# Progressive Dynamics for Cloth and Shell Animation

JIAYI ERIS ZHANG, Adobe & Stanford University, USA
DOUG L. JAMES, Stanford University, USA
DANNY M. KAUFMAN, Adobe, USA

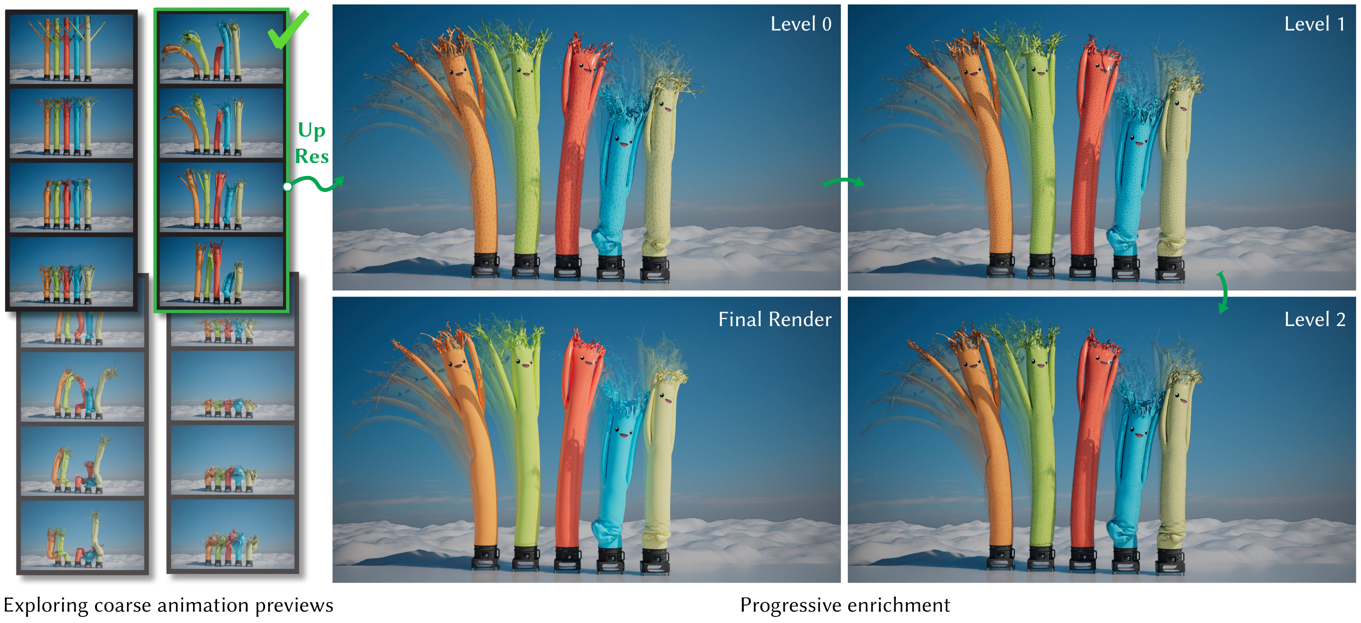


Fig. 1. **Animation Design of "Sky Dancers" with Progressive Dynamics:** (Left) With Progressive Dynamics's efficient, coarsest-level previewing, artists quickly explore a wide palette of physical design parameters (e.g., material, layout, wind forcing) to produce numerous variations of a "Sky Dancers" animation concept. Once finalized in preview, we refine the selected animation (green-boxed) to higher-resolutions (Right) maintaining the overall physical behavior of the animation while progressively enriching it with finer-scale rich dynamic wrinkling behaviors. (Bottom) The finest-level solution elevates the selected coarse-scale preview to a final, polished, high-quality animation ready for screen display that preserves the preview's physical narrative.

We propose Progressive Dynamics, a coarse-to-fine, level-of-detail simulation method for the physics-based animation of complex frictionally contacting thin shell and cloth dynamics. Progressive Dynamics provides tight-matching consistency and progressive improvement across levels, with comparable quality and realism to high-fidelity, IPC-based shell simulations [Li et al. 2021] at finest resolutions. Together these features enable an efficient animation-design pipeline with predictive coarse-resolution previews providing rapid design iterations for a final, to-be-generated, high-resolution animation. In contrast, previously, to design such scenes with comparable dynamics would require prohibitively slow design iterations via repeated direct simulations on high-resolution meshes. We evaluate and demonstrate Progressive Dynamics's features over a wide range of challenging stress-tests, benchmarks, and animation design tasks. Here Progressive Dynamics efficiently computes consistent previews at costs comparable to coarsest-level direct simulations. Its matching progressive refinements across levels then generate rich, high-resolution animations with high-speed dynamics, impacts, and the complex detailing of the dynamic wrinkling, folding, and sliding of frictionally contacting thin shells and fabrics.

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

Additional Key Words and Phrases: Progressive Simulation, LOD Animation, Cloth and Shells

**ACM Reference Format:**
Jiayi Eris Zhang, Doug L. James, and Danny M. Kaufman. 2024. Progressive Dynamics for Cloth and Shell Animation. *ACM Trans. Graph.* 43, 4, Article 104 (July 2024), 18 pages. https://doi.org/10.1145/3658214

Authors' addresses: Jiayi Eris Zhang, Adobe & Stanford University, USA, eriszhan@stanford.edu; Doug L. James, Stanford University, USA, djames@cs.stanford.edu; Danny M. Kaufman, Adobe, USA, dannykaufman@gmail.com.

## 1 INTRODUCTION

We present Progressive Dynamics, a coarse-to-fine, level-of-detail, physics-based animation method and design pipeline that provides rapid (and so practical) coarse-resolution previews of frictionally

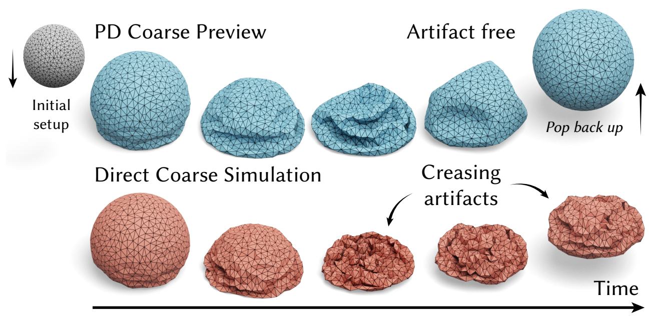


Fig. 2. **A Smashing Inflated Ball.** Direct simulations on coarse-resolution meshes (Bottom) often fail outright, because of severe locking artifacts, whereas Progressive Dynamics (PD) (Top) provides high-quality previews on the same coarse meshes.

contacting thin shell and cloth dynamics with progressive improvement to much higher-resolution animations of complex dynamics with comparable quality and realism to high-fidelity shell simulation output. To enable this workflow and to provide effective animation design cycles, we construct a *progressive* simulation method for dynamics that generates coarse preview trajectories that closely match (and so are highly predictive of) each following progressive refinement of the animation at finer resolutions.

Specifically, to avoid prohibitively expensive high-resolution design cycles, Progressive Dynamics generates animated sequences that improve physical quality (i.e., finer wrinkling, buckling, contact compliance, and geometric detail) at each increasing level of refinement. In turn, in order to provide useful previewing, bulk trajectories generated by Progressive Dynamics do not diverge across these levels of resolution—even over very long time-spans. This allows designers to efficiently explore a wide range of animation variations across physical and geometric design parameters at coarse levels, before safely investing in computing a final, matching, highest-resolution animation.

Progressive Dynamics thus complements prior methods in physics-based animation design (see Section 2) that employ combinations of control, enrichment and/or tracking to augment prescribed input shapes and sequences with enhanced physical realism. Such methods work to balance satisfaction between starting input and the physically based changes applied. Offering various trade-offs between these two extremes (e.g., via orthogonality, displacement, and constraints) provides numerous powerful modes for animation design. However, in all such methods, there is a common tension that must be navigated between satisfaction of the user input and the changes applied to them by the physical model enrichment. At the same time, because these methods are additive and/or constraint-based they do not offer the opportunity to capture the complex and rich coupling between material variations, fine-detail dynamics, and the underlying overall trajectories taken by a system [Chen et al. 2021a, 2023]. For example, changes in materials and folding patterns can drive a colliding body in an entirely different direction than originally envisioned (e.g., Figures 22, 10 and 9) while even subtle material variations (e.g, Figure 3) change the overall shape of frames—not just their wrinkling details.

As a complementary alternative, direct simulations of shell and cloth models by construction capture, and so allow animators to explore, the rich variations in dynamics generated by changing materials, geometries, scenes, and conditions. At the same time, direct simulation provides a seamless full coupling of dynamics across scales, ranging from the fine-scale detailed evolution of wrinkle patterns, contacts, and folds to the large-scale evolution of shape and trajectory. However, the catch is that employing direct simulation for design has previously required slow, high-fidelity simulations employing high-resolution meshes. To circumvent this obstacle, previs tools enabling artists to quickly set up and iterate over physics-based animations with fast, low-accuracy methods and low-resolution models have long been employed. However, their utility has been restricted by two common, fundamental, and well-documented [Bergou et al. 2007; Zhang et al. 2023] limitations: 1) final (compute-heavy) high-res hero simulations based on previs designs are not consistent and instead regularly generate entirely different results than their carefully designed, low-resolution starting points (see, e.g., Figure 4); and 2) fast low-res shell simulations employed for previs often generate significant artifacts, instabilities and errors that can preclude their use entirely for many real-world materials and settings (see, e.g., Figure 2).

To remove these restrictions and enable effective LOD simulation-based animation-design workflows for all shell materials, we begin with recent prior work in progressive simulation that enable the multi-resolution design of static cloth and shell drapes [Zhang et al. 2023, 2022] across a hierarchy of increasingly finer triangle meshes with high-quality IPC [Li et al. 2021] shell simulation. In this work, we extend the progressive simulation framework from the modeling of just static scenes to the LOD animation of high-quality finely detailed frictionally contacting shell and cloth dynamics.

Extending the progressive simulation framework to shell and cloth animations requires addressing three core challenges. First, how do we generate lowest-level preview simulations of dynamics with small numbers of DOF that avoid the aforementioned coarse-mesh shell simulation artifacts and instabilities? Second, how do we ensure that the frames generated by these same, low-DOF previews maintain close configuration matching, per time step, across levels of resolution all the way up to their corresponding highest-quality animation frames, generated by finest-resolution dynamics? Third, as discussed above, matching across resolutions is fundamentally at odds with providing unconstrained, high-quality enrichment of physical details and a natural flow of dynamics across frames at each level. How can we balance consistency per frame across levels while maintaining causality and avoiding collision tunneling and jumps in state, per level?

*Contributions.* Progressive Dynamics addresses these challenges by enabling rapid, predictive exploration of design parameters at coarse levels without standard low-resolution simulation artifacts, while regaining the expressiveness of high-quality cloth and shell simulation methods as refinement is applied to progressively finer levels of resolution. As we show in our evaluation, exceedingly close matches per frame are preserved across resolutions, even over long animation time spans of highly sensitive (e.g., high-speed, contact rich, and stiff) physical behaviors. At the same time, consistency

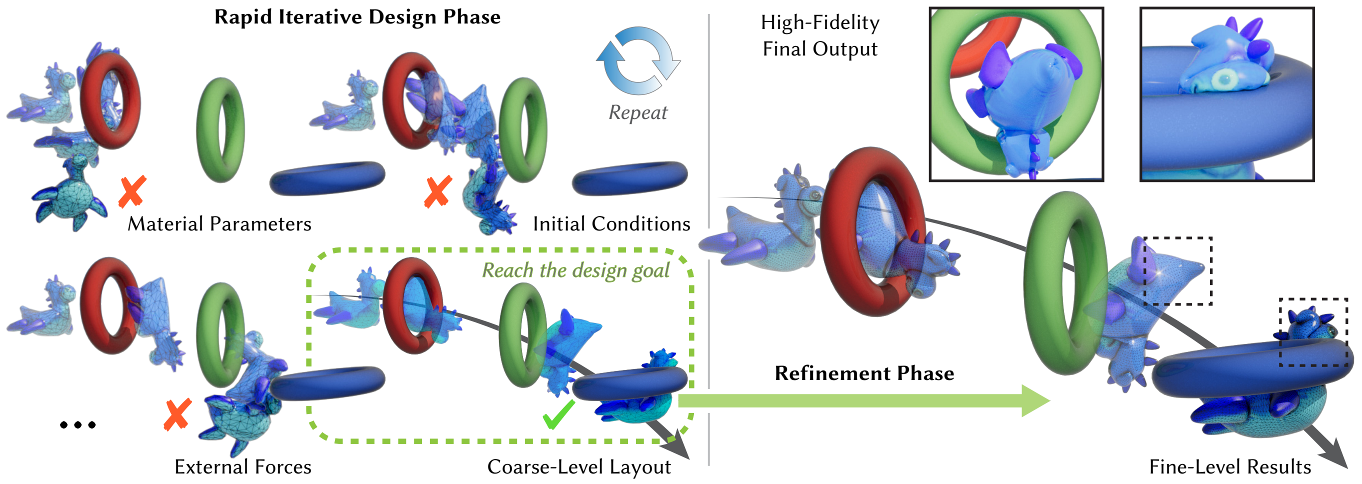


Fig. 3. **Progressive Design Cycle.** Left: quick explorations of design variations with Progressive Dynamics' coarse-resolution previewing, allows an artist to rapidly find the "just-right" layout and parameters for a lively Toy Toss with a sad, final face plant. Once found, progressive refinement simulates increasingly rich details over levels to produce a final animation respecting the initial design. In Figure 4 we attempt this same animation task with direct simulation tools.

across levels, per time step, is maintained without obstructing the temporal continuity and coherence of animations across frames. To do so, our core technical contributions include

- A new, simple, and efficient implicit time-stepping method for coarsest-scale high-quality previewing of fine-scale shell dynamics that significantly mitigates standard coarse-scale artifacts;
- A multi-resolution progressive formulation and prolongation model for shell dynamics on triangle mesh hierarchies; and
- A progressive dynamics LOD time-step and refinement algorithm for advancing shell and cloth animation sequences in *time and resolution* with both tight consistency across mesh-resolution levels per time-step, and coherence across each level's time steps.

We evaluate these features over a wide range of examples, stress tests, and comparisons—demonstrating that Progressive Dynamics now enables the rapid design of lively, realistic, and highly specific detailed material behaviors and trajectories to support the creation of animated physical narratives.

## 2 RELATED WORK

### 2.1 Thin Shell Simulation

The intricate wrinkling, buckling, and draping dynamics of thin shell materials and cloth fabrics have long played a fundamental role in animation. Artists seek to capture the lively, detailed, and highly specific ("just right") behaviors of these real-world materials to support their vision.

With these goals in mind, thin-shell simulation has remained a long-standing research focus in computer graphics [Baraff and Witkin 1998; Bridson et al. 2002; Grinspun et al. 2003; Harmon et al. 2009; Li et al. 2020b; Narain et al. 2012; Terzopoulos et al. 1987; Volino and Thalmann 2000]. As we do in this work, modern shell simulation methods largely adopt implicit time-stepping methods [Baraff and Witkin 1998; Bridson et al. 2002; Kim 2020; Li et al. 2020b, 2018; Narain et al. 2012; Otaduy et al. 2009; Tang et al. 2016, 2018] to provide stable numerical integration, with enhancements applied to jointly improve contact processing and limit strain [Bridson et al. 2002; Goldenthal et al. 2007; Harmon et al. 2008; Li et al. 2021; Narain et al. 2012].

It remains an open challenge to simulate shells with *combined* high quality and high speed. High-speed methods [Bender et al. 2013; Bouaziz et al. 2014; Daviet 2020; Li et al. 2020b; Ly et al. 2020; Schmitt et al. 2013; Selle et al. 2008; Tang et al. 2013, 2016, 2018; Wang 2021; Wu et al. 2020; Zhang et al. 2019] trade fidelity, controllability, and expressiveness for speed. This same theme is also reflected in commercial cloth simulation tools [Designer 2022; SideFX 2024] as well [Li et al. 2021; Zhang et al. 2022], e.g., see Figure 4. Correspondingly, increasingly high-quality models and methods for shell simulation provide ever-improving animations of rich cloth and shell behaviors [Chen et al. 2018a; Clyde et al. 2017; Guo et al. 2018; Harmon et al. 2009; Jiang et al. 2017; Li et al. 2018, 2021; Miguel et al. 2012; Narain et al. 2013, 2012; Vouga et al. 2011; Weischedel 2012] but generally come with the cost of an increased compute budget.

*Progressive Simulation.* The recently developed progressive simulation framework [Zhang et al. 2023, 2022] balances between these two extremes by allowing interactive exploration of design parameters with coarse preview simulations of shell equilibria for static modeling. The framework then computes a progressive nonlinear refinement to finer-level meshes once the coarse-preview model has been finalized. At the finest-level of resolution, prior progressive work delivers a final, fully converged solution to the underlying shell equilibrium model. In contrast, to enable exploratory physics-based *animation* design, we extend the progressive simulation framework from statics to dynamics. At the same time, we develop a method custom-suited for animation workflows, focusing on rapid generation of predictive coarse animations, progressing to high-quality realism in our finest-level animation results, without the constraint that the trajectories generated match a converged, numerical time-integration model.

*Multigrid Methods.* The progressive simulation framework is originally inspired [Zhang et al. 2022] by Sensitive Couture's [Umetani et al. 2011] application of a Cascadic multigrid [Bornemann and Deuflhard 1996] for hierarchical and successive solutions of draped equilibria. In turn, multigrid methods are applied more broadly to accelerate a wide range of linear-system solves, as when solving Newton-type problems for shell models [Tamstorf et al. 2015; Wang et al. 2018; Xian et al. 2019]. We note that multigrid methods largely focus on accelerating the inner-loop solves within time-step solvers, and so are complementary to progressive simulation methods. For Progressive Dynamics, they could be applied to accelerate the solution of the linear problems that we process within each Newton iteration, at each level of our progressive solver.

## 2.2 Enrichment, Tracking and Constraints

To complement physical simulation methods, a diverse toolkit of physics-based methods have been developed to help animators design and finalize complex cloth animations by employing varying combinations of enrichment, tracking, and/or constraint-based control.

*Tracking-based Enrichment.* A wide range of methods have been developed to synthesize shell-like wrinkles on a coarser starting base surface. When this input is in the form of art-directed low-resolution animation sequences (either coarsely simulated or created by animators), *tracking* methods have been developed which use carefully designed constraints that track the deforming input shape while allowing enrichment with simulated finer-resolution wrinkling details. The TRACKS method [Bergou et al. 2007] augments fine-mesh simulations with moment-based constraints that track the input animated geometries in an average sense while solving shell simulation subject to those constraints. Remillard and Kry [2013] and Wrinkle Meshes [Müller and Chentanez 2010] similarly apply tracking strategies to wrinkle-augment the skin and cloth of animated characters via constraints that couple shell layers and patches to input animated base meshes. While tracking provides a powerful and effective way to combine animation intent with physically based wrinkle synthesis, animators must rig their constraints (or else apply heuristics to do so) for each animation. In turn, changes in rigging constraint choices, for example the decomposition of surfaces into subdomains for the weighted-average constraints in TRACKS, generate input-sensitive animation enhancements, with hard-to-predict changes [Zhang et al. 2020]. At the same time, the tracking constraints themselves do not ensure that art-directed deformations in the coarse input will be maintained by the final fine result [Bai et al. 2016]. Likewise, to animate realistic shell dynamics a source coarse-mesh simulation for base animation must be generated which will necessarily have significant coarse animation artifacts (see Section 4.2). Solutions at finer scales will inherit these artifacts and can generate non-physical wrinkling patterns (e.g., wrinkles on wrinkles) and overall geometries [Chen et al. 2023].

*Complementary and Direct Enrichment.* To address the above challenges in tracking methods, the recently developed Complementary Dynamics [Benchekroun et al. 2023; Zhang et al. 2020] method applies an alternative approach in which physically simulated enrichment is instead constrained to be orthogonal to a rigged animation's kinematics. This ensures that all additional dynamic enrichment modifies only remaining, free DOF, and so respects animator intent. This provides an intuitive and direct tool for physics-based animation design. However, when it comes to shells and cloth, it remains challenging to define a *base* rigging that is suitably orthogonal to wrinkle enrichment. In part answering this implicit question, a natural and broadly applied solution has long been to effectively treat art-directed input coarse surfaces as a base kinematics. Wrinkle displacements away from this surface, for example, through normal maps [Lähner et al. 2018] or displacements [Chen et al. 2018b, 2021c,a, 2023; Santesteban et al. 2019; Zhang et al. 2021], are then treated as the "complementary" subspace. However, all such methods are fundamentally one-way coupled and so restrict enrichment to leave the overall underlying dynamics oblivious to and unchanged by changing surface behavior. In particular, it remains an open challenge to resolve dynamic interactions and self-contacts with these representations [Chen et al. 2021a]. Likewise in this direction are a range of fast, physically motivated post-processing methods [Gillette et al. 2015; Rohmer et al. 2010] that generate surface wrinkling via direct coarse-input deformation analysis—here, as in tracking methods, final wrinkle augmentation results then vary with the user parameter choice (e.g., choice of wrinkle size), rather than from the underlying physics of a shell model simulation.

*Data-Driven Direct Enrichment.* Along with the above empirical techniques, direct enrichment via learned wrinkling offers another powerful tool for physically based wrinkle augmentation. Fine wrinkle enrichments derived from both real-world motion capture [Lähner et al. 2018] and large collections of synthetic high-resolution simulations [Kim et al. 2013] have both been explored. A common target for these applications is a restricted domain of learned deformations, e.g., adding enrichment to a space of deformations for a parameterized garment conditioned on the learned kinematics of a draped shape [Hahn et al. 2014; Santesteban et al. 2019; Wang et al. 2010]. Even more broadly, upsampling has been learned for general geometric enrichment of details without conditioning on an underlying shape [Chen et al. 2021b; Kavan et al. 2011; Lee et al. 2019; Oh et al. 2018; Seiler et al. 2012]. This is a rapidly evolving and highly active area. However, along with the above-discussed general limitations in direct enrichment, these methods often still demonstrate commonly encountered issues in data-driven enrichment strategies: artifacts appear and quality degrades as methods are applied to synthesize beyond the scope of their data, synthesized wrinkle quality and resolution is generally much lower quality than that produced by direct capture and high-fidelity simulation, and, when it comes to dynamics, temporal coherence is often lost in the final result.

*Spacetime Constraints.* Spacetime control methods [Witkin and Kass 1988] are a natural complement to tracking strategies. Rather than constraining physical enrichment to track an animated surface well-defined in time, spacetime strategies instead constrain physics-based animation to hit target keyframes, sparsely distributed in time. Here, the fundamental balancing act, in all such spacetime optimization problems [Popović et al. 2003], is that either physical plausibility (i.e., shell dynamics in our case) can be a soft objective and the targets can be treated as constraints, or vice versa. Most methods adopt the former strategy for an as-physical-as-possible interpolation between frames. Nevertheless, as with many comparable interpolation problems, in-betweened trajectories are often brittle: they can generate surprisingly non-physical and hard-to-control trajectories as artists change and fine-tune their animations with unlikely targets and increased keyframe numbers. At the same time, notwithstanding many recent performance improvements [Barbič et al. 2009; Hildebrandt et al. 2012; Li et al. 2014; Schulz et al. 2014], the costs of large-scale spacetime optimizations (required after every artist refinement) for fine-resolution cloth dynamics are prohibitively expensive.

## 3 BACKGROUND: PROGRESSIVE QUASISTATICS

Zhang et al. [2023; 2022] recently introduced the progressive simulation framework for hierarchical coarse-to-fine, LOD modeling of complex frictionally contacting, shell and cloth *equilibria* on hierarchies of triangle-meshed geometries. Progressive quasistatics applies quasistatic stepping in a coarsest-level preview mode to discover local stable equilibria approximations and then generates consistent and improving solutions of this equilibrium shape over the hierarchy's increasingly finer-resolution meshes. This process ends, at the finest level, with a converged, high-fidelity C-IPC [Li et al. 2021] simulation solution of the (at rest) equilibrium system, on the hierarchy's highest-resolution mesh.

### 3.1 Hierarchy

For progressive simulation, a multi-resolution mesh hierarchy is constructed from the input geometry of the modeled system [Zhang et al. 2023]. The hierarchy is a collection of improving-resolution triangle meshes and a corresponding set of prolongation operators, each mapping to the next finer-resolution mesh. The meshes in the hierarchy are indexed in increasing resolution by subscript $l \in [0, L]$. At level $l$ we denote the undeformed (rest) and deformed positions of the mesh nodes as $\bar{x}_l$ and $x_l \in \mathbb{R}^{3n_l}$ respectively, where $n_l$ is the number of nodes at level $l$. Each level $l$ is equipped with a prolongation operator, $P^l_{l+1}(\cdot)$, that maps nodal positions and associated surface quantities from the current to the next level, $l+1$. To simplify the discussion, throughout, when clear, we will designate finest-level resolution quantities without decoration so that, e.g., $\bar{x}=\bar{x}_L$, $x=x_L$ and $n=n_L$.

Each simulation mesh[1] is modeled with shell ($\Psi$), contact barrier ($B$), friction ($D$), and, when required, strain-limiting potential energies ($S$) to compute the stable *equilibria* of frictionally contacting shells subject to imposed boundary conditions and external forces. These are the local (constrained) minimizers of the total potential energy constructed from the sum of the above potentials, $E_l = \Psi_l + B_l + D_l + S_l$.

[1]Throughout this work we apply Neo-Hookean membrane [Vouga 2024] and discrete-hinge bending [Grinspun et al. 2003; Tamstorf and Grinspun 2013] for shell elastics, and C-IPC [Li et al. 2021] barriers for contact, friction and strain limiting.

### 3.2 Per-Level Proxy Energies and Prolongation

At each coarsened level $l < L$, the progressive quasistatic framework computes improving approximations of the final equilibrium geometry by computing minimizers of a *proxy* for the finest-level's potential energy,

$$F_l(x_l) = \underbrace{B_l(x_l) + D_l(x_l) + S_l(x_l)}_{C_l(x_l)} + \Psi_L\big(P^l(x_l)\big). \tag{1}$$

Here, the proxy $F_l$, in contrast to $E_l$ above, allows coarsened levels to directly evaluate shell elastics at the finest resolution model, $\Psi_L$, via a direct prolongation, $P^l(x_l)$, from level $l$ up to the finest scale, while coarse barrier-based potential terms in $C_l(x_l)$ efficiently enforce contact and strain-limit feasibility directly on the current level-$l$'s updated geometry.

Zhang et al.[2023] construct a new nonlinear prolongation operator through the construction of decimation-based, fine-to-coarse hierarchies customized for shell simulation (both curved and flat geometries) that enables progressive simulation preserving the rest shape of all input triangle-meshed geometries. The per-level operators,

$$P^l_{l+1}(x_l) = U^l_{l+1} x_l + a^l_{l+1}(x_l) \tag{2}$$

(and the corresponding direct operators, $P^l_L = P^l$), each composed of the sum of a (sparse) linear *intrinsic* [Liu et al. 2021] map, $U$, and an *extrinsic* nonlinear offset, $a(\cdot)$, significantly improve over Zhang et al.'s [2022] originally proposed subdivision-based operator.

### 3.3 Refinement with Safe Initialization

When finalizing an update of the equilibrium solution at a level $l < L$, refinement to the next level, $l+1$, requires finding a feasible (non-interpenetrating and strain-limit satisfying) starting point for the next level's solve, close to the current level $l$'s converged solution, $x^*_l$. Notice that just directly prolonging to the next level, $P^l_{l+1}(x^*_l)$, would disregard contacts and strain limits, and therefore introduce unacceptable intersections and violate strain limits [Zhang et al. 2022]. Zhang et al. [2023] provide safe refinement with an efficient, shape-preserving edge-expansion-based upsampling and strain-limit relaxation that ensures feasibility for each refinement level in the progressive framework's decimation-constructed fine-to-coarse hierarchy. This, in turn, provides initialization close to the prolongation of the just-completed prior level's solution, so that progressive refinement towards the equilibrium solution can remain feasible across all levels of the hierarchy.

## 4 PROGRESSIVE DYNAMICS

### 4.1 Formulation

To extend the progressive simulation framework to Progressive Dynamics we broaden scope to consider both progressive improvement in spatial resolution (as in Zhang et al. [2023; 2022]) *and* forward dynamics in time. Our full state is now described by a space-time (multi-resolution) grid with spatially discretized positions, $x^t_l$, and velocities, $v^t_l$, at each grid point $(t, l)$ corresponding to time step

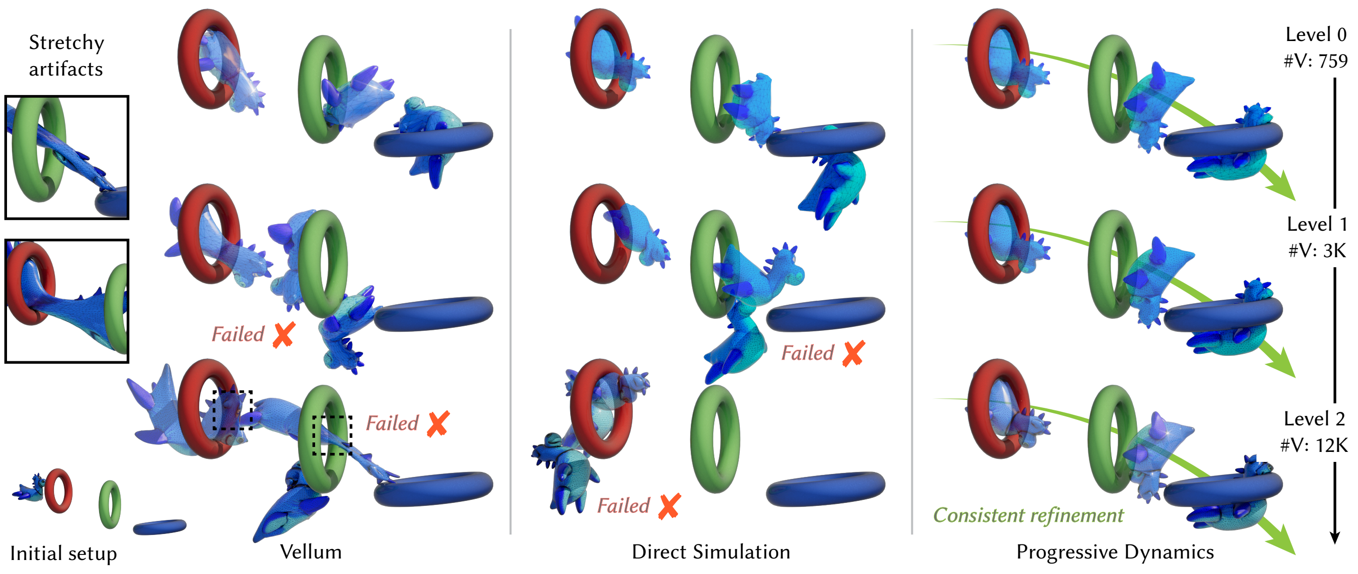


Fig. 4. **It's a Toy Toss between Vellum and Direct IPC Simulation, compared to Progressive Dynamics.** Even moderate changes in resolution make a huge differences in simulation outcomes for both fast simulation methods like Vellum's PBD (Left), and FE-based direct simulation with IPC (Middle)—here they change a carefully tuned toss through hoops at coarsest-level (Top row for each method) into a failed mission (Middle and Bottom rows for each method). In contrast, Progressive Dynamics (Right) obtains the same consistent, coarse-level designed trajectory at all level's resolution (all rows).

$t \in [0, N]$ and level of resolution $l \in [0, L]$.

$$\begin{array}{c} \\ \\ \text{Mesh refinement}\left\{\begin{array}{c} \\ \\ \\ \end{array}\right. \end{array}
\begin{array}{c} \overbrace{\qquad}^{\text{Time step}} \\ \begin{bmatrix} x_L^0, v_L^0 & \longrightarrow & x_L^1, v_L^1 & \cdots & x_L^N, v_L^N \\ \vdots & & \vdots & & \vdots \\ x_1^0, v_1^0 & \longrightarrow & x_1^1, v_1^1 & \cdots & x_1^N, v_1^N \\ \uparrow & & \uparrow & & \uparrow \\ x_0^0, v_0^0 & \longrightarrow & x_0^1, v_0^1 & \cdots & x_0^N, v_0^N \end{bmatrix} \end{array}$$

We say that Progressive Dynamics advances in the grid *horizontally* by forward-stepping system dynamics from time step $t$ to $t+1$ at a level $l$, and *vertically* by progressive refinement from resolution level $l$ to $l+1$.

## 4.2 Per-level Time Stepping

We start by recalling that forward time stepping of frictionally contacting shell dynamics can be cast variationally for a wide range of implicit numerical time integration methods as the minimization of an incremental potential (IP) [Li et al. 2020a, 2021; Ortiz and Stainier 1999]. Concretely, for a fixed discretization level $l$, an implicit Euler time step is solved by the minimization,

$$x_l^{t+1} = \operatorname*{argmin}_x \frac{1}{2h^2} \|x - \hat{x}_l^t\|_{M_l}^2 + E_l(x), \tag{3}$$

where $h$ is time step size, $M_l$ is the level-$l$'s mass matrix, $E_l$ is the level's above-defined total potential energy (with body and external forcing added), and $\hat{x}_l^t = x_l^t + hv_l^t$, is the explicit momentum update component of the time-step solve. Alternatives, e.g., BDF2 or implicit Newmark, follow similarly.

Naive, direct simulation for progressive dynamics by stepping each level $l$ individually with Equation 3, as discussed above in Section 2, can generate significant and unacceptable simulation artifacts at coarse levels and simulation trajectories that widely diverge across levels. E.g., see Figures 4, 2, 10, 14, 16, 9 and 19, and our analysis in Section 5. This means that coarse-resolution frames will not match fine, and can not be used for designing high-quality, fine-resolution animation sequences.

So, how do we generalize to multilevel progressive time stepping? Our first goal is to provide high-quality unconstrained previews of dynamics at our coarsest resolutions. To do so, we first observe that direct coarse model time stepping, as in Equation 3 above, is oblivious to the finer-level models in our hierarchy. So, to enrich our coarse time stepping with fine-level material information, we follow Zhang et al. [2022] and similarly evaluate shell elastic energies from the finest-resolution model in our hierarchy via direct prolongation (see Equation 2) with Zhang et al.'s [2023] shell-customized PSQ upsampling operator, $P_l()$, in each of our time step solves. For implicit Euler, this amounts to simply solving each time step with a modified coarse-and-fine prolonged IP,

$$x_l^{t+1} = \operatorname*{argmin}_x \frac{1}{2h^2} \|x - \hat{x}_l^t\|_{M_l}^2 + C_l(x) + \Psi_L\big(P^l(x)\big). \tag{4}$$

where, recalling from Section 3.2 above, $\Psi_L$ is the total fine-level shell elastics potential and $C_l$ the sum of remaining current level-$l$ contact and strain-limiting potentials for the physical system.

*Discussion.* The prolonged IP in Equation 4 is a simple and natural extension of Zhang et al.'s coarse-level quasi-static stepping model

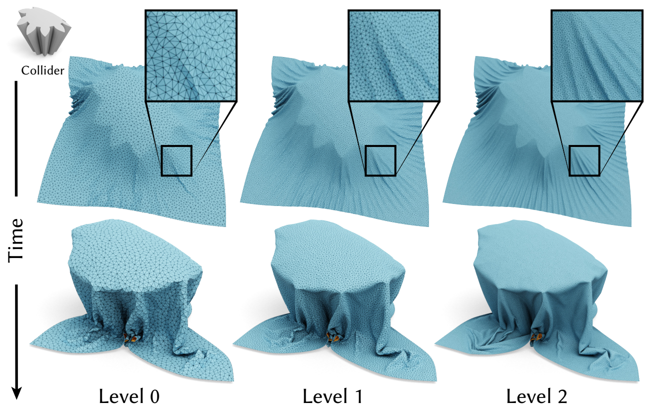


Fig. 5. **Dynamically Draping.** Progressive Dynamics generates consistent frames of cloth while progressively improving the wrinkle and wrap of the frames at each level of resolution.

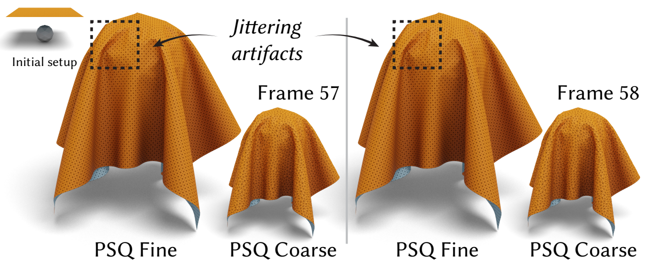


Fig. 6. **PSQ Extended to Dynamics.** A "horizontal-and-then-vertical" application of PSQ solved for dynamics (momentum-balance rather than equilibrium) gives consistent frames at each time-step across levels, but still produces inconsistent dynamics (jitters and popping) in animations across each level.

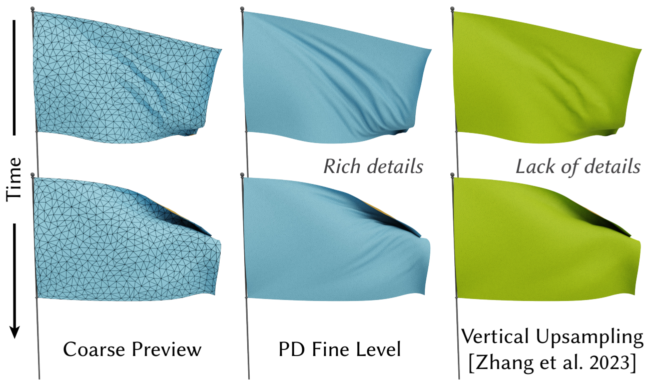


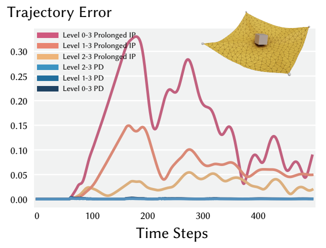
Fig. 7. **Of Coarse its a Fine Flag.** (Left, blue) Starting with a pair of coarse-mesh time-stepped flapping flag frames, applying just PSQ's geometric upsampling (Right, green) safely refines the mesh but does not enrich with simulated details. In contrast, Progressive Dynamics (PD)' (Middle, blue) simulated enrichment refines both wrinkles and additional fine-scale wave propagation.

for computing equilibria [2022] extended to dynamics[2]. However, we note a critical distinction with implications for our final method. For animation, our output at each level is, of course, a sequence of animation frames. As such, we require high-quality geometry output for all time-step solves; therefore, *we solve time steps to low-tolerance accuracy with multiple Newton iterations.* In contrast, Zhang et. al [2022] apply quasi-static stepping just to compute a final, at-equilibrium shape via a sequence of low-accuracy quasi-static time steps, and so apply just a small, fixed-numbers of Newton iterations (generally just one [Zhang et al. 2023]) per time-step solve, until convergence to a final, accurate equilibrium shape model is reached.

With these changes in place, we see that, consistent with Zhang et al.'s [2023] analysis of the PSQ operator, per-level time-stepping with the prolonged IP in Equation 4 significantly improves the quality (including reduced locking) and stability of our coarse-level preview simulations and so provides a meaningful "ground-floor" foundation to explore previews for final, high-quality results in our progressive framework. See Figure 2 and Section 5 for comparisons, examples, and analysis. Equally important, as we will see in the following sections, applying these prolonged, finest-level energies in our time-stepping also serves a second critical role of promoting consistency across our hierarchy as we progressively improve resolution, by providing all levels with the same, highest-quality energy evaluations for our material model.

## 4.3 Challenges to Progressive Refinement for Dynamics

We now have a simple method to compute a full, coarsest-level preview of our animation end-to-end with prolonged IP time stepping via Equation 4. The next question, that we address here and in the following sections, is how can we advance our preview steps vertically to improving levels of resolution in our hierarchy?



Given the effectiveness of time stepping in-level with our prolonged IP model, a simple, comparable method to solve our hierarchy suggests itself: as in standard pre-vis methods, time-stepping each level independently but now with our prolonged IP applied to obtain consistency. However, in trying this horizontal approach, we see that while the animation results for each individual simulated level improve over simple direct simulation, with nice continuity of dynamics, they still significantly diverge from each other and therefore cannot be used for Progressive Dynamics (see inset). Although disappointing, this is not entirely surprising in this approach, as each time step at each level will sample its fine-level energies from a different configuration, so that the corresponding fine-level forces will vary significantly over time. Again, as in direct simulation, this leads to wide divergence in trajectories at each level.

[2]Replacing $\hat{x}_l^t$ with $x_l^t$ in Equation 4 directly retrieves Zhang et al.'s quasi-static stepping model.

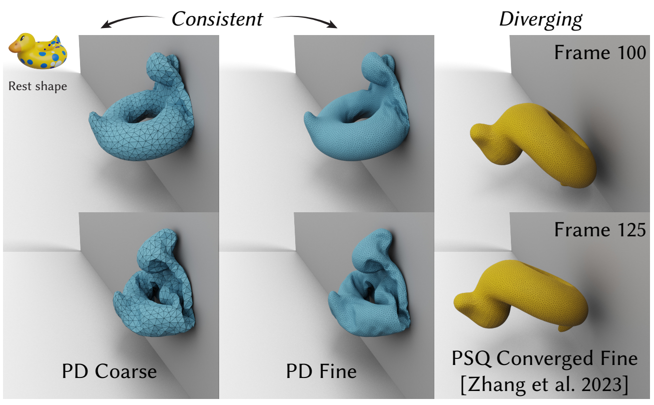


Fig. 8. **Poor PSQ Ducky.** PSQ's progressive enrichment can be applied to each frame in an initial coarse time-stepped simulation like this inflated, thrown ducky toy sequence (Left, blue coarse). However, as we see (yellow meshes, Right) when PSQ is applied to each of these coarse preview time-steps, PSQ's equilibrium assumption immediately drives the crumpled duck, in the new enriched frames to its rest shape (nearest equilibrium) creating an immediate and unusable inconsistency in the duck's animation. For reference compare with the fine-level Progressive Dynamics (PD) solution (Middle, blue fine).

Alternately, noting the undesirable trajectory divergence in our above experiment, we could try to ensure consistent frames at each time step, across levels, by applying Zhang et al.'s [2023] PSQ refinement to vertically improve each coarse-level timestep individually, with finer simulated details. PSQ progressively enriches the physical details of an input coarse mesh while promoting frame consistency across all levels, and so, on first glance, it seems well-suited for the job. Indeed, as we see in Figure 6 and Section 5.2, applying PSQ to each frame of our *dynamically* time-stepped coarse-level input both enriches finer-resolution frames with physical detailing and promotes *vertical* consistency with well-matching frames across each time-step's levels[3], as it immediately breaks consistency with the coarse-level preview and removes all dynamics after just the first level of refinement. However, *at each level* the enriched simulated details lose temporal coherence, leading to unacceptable inconsistencies and discontinuities (e.g., jitters, popping and jumps) across time steps in the finer-level geometric details that are generated.

In summary, purely horizontal time-stepping strategies produce continuous trajectories across time, per level, but can not maintain consistency across levels per time step—they are under-constrained. On the other hand, purely vertical improvements of resolution per time-step (even via prior progressive simulation methods) can sometimes produce consistency across resolutions per time step, but generate discontinuities in dynamics across time per level—they are over-constrained.

[3]Note that the original PSQ refinement operation enriches the geometry at each finer-level with multiple simulated quasi-static steps to reach equilibrium. Here in Figure 6 we apply the natural modification of PSQ to dynamics by solving each new level's PSQ refinement with a single, converged dynamics solve. Directly applying the unmodified PSQ operator, which solves each new level's refinement by relaxing each refined frame to a nearest equilibrium solution, gives even worse behavior, as we see in Figure 8

## 4.4 Progressive Advancement

Based on the above analysis we seek to balance continuity in time with consistency per frame, across levels of resolution. To do so, consider two levels of refinement, $l$ and $l+1$.

As analyzed above, repeated purely vertical refinement, $(t, l) \mapsto (t, l+1)$, gains us consistency across resolutions at the cost of temporal coherence. On the other hand, repeated horizontal time stepping, $(t, l) \mapsto (t+1, l)$, is a well-posed time integration and so clearly obtains temporal coherence but produces distinctly different trajectories for each level. These diverging trajectories are a direct consequence of each level's changing discretization computing the solution to a different (albeit closely related) ODE [Courant et al. 1967]. Over time, these solutions diverge, with the thin-shell materials and strong contact forces we treat here only exacerbating these sensitivities and accelerating the divergence.

Considering these tradeoffs, we instead propose to keep bulk trajectories aligned across levels by prolonging each level's momentum to the next. Specifically, we loosely couple interlevel dynamics by applying each prior level's explicit, finite-difference extrapolation of position to define the inertial update for the time-step solves at the next level. In practice, this process gives us an exceedingly simple and effective algorithm for Progressive Dynamics.

We begin by first forward time-stepping our coarsest-level, $l = 0$, prolonged IP (Equation 4) across our time span to compute our preview. This generates our full preview state, $x_0^t, v_0^t$ for all $t \in [0, N]$ and so populates the bottom row of our Progressive Dynamics solution grid.

This coarse preview simulation can be efficiently repeated with varying design parameters (e.g., changing geometries, boundary conditions, materials) until trajectory results satisfy our goals. Once a suitable preview run is completed, we then begin our progressive LOD refinement.

For each new level $l+1 > 0$, for all $t+1 \in [1, N]$, we first compute momentum updates prolonged from our last level. Specializing to implicit Euler, recall velocities are given, in-level, by the finite difference stencil $v_l^t \leftarrow (x_l^t - x_l^{t-1})/h$. Correspondingly, we construct our prolonged momentum updates as

$$\begin{aligned} \hat{x}_{l+1}^t &= P_{l+1}^l(x_l^t) + h\big(V_{l+1}^l(x_l^t)\big)v_l^t \\ &= P_{l+1}^l(x_l^t) + \big(V_{l+1}^l(x_l^t)\big)(x_l^t - x_l^{t-1}). \end{aligned} \tag{5}$$

Here $P_{l+1}^l(x)$ is the PSQ shell-prolongation operator [Zhang et al. 2023], as covered in Section 3, and $V_{l+1}^l(x) = \nabla P_{l+1}^l(x)$ our corresponding velocity prolongator.

We then compute each new level's time-step advancement, at grid points $(t+1, l+1)$, with our progressive IP solve

$$x_{l+1}^{t+1} = \operatorname*{argmin}_x \frac{1}{2h^2}\|x - \hat{x}_{l+1}^t\|_{M_{l+1}}^2 + C_{l+1}(x) + \Psi_L\big(P^{l+1}(x)\big). \tag{6}$$

We construct our progressive IP above with three key terms whose sum is minimized. First, we have IPC barriers, $C_{l+1}(x)$, defined on the current level's geometry. These barriers ensure strict feasibility (nonintersection and strain-limit satisfaction) for each time step's solution so that at all levels our solutions are guaranteed intersection-free. Second, we again (as in our preview solves) apply here our

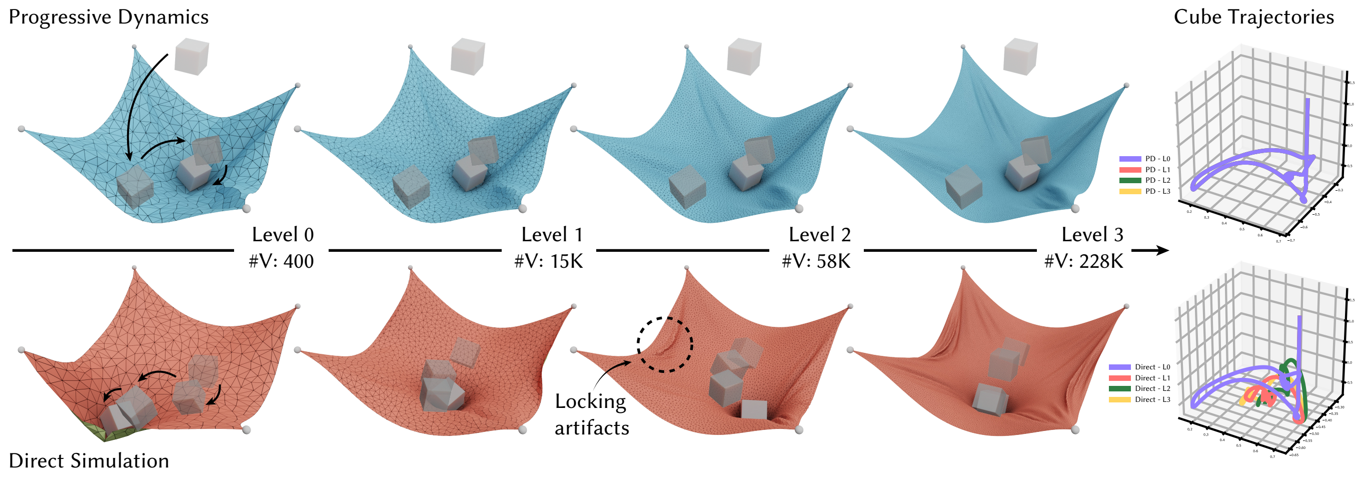


Fig. 9. **Bouncy Cube.** Progressive Dynamics (Left), in contrast to direct simulation (Right), resolves consistent dynamics across resolutions for impacts, even between between widely varying, soft and stiff materials.

prolonged elasticity[4] potential, $\Psi_L\left(P^{l+1}(x)\right)$, which enriches our current level's solution by evaluating finest-level shell elasticity forces. Finally, the time step solve's inertial energy, $\frac{1}{2h^2}\|x-\hat{x}^t_{l+1}\|^2_{M_{l+1}}$, advances our solution at time step $t+1$ with a momentum update lagged from the last level's explicit extrapolation of position. This ensures that neighboring time-step solutions on either side of each time-step solve at the same level, $(t, l+1)$ and $(t+2, l+1)$, use the *same, temporally consistent* extrapolation of motion from the completed finite differencing of the *same* time-step sequence generated by the prior level $l$'s solution.

Each new level's dynamics are thus driven by the momentum of the preceding so that consistency is preserved. At the same, time elasticity and contact generate enriched geometry in the new level which then, in turn, generate new higher-resolution dynamics, via momentum, for the following level.

## 4.5 Diagonal Refinement Perspective

Once a level $l$'s time-step solves are complete, we can proceed to the hierarchy's next level and repeat this process until we finalize our finest-resolution, level-$L$ solves. Starting with a full preview solution at level $l=0$, Progressive Dynamics thus advances *diagonally* across the space-time grid. Each time step updates from $(t, l)$ to $(t+1, l+1)$ and so advances both mesh refinement and time simultaneously.

We observe that from this perspective, we are effectively just solving each time step *mesh-adapted* [Manteaux et al. 2017] to the next finer discretization of our domain. We simply first map all fields needed for each next time solve from our $l$-level mesh to $l+1$, and then solve the next time step with prolonged elasticity evaluations. Starting from level $l=0$, we could thus advance diagonally from any time $t$ until a boundary of our space-time grid is reached. Doing so for all grid points then progressively fills our entire multilevel solution in our grid, all the way up to our finest level $L$. In turn, this enables a great deal of flexibility in how time-step solves can be computed for Progressive Dynamics simulations.

## 4.6 Solving Progressive Dynamics Time Steps

For minimization solves of each prolonged IP time step in Equation (6), we employ Li et al.'s [2021] barrier-filtered Newton-type solver for shells. Following Zhang et al. [2023], we hold the nonlinear offset term (see Section 3), $a^l_{l+1}$, constant in the gradient computations of the PSQ shell-prolongation operator. Along with simplifying optimization steps in minimizing our prolonged IPs, this means that we apply velocity prolongation with the intrinsic map $U^{l-1}_l \approx V^{l-1}_l(x)$. For all preview time-step solves for levels below finest resolution, we solve each time-step problem until the back-solved decrement is below tolerance (Li et al.'s [2021] tolerance measure at $10^{-3}$) or (rarely) until the decrement no longer provides descent. For all finest-level solves, we terminate solely when convergence is reached with Newton decrement below tolerance.

Progressive Dynamics' diagonal structure removes the necessity of sequential time-step computation that is generally required by standard implicit time-integration methods. Each prolonged momentum update, $\hat{x}^t_l$, can be computed from a grid point $(t, l)$'s support, $(t-1, l-1)$ and $(t-2, l-1)$, at the prior level, and so in principle, each time-step solve can be computed *independently* in parallel from all other time steps in the same level.

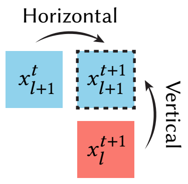
However, there are practical restrictions to this freedom. As we are solving IPC-based time-step solves we require a feasible (intersection-free, strain-limit-satisfying) initialization to begin each Newton-type solve of the barrier-based IP's in Equation (6). Likewise, more broadly, as in all implicit time-stepping methods for nonlinear systems, this initializer should not be "too far" from the solution. This latter criteria is of course fuzzy and is usually met by simply initializing each time-step solve with the last time-step's solution. In turn, for IPC time-stepping this choice of initializer also conveniently meets the requirement of providing a feasible starting point for each solve as well. For Progressive Dynamics time-step

[4]Note that for our finest-level IP solves there is, of course, no prolongation to fine-level mesh applied inside elasticity energy evaluations, as $P^L = P^L_L = Id$.

**Algorithm 1** Progressive Dynamics Algorithm

```
 1: procedure ProgressiveDynamics($x_0^0, \ldots, x_L^0, v_0^0, \ldots, v_L^0, N$)
 2:     $l \leftarrow 0$                                              ▷ Begin with coarsest-level solve
 3:     for each time step $t \in [1, N]$ do                          ▷ Section 4.2
 4:         $\hat{x}_l^t \leftarrow x_l^t + h v_l^t$
 5:         $x_l^{t+1} \leftarrow \text{argmin}_x \frac{1}{2h^2} \|x - \hat{x}_l^t\|_{M_l}^2 + C_l(x) + \Psi_L\big(P^l(x)\big)$
 6:         $v_l^{t+1} \leftarrow (x_l^{t+1} - x_l^t)/h$
 7:     while $l < L$ do                                              ▷ Section 4.3
 8:         for each time step $t \in [1, N]$ do                      ▷ Section 4.5
 9:             $\hat{x}_{l+1}^t \leftarrow P_{l+1}^l(x_l^t) + h\big(V_{l+1}^l(x_l^t)\big)v_l^t$     ▷ Section 4.4
10:             solve
11:                 $x_{l+1}^{t+1} \leftarrow \text{argmin}_x \frac{1}{2h^2} \|x - \hat{x}_{l+1}^t\|_{M_{l+1}}^2 + C_{l+1}(x) + \Psi_L\big(P^{l+1}(x)\big)$
12:                 warm starting with $x_{l+1}^t$ or PSQ safe-upsampled $x_l^{t+1}$   ▷ Section 4.6
13:             $v_{l+1}^{t+1} \leftarrow (x_{l+1}^{t+1} - x_{l+1}^t)/h$
14:         $l \leftarrow l + 1$                                      ▷ Progress to next finer level
```

solves, we call this sequential approach (across a level) to initialization *horizontal warm-starting*. We also construct a complementary feasible, nearby *vertical warm start* to initialize a next time-step solve at a grid point $(t, l)$ using only state from the prior level. To do so, we begin with the already computed solution for the time-step $x_{l-1}^t$ at the prior resolution that is already in the support of our new solve. We then create the new feasible initializer for our Newton solve at $(t, l)$ by applying Zhang et al.'s [2023] safe upsampling algorithm (Section 3.3) to map $x_{l-1}^t$ to the current resolution $l$ while preventing intersection and strain-limit violations.



With both our vertical and horizontal warm starts defined, we can now solve time steps at each level with any combination of horizontal and vertical warm starting; see Algorithm 1. The only requirement is that to solve a time step at $(t, l)$, all prior time-step solves required for its support (down and left in the space-time grid, starting with our initial preview solve at $(0, 0)$) have been previously computed. This opens the door to many potential parallelization strategies. As one concrete practical example, a simulation can be divided into multiple subsets of time per level with each subset solved in parallel by warm starting vertically at the subset's start and horizontally across the subset's remaining time steps. In the extreme, after solving the coarsest preview level, we can solve levels sequentially, with each time step for that level solved in parallel using vertical warm-starts. Although this latter strategy is unlikely to be practical, it is an extreme test of our vertical warm-starting. In our supplemental video, we demonstrate both this latter extreme version and the former time subdomain hybrid warm-start for the highly dynamic Waving Flag animation in Figure 12. Aside from these proof-of-concept examples, the remainder of the animations we generate here are computed with the per-level horizontal warm-starting. We leave additional investigation of these asynchronous time-stepping opportunities to mix and match vertical and horizontal warm-starting in Progressive Simulation to future work.

## 5 EVALUATION

We implement a common test-harness code for evaluating both our Progressive Dynamics method and prior Progressive Quasistatic and direct IPC simulations in C++, applying CHOLMOD for linear

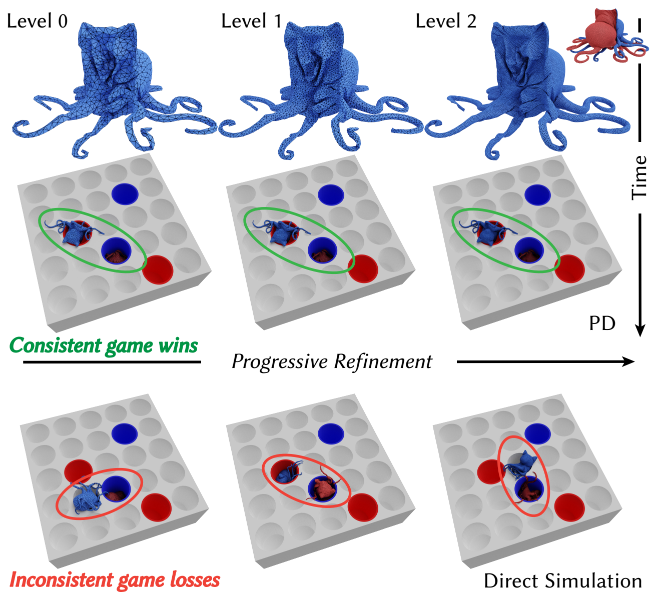


Fig. 10. **Gaming Octocats.** The landing position of octocat balloons fired at one another depends sensitively on simulation conditions. Direct simulation gives different answers at every resolution. Progressive Dynamics (PD) simulates a winning throw from its coarse preview and repeats it at every resolution.

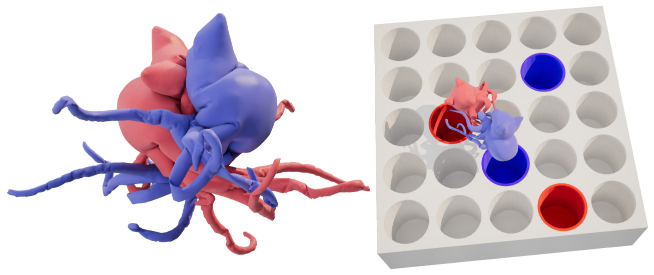


Fig. 11. **Octocats Detail.** See Figure 10.

solves and Eigen for other linear algebra routines [Guennebaud et al. 2010]. We summarize example statistics and timings on an Ubuntu Linux machine (Intel i7-10700K 3.80GHz, 32 GB RAM).

### 5.1 Benchmark Examples

We instrument a wide range of examples to evaluate and compare Progressive Dynamics on stress-test animations with increasing complexity. Please see our supplemental video for all animations.

*Dynamic Drape.* We start with a simple *dynamic* drape test (see Figure 5), analogous to static-drape modeling problems in quasistatic progressive simulation [Zhang et al. 2022]. Here we apply Progressive Dynamics to progressively animate the dynamics of a thin (0.07mm thick) 1m square cloth dropped on a sharp-edged gear obstacle. Across increasing resolutions per level of 5.8K , 23K, and 90K vertices respectively, the progressive animations consistently

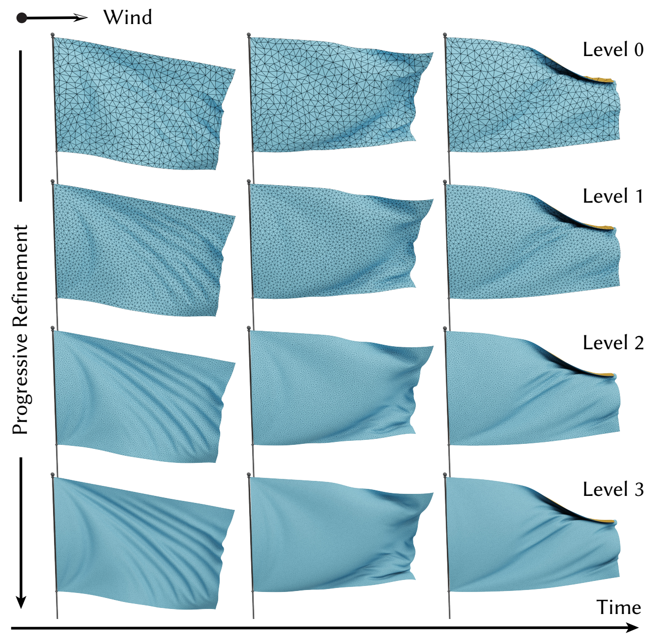


Fig. 12. **Waving the Flag.** Progressive Dynamics maintains temporally coherent motion, and frame-consistency across refinement levels.

follow the same overall trajectory and final drape shape, while each progressively finer simulation generates increasing physical details in the wrinkles, fold-overs, and obstacle wrapping.

*Bouncy Cube.* Next, we test a collision between a heavy and very stiff (effectively rigid, $Y = 8.2 \times 10^{10}$ Pa) cube-shaped shell with sharp edges, dropped onto a soft "trampoline" formed from a 1m-square pinned thin (0.07mm thick) rubber sheet simulated with four resolution levels (400, 15K, 58K, 228K vertices per level). As we see in Figure 9 (bottom), numerical stiffening and membrane locking are especially challenging to overcome when colliding materials vary in stiffness and stretch significantly (as they do here) under rapid impact—resulting in widely disparate sheet deformations and cube trajectories for each direct simulation at each different resolution. In contrast, in Figure 9 (top) and Figure 19 (left), we confirm both visually and numerically (tracking the cube's trajectory) that the progressive simulation forms consistent and comparable sheet deformations and rebounding trajectories across resolutions; also, finer-scale simulations better capture the sharp impact of the cube's edges on the cloth, and the fine wrinkling of the sheet under tension.

*Waving the Cat Flag.* To evaluate longer-term consistency under persistent, highly dynamic motion, we set a 20s long flag-waving animation test. Here we drive a cloth flag (0.4mm thick, strain-limited to 10%) from the state of Catifornia with wind. Starting with a coarse (585 vertices) lowest-resolution preview mesh, Progressive Dynamics captures the flag's overall rapidly changing flapping and folding motion throughout the animation. As Progressive Dynamics proceeds with its finer-resolution levels (2.2K, 8.6K, 34.1K vertices), it progressively captures more of the fine-detailed waves propagating

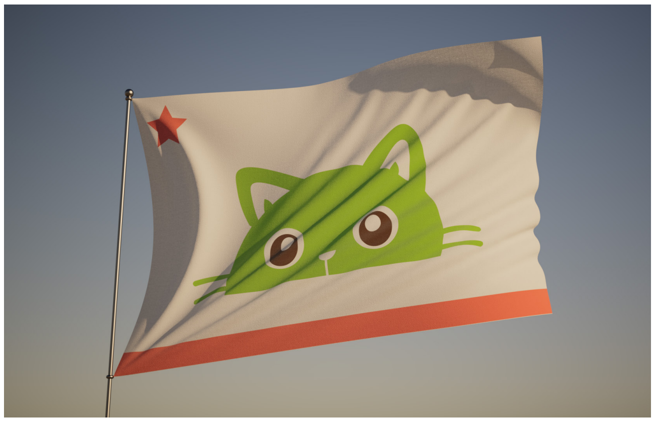


Fig. 13. **The Catifornia State Flag.** See Figure 12.

across the flag along with sharper-detailed creasing, folding, and self-contact. As we see in Figure 12, tight matches across resolutions are preserved throughout all frames of the sequence, without drifting as the simulation progresses through the entire animation. Please also see our additional analysis of this match in Section 5.2 below.

*Bouncing Jumble.* Complex multibody trajectories, especially with sharp contacts and friction, are notorious for their extreme sensitivity to small perturbations. Minor changes in state, much less resolution, can lead to entirely different pile-ups and jumbles. Here we extend our bouncing cube scene (with comparable levels of progressive resolution) to a challenging stress test by dropping a pile of stiff, heavy and sharp shell shapes (same material as the original cube) onto the bouncy trampoline. In Figure 14, left and 19, right (also see our supplemental video), we confirm both visually and numerically that despite undergoing large numbers of rapidly changing and varied inter-body collisions, the progressive solution maintains close matches throughout the animations across all levels. We note, however, one interesting exception, where a single cube mismatches across the levels for 5 successive frames (for 0.05s total) and then, even more interestingly, is corrected by Progressive Dynamics to preserve consistency for its trajectory, along with the rest of the simulated domain, for the remainder of the 10s long animation. We analyze this interesting exception in detail below in Section 5.2.

*Ball Drop.* As covered in Section 4.2 above, coarse-resolution direct simulations of shells can suffer from severe membrane-locking artifacts. These artifacts are significantly reduced by Progressive Dynamics's prolonged, coarse-level time-stepping, enabling high-quality previews. As an extreme test of Progressive Dynamics's ability to mitigate locking at coarse resolutions, we fire a stiff inflated elastic ball at the ground, using a coarse 930-vertex mesh. In Figure 2, we see that while the direct simulation immediately suffers severe locking, crumpling against the ground, Progressive Dynamics's coarse-level preview on the same mesh compresses on impact, dimples, and rebounds, with the buckled dimples successively popping back out.

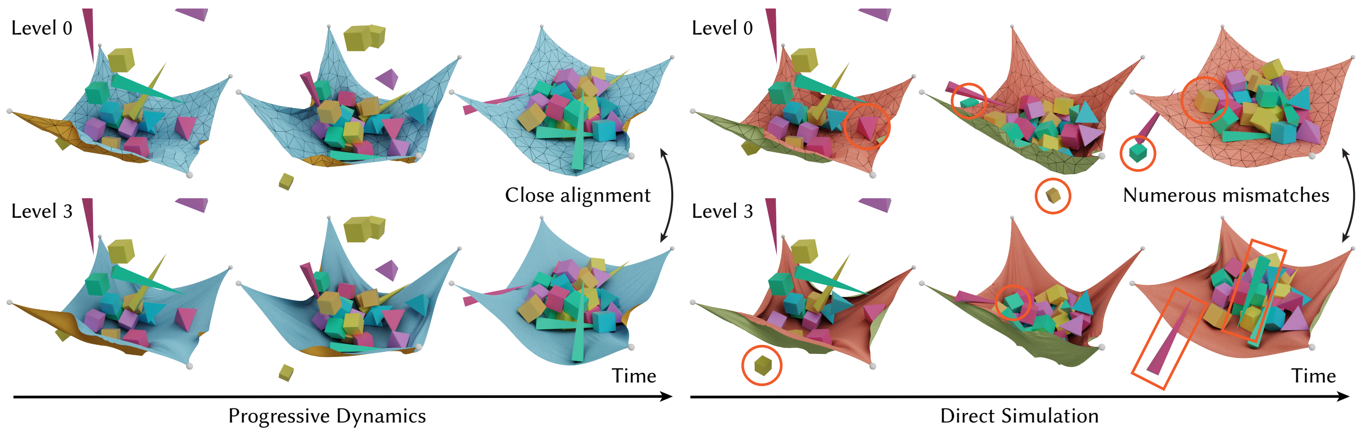


Fig. 14. **Bouncing Jumble.** Sharp and complex multibody contacts, as in this shell drop on trampoline example, destroy consistency across levels for direct simulation (Right), whereas Progressive Dynamics (Left) maintains consistency across its progressive resolutions.

*Laundry Basket.* We evaluate the ability of Progressive Dynamics to capture detailed wrinkling and folding behavior by dropping a thin (0.08mm thick) cotton sheet across the top of a smooth laundry basket. At highest resolution, the sheet first puckers with fine wrinkles along the edge of the basket, flops off one side, and then slowly slides and buckles with a cascade of detailed wrinkles into a folded crumple at the bottom of the basket. As we see in Figures 15 and 16, Progressive Dynamics' improving solutions across a 3-level hierarchy (5.8K, 23K, and 90K vertices) each capture the same overall shape and dynamics while progressively enriching the simulated wrinkling geometries to capture the dynamics of the branching and merging finer folds. In contrast, as we see in Figure16, direct simulations diverge almost immediately from each other across levels of resolution and exhibit locking artifacts at coarse resolution. In Figure 15, we also qualitatively compare side-by-side the detailed wrinkling and folding evolution, at key phases of the aforementioned dynamics, of the finest-level Progressive Dynamics animation and the finest-resolution direct IPC simulation. Here we see, in examination, that per phase (which both follow) each method obtains different but visually comparable quality thin-cloth material behavior.

*Spin Cycle.* Progressive Dynamics' ability to mitigate locking at coarse resolution enables it to animate high-speed, tightly contacting frictional dynamics. Here, we drop the same cotton sheet from the Laundry Basket onto a high-friction ($\mu = 0.5$), scripted sphere that then begins to spin at high speed. This rapidly twists the cloth into a tightly wound, high-contact, layered spiral. After 4 seconds of spinning, the sphere slows it rotational velocity by half, causing the cloth to slowly unwind. After unwinding, the cloth wobbles as it begins to slowly stick-slip across and eventually fly off the sphere. In Figure 17, we see that even at a coarse preview level with a 5.8K vertex mesh, Progressive Dynamics is able to capture the entire range of behavior for this animation; progressive updates (at 23K and 90K vertices, respectively) then enrich the wrinkling and contact details while consistently maintaining shape *and* the same frictional stick-to-slip behavior across resolutions, so that the time of transition where the cloth flies off, as the spinning slows, remains the same across levels.

## 5.2 Comparisons and Analysis

*Direct Methods Comparisons.* For each of our above benchmark examples, we have run corresponding direct simulations using the same base IPC simulation code in our test code at each mesh resolution level. Please see our figures throughout this paper (Figures 4, 10, 14, 16 and 9) and our supplemental video, demonstrating the large and rapid divergence of both shape and trajectory across resolutions of these direct simulations. We observe that both IPC and Vellum produce completely different animations, with widely different material behaviors and trajectories, per resolution. Likewise, see Figures 2, 16, 9 and supplemental videos for demonstrations of the above-discussed locking artifacts for direct simulation previews on coarse-level meshes. We also note that these issues are not restricted to just high-fidelity FE-based methods. In Figure 4, we evaluate Houdini Vellum's [SideFX 2024] XPBD-based simulation side-by-side with direct IPC simulation on the Toy Toss (see below) design task. In contrast, we invite careful frame-by-frame examination in our Figures' 1, 4, 8, 5, 22, 10, 12, 14, 16, 17, 9 videos and accompanying supplemental data of per-frame mesh geometries demonstrating qualitative consistency with simulated refinement across resolutions.

*Progressive Quasistatic Simulation Comparison.* As covered in Section 2, the recently developed Progressive Shell Quasistatics (PSQ) [Zhang et al. 2023] method is the most closely related competing method to Progressive Dynamics—offering unconstrained vertical enrichment, across levels of increasing mesh resolution, via shell simulation on an initial coarse frame input. In Figure 8, we apply PSQ's full simulation-based vertical refinement algorithm, to improve each coarsest-level dynamic time step of an inflated shell ducky-toy thrown at a wall without gravity. Applying coarsest-level Progressive Dynamics (top row), the toy flies at the wall and crumples against it.

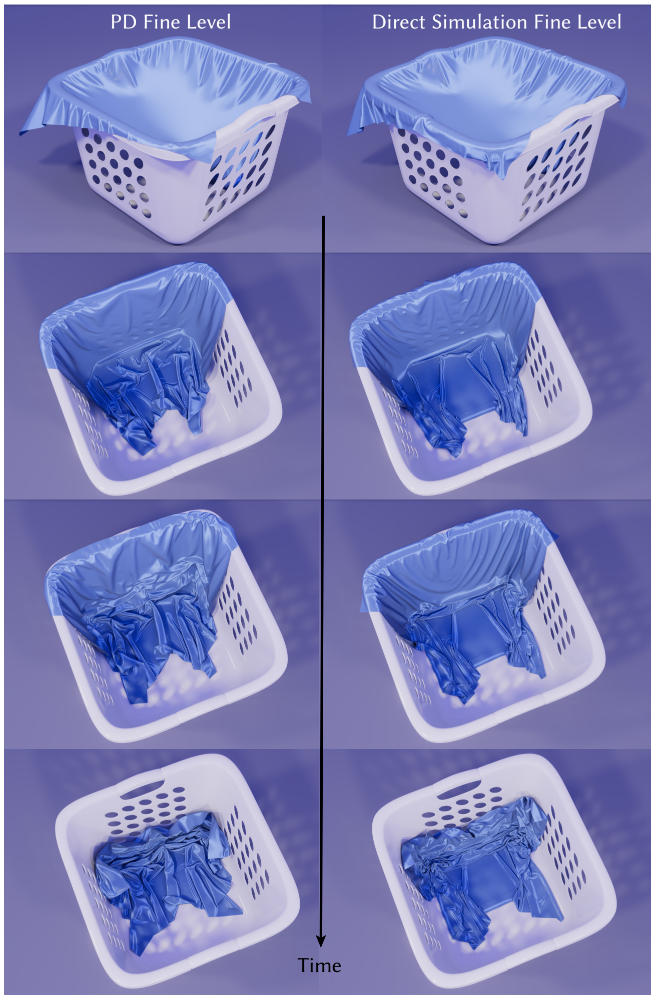


Fig. 15. **Laundry Basket.** Dropping a high-resolution cotton sheet in a basket gives qualitatively comparable, but different high quality results, for frames of a finest-resolution direct simulation (PD) (Right) and Progressive Dynamics (Left) animation. However, as we see next in Figure 16, that Progressive Dynamics maintains consistent animation across resolutions while direct simulation diverges and suffers from locking artifacts at lower resolutions.

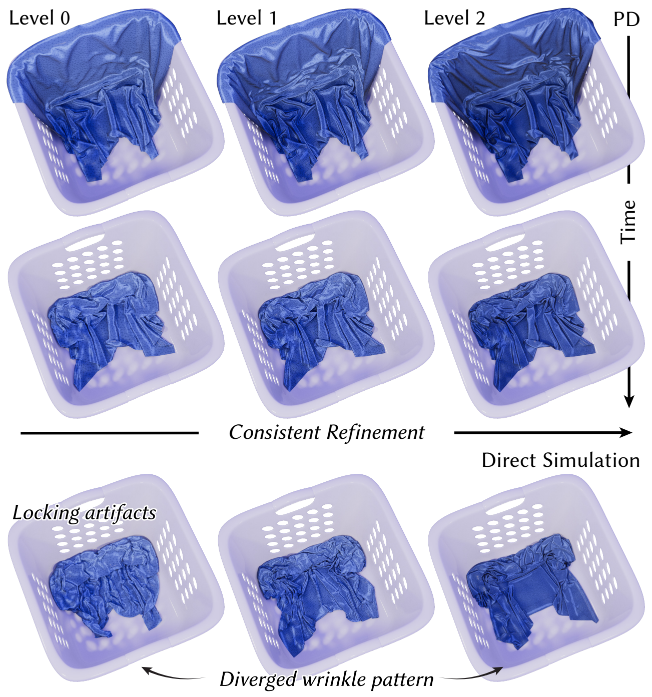


Fig. 16. **Laundry Basket Comparison.** See discussion in Figure 15.

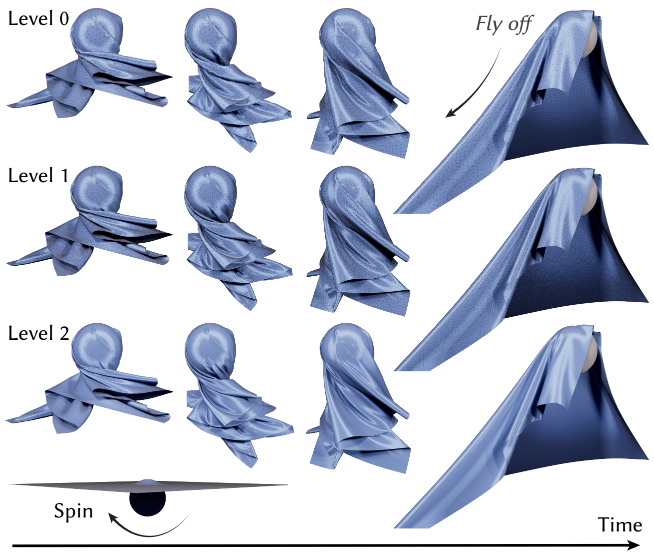


Fig. 17. **Spin Cycle.** Progressive Dynamics captures the same high-speed, tight-winding contact, and complex frictional stick-slip dynamics across all levels of refinement—please see our video for detailed playback of this animation.

Examining PSQ's generated refinement on just one simple frame of the duck, at maximum compression in impact, is already sufficient to demonstrate PSQ's inability to progressively simulate dynamics. Recall that, PSQ's vertical enrichment applies multiple simulated quasi-static steps to reach equilibrium at each level's refinement to generate new simulated details. As such, for the duck's flight, it immediately during the very first level of enrichment removes all crumpling from the impacting duck, returning back to rest shape (the nearest equilibrium) creating an immediate and unusable inconsistency in the duck's refined trajectory.

The thrown duck demonstrates that the equilibrium assumption of the PSQ method prevents its application to dynamics. However, it is also tempting to consider applying only PSQ's contact-safe vertical upsampling algorithm [Zhang et al. 2023], to similarly refine each coarse-level time step. Recall that this method upsamples geometry to a closest interpenetration-free geometry, without static-simulated enrichment, and so would not suffer the same temporal

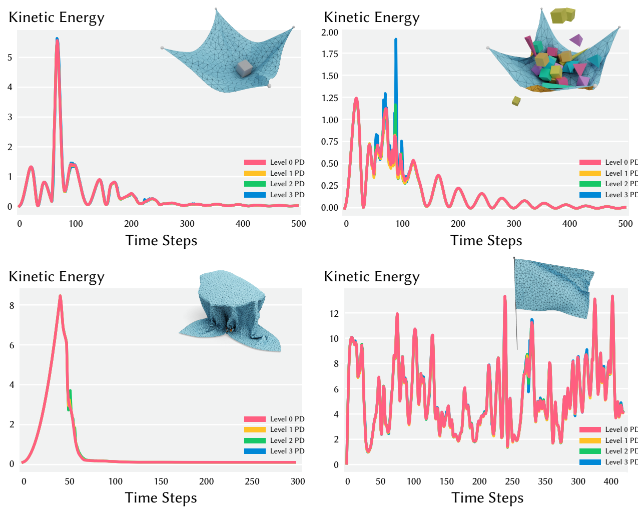


Fig. 18. **Kinetic Energy vs Resolution.** We plot the kinetic energy, per time step, of each resolution in the Progressive Dynamics hierarchy across a range of varying examples. Here, with the exception of a small span of frames in the Bouncy Jumble benchmark (please see our evaluation section for a detailed analysis of this span), we see close first-order consistency in the overlaid kinetic energies.

discontinuity issues as in the full PSQ pipeline. In Figure 7 we apply PSQ upsampling to the Waving Cat Flag sequence. Examining the finest-resolution results, we see that upsampling, not surprisingly, gives almost identical geometries (with minor variations responding to contact resolution) to the initial coarse-resolution time steps we begin with. For PSQ upsampling, we observe that simulated enrichment is key to useful enrichment, while for PSQ's full algorithm, we see that statically simulated, vertical enrichment is likewise not a solution for dynamics.

*Consistency, Evaluating Trajectory Difference.* Accompanying our qualitative comparisons, in Figure 9 we plot the overlaid trajectory of the center of mass of the cube shell across resolutions in the Bouncy Cube. In Figure 19, we correspondingly plot trajectory errors for both Bouncy Cube and Bouncing Jumble examples by plotting summed 2-norm differences of all centers of mass, between resolutions, per time step. Across both example trajectories, we see the rapid deviation of direct methods and the long-term consistency of Progressive Dynamics.

*Consistency and Limitations in Physical Behavior.* While the previous analysis demonstrates the coherence of our simulation trajectories, a natural question is whether high-order consistency of the physical behavior is preserved by Progressive Dynamics animations. In Figure 18, across a range of different animation behaviors (Bouncy Cube, Dynamic Drape, Bouncy Jumble and Waving Flag), we plot the kinetic energy per time step of each level in the Progressive Dynamics hierarchy. Here, with the exception of the aforementioned small span of frames in the Bouncy Jumble benchmark, we again see close first-order consistency in the overlaid trajectory's kinetic energies.

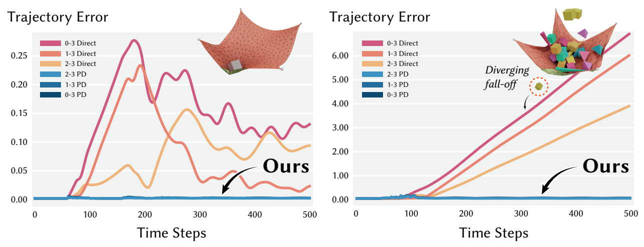
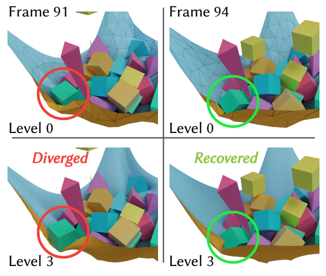


Fig. 19. **Center of Mass Deviation.** Analyzing the trajectory errors for both the Bouncy Cube (Left) and Bouncing Jumble (Right) examples, we plot the summed, 2-norm differences of all centers of mass, between resolutions, per time step. Across both example trajectories, we see the rapid deviation of direct simulation methods and the long-term consistency of Progressive Dynamics across levels.

Next, if we zoom in to the divergence of kinetic energies for the Bouncy Jumble, we observe that it is centered around a peak difference at frame 91. As we see in see in the inset figure, when we make a detailed examination of the frame geometries around this peak we find that, starting at frame 89 in the animation, one lonely cube in the animation and the corresponding deformation of the trampoline around it, begins to diverge by a small but significant amount from the positions predicted for it by coarser levels. We then observe that, at frame 91, corresponding to the peak difference in the kinetic energy plot, we see a maximum difference in the cube's configuration per level. When we closely examine other regions of the frames, we do not see differences elsewhere in the geometries. Even more interesting, we then observe that by frame 94 (0.05s later from the initial divergence), Progressive Dynamics appears to repair itself so that trajectories for both the offending cube and all other shells in the simulation remain matching across resolutions. This behavior is then reflected as well with the small kinetic energy differences in the remainder of the 10s simulation's plots.



*Timing.* Here we report timing results for Progressive Dynamics. There are two key takeaways. First, by construction, each step of simulation in a finest-level Progressive Dynamics simulation has identical (no overhead) runtime costs per iteration as its corresponding finest-level direct simulation. However, there is one caveat, as discussed earlier in Section 4.6 that subsets of Progressive Dynamics time steps allow for parallel computation of time steps per level with vertical warm-starting, while direct simulation requires a purely sequential processing. Second, Progressive Dynamics preview simulations are significantly faster than direct, fine-level preview simulations that would previously have been required to create their results. Concretely, across the Laundry Basket, Smushing Octocats, Bouncing Jumble and Waving Cat Flag examples, we see a 21x, 30x, 52x, and 75x speed-up for Progressive Dynamics preview simulation steps over the direct, fine-level preview steps that would previously have been required to create these results, respectively. We have also observed that, compared to a direct simulation of the same resolution, our coarse-level preview simulation incurs approximately a 2x overhead—primarily due to the necessity of gradient sampling and restriction from the finest level. Finally, we observe, as expected that the cost of computing each intermediary level goes up proportionally with its resolution (slower compared to coarse preview and still significantly faster than fine) with increased costs (e.g., 3.7x for Bouncing Jumble as mesh resolution goes up by 4x). Speedups then vary significantly depending on example, finest target mesh resolution, number of preview levels employed, and the complexity of scene dynamics, with substantial possible increased performance.

## 5.3 Changing Time Step and Levels

We focus on generating animations with practical, frame-rate size time steps with $h$ ranging in 0.04s to 0.01s. We also apply modest numbers of intermediate levels, generally one or two, in between the coarsest and finest meshes in our hierarchies. We find that this balances sufficient intermediate previewing *and* solution improvement against the increased cost (see above) of computing final animations as the number of in-between levels grows. With these settings we are able to generate all examples evaluated in the previous sections, as well as in the design tasks we cover below in Section 5.4. At the same time it remains an interesting question to consider how Progressive Dynamics behaves as we further decrease timestep size and increase numbers of intermediate levels.

*Varying Time Step Sizes.* Decreasing time step sizes further for Progressive Dynamics enables the generation of animations that capture higher-frequency dynamics and reduces the numerical damping inherited from the underlying implicit Euler time integrator we employ in our model. Applying smaller time steps likewise incurs the obvious cost of more computation (more time-step solves) required for the same span of animation time—just as in direct time stepping. In Figure 21 we explore this behavior by dropping a thin (0.07mm thick) 1m square cloth on a bunny obstacle animated with Progressive Dynamics stepped at $h = 0.025$s and 0.0025s respectively. Considering each animation's behavior at three key points in their trajectory: during initial impact, sliding across the ground, and finally at rest, we see different respective behaviors consistent with a more and less damped system. Faster collisions at smaller timesteps propagate finer and more chaotic wrinkles during collision, and fine-detailed curved wrinkles opposing the sliding along the ground. In contrast, increased damping of dynamics at larger time steps produces regular wrinkling during initial impact that evolves into finer branches along the ground during the slower sliding. At rest, both animations exhibit similar draping behavior with, of course, different final configurations and some material variation. For each frame, we also include the corresponding coarsest preview demonstrating consistent previewing generated across these widely varying timestep sizes.

*Discussion of Time Step.* More broadly, beyond this example, we can consider the expected behavior of the Progressive Dynamics model as the time step becomes small. Each diagonal step $(t, l) \mapsto (t + 1, l + 1)$ will, of course, apply less (compare Figure 21 left and right) change per level of advancement as the time step decreases. Correspondingly, adding more intermediate levels, and so more vertical resolution (see Figure 20 and our analysis below) can balance for decreasing timestep sizes with increasing amounts of diagonal steps of enrichment.

*Number of Levels.* As discussed above, additional intermediate levels in the Progressive Dynamics hierarchy incur more cost to produce final, finest-level animations. In Figure 20, we compare frames of the coarsest and finest levels, using the above bunny drop example, generated by Progressive Dynamics using hierarchies with increasing numbers of intermediate levels (the same first and last level meshes). In the coarsest-level results (top row), we see that Progressive Dynamics preview results are the same (by construction) irrespective of the number of intermediate levels we employ. The finest-level animations produced (bottom row) then of course differ depending on the number of intermediate levels. We observe here a gradual but clear improvement in the amount of fine-scale details generated by Progressive Dynamics as we increase the number of intermediate levels applied in the hierarchy. At the same time, we note that the bulk geometry of each fine-level result remains close to the coarsest-level preview.

*Discussion of Levels.* Better understanding the tradeoffs for improving details via higher-resolution final meshes vs increasing the numbers of levels in the hierarchy remains important future work to explore. Currently we so far find that, at the extreme, aggressive two-level hierarchies (see Figure 20 left) without any intermediate levels are generally insufficient for diagonal resolution and previewing, while a modest number of intermediate levels (one or two) combined with an appropriately high-quality final mesh of sufficient resolution offer practical mid-process previewing (see our below design examples) and a good starting point to balance cost against generated dynamic detail. At the same time, as covered above in our analysis of time resolution, smaller time-step applications may also require finer vertical resolution and so the use of more intermediate levels in our hierarchy. For such applications, this also warrants additional future exploration and analysis. Finally, we can observe that, in the limit, as we increase the numbers of intermediate levels, we decrease the DOF change per level, and so the amount of enrichment possible for each level's solve. Empirically, as in Figure 20 bottom, we correspondingly observe the consistency maintained between the bulk shape and coarse geometric previews in increasing vertical resolution solutions, but this requires further analysis for better understanding.

## 5.4 Animation Design with Progressive Dynamics

We use coarse-level Progressive Dynamics simulations to design multiple detailed physics-based animations.

*Toy Toss:* This aforementioned design example was shown earlier in Figures 3 and 4 and demonstrates the coarse-level design benefits of Progressive Dynamics. Please see the video for a detailed animation design exposition for this example, with comparisons between Progressive Dynamics, Direct IPC, and Houdini Vellum solvers.

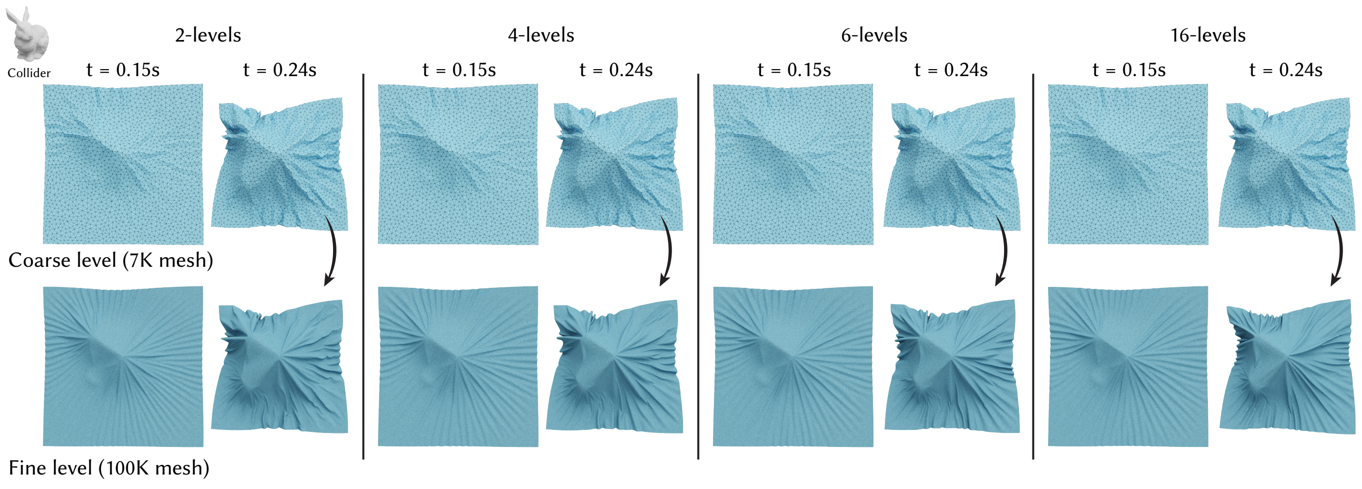


Fig. 20. **How Many Levels?** Left to right we compare coarsest and finest-level frames generated by Progressive Dynamics as we vary the numbers of intermediate levels in the hierarchy. Coarsest-level previews (Top row) remain the same (by construction) irrespective of how many intermediate levels are applied. Finest-level animations produced (Bottom row) differ depending on the number of levels, with the bulk geometries of each fine-level remaining close to the coarsest-level preview, and a gradual improvement in the amount of fine-scale details generated by Progressive Dynamics as we increase the number of intermediate levels.

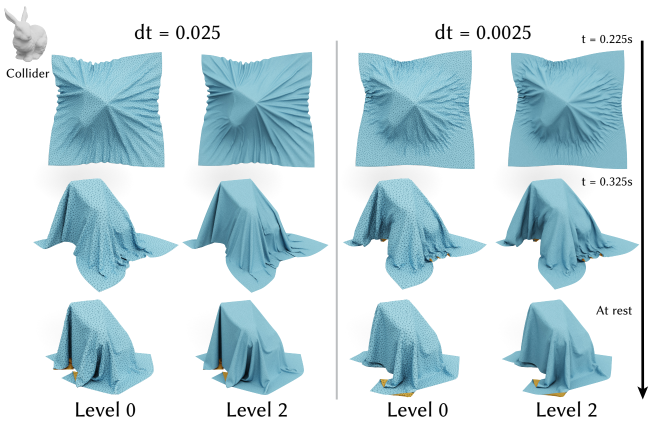


Fig. 21. **Changing Time Step.** Dropping a thin square cloth on a bunny obstacle animated with Progressive Dynamics stepped at $h = 0.025$s and 0.0025s respectively, we see different respective behaviors consistent with a more and less damped system at three key points in their trajectory: during initial impact, sliding across the ground, and finally at rest.

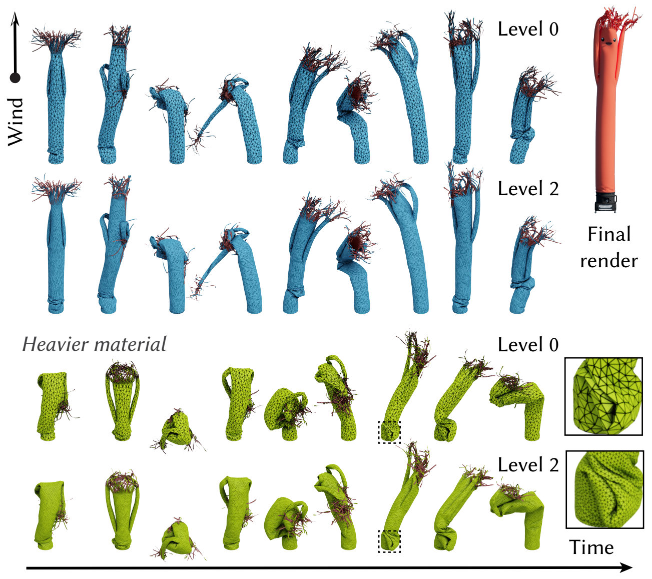


Fig. 22. **Let's Dance.** Artists can apply Progressive Dynamics for an end-to-end workflow, beginning by rapidly exploring and composing complex animations like this inflatable sky dancer with dynamics driven by the interplay of wind forces, inflation, elasticity, and frictional contact.

*Smushing Octocats:* We throw two inflated octocat objects at each other and use fast coarse-level design to ensure that the red and blue octocats hit in an interesting way and also land in colored holes. In contrast, traditional direct IPC simulation fails to achieve cross-resolution consistency. See Figures 10 and 11.

*Sky Dancers:* We designed a group of air-blown sky dancer (aka "inflatable tall boy") models with detailed and fun cloth dynamics (see Figure 1). Given their chaotic and unpredictable motion, coarse-level previews were essential to obtaining desirable animation results before running expensive fine-scale simulations. The line of sky dancers are simulated using an upward wind force, allowing for interactions involving contact and friction. Artists can effectively utilize our coarse preview to rapidly experiment with various designs, including initial conditions, materials, and layouts, by swiftly generating multiple samples. After finalizing the design, the standout sequence undergoes refinement through our progressive dynamics framework. This process elevates the sequence into a polished, high-quality animation, ready for screen display and distinguished by its intricate dynamic wrinkling behaviors.

## 6 CONCLUSION

We have presented Progressive Dynamics, a coarse-to-fine, level-of-detail, physics-based animation method and design pipeline. We demonstrate that it enables rapid coarse-resolution previews of frictionally contacting thin-shell and cloth dynamics with progressive improvement to much higher-resolution animations of complex dynamics with quality and realism comparable to high-fidelity shell simulation output.

There are many interesting directions and opportunities for future work in regard to the questions and limitations discussed above. Foremost, coarse-level design is powerful; however, there remain limitations in the application of potentially too-coarse preview meshes due to kinematic locking that can constrain our simulation results. Adaptive remeshing could thus be interesting future work. Future research should also consider generalization of Progressive Simulation from cloth and shells to additional physical models, such as volumetric and higher co-dimension models.

## ACKNOWLEDGMENTS

Jiayi Eris Zhang is supported by a Stanford Graduate Fellowship. Doug James acknowledges support from Adobe, Meta, SideFX, NVIDIA, and the Department of Energy, National Nuclear Security Administration under Award Number DE-NA0003968. We thank Alec Jacobson, Raymond Yun Fei, and Jérémie Dumas for valuable discussions, and our anonymous reviewers for constructive feedback.

## REFERENCES

Yunfei Bai, Danny M. Kaufman, C. Karen Liu, and Jovan Popović. 2016. Artist-directed dynamics for 2D animation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–10.

David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques.* 43–54.

Jernej Barbič, Marco da Silva, and Jovan Popović. 2009. Deformable Object Animation Using Reduced Optimal Control. *ACM Transactions on Graphics* 28, 3 (July 2009), 53:1–53:9.

Otman Benchekroun, Jiayi Eris Zhang, Siddartha Chaudhuri, Eitan Grinspun, Yi Zhou, and Alec Jacobson. 2023. Fast Complementary Dynamics via Skinning Eigenmodes. *ACM Trans. Graph.* 42, 4, Article 106 (jul 2023), 21 pages.

Jan Bender, Daniel Weber, and Raphael Diziol. 2013. Fast and stable cloth simulation based on multi-resolution shape matching. *Computers & Graphics* 37, 8 (2013), 945–954.

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.

Folkmar A Bornemann and Peter Deuflhard. 1996. The cascadic multigrid method for elliptic problems. *Numer. Math.* 75, 2 (1996), 135–152.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.

Robert Bridson, Ronald Fedkiw, and John Anderson. 2002. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. *ACM Trans. on Graph.* 21 (05 2002).

Hsiao-Yu Chen, Arnav Sastry, Wim M van Rees, and Etienne Vouga. 2018a. Physical simulation of environmentally induced thin shell deformation. *ACM Trans. Graph. (TOG)* 37, 4 (2018), 1–13.

Lan Chen, Lin Gao, Jie Yang, Shibiao Xu, Juntao Ye, Xiaopeng Zhang, and Yu-Kun Lai. 2021b. Deep deformation detail synthesis for thin shell models. *Computer Graphics Forum* (2021).

Lan Chen, Juntao Ye, Liguo Jiang, Chengcheng Ma, Zhanglin Cheng, and Xiaopeng Zhang. 2018b. Synthesizing cloth wrinkles by CNN-based geometry image super-resolution. *Computer Animation and Virtual Worlds* 29 (05 2018), e1810.

Lan Chen, Juntao Ye, and Xiaopeng Zhang. 2021c. Multi-Feature Super-Resolution Network for Cloth Wrinkle Synthesis. *Journal of Computer Science and Technology* 36 (06 2021), 478–493.

Zhen Chen, Hsiao-Yu Chen, Danny M. Kaufman, Mélina Skouras, and Etienne Vouga. 2021a. Fine Wrinkling on Coarsely Meshed Thin Shells. *ACM Transactions on Graphics (TOG)* 40, 5 (2021), 1–32.

Zhen Chen, Danny Kaufman, Mélina Skouras, and Etienne Vouga. 2023. Complex Wrinkle Field Evolution. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–19.

David Clyde, Joseph Teran, and Rasmus Tamstorf. 2017. Modeling and data-driven parameter estimation for woven fabrics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 1–11.

Richard Courant, Kurt Friedrichs, and Hans Lewy. 1967. On the partial difference equations of mathematical physics. *IBM J. of Research and Development* 11, 2 (1967).

Gilles Daviet. 2020. Simple and scalable frictional contacts for thin nodal objects. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 61–1.

Marvelous Designer. 2022. https://www.marvelousdesigner.com

Russell Gillette, Craig Peters, Nicholas Vining, Essex Edwards, and Alla Sheffer. 2015. Real-Time Dynamic Wrinkling of Coarse Animated Cloth. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '15)*. 17–26. https://doi.org/10.1145/2786784.2786789

Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. 2007. Efficient simulation of inextensible cloth. In *ACM SIGGRAPH 2007 papers.* 49–es.

Eitan Grinspun, Anil N Hirani, Mathieu Desbrun, and Peter Schröder. 2003. Discrete Shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation.* Citeseer, 62–67.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3.

Qi Guo, Xuchen Han, Chuyuan Fu, Theodore Gast, Rasmus Tamstorf, and Joseph Teran. 2018. A material point method for thin shells with frictional contact. *ACM Trans. Graph. (TOG)* 37, 4 (2018), 1–15.

Fabian Hahn, Bernhard Thomaszewski, Stelian Coros, Robert W. Sumner, Forrester Cole, Mark Meyer, Tony DeRose, and Markus Gross. 2014. Subspace Clothing Simulation Using Adaptive Bases. *ACM Trans. Graph.* 33, 4, Article 105 (July 2014), 9 pages. http://doi.acm.org/10.1145/2601097.2601160

David Harmon, Etienne Vouga, Breannan Smith, Rasmus Tamstorf, and Eitan Grinspun. 2009. Asynchronous contact mechanics. In *ACM Trans. on Graph. (TOG)*, Vol. 28. ACM.

David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. 2008. Robust treatment of simultaneous collisions. In *ACM SIGGRAPH 2008 papers.* 1–4.

Klaus Hildebrandt, Christian Schulz, Christoph von Tycowicz, and Konrad Polthier. 2012. Interactive spacetime control of deformable objects. *ACM Transactions on Graphics* 31, 4 (July 2012), 71:1–71:8.

Chenfanfu Jiang, Theodore Gast, and Joseph Teran. 2017. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Trans. Graph. (TOG)* 36, 4 (2017).

Ladislav Kavan, Dan Gerszewski, Adam Bargteil, and Peter-Pike Sloan. 2011. Physics-Inspired Upsampling for Cloth Simulation in Games. *ACM Trans. Graph.* 30 (07 2011), 93. https://doi.org/10.1145/2010324.1964988

Doyub Kim, Woojong Koh, Rahul Narain, Kayvon Fatahalian, Adrien Treuille, and James F. O'Brien. 2013. Near-Exhaustive Precomputation of Secondary Cloth Effects. *ACM Trans. Graph.* 32, 4, Article 87 (July 2013), 8 pages. https://doi.org/10.1145/2461912.2462020

Theodore Kim. 2020. A Finite Element Formulation of Baraff-Witkin Cloth. In *Symposium on Computer Animation.*

Zorah Lähner, Daniel Cremers, and Tony Tung. 2018. DeepWrinkles: Accurate and Realistic Clothing Modeling. In *ECCV*.

Tae Min Lee, Young Jin Oh, and In-Kwon Lee. 2019. Efficient cloth simulation using miniature cloth and upscaling deep neural networks. *ACM Transactions on Graphics* (2019).

Cheng Li, Min Tang, Ruofeng Tong, Ming Cai, Jieyi Zhao, and Dinesh Manocha. 2020b. P-Cloth: Interactive Cloth Simulation on Multi-GPU Systems using Dynamic Matrix Assembly and Pipelined Implicit Integrators. *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia)* 39, 6 (December 2020), 180:1–15.

Jie Li, Gilles Daviet, Rahul Narain, Florence Bertails-Descoubes, Matthew Overby, George E Brown, and Laurence Boissieux. 2018. An implicit frictional contact solver for adaptive cloth simulation. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–15.

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. 2020a. Incremental Potential Contact: Intersection- and Inversion-free Large Deformation Dynamics. *ACM Trans. Graph.* 39, 4 (2020).

Minchen Li, Danny M. Kaufman, and Chenfanfu Jiang. 2021. Codimensional Incremental Potential Contact. *ACM Trans. Graph.* 40, 4, Article 170 (jul 2021), 24 pages.

Siwang Li, Jin Huang, Fernando de Goes, Xiaogang Jin, Hujun Bao, and Mathieu Desbrun. 2014. Space-time editing of elastic motion through material optimization and reduction. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.

Hsueh-Ti Derek Liu, Jiayi Eris Zhang, Mirela Ben-Chen, and Alec Jacobson. 2021. Surface Multigrid via Intrinsic Prolongation. *ACM Trans. Graph.* 40, 4 (2021).

Mickaël Ly, Jean Jouve, Laurence Boissieux, and Florence Bertails-Descoubes. 2020. Projective dynamics with dry frictional contact. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 57–1.

P-L Manteaux, Christopher Wojtan, Rahul Narain, Stéphane Redon, François Faure, and M-P Cani. 2017. Adaptive physically based models in computer graphics. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 312–337.

Eder Miguel, Derek Bradley, Bernhard Thomaszewski, Bernd Bickel, Wojciech Matusik, Miguel A Otaduy, and Steve Marschner. 2012. Data-driven estimation of cloth simulation models. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library.

Matthias Müller and Nuttapong Chentanez. 2010. Wrinkle Meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Madrid, Spain) *(SCA '10)*. 85–92. https://doi.org/10.2312/SCA/SCA10/085-091

Rahul Narain, Tobias Pfaff, and James F. O'Brien. 2013. Folding and Crumpling Adaptive Sheets. *ACM Trans. Graph.* 32, 4, Article 51 (jul 2013), 8 pages.

Rahul Narain, Armin Samii, and James F. O'Brien. 2012. Adaptive Anisotropic Remeshing for Cloth Simulation. *ACM Trans. Graph.* 31, 6, Article 152 (nov 2012), 10 pages.

Young Jin Oh, Tae Min Lee, and In-Kwon Lee. 2018. Hierarchical cloth simulation using deep neural networks. In *Computer Graphics International*.

Michael Ortiz and Laurent Stainier. 1999. The variational formulation of viscoplastic constitutive updates. *CMAME* 171, 3-4 (1999).

Miguel Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus Gross. 2009. Implicit Contact Handling for Deformable Objects. *Comp. Graph. Forum* 28 (04 2009).

Jovan Popović, Steven M. Seitz, and Michael Erdmann. 2003. Motion sketching for control of rigid-body simulations. *ACM Transactions on Graphics* 22, 4 (Oct. 2003), 1034–1054.

Olivier Rémillard and Paul G. Kry. 2013. Embedded thin shells for wrinkle simulation. *ACM Trans. Graph.* 32, Article 82 (July 2013), 8 pages. Issue 4. https://doi.org/10.1145/2461912.2462018

Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Alla Sheffer. 2010. Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-Looking Wrinkles. *ACM Trans. Graph.* 29, 6, Article 157 (dec 2010), 8 pages.

Igor Santesteban, Miguel A. Otaduy, and Dan Casas. 2019. Learning-Based Animation of Clothing for Virtual Try-On. *Computer Graphics Forum* 38, 2 (2019), 355–366. https://doi.org/10.1111/cgf.13643

Nikolas Schmitt, Martin Knuth, Jan Bender, and Arjan Kuijper. 2013. Multilevel Cloth Simulation using GPU Surface Sampling. *VRIPHYS* 13 (2013), 1–10.

Christian Schulz, Christoph von Tycowicz, Hans-Peter Seidel, and Klaus Hildebrandt. 2014. Animating deformable objects using sparse spacetime constraints. *ACM Transactions on Graphics* 33, 4 (July 2014), 109:1–109:10.

Martin Seiler, Jonas Spillmann, and Matthias Harders. 2012. Enriching Coarse Interactive Elastic Objects with High-Resolution Data-Driven Deformations. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*. https://doi.org/10.2312/SCA/SCA12/009-017

Andrew Selle, Jonathan Su, Geoffrey Irving, and Ronald Fedkiw. 2008. Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. *IEEE transactions on visualization and computer graphics* 15, 2 (2008), 339–350.

SideFX. 2024. *Houdini Vellum*. https://www.sidefx.com/products/houdini/

Rasmus Tamstorf and Eitan Grinspun. 2013. Discrete bending forces and their Jacobians. *Graphical models* 75, 6 (2013), 362–370.

Rasmus Tamstorf, Toby Jones, and Stephen F McCormick. 2015. Smoothed aggregation multigrid for cloth simulation. *ACM Trans. Graph. (TOG)* 34, 6 (2015), 1–13.

Min Tang, Ruofeng Tong, Rahul Narain, Chang Meng, and Dinesh Manocha. 2013. A GPU-based streaming algorithm for high-resolution cloth simulation. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 21–30.

Min Tang, Huamin Wang, Le Tang, Ruofeng Tong, and Dinesh Manocha. 2016. CAMA: Contact-aware matrix assembly with unified collision handling for GPU-based cloth simulation. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 511–521.

Min Tang, Tongtong Wang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. 2018. I-Cloth: Incremental Collision Handling for GPU-Based Interactive Cloth Simulation. *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia)* 37, 6 (November 2018), 204:1–10.

Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically Deformable Models. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. 205–214.

Nobuyuki Umetani, Danny M. Kaufman, Takeo Igarashi, and Eitan Grinspun. 2011. Sensitive Couture for Interactive Garment Modeling and Editing. *ACM Trans. Graph.* 30, 4, Article 90 (jul 2011), 12 pages.

Pascal Volino and N Magnenat Thalmann. 2000. Implementing fast cloth simulation with collision response. In *Proceedings Computer Graphics International 2000*. IEEE.

Etienne Vouga. 2024. libshell. https://github.com/evouga/libshell.

Etienne Vouga, David Harmon, Rasmus Tamstorf, and Eitan Grinspun. 2011. Asynchronous variational contact mechanics. *CMAME* 200, 25-28 (2011).

Huamin Wang. 2021. GPU-Based Simulation of Cloth Wrinkles at Submillimeter Levels. *ACM Trans. Graph.* 40, 4, Article 169 (jul 2021), 14 pages.

Huamin Wang, Florian Hecht, Ravi Ramamoorthi, and James F. O'Brien. 2010. Example-Based Wrinkle Synthesis for Clothing Animation. *ACM Trans. Graph.* 29, 4, Article 107 (July 2010), 8 pages. https://doi.org/10.1145/1778765.1778844

Zhendong Wang, Longhua Wu, Marco Fratarcangeli, Min Tang, and Huamin Wang. 2018. Parallel Multigrid for Nonlinear Cloth Simulation. *Computer Graphics Forum* (2018).

Clarisse Weischedel. 2012. *A discrete geometric view on shear-deformable shell models*. PhD dissertation. Georg-August-Universität Göttingen.

Andrew Witkin and Michael Kass. 1988. Spacetime Constraints. In *Computer Graphics (Proceedings of SIGGRAPH 88)*. 159–168.

Longhua Wu, Botao Wu, Yin Yang, and Huamin Wang. 2020. A Safe and Fast Repulsion Method for GPU-Based Cloth Self Collisions. *ACM Trans. Graph.* 40, 1, Article 5 (dec 2020), 18 pages.

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).

Juyong Zhang, Yue Peng, Wenqing Ouyang, and Bailin Deng. 2019. Accelerating ADMM for Efficient Simulation and Optimization. *ACM Trans. Graph.* 38, 6, Article 163 (nov 2019), 21 pages.

Jiayi Eris Zhang, Seungbae Bang, David IW Levin, and Alec Jacobson. 2020. Complementary Dynamics. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–11.

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. https://doi.org/10.1145/3550454.3555510

Meng Zhang, Tuanfeng Wang, Duygu Ceylan, and Niloy J Mitra. 2021. Deep detail enhancement for any garment. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 399–411.