Position-Based Nonlinear Gauss-Seidel for Quasistatic Hyperelasticity

YIZHOU CHEN, University of California, Los Angeles Epic Games, Inc, USA
YUSHAN HAN, University of California, Los Angeles Epic Games, Inc, USA
JINGYU CHEN, University of California, Los Angeles, USA
ZHAN ZHANG, University of California, Davis, USA
ALEX MCADAMS, Epic Games, Inc, USA
JOSEPH TERAN, University of California, Davis
Epic Games, Inc, USA

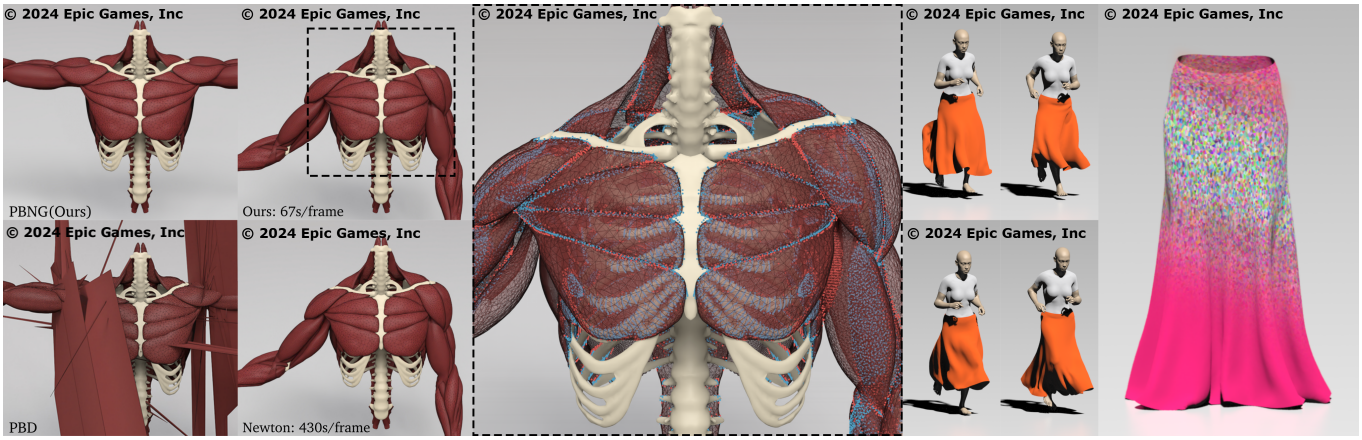

Fig. 1. **Quasistatic Muscle Simulation with Collisions**. **Left.** Our method (PBNG) produces high-quality results visually comparable to Newton's method but with a 6x speedup. PBD (lower left) becomes unstable with this quasistatic example after a few iterations. **Middle.** In this hyperelastic simulation of muscles, we use weak constraints to bind muscles together and resolve collisions. *Red* indicates a vertex involved in a contact constraint. *Blue* indicates a vertex is bound with connective tissues. **Right.** A dress of 24K particles is simulated with MPBNG on a running mannequin. The rightmost image visualizes our multiresolution mesh.

Position based dynamics [Müller et al. 2007] is a powerful technique for simulating a variety of materials. Its primary strength is its robustness when run with limited computational budget. Even though PBD is based on the projection of static constraints, it does not work well for quasistatic problems. This is particularly relevant since the efficient creation of large data sets of plausible, but not necessarily accurate elastic equilibria is of increasing importance with the emergence of quasistatic neural networks [Bailey et al. 2018; Chentanez et al. 2020; Jin et al. 2022; Luo et al. 2020]. Recent work [Macklin et al. 2016] has shown that PBD can be related to the Gauss-Seidel approximation of a Lagrange multiplier formulation of backward Euler time stepping, where each constraint is solved/projected independently of the others in an iterative fashion. We show that a position-based, rather than constraint-based nonlinear Gauss-Seidel approach resolves a number of issues with PBD, particularly in the quasistatic setting. Our approach retains the essential PBD feature of stable behavior with constrained computational budgets, but also allows for convergent behavior with expanded budgets. We demonstrate the efficacy of our method on a variety of representative hyperelastic problems and show that both successive over relaxation (SOR), Chebyshev and multiresolution-based acceleration can be easily applied.

