Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Primal Residual Reduction With Extended Position Based Dynamics and Hyperelasticity

Yizhou Chen[a,*], Yushan Han[a], Jingyu Chen[b], Shiqian Ma[c], Ronald Fedkiw[d], Joseph Teran[e]

[a] *University of California, Los Angeles; Epic Games, Inc*
[b] *University of California, Los Angeles*
[c] *Rice University*
[d] *Stanford University; Epic Games, Inc*
[e] *University of California, Davis; Epic Games, Inc*

ARTICLE INFO

Article history:
Received February 27, 2024

Keywords: Position-based dynamics, physics simulation, constrained dynamics

ABSTRACT

The Extended Position Based Dynamics (XPBD) approach of Macklin et al. [29] addresses issues with iteration-dependent behavior in the original Position Based Dynamics [35] (PBD). PBD itself is a powerful method for the real-time simulation of elastic objects, however, it is limited in its application to hyperelastic solids. It can only treat models with a strain energy density that is quadratic in some notion of constraint. Furthermore, we show that even when applicable the formulation does not always lead to convergent behaviors with hyperelasticity. We isolate the root cause to be the approximate linearization of the nonlinear backward Euler systems utilized by XPBD. We provide two fixes to these terms that allow for convergent behavior. The first (B-PXPBD) is a small modification to an existing XPBD code, but can only be used with models addressable by the original XPBD. The second (FP-PXPBD) is a more general formulation that extends XPBD (and our residual correction) to arbitrary hyperelasticity. We show that our modifications allow for convergent behavior that rivals accurate techniques like Newton's method when the computational budget is large without sacrificing the stable and robust behavior exhibited by the original PBD and XPBD when the computational budget is limited.

1. Introduction

In the present work, we consider large strain hyperelastic solids [6] whose governing equations are discretized in space with the finite element method (FEM) [44] and in time with implicit backward Euler [1] or quasistatics [53, 6, 63, 43, 47]. Hyperelastic solid models define continuum stresses from a notion of elastic potential. In graphics applications, these models are commonly used for simulation-based enhancement of character flesh and musculature animation [31, 55, 46, 10, 32, 52]. Wang et al. [55] provide a thorough discussion of the state-of-art. In these applications, the constitutive control enabled by continuum models is essential for realism, e.g., anisotropic contraction along muscle fibers and volume preservation in soft tissues.

Various methods have been proposed for solving the FEM-discretized equations of motions for these materials [24, 37, 7, 53, 13, 29, 19, 63, 62]. Thorough summaries of the state of the art are given by Zhu et al. [63] and Li et al. [24]. The preferred approach in a given application generally depends on the relative importance of constitutive accuracy, robustness/stability and computational efficiency. There is no one method that is optimal in all computer graphics as different applications place different relative importance on these considerations. These equations are nonlinear, and an iterative solver must be used to

[★] Only capitalize first word and proper nouns in the title.
[*] Corresponding author:
e-mail: chenyizhou@ucla.edu (Yizhou Chen)

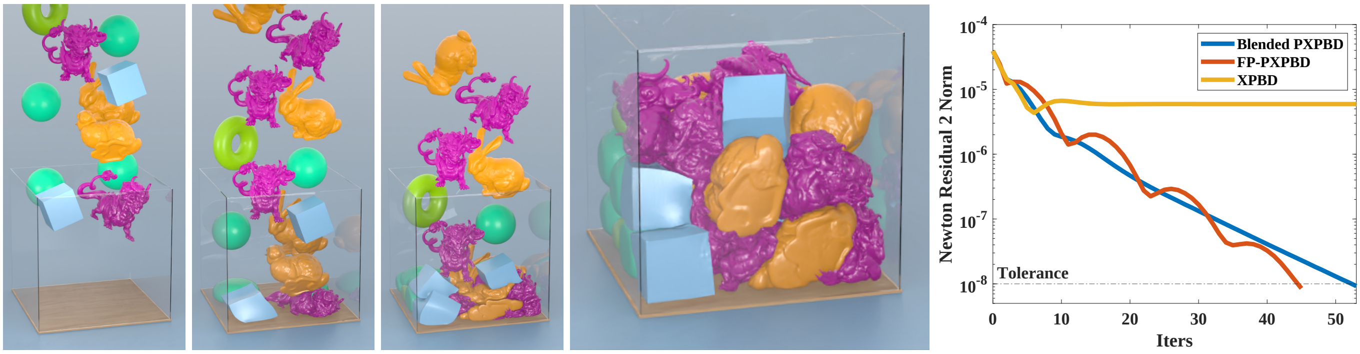


Fig. 1: **30 Objects Dropping (left)**. Our Blended PXPBD (B-PXPBD) approach robustly handles large elastic deformations. **FEM Residual Comparison (right)**. B-PXPBD and FP-PXPBD reduce the backward Euler residual while XPBD stagnate in a representative step of a hyperelasticity simulation.

improve the accuracy of an initial guess by reducing the magnitude of the system residual. While Newton's method [39] generally requires the fewest iterations to reach a desired tolerance (often achieving quadratic convergence), each iteration can be costly and a line search is typically required for stability [13]. However, it is not always necessary to reduce the residual beyond a few orders of magnitude for satisfactory visual accuracy (see discussion in Liu et al. [26], Bouaziz et al. [7], Zhu et al. [63]). In these cases, Newton's method is often outperformed by alternative techniques. Methods like the Alternating Direction Method of Multipliers (ADMM) [8, 37], the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm (L-BFGS) [4, 63, 27, 58] and Sobolev preconditioned gradient descent (SGD) [38, 7, 26, 47] do not require computation of the exact energy Hessian and many of these simplifications leverage direct solvers based on pre-computed matrix factorizations of simplified discrete elliptic operators to allow for reduced per-iteration cost compared to Newton's method.

Real-time applications have very modest computational budgets and this restricts which techniques can be used. The ability to reduce the nonlinear system residuals degrades considerably and many methods that behave admirably with large computational budgets can give unstable and/or visually implausible results when computation is limited. Wang [56] provides a thorough discussion of these issues. For example, methods that make use of a linear solver at each iteration (Newton, ADMM, SGD, some L-BFGS) can only do so at either extremely low resolution or with significant constraints on the accuracy of their approximation. In particular, those that leverage a pre-factored discrete elliptic operator are limited since forward and backward substitutions do not parallelize. Furthermore, the memory requirements for storing the factors are often too burdensome. Moreover, simply replacing the direct solver with an iterative solver with minimal iteration counts can lead to visually implausible or even unstable behaviors (see Figure 2 and [56]). At a minimum, real-time applications require methods that will remain stable and be as visually plausible as possible when the compute budget is limited.

The Position Based Dynamics (PBD) approach of Müller et al. [35] is remarkably powerful due to its robust and stable behavior in applications with minimal computational budgets. PBD has gained wide adoption since there are often no other methods that can provide comparably reliable behavior under extreme computation constraints. For elastic materials, PBD uses a constraint view of the material resistance to deformation and is similar to strain limiting [42] and shape matching [36] techniques. In the context of elasticity, this has been shown to be equivalent to a Gauss-Seidel minimization of an elastic potential that is quadratic in the constraints [26, 7, 29]. However, constitutive control over PBD behavior is challenging as effective material stiffnesses etc. vary with iteration count and time step size. The Extended Position Based Dynamics (XPBD) approach of Macklin et al. [29] addresses these issues by reformulating the original PBD approach in terms of a Gauss-Seidel technique for discretizing a total Lagrange multiplier formulation of the backward Euler system for implicit time stepping. This formulation has similarities to PBD, but with the elastic terms handled properly where PBD can be seen as the extreme case of infinite elastic modulus (or hard strain constraints). In this case, the Lagrange multiplier terms can be interpreted as stress-like and associated with enforcing the constraints (e.g. pressure in an incompressible fluid [3]). With this view, XPBD handles these terms correctly as weak constraints (e.g. as in weakly compressible materials [50, 23]) where PBD can be interpreted as a splitting scheme where non-stress based forces are first integrated, followed by a projection step.

Despite its many strengths, XPBD can only discretize hyperelastic models that are quadratic in some notion of strain constraint [29, 28]. This prevents the adoption of many models from the computational mechanics literature, e.g., for many biomechanical soft tissues. Furthermore, while XPBD is based on a Gauss-Seidel procedure for the Lagrange multiplier formulation of the backward Euler equations, it simplifies the system by omitting the Hessian of the constraints and the residual of the primary (position) equations. The omission of the primary equations is perfectly accurate in the first iteration, but as Macklin et al. [29] point out, less so in latter iterations when constraint gradients vary significantly. We observe that this rapid variation occurs for many hyperelastic formulations and that its omission degrades residual reduction. However, the inclusion of this term introduces instabilities into XPBD.

We provide a modification to the XPBD position update that more accurately guarantees that the primary residual is zero and may be omitted. We call our approach Primal Extended

Position Based Dynamics (PXPBD). It can be done in two ways. The first (B-PXPBD) uses fixed-point iteration to zero the primary residual after the Gauss-Seidel update of the Lagrange multiplier. The second (FP-PXPBD) is a reformulation of XPBD that allows for arbitrary hyperelastic models. We observe that the constraint Hessians and primary residual terms are exactly zero and can be omitted with no error if the first Piola-Kirchhoff stress [6] is used as the auxiliary unknown (in place of the Lagrange multipliers in the original XPBD). We advocate for two models because each have relative strengths and weaknesses in their resolution of the primary residual omission in XPBD. B-PXPBD can be done with a simple modification to an existing XPBD code, however it requires the use of a blending parameter (see Section 3.3.1) since accurate fixed-point iteration is too costly. FP-PXPBD is a larger modification to an existing XPBD code and requires element-wise Newton solves, but it exactly resolves the the issues with both Hessian and residual omission in XPBD. Furthermore, FP-PXPBD allows for arbitrary hyperelastic models while B-PXPBD is based on constraint formulations as with XPBD.

We demonstrate our method with collision-intensive scenarios by applying it to the updated-Lagrangian formulation of hyperelasticity where the simulation mesh is embedded in a regular grid at each time step as in [18].We note that we expand on [61] by adding:

- Explicit expressions for the incorporated anisotropic hyperelastic models.
- Implementation and optimization details for B-PXPBD.
- Detailed derivation of FP-PXPBD and its associated affine transforms.
- Extensive illustrations for coloring strategies related to parallelism structures.

