1145/2629697
- Breannan Smith, Fernando De Goes, and Theodore Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Trans. Graph.* 37, 2, Article 12 (March 2018), 15 pages. doi:10.1145/3180491
- Oliver G. Staadt and Markus H. Gross. 1998. Progressive Tetrahedralizations. In *Proceedings of the Conference on Visualization '98*. IEEE Computer Society Press, Research Triangle Park, North Carolina, United States, 397–402. doi:10.1109/VISUAL.1998.745329
- Yun Teng, Mark Meyer, Tony DeRose, and Theodore Kim. 2015. Subspace condensation: Full space adaptivity for subspace deformations. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–9.
- Ulrich Trottenberg, Cornelis W. Oosterlee, and Anton Schüller. 2001. *Multigrid*. Academic Press.
- Issac J. Trotts, Bernd Hamann, and Kenneth I. Joy. 1999. Simplification of tetrahedral meshes with error bounds. *IEEE Transactions on Visualization and Computer Graphics* 5, 3 (1999), 224–237.
- Ty Trusty, Yun (Raymond) Fei, David I.W. Levin, and Danny M. Kaufman. 2024. Trading Spaces: Adaptive Subspace Time Integration for Contacting Elastodynamics. *ACM Transactions on Graphics* 43, 6 (2024), 1–16. doi:10.1145/3687946
- Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)* 36, 4 (2017), 1–11.
- Yu Wang, Alec Jacobson, Jernej Barbič, and Ladislav Kavan. 2015. Linear subspace design for real-time shape deformation. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.
- Zangyueyang Xian, Xin Tong, and Tiantian Liu. 2019. A Scalable Galerkin Multigrid Method for Real-time Simulation of Deformable Objects. *ACM Trans. Graph. (TOG)* 38, 6 (2019).
- Jiayi Eris Zhang, Jérémie Dumas, Yun Fei, Alec Jacobson, Doug L. James, and Danny M. Kaufman. 2023. Progressive Shell Quasistatics for Unstructured Meshes. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–17.
- Jiayi Eris Zhang, Jérémie Dumas, Yun (Raymond) Fei, Alec Jacobson, Doug L. James, and Danny M. Kaufman. 2022. Progressive Simulation for Cloth Quasistatics. *ACM Trans. Graph.* 41, 6, Article 218 (nov 2022), 16 pages. doi:10.1145/3550454.3555510
- Jiayi Eris Zhang, Doug James, and Danny M. Kaufman. 2024. Progressive Dynamics for Cloth and Shell Animation. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–18.
- Jiayi Eris Zhang, Doug James, and Danny M. Kaufman. 2025. Progressive Dynamics++: A Framework for Stable, Continuous, and Consistent Animation Across Resolution and Time. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–20.
- Changxi Zheng and Doug L. James. 2011. Toward high-quality modal contact sound. *ACM Transactions on Graphics (TOG)* 30, 4 (2011), 1–12.
- Yongning Zhu, Eftychios Sifakis, Joseph Teran, and Achi Brandt. 2010. An efficient multigrid method for the simulation of high-resolution elastic solids. *ACM Transactions on Graphics (TOG)* 29, 2 (2010), 1–18.

A Orthogonal Split

LEMMA 1. Let $M_{l+1} \in \mathbb{R}^{n \times n}$ be symmetric positive definite, and let $V_{l+1}^l \in \mathbb{R}^{n \times k}$ have full column rank. Define

$$\Pi_{l+1} := V_{l+1}^l ((V_{l+1}^l)^\top M_{l+1} V_{l+1}^l)^{-1} (V_{l+1}^l)^\top M_{l+1}.$$

Then Π_{l+1} is an M_{l+1} -orthogonal projection onto $\text{col}(V_{l+1}^l)$. In particular, the residual operator $(I - \Pi_{l+1})$ maps any vector to the M_{l+1} -orthogonal complement of $\text{col}(V_{l+1}^l)$, i.e.,

$$(V_{l+1}^l)^\top M_{l+1} (I - \Pi_{l+1}) = 0, \quad (8)$$

and consequently, for any $u \in \mathbb{R}^n$ and any $w \in \mathbb{R}^k$,

$$\langle V_{l+1}^l w, (I - \Pi_{l+1})u \rangle_{M_{l+1}} = 0, \quad (9)$$

where $\langle a, b \rangle_{M_{l+1}} := a^\top M_{l+1} b$.

PROOF. For brevity, we write $V := V_{l+1}^l$, $M := M_{l+1}$, $A := (V^\top M V)^{-1}$, and $\Pi := V A V^\top M$. Observe that Π is a projection:

$$\Pi^2 = V A V^\top M V A V^\top M = V A (V^\top M V) A V^\top M = V A V^\top M = \Pi,$$

and $\text{range}(\Pi) = \text{col}(V)$ since $\Pi u \in \text{col}(V)$ and $\Pi(Vw) = Vw$. To see the orthogonality, we compute

$$V^\top M (I - \Pi) = V^\top M - V^\top M V (V^\top M V)^{-1} V^\top M = 0,$$

which is (8). Therefore, any r of the form $r = (I - \Pi)u$ satisfies $V^T M r = 0$, meaning r lies in the M -orthogonal complement of $\text{col}(V)$. Thus, for any coarse vector $Vw \in \text{col}(V)$,

$$\langle Vw, (I - \Pi)u \rangle_M = (Vw)^T M r = w^T V^T M r = 0,$$

which is (9). $\square$

COROLLARY 1 (ORTHOGONALITY OF THE VELPRO SPLITTER UPDATE). *With the notation above, the two components in*

$$\hat{v}_{l+1}^t := \underbrace{V_{l+1}^l v_l^t}_{\text{coarse-consistent low-res component}} + \underbrace{(I - \Pi_{l+1}) v_{l+1}^t}_{\text{preserved high-res component}}$$

are M_{l+1} -orthogonal:

$$\langle V_{l+1}^l v_l^t, (I - \Pi_{l+1}) v_{l+1}^t \rangle_{M_{l+1}} = 0.$$

PROOF. Apply (9) with $w = v_l^t$ and $u = v_{l+1}^t$. $\square$

B Deriving the VelPro Splitter predictor for Explicit Euler and BDF2

B.1 Deriving the VelPro Splitter predictor for Explicit Euler

For the first-order kinematic relation $\dot{x} = v$, explicit Euler (forward Euler) evaluates the velocity at the current time:

$$x^{n+1} = x^n + h v^n.$$

Thus, the “predictor” is simply the update itself,

$$\hat{x}^{n+1} := x^n + h v^n,$$

so no extrapolation is needed. Injecting the same resolution splitting used by *VelPro Splitter* amounts to replacing v_{l+1}^n by the split velocity

$$\hat{v}_{l+1}^n := V_{l+1}^l v_l^n + (I - \Pi_{l+1}) v_{l+1}^n,$$

yielding the explicit-Euler split predictor

$$\hat{x}_{l+1}^{n+1} = x_{l+1}^n + h \hat{v}_{l+1}^n.$$

This is the forward-time counterpart of the BDF1 Splitter predictor, with the same spatial orthogonal decomposition induced by Π_{l+1} (unchanged since it depends only on the mass-weighted inner product).

B.2 Deriving the VelPro Splitter predictor for BDF2

This appendix shows how we extend the *VelPro Splitter* predictor from BDF1 (backwards Euler) to BDF2.

BDF2 kinematics in position form. For the first-order kinematic relation $\dot{x} = v$, the BDF2 discretization at timestep t^{n+1} (with timestep size h) is

$$v^{n+1} = \frac{3x^{n+1} - 4x^n + x^{n-1}}{2h}. \quad (10)$$

Equivalently, solving (10) for x^{n+1} yields

$$x^{n+1} = \frac{4}{3}x^n - \frac{1}{3}x^{n-1} + \frac{2}{3}h v^{n+1}. \quad (11)$$

Thus, compared to BDF1 ($x^{n+1} = x^n + h v^{n+1}$), BDF2 replaces the “anchor” x^n by the affine combination $\frac{4}{3}x^n - \frac{1}{3}x^{n-1}$ and rescales the velocity contribution by $\frac{2}{3}$.

A BDF2 predictive velocity and position. To build the usual “predictive position” (a.k.a. $\hat{x}$) used by implicit solvers, we extrapolate v^{n+1} from previous velocities via the standard second-order explicit extrapolation

$$v^{n+1, \text{pred}} := \frac{4}{3}v^n - \frac{1}{3}v^{n-1}. \quad (12)$$

Plugging (12) into (11) gives the BDF2 predictive position

$$\begin{aligned} \hat{x}^{n+1} &:= \frac{4}{3}x^n - \frac{1}{3}x^{n-1} + \frac{2}{3}h \left(\frac{4}{3}v^n - \frac{1}{3}v^{n-1} \right) \\ &= \frac{4}{3}x^n - \frac{1}{3}x^{n-1} + \frac{2}{9}h(4v^n - v^{n-1}), \end{aligned} \quad (13)$$

which matches the implementation pattern $\hat{x} = \frac{4}{3}x^n - \frac{1}{3}x^{n-1} + \frac{2}{9}h(4v^n - v^{n-1})$.

Injecting resolution splitting (Splitter) into BDF2. Let Π_{l+1} be the mass-weighted projection onto the coarse-representable subspace (Equation (4)), and let V_{l+1}^l be the prolongation operator. For BDF2, we apply the same spatial split to the *predicted* velocity at each level. Define the BDF2-predicted velocities at coarse and fine levels by

$$v_l^{n+1, \text{pred}} := \frac{4}{3}v_l^n - \frac{1}{3}v_l^{n-1}, \quad v_{l+1}^{n+1, \text{pred}} := \frac{4}{3}v_{l+1}^n - \frac{1}{3}v_{l+1}^{n-1}. \quad (14)$$

Then the *VelPro Splitter* BDF2 predicted velocity is

$$\hat{v}_{l+1}^{n+1, \text{pred}} := \underbrace{V_{l+1}^l v_l^{n+1, \text{pred}}}_{\text{coarse-consistent low-resolution}} + \underbrace{(I - \Pi_{l+1}) v_{l+1}^{n+1, \text{pred}}}_{\text{preserved high-resolution}}, \quad (15)$$

i.e., we replace only the low-resolution component by the prolonged coarse prediction while keeping the high-resolution detail from the fine prediction.

Finally, substituting (15) into the BDF2 position relation (11) yields the *VelPro Splitter* BDF2 predictive position:

$$\hat{x}_{l+1}^{n+1} := \frac{4}{3}x_{l+1}^n - \frac{1}{3}x_{l+1}^{n-1} + \frac{2}{3}h \hat{v}_{l+1}^{n+1, \text{pred}}. \quad (16)$$

Equations (15)–(16) give the BDF2 analogue of the BDF1 Splitter predictor ($\hat{x} = x^n + h \hat{v}^{n, \text{pred}}$), with the same spatial orthogonal decomposition induced by Π_{l+1} as in Lemma 1. The orthogonality is unchanged because Π_{l+1} depends only on the mass-weighted inner product, not on the time discretization.

BDF2 incremental potential problem. The BDF2 predictor is then used to initialize the corresponding BDF2 incremental potential minimization. Importantly, we do not substitute (16) into the BDF1 objective in Equation (1). Since the BDF2 position relation in (11) has the effective update scale $\frac{2}{3}h$, the inertial term must be correspondingly changed to:

$$x_{l+1}^{n+1} = \underset{x}{\text{argmin}} \frac{9}{8h^2} \|x - \hat{x}_{l+1}^{n+1}\|_{M_{l+1}}^2 + E_{l+1}(x). \quad (17)$$

Where the inertial scaling coefficient we use for BDF2 is $\frac{9}{8h^2} = \frac{1}{2(\frac{2}{3}h)^2}$ (in place of $\frac{1}{2h^2}$ for backwards Euler) in our implementation, leaving energy terms E_{l+1} unchanged.

Algorithm 1 Fine-to-Coarse Binding for Prolongation Construction**Require:** Fine mesh $\mathcal{M}_F = (\mathcal{V}_F, \mathcal{T}_F)$, coarse mesh $\mathcal{M}_C = (\mathcal{V}_C, \mathcal{T}_C)$ **Ensure:** Home element $H(v_f) \in \mathcal{T}_C$ and barycentric coordinates $\beta(v_f)$ for each $v_f \in \mathcal{V}_F$

```

1: Precompute one-ring adjacency on  $\mathcal{M}_F$ 
2: Precompute boundary-facet adjacency on  $\mathcal{M}_C$ 
3:  $\mathcal{A} \leftarrow \emptyset, \mathcal{U} \leftarrow \emptyset$   $\triangleright$  assigned and unassigned fine vertices
4: for all  $v_f \in \mathcal{V}_F$  do
5:   if  $v_f$  lies inside some  $\tau_c \in \mathcal{T}_C$  then
6:      $H(v_f) \leftarrow \tau_c$   $\triangleright$  direct interior binding
7:      $\beta(v_f) \leftarrow \text{BARYCOORDS}(v_f, H(v_f))$ 
8:      $\mathcal{A} \leftarrow \mathcal{A} \cup \{v_f\}$ 
9:   else
10:     $\mathcal{U} \leftarrow \mathcal{U} \cup \{v_f\}$ 
11:   end if
12: end for
13: while  $\mathcal{U} \neq \emptyset$  do
14:    $v_f^* \leftarrow \arg \max_{v_f \in \mathcal{U}} s(v_f)$   $\triangleright$  most assigned one-ring neighbors
15:    $C_{\text{seg}} \leftarrow \text{SEGMENTCANDIDATES}(v_f^*)$   $\triangleright$  from segment-boundary intersections
16:    $C_{\text{nbr}} \leftarrow \text{NEIGHBORCANDIDATES}(v_f^*, H)$   $\triangleright$  from assigned one-ring neighbors
17:    $C \leftarrow \text{UNIQUE}(C_{\text{seg}} \cup C_{\text{nbr}})$   $\triangleright$  restricted search subset
18:    $H(v_f^*) \leftarrow \text{CLOSESTELEMENT}(v_f^*, C)$   $\triangleright$  nearest point over  $C$ 
19:    $\beta(v_f^*) \leftarrow \text{BARYCOORDS}(v_f^*, H(v_f^*))$ 
20:    $\mathcal{A} \leftarrow \mathcal{A} \cup \{v_f^*\}$ 
21:    $\mathcal{U} \leftarrow \mathcal{U} \setminus \{v_f^*\}$ 
22:   Update  $s(v_f)$  for affected  $v_f \in \mathcal{U}$   $\triangleright$  new one-ring support
23: end while
24: return  $H, \beta$ 

```

Implementation note. Combining (14), (15), and (16) leads to the code-friendly form

$$\hat{x}_{l+1}^{n+1} = \frac{4}{3}x_{l+1}^n - \frac{1}{3}x_{l+1}^{n-1} + \frac{2}{9}h(4\hat{v}_{l+1}^n - \hat{v}_{l+1}^{n-1}),$$

where $\hat{v}_{l+1}^k := V_{l+1}^l v_l^k + (I - \Pi_{l+1})v_{l+1}^k$ is the same Splitter velocity defined in the main text, but evaluated at times $k = n, n - 1$.

C Pseudocode

In this section, we present the pseudocode for the core algorithm for fine-to-coarse binding for prolongation construction as in Section 4.3.2. The helper subroutines are defined as follows:

- **SEGMENTCANDIDATES**(v_f): returns coarse elements adjacent to coarse boundary facets intersected by incident fine edge segments of v_f .
- **NEIGHBORCANDIDATES**(v_f, H): returns home coarse elements already assigned to one-ring neighbors of v_f .
- **CLOSESTELEMENT**(v_f, C): runs a point-to-mesh distance query restricted to candidate elements C and returns the coarse element containing the nearest point.
- **BARYCOORDS**($v_f, H(v_f)$): computes the barycentric, or extrapolated barycentric, coordinates of v_f with respect to its home element.