CCS Concepts: • **Computing methodologies** → Realtime Simulation.

Additional Key Words and Phrases: Position-based dynamics, physics simulation, constrained dynamics, quasistatics simulation

ACM Reference Format:
Yizhou Chen, Yushan Han, Jingyu Chen, Zhan Zhang, Alex McAdams, and Joseph Teran. 2024. Position-Based Nonlinear Gauss-Seidel for Quasistatic Hyperelasticity. *ACM Trans. Graph.* 1, 1, Article 115 (July 2024), 15 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' addresses: Yizhou Chen, chenyizhou@ucla.edu, University of California, Los Angeles Epic Games, Inc, USA; Yushan Han, yushanh1@math.ucla.edu, University of California, Los Angeles Epic Games, Inc, USA; Jingyu Chen, chenjy@g.ucla.edu, University of California, Los Angeles, USA; Zhan Zhang, zzzzhan@ucdavis.edu, University of California, Davis, USA; Alex McAdams, alex.mcadams@epicgames.com, Epic Games, Inc, USA; Joseph Teran, jteran@math.ucdavis.edu, University of California, Davis
Epic Games, Inc, USA.

0730-0301/2024/7-ART115 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

We consider large strain hyperelastic solids [Bonet and Wood 2008] whose governing equations are discretized in space with the finite element method (FEM) [Sifakis and Barbic 2012]. Our primary focus is quasistatic problems with negligible inertial effects. Quasistatic

solvers are becoming increasingly important due to their use in generating training data for neural networks (or QNNS: quasistatic neural networks). For example, various authors have shown that QNNs can be effectively trained for elastic materials in cloth and skinning applications [Bailey et al. 2018; Bertiche et al. 2021; Chentanez et al. 2020; Geng et al. 2020; Jin et al. 2020, 2022; Luo et al. 2020]. These networks engender real-time performance at resolutions orders of magnitude above what is achievable with any existing simulation techniques on modern hardware. However, QNNs require tens of thousands of high-resolution equilibria for training data sets. While the creation of these data sets is an "off-line" process, it is desirable to create them with minimal user interaction and computation time. Furthermore, extremely accurate solutions to the governing equations are not necessary since the network need only approximate visually convincing behaviors. Therefore, simulation techniques that generate visually plausible behavior in a minimal amount of computation with minimal user interaction/parameter tuning are ideal.

While many methods exist for solving the FEM-discretized implicit equations of motion for hyperelastic solids (see Zhu et al. [2018] and Li et al. [2019] for recent summaries), the Position Based Dynamics (PBD) approach of Müller et al. [2007] is a natural candidate for generating training data for QNNs. It has remarkable robustness and stability properties and can produce visually plausible results with minimal computational budgets. 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. [2016] addressed 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.

Despite its many strengths, PBD/XPBD has a few limitations that hinder its use in quasistatic applications. First, XPBD is designed for backward Euler and omitting the inertial terms for quasistatics is not possible (it would require dividing by zero). Indeed Chentanez et al. [2020] generate quasistatic training data with XPBD by running backward Euler simulations to steady state. We show that PBD when viewed as the limit of infinite stiffness in XPBD (as detailed in Macklin et al. [2016]) is an approximation to the quasistatic equations. Unfortunately, this limit incorrectly and irrevocably removes the external forcing terms. Second, PBD/XPBD can only discretize hyperelastic models that are quadratic in some notion of strain constraint [Macklin and Muller 2021; Macklin et al. 2016]. As noted in [Chen et al. 2023], simply interpreting the square root of the hyperelastic potential as the constraint results in instability. This prevents the adoption of many models from the computational mechanics literature. Lastly, as noted in Chen et al. [2023] the constraint-centric Gauss-Seidel iteration in PBD/XPBD does not reliably reduce time stepping system residuals. We show that in quasistatic problems this causes artifacts near vertices that appear in different types of constraints (see Figure 2).