2. Previous work

O'Brien et al. [41, 40] first demonstrated the power of FEM simulation of hyperelasticity in graphics applications. Early approaches used mostly St. Venant-Kirchhoff hyperelasticity [41, 40, 2]. However, as noted in [6, 17] these models weaken under compression and are ill-suited for the large deformation problems encountered in computer graphics. As rigid as possible (ARAP) [47] and corotational models behave more robustly with large deformation [34, 9, 49]. Extension of the elastic response to non-bijective deformation mappings was shown to be useful for graphics applications by Irving et al. [17] with the key insight being the sign convention established in the polar singular value decomposition. This was extended to hyperelasticity by Stomakhin et al. [49], Smith et al. [46] and Kim et al. [20]. Martin et al. [30] design hyperelastic potentials from examples of desired material behaviors. Hyperelasticity and continuum modeling are useful for designing anisotropic models that capture the fiber-driven dynamics of muscle and tendon [51, 52, 10, 20, 32]. Xu et al. [59] develop isotropic and anisotropic hyperelastic models in terms of principle stretches. Tournier et al. [33] develop a technique for bridging elasticity and constraint-based approaches that is robust to large stiffness. They use a similar primal/dual setup to XPBD, however unlike XPBD their approach solves the entire system at once, rather than iterating over individual constraints. Wang and Yang [57] use a Chebyshev accelerated gradient descent approach for general hyperelasticity and FEM.

Baraff and Witkin first demonstrated that implicit time stepping with elasticity is essential for efficiency [1]. Many approaches characterize implicit time stepping with hyperelasticity as a minimization of an incremental potential [30, 13, 26, 48, 7, 37]. This is often referred to as variational implicit Euler [48, 30] or optimization implicit Euler [26]. Quasistatic time stepping is an extreme case where inertia terms are ignored and only the strain energy is minimized [53, 47, 25, 43, 21]. Rotational invariance of the hyperelastic potential makes the energy minimization non-convex with potentially non-unique solutions in quasistatic problems [6]. Minimizers are usually found by setting the gradient of the energy to zero and solving the associated nonlinear system of equations with Newton's method. However, the non-convexity yields indefinite energy Hessians that can prevent convergence. Quasi-Newton methods can be used to approximate the Hessian with a symmetric semi-definite counterpart [53, 39, 63, 45, 24].

Many methods avoid the indefiniteness issue with the inclusion of auxiliary (or secondary) variables. Narain et al. [37], Bouaziz et al. [7], Liu et al. [26] are recent examples of this, but similar approaches have been used in graphics since the local/global approach with ARAP by Sorkine et al. [47]. Rabinovich et al. [43] generalize this approach to a wider range of distortion energies. The global solve in these approaches requires the inversion of a constant discrete elliptic operator (component wise-Laplacian) which can be pre-factored for efficiency. Zhu et al. [63] points out that approaches that make use of this operator can be classified as SGD [38]. While this discrete operator does not suffer from indefiniteness issues, various authors note that SGD approaches may converge initially faster than Newton, but will often taper off [63, 26, 7, 56]. Zhu et al. [63] tailor their approach to this observation and use SGD initially and then combine with L-BFGS to incorporate more second-order information. Liu et al. [27] and Witemeyer et al. [58] also use combinations of SGD and L-BFGS. Kovalsky et al. [21] add Nesterov acceleration to SGD. Hecht et al. [16] develop efficient updates for a pre-factored Hessian with corotated materials. Wang [56] discusses the challenges of using direct solution/pre-factoring in the SGD style approaches of Narain et al. [37], Bouaziz et al. [7] and Liu et al. [26] and develops a Chebyshev acceleration technique as an alternative.

3. Methods

3.1. Equations

We consider implicit time stepping methods for integrating the FEM-discretized partial differential equations (PDEs) describing momentum balance with hyperelastic materials

$$\mathbf{M}\frac{\partial^2 \mathbf{x}}{\partial t^2} = -\frac{\partial PE}{\partial \mathbf{x}} + \mathbf{f}^{ext}, \; PE(\mathbf{x}) = \sum_e \Psi(\mathbf{F}^e(\mathbf{x}))V^e. \qquad (1)$$

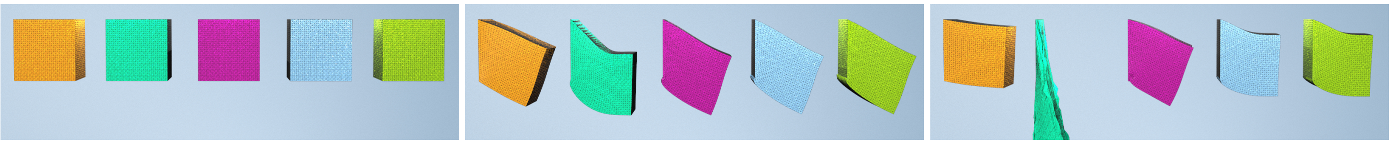


Fig. 2: **Equal Budget Comparison**. From left to right: Newton (converged), Newton, FP-PXPBD, B-PXPBD, XPBD. With a limited budget, XPBD-style methods are stable, whereas the Newton's method suffers from instability. Frame 0, 10, 60 are shown in the figure.

Here $\mathbf{M} \in \mathbb{R}^{3N_p \times 3N_p}$ is a lumped (diagonal) mass matrix, $\mathbf{x} \in \mathbb{R}^{3N_p}$ are the deformed positions of the FEM mesh and the potential energy PE in the system is related to the hyperelastic potential energy density as Ψ. We use linear interpolation over tetrahedron (3D) or triangle (2D) meshes in our FEM formulation. V^e is the volume (3D) or area (2D) of the undeformed e^{th} element arising from the piecewise constant terms in an integrands associated with linear interpolation. $\mathbf{f}^{ext}$ are external forces (gravity etc.). The hyperelastic potential Ψ is a function of the deformation gradient in the e^{th} element ($\mathbf{F}^e$) which is related to deformed positions as

$$\mathbf{F}^e_{\alpha\beta}(\mathbf{x}) = \sum_i x_{i\alpha} \frac{\partial N_i}{\partial \mathbf{X}_\beta}(\mathbf{X}^e) \tag{2}$$

where N_i are the piecewise linear interpolation functions in the FEM formulation and $\mathbf{X}^e$ is the centroid of the undeformed element. We refer the reader to the Bonet and Wood [6] and Barbič and Sifakis [44] for more details.

3.1.1. Hyperelastic Energy Density

The hyperelastic potential defines the constitutive response of the material. We demonstrate our method with the fixed corotated potential from Stomakhin et al. [49]

$$\Psi^{cor}(\mathbf{F}) = \mu|\mathbf{F} - \mathbf{R}(\mathbf{F})|_F^2 + \frac{\lambda}{2}(\det(\mathbf{F}) - 1)^2. \tag{3}$$

Here $\mathbf{R}(\mathbf{F})$ is the closest rotation to $\mathbf{F}$ which we compute from the polar singular value decomposition [12] and μ and λ are the Lamé coefficients. XPBD assumes that the potential is of the form

$$PE(\mathbf{x}) = \sum_c \frac{1}{2} C_c(\mathbf{x}) \frac{1}{a_c} C_c(\mathbf{x}) \tag{4}$$

The corotated potential Ψ^{cor} can be adapted to this from in terms of the following constraints (in element) on the deformation gradient

$$\hat{C}_1(\mathbf{F}) = |\mathbf{F} - \mathbf{R}(\mathbf{F})|_F, \hat{C}_2(\mathbf{F}) = \det(\mathbf{F}) - 1. \tag{5}$$

The gradient of $\hat{C}_1$ is not defined when $\mathbf{F} = \mathbf{R}(\mathbf{F})$ (a common occurrence) and we use the modification $\tilde{C}_1 = \sqrt{\hat{C}_1^2 + \epsilon}$, where ϵ is an arbitrary positive constant to ensure that the gradient is always defined. We use constraints $C_1^e(\mathbf{x}) = \tilde{C}_1(\mathbf{F}^e(\mathbf{x}))$ and $C_2^e(\mathbf{x}) = \hat{C}_2(\mathbf{F}^e(\mathbf{x}))$ with weighting μ and λ respectively (in element e). This is equivalent to using the hyperelastic potential $\Psi^{cor} + \epsilon$ so it produces the same behavior as the corotated model.

We also demonstrate our method with an anisotropic model for muscle contraction (see Figure 4). Here the potential is

$$\Psi^{aniso}(\mathbf{F}) = \Psi^{cor} + \frac{\sigma_{max}}{\lambda_{ofl}}(f_a + \alpha_{act} f_p) \tag{6}$$

where the parameter $\alpha_{act} \in [0, 1]$ controls the degree of active contractile tension and f_a and f_p are based on the anisotropic fiber terms in Blemker et al. [5]. More specifically, f_a and f_p are computed as the following:

$$f_p = \begin{cases} 0.0076\lambda_{ofl} e^{6.6(\frac{l^e}{\lambda_{ofl}} - 1)} - 0.05(l^e - \lambda_{ofl}) & l^e > \lambda_{ofl} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$f_a = \begin{cases} 0 & l^e \leq 0.4\lambda_{ofl} \\ 3\lambda_{ofl}(\frac{l^e}{\lambda_{ofl}} - 0.4)^3 & 0.6\lambda_{ofl} > l^e > 0.4\lambda_{ofl} \\ -0.6612\lambda_{ofl} + l^e - 1.33\lambda_{ofl}(\frac{l^e}{\lambda_{ofl}} - 1)^3 & 1.4\lambda_{ofl} \geq l^e \geq 0.6\lambda_{ofl} \\ 0.6774\lambda_{ofl} + 3\lambda_{ofl}(\frac{l^e}{\lambda_{ofl}} - 1.6)^3 & 1.6\lambda_{ofl} \geq l^e \geq 1.4\lambda_{ofl} \\ 0.6774\lambda_{ofl} & l^e > 1.6\lambda_{ofl} \end{cases} \tag{8}$$

where $l^e = \mathbf{F}^e \mathbf{v}^e$ and $\mathbf{v}^e$ is the fiber direction of the element e.

3.1.2. Implicit Time Stepping

We consider both backward Euler and quasistatic time stepping schemes

$$\mathbf{M}\left(\frac{\frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{\Delta t} - \mathbf{v}^n}{\Delta t}\right) = -\frac{\partial PE}{\partial \mathbf{x}}(\mathbf{x}^{n+1}) + \mathbf{f}^{ext}. \tag{9}$$

Here $\mathbf{x}^n, \mathbf{v}^n$ represent the time $t^n = n\Delta t$ position and velocities. Quasistatic time stepping is the same but with the left hand side of Equation (9) replaced with $\mathbf{0}$. Note that we also may constrain some vertices $\mathbf{x}_i^n, 0 \leq i < N_p$ in practice to enforce boundary conditions and these equations are removed from Equation (9), however we omit the explicit representation of this for concise exposition.

