# Progressive Dynamics++: A Framework for Stable, Continuous, and Consistent Animation Across Resolution and Time

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

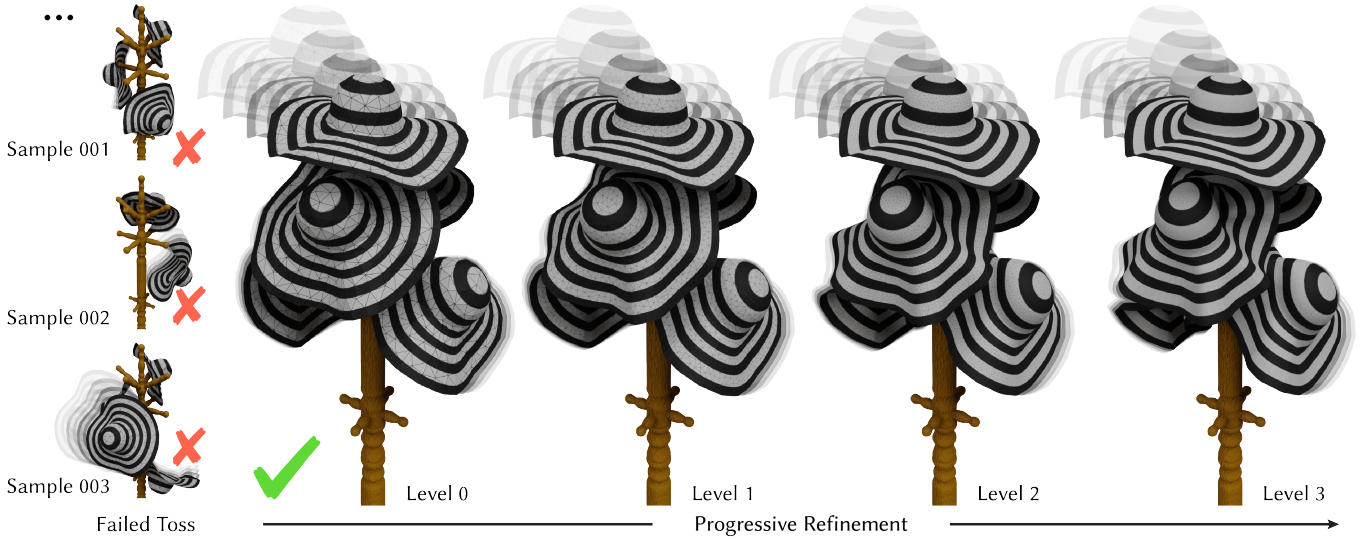



Fig. 1. **Designing a five-hat-trick animation with Progressive Dynamics:** Tossing a floppy hat onto a coat rack is hard, but adjusting initial conditions to successfully throw five hats is nearly impossible when using slow-to-compute, direct, fine-level simulations, due to tricky dynamics, sensitive collisions, and frictional contact. However, using our new Progressive Dynamics++ framework we can quickly explore a large number of coarse preview simulations (Left) to rapidly find a sample that makes the toss work. Once we find this rare, needle-in-a-haystack scenario, with our coarse simulation (Level 0), where all the hats magically fit and delicately settle on the hooks and one another, Progressive Dynamics++ then progressively (Levels 1-3) synthesizes production-level animations with millions of fine-scale vertices, capturing intricate physical details and, most importantly, maintaining consistent five-hat-trick outcomes.

The recently developed Progressive Dynamics framework [Zhang et al. 2024] addresses the long-standing challenge in enabling rapid iterative design for high-fidelity cloth and shell animation. In this work, we identify fundamental limitations of the original method in terms of stability and temporal continuity. For robust progressive dynamics simulation we seek methods that provide: (1) stability across all levels of detail (LOD) and timesteps, (2) temporally continuous animations without jumps or jittering, and (3) user-controlled balancing between geometric consistency and enrichment at each timestep, thereby making it a practical previewing tool with high-quality results at the finest level to be used as the final output.

We propose a general framework, Progressive Dynamics++, for constructing a family of progressive dynamics integration methods that advance physical simulation states forward in both time and spatial resolution, which includes Zhang et al. [2024]'s method as one member. We analyze necessary stability conditions for Progressive Dynamics integrators and introduce a novel, stable method that significantly improves temporal continuity, supported by a new quantitative measure. Additionally, we present a quantitative analysis of the trade-off between geometric consistency and enrichment, along with strategies for balancing between these aspects in transitions across resolution and time.

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


0730-0301/2025/8-ART53 $15.00
https://doi.org/10.1145/3731202

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

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

**ACM Reference Format:**
Jiayi Eris Zhang, Doug L. James, and Danny M. Kaufman. 2025. Progressive Dynamics++: A Framework for Stable, Continuous, and Consistent Animation Across Resolution and Time. *ACM Trans. Graph.* 44, 4, Article 53 (August 2025), 20 pages. https://doi.org/10.1145/3731202

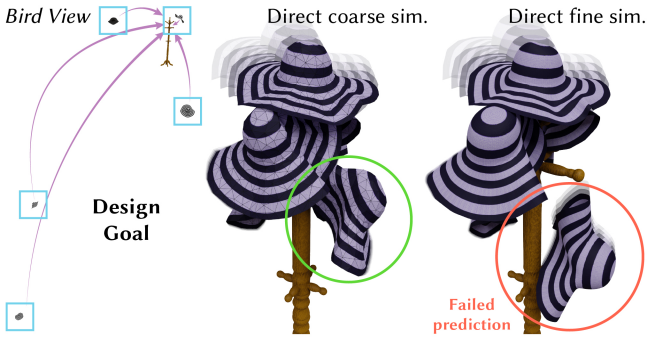



Fig. 2. **Hat Trick Challenges for Direct Simulation:** (Left) Reversing ballistic motion gives rough estimates of initial conditions for these hats' approximate motions and impact times. (Middle) Using direct simulation, fast coarse-scale deformable time stepping allows further fine-tuning of initial conditions and material properties to achieve the five-hat-trick design goal. Unfortunately, direct simulation methods break the consistency with mesh refinement. (Right) This generates incorrect final, fine-mesh animation results that no longer satisfy design goals, with hats sliding off hooks.

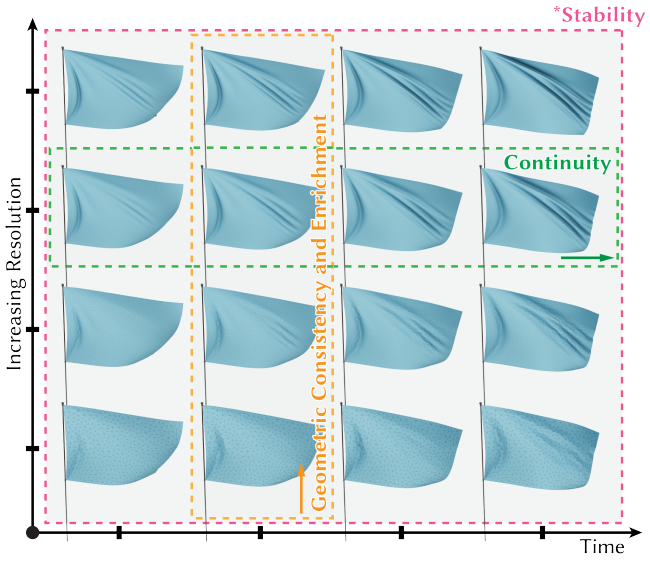



Fig. 3. **Illustration of Stability, Continuity, Consistency, and Enrichment:** Using a waving flag example, we visualize how a well-designed Progressive Dynamics integrator behaves across both resolution and time. Vertically (within each column, from bottom to top), the flag shape is progressively enriched with high-frequency detail while preserving geometric consistency—i.e., the coarse-level shape approximates the bulk deformation of finer levels without drift or divergence. Horizontally (across each row, from left to right), the animation evolves smoothly over time without jumps, jittering, or nonphysical artifacts. Throughout the space-time grid, all states remain stable—free from jitters or explosions—highlighting robust integration across all levels of detail (LOD) and time steps.

## 1 INTRODUCTION

The recently developed Progressive Dynamics framework [Zhang et al. 2023, 2022, 2024] addresses the long-standing challenge in enabling rapid iterative design for high-fidelity cloth and shell animation. In this work, we identify fundamental limitations of the original method in terms of stability and temporal continuity. For framerate animations (e.g., 24 FPS), Zhang et al. [2024] apply small numbers of levels (e.g., 2–4), which are generally sufficient for Progressive Dynamics. However, as animators seek to capture smaller timestep, higher speed animations, the number of levels in the LOD necessarily must correspondingly increase (for sufficient "vertical" resolution) to obtain comparable enrichment from coarse to fine. Unfortunately, in exactly such cases, with larger numbers of levels, we observe that Zhang et al.'s Progressive Dynamics method can become unstable, with jittering artifacts and instabilities.

In this work, we advocate for a robust Progressive Dynamics framework that enables users to progressively design animations by exploring variations in design parameters without restriction. Once a target coarse-level animation design is identified, it should be progressively refined with *temporal continuity*, ensuring that, at any timestep, results across *all* levels remain stable and physically realistic, and with *geometric consistency*, so that as animations progress, they only differ in *enrichment* details. Ultimately, the finest-level output animation, and ideally all prior preview animations at intermediary levels, should all give temporally continuous, artifact-free, and increasingly high-fidelity animations as output. To this end, we define the following desirable properties (see Figure 3):


**Stability** ensures simulations remain numerically stable not only over time, as in standard time integration, but also across resolution levels. This stability criterion must hold across all timesteps and all levels of detail (LOD).

**Temporal Continuity** requires physically continuous animations over time, at each level, free from jumps, jittering, or nonphysical abrupt changes, thereby ensuring natural and realistic transitions between frames.

**Geometric consistency** requires that each coarser-level timestep captures the bulk deformation of the corresponding finer-level timestep, with finer levels contributing only high-frequency details (see below) that do not alter coarse-scale behavior.

**Enrichment** is the progressive enhancement of simulations through the addition of increasingly fine-scale geometric and dynamic details at higher resolution levels.

**Controllability** is the ability to provide users with explicit control over the trade-off between geometric consistency and enrichment—two inherently connected but often competing objectives—during level-of-detail transitions across both resolution and time.

Toward these goals, we introduce Progressive Dynamics++ (*PD++*), a general framework for constructing Progressive Dynamics integrators that step through both time and spatial resolutions. Although effective in many cases, the original Progressive Dynamics integration method proposed by Zhang et al. [2024]—referred to throughout

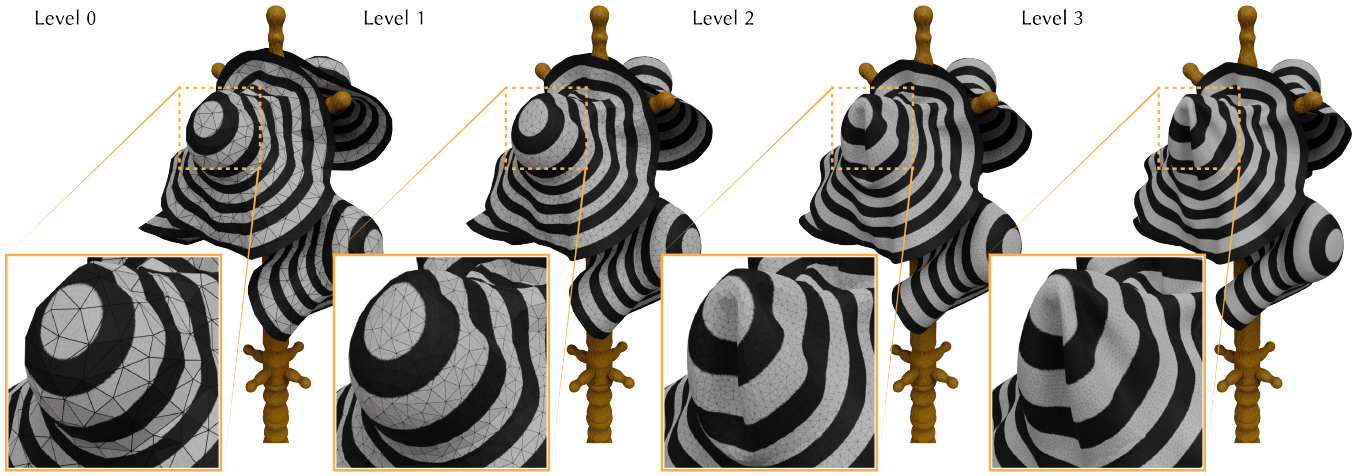



Fig. 4. **Hat Detail Enrichment:** Progressive Dynamics++ progressively synthesizes production-quality animations with millions of fine-scale vertices, capturing intricate detail and consistently achieving the same five-hat-trick variation across all levels. Built on our Progressive Dynamics++ framework, our *VelPro* method delivers stable, continuous, artifact-free animation results that show increasingly detailed collisions and deformations, refining from coarse to fine levels. In contrast, the semi-diagonal method from Zhang et. al. [2024] generates jittering and instability artifacts in this delicate example.

this paper as the **semi-diagonal** integrator (named after its integration path)—does not fully satisfy these properties. Specifically, we identify critical limitations in the stability and temporal continuity of this resolution-integration method. Likewise, additional improvements in enrichment and consistency are desirable. As we shall see, the method proposed by Zhang et al. [2024] represents a specific instance within a much broader family of possible PD++ constructions. This opens the door to significantly improved Progressive Dynamics animation generation. Concretely, our work here constructs and demonstrates two new specific Progressive Dynamics integration methods in this family: the **full-diagonal** method (named after its integration path) and the velocity-prolongation method (abbreviated as ***VelPro*** integrator), with the latter, as we will see, especially effective and emphasized as a primary focus in this work; see Section 4 for details. Throughout, we color-code animation results for clarity: ■ **direct** simulation generated, and ■ **semi-diagonal** , ■ **full-diagonal** , and ■ ***VelPro*** integration for progressive dynamics.

In this work, we first qualitatively demonstrate that Zhang et al.'s [2024] method becomes unstable with increasing levels, exhibiting jittering artifacts and instabilities. To quantitatively evaluate the above properties in the PD++ framework, we introduce two new metrics: one to measure *temporal continuity* within each level and another to assess *geometric consistency* across resolution levels at each time step. Together, these metrics enable comprehensive benchmarking of both the Progressive Dynamics methods evaluated in this paper and future developments in Progressive Simulation. With these metrics in hand, we also investigate how the balance between consistency and enrichment can additionally be further controlled with the application of a soft quadratic regularization term.

*Summary of Technical Contributions.* We propose a general framework, Progressive Dynamics++, for constructing a family of progressive dynamics integration methods that advance physical simulation states forward in both time and spatial resolution, which includes Zhang et al. [2024]'s method as one member. We analyze the stability conditions necessary for Progressive Dynamics integrators and introduce a novel and stable *VelPro* method that significantly improves temporal continuity, supported by a new quantitative measure. Additionally, we present a quantitative analysis of the trade-off between geometric consistency and enrichment, along with strategies for balancing these aspects in level-of-detail transitions across resolution and time. In addition to theoretical analysis, we conduct extensive numerical evaluations and comparisons across diverse examples to support our claims and demonstrate the properties of each method.

## 2 RELATED WORK

Since our Progressive Dynamics++ framework builds upon and improves the original Progressive Dynamics, as in Zhang et al. [2024], we refer readers to the comprehensive list of related work provided there. Here, we focus on the most relevant aspects of our Progressive Dynamics++ integration framework.

### 2.1 Progressive Simulation

The recent progressive simulation framework proposed by Zhang et al. [2023; 2022; 2024] addresses the long-standing challenge of enabling rapid iterative design for high-fidelity cloth and shell simulation. This framework facilitates a workflow that supports effective modeling and animation design cycles by generating coarse preview simulations that are predictive of, and closely match, subsequent refinements at finer resolutions. Specifically, Zhang et al. [2023; 2022] focused on quasistatic simulations that enable interactive exploration of design parameters by generating coarse previews of

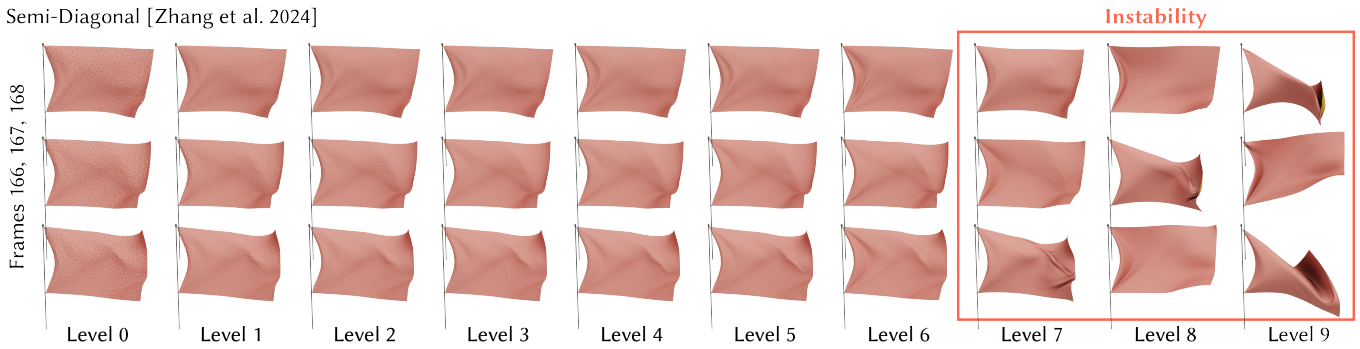



Fig. 5. **Exploding Flag:** Here we demonstrate instability in the semi-diagonal integration method [Zhang et al. 2024] with a wind-driven waving flag, generated with a ten-level hierarchy. We show computed frames 166 to 168, across resolutions from coarsest level (0) to the finest level (9). While levels 0–6 appear visually stable, significant and obvious instabilities emerge at level 7 and progressively worsen, resulting in severe jittering, nonphysical deformation, and distorted flag shapes by level 9.

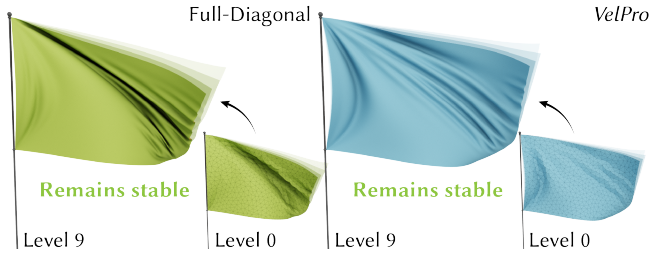



Fig. 6. **Stable Flags:** In the same waving-flag setup as Figure 5 above, we evaluate long-term consistency under sustained, highly dynamic motion. Here, our newly proposed full-diagonal and *VelPro* integrators both remain stable and produce visually artifact-free results throughout the 20-second animation.

cloth (via a coarse-to-fine hierarchy) and shell (via a fine-to-coarse hierarchy) equilibrium states for static modeling. These previews form a sequence of increasingly refined solutions that converge to an accurate high-fidelity C-IPC simulation of the final cloth drape or shell configuration.

Zhang et al. [2024] extend the progressive framework to dynamic cloth and shell animations, via the rapid generation of coarse animations that progressively and predicatively refine toward high-fidelity results at the finest level. However, Progressive Dynamics (PD) in both this prior work and here focus on coarse-to-fine, level-of-detail (LOD) generation of increasingly plausible cloth and shell animations. Unlike quasistatic progressive simulation, PD does not produce cloth and shell dynamics that converge to a single, accurate finest-level simulation. Therefore, no ground-truth fine-level solution exists for direct frame-by-frame visual comparison with PD results.

Our work builds on and generalizes the Progressive Dynamics framework introduced by Zhang et al. [2024]. We propose a general framework for constructing progressive dynamics integrators, investigate key properties including stability, temporal continuity, geometric consistency, enrichment, and the controllability of their trade-offs, and introduce two specific instances that substantially improve stability and temporal continuity, supported by a new comprehensive theoretical analysis and new quantitative evidence.

## 2.2 Stability of Time Integrators in Graphics

Numerical time integration methods, particularly implicit integrators [Baraff and Witkin 1998; Bridson et al. 2002; Li et al. 2020, 2018; Narain et al. 2012; Otaduy et al. 2009; Tang et al. 2018], are fundamental in physics-based animation and simulation. Time integrators discretize steps in time to forward solve differential equations that govern evolving physical system dynamics. Explicit methods like Forward Euler and Verlet integration prioritize efficiency, but struggle with stability at large timesteps and with stiff systems. On the other hand, implicit methods such as backward Euler and BDF2 can provide robust timesteps with stability, albeit at the expense of computational overhead and numerical damping. Stability in these methods classically refers to bounding numerical error during time integration, ensuring stable dynamics over time, and robust numerical solutions, particularly in response to large timesteps or stiff systems.

Building on Progressive Dynamics' resolution-time integration, our work extends the analysis of stability from the classic temporal domain to jointly consider stability across both temporal *and* resolution dimensions. In addition to standard time integrators, where stability addresses the accumulation of integration error over time, our analysis focuses on stability as we traverse both resolution levels (or "vertical stability") and timesteps[1], examining how differences between coarse and fine solutions propagate and grow. To our knowledge, this is the first attempt to analyze stability in the progressive simulation framework, where the interaction between resolution and time introduces unique challenges in stability analysis that differ fundamentally from single-resolution problems.

[1]In this paper, we follow the convention from Zhang et al. [2024] that a "horizontal" direction represents time and "vertical" direction represents resolution.

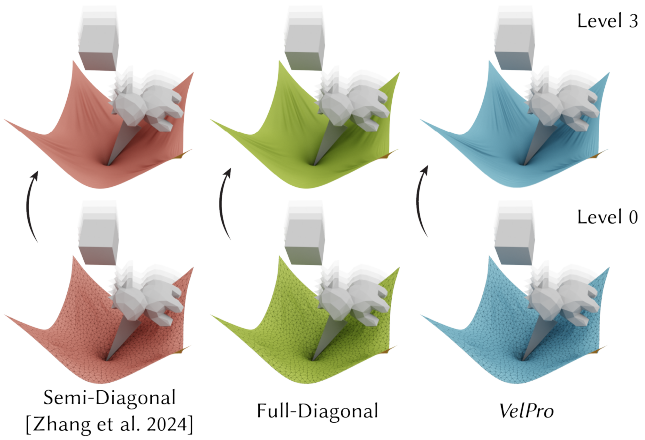



Fig. 7. **Stability with a Limited Number of Levels:** For large-timestep (e.g., 0.04s), lower frequency animations, Zhang et al. [2024] typically use a limited hierarchy of 2–4 levels, which is generally sufficient. We confirm that all three methods—semi-diagonal [Zhang et al. 2024], full-diagonal and *VelPro* integration methods—produce stable and visually plausible results in this regime.

## 2.3 Temporal Continuity

Ensuring temporal continuity between frames is a critical aspect in fields like optical flow estimation and video super-resolution. Optical flow methods such as PWC-Net [Sun et al. 2018] and RAFT [Teed and Deng 2020] utilize hierarchical, coarse-to-fine frameworks with iterative refinement and feature warping to maintain smooth and temporally coherent motion fields across consecutive frames. Similarly, video super-resolution techniques such as VSRNet [Kappeler et al. 2016] and EDVR [Wang et al. 2019] emphasize temporal alignment through motion compensation and attention mechanisms, ensuring that details are consistently preserved without introducing temporal artifacts. Recent methods [Xue et al. 2019] also integrate optical flow and super-resolution into a unified framework to improve temporal continuity. These approaches collectively highlight the importance of designing systems that preserve temporal continuity across frames. Although these approaches align with our goal of maintaining temporal continuity in multilevel simulations over time, the concept of temporal continuity in physics simulation extends far beyond the smoothness and coherence of the estimated motion field, as seen in contexts like optical flow. Although optical flow primarily focuses on the perception of motion between consecutive frames, physics simulations involve the accurate evolution of dynamic properties such as momentum, energy, and interaction forces over time. Temporal continuity in this context requires adherence to the governing physical laws, such as Newtonian mechanics and conservation principles, ensuring that animations evolve consistently without introducing significant artificial discontinuities or artifacts, and yet accounting for physically accurate jumps, e.g., due to impacts.

## 2.4 Consistency and Enrichment

Ensuring consistency while enriching animations is a critical challenge in physics-based simulation. Tracking-based enrichment methods, like TRACKS [Bergou et al. 2007] and Wrinkle Meshes [Müller and Chentanez 2010], use constraints to track coarse input animations and add fine details such as wrinkles. However, these methods often struggle to maintain consistency, as tracking constraints do not guarantee that coarse deformations are preserved in the fine-scale result [Bai et al. 2016], and coarse-level artifacts can propagate to finer scales, producing non-physical wrinkles [Chen et al. 2023]. Complementary enrichment approaches, such as Complementary Dynamics [Zhang et al. 2020], address some of these challenges by constraining enrichment to orthogonal subspaces, ensuring that added details respect the animator's intent. While effective, they face challenges in defining suitable orthogonal bases for shell dynamics and can fail to resolve interactions and self-contacts [Chen et al. 2021a]. Data-driven approaches, while promising, frequently encounter issues with artifacts and limited applicability beyond their training domains, which can compromise consistency and temporal coherence [Kim et al. 2013; Lahner et al. 2018]. These limitations highlight the need for methods that can maintain physical and temporal consistency while enriching with physical and geometric details. Given the high computational cost of direct fine-resolution cloth simulations, neural networks have become a popular direction for achieving high-resolution final simulation results by adding details to coarse simulations [Chen et al. 2021b; Halimi et al. 2023; Kavan et al. 2011; Lee et al. 2019; Oh et al. 2018]. For example, Yu and Wang [2024] integrate simulation and correction modules to generate high-resolution animations while maintaining spatial consistency and temporal coherence between frames. Similarly, Zhang and Li [2024] employ a lightweight learning-based approach to enhance high-frequency details in coarse garment simulations, which provides an efficient balance between computational cost and dynamic fidelity.

## 2.5 Parallel-in-Time Multigrid Methods

Parallel-in-Time (PinT) methods [Gander and Lunet 2024] are effective tools to accelerate time-dependent simulations by leveraging parallelism across temporal dimensions. Lions et al. [2001] introduce the concept of iterative coarse-to-fine corrections in time, demonstrating significant parallel speedups for moderate problem sizes. Extending this work, Multigrid Time Reduction in Time (MGRIT) [Friedhoff et al. 2013] formalizes a robust multilevel framework, offering improved scalability for large-scale simulations. While PinT multigrid methods incorporate the time dimension, aligning conceptually with our goal of addressing dynamic problems using a multiresolution hierarchy, they focus on accelerating convergence to a solution at the finest *time* resolution, similar to generic multiresolution frameworks for linear and nonlinear systems. In contrast, our work focuses on progressive simulation, where coarse-to-fine, level-of-detail solutions progressively refined *in space*, across levels of resolution, are designed to facilitate faster iterative design and exploration. Finally and importantly, unlike PinT methods, the Progressive Dynamics problem we address here also does not impose

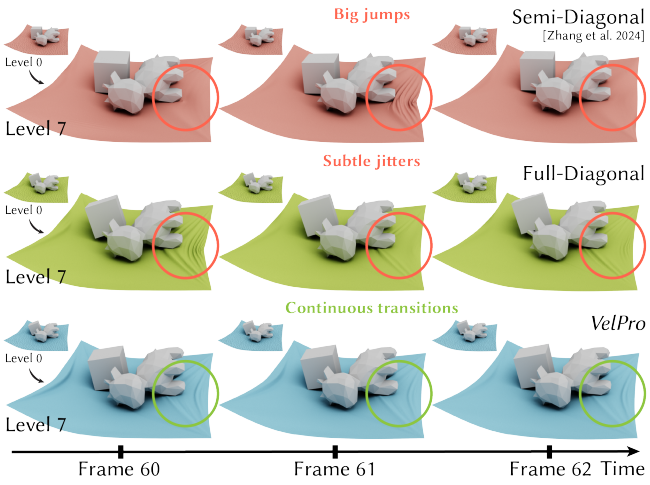



Fig. 8. **Instability with Increasing Levels:** Jumps and jittering issues in the semi-diagonal and full-diagonal integration methods emerge as the number of levels increases. Using an eight-level hierarchy (in contrast with four as in Figure 7), the semi-diagonal integration method shows severe instability, with significant jumps (e.g., frame 40), while the full-diagonal integration method exhibits subtle jitters due to its lack of continuity guarantees. In contrast, our proposed *VelPro* integration method ensures smooth and continuous simulations across all levels.

the constraint that trajectories must strictly match a converged numerical time-integration model at the finest level, which enables additional design freedoms.

## 3 BACKGROUND: PROGRESSIVE SIMULATION

Before introducing the Progressive Dynamics++ framework and analyzing the properties of the Zhang et al.'s [2024] original Progressive Dynamics method, we first introduce here necessary background and constructions for progressive simulation [Zhang et al. 2023, 2022, 2024]. We start with hierarchy construction and then cover direct, per-level time integration, the extension to progressive quasistatic simulation [Zhang et al. 2023, 2022], and finally detail progressive simulation's most recent extension to Progressive Dynamics [Zhang et al. 2024].

### 3.1 Multiresolution Time Integration

To progress from direct time integration to progressive simulation, we construct a multi-resolution mesh hierarchy consisting of triangle meshes with increasingly finer resolutions. We apply the decimation-based LOD construction method from Zhang et al. [2023], starting with high-quality, finest-level triangulations, and applying quadric error edge collapse [Garland and Heckbert 1997] with probabilistic quadrics [Trettner and Kobbelt 2020] for decimation. We refer to Zhang et al. [2023, 2022] for a detailed analysis of the impact of alternative LOD constructions on simulation quality. At each level of the hierarchy, we solve a specially constructed timestepping problem by independently minimizing a proxy energy (see next section) at each level using implicit integration methods. We index meshes in the hierarchy by resolution level, with subscripts $l \in \{0, 1, \ldots, L\}$. For level $l$, the undeformed (rest) and deformed positions of the mesh vertices are $\bar{x}_l$, $x_l \in \mathbb{R}^{3n_l}$, respectively, where $n_l$ is the number of vertices at that level $l$.

For frictionally contacting cloth and shells subject to imposed boundary conditions and external forces, we model each simulation mesh[2] with potential energies for shell elasticity ($\Psi$), contact barriers ($B$), friction ($D$), and, when necessary, strain limiting ($S$). These potentials contribute to the total potential energy at level $l$, given by $E_l = \Psi_l + B_l + D_l + S_l$. *Direct* per-level timestep solves with implicit Euler method then minimize the incremental potential (IP) energy

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

where $h$ is timestep size, $M_l$ is the level-$l$'s mass matrix, and $E_l$ is the total potential energy, including contributions from body and external forces. We use the notation $||x||_M = x^\top Mx/2$ throughout.

### 3.2 Proxy Energies and Prolongations

Next, we detail here the construction of the proxy energies applied to step progressive simulations. Instead of solving each level's standard IP problem independently, as in the direct per-level time-stepping in Equation 1 above, progressive simulation methods first introduce (see Zhang et al. [2023; 2022]) prolongation operators $P_{l+1}^l(\cdot)$. These operators map nodal positions and surface quantities from current levels $l$, to next levels, $l+1$, and are precomputed during mesh hierarchy construction. At each coarsened level $l < L$, the progressive simulation framework computes successively refined approximations of the final equilibrium geometry by minimizing a *proxy* for the finest-level potential energy:

$$F_l(x_l) = \underbrace{B_l(x_l) + D_l(x_l) + S_l(x_l)}_{C_l(x_l)} + \Psi_L\big(P^l(x_l)\big). \tag{2}$$

Unlike the direct simulation's total potential, $E_l$, defined above, the progressive simulation proxy potential, $F_l$, allows coarsened levels to directly evaluate shell elastics at the finest resolution, $\Psi_L$, by direct prolongation, $P^l(x_l)$, from current level $l$, to the finest mesh, while efficiently enforcing contact and strain limit constraints on the coarse geometry through barriers-based terms $C_l(x_l)$. This mitigates membrane-locking artifacts that direct solves would generate at coarser levels, and promotes consistent simulation behaviors across resolutions. For quasistatics, a progressive solver (detailed further below in Section 3.3) minimizes this proxy energy to equilibrium at each level. Solution's at each level are then safely initialized and advanced to the next finer level [Zhang et al. 2023, 2022]. This process is repeated until convergence at the finest level gives the final simulation result.

### 3.3 Progressive Quasistatics and Dynamics

In analogy to Equation 1 above, progressive simulation methods step, for all levels $l+1 \in \{0, \ldots, L\}$, by minimizing a modified IP

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

using the proxy potential

$$x_{l+1}^{t+1} = \underset{x}{\operatorname{argmin}} \frac{1}{2h^2}\|x - \hat{x}_{l+1}^t\|_{M_{l+1}}^2 + C_{l+1}(x) + \Psi_L\big(P^{l+1}(x)\big). \quad (3)$$

Recall that for the finest-resolution level, $L$, the prolongation, $P^L$, is just the identity. Then, solely at the finest level, the above IP solve is identical to Equation 1.

For Progressive Quasistatic simulation, Zhang et al. [2023, 2022] repeatedly step the above modifed IP in Equation 3, at each level with $\hat{x}_{l+1}^t = x_{l+1}^t$. Initializing each level $l+1$'s solves with the last level $l$'s prolonged solution, this artificial timestepping process continues at each level until the system reaches equilibrium.

To then extend progressive simulation to animating dynamics, Progressive Dynamics [Zhang et al. 2024] considers progressive refinement across spatial resolution (as in Zhang et al. [2023; 2022]) and forward dynamics over time. The full progressive simulation state is now defined on a space-time multiresolution grid, with spatially discretized positions $x_l^t$, and velocities $v_l^t$ defined at each grid point $(t, l)$, corresponding to the timestep $t \in \{0, 1, \ldots, N\}$ and the resolution level $l \in \{0, 1, \ldots, L\}$.

For all coarsened levels $l+1 < L$, Progressive Dynamics timestepping is then likewise solved with Equation 3. Here a key contribution of Zhang et al. [2024] is the construction of the $\hat{x}_l^t$ term to balance temporal continuity with frame-wise geometric consistency. At the coarsest level, $(l+1 = 0)$, Zhang et al.'s [2024] semi-diagonal method performs forward timestepping by solving the prolonged IP problem (Equation 3) with the velocity update $\hat{x}_0^t = x_0^t + hv_0^t$. This generates all preview states $x_0^t, v_0^t$, for all $t \in \{0, 1, \ldots, N\}$ and so populates the bottom row of our Progressive Dynamics solution grid. Then, for finer levels, $l+1 > 0$, the semi-diagonal method computes velocity updates prolonged from the previous level for all $t+1 \in \{1, 2, \ldots, N\}$. Specifically (following implicit Euler) velocities are defined at each prior level as $v_l^t \leftarrow (x_l^t - x_l^{t-1})/h$. The prolonged velocity updates are then constructed as:

$$\begin{aligned} \hat{x}_{l+1}^t &= P_{l+1}^l(x_l^t) + h\big(V_{l+1}^l(x_l^t)\big)v_l^t \\ &= P_{l+1}^l(x_l^t) + \big(V_{l+1}^l(x_l^t)\big)(x_l^t - x_l^{t-1}). \end{aligned} \quad (4)$$

where $P_{l+1}^l$ and $V_{l+1}^l = \nabla P_{l+1}^l$ are the PSQ shell-prolongation operators [Zhang et al. 2023] for positions and velocities respectively. Progressive Dynamics then computes the timestep advancement for each new level at grid points $(t+1, l+1)$ using the progressive IP solve in Equation 3.

## 4 PROGRESSIVE DYNAMICS++ FRAMEWORK

We now generalize Zhang et al.'s [2024] Progressive Dynamics method with a broadened framework, Progressive Dynamics++, for constructing a range of progressive simulation integrators that step through both time and spatial resolutions. To do so we focus on a family of progressive advancement integrations in the resolution-time grid, by exploring variations in the construction of the prolonged term $\hat{x}_{l+1}^t$, in Equation 3. Recall that Zhang et al.'s [2024] semi-diagonal method constructs this term with a velocity update using jointly prolonged prior positions *and* prior velocities from the previous level (see Equation 4). Of course this is just one potential choice. Progressive Dynamics++ instead considers the range of Progressive Dynamics integrators parameterized by

$$\hat{x}_{l+1}^t = \bar{x}_{l+1}^t + h\bar{v}_{l+1}^t, \quad (5)$$

where $\bar{x}_{l+1}^t$ and $\bar{v}_{l+1}^t$ are opened to all choices for *approximations of* the converged positions and velocities $(x_{l+1}^t, v_{l+1}^t)$ at each time $t$, and level $l+1$, in the resolution-time grid. While broader families of Progressive Dynamics integrators are certainly possible (and of interest for future investigation), as we will see below, this simple generalization enables the careful analysis of Progressive Dynamics properties across existing methods, and the construction of new, improved Progressive Dynamics integration methods.

Concretely, as we vary how $\bar{x}_{l+1}^t$ and $\bar{v}_{l+1}^t$ are constructed by choices of prolongations and prior states available in the Progressive Dynamics resolution-time grid we can retrieve prior integration methods and design new alternatives.

In terms of retrieving prior methods, the semi-diagonal method is easily built as a specific instance within this family by choosing $\bar{x}_{l+1}^t = P_{l+1}^l(x_l^t)$ and $\bar{v}_{l+1}^t = \big(V_{l+1}^l(x_l^t)\big)(x_l^t - x_l^{t-1})/h$. Likewise, as an even simpler case, direct per-level simulation (at each level $l+1$) just takes exact values of $\bar{x}_{l+1}^t = x_{l+1}^t$ and $\bar{v}_{l+1}^t = v_{l+1}^t$[3].

In terms of designing new, improved methods, as we will analyze in the following sections, it is important to note that stability, vertical geometric consistency, and temporal continuity all heavily depend on careful construction of $\hat{x}_{l+1}^t$. In turn, this requires appropriately incorporating lagged "prior" states from coarser levels and prior timesteps. As a preview, we apply the Progressive Dynamics++ framework to construct a new, stable Progressive Dynamics method, the VelPro integrator[4], which significantly improves temporal continuity and comes with unconditional stability guarantees in the linear setting. The update for *VelPro* is constructed with

$$\hat{x}_{l+1}^t = x_{l+1}^t + P_{l+1}^l(x_l^t - x_l^{t-1}).$$

See Section 5.4.2 for further details.

For additional analysis and exercise of Progressive Dynamics++ framework, we also construct the full-diagonal integration method using the update

$$\hat{x}_{l+1}^t = P_{l+1}^l(2x_l^t - P_l^{l-1}x_{l-1}^{t-1}),$$

for $l \in \{1, 2, \ldots, L\}$, and $\hat{x}_1^t = P_1^0(2x_0^t - x_0^{t-1})$. For comparison, while the full-diagonal method is stable, as we will see below, it can generate animations that break temporal continuity. See Section 5.4.1 for further details.

Finally to equip the Progressive Dynamics++ framework with better mechanisms for analyzing targeted properties and their tradeoffs across Progressive Dynamics integrators we additionally propose two new metrics to quantitatively evaluate their temporal continuity (see Section 6) and geometric consistency (see Section 7).

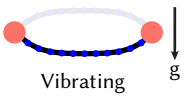
## 5 STABILITY

We begin by analyzing the occurrence of instabilities in Zhang et al.'s [2024] semi-diagonal method, and next systematically develop a theory to explain their underlying causes. Throughout, we base our

[3] Of course this construction quickly leads to divergence solutions across levels.
[4] As this update prolongs only the prior level's velocity, $v_l^t = (x_l^t - x_l^{t-1})/h$, while using position, $x_{l+1}^t$, from the current level, we call it the *VelPro* integrator.

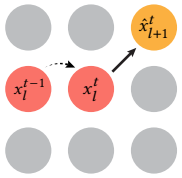
analysis on the implicit Euler integration model employed. Across a wide range of examples, we observe similar behavior as reported in Section 5.3 of Zhang et al. [2024] for changing level numbers. Specifically, for large-timestep (e.g., 0.04s), lower frequency animations, small numbers of levels (e.g., 2-4) are generally sufficient for Progressive Dynamics. Although Zhang et al. [2024]'s method then performs well in this regime (e.g., see Figure 7), capturing higher-speed, small-timestep animations demands more LOD levels to maintain sufficient vertical resolution and enrichment. In these cases, we observe that Zhang et al.'s [2024] method can generate severe jittering and explosion artifacts, with instability worsening as the number of levels increase (see Figures 5, 8, 11, and 12). These examples highlight a critical stability issue with Zhang et al.'s [2024] method when many levels are used. We note that, empirically, the threshold at which these instabilities and artifacts emerge is highly condition-dependent, making it difficult to predetermine when animations are going to exhibit instabilities and so give unusable results.

## 5.1 Test Model Problem (2D Mass-Spring System)




We begin by applying Zhang et al.'s [2024] semi-diagonal method to a simple 2D example consisting of a mass-spring chain, with both ends pinned under gravity, to clearly illustrate its observed instability behaviors (see inset). With a few levels (less than ten in this case), the semi-diagonal method performs as expected, consistent with the results reported in Zhang et al. [2024]. However, further increasing the total number of levels results in explosion artifacts similar to those observed in 3D examples. In Figure 9 we see that once instability artifacts appear, characterized here by distorted deformation shapes, they propagate diagonally across both timesteps *and* levels. Here we also note that the scale of this growing distortion roughly doubles over time (see red arrows in Figure 9).

## 5.2 Source of Instability




We next analyze the source of the above instabilities in the semi-diagonal method. For the coarsest-level simulation, we follow Section 3.3 to generate a full-sequence preview animation, $\{x_0^0, x_0^1, x_0^2, ..., x_0^N\}$, using prolonged IP timestepping. For progressive refinement to finer levels, we first consider a simplified setup for the prolonged diagonal momentum update at grid points $(t+1, l+1)$, defined as

$$\begin{aligned}\hat{x}^t_{l+1} &= P^l_{l+1}(x^t_l) + h(V^l_{l+1}(x^t_l))v^t_l \\ &= P^l_{l+1}(2x^t_l - x^{t-1}_l).\end{aligned} \tag{6}$$

Here, notice that we simplify by treating $P^l_{l+1}$ as a linear operator, using only its intrinsic component [Zhang et al. 2023], so that $V^l_{l+1}(x) = \nabla P^l_{l+1}(x) = P^l_{l+1}(x)$. In the above, we also continue to use the finite-difference stencil for implicit Euler velocity updates, with $v^t_l = (x^t_l - x^{t-1}_l)/h$. The progressive IP timestep solve then follows Equation 3.

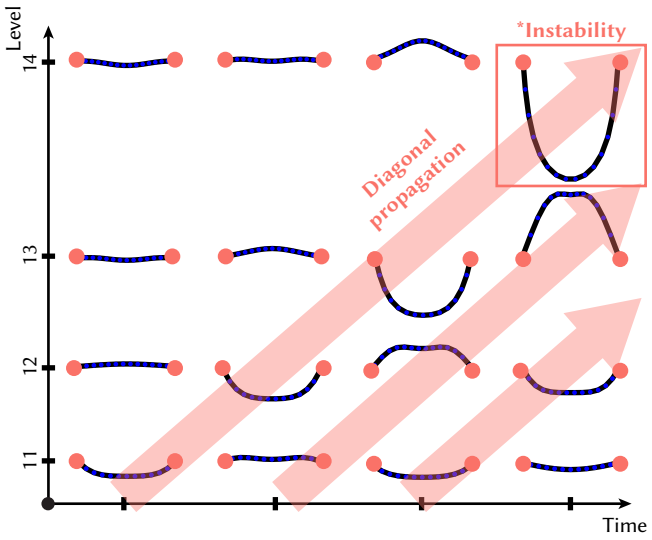



Fig. 9. **The semi-diagonal Method's Instability in a 2D Mass-Spring Example:** A simple 2D mass-spring example reveals that, for the semi-diagonal method, as the number of levels increases, instability artifacts start to appear in the form of exaggerated deformations that propagate diagonally across both timesteps and resolution levels. The red arrows highlight how the scale of these shapes grows roughly twofold over time.

The aforementioned explosion artifacts in Zhang et al. [2024] arise from the accumulation and propagation of discrepancies between levels during time integration. Specifically, these discrepancies grow exponentially over successive timesteps. To characterize this, we define the discrepancy at level $l$ as

$$\delta_l = P^{l-1}_l x^t_{l-1} - x^t_l.$$

After $n$ timesteps at level $l+n$, this discrepancy propagates from level $l$ as $\delta_{l+n} \approx 2^n P^l_{l+n}\delta_l$, thus amplifying exponentially. We also recall from Zhang et al. [2023] that each linear term of the prolongation, $P^l_{l'}$, $l < l'$, has non-negative entries with row sums equal to one. Here, we focus our stability proofs on the linear setting and then empirically demonstrate (across all examples, including extreme cases of 16 levels and large time steps of 0.04s) in our validation that these results hold in the general nonlinear case with nonlinear prolongation operators and force contributions.

## 5.3 Theoretical Results: Exponential Growth of $\delta_l$

Above we built intuition for the source of instability in Zhang et al.'s [2024] progressive advancement. We now formally present our main results on the exponential growth of the level discrepancy $\delta_l$ in this linear setting. Detailed proofs are provided in Appendix A.

Theorem 1. *Assume that each $P^l_{l+1}$ has full rank. For multi-level diagonal timestepping, if $x^t_l = P^{l-1}_l(2x^{t-1}_{l-1} - x^{t-2}_{l-1})$ with boundary values $x^t_0, x^0_l, v^0_l, \forall t, l$, then $x^t_l = P^{l-n}_l((n+1)x^{t-n}_{l-n} - nx^{t-n-1}_{l-n})$ $\forall t, l, n$ **if and only if** $x^t_{l+1} = P^l_{l+1}x^t_l$ $\forall t, l$.*[5]

[5]Throughout this work, "$\forall t, l$" always means for all $t \in \{0, 1, \ldots, N\}$ and $l \in \{0, 1, \ldots, L\}$. Positive integers $L, N$ are considered fixed. In this statement, "$\forall n$" means for all positive integers $n$ such that the vector $x$ is always well defined.

The diagonal stepping regime, $x_l^t = P_l^{l-1}(2x_{l-1}^{t-1} - x_{l-1}^{t-2})$, and boundary values uniquely determine all values $x_l^t$ on the resolution-time grid. Theorem 1 states that solutions across different levels are direct prolongations of one another (from coarse to fine) and are thus absolutely consistent if and only if, for any level $l$, the simulation is performed using standard single-level timestepping and all finer-level solutions can be obtained by directly prolonging results from that level. Please see Appendix A for a detailed discussion.

Theorem 2. *Assume that multi-level diagonal timestepping is applied with $x_l^t = P_l^{l-1}(2x_{l-1}^{t-1} - x_{l-1}^{t-2})$, with given boundary conditions $x_0^t, x_l^0, v_l^0, \forall t, l$ and $t_0, l_0$ with $t_0 > 0$ such that:*

- $x_{l+1}^t = P_{l+1}^l x_l^t$ *for* $t \in \{t_0 - 1, t_0\}$ *and any* $l$, *except when* $(t, l) = (t_0, l_0)$;
- $x_{l_0+1}^{t_0} + \delta_{l_0+1} = P_{l_0+1}^{l_0} x_{l_0}^{t_0}$ *with* $\delta_{l_0+1} \neq 0$.

*Then for any $n \geq 1$ such that the vectors are well-defined,*

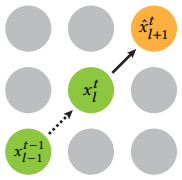
$$x_{l_0+n}^{t_0+n} = P_{l_0+n}^{l_0}((n+1)x_{l_0}^{t_0} - nx_{l_0}^{t_0-1}) + (2^n - n - 1)P_{l_0+n}^{l_0+1}\delta_{l_0+1}.$$

Theorem 2 shows that any violation of the condition $x_{l+1}^t = P_{l+1}^l x_l^t$ for any $t$ and $l$ leads to an asymptotically exponential growth of error, unless the prolongation operators do not have full rank. In practice, the prolongation operators used for Progressive Simulation [Zhang et al. 2023, 2022, 2024] have full rank.

## 5.4 Sufficient Conditions for Ensuring Stability

Building on Theorems 1 and 2, we see that stable time integration in this setting of Progressive Dynamics requires either (i) the cancellation of $\delta_l$ when constructing the time integration stencil, or (ii) ensuring that the coefficient in front of the source term, $\delta_l$, is less than or equal to 1, thus avoiding exponential growth. These conditions open the door to a family of integrators that meet these requirements while adding consistency by leveraging lagged position and velocity information from previous timesteps and levels. We briefly previewed two examples of such methods in Section 4. Together with the integrator proposed in Zhang et al. [2024], all three are specific instances of the general Progressive Dynamics++ model for constructing Progressive Dynamics integrators proposed in Equation 5. Recall that, based on their stencil constructions, we call them the **semi-diagonal** [Zhang et al. 2024], **full-diagonal** and ***VelPro*** integration methods.

*5.4.1 Full-Diagonal Integration Method.* Building on the above analysis of instabilities, we aim to develop improved Progressive Dynamics time integration methods to address stability. Since the discrepancy $\delta_l$ is inherent in progressive refinement and grows exponentially, propagating diagonally as $\delta_{l+n} \approx 2^n P_{l+n}^l \delta_l$, a straightforward approach is to eliminate it by adding a correction term during time integration. Following this idea, we recall that $\delta_l$ originates from $x_l^{t-1} + \delta_l = P_l^{l-1} x_{l-1}^{t-1}$. Thus, eliminating $\delta_l$ requires modifying the multi-scale timestepping stencil. Accordingly, instead of Zhang et al.'s [2024] original update of $\hat{x}_{l+1}^t = P_{l+1}^l(2x_l^t - x_l^{t-1})$, we first consider a new Progressive Dynamics integration in the framework using the update, for $l \in \{1, 2, \ldots, L\}$,



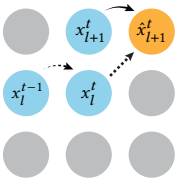

$$\begin{aligned}\hat{x}_{l+1}^t &= P_{l+1}^l\left(x_l^t + h\frac{(x_l^t - x_l^{t-1} - \delta_l)}{h}\right)\\ &= P_{l+1}^l\left(x_l^t + h\frac{(x_l^t - P_l^{l-1}x_{l-1}^{t-1})}{h}\right)\\ &= P_{l+1}^l(2x_l^t - P_l^{l-1}x_{l-1}^{t-1}), \quad \forall t,\end{aligned} \tag{7}$$

and

$$\hat{x}_1^t = P_1^0(2x_0^t - x_0^{t-1}), \quad \forall t. \tag{8}$$

In Appendix B, we provide detailed stability analysis for this construction. More precisely, we show that, modulo non-linearity, a perturbation $\delta_0$ applied to a single boundary value results in a perturbation on $x_l^t$ of magnitude at most $\sqrt{n_L}(\min\{t, l\}+1)||\delta_0||$, where we recall that $n_L$ is the number of vertices at the finest level $L$. Note that if $n_L$ is fixed, the amplification of the perturbation $\delta_0$ grows at most linearly in $l$, in contrast to Zhang et al. [2024]'s semi-diagonal time integration, where the growth is exponential in $l$.

*5.4.2 VelPro Integration Method.* On the other hand, since the primary source of instability is the expansive coefficient 2 in front of $x_l^t$, we can construct an alternative method that replaces the coefficient 2 with 1. Analysis for error follows similarly and likewise prevents exponential growth. Concretely, we propose to define the prolonged diagonal update at grid points $(t+1, l+1)$ as




$$\begin{aligned}\hat{x}_{l+1}^t &= x_{l+1}^t + h\,P_{l+1}^l v_l^t\\ &= x_{l+1}^t + h\,P_{l+1}^l\frac{(x_l^t - x_l^{t-1})}{h}\\ &= x_{l+1}^t + P_{l+1}^l(x_l^t - x_l^{t-1}).\end{aligned} \tag{9}$$

Detailed stability analysis for this integrator is provided in Appendix C, where we show that modulo non-linearity, a perturbation $\delta_0$ applied to a boundary value results in a change on $x_l^t$ of magnitude at most $\sqrt{n_L}(\min\{t, l\}+1)||\delta_0||$, which is at most linear in $l$ for fixed $n_L$.

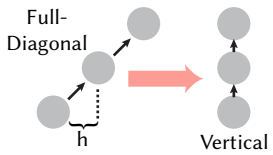
# 6 CONTINUITY

Above we have analyzed the stability properties of our proposed full-diagonal and *VelPro* time integration methods. However, another critical property for animation remains unaddressed: temporal continuity. We define temporal continuity as animations that produce continuous, uninterrupted animations over time, free from *non-physical* jumps or jittering, thereby enabling high-quality transitions between frames.

It is critical to jointly consider both stability and continuity in Progressive Dynamics integration. Our definition of stability for Progressive Dynamics integration as in Section 5 is analogous to standard forward ("horizontal") time integration. Progressive Dynamics integration is stable when a Progressive Dynamics integrator's iterated sequence of timestep solves (advancing in both time and resolution) remain bounded (and so do not diverge). However, as we cover below, and in contrast to horizontal time integration, a stable Progressive Dynamics integration can still exhibit jittering

artifacts due to broken continuity. This is because, as in the fully diagonal method, stable yet independent Progressive Dynamics integration paths can arrive at adjacent timestep frames, at the same level, with disagreement. When sequenced horizontally, the resulting animations then exhibit jitters exactly where the integrations disagree. This again highlights the importance of satisfying both stability and continuity in Progressive Dynamics integration.

## 6.1 Continuity Properties of Progressive Integrators




Here, we examine the continuity properties of the three Progressive Dynamics integrators covered so far. Stability is a predicate for continuity. As the number of levels increases, we have seen that the semi-diagonal integrator can become unstable, generating artifacts ranging from minor jitters to severe explosions and breaking the temporal continuity. In the case of the full-diagonal integrator, discontinuities can still arise, *without instability*, as covered above, from abrupt jumps between frames adjacent in time in-level, but along different diagonal lines of integration. Fundamentally, vertical timestepping—directly prolonging coarse simulation results—as introduced and analyzed by Zhang et al. [2024] can be seen as an extreme form of the full-diagonal Progressive Dynamics integration, where the integration angle is fully vertical due to the vanishing timestep sizes.

In contrast, the *VelPro* integration method significantly mitigates discontinuities. Compared to direct simulation at level $l+1$, the *VelPro* integration method's momentum update is defined as $\hat{x}^t_{l+1} = x^t_{l+1} + v^t_{l+1}h$, with the critical distinction being the way the velocity term is handled. Intuitively, temporal coherence improves when each timestep is consistently integrated from the prior state $x^t_{l+1}$, and the solve is warm-started from the same state. Neither full-diagonal nor semi-diagonal satisfies this requirement. This gives a significant improvement over the semi-diagonal and fully diagonal methods, both of which face the aforementioned challenges in maintaining temporal continuity.

## 6.2 Quantitative Metric for In-Level Temporal Continuity

So far, we have relied on qualitative visual observations to evaluate continuity breaks. To quantitatively analyze and compare the severity of continuity breaks across methods, we propose a new computational measure. This measure provides a reliable tool for assessing physical continuity, and identifying temporal artifacts of arbitrary physical animations.

To do so, we reformulate forward numerical timestepping as a discrete boundary value problem (BVP). This, in turn, enables the evaluation of physical continuity (evaluated as the accuracy of satisfying the timestep update at level $l$) of a proposed position, $y^t_l$, generated by a Progressive Dynamics algorithm at timestep $t$ and level $l$, relative to its time-stencil neighbors at the same level, $y^{t-1}_l$ and $y^{t+1}_l$. For single-level timestepping, the construction is straightforward. The standard (non-progressive) implicit Euler timestepping update equation can be expressed as

$$M(x^{t+1} - 2x^t + x^{t-1}) + h^2\nabla E(x^{t+1}) = 0, \tag{10}$$

which corresponds to the optimality conditions of Equation 1.

We then define a midpoint state estimator, $\phi^t$, that takes neighboring positions $x^{t+1}$ and $x^{t-1}$ as input:

$$x^t = \phi^t(x^{t+1}, x^{t-1}) = \tfrac{1}{2}(x^{t+1} + x^{t-1}) + \tfrac{h^2}{2}M^{-1}\nabla E(x^{t+1}),$$

and now treat $x^t$ as evaluated via "known" positions $x^{t-1}$ and $x^{t+1}$, rather than the usual forward integration of solving for $x^{t+1}$, based on states at times $t-1$ and $t+1$. Importantly, $x^t$ is "fully explicit," and so does not require linear nor nonlinear solves. Conceptually, $x^t$ is given by the average of its adjacent time-stencil positions, corrected by the energy gradient, which accounts for half the integrated force contribution. In the context of Progressive Dynamics, we can apply this same definition for levels $l < L$, resulting in the following evaluation of expected position at time $t$ and and level $l$, given its time-stencil neighbors on either side,

$$x^t_l = \phi^t_l(x^{t+1}_l, x^{t-1}_l) = \tfrac{1}{2}(x^{t+1}_l + x^{t-1}_l) + \tfrac{h^2}{2}M^{-1}_l\nabla F_l(x^{t+1}_l),$$

with $F_l$ being the per-level proxy energy at level $l$, as defined in Equation 2.

To measure our system's continuity, we then can construct continuity error measures for each "proposed" position $y^t_l$ in the resolution-time grid relative to its horizontal (time) neighbors, $y^{t-1}_l$ and $y^{t+1}_l$, as

$$\begin{aligned} e^t_l &= \|y^t_l - \phi^t_l(y^{t+1}_l, y^{t-1}_l)\|^2_{M_l} \\ &= \|\tfrac{1}{2}(y^{t+1}_l - 2y^t_l + y^{t-1}_l) + \tfrac{h^2}{2}M^{-1}_l\nabla F_l(y^{t+1}_l)\|^2_{M_l}. \end{aligned} \tag{11}$$

This provides a per-timestep error with physical units of distance (meters in our plots), integrated over the entire surface domain using the mass matrix $M_l$ (with units of $[\mathrm{kg}\cdot\mathrm{m}^2]$) to ensure proper scaling across different mesh resolutions.

Although the measure $e^t_l$ is well defined, in practice, during Progressive Dynamics simulation, each timestep is solved to the same tolerance, $\epsilon$, with respect to the norm of the Newton decrement divided by the timestep size $h$ [Li et al. 2021]. However, the gradient's norm then differs for each timestep, which means that Equation 10 is, of course, never perfectly satisfied numerically and instead varies in residual across steps. Even for single-level direct timestepping, the error $e^t_l$ computed from Equation 11 will not vanish. Given that each timestep solution inherently achieves a different tolerance in this measure, we quantify the error of the original progressive timestep solve, denoted by $\hat{e}^t_l$, and use it to normalize $e^t_l$. We define $\hat{e}^t_l$ as:

$$\hat{e}^t_l = \|(x^{t+1}_l - \hat{x}^t_l) + h^2M^{-1}_l\nabla F_l(x^{t+1}_l)\|^2_{M_l},$$

where $\hat{x}^t_l$ is the term defining the specific Progressive Dynamics integrator used (see Section 5.4 above). For practical application, we then apply our continuity error normalized as:

$$n^t_l = \frac{e^t_l}{\hat{e}^t_l}. \tag{12}$$

Here, $n^t_l$ is a dimensionless quantity that provides a standardized measure of temporal continuity. We refer to Section 8.2.2 for detailed analysis with concrete examples.

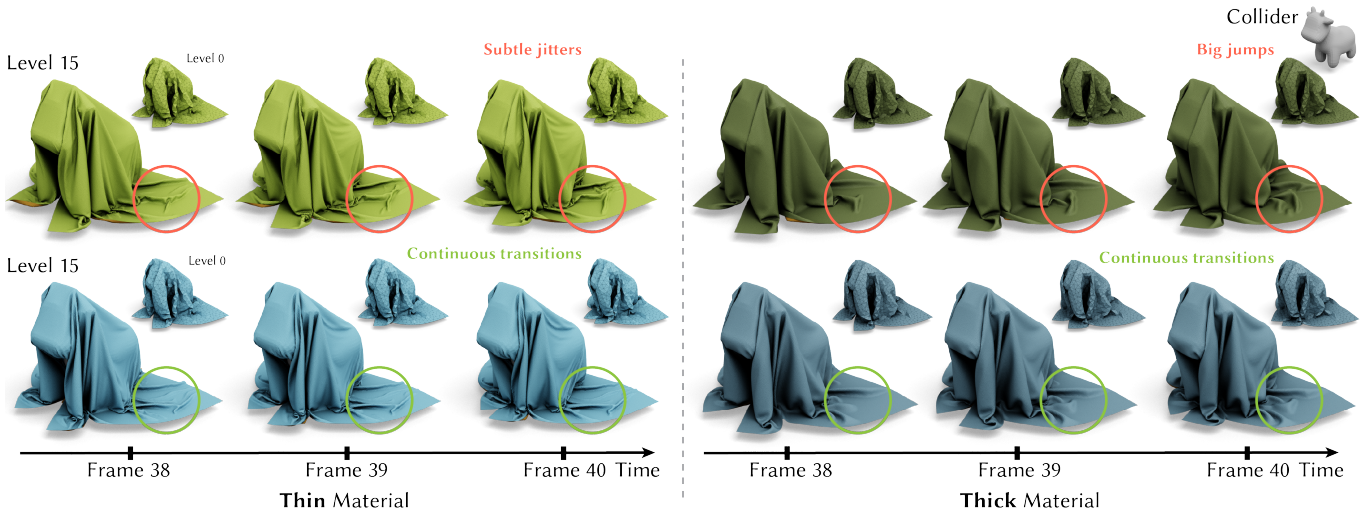



Fig. 10. **Thin and Thick Materials:** We simulate cloth with a 16-level hierarchy and $h = 0.01$s, dropped onto a rigid spot collider with a frictionless ground. For the thin material (0.07 mm thick, strain limit 6.8%, left panel), the full-diagonal integrator exhibits fine-level jittering due to continuity breakage. For the thicker material (0.4 mm thick, right panel), it produces larger discontinuous jumps between folds. In contrast, the *VelPro* integrator yields stable, artifact-free results with continuous animation across all levels, which captures detailed wrinkles and realistic sliding behavior in both material cases.

## 7 CONSISTENCY AND ENRICHMENT

The ability to achieve consistent refinement across levels is a fundamental feature and goal of Progressive Dynamics. Consistency is primarily determined by a careful construction of the $\hat{x}^t_{l+1}$ term, via temporally consistent extrapolation of positions and velocities generated by the current and prior level solutions. Ideally, as described in Section 4, this term should well-approximate predicted mesh positions at $x^t_{l+1} + v^t_{l+1}h$ at the finer level $l+1$, by lagging behind the explicit extrapolation of position and velocity from the prior levels. Toward this goal, as covered above, the Progressive Dynamics++ framework is divided into two parts: approximating position and approximating velocity. In Zhang et al. [2024]'s earlier semi-diagonal method, both position and velocity are extended from level $l$, introducing a factor of 2 for $x^t_l$, but ensuring that $x^t_{l+1}$ remains consistent with $x^t_l$. However, by prolonging only the velocity, $v^t_l$, from the prior level, as in the construction of the *VelPro* integrator, the coefficient for $x^t_l$ is reduced from 2 to 1, preserving positional consistency while maximizing the enrichment of velocity information. This forms the basis of the *VelPro* integration scheme. A similar analysis applies to the full-diagonal integrator, where a modified stencil is used for velocity updates.

Meanwhile, balancing enrichment and consistency involves an inherent trade-off. For instance, as analyzed in Zhang et al. [2024] purely vertical prolongation can generate perfectly consistent upsampled shapes, but lacks enrichment, as no additional physical details can emerge at finer levels of resolution. In such cases, using more levels and finer meshes becomes unnecessary. Conversely, direct simulations provide increasingly enriched results with finer levels, but lack any guarantee of geometric consistency. Thus, achieving a balance between enrichment, by adding fine-scale details, and consistency, by ensuring consistent bulk approximations across levels, poses a fundamental trade-off. To evaluate the balance between consistency and enrichment, across Progressive Dynamics integration methods, we propose the following geometric consistency metric:

$$d^t_{l-1} = ||\Pi^l_{l-1}(x^t_l) - x^t_{l-1}||^2_{M_{l-1}}, \tag{13}$$

where $\Pi^l_{l-1}(\cdot)$ is the projection operator mapping any intermediate level ($l > 0$) geometry $x^t_l$ to the next coarser level $l-1$. We define $\Pi^l_{l-1}(\cdot) = \left((U^{l-1}_l)^T(U^{l-1}_l)\right)^{-1}(U^{l-1}_l)^T$, where $U^{l-1}_l$ denotes the linear intrinsic part of the prolongation operator $P^{l-1}_l(\cdot)$. We refer to Section 8.2.3 for detailed analysis and application of this measure.

## 8 EVALUATION

We implement all semi-diagonal, full-diagonal, and *VelPro* integration methods, as well as direct C-IPC simulation [Li et al. 2021], in C++ for evaluation in a unified test bed. We apply Apple's Accelerate solver for linear solves and Eigen for remaining linear algebra routines [Guennebaud et al. 2010]. We report our example statistics and perform benchmarks on an Apple MacBook Pro with a M4 Max chip and 128 GB of RAM.

### 8.1 Benchmark Examples

*Bouncy Trampolines.* We begin with a simple setup involving three nearly rigid stiff shell objects (with Young's modulus $Y = 6 \times 10^{10}$ Pa)—a spot, a wedge, and a cube—dropped onto an extremely thin bouncy trampoline (thickness = 0.01mm) under gravity. In the first set of experiments, we use a 4-level hierarchy (as advocated for by Zhang et al. [2024]) with $h = 0.01$s as a sanity check for stability and continuity using a hierarchy with a small number of levels. As shown in Figure 7, when the wedge, with its sharp corner, collides with the trampoline, it creates sharp wrinkling at the finest

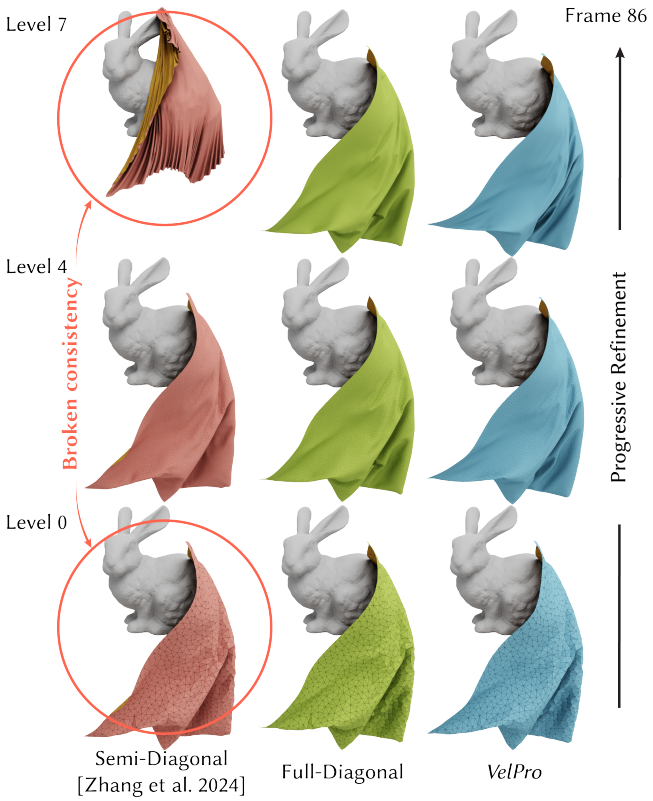



Fig. 11. **Consistency Comparison:** As more levels are introduced (e.g., here an eight-level hierarchy), the semi-diagonal method suffers not only from instabilities and discontinuities, but also violates the geometric consistency goal of the Progressive Dynamics framework. In contrast, the full-diagonal method remains per-frame consistent across levels, despite lacking continuity, while our *VelPro* method achieves both continuity and consistency.

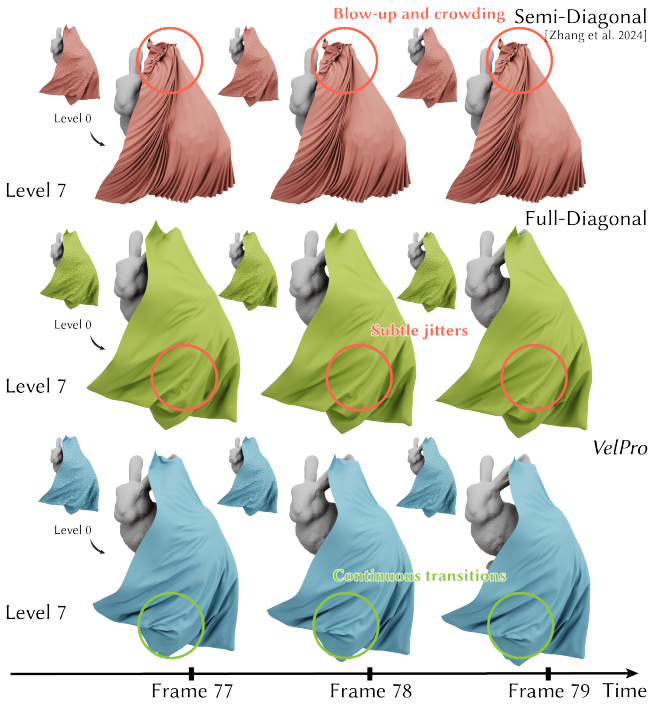



Fig. 12. **Continuity Comparison:** In the same setup as Figure 11, both semi-diagonal and full-diagonal methods violate the continuity goal of the Progressive Dynamics framework from different causes. The semi-diagonal method breaks continuity due to generated instabilities. On the other hand, the full-diagonal method, although stable, breaks continuity because its independent Progressive Dynamics integration paths can arrive at adjacent timestep frames, at the same level, with disagreement. In contrast, the *VelPro* method yields stable, artifact-free results with continuous transitions across frames.

level. In this setting, all three PD integrators (semi-diagonal, full-diagonal and *VelPro* integrators) produce artifact-free results. In a comparable experiment, we now instead construct an 8-level hierarchy, keeping the coarsest and finest mesh resolutions unchanged, as shown in Figure 8. In this setting, the stability issues analyzed above with the semi-diagonal integrator are amplified and now clearly evident, with significant jumps at certain frames (e.g., frame 40) in the animation. For the full-diagonal integrator, subtle jitters emerge in the finest-level results due to the full-diagonal method's above-discussed breaks in temporal continuity. In contrast, our proposed *VelPro* integrator achieves continuous simulations with seamless transitions across levels in both examples.

*Cloth Drop Over Bunny.* We next consider the behavior of a thin cloth (0.07 mm thickness) simulation with an enforced 6.8% strain limit, draped over a rigid bunny collider, under gravity with friction ($\mu = 0.3$). For progressive simulations, we use an 8-level hierarchy with $h = 0.01$s, with mesh resolutions ranging up from 1.5K vertices (coarsest level) to 90K vertices (finest level). In this example, with lower friction, the cloth initially collides with the bunny and then slides off. Applying the semi-diagonal integrator, we see expected instabilities, leading to unexpected deformations and significant animation jumps. In contrast, the full-diagonal integrator produces stable results without explosions or other instability artifacts, but subtle jitters again persist throughout the animation due to continuity breaks. In comparison, the *VelPro* integrator consistently generates animations that are stable and free of visual artifacts, demonstrating its robustness in this higher-level example. We visually compare the three methods in Figure 11 for this example. Here we see frames with the semi-diagonal method's instabilities, and the *VelPro* method's improved enrichment over the full-diagonal method. In Figure 12 we correspondingly see the semi-diagonal and full-diagonal method's continuity breaks in comparison to the physically continuous animation generated by the *VelPro* integrator.

*Cloth Drop Over Spot.* We extend the above experiment to a setting with more wrinkling details. Here, the same cloth geometry is dropped onto a rigid spot collider under gravity, with an underlying ground plane without friction. The ground plane introduces additional wrinkling details when the cloth folds and forms more pronounced wrinkles while sliding inward. To further investigate stability, we applied a 16-level hierarchy with $h = 0.01$s for this

example (reusing same coarsest and finest mesh resolutions as the above bunny drop example). In the first set of experiments, the same thin material (0.07 mm thick with a strain limit of 6.8%) is used. Here, the full-diagonal integration method exhibits significant jittering from continuity breakage, particularly at finer mesh levels, as shown in the left panel of Figure 10. We then also simulate the same drop with a thicker material (0.4 mm thick), as shown in the right panel of Figure 10. In this case, the folds are larger and less detailed, and the discontinuities now manifest themselves as larger jumps between the folds rather than the previous small jitters in the thinner-material example. Across both material setups, the *VelPro* integrator remains stable, producing smooth and continuous animations without visible artifacts, and with intricate folds and detailed sliding dynamics.

*Waving Flag.* To evaluate long-term consistency under persistent and highly dynamic motion, we set up a flag example similar to that of Zhang et al. [2024] over an extended time span. For comparison, we closely replicate the same cloth flag setup as Zhang et al. [2024], using a thin material (0.4 mm thick) with a strain limit of 10%. However, instead of employing a 4-level hierarchy with $h = 0.04$s as in Zhang et al. [2024], we retain the same coarsest (550 vertices) and finest (35K vertices) meshes but instead apply a 10-level hierarchy, with intermediate levels of progressively increasing resolutions. When an IPC simulation is run directly [Li et al. 2021] on the example, the dynamic behavior of the waving flag (as expected) diverges rapidly across resolutions, resulting in inconsistent deformations after only a short time span (see the inset in Section 1). At coarser levels, membrane locking in direct simulations also generates significant simulation artifacts, making direct simulation unsuitable for use as a progressive previewer.

Next, given the large number of levels used here, we can apply this example as a stress test to evaluate the stability properties of the semi-diagonal method of Zhang et al. [2024]. In Figure 5, frames 166 to 168 from the corresponding semi-diagonal simulation are displayed in a grid from the coarsest level (level 0) to the finest level (level 9). The first six coarser levels remain mostly stable, but instability becomes evident and progressively worsens starting from level 7. At the finest level, Zhang et al. [2024]'s semi-diagonal method demonstrates severe instability, with the flag cloth experiencing drastic deformation and forming irregular shapes. In contrast, with the same setup, our newly proposed full-diagonal and *VelPro* integrators remain stable, delivering animations free of visual artifacts throughout the 20-second simulation; see Figure 6. Also see our supplemental videos for animations.

*Smashing Balloons.* To evaluate a challenging scenario with high-speed impact, extreme deformation, and large contacting areas, we throw two happy face character balloons at each other in Figure 17. Here, balloons approach from opposite directions, colliding head-on, resulting in large deformations as they entangle and subsequently detangle from each other. This highly dynamic interaction poses a significant challenge for simulation stability and detail preservation. To handle this complexity, we construct a 4-level hierarchy, with the finest level consisting of 440K vertices and 0.9 million triangles, using $h = 0.01$s. To ensure that the coarsest level has sufficient degrees of freedom for effective preview simulations, we used a coarsest mesh with 50K vertices. The Progressive Dynamics framework, equipped with our newly proposed *VelPro* integration method, is applied to simulate this example. As shown in Figure 17, each level of the hierarchy produces results with progressively enhanced detail, capturing increasingly intricate wrinkles and deformations. The animations' transitions, such as times where the balloons entangle and later detangle, consistently occur at the same moment across all levels, providing animations across levels with geometric consistency, enrichment, and temporal continuity. At the finest level, the animation captures highly detailed and natural wrinkles, with artifact-free results. The final animation is production-ready, offering animators a high-quality output that realistically captures the dramatic interaction between the two balloons.

## 8.2 Comparisons and Analysis

*8.2.1 Stability.* Our benchmark examples collectively demonstrate that the results of our experiments (with nonlinear contributions) align with our theoretical findings (based on a linear analysis) on the exponential growth of the level difference, $\delta^l$, as the total number of levels $L$ increases. Recall that our analysis highlights why the semi-diagonal integrator of Zhang et al. [2024] performs reasonably well with small numbers of levels: when $L$ is relatively small, the level difference, $\delta^l$, does not grow too large to be significantly perceptible in many practical scenarios (e.g., Figure 7). This instability behavior of the semi-diagonal method varies with simulation scene setup, including factors of mesh resolutions and material properties. Meanwhile, even as $L$ increases just a little more in applications, the exponential growth of this level difference amplifies errors rapidly. These errors then appear in the final animation results as jitters or, in worse cases, significant explosion artifacts, as demonstrated in Figures 5, 8, 11, and 12. In turn, they can also break continuity and consistency. Overall, such resulting animations are not usable for practical purposes. In contrast, the full-diagonal and *VelPro* integrators, with stability in both resolution and time (as analyzed above), remain stable across all benchmark examples without numerical explosions, as illustrated in Figures 6, 7, 8, 10, 11 and 12.

*8.2.2 Temporal Continuity.* As discussed in Section 6.1, the instability of Zhang et al.'s [2024] semi-diagonal integrator can then correspondingly also break temporal continuity. On the other hand, the full-diagonal integrator, when applied with a small number of levels $L$, has a shorter diagonal integration path (bounded by the number of levels $L$ when $L < N$). This reduces not only enrichment but also trajectory divergence, in the form of breaking temporal continuity, which can generate animation jitters and jumps. Correspondingly, the results for the full-diagonal integrator in Figures 6 and 7 are mostly continuous, with only minor, visually imperceptible jumps. However, with a larger number of levels in the hierarchy, or else different motions and/or materials, continuity issues can become more significant for the full-diagonal integrator, as shown in Figures 8, 10, 11 and 12.


To quantitatively analyze these temporal continuity breaks, we use our proposed metric from Section 6.2. Figures 13 and 14 compare continuity errors for a 4-level hierarchy using direct simulation,

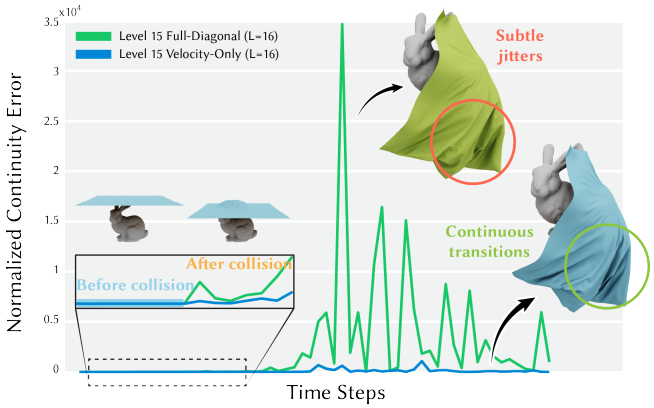



Fig. 13. **Temporal Continuity Error Comparisons:** In the 16-level cloth-on-bunny example, our proposed quantitative continuity metric clearly reveals temporal discontinuities in the full-diagonal method, while the *VelPro* method maintains visually continuous frame transitions with corresponding low error. This metric also precisely captures the exact moment of cloth–bunny contact. See Section 8.2.2 for details.

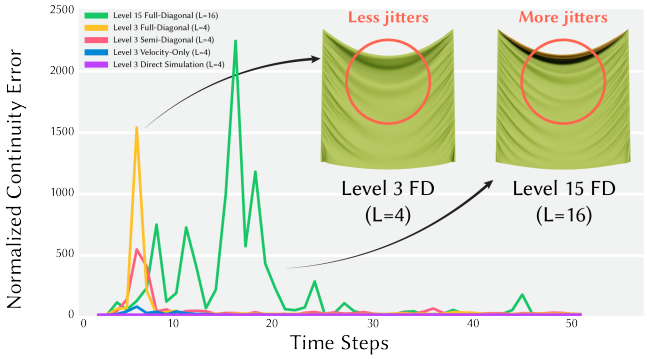



Fig. 14. **Temporal Continuity Error with Increased Levels:** We compare temporal continuity errors across 4-level hierarchies using direct simulation, semi-diagonal, full-diagonal and *VelPro* methods. As expected, direct simulation yields a baseline value of $n_t = 1$ throughout, while the *VelPro* method achieves the lowest continuity error among the three progressive methods. To highlight the impact of increasing hierarchy depth, we also show that the full-diagonal method exhibits a substantial rise in error when scaling from 4 to 16 levels. See Section 8.2.2 for details.

and the fully-diagonal, semi-diagonal and *VelPro* integrators. We simulate a thin cloth (0.07mm thick) pinned at two sides and draped under gravity. For comparison, we also evaluate the same setup with a 16-level hierarchy (using the same finest and coarsest meshes). Evaluating $n_l^t$ from (12), we solve all nonlinear Newton iterations to a tight tolerance (norm $< 10^{-4}$) for convergence. By construction, direct simulation yields a baseline of $n_l^t = 1$. Since the 4-level hierarchy is below the semi-diagonal method's instability threshold in this specific case, its continuity error remains reasonable. However, the full-diagonal integrator already exhibits a significant continuity issue, both visually and quantitatively, particularly in the 16-level hierarchy, due to trajectory divergences during its longer integration path, which matches our expectations. In contrast, the *VelPro* integrator shows only a slight deviation from $n_l^t = 1$, demonstrating significant continuity improvement within the Progressive Dynamics++ framework.

Finally, we also evaluate the temporal continuity metric on the cloth-dropped-on-bunny example (Figure 11) using a 16-level hierarchy for both *VelPro* and full-diagonal integrators. The results clearly show that the full-diagonal integrator is far less continuous, and hence generates more jittering, in animation, at the finest level. In particular, the temporal continuity metric accurately captures the effects of physical transitions: before collisions and deformations, $n_l^t$ remains at 1; after collisions, when jitters occur, the continuity error increases accordingly.

*8.2.3 Geometric Consistency and Enrichment.* As analyzed in Section 7, the semi-diagonal, full-diagonal, and *VelPro* integration methods are designed to produce consistent results with progressively increasing detail. However, due to the instability issues inherent to the semi-diagonal integration method, geometric consistency can breakdown, leading to artifacts shown in Figure 11. Both the full-diagonal and *VelPro* methods achieve qualitatively consistent results frame by frame. However, the interaction between consistency and enrichment often complicates the evaluation of their performance, as discussed in Section 7.

Qualitatively, we observe that, given the same simulation setup and number of levels, the *VelPro* integration method consistently outperforms the semi-diagonal and full-diagonal ones by producing results with richer detail, such as finer wrinkles. Moreover, we observe that the *VelPro* integrator significantly mitigates the stability and continuity issues observed in the other methods, as shown in Figure 7. To quantify this observation, we use the geometric consistency metric defined in (13), which effectively captures subtle differences in enrichment between different methods.

We observe that while the *VelPro* integration method generally achieves stable, continuous, and consistent results, one limitation remains: not surprisingly, per-frame geometric consistency can break for some extremely challenging scenarios. For example, maintaining consistency in a large colliding pile-up of thin shell objects is one such particularly difficult case. When we wish to further enhance the consistency in such cases, we propose adding a small quadratic consistency biasing term to our integrators. This allows users to balance between consistency and enrichment as needed. Our such modified potential energy per level is then

$$W_l(x) = F_l(x) + w||x - P_l^{l-1}x_{l-1}^{t+1}||_{M_l}^2,$$

with $w \geq 0$ the consistency penalty weight. The above quadratic, mass-weighted penalty then targets $P_l^{l-1}x_{l-1}^{t+1}$, and so encourages consistency between solutions of current levels $l$ and next-coarser levels $l-1$.

We evaluate the effectiveness of this consistency penalty in a challenging jumble pile example, using a 3-level hierarchy with $h = 0.04$s. Here, we consider increasing penalty weights with $w = 0, 0.1, 0.25, 0.5$. In this example, a set of very stiff (close-to-rigid), polyhedral shell shapes are dropped onto a bouncy trampoline with three of its corners pinned. This stress test for the general Progressive Dynamics framework highlights the challenge of maintaining

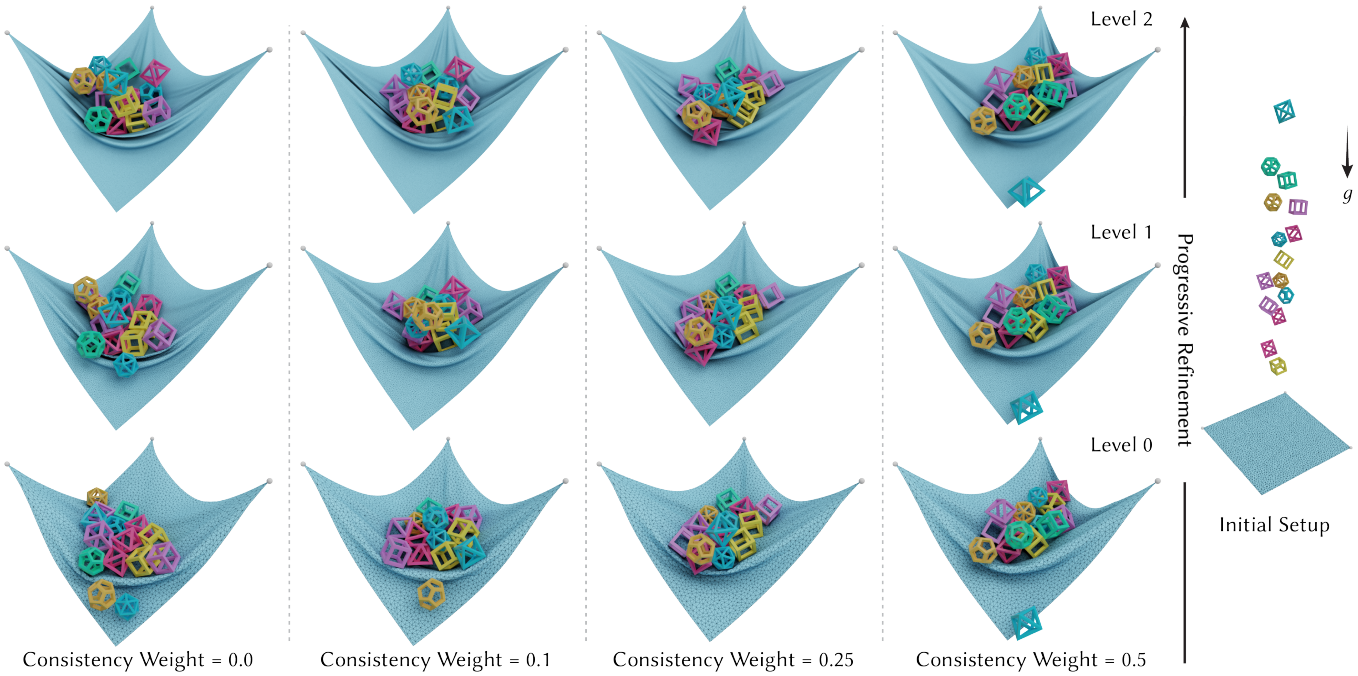



Fig. 15. **Increasing Consistency Weights in a Jumble Pile Example:** We test consistency penalty weights ($w = 0, 0.1, 0.25, 0.5$) on a 3-level hierarchy with a set of stiff polyhedral shells dropped onto a bouncy trampoline (three corners pinned, $h = 0.04$s). This challenging setup stresses the Progressive Dynamics framework's ability to maintain per-frame consistency across resolutions. Without a penalty ($w = 0$), the simulation exhibits visible inconsistency across levels. Adding even a small consistency penalty term ($w = 0.1$) significantly improves consistency, while larger weights further enhance cross-level consistency at the cost of reduced enrichment.

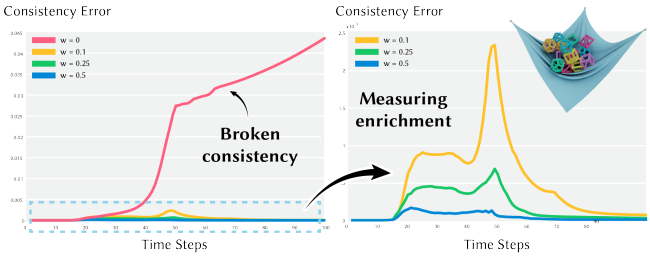



Fig. 16. **Measuring Consistency and Enrichment:** Using the same setup as Figure 15, the left plot shows that our geometric consistency metric (Equation 13) effectively captures the break in consistency, in this example, in the absence of a penalty term. As the penalty weight increases, consistency improves, while the right plot shows a corresponding decrease in fine-level wrinkle enrichment, which is also successfully captured by our metric.

per-frame consistency across resolutions. Without any consistency penalty (that is, using the default PD potential energy $F_l(x)$), we see the animation results break consistency in some frames over time. However, adding even the smallest quadratic consistency penalty term (e.g., $w = 0.1$) significantly improves consistency. As $w$ increases to 0.5, consistency improves further, as shown in Figure 15. To quantify consistency, we apply the metric defined in (13) by projecting the finest-level geometry onto the coarsest level. The left panel of Figure 16 illustrates the issue of broken consistency through consistency error plots when the penalty term is absent. Conversely, experiments with increasing penalty weights show that the metric also reflects the level of enrichment: higher penalty weights constrain the system more, leading to less wrinkle enrichment at the finest level.

*8.2.4 Timing and Scalability.* Our Progressive Dynamics++ framework modifies PD integrators only for progressive advancement at finer levels, and so leaves the underlying, coarsest-level preview stepping unchanged across methods. As a result, the computation timing for the coarsest-level preview remains identical to that of the original Progressive Dynamics as in Zhang et al. [2024], which is approximately twice as fast as a direct coarse simulation. For progressive advancement to finer levels, despite the change in the time integration stencil, we observe no significant difference in speed or convergence rate for the full-diagonal and *VelPro* integration methods compared to the semi-diagonal integration method, except when stability issues of the semi-diagonal integration method dominate. Under such conditions, the semi-diagonal integration method struggles to converge due to the drastically deformed and irregular, non-smooth shapes in the $\hat{x}_l^t$ terms. Specifically, using the *VelPro* integration method, we observe 30× and 55× speedups in the preview simulation compared to the direct finest-level simulation for the hat-toss (see Section 8.3 below) and smashing balloon examples, respectively.

At the same time, we emphasize that analysis of such speedups (and scalability) for Progressive Dynamics is different from that in standard simulation pipelines. At the finest level, the performance of all three Progressive Dynamics methods is comparable to that of the underlying finest-level direct simulator—neither better nor worse. Speedup, and so scalability, is then considered here in terms of the Progressive Dynamics++ framework's ability to enable rapid design iterations on coarse-level meshes. This allows animators to iteratively explore designs with quick coarse-level solves (for large speedups over otherwise necessary fine-level design iterations) first, and then only pay the high-resolution simulation costs once, to refine a finalized design.

### 8.3 Animation Design with Progressive Dynamics++

*Five-Hats-Toss.* In the five-hats-toss design and editing process (see Figures 1 and 2), the Progressive Dynamics++ framework enables users to efficiently perform preview simulations at coarse levels, streamlining exploration and editing over variations in animation designs. In this example, we demonstrate its use in an animation design task using a four-level hierarchy, and a timestep of $h = 0.01$s. Here the animator's goal is to throw five floppy hats from a distance, and ensure that all hats land on a hook, with detailed fabric deformations at the finest level. By simply specifying initial conditions and a constant initial velocity for each hat, the hats follow ballistic trajectories before colliding with the hooks. However, once the hats make contact, the combined effects of gravity, friction, and contact interactions between the five hats introduce significant complexities. Tuning material parameters and initial conditions to achieve perfect "toss success", where all five hats land on the hooks with pleasing deformations, is a tedious and time-consuming trial-and-error process. With the new *VelPro* integrator enabled by our proposed Progressive Dynamics++ framework, users can apply coarsest-level preview animations to rapidly iterate through a wide range of material parameters and initial conditions to capture the desired and challenging-to-capture behavior. They can efficiently identify parameter sets that achieve their successful toss under a variety of conditions. Once a desired animation is finalized, Progressive Dynamics progressively synthesizes production-quality animations with millions of fine-scale vertices, capturing intricate deformation details while maintaining consistency in achieving the five-hat-toss outcome. Importantly, Progressive Dynamics++ provides integrators with stable, continuous, and artifact-free previews at coarse levels. In contrast, the original semi-diagonal method of Zhang et al. [2024] struggles with jittering and stability issues, making it unsuitable for handling delicate examples like this.

## 9 CONCLUSION

We have proposed Progressive Dynamics++, a general framework for constructing a family of Progressive Dynamics integrators. To do so, we have generalized Progressive Dynamics integrators, defined targeted properties for them, and constructed new quantitative measures for evaluating these properties across Progressive Dynamics integrators, both new and prior. To ensure stable Progressive Dynamic animation, we have carefully evaluated and defined the necessary conditions for stable Progressive Dynamics integration across time *and* resolution. Considering these conditions, we have then applied the framework to design and evaluate a new *VelPro* integration method that provides stability across all LODs and timesteps, and produces continuous animations without jumps or jittering artifacts. Across a wide range of benchmarks and challenging animation design tasks, we have demonstrated the application of *VelPro* for high-quality progressive animation design and look forward to future extensions and improvements in Progressive Simulation via the Progressive Dynamics++ framework.

Here important work remains to extend Progressive Dynamics' application, beyond animation and VFX, to accurate physical simulation tasks. Exploring alternatives such as material fitting and homogenization may help address cross-resolution material behavior. Additionally, we see that variations in the stencils of Progressive Dynamics integrators, within the Progressive Dynamics++ framework, enable different degrees of parallelism. Since we do not yet observe a strict trade-off, an interesting direction for future work is whether new Progressive Dynamics integrators can jointly achieve improved consistency, stability, and parallelism.

## ACKNOWLEDGMENTS

Jiayi Eris Zhang is supported by the Stanford Graduate Fellowship, the Roblox Graduate Fellowship, and the NVIDIA Graduate Fellowship. Doug James acknowledges support from Adobe, SideFX, NVIDIA, and the Department of Energy, National Nuclear Security Administration under Award Number DE-NA0003968; Houdini software courtesy of SideFX. We thank Zhenyuan Zhang for insightful discussions and for proofreading all the appendix proofs.

## 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.

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.

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).

Lan Chen, Juntao Ye, and Xiaopeng Zhang. 2021b. Multi-Feature Super-Resolution Network for Cloth Wrinkle Synthesis. *Journal of Computer Science and Technology* 36 (2021), 478–493. https://doi.org/10.1007/s11390-021-1331-y

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.

Zhenyu Chen, Rahul Narain, Eitan Grinspun, and Danny M. Kaufman. 2023. Complex Shell Simulation with Contact. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–16.

S. Friedhoff, R. D. Falgout, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder. 2013. A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel. In *Sixteenth Copper Mountain Conference on Multigrid Methods*. https://parallel-in-time.org/methods/mgrit.html

Martin J. Gander and Thibaut Lunet. 2024. *Time Parallel Time Integration*. SIAM - Society for Industrial and Applied Mathematics. https://www.amazon.com/Time-Parallel-Integration-Martin-Gander/dp/1611978017

Michael Garland and Paul S Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 209–216.

Eitan Grinspun, Anil N Hirani, Mathieu Desbrun, and Peter Schröder. 2003. Discrete shells. In *Symposium on Computer Animation*.

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

Oshri Halimi, Egor Larionov, Zohar Barzelay, Philipp Herholz, and Tuur Stuyck. 2023. PhysGraph: Physics-Based Integration Using Graph Neural Networks. *arXiv preprint arXiv:2301.11841* (2023). https://arxiv.org/abs/2301.11841

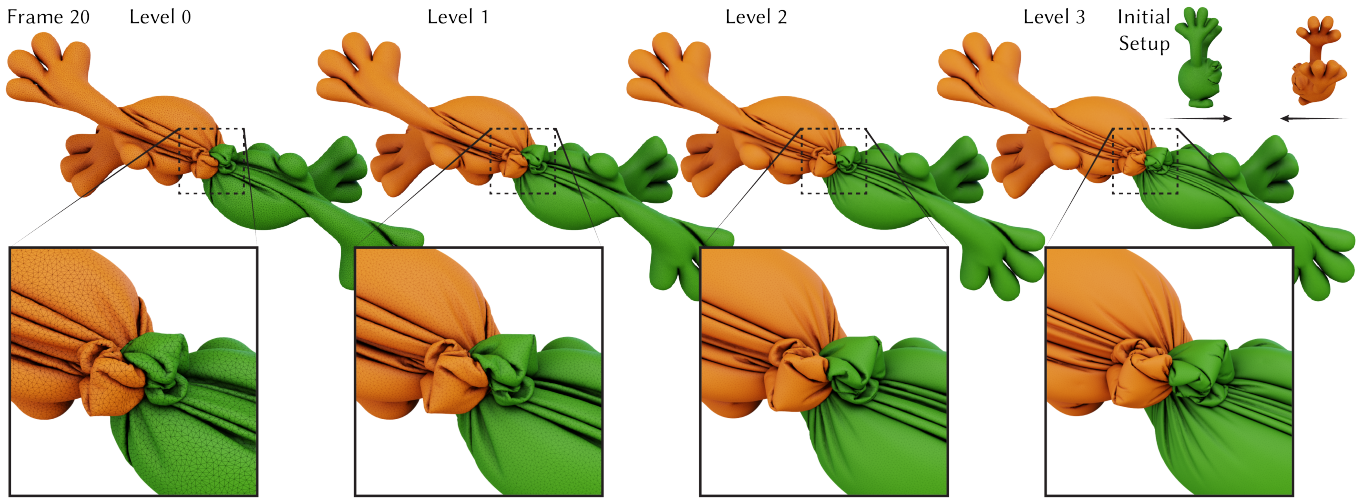



Fig. 17. **Happy Face Balloons with Detailed Enrichment:** In this high-speed collision scenario, two character balloons entangle and detangle under extreme deformation. Using a 4-level hierarchy, the *VelPro* integration method in the Progressive Dynamics++ framework captures increasingly intricate wrinkles across levels while preserving consistent deformation, timing, and trajectories.

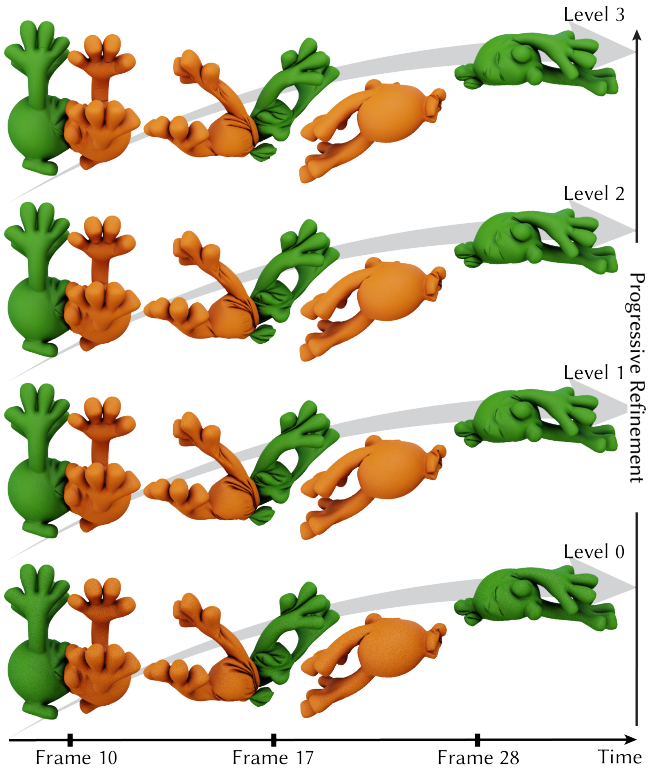



Fig. 18. **Happy Face Balloons with Strong Consistency:** From this alternate viewpoint of the same animation shown in Figure 17, we see the animation's frames are consistent across resolutions and time, with matching deformation and timing throughout.

Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K Katsaggelos. 2016. Video super-resolution with convolutional neural networks. *IEEE transactions on computational imaging* 2, 2 (2016), 109–122.

Ladislav Kavan, Dan Gerszewski, Adam W Bargteil, and Peter-Pike Sloan. 2011. Physics-inspired upsampling for cloth simulation in games. In *ACM SIGGRAPH 2011 papers*. 1–10.

Tae-Yong Kim, Vladimir Vendrovsky, and Nancy S. Pollard. 2013. Data-Driven Dynamic Deformation Component Separation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–9.

Zorah Lahner, Stefanie Wuhrer, and Hans-Peter Seidel. 2018. DeepWrinkles: Accurate and Realistic Clothing Modeling. *Computer Graphics Forum* 37, 2 (2018), 361–373.

Tae Min Lee, Young Jin Oh, and In-Kwon Lee. 2019. Efficient Cloth Simulation using Miniature Cloth and Upscaling Deep Neural Networks. *arXiv preprint arXiv:1907.03953* (2019). https://arxiv.org/abs/1907.03953

Cheng Li, Min Tang, Ruofeng Tong, Ming Cai, Jieyi Zhao, and Dinesh Manocha. 2020. 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, Danny M. Kaufman, and Chenfanfu Jiang. 2021. Codimensional Incremental Potential Contact. *ACM Trans. Graph.* 40, 4, Article 170 (jul 2021), 24 pages.

Jacques-Louis Lions, Yvon Maday, and Gabriel Turinici. 2001. A "parareal" in time discretization of PDE's. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics* 332, 7 (2001), 661–668. https://doi.org/10.1016/S0764-4442(00)01793-6

Matthias Müller and Nuttapong Chentanez. 2010. Wrinkle Meshes. In *Symposium on Computer Animation*. Madrid, Spain, 85–91.

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 *Proceedings of Computer Graphics International (CGI)*. 139–146. https://doi.org/10.1145/3208159.3208175

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

Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. 2018. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 8934–8943. https://doi.org/10.1109/CVPR.2018.00931

Rasmus Tamstorf. 2013. Derivation of discrete bending forces and their gradients. *Technical Report* (2013).

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.

Zachary Teed and Jia Deng. 2020. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 402–419. https://doi.org/10.1007/978-3-030-58555-6_24

Philip Trettner and Leif Kobbelt. 2020. Fast and Robust QEF Minimization using Probabilistic Quadrics. *Computer Graphics Forum* (2020). https://doi.org/10.1111/cgf.13933

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

Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. 2019. EDVR: Video Restoration with Enhanced Deformable Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 1954–1963. https://doi.org/10.1109/CVPRW.2019.00247

Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. 2019. Video Enhancement with Task-Oriented Flow. In *International Journal of Computer Vision (IJCV)*, Vol. 127. 1106–1125. https://doi.org/10.1007/s11263-018-1136-7

Jiawang Yu and Zhendong Wang. 2024. Super-Resolution Cloth Animation with Spatial and Temporal Coherence. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 105:1–105:14. https://doi.org/10.1145/3658143

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

Jiayi Eris Zhang, Doug James, and Danny M Kaufman. 2024. Progressive Dynamics for Cloth and Shell Animation. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–18.

Meng Zhang and Jun Li. 2024. Neural Garment Dynamic Super-Resolution. *arXiv preprint arXiv:2412.06285* (2024). https://arxiv.org/abs/2412.06285

## A PROOF OF EXPONENTIAL GROWTH OF $\delta$

Before proving Theorems 1 and 2, we first recall the setting and clarify the intuition behind Theorem 1.

- The condition
$$x_l^t = P_l^{l-1}(2x_{l-1}^{t-1} - x_{l-1}^{t-2}) \quad \forall t, l \tag{14}$$
along with the boundary values $\{x_0^t\}_{t\in\{0,1,\dots,N\}}$, $\{x_l^0\}_{l\in\{0,1,\dots,L\}}$, and $\{v_l^0\}_{l\in\{0,1,\dots,L\}}$ describes the diagonal stepping scheme, the precomputed solutions at the coarsest level and the initial states of all the levels. With this information, we can populate the entire space-time grid for any given timestep $t$ and level $l$.
- The formula
$$x_l^t = P_l^{l-n}((n+1)x_{l-n}^{t-n} - nx_{l-n}^{t-n-1}) \quad \forall t, l, n \tag{15}$$
suggests that multi-level diagonal timestepping essentially mirrors single-level horizontal timestepping. The main difference lies in the application of the prolongation operator, which adjusts for level differences.
- On the other hand, the condition
$$x_{l+1}^t = P_{l+1}^l x_l^t \quad \forall t, l \tag{16}$$
suggests that the solutions at different levels are **perfectly consistent** at every timestep.

Proof of Theorem 1. We first show "⇒". Let $t, l, n$ be arbitrary and we assume that (15) always holds. By (14), we have

$$\begin{aligned} x_{l+1}^{t+1} = P_{l+1}^l(2x_l^t - x_l^{t-1}) &= P_{l+1}^l(2(P_l^{l-1}(2x_{l-1}^{t-1} - x_{l-1}^{t-2})) - x_l^{t-1}) \\ &= 4P_{l+1}^{l-1}x_{l-1}^{t-1} - 2P_{l+1}^{l-1}x_{l-1}^{t-2} - P_{l+1}^l x_l^{t-1}. \end{aligned} \tag{17}$$

In addition, (15) applied with $n = 2$ yields

$$x_{l+1}^{t+1} = P_{l+1}^{l-1}(3x_{l-1}^{t-1} - 2x_{l-1}^{t-2}). \tag{18}$$

Combining (17) and (18), we get

$$P_{l+1}^l(P_l^{l-1}x_{l-1}^{t-1} - x_l^{t-1}) = 0.$$

Since $P_{l+1}^l$ has full rank, we must have $x_{l+1}^t = P_{l+1}^l x_l^t \quad \forall t, l$.

Next we show "⇐". Assume (16). By (14), we have that (15) holds with $n = 1$. Assume (15) holds for $n-1$ for all $t, l$. Then by (16), for all $t, l$,

$$\begin{aligned} x_l^t &= P_l^{l-n+1}(nx_{l-n+1}^{t-n+1} - (n-1)x_{l-n+1}^{t-n}) \\ &= P_l^{l-n+1}(nP_{l-n+1}^{l-n}(2x_{l-n}^{t-n} - x_{l-n}^{t-n-1}) - (n-1)P_{l-n+1}^{l-n}x_{l-n}^{t-n}) \\ &= P_l^{l-n}((n+1)x_{l-n}^{t-n} - nx_{l-n}^{t-n-1}). \end{aligned}$$

Therefore, (15) holds by induction. □

Proof of Theorem 2. Assume there exists $(t_0, l_0)$ such that $x_{l_0+1}^{t_0} + \delta_{l_0+1} = P_{l_0+1}^{l_0}x_{l_0}^{t_0}$ where $\delta_{l_0+1} \neq 0$, and that for $t \in \{t_0 - 1, t_0\}$ and $l \geq 0$, $x_{l+1}^t = P_{l+1}^l x_l^t$, except when $(t, l) = (t_0, l_0)$. It remains to prove by induction that for any $n \geq 1$,

$$x_{l_0+n}^{t_0+n} = (n+1)P_{l_0+n}^{l_0}x_{l_0}^{t_0} - nP_{l_0+n}^{l_0}x_{l_0}^{t_0-1} + (2^n - n - 1)P_{l_0+n}^{l_0+1}\delta_{l_0+1}, \tag{19}$$

which indicates exponential growth of the error. The base case $n = 1$ follows from (14). Suppose that (19) holds for $n-1$. Then by (14),

$$x_{l_0+n}^{t_0+n} = 2P_{l_0+n}^{l_0+n-1}x_{l_0+n-1}^{t_0+n-1} - P_{l_0+n}^{l_0+n-1}x_{l_0+n-1}^{t_0+n-2}. \tag{20}$$

The first term on the right-hand side of (20) can be computed further using the induction hypothesis:

$$\begin{aligned} x_{l_0+n-1}^{t_0+n-1} = nP_{l_0+n-1}^{l_0}x_{l_0}^{t_0} - (n-1)P_{l_0+n-1}^{l_0}x_{l_0}^{t_0-1} \\ + (2^{n-1} - n)P_{l_0+n-1}^{l_0+1}\delta_{l_0+1}. \end{aligned} \tag{21}$$

On the other hand, the second term on the right-hand side of (20) can be computed using Theorem 1. Since $x_{l+1}^t = P_{l+1}^l x_l^t$ holds for all $l_0 < l \leq L$ and $t \in \{t_0 - 1, t_0\}$, Theorem 1 yields that

$$\begin{aligned} x_{l_0+n-1}^{t_0+n-2} &= (n-1)P_{l_0+n-1}^{l_0+1}x_{l_0+1}^{t_0} - (n-2)P_{l_0+n-1}^{l_0+1}x_{l_0+1}^{t_0-1} \\ &= (n-1)P_{l_0+n-1}^{l_0+1}(P_{l_0+1}^{l_0}x_{l_0}^{t_0} - \delta_{l_0+1}) - (n-2)P_{l_0+n-1}^{l_0}x_{l_0}^{t_0-1}, \end{aligned} \tag{22}$$

where in the last step we used our assumption $x_{l_0+1}^{t_0} + \delta_{l_0+1} = P_{l_0+1}^{l_0}x_{l_0}^{t_0}$. Combining (20), (21), and (22), and rearranging the terms leads to

$$\begin{aligned} x_{l_0+n}^{t_0+n} &= (n+1)P_{l_0+n}^{l_0}x_{l_0}^{t_0} - nP_{l_0+n}^{l_0}x_{l_0}^{t_0-1} \\ &\quad + (2(2^{n-1} - n) + (n-1))P_{l_0+n}^{l_0+1}\delta_{l_0+1} \\ &= (n+1)P_{l_0+n}^{l_0}x_{l_0}^{t_0} - nP_{l_0+n}^{l_0}x_{l_0}^{t_0-1} + (2^n - n - 1)P_{l_0+n}^{l_0+1}\delta_{l_0+1}, \end{aligned}$$

which aligns with (19), as desired. □

*Discussion.* Single-level horizontal timestepping also uses a similar time integration form, where (a simplified version of) the velocity update is defined as $x_l^{t+1} = 2x_l^t - x_l^{t-1}, \ \forall t, l$. In the following, we explain the fundamental differences between horizontal timestepping (single-level) and our diagonal stepping (multi-level) methods, particularly by highlighting that single-level horizontal timestepping exhibits only linear error growth.

Proposition 3. *For single-level horizontal timestepping, the addition of a perturbation $\delta_l$ to $x_l^t$ leads to a linear error growth over $n$ timesteps, rather than exponential.*

Proof. It is clear by induction that

$$x_l^t = (n+1)x_l^{t-n} - nx_l^{t-n-1}, \quad \forall\, n \in \{0, 1, \ldots, t-1\}.$$

Therefore, by adding a perturbation $\delta_l$ to $x_l^{t-n}$, after $n$ timesteps, the induced error is $(n+1)\delta_l$ on $x_l^t$, resulting in linear error growth. □

Proposition 3 indicates that a single-level horizontal implicit Euler timestepper experiences only linear error growth (modulo nonlinearity). In contrast, Theorem 1 describes the multilevel diagonal stepper as performing similarly to the horizontal stepper, except for the difference introduced by prolongation. However, this equivalence holds only under conditions of perfect consistency, such as in the simplified model where nonlinear solves are trivial. In practical scenarios with *approximate* nonlinear solves, discrepancies between solutions at different levels inevitably arise at each timestep, quantified as $||x_{l+1}^t - P_{l+1}^l x_l^t||_{M_{l+1}} > 0$. Theorem 2 further suggests that such discrepancies, when present, can propagate diagonally and grow exponentially.

## B STABILITY ANALYSIS FOR FULL-DIAGONAL TIME INTEGRATION SCHEME

In this appendix, we show how the full-diagonal time integration scheme addresses the stability issue by reintroducing the effects of nonlinear solves into our analysis. Recall from (7) and (8) that in this formulation, for $l \in \{1, 2, \ldots, L\}$,

$$\hat{x}_{l+1}^t = P_{l+1}^l(2x_l^t - P_l^{l-1}x_{l-1}^{t-1}),$$

and

$$\hat{x}_1^t = P_1^0(2x_0^t - x_0^{t-1}).$$

Considering that an implicit Euler timestep is resolved through minimization,

$$x_l^{t+1} = \operatorname*{argmin}_x \frac{1}{2h^2}||x - \hat{x}_l^t||_{M_l}^2 + E_l(x), \tag{23}$$

we may write $x_l^{t+1} = \hat{x}_l^t + \gamma_l^{t+1}$, where $\gamma_l^{t+1}$ captures the nonlinear component of the minimization problem. Furthermore, as the timestep $h$ decreases, $x_l^{t+1}$ approaches $\hat{x}_l^t$ more closely.

In the full-diagonal time integration regime, the recursive relations are thus given by

$$x_{l+1}^{t+1} = P_{l+1}^l(2x_l^t - P_l^{l-1}x_{l-1}^{t-1}) + \gamma_{l+1}^{t+1}, \quad \forall t,\ \forall l \in \{1, 2, \ldots, L\} \tag{24}$$

and

$$x_1^{t+1} = P_1^0(2x_0^t - x_0^{t-1}) + \gamma_1^{t+1}, \quad \forall t. \tag{25}$$

Along with the boundary values $\{x_0^t\}_{t\in\{0,1,\ldots,N\}}$, $\{x_l^0\}_{l\in\{0,1,\ldots,L\}}$, and $\{v_l^0\}_{l\in\{0,1,\ldots,L\}}$, these relations uniquely determine the output array $\{x_l^t\}_{t\in\{0,1,\ldots,N\},\, l\in\{0,1,\ldots,L\}}$.

For our stability analysis, we estimate the fluctuation of the output $x_l^t$ given a perturbation of the boundary values, where for simplicity we assume that the perturbation does not affect the values $\{\gamma_l^t\}_{t\in\{0,1,\ldots,N\},\, l\in\{0,1,\ldots,L\}}$. Suppose that for some $t_0 \in \{0, \ldots, N\}$, a single boundary value $x_0^{t_0}$ is replaced by $\bar{x}_0^{t_0} = x_0^{t_0} + \delta_0$, and denote by $\{\bar{x}_l^t\}_{t\in\{0,1,\ldots,N\},\, l\in\{0,1,\ldots,L\}}$ the output determined by the recursive relations above with the updated boundary values.

We define

$$y_l^t = x_l^t - P_l^{l-1}x_{l-1}^{t-1} \quad \text{and} \quad \bar{y}_l^t = \bar{x}_l^t - P_l^{l-1}\bar{x}_{l-1}^{t-1}. \tag{26}$$

Also write their fluctuations as

$$\Delta x_l^t = \bar{x}_l^t - x_l^t \quad \text{and} \quad \Delta y_l^t = \bar{y}_l^t - y_l^t. \tag{27}$$

The relation (24) then implies that

$$\Delta y_{l+1}^{t+1} = P_{l+1}^l \Delta y_l^t,$$

and hence induction yields that for any $t > l$,

$$\Delta y_{l+1}^{t+1} = P_{l+1}^1 \Delta y_1^{t-l}.$$

Applying (25) and (26) gives

$$\Delta y_1^{t-l} = P_1^0(\Delta x_0^{t-l-1} - \Delta x_0^{t-l-2}).$$

Therefore, for each $t, l$ with $t > l$, there exists $\xi_{t,l}$ such that $||\xi_{t,l}|| \le ||\delta_0||$ and

$$\Delta y_l^t = P_l^0 \xi_{t,l}. \tag{28}$$

Next, we show by induction on $l$ that there exists $\zeta_{t,l}$ such that $||\zeta_{t,l}|| \le (l+1)||\delta_0||$ and

$$\Delta x_l^t = P_l^0 \zeta_{t,l}. \tag{29}$$

Indeed, the case $l = 0$ follows by our assumption on the perturbation. Suppose that (29) holds for $l-1$ and $||\zeta_{t-1,l-1}|| \le l||\delta_0||$. Note that (26) and (27) together yields

$$\Delta x_l^t = \Delta y_l^t + P_l^{l-1}\Delta x_{l-1}^{t-1}.$$

By (28) and the induction hypothesis, we have

$$\Delta x_l^t = P_l^0\xi_{t,l} + P_l^{l-1}P_{l-1}^0\zeta_{t-1,l-1} = P_l^0(\xi_{t,l} + \zeta_{t-1,l-1}),$$

where by triangle inequality,

$$||\xi_{t,l} + \zeta_{t-1,l-1}|| \le ||\xi_{t,l}|| + ||\zeta_{t-1,l-1}|| \le (l+1)||\delta_0||.$$

This proves (29) by setting $\zeta_{t,l} = \xi_{t,l} + \zeta_{t-1,l-1}$.

Since $P_l^0$ has nonnegative entries with row sums bounded by one, we obtain $||P_l^0|| \le \sqrt{n_l} \le \sqrt{n_L}$. We conclude from (29) that

$$||\Delta x_l^t|| \le ||P_l^0||\ ||\zeta_{t,l}|| \le \sqrt{n_L}(l+1)||\delta_0||,$$

which is the desired stability statement. The other cases of $t \le l$ and perturbing the values in $\{x_l^0\}_{l\in\{0,1,\ldots,L\}}$ can be similarly established.

In other words, the nonlinear components of each timestep's solution contribute to only linear growth over time, ensuring stability. On the other hand, the previous semi-diagonal time integration scheme results in unstable exponential growth, which can be shown analogously.

## C STABILITY ANALYSIS FOR VELOCITY-ONLY PROLONGATION TIME INTEGRATION

First, recall from (9) the construction of the update term using the velocity-only prolongation time integration scheme, defined as

$$\hat{x}_{l+1}^t = x_{l+1}^t + P_{l+1}^l(x_l^t - x_l^{t-1}). \tag{30}$$

Meanwhile, considering the form of the implicit Euler timestep,

$$x_l^{t+1} = \operatorname*{argmin}_x \frac{1}{2h^2}||x - \hat{x}_l^t||_{M_l}^2 + E_l(x), \tag{31}$$

we may write $x_l^{t+1} = \hat{x}_l^t + \gamma_l^{t+1}$, where $\gamma_l^{t+1}$ captures the nonlinear component of the minimization problem. It follows that

$$x_{l+1}^{t+1} = x_{l+1}^t + P_{l+1}^l(x_l^t - x_l^{t-1}) + \gamma_{l+1}^{t+1}, \quad \forall t, l. \tag{32}$$

Here, the boundary values are given by $\{x_0^t\}_{t\in\{0,1,\ldots,N\}}$, $\{x_l^0\}_{l\in\{0,1,\ldots,L\}}$, and $\{v_l^0\}_{l\in\{0,1,\ldots,L\}}$. Using

$$v_l^0 = \frac{x_l^0 - x_l^{-1}}{h}, \quad \forall l \tag{33}$$

and the relation (32), the boundary values uniquely determine the array of values $\{x_l^t\}_{t\in\{0,1,\ldots,N\},\, l\in\{0,1,\ldots,L\}}$.

In the following, we analyze the stability of the output $x_l^t$ with respect to the boundary values, while assuming for simplicity that the perturbation does not affect the values $\gamma_l^t$. Suppose that for some $l_0 \in \{0, 1, \ldots, L\}$, a single boundary value $x_{l_0}^0$ is replaced by $\bar{x}_{l_0}^0 = x_{l_0}^0 + \delta_0$, and denote by $\{\bar{x}_l^t\}_{t\in\{0,1,\ldots,N\},\, l\in\{0,1,\ldots,L\}}$ the output determined by the recursive relations above. In other words, we have

$$\bar{x}_{l+1}^{t+1} = \bar{x}_{l+1}^t + P_{l+1}^l(\bar{x}_l^t - \bar{x}_l^{t-1}) + \gamma_{l+1}^{t+1}, \quad \forall t, l. \tag{34}$$

Let

$$y_l^t = x_l^t - x_l^{t-1} \quad \text{and} \quad \bar{y}_l^t = \bar{x}_l^t - \bar{x}_l^{t-1}. \tag{35}$$

Combining (32), (34), and (35) yields that

$$\bar{y}_{l+1}^{t+1} - y_{l+1}^{t+1} = P_{l+1}^l(\bar{y}_l^t - y_l^t),$$

and hence by induction, for $t \geq l$,

$$\bar{y}_{l+1}^{t+1} - y_{l+1}^{t+1} = P_{l+1}^0(\bar{y}_0^{t-l} - y_0^{t-l}),$$

and for $t < l$,

$$\bar{y}_{l+1}^{t+1} - y_{l+1}^{t+1} = P_{t+1}^0(\bar{y}_{l-t}^0 - y_{l-t}^0).$$

In the case $t \geq l$, since the boundary values $\{x_0^t\}_{t\in\{0,1,\ldots,N\}}$ and $\{v_l^0\}_{l\in\{0,1,\ldots,L\}}$ are not perturbed, we have $\bar{y}_0^{t-l} = y_0^{t-l}$ and hence $\bar{y}_{l+1}^{t+1} = y_{l+1}^{t+1}$. Suppose now that $t < l$. We have

$$||\bar{y}_{l+1}^{t+1} - y_{l+1}^{t+1}|| \leq ||P_{t+1}^0||\, ||\bar{y}_{l-t}^0 - y_{l-t}^0||.$$

Since each prolongation operator $P_{t+1}^0$ has nonnegative entries with row sums bounded by one, it holds $||P_{t+1}^0|| \leq \sqrt{n_{t+1}} \leq \sqrt{n_L}$. It follows from our assumption on the perturbation $\bar{x}_{l_0}^0 = x_{l_0}^0 + \delta_0$ that $||\bar{y}_{l+1}^{t+1} - y_{l+1}^{t+1}|| \leq \sqrt{n_L}||\delta_0||$ uniformly in $t, l$. By (35) and the triangle inequality,

$$||x_l^t - \bar{x}_l^t|| \leq (t+1)\sqrt{n_L}||\delta_0||, \quad \forall t, l,$$

showing the desired stability.

The stability with respect to other boundary values $\{x_l^0\}_{l\in\{0,1,\ldots,L\}}$ and $\{v_l^0\}_{l\in\{0,1,\ldots,L\}}$ can be derived analogously.