We present a position-based (rather than constraint-based) nonlinear Gauss-Seidel method that resolves the key issues with PBD/XPBD and hyperelastic quasistatic time stepping. In our approach, we iteratively adjust the position of each simulation node to minimize the potential energy (with all other coupled nodes fixed) in a Gauss-Seidel fashion. This makes each position update aware of all constraints that a node participates in and removes the artifacts of PBD/XPBD that arise from processing constraints separately. Our approach maintains the essential efficiency and robustness features of PBD and has an accuracy that rivals Newton's method for the first few orders of magnitude in residual reduction. Furthermore, unlike Newton's method, our approach is stable when the computational budget is extremely limited. Lastly, since our approach is based on Gauss-Seidel, we show that its convergence is naturally accelerated with successive over relaxation (SOR), Chebyshev and novel multiresolution-based techniques.

The minimization involved in the position update of each node amounts to a nonlinear system of equations (3 equations in 3D and 2 in 2D). We approximate the solution with Newton's method. The linearization of hyperelastic terms can have symmetric indefinite matrices. We develop an inexpensive yet effective technique for projecting any isotropic potential energy density Hessian to a symmetric positive definite counterpart as in [Teran et al. 2005]. However, unlike the definiteness projections in [Teran et al. 2005] and [Smith et al. 2019], it does not require the singular value decomposition of the deformation gradient. Furthermore, unlike the definiteness projection in [Teran et al. 2005], it does not require the solution of 3×3 or 2×2 symmetric eigensystems. As with PBD and other Gauss-Seidel approaches, a degree of freedom coloring technique is needed for efficient parallel performance. We provide a simple approach for this coloring and show that the position-based view tends to have far fewer colors than the constraint-based view in PBD and that this improves scalability and performance. Lastly, although our technique is designed for quasistatics, it is easily applicable to backward Euler discretizations of problems with inertia if we minimize the incremental potential [Bouaziz et al. 2014; Gast et al. 2015; Liu et al. 2013; Martin et al. 2011; Narain et al. 2016; Stern and Desbrun 2006] rather than the potential energy. We summarize our contributions as:

- A position-based, rather than constraint-based, nonlinear Gauss-Seidel technique for hyperelastic implicit time stepping.
- A hyperelastic energy density Hessian projection to efficiently guarantee definiteness of linearized equations that does not require a singular value decomposition or symmetric eigen solves.
- A node coloring technique that allows for efficient parallel performance of our Gauss-Seidel updates.
- A novel multiresolution acceleration technique for reducing iteration counts at high resolution.

2 PREVIOUS WORK

Baraff and Witkin first demonstrated that implicit time stepping with elasticity is essential for efficiency [Baraff and Witkin 1998]. Many approaches characterize implicit time stepping with hyperelasticity as a minimization of an incremental potential [Bouaziz et al. 2014; Gast et al. 2015; Liu et al. 2013; Martin et al. 2011; Narain

et al. 2016; Stern and Desbrun 2006]. This is often referred to as variational implicit Euler [Martin et al. 2011; Stern and Desbrun 2006] or optimization implicit Euler [Liu et al. 2013]. Quasistatic time stepping is an extreme case where inertia terms are ignored and only the strain energy is minimized [Kovalsky et al. 2016; Liu et al. 2008; Rabinovich et al. 2017; Sorkine and Alexa 2007; Teran et al. 2005]. 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. While Newton's method [Nocedal and Wright 2006] 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 [Gast et al. 2015]. There are many techniques that are less costly than Newton, but that can only reduce the system residual by a few orders of magnitude. However, many are satisfactory for visual accuracy. See discussion in Liu et al. [2013], Bouaziz et al. [2014] and Zhu et al. [2018].

Hyperelastic potentials must be rotationally invariant, non-negative and have global minima equal to zero at rotations. These considerations make the energy minimization non-convex with potentially non-unique solutions in quasistatic problems [Bonet and Wood 2008]. 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 [Li et al. 2019; Nocedal and Wright 2006; Smith et al. 2019; Teran et al. 2005; Zhu et al. 2018]. Many methods avoid the indefiniteness issue with the inclusion of auxiliary (or secondary) variables. Narain et al. [2016], Bouaziz et al. [2014], Liu et al. [2013] are recent examples of this, but similar approaches have been used in graphics since the local/global approach with ARAP by Sorkine et al. [2007]. Rabinovich et al. [2017] generalize this approach to a wider range of distortion energies.