3.2. XPBD

Macklin et al. [29] solve Equation (9) with the introduction of a Lagrange multiplier λ_c associated with each constraint C_c. They assume the potential energy gradient is of the form

$$\Delta t^2 \frac{\partial PE}{\partial x_{i\alpha}} = -\sum_c \frac{\partial C_c}{\partial x_{i\alpha}}(\mathbf{x})\lambda_c \tag{10}$$

where they introduce $\lambda_c = -\frac{\Delta t^2}{a_c} C_c$ as an additional unknown which converts Equation (9) into the system

$$\mathbf{g}(\mathbf{x}^{n+1}, \boldsymbol{\lambda}) = \mathbf{M}\left(\mathbf{x}^{n+1} - \tilde{\mathbf{x}}\right) - \sum_c \lambda_c^T \nabla \mathbf{C}_c(\mathbf{x}^{n+1}) = \mathbf{0} \quad (11)$$

$$\mathbf{h}(\mathbf{x}^{n+1}, \boldsymbol{\lambda}) = \mathbf{C}(\mathbf{x}^{n+1}) + \frac{\mathbf{A}}{\Delta t^2}\boldsymbol{\lambda} = \mathbf{0}. \quad (12)$$

Here $\tilde{\mathbf{x}} = \mathbf{x}^n + \Delta t(\mathbf{v}^n + \mathbf{M}^{-1}\mathbf{f}^{ext})$ are the positions updated under the influence of inertia and external forces, $\boldsymbol{\lambda}$ is the vector of all Lagrange multipliers and $\mathbf{A}$ is a diagonal matrix with entries equal to a_c. The solution is approximated iteratively with $\mathbf{x}_k^{n+1}$ and $\boldsymbol{\lambda}_k$ denoting the k^{th} iterates. $\mathbf{g}(\mathbf{x}_k^{n+1}, \boldsymbol{\lambda}_k)$ is used to denote the residual of the position (primary) unknowns and $\mathbf{h}(\mathbf{x}_k^{n+1}, \boldsymbol{\lambda}_k)$ to denote the residual of the Lagrange multiplier (secondary) unknowns.

XPBD uses a nonlinear Gauss-Seidel procedure based on the linearization

$$\begin{pmatrix} \mathbf{M} + \sum_c \lambda_{ck} \frac{\partial^2 C_c}{\partial \mathbf{x}^2}(\mathbf{x}_k^{n+1}) & -\nabla C_c^T(\mathbf{x}_k^{n+1}) \\ \nabla \mathbf{C}(\mathbf{x}_k^{n+1}) & \frac{\mathbf{A}}{\Delta t^2} \end{pmatrix} \begin{pmatrix} \Delta \mathbf{x}_{k+1} \\ \Delta \boldsymbol{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \mathbf{g}(\mathbf{x}_k^{n+1}, \boldsymbol{\lambda}_k) \\ \mathbf{h}(\mathbf{x}_k^{n+1}, \boldsymbol{\lambda}_k) \end{pmatrix}. \quad (13)$$

In XPBD, the red terms are omitted to enable the update

$$\left(\mathbf{C}^T(\mathbf{x}_k^{n+1})\mathbf{M}^{-1}\mathbf{C}(\mathbf{x}_k^{n+1}) + \frac{\mathbf{A}}{\Delta t^2}\right)\Delta\boldsymbol{\lambda}_{k+1} = -\mathbf{h}(\mathbf{x}_k^{n+1}, \boldsymbol{\lambda}_k) \quad (14)$$

$$\Delta \mathbf{x}_{k+1} = \mathbf{M}^{-1}\nabla \mathbf{C}(\mathbf{x}_k^{n+1})\Delta\boldsymbol{\lambda}_{k+1}. \quad (15)$$

Furthermore, Equation (14) is updated in a Gauss-Seidel fashion where the d^{th} Lagrange multiplier is updated via

$$\Delta\tilde{\lambda}_{k+1d} = \frac{-h_d(\mathbf{x}_k^{n+1}, \lambda_{kd})}{\nabla C_d^T(\mathbf{x}_k^{n+1})\mathbf{M}^{-1}\nabla C_d(\mathbf{x}_k^{n+1}) + \frac{a_d}{\Delta t^2}\lambda_{kd}}, \quad \lambda_{k+1d} = \lambda_{kd} + \Delta\tilde{\lambda}_{k+1d}. \quad (16)$$

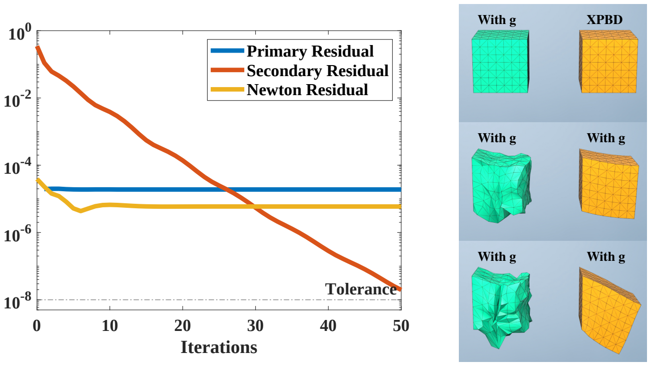
Note that we distinguish $\Delta\tilde{\lambda}_{k+1d}$ in Equation (16) from from $\Delta\lambda_{k+1d}$ in Equation (14) since only one one step of Gauss-Seidel iteration is performed on the linear system. Then the positions associated with the constraint are updated via Equation (15) to create

$$\mathbf{x}_{k+1}^{n+1} = \mathbf{x}_k^{n+1} + \mathbf{M}^{-1}\nabla \mathbf{C}_d(\mathbf{x}_k^{n+1})\Delta\tilde{\lambda}_{k+1d}. \quad (17)$$

The system (Equations (11)-(12)) is then re-linearized (Equation (13)) and the process (Equations (16)-(17)) is repeated iteratively.

3.3. Primary residual XPBD (PXPBD)

The motivation for the omission of the residual and constraint Hessian terms (red) in Equation (13) is natural. The constraint Hessian is non-diagonal and its retention would preclude the decoupling of primary variables from the Lagrange multipliers in Equation (14). Furthermore, the primary residual term $\mathbf{g}(\mathbf{x}_k^{n+1}, \boldsymbol{\lambda}_k)$ requires more floating point operations and generally a gather operation for efficient parallel evaluation. As Macklin et al. [29] point out, the initial guess of $\boldsymbol{\lambda}_0 = \mathbf{0}$ and $\mathbf{x}_0^{n+1} = \tilde{\mathbf{x}}$ means that $\mathbf{g}(\mathbf{x}_0^{n+1}, \boldsymbol{\lambda}_0) = \mathbf{0}$. However, its omission is harder to justify in latter iterates, though Macklin et al. [29] argue that it is justified when the constraint gradients vary slowly and further that its omission makes the approach similar to that of Goldenthal et al. [14]. While omission of secondary information is commonly done in quasi-Newton approaches, we observe that omission of the primary residual terms can lead to stagnation in residual reduction (see Figure 3(a)). Unfortunately, we also notice that inclusion of this term can cause XPBD to lose its favorable stability properties (see Figure 3(b)). We note though that if the global system in Equation (13) is solved with sufficient accuracy (e.g. with a Krylov method and without omission of the red terms), then stability and residual reduction can be achieved, however this is more costly than Newton's method for Equation (9) since the system size is larger with the inclusion of the $\boldsymbol{\lambda}$ unknowns.



(a) (b)

Fig. 3: **(a) Primal Residual Comparison: Stagnation**. While XPBD reliably reduces the secondary residual, its omission of the primary residual in the linearization causes its primary residual to stagnate, making its true (Newton) residual stagnate as well. **(b) Primal Residual Inclusion: Instability**. XPBD is unstable when the primal residual term is not omitted.

3.3.1. Blended Primal XPBD (B-PXPBD)

We believe that the stability of XPBD is due to the omission of this primary residual term $\mathbf{g}(\mathbf{x}_k^{n+1}, \boldsymbol{\lambda}_k)$. We observe that this omission can be done without any error if the position update is chosen to guarantee that the primary residual is zero. This can be done by solving Equation (11) for $\mathbf{x}_{k+1}^{n+1}$ with $\boldsymbol{\lambda}_{k+1}$ fixed after the update (of a single Lagrange multiplier λ_{k+1d}) in Equation (16). We again note that in this context, the Lagrange multipliers $\boldsymbol{\lambda}_{k+1}$ are similar to stresses. Indeed as the a_c are taken to infinity we can see similarities between Equations (11)-(12) and the discretized equations for incompressible fluids and for finite values of a_c the formulation is similar to the compressible formulations in Stomakhin et al. [50] and Kwatra et al. [22]. Therefore, the process of solving Equation (11) for $\mathbf{x}_{k+1}^{n+1}$ with $\boldsymbol{\lambda}_{k+1}$ fixed is akin to solving for the change in positions given a fixed stress state (that does not depend on positions).

Unfortunately, solving Equation (11) for $\mathbf{x}_{k+1}^{n+1}$ is complicated by the dependence of the constraint gradient $\nabla \mathbf{C}(\mathbf{x}_{k+1}^{n+1})$ on positions and solving it accurately would be nearly as difficult as solving the original system in Equation (9). Furthermore, this dependence of the constraint gradient on positions means changing the stress in one constraint propagates to changes in positions in adjacent constraints and therefore throughout the mesh. For example if fixed point iteration were used to solve for $\mathbf{x}_{k+1}^{n+1}$ given $\boldsymbol{\lambda}_{k+1}$ where the only change to $\boldsymbol{\lambda}_k$ was in a single constraint d (as in Equation (16)), then first only the positions of the vertices in the constraint would be changed, but then in the second iteration, any other constraint gradients with dependence on these positions would change, and all positions associated with those constraints would change, and so on. This would quickly become computationally inefficient, however performing one iteration results in an update that only changes the positions involved in the constraint associated with the Lagrange multiplier update in Equation (16)

$$\mathbf{x}_{k+1}^{n+1} = \tilde{\mathbf{x}} + \sum_c \lambda_{kc}\mathbf{M}^{-1}\nabla\mathbf{C}_c(\mathbf{x}_k^{n+1}) + \mathbf{M}^{-1}\nabla\mathbf{C}_d(\mathbf{x}_k^{n+1})\Delta\tilde{\lambda}_{k+1d}. \tag{18}$$

Note that when the residual $\mathbf{g}(\mathbf{x}_k^{n+1}, \boldsymbol{\lambda}_k) = \mathbf{0}$ is zero this update coincides with that of Equation (15). We found that even using this first fixed point iterate was enough to improve residual reduction, however we also found that it reduced the stability compared to Equation (15). We remedy this by taking a linear combination of the updates in Equations (15) and (18)

$$\mathbf{x}_{k+1}^{n+1} = \zeta\left(\mathbf{M}^{-1}\nabla C_d(\mathbf{x}_k^{n+1})\Delta\tilde{\lambda}_{k+1d}\right) + (1-\zeta)\Delta\mathbf{x}_{k+1}^{fp} + \mathbf{x}_k^{n+1} \tag{19}$$

$$\Delta\mathbf{x}_{k+1}^{fp} = \tilde{\mathbf{x}} + \sum_c \lambda_{kc}\mathbf{M}^{-1}\nabla C_c(\mathbf{x}_k^{n+1}) + \Delta\tilde{\lambda}_{k+1d}\mathbf{M}^{-1}\nabla C_d(\mathbf{x}_k^{n+1}) - \mathbf{x}_k^{n+1}. \tag{20}$$

The parameter ζ can usually chosen to be 0.5. We increase it if we observe instability and raise it if we see residual stagnation.

3.3.2. Implementation

Blended P-XPBD can be implemented as a small modification to XPBD. At the m^{th} iteration, we store $\Delta\mathbf{x}_d^{total} = \sum_{k<m}\Delta\mathbf{x}_{kd}$ at the d^{th} constraint where

$$\Delta\mathbf{x}_{kd} = \zeta\left(\mathbf{M}^{-1}\nabla C_d(\mathbf{x}_k^{n+1})\Delta\tilde{\lambda}_{kd}\right) + (1-\zeta)\Delta\mathbf{x}_k^{fp}. \tag{21}$$

Then we can obtain $\Delta\mathbf{x}_k^{fp}$ easily by

$$\Delta\mathbf{x}_k^{fp} = \lambda_{kd}\mathbf{M}^{-1}\nabla C_d(\mathbf{x}_k^{n+1}) - \Delta\mathbf{x}_d^{total}. \tag{22}$$

Pseudo-code similar to that of XPBD in Macklin et al. [29] is provided in Algorithm 1.

ALGORITHM 1: B-PXPBD Simulation Loop

```
Initialize $\Delta\mathbf{x}_d^{total} = \mathbf{0}$.
while not reached maximal iterations do
    for constraint d do
        1. Compute $\Delta\tilde{\lambda}_{kd}$ as in Equation 16. ;
        2. Compute $\Delta\mathbf{x}_k^{fp}$ using Equation 22. ;
        3. Update $\mathbf{x}^{n+1}$ using Equation 20. ;
        4. Update $\lambda_{k+1d}$ using Equation 16. ;
        5. Update $\Delta\mathbf{x}_d^{total} \leftarrow \Delta\mathbf{x}_d^{total} + \Delta\mathbf{x}_{kd}$;
    end
end
```

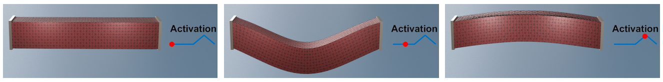


Fig. 4: **Muscle Box Activation**. A rectangular bar with both ends clamped falls under gravity. Two seconds later, the muscle box is activated and contracts along the horizontal direction. The level of activation is shown on the right side of the images. $t = 0.0333, 1.2, 2.9$ seconds are shown in the footage.

ALGORITHM 2: FP-PXPBD Simulation Loop

```
while not reached maximal iterations do
    for element e do
        while not converged or reached maximal iterations do
            begin Solve Newton system
                1. Compute Newton residual via
                  Equation (29);
                2. Compute $\mathbf{b}_{k+1l}^e$ via Equation 33;
                3. Compute $\delta\mathbf{F}_{k+1l}^e$ via Equation 34 with the
                  approximation in Equation 48;
                4. Compute $\delta\mathbf{x}_{k+1l}^e$ as in Equation 32 ;
                5. Update the nodes on the element with
                  $\mathbf{x}_{i^e k+1l+1}^{n+1} = \mathbf{x}_{i^e k+1l}^{n+1} + \delta\mathbf{x}_{i^e k+1l}^e$;
                6. Update $\mathbf{P}_{k+1l+1}^e = \frac{\partial\Psi}{\partial\mathbf{F}}(\mathbf{F}^e(\mathbf{x}_{i^e k+1l+1}^{n+1}))$;
            end
        end
    end
end
```

3.3.3. First Piola-Kirchhoff Primal XPBD (FP-PXPBD)

Noting that the auxiliary Lagrange multiplier variables are similar to stresses, we observe some convenient properties that arise from choosing an alternative stress measure in an analogous primary/secondary formulation of Equation 9. In a general FEM-discretized hyperelastic formulation (see Barbič and Sifakis [44]), the potential energy gradient has the expression

$$\frac{\partial PE}{\partial x_{i\alpha}}(\mathbf{x}) = \sum_{e,\beta,\gamma} P_{\beta\gamma}(\mathbf{F}^e(\mathbf{x}))\delta_{\alpha\beta}\frac{\partial N_i}{\partial X_\gamma}(\mathbf{X}^e)V^e \tag{23}$$

where $\delta_{\alpha\beta}$ is the Kronecker delta tensor and $\mathbf{P} = \frac{\partial\Psi}{\partial\mathbf{F}}$ is the gradient of the hyperelastic potential energy density with respect to the deformation gradient. This is the first Piola-Kirchhoff stress [6]. If we introduce it as an unknown (analogous to λ_c), then tensor $B_{i\alpha\beta\gamma}^e = \delta_{\alpha\beta}\frac{\partial N_i}{\partial X_\gamma}(\mathbf{X}^e)V^e$ is analogous to the ∇C_c terms in XPBD since they convert the auxiliary (stress) terms to force in the expression in Equation (10). With this formulation, an analogous method consists of

$$\mathbf{g}(\mathbf{x}^{n+1}, \mathbf{P}^{n+1}) = \mathbf{M}\left(\mathbf{x}^{n+1} - \tilde{\mathbf{x}}\right) + \Delta t^2\mathbf{B}\mathbf{P} = \mathbf{0} \tag{24}$$

$$\mathbf{h}^e(\mathbf{x}^{n+1}, \mathbf{P}^{n+1}) = \frac{\partial\Psi}{\partial\mathbf{F}}(\mathbf{F}^e(\mathbf{x}^{n+1})) - \mathbf{P}^e = \mathbf{0}. \tag{25}$$

Note that with this expression, the tensor $\mathbf{B}$ does not depend on the positional unknowns $\mathbf{x}^{n+1}$. In contrast, the analogous expression $\nabla\mathbf{C}(\mathbf{x}^{n+1})$ in Equations (11) does have this dependence, and it is precisely this issue that leads to the red terms in the linearization in Equation (13). Therefore, a formulation based on Equations (24) and (25) rather than Equations (11) and (12) will automatically satisfy the constraint that $\mathbf{g} = \mathbf{0}$ at each Gauss-Newton iteration and will not require the omission of the constraint Hessian since it is exactly zero. We adopt this strategy and iteratively solve Equations (24) and (25) for primary position unknowns $\mathbf{x}_k^{n+1}$ and secondary element stresses $\mathbf{P}_k^e$ in a Gauss-Seiedel manner analogous to that of the original XPBD. We observe that this retains the favorable stability properties of XPBD, while allowing for accurate residual reduction and application to arbitrary hyperelastic constitutive models.

This approach shifts the difficulty from the primary Equation (24) to the secondary Equation (25). It is trivial to maintain a zero primary residual, which simply requires plugging the current guess for the element stresses $\mathbf{P}_k$ into Equation (24) to define the current guess for $\mathbf{x}_k^{n+1}$. We update this guess iteratively by solving for the positions $\mathbf{x}_{k+1}^{n+1,e}$ in element e that satisfy Equation (25). This is equivalent to solving the nonlinear system equations for one element with the stresses in all adjacent elements held fixed, with their dependence on the element positions ignored. We use Ω^e to denote set of the mesh vertices i^e in element e and solve

$$\mathbf{M}^e(\mathbf{x}_{k+1}^{n+1,e} - \tilde{\mathbf{x}}^e) + \Delta t^2 \mathbf{B}^e \mathbf{P}_{k+1}^e = \mathbf{f}^e \tag{26}$$

$$\frac{\partial \Psi}{\partial \mathbf{F}}(\mathbf{F}^e(\mathbf{x}_{k+1}^{n+1})) - \mathbf{P}_{k+1}^e = \mathbf{0} \tag{27}$$

where $f_{i^e\alpha k}^e = \Delta t^2 (f_{i^e\alpha}^{ext} - \sum_{\tilde{e}\neq e,\gamma,\delta} B_{i^e\alpha\gamma\delta}^{\tilde{e}} P_{k\gamma\delta}^{\tilde{e}})$, $\mathbf{f}^{ext}$ is the external force. In index notations Equation 26 can be written as:

$$\sum_{j^e,\beta} m_{i^e j^e} \delta_{\alpha\beta} \left(x_{k+1 j^e\beta}^{n+1} - \tilde{x}_{j^e\beta} \right) + \Delta t^2 \sum_{\gamma,\delta} B_{i^e\alpha\gamma\delta}^e P_{k+1\gamma\delta}^e = f_{i^e\alpha k}^e \tag{28}$$

Here Equation (27) can be satisfied trivially by setting $\mathbf{P}_{k+1}^e = \frac{\partial\Psi}{\partial\mathbf{F}}(\mathbf{F}^e(\mathbf{x}_{k+1}^{n+1}))$. With this simplification, Equations (27)-(28) can be rewritten as

$$\mathbf{M}^e(\mathbf{x}_{k+1}^{n+1,e} - \tilde{\mathbf{x}}^e) + \Delta t^2 \mathbf{B}^e \frac{\partial \Psi}{\partial \mathbf{F}}(\mathbf{F}^e(\mathbf{x}_{k+1}^{n+1,e})) - \mathbf{f}^e = \mathbf{0} \tag{29}$$

where $M_{i^e\alpha j^e\beta}^e = m_{i^e}\delta_{i^e j^e}\delta_{\alpha\beta}$ is portion of the mass matrix (where m_{i^e} is the mass of node i^e) composed of entries only in the element and $\mathbf{x}_k^{n+1,e}$ and $\tilde{\mathbf{x}}^e$ are extractions of element-wise positions from $\mathbf{x}_k^{n+1}$ and $\tilde{\mathbf{x}}$ respectively. Note that $\frac{\partial\Psi}{\partial\mathbf{F}}(\mathbf{F}^e(\mathbf{x}_{k+1}^{n+1})) = \frac{\partial\Psi}{\partial\mathbf{F}}(\mathbf{F}^e(\mathbf{x}_{k+1}^{n+1,e}))$ since the element deformation gradient only depends on the nodes of the element. Lastly, $\mathbf{B}^e$ has entries $B_{i^e\alpha\gamma\delta}^e = \delta_{\alpha\gamma}\frac{\partial N_{i^e}}{\partial X_\delta}(\mathbf{X}^e)V^e$ from Equation (23).

We use Netwon's method to solve Equation (29). $\mathbf{x}_{i^e k+1 l}^{n+1,e}$ denotes the l^{th} iteration of the local Newton procedure for computing the $k+1^{\text{th}}$ global iteration, which modifies the nodes i^e of element e. These nodes are updated in Newton's method as $\mathbf{x}_{i^e k+1 l+1}^{n+1,e} = \mathbf{x}_{i^e k+1 l}^{n+1,e} + \delta\mathbf{x}_{i^e k+1 l}^e$. To solve for $\delta\mathbf{x}_{i^e k+1 l}^e$, we make the following approximation:

$$\frac{\partial \Psi}{\partial \mathbf{F}}(\mathbf{F}^e(\mathbf{x}_{k+1 l+1}^{n+1,e})) \approx \frac{\partial^2 \Psi}{\partial \mathbf{F}^2}(\mathbf{F}^e(\mathbf{x}_{k+1 l}^{n+1,e})) : \delta\mathbf{F}_{k+1 l}^e + \mathbf{P}_{k+1 l}^e \tag{30}$$

and solve the system

$$\mathbf{M}^e \delta\mathbf{x}_{k+1 l}^e + \Delta t^2 \mathbf{B}^e \frac{\partial^2 \Psi}{\partial \mathbf{F}^2}(\mathbf{F}^e(\mathbf{x}_{k+1}^{n+1,e}))\delta\mathbf{F}_{k+1 l}^e = -\mathbf{g}_{k+1 l} \tag{31}$$

where $\mathbf{g}_{k+1 l}^e = \mathbf{M}^e(\mathbf{x}_{k+1 l}^{n+1,e} - \tilde{\mathbf{x}}^e) + \Delta t^2 \mathbf{B}^e \frac{\partial\Psi}{\partial\mathbf{F}}(\mathbf{F}^e(\mathbf{x}_{k+1 l}^{n+1,e})) - \mathbf{f}^e$. This is a linear system of size 12 × 12 (6 × 6 in 2D). To reduce the size of the system, we use an affine basis for the change in positions determined by a Newton step:

$$\delta\mathbf{x}_{i^e k+1 l}^e = \delta\mathbf{F}_{k+1 l}^e (\mathbf{X}_{i^e}^e - \mathbf{X}_{\text{com}}^e) + \mathbf{b}_{k+1 l}^e \tag{32}$$

where $\delta\mathbf{F}_{k+1 l}^e$ are distortional degrees of freedom in the element, $\mathbf{b}_{k+1 l}^e$ are translational degrees of freedom and $\mathbf{X}_{\text{com}}^e$ is the center of mass of the element in the undeformed configuration. Similarly, $\mathbf{X}_{i^e}^e$ refer to the undeformed positions of the vertices in the element. Distortional and translational degrees of freedom can be decoupled for more efficient solution. This can be seen by first summing over $i^e \in \Omega^e$ in Equation 31 and noting that $\sum_{i^e\in\Omega^e} B_{i^e\alpha\gamma\delta}^e = 0$ (when the interpolating functions N_{i^e} satisfy the partition of unity property)

$$\sum_{i^e, j^e, \beta} M_{i^e\alpha j^e\beta} \delta x_{j^e\beta k+1 l}^e \\ + \sum_{i^e,\gamma,\delta,\sigma,\nu} \Delta t^2 B_{i^e\alpha\gamma\delta}^e \frac{\partial^2\Psi}{\partial F_{\gamma\delta}\partial F_{\sigma\nu}}(\mathbf{F}^e(\mathbf{x}_{k+1 l}^{n+1,e}))\delta F_{\sigma\nu k+1 l}^e = \\ \sum_{i^e} m_{i^e}\delta x_{i^e\alpha k+1 l}^e.$$

Next, by defining tensor $D_{i^e\alpha\epsilon\tau}^e = \delta_{\alpha\epsilon}(X_{i^e\tau}^e - X_{\text{com}}^{e\tau})$ relating the original variables to the affine with $\delta x_{i^e\alpha k+1 l}^e = \sum_{\epsilon,\tau} D_{i^e\alpha\epsilon\tau}^e \delta F_{\epsilon\tau k+1 l}^e + b_{\alpha k+1 l}^e$ we see that

$$\sum_{i^e} m_{i^e}\delta x_{i^e\alpha k+1 l}^e = \sum_{i^e,\epsilon,\tau} m_{i^e} D_{i^e\alpha\epsilon\tau}^e \delta F_{\epsilon\tau k+1 l}^e + \sum_{i^e} m_{i^e} b_{\alpha k+1 l}^e \\ = \sum_\tau (m_e X_{\tau\text{com}}^e - m_e X_{\tau\text{com}}^e)\delta F_{\alpha\tau k+1 l}^e + m_e b_{\alpha k+1 l}^e = m_e b_{\alpha k+1 l}^e$$

where $m^e = \sum_{i^e\in\Omega^e} m_{i^e}$ is the element mass and the undeformed element center of mass is defined as $\mathbf{X}^e = \frac{1}{m^e}\sum_{i^e} m_{i^e}\mathbf{X}_{i^e}$. Therefore the translational degrees of freedom can easily be obtained from

$$\mathbf{b}_{k+1 l}^e = \frac{-1}{m_e}\sum_{i^e\in\Omega^e} \mathbf{g}_{i^e k+1 l}. \tag{33}$$

Once the translational component $\mathbf{b}_{k+1 l}^e$ is determined from Equation (33), we multiply Equation 31 on the left side by $\mathbf{D}^{eT}$ to reveal a decoupled system of equations for the distortional degrees of freedom $\delta\mathbf{F}_{k+1 l}^e$

$$\left(\tilde{\mathbf{M}}^e + \Delta t^2 V^e \frac{\partial^2\Psi}{\partial\mathbf{F}^2}(\mathbf{F}_e(\mathbf{x}_{k+1 l}^{n+1}))\right)\delta\mathbf{F}_{k+1 l}^e = -\mathbf{G}_{k+1 l}^e \tag{34}$$

where

$$G_{\eta\nu k+1 l}^e = \sum_{i^e,\alpha} D_{i^e\alpha\eta\nu}^e \left(g_{i^e\alpha k+1 l} + m_{i^e} b_{\alpha k+1 l}^e \right) \tag{35}$$

$$\tilde{M}_{\eta\nu\epsilon\tau}^e = \sum_{i^e, j^e, \alpha} D_{i^e\alpha\eta\nu}^e M_{i^e\alpha j^e\beta}^e D_{j^e\alpha\epsilon\tau}^e. \tag{36}$$

For efficiency note that the matrix $\tilde{\mathbf{M}}^e$ is block diagonal with the structure $\tilde{M}_{\alpha\beta\gamma\delta}^e = \delta_{\alpha\gamma}\hat{M}_{\beta\delta}^e$, where interestingly

$\hat{\mathbf{M}}^e = \sum_{i^e} m_{i^e}(\mathbf{X}_{i^e} - \mathbf{X}^e_{\text{com}})(\mathbf{X}_{i^e} - \mathbf{X}^e_{\text{com}})^T$ is the affine inertia tensor used in [18]. This can be seen from

$$\begin{aligned}\tilde{M}^e_{\eta\nu\epsilon\tau} &= \sum_{i^e, j^e, \alpha, \beta} D^e_{i^e\alpha\eta\nu} m_{i^e} \delta_{i^e j^e} \delta_{\alpha\beta} D^e_{j^e\beta\epsilon\tau} \\ &= \sum_{i^e, j^e, \alpha, \beta} \delta_{\alpha\eta}(X^e_{i^e\nu} - X^e_{\nu\text{com}})) m_{i^e} \delta_{i^e j^e} \delta_{\alpha\beta} \delta_{\alpha\epsilon}(X^e_{j^e\tau} - X^e_{\tau\text{com}})) \\ &= \delta_{\eta\epsilon} \sum_{i^e} m_{i^e}(X^e_{i^e\nu} - X^e_{\nu\text{com}})(X^e_{i^e\tau} - X^e_{\tau\text{com}}).\end{aligned}$$

However note that unlike in [18], the matrix $\hat{\mathbf{M}}^e$ is not in general diagonal.

3.3.4. Partition of unity and reproduction of linear functions

We note that the expression in Equation 34 relies on the identity $\mathbf{D}^T\mathbf{B} = V^e\mathbf{I}$. This can be shown when the interpolating functions N_i form a partition of unity $\sum_i N_i = 1$ and reproduce linear functions as $\mathbf{X} = \sum_i \mathbf{X}_i N_i(\mathbf{X})$. In particular, we show

$$\begin{aligned}(\mathbf{D}^T\mathbf{B})_{\epsilon\tau\beta\nu} &= D^e_{i^e\alpha\epsilon\tau} B^e_{i^e\alpha\beta\nu} && (37)\\ &= \sum_{i^e,\alpha} \delta_{\alpha\beta}\delta_{\alpha\epsilon}(X^e_{i^e\tau} - X^e_{\tau\text{com}}) \frac{\partial N_{i^e}}{\partial X_\nu}(\mathbf{X}^e) V^e && (38)\\ &= \delta_{\epsilon\beta} V^e \sum_{i^e} (X^e_{i^e\tau} - X^e_{\tau\text{com}}) \frac{\partial N_{i^e}}{\partial X_\nu}(\mathbf{X}^e) && (39)\\ &= \delta_{\epsilon\beta} V^e (\sum_{i^e} \frac{\partial N_{i^e}}{\partial X_\nu}(\mathbf{X}^e) X^e_{i^e\tau} - \sum_{i^e} \frac{\partial N_{i^e}}{\partial X_\nu}(\mathbf{X}^e) X^e_{\tau\text{com}}) && (40)\\ &= \delta_{\epsilon\beta} V^e (\sum_{i^e} \frac{\partial N_{i^e}}{\partial X_\nu}(\mathbf{X}^e) X^e_{i^e\tau} - (\sum_{i^e} \frac{\partial N_{i^e}}{\partial X_\nu}(\mathbf{X}^e)) X^e_{\tau\text{com}}) && (41)\\ &= \delta_{\epsilon\beta} V^e (\sum_{i^e} \frac{\partial N_{i^e}}{\partial X_\nu}(\mathbf{X}^e) X^e_{i^e\tau} - \frac{\partial(\sum_{i^e} N_{i^e})}{\partial X_\nu}(\mathbf{X}^e) X^e_{\tau\text{com}}) && (42)\\ &= \delta_{\epsilon\beta} V^e (\sum_{i^e} \frac{\partial N_{i^e}}{\partial X_\nu}(\mathbf{X}^e) X^e_{i^e\tau} - \frac{\partial 1}{\partial X_\nu}(\mathbf{X}^e) X^e_{\tau\text{com}}) && (43)\\ &= \delta_{\epsilon\beta} V^e \sum_{i^e} \frac{\partial N_{i^e}}{\partial X_\nu}(\mathbf{X}^e) \mathbf{X}^e_{i^e\tau} && (44)\\ &= V^e \delta_{\epsilon\beta}\delta_{\tau\nu}. && (45)\end{aligned}$$

3.3.5. Quasi-Newton

In general, solving Equation (34) for the distortional $\delta\mathbf{F}^e_l$ requires the solution of a 9×9 linear system (4×4 in 2D). However, we generally know (or can compute with minimal effort) the eigen decomposition of $\frac{\partial^2\Psi}{\partial\mathbf{F}^2}(\mathbf{F}_e(\mathbf{x}^{n+1,e}_{k+1l}))$ [53, 45]. Since $\tilde{\mathbf{M}}^e$ is constant and block diagonal, its inverse can be precomputed with minimal storage and the inverse of $\tilde{\mathbf{M}}^e + \Delta t^2 \frac{\partial^2\Psi}{\partial\mathbf{F}^2}(\mathbf{F}_e(\mathbf{x}^{n+1}_{k+1l}))$ can be approximated using the Sherman-Morrison rank-1 update formula [15]. However, if all eigen modes are used, this computation can be costly. We therefore use just a few modes in a quasi-Newton strategy, where the cost of the slow down in Newton convergence must be balanced against higher computational time per iteration, brought by using more modes in the Sherman-Morrison formula. In the case of the corotated model, we can use

$$\begin{aligned}\frac{\partial^2\Psi}{\partial\mathbf{F}^2}(\mathbf{F}^e) &= 2\mu\mathbf{I} + 2\mu\frac{\partial\mathbf{R}(\mathbf{F})}{\partial\mathbf{F}} + \lambda\frac{\partial\det(\mathbf{F}^e)}{\partial\mathbf{F}^e} \otimes \frac{\partial\det(\mathbf{F}^e)}{\partial\mathbf{F}^e} && (46)\\ &\quad + \lambda\det(\mathbf{F}^e)\frac{\partial^2\det(\mathbf{F}^e)}{\partial(\mathbf{F}^e)^2} && (47)\\ &\approx 2\mu\mathbf{I} + \lambda\frac{\partial\det(\mathbf{F}^e)}{\partial\mathbf{F}^e} \otimes \frac{\partial\det(\mathbf{F}^e)}{\partial\mathbf{F}^e} && (48)\end{aligned}$$

where $\mathbf{I}$ is the 9×9 (4×4 in 2D) identity matrix. Note the term $2\mu\frac{\partial\mathbf{R}(\mathbf{F})}{\partial\mathbf{F}} + \lambda\det(\mathbf{F}^e)\frac{\partial^2\det(\mathbf{F}^e)}{\partial(\mathbf{F}^e)^2}$ is omitted in the approximation. With this approximation, we can use the Sherman-Morrison formula for

$$\left(\tilde{\mathbf{M}}^e + \Delta t^2 V^e \frac{\partial^2\Psi}{\partial\mathbf{F}^2}(\mathbf{F}_e(\mathbf{x}^{n+1}_{k+1l}))\right)^{-1} \approx \mathbf{Z}^e - \frac{\mathbf{Z}^e(\mathbf{W}^e \otimes \mathbf{W}^e)\mathbf{Z}^e}{1 + \mathbf{W}^e : (\mathbf{Z}^e\mathbf{W}^e)} \tag{49}$$

where $\mathbf{W}^e = \frac{\partial\det(\mathbf{F}^e)}{\partial\mathbf{F}^e}$ and $\mathbf{Z}^e = (\tilde{\mathbf{M}}^e + 2V^e\Delta t^2\mu\mathbf{I})^{-1}$. Note that $\mathbf{Z}^e$ is constant and has the same symmetric block diagonal structure as $\tilde{\mathbf{M}}^e$ so its inverse can be precomputed and stored with only 6 floats (3 in 2D).

While the procedure outlined in Section 3.3.5 requires some elaborate notation, we note that it is effectively a standard Newton's method for FEM-discretized hyperelasticity on a single element. The only difference is that the stresses from the neighboring elements do not change when the element nodal positions change. This is inherent in the introduction of the stresses $\mathbf{P}^e$ as additional variables in Equations (24)-(25). The stresses from the neighboring elements just contribute forces that effect the right hand side of the Newton procedure, but not the matrix in the linearization. We summarize the process in Algorithm 2. In general, we run with 1-5 Newton iterations. As discussed in Section 3.3.5, with our novel approximation of the Hessian, the cost of solving the linear system becomes trivial. The major cost of the computation time for both XPBD and FP-PXPBD is computing the singular value decomposition of $\mathbf{F}^e$. As shown in Section 5.1, 5.2, 5.3 and 5.4 the speed of FP-PXPBD is comparable to XPBD.

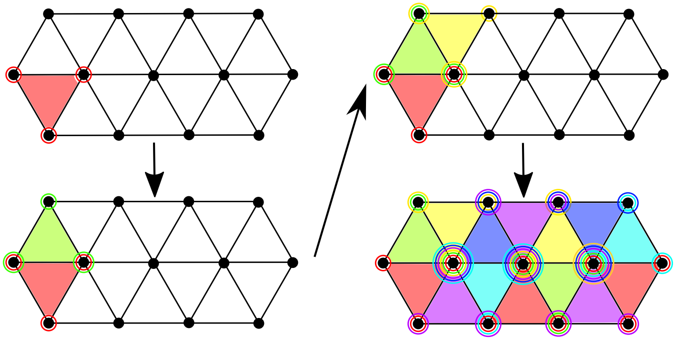


Fig. 5: **Triangle Mesh Coloring**. A step-by-step illustration of coloring a mesh in 2D is shown. Each node registers the colors used by its incident triangles. We go over each triangle to determine its color as the minimal color unregistered by its incident nodes. All its incident nodes then register the triangle's color as used.

4. Parallelism

Computation of the element-wise updates must be done in parallel for optimal efficiency. Even though we use a Gauss-Seidel (as opposed to Jacobi) approach, we can achieve this with careful ordering of element-wise updates. This was discussed by Macklin and Müller [28], however their approach is limited to tetrahedral meshes created from hexahedral meshes. We provide a simple coloring scheme that works for all tetrahedron meshes. The coloring is done so that elements in the same color do not share vertices and can be updated in parallel without race conditions. For each vertex $\mathbf{x}_i$ in the mesh we maintain a set $S_{\mathbf{x}_i}$ that stores the colors used by its incident elements. For each mesh element e, we find the minimal color that is not contained in the set $\cup_{\mathbf{x}_{i^e} \in e} S_{\mathbf{x}_{i^e}}$. Then, we register the color by adding it into $S_{\mathbf{x}_{i^e}}$ for each $\mathbf{x}_{i^e}$ in element e. This coloring strategy does not depend on the topology of the mesh and requires only a one-time cost at the beginning of the simulation. The process is illustrated in Figure 5.

For the grid-based variation mentioned in Section 5.5, we do a similar process as the coloring scheme for the mesh, except the incident points of an element are now a subset of the grid nodes. Since the particle positions are interpolated by grid nodes, an element would be incident to all the grid nodes that interpolate to its incident particles on the mesh. So for each element, we choose its color as the the color with the least color index such that it is not yet registered by the incident grid nodes. The process is illustrated in Figure 6. Since grid nodes incident to an element change every timestep, the elements have to be recolored every timestep. We speed up the coloring process by using the coloring results from previous timestep as an initial guess. We note that Fratarcangeli et al. [11] develop a randomized and effective ordering technique that could be used here as well.

5. Examples

We demonstrate our methods in a variety of representative scenarios with elastic deformation. Our approach has comparable computational complexity to XPBD, so we only provide limited run-time statistics in Table 1. Examples run with the corotated model (Equation 3) use the algorithm from [12] for its accuracy and efficiency. All the examples were run on an AMD Ryzen Threadripper PRO 3995WX CPU with 64 cores and 128 threads.

In each of the examples, we compute the mass m_i of node $\mathbf{x}_i$ from a user-specified density ρ. We denote $\mathcal{I}$ to be the set of elements that contain node i. We define

$$m_i = \sum_{e \in \mathcal{I}} \frac{V^e \rho}{n_e} \tag{50}$$

Then the mass matrix is set with $M_{i\alpha j\beta} = \delta_{ij}\delta_{\alpha\beta} m_i$. We compute Lamé parameters μ and λ with Poisson ratio ν and Young's modulus E. They are computed as following:

$$\mu = \frac{E}{2(1+\nu)}, \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{51}$$

For all the examples in this paper we set Poisson ratio $\nu = 0.3$ and density $\rho = 10$.

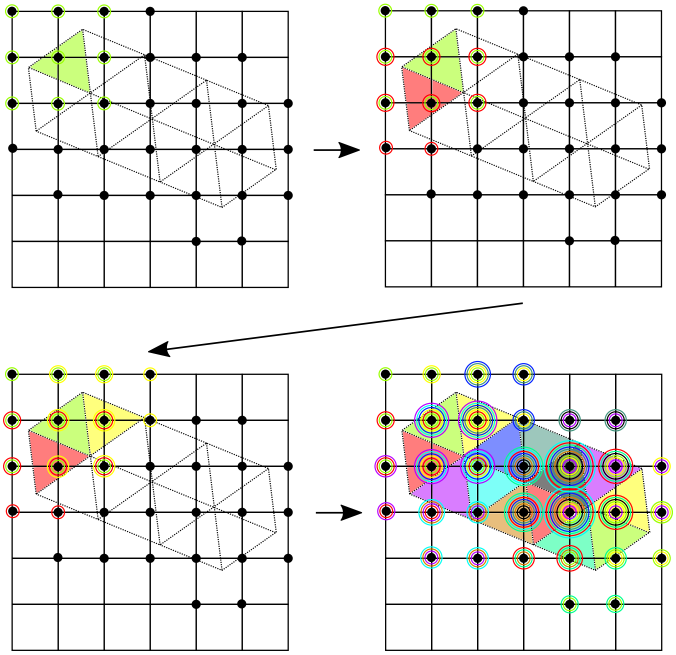


Fig. 6: **Grid-based Mesh Coloring**. A step-by-step grid-based simplex mesh coloring scheme for 2D is shown. The illustration assumes the grid uses linear interpolation where interpolation over a cell only requires the 4 grid nodes on its corners. An element can have at maximum 12 incident grid nodes. After the first element is colored green, 9 grid nodes that are incident will register green as a used color. The elements incident to those nodes then cannot be labeled green.

5.1. Residual Comparison

We compare the residual reduction between XPBD, B-PXPBD and FP-PXPBD. Figure 1 (right) shows the residual reduction in a representative time step of a simple elasticity simulation. While B-PXBD and FP-PXBD continually reduce the nonlinear backward Euler residual, XPBD stagnates. Note that XPBD effectively reduces the auxiliary residual, but not the primary residual and that it makes rapid initial progress when the omission of the primary residual is well-justified. The example setup is the same as the one shown in Figure 3(b). B-PXPBD has blending parameter $\zeta = 0.5$.

5.2. Equal Budget Comparison

In Figure 2 we compare methods when simulated with a restricted computational budget. At the left we show Newton's method run to full-convergence (residual of Equation (9) less than $1e^{-8}$), which is computationally expensive. Then, we compare (from left to right) Newton's method, FP-PXPBD, B-PXPBD and XPBD when only allowed 200ms of computation time per frame. Newton's method is remarkably unstable, but the XPBD-style methods are stable and visually plausible. Here we fix the left side of the tetrahedron mesh created from a $32 \times 32 \times 32$ grid and apply gravity. The Young's modulus is $E = 1000$ and the time step is $\Delta t = 0.01$. B-PXPBD has blending parameter $\zeta = 0.1$.

5.3. XPBD Hyperelastic

In this example we demonstrate that XPBD is incapable of dealing with certain hyperelastic models. The top bar is simulated with XPBD and the corotated model, where the constraint

Table 1: Timing Comparisons: runtime is measured for each frame (averaged over the course of the simulation). Each frame is run after advancing time .033.

Example	# Vertices	# Elements.	XPBD Runtime	B-PXPBD Runtime	FP-PXPBD Runtime	XPBD # iter	B-PXPBD # iter	FP-PXPBD # iter
Residual Reduction (Figure 3(b))	4K	17K	200ms	200ms	216ms	40	40	40
Equal Budget Comparison (Figure 2)	33K	149K	210ms	210ms	200ms	7	7	5
XPBD Hyperelastic (Figure 7)	4K	17K	22ms	-	44ms	4	-	4
XPBD Neohookean (Figure 8)	4K	17K	795ms	-	345ms	400	-	40
Simple Muscle (Figure 4)	5k	20k	-	-	160ms	-	-	4

is reformulated as $C_e(\mathbf{x}) = \sqrt{\Psi^{cor}(\mathbf{F}^e)}$. The middle bar is simulated with FP-PXPBD and the bottom bar XPBD as formulated in Equation (5). As demonstrated in Figure 7, the top bar becomes unstable after a couple of time steps. The reformulation at the top is a simple means of addressing general hyperelasticity with a XPBD formulation, however it does not behave stably. For this example we take a rectangular mesh and clamp both ends which are then stretched and then squeezed. We set Young's modulus as $E = 1e^4$ and time step $\Delta t = 0.01$.

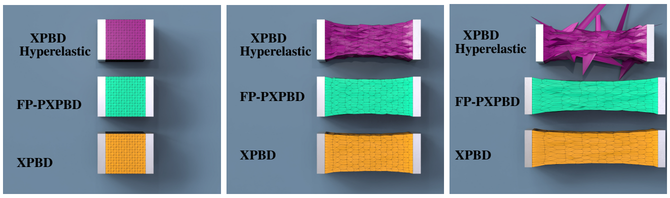


Fig. 7: **XPBD Hyperelastic**. Defining the XPBD constraint as the square root of the hyperelastic potential is not stable (top). Results at frame 0, 10, 30 are shown.

5.4. *XPBD Neohookean*

In this example we compare XPBD and FP-PXPBD when used with the Neo-Hookean model proposed in Macklin et al.[28]. We generalize the low-rank approximation to the Hessian from Equation 48 to this model as

$$\frac{\partial^2\Psi}{\partial \mathbf{F}^2}(\mathbf{F}_e) \approx \mu\mathbf{I} + \lambda\frac{\partial \det(\mathbf{F}^e)}{\partial \mathbf{F}^e} \otimes \frac{\partial \det(\mathbf{F}^e)}{\partial \mathbf{F}^e} \tag{52}$$

Similarly, we can approximate the Hessian inverse as in Equation 49 with $\mathbf{Z}^e = (\tilde{\mathbf{M}}^e + V^e\Delta t^2\mu\mathbf{I})^{-1}$. The test scenario is similar to that in Section 5.3. We use Young's modulus $E = 1000$ and time step $\Delta t = 0.01$. Results are shown in Figure 8. The top bar is simulated with XPBD and is run with 100 iterations per time step. However, it does not converge to the ground truth run with Newton's method, which is shown in the bottom row. It is also visibly less volume conserving. On the other hand, FP-PXPBD converges to the ground truth with 10 iterations per time step.

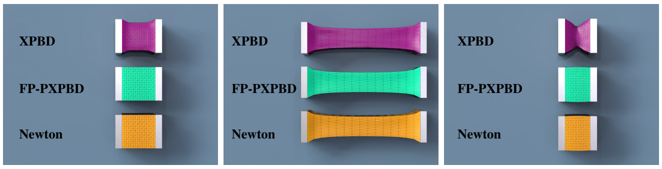
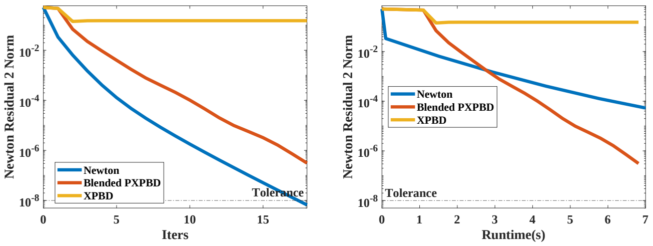


Fig. 8: **XPBD Neohookean**. XPBD is less volume-conserving than FP-PXPBD when the cube is squeezed. Results at frame 1, 25, 52 are shown.



(a)

(b)

Fig. 9: **(a) Grid-Based Residual vs. Iterations**. The residual norm vs. iterations is plotted at a representative time step with grid-based collision. Newton's method and B-PXPBD reliably reduce the residual, but XPBD stagnates. **(b) Grid-based Residual vs. Runtime**. The residual norm vs. computation time is plotted at a representative time step. Grid-based B-PXPBD and grid-based XPBD take an extra 1 second at the beginning of each timestep to compute preprocessing data. Note that B-PXPBD achieves fasters convergence than Newton's method.

5.5. *Grid-Based B-PXPBD Examples*

We showcase the versatility and the robustness of B-PXPBD through a variety of collision intensive examples. We use the grid-based approach of Jiang et al. [18] since this approach does not require modification of the potential energy to address collision/contact and therefore clearly demonstrates our solver performance. Here, the backward Euler degrees of freedom are over a regular grid where the tetrahedron mesh is embedded/interpolated from its motion. That is, we use B-PXPBD to solve the system of equations for implicit time stepping outlined in Jiang et al. [18], but where the energy is written in the XPBD way using the constraints Equation (5). This requires a modification to the coloring strategy used for parallel implementation (see supplementary technical document for specifics [60]) but is otherwise a straightforward application of our techniques so we omit explicit detail.

In Figure 1, we drop 30 objects stacked on top of each other into a glass box. The objects include bunnies, dragons, balls, boxes and tori. The bunny mesh used is obtained from [54] The total vertex count of the mesh is around 800,000. We visualize the convergence behaviors of grid-based XPBD, grid-based B-PXPBD and Newton's method in Figure 9. While the residual of grid-based XPBD stagnates, grid-based B-PXPBD continually reduces the nonlinear residual. Though grid-based B-PXPBD has a convergence rate that is slower than Newton's method, it has a much lower computational budget than Newton's method. As the right plot in Figure 9 indicates, given the same computational budget, grid-based B-PXPBD would reduce residual more than Newton's method. On average, the grid-based B-PXPBD takes 17.6s/frame, whereas Newton's method takes 58.9s/frame. We demonstrate more collision-intensive scenarios in Figures 10, 12 and Figure 11. For these examples, the Young's modulus is $E = 1000$ and CFL number is .4. B-PXPBD has blending parameter $\zeta = 0.5$.

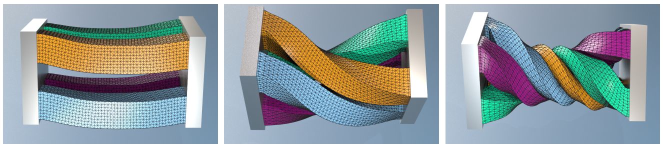


Fig. 10: **Four Bars Twisting**. Grid-based B-PXPBD is capable of handling large deformation and complex collisions.

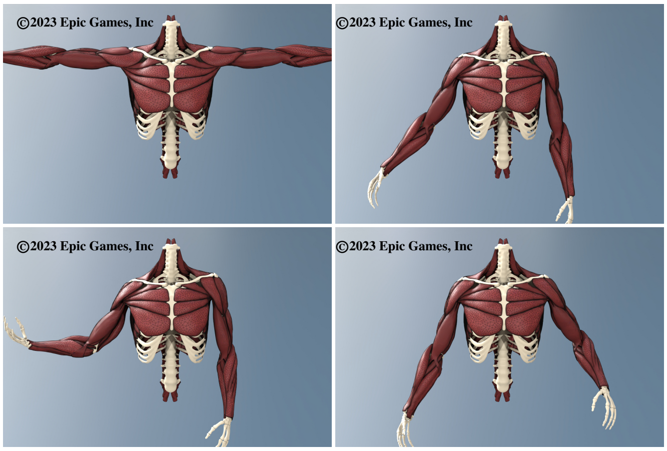


Fig. 11: **Muscle**. Large-scale muscle simulation using grid-based B-PXPBD. Frames 0, 30, 60, 140 are shown.

6. Discussion and Limitations

Our framework effectively addresses XPBD convergence issues with hyperelasticity and allows for generalization to arbitrary constitutive models. Furthermore, we attain the favorable stability and efficiency properties that make PBD and XPBD techniques so powerful. However, our approach does have limitations.

With B-PXPBD the blending parameter ζ can require numerous simulations to establish a useful value. FP-PXPBD is more general, but the local step may be more costly and the Sherman-Morrison formula must be applied on a case-by-case basis for different constitutive models. Lastly, while the grid-base approach of Jiang et al. [18] is an easy way to handle collisions within our framework, it is not ideal for many applications since a grid-based CFL must still be used. Furthermore, it does not naturally allow for use with FP-PXBPD because the accelerations we apply in the local step are not directly applicable. We provide two approaches B-PXPBD and FP-PXPBD since both have different strengths and weaknesses. While B-PXPBD is simple to implement (and requires only a small modificaiton to an existing XPBD code), it requires the tuning of the blending parameter and does not address general hyperelastic models models. Alternatively, FP-PXPBD can be applied to general models but it requires a larger deviation from an existing XPBD code. Furthermore, the number of iterations in the Newton method for each element effects the efficiency of the approach relative to B-PXPBD. In practice, these considerations must be weighed when deciding which approach to use.

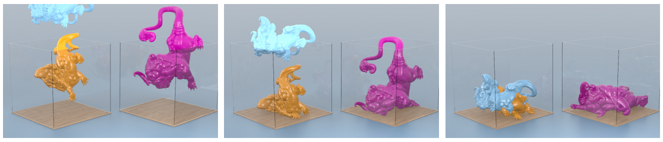


Fig. 12: **Dropping Dragons**. Grid-based simulation with B-PXPBD exhibits many collision-driven large deformations.

References

[1] D. Baraff and A. Witkin. 1998. Large Steps in Cloth Simulation. In *Proc ACM SIGGRAPH (SIGGRAPH '98)*. 43–54.

[2] J. Barbič and D. James. 2005. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. Graph.* 24, 3 (2005), 982?990.

[3] C. Batty, F. Bertails, and R. Bridson. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Trans Graph* 26, 3 (2007).

[4] D. Bertsekas. 1997. Nonlinear programming. *J Op Res Soc* 48, 3 (1997), 334–334.

[5] S. Blemker, P. Pinsky, and S. Delp. 2005. A 3D model of muscle reveals the causes of nonuniform strains in the biceps brachii. *J. Biomech* 38, 4 (2005), 657–665.

[6] J. Bonet and R. Wood. 2008. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press.

[7] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans Graph* 33, 4 (2014), 154:1–154:11.

[8] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning* 3, 1 (2011), 1–122.

[9] I. Chao, U. Pinkall, P. Sanan, and P. Schröder. 2010. A Simple Geometric Model for Elastic Deformations. *ACM Trans Graph* 29, 4, Article 38 (2010), 6 pages.

[10] Y. Fan, J. Litven, and D. Pai. 2014. Active Volumetric Musculoskeletal Systems. *ACM Trans Graph* 33, 4 (2014), 152:1–152:9.

[11] M. Fratarcangeli, T. Valentina, and F. Pellacini. 2016. Vivace: a practical gauss-seidel method for stable soft body dynamics. *ACM Trans Graph* 35, 6 (Nov 2016), 1–9. `https://doi.org/10.1145/2980179.2982437`

[12] T. Gast, C. Fu, C. Jiang, and J. Teran. 2016. *Implicit-shifted Symmetric QR Singular Value Decomposition of 3x3 Matrices*. Technical Report. University of California Los Angeles.

[13] T. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J. Teran. 2015. Optimization Integrator for Large Time Steps. *IEEE Trans Vis Comp Graph* 21, 10 (2015), 1103–1115.

[14] R. Goldenthal, D. Harmon, R. Fattal, M. Bercovier, and E. Grinspun. 2007. Efficient Simulation of Inextensible Cloth. *ACM Trans Graph* 26, 3, Article 49 (July 2007).

[15] W. Hager. 1989. Updating the inverse of a matrix. *SIAM review* 31, 2 (1989), 221–239.

[16] F. Hecht, Y. Lee, J. Shewchuk, and J. O'Brien. 2012. Updated Sparse Cholesky Factors for Corotational Elastodynamics. *ACM Trans Graph* 31, 5, Article 123 (2012), 13 pages.

[17] G. Irving, J. Teran, and R. Fedkiw. 2004. Invertible Finite Elements for Robust Simulation of Large Deformation. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*. 131–140.

[18] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. 2015. The Affine Particle-In-Cell Method. *ACM Trans Graph* 34, 4 (2015), 51:1–51:10.

[19] L. Kharevych, W. Yang, Y. Tong, E. Kanso, J.E. Marsden, P. Schröder, and M. Desbrun. 2006. Geometric, Variational Integrators for Computer Animation. In *Proc 2006 ACM SIGGRAPH/Eurograph Symp Comp Anim* (Vienna, Austria) *(SCA '06)*. Eurographics Association, 43?51.

[20] T. Kim, F. De Goes, and H. Iben. 2019. Anisotropic Elasticity for Inversion-Safety and Element Rehabilitation. *ACM Trans. Graph.* 38, 4 (2019), 15 pages.

[21] S. Kovalsky, M. Galun, and Y. Lipman. 2016. Accelerated Quadratic Proxy for Geometric Optimization. *ACM Trans Graph* 35, 4, Article 134 (2016), 11 pages.

[22] N. Kwatra, J. Gretarsson, and R. Fedkiw. 2010. Practical Animation of Compressible Flow for ShockWaves and Related Phenomena.. In *Symp Comp Anim*. 207–215.

[23] N. Kwatra, J. Su, J. Grétarsson, and R. Fedkiw. 2009. A method for avoiding the acoustic time step restriction in compressible flow. *J Comp Phys* 228, 11 (2009), 4146–4161.

[24] M. Li, M. Gao, T. Langlois, C. Jiang, and D. Kaufman. 2019. Decomposed Optimization Time Integrator for Large-Step Elastodynamics. *ACM Trans Graph* 38, 4 (jul 2019), 10 pages. https://doi.org/10.1145/3306346.3322951

[25] L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. Gortler. 2008. A Local/Global Approach to Mesh Parameterization. In *Proc Symp Geom Proc (SGP '08)*. Eurograph Assoc, 1495?1504.

[26] T. Liu, A. Bargteil, J. O'Brien, and L. Kavan. 2013. Fast Simulation of Mass-Spring Systems. *ACM Trans Graph* 32, 6 (2013), 209:1–7.

[27] T. Liu, S. Bouaziz, and L. Kavan. 2017. Quasi-Newton Methods for Real-Time Simulation of Hyperelastic Materials. *ACM Trans Graph* 36, 4, Article 116a (2017), 16 pages.

[28] M. Macklin and M. Muller. 2021. A Constraint-based Formulation of Stable Neo-Hookean Materials. In *Motion, Interaction and Games*. ACM, 1–7. https://dl.acm.org/doi/10.1145/3487983.3488289

[29] M. Macklin, M. Müller, and N. Chentanez. 2016. XPBD: Position-Based Simulation of Compliant Constrained Dynamics. In *Proc 9th Int Conf Motion Games* (Burlingame, California) *(MIG '16)*. ACM, 49?54.

[30] S. Martin, B. Thomaszewski, E. Grinspun, and M.Gross. 2011. Example-Based Elastic Materials. In *ACM SIGGRAPH 2011* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. ACM, Article 72, 8 pages.

[31] A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis. 2011. Efficient Elasticity for Character Skinning with Contact and Collisions. *ACM Trans Graph* 30, 4 (2011), 37:1–37:12.

[32] V. Modi, L. Fulton, A. Jacobson, S. Sueda, and D. Levin. 2021. Emu: Efficient muscle simulation in deformation space. In *Comp Graph Forum*, Vol. 40. Wiley Online Library, 234–248.

[33] M.Tournier, M. Nesme, B. Gilles, and F. Faure. 2015. Stable Constrained Dynamics. *ACM Trans Graph* (2015), 1–10.

[34] M. Müller and M. Gross. 2004. Interactive virtual materials. In *Proc Graph Int*. Canadian Human-Computer Communications Society, 239–246.

[35] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. 2007. Position based dynamics. *J Vis Comm Im Rep* 18, 2 (2007), 109–118.

[36] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. 2005. Meshless deformations based on shape matching. In *ACM transactions on graphics (TOG)*, Vol. 24. ACM, 471–478.

[37] R. Narain, M. Overby, and G. Brown. 2016. ADMM Projective Dynamics: Fast Simulation of General Constitutive Models. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim* (Zurich, Switzerland) *(SCA '16)*. Eurograph Assoc, 21?28.

[38] J. Neuberger. 1985. Steepest descent and differential equations. *J Math Soc Japan* 37, 2 (1985), 187–195.

[39] J. Nocedal and S. Wright. 2006. Conjugate gradient methods. *Num Opt* (2006), 101–134.

[40] J. O'Brien, A. Bargteil, and J. Hodgins. 2002. Graphical modeling and animation of ductile fracture. In *Proc ACM SIGGRAPH 2002*. 291–294.

[41] J. O'Brien and J. Hodgins. 1999. Graphical modeling and animation of brittle fracture. In *Proc 26th Conf Comp Graph Int Tech (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., 137–146. https://doi.org/10.1145/311535.311550

[42] X. Provot. 1995. Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Graph Int*. Canadian Information Processing Society, 147–155.

[43] M. Rabinovich, R. Poranne, D. Panozzo, and O. Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans Graph* 36, 2, Article 16 (2017), 16 pages.

[44] E. Sifakis and J. Barbic. 2012. FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses (SIGGRAPH '12)*. ACM, 20:1–20:50.

[45] B. Smith, F. Goes, and T. Kim. 2019. Analytic eigensystems for isotropic distortion energies. *ACM Trans Graph (TOG)* 38, 1 (2019), 1–15.

[46] B. Smith, F. De Goes, and T. Kim. 2018. Stable neo-hookean flesh simulation. *ACM Trans Grap (TOG)* 37, 2 (2018), 1–15.

[47] O. Sorkine and M. Alexa. 2007. As-Rigid-As-Possible Surface Modeling. In *EUROGRAPHICS SYMPOSIUM ON GEOMETRY PROCESSING*.

[48] A. Stern and M. Desbrun. 2006. Discrete Geometric Mechanics for Variational Time Integrators. In *ACM SIGGRAPH 2006 Courses* (Boston, Massachusetts) *(SIGGRAPH '06)*. ACM, 75?80.

[49] A. Stomakhin, R. Howes, C. Schroeder, and J. Teran. 2012. Energetically consistent invertible elasticity. In *Proc Symp Comp Anim*. 25–32.

[50] A. Stomakhin, C. Schroeder, C. Jiang, L. Chai, J. Teran, and A. Selle. 2014. Augmented MPM for phase-change and varied materials. *ACM Trans Graph* 33, 4 (2014), 138:1–138:11.

[51] J. Teran, S. Blemker, V. Hing, and R. Fedkiw. 2003. Finite volume methods for the simulation of skeletal muscle. In *Proc 2003 ACM SIGGRAPH/Eurograph Symp Comp Anim*. Eurographics Association, 68–74.

[52] J. Teran, E. Sifakis, S. Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw. 2005. Creating and simulating skeletal muscle from the visible human data set. *IEEE Trans Vis Comp Graph* 11, 3 (2005), 317–328.

[53] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw. 2005. Robust quasistatic finite elements and flesh simulation. In *Proc 2005 ACM SIGGRAPH/Eurograph Symp Comp Anim*. 181–190.

[54] G. Turk and M. Levoy. 1994. Stanford Bunny. http://graphics.stanford.edu/data/3Dscanrep/ Stanford University Computer Graphics Laboratory.

[55] B. Wang, M. Zheng, and J. Barbič. 2020. Adjustable Constrained Soft-Tissue Dynamics. *Pac Graph 2020 and Comp Graph Forum* 39, 7 (2020).

[56] H. Wang. 2015. A Chebyshev Semi-Iterative Approach for Accelerating Projective and Position-Based Dynamics. *ACM Trans Graph* 34, 6, Article 246 (nov 2015), 9 pages.

[57] H. Wang and Y. Yang. 2016. Descent methods for elastic body simulation on the GPU. *ACM Trans Graph* 35, 6 (Nov 2016), 1–10. https://doi.org/10.1145/2980179.2980236

[58] B. Witemeyer, N. Weidner, T. Davis, T. Kim, and S. Sueda. 2021. QLB: Collision-Aware Quasi-Newton Solver with Cholesky and L-BFGS for Nonlinear Time Integration. In *Proc 14th ACM SIGGRAPH Conf Mot Int Games (MIG '21)*. ACM, Article 14, 7 pages.

[59] H. Xu, F. Sin, Y. Zhu, and J. Barbič. 2015. Nonlinear Material Design Using Principal Stretches. *ACM Trans Graph* 34, 4 (2015).

[60] Y.Chen, Y.Han, J.Chen, S.Ma, R.Fedkiw, and J.Teran. 2023. *Supplementary Technical Document* (2023).

[61] Y.Chen, Y.Han, J.Chen, S.Ma, R.Fedkiw, and J.Teran. 2023. Primal Extended Position Based Dynamics for Hyperelasticity, In MIG '23: Proceedings of the 16th ACM SIGGRAPH Conference on Motion, Interaction and Games. *MIG '23: Proceedings of the 16th ACM SIGGRAPH Conference on Motion, Interaction and Games*, 1–10.

[62] D. Zhao, Y. Li, and J. Barbič. 2016. Asynchronous Implicit Backward Euler Integration. (2016).

[63] Y. Zhu, R. Bridson, and D. Kaufman. 2018. Blended Cured Quasi-Newton for Distortion Optimization. *ACM Trans Graph* 37, 4, Article 40 (jul 2018), 14 pages.