Methods like the Alternating Direction Method of Multipliers (ADMM) [Boyd et al. 2011; Narain et al. 2016], the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm (L-BFGS) [Bertsekas 1997; Liu et al. 2017; Witemeyer et al. 2021; Zhu et al. 2018] and Sobolev preconditioned gradient descent (SGD) [Bouaziz et al. 2014; Liu et al. 2013; Neuberger 1985; Sorkine and Alexa 2007] require the inversion of a constant discrete elliptic operator (component-wise Laplacian) which can be pre-factored for efficiency. 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 [Bouaziz et al. 2014; Liu et al. 2013; Wang 2015; Zhu et al. 2018]. Zhu et al. [2018] 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. [2017] and Witemeyer et al. [2021] also use combinations of SGD and L-BFGS. Kovalsky et al. [2016] add Nesterov acceleration to SGD. Hecht et al. [2012] develop efficient updates for a pre-factored Hessian with corotated materials. Wang [2015] discusses the challenges of using direct solution/pre-factoring in the SGD-style approaches of Narain et al. [2016], Bouaziz et al. [2014] and Liu et al. [2013] and develops a Chebyshev acceleration technique as an alternative. In particular, they show that pre-factored discrete elliptic operators are memory-intensive (particularly at high-resolution) and limited since forward and backward substitutions do not parallelize. Moreover, [Wang 2015] show that simply replacing the direct solver with an iterative solver with reduced iteration count can lead to visually implausible or even unstable behaviors. However, Fratarcangeli et al. [2016] show that Gauss-Seidel iteration does not suffer from the same limitations in this context, although it does require degree of freedom coloring to facilitate parallel computation.

Tournier et al. [2015] 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 [2016] use a Chebyshev accelerated gradient descent approach for general hyperelasticity and FEM. [Lan et al. 2023] use a Gauss-Seidel approach similar to our own (although they focus on problems with inertia). Rather than iteratively adjusting the position of each node as we do, they adjust all nodes of each tetrahedron element (which requires the solution of a 12×12 system). The computational burden of this larger system is significant though since information from all adjacent elements must be included in the computation. This reduces the efficiency of coloring for parallelism and is more appropriate for offline/highly accurate computations.

3 EQUATIONS

We consider continuum mechanics conceptions of the governing physics where a flow map $\boldsymbol{\phi} : \Omega^0 \times [0,T] \rightarrow \mathbb{R}^d$, $d = 2$ or $d = 3$, describes the motion of the material. Here the time $t \in [0,T]$ location of the particle $\mathbf{X} \in \Omega^0 \subset \mathbb{R}^d$ is given by $\boldsymbol{\phi}(\mathbf{X},t) \in \Omega^t \subset \mathbb{R}^d$ where Ω^0 and Ω^t are the initial and time t configurations of material respectively. The flow map $\boldsymbol{\phi}$ obeys the partial differential equation associated with momentum balance

$$R^0 \frac{\partial^2 \boldsymbol{\phi}}{\partial t^2} = \nabla^{\mathbf{X}} \cdot \mathbf{P} + \mathbf{f}^{\text{ext}} \tag{1}$$

where R^0 is the initial mass density of the material, $\mathbf{P}$ is the first Piola-Kirchhoff stress and $\mathbf{f}^{\text{ext}}$ is external force density. This is also subject to boundary conditions

$$\boldsymbol{\phi}(\mathbf{X},t) = \mathbf{x}_D(\mathbf{X},t),\ \mathbf{X} \in \partial\Omega^0_D \tag{2}$$

$$\mathbf{P}(\mathbf{X},t)\hat{\mathbf{N}}(\mathbf{X},t) = \mathbf{T}(\mathbf{X},t),\ \mathbf{X} \in \partial\Omega^0_{\hat{\mathbf{N}}} \tag{3}$$

where $\hat{\mathbf{N}}$ is the outward-pointing normal to the initial boundary $\partial\Omega^0$ and $\partial\Omega^0$ is split into Dirichlet ($\partial\Omega^0_D$) and Neumann ($\partial\Omega^0_{\hat{\mathbf{N}}}$) regions where the deformation and applied traction respectively are specified. Here $\mathbf{T}$ denotes externally applied traction boundary conditions. For hyperelastic materials, the first Piola-Kirchhoff stress is related to a notion of potential energy density $\Psi : \mathbb{R}^{d\times d} \rightarrow \mathbb{R}$ as

$$\mathbf{P}(\mathbf{X},t) = \frac{\partial \Psi}{\partial \mathbf{F}}\left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}(\mathbf{X},t)\right),\ \text{PE}(\boldsymbol{\phi}(\cdot,t)) = \int_{\Omega^0} \Psi\left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}\right) d\mathbf{X} \tag{4}$$

where $\text{PE}(\boldsymbol{\phi}(\cdot,t))$ is the potential energy of the material when it is in the configuration defined by the flow map at time t. Note that we will typically use $\mathbf{F} = \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}$ to denote the spatial derivative of the flow map (or deformation gradient). We refer the reader to [Bonet and Wood 2008; Gonzalez and Stuart 2008] for more continuum mechanics detail.

In quasistatic problems, the inertial terms in the momentum balance (Equation (1)) can be neglected and the material motion is

defined by a sequence of equilibrium problems

$$\mathbf{0} = \nabla^{\mathbf{X}} \cdot \mathbf{P} + \mathbf{f}^{\text{ext}} \tag{5}$$

subject to the boundary conditions in Equations (2)-(3). This is equivalent to the minimization problems

$$\boldsymbol{\phi}(\cdot, t) = \underset{\Upsilon \in \mathcal{W}^t}{\text{argmin}} \ \text{PE}(\Upsilon) - \int_{\Omega_0} \mathbf{f}^{\text{ext}} \cdot \Upsilon d\mathbf{X} - \int_{\partial\Omega^0_{\hat{\mathbf{N}}}} \mathbf{T} \cdot \Upsilon ds(\mathbf{X}) \tag{6}$$

where $\mathcal{W}^t = \left\{ \Upsilon : \Omega_0 \rightarrow \mathbb{R}^d \,\middle|\, \Upsilon(\mathbf{X}) = \mathbf{x}_D(\mathbf{X}, t),\ \mathbf{X} \in \partial\Omega^0_D \right\}$. We note that even though the velfocity does not affect the quasistatic equilibrium equations in Equation (5), the time dependence in the boundary conditions gives rise to solutions $\boldsymbol{\phi}(\mathbf{X}, t)$ that change with respect to time.

3.1 Constitutive Models

We demonstrate our approach with a number of different hyperelastic potentials commonly used in computer graphics applications. The "corotated" or "warped stiffness" model [Chao et al. 2010; Etzmuss et al. 2003; Müller et al. 2002; Müller and Gross 2004; Schmedding and Teschner 2008] has been used for many years with a few variations. We use the version with the fix to the volume term developed by Stomakhin et al. [2012]

$$\Psi^{\text{cor}}(\mathbf{F}) = \mu|\mathbf{F} - \mathbf{R}(\mathbf{F})|_F^2 + \frac{\lambda}{2}(\det(\mathbf{F}) - 1)^2. \tag{7}$$

Here $\mathbf{F} = \mathbf{R}(\mathbf{F})\mathbf{S}(\mathbf{F})$ is the polar decomposition of $\mathbf{F}$. Neo-Hookean models [Bonet and Wood 2008] have also been used since they do not require polar decomposition and recently some of them have been shown to have favorable behavior with nearly incompressible materials [Smith et al. 2018].

$$\Psi^{\text{nh}}(\mathbf{F}) = \frac{1}{2}\mu|\mathbf{F}|_F^2 + \frac{\hat{\lambda}}{2}(\det(\mathbf{F}) - 1 - \frac{\mu}{\hat{\lambda}})^2. \tag{8}$$

Here $\hat{\lambda} = \mu + \lambda$. λ and μ are the Lamé parameters and are related to the Young's modulus (E) and Poisson's ratio (ν) as

$$\mu = \frac{E}{2(1+\nu)},\ \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}. \tag{9}$$

Note that we distinguish between the $\hat{\lambda}$ used in Macklin and Müeller [2021] and the Lamé parameter λ; we discuss the reason for this in more detail in Section 9. We also support the stable Neo-Hookean model proposed in [Smith et al. 2018]

$$\Psi^{\text{snh}}(\mathbf{F}) = \frac{1}{2}\mu(|\mathbf{F}|_F^2 - d) + \frac{1}{2}(\det(\mathbf{F}) - 1 - \frac{3\mu}{4\lambda})^2 - \frac{1}{2}\mu\log(1 + |\mathbf{F}|_F^2). \tag{10}$$

4 DISCRETIZATION

We use the FEM discretization of the quasistatic problem in Equation (5)

$$\mathbf{f}_i(\mathbf{x}^{n+1}) + \hat{\mathbf{f}}_i^{\text{ext}} = \mathbf{0},\ \mathbf{X}_i \notin \Omega^0_D \tag{11}$$

$$\mathbf{x}_i^{n+1} = \mathbf{x}_D(\mathbf{X}_i, t^{n+1}),\ \mathbf{X}_i \in \Omega^0_D. \tag{12}$$

Here the flow map is discretized as $\boldsymbol{\phi}(\mathbf{X}, t^{n+1}) = \sum_{j=0}^{N^V-1} \mathbf{x}_j^{n+1}\chi_j(\mathbf{X})$ where the $\chi_j(\mathbf{X})$ are piecewise linear interpolating functions defined over a tetrahedron mesh ($d = 3$) or triangle mesh ($d = 2$), and $\mathbf{x}_j^{n+1} \in \mathbb{R}^d$, $0 \le j < N^V$ are the locations of the vertices of the mesh at time t^{n+1}. Note that we use $\mathbf{x}^{n+1} \in \mathbb{R}^{dN^V}$ to denote the vector of all vertex locations and $x_{i\beta}^{n+1}$ to denote the $0 \le \beta < d$ components of the position of vertex i in the mesh. The forces are given as

$$\mathbf{f}_i(\mathbf{y}) = -\frac{\partial \hat{\text{PE}}}{\partial \mathbf{y}_i}(\mathbf{y}) \tag{13}$$

$$\hat{\text{PE}}(\mathbf{y}) = \hat{\text{PE}}^{\Psi}(\mathbf{y}) + \hat{\text{PE}}^{\text{wc}}(\mathbf{y}) \tag{14}$$

$$\hat{\text{PE}}^{\Psi}(\mathbf{y}) = \sum_{e=0}^{N^E-1} \Psi(\sum_{j=0}^{N^V-1} \mathbf{y}_j \frac{\partial \chi_j}{\partial \mathbf{X}}(\mathbf{X}^e))V_e^0 \tag{15}$$

$$\hat{\mathbf{f}}_i^{\text{ext}} = \int_{\Omega^0} \mathbf{f}^{\text{ext}}\chi_i d\mathbf{X} + \int_{\partial\Omega^0_{\hat{\mathbf{N}}}} \mathbf{T}\chi_i ds(\mathbf{X}) \tag{16}$$

where $\hat{\text{PE}}^{\Psi} : \mathbb{R}^{dN^V} \rightarrow \mathbb{R}$ is the discretization of the potential energy, $\sum_{j=0}^{N^V-1} \mathbf{y}_j \frac{\partial \chi_j}{\partial \mathbf{X}}(\mathbf{X}^e)$ is the deformation gradient induced by nodal positions $\mathbf{y} \in \mathbb{R}^{dN^V}$ in tetrahedron ($d = 3$) or triangle ($d = 2$) element e with $0 \le e < N^E$, $\frac{\partial \chi_i}{\partial \mathbf{X}}(\mathbf{X}^e)$ is the derivative of the interpolating function in element e (which is constant since we use piecewise linear interpolation) and V_e^0 is the measure of the element. We refer the reader to [Bonet and Wood 2008; Sifakis and Barbic 2012] for more detail on the FEM derivation of potential energy terms in a hyperelastic formulation. Also, note that we add another term to the discrete potential energy $\hat{\text{PE}}^{\text{wc}} : \mathbb{R}^{dN^V} \rightarrow \mathbb{R}$ in Equation (14) to account for self-collisions and similar weak constraints (see Section 4.1). Similar to the non-discrete case, the constrained minimization problem

$$\mathbf{x}^{n+1} = \underset{\mathbf{y} \in \mathcal{W}_{\Delta x}^{n+1}}{\text{argmin}} \ \hat{\text{PE}}(\mathbf{y}) - \mathbf{y} \cdot \hat{\mathbf{f}}^{\text{ext}} \tag{17}$$

where $\mathcal{W}_{\Delta x}^{n+1} = \left\{ \mathbf{y} \in \mathbb{R}^{dN^V} \,\middle|\, \mathbf{y}_i = \mathbf{x}_D(\mathbf{X}_i, t^{n+1}),\ \mathbf{X}_i \in \partial\Omega^0_D \right\}$ is equivalent to Equations (11)-(12).

4.1 Weak Constraints

We support weak constraints for self-collision and other similar purposes (as in [McAdams et al. 2011]). These are terms added to the potential energy in the form

$$\hat{\text{PE}}^{\text{wc}}(\mathbf{y}) = \frac{1}{2}\sum_{c=0}^{N^{\text{wc}}-1} \mathbf{C}_c(\mathbf{y})^T \mathbf{K}_c \mathbf{C}_c(\mathbf{y}) \tag{18}$$

$$\mathbf{C}_c(\mathbf{y}) = \sum_{j=0}^{N^V-1} w_{0j}^c \mathbf{y}_{0j} - w_{1j}^c \mathbf{y}_{1j}. \tag{19}$$

Here the w_{0j}^c, w_{1j}^c are interpolation weights that sum to one and are non-negative. This creates constraints between the interpolated points $\sum_{j=0}^{N^V-1} w_{0j}^c \mathbf{y}_{0j}$ and $\sum_{j=0}^{N^V-1} w_{1j}^c \mathbf{y}_{1j}$. The stiffness of the constraint is represented in the matrix $\mathbf{K}_c$. This can allow for anisotropic responses where $\mathbf{K}_c = k_n \mathbf{n}\mathbf{n}^T + k_\tau \left(\boldsymbol{\tau}_0 \boldsymbol{\tau}_0^T + \boldsymbol{\tau}_1 \boldsymbol{\tau}_1^T \right)$. Here $\mathbf{n}^T \boldsymbol{\tau}_i = 0$, $i = 0, 1$ and k_n is the stiffness in the $\mathbf{n}$ direction while k_τ is the stiffness in response to the motion in the plane normal to $\mathbf{n}$. $\boldsymbol{\tau}_0^T \boldsymbol{\tau}_1 = 0$ and $||\boldsymbol{\tau}_i|| = 1, i = 0, 1$. In the case of an isotropic constraint ($k_c = k_n = k_\tau$), we use the scalar k_c in place of $\mathbf{K}_c$ since

$\mathbf{K}_c = k_c\mathbf{I}$ is diagonal. We note that, in most of our examples, the anisotropic model is used for collision constraints where $\mathbf{n}$ is the collision constraint direction.

5 GAUSS-SEIDEL NOTATION

Our approach, PBD and XPBD all use nonlinear Gauss-Seidel to iteratively improve an approximation to the solution $\mathbf{x}^{n+1} \in \mathbb{R}^{dN^V}$ of Equation (11) (or equivalently, Equation (17)). Here we introduce detailed notation to help clarify the specific details of our method as well as its convergence behaviors. We refer to one Gauss-Seidel iteration as the process of updating all vertices once and use l to denote the iteration count as $\mathbf{x}^{n+1,l} \approx \mathbf{x}^{n+1}$. During the course of one Gauss-Seidel iteration, individual vertex degrees of freedom in the approximate solution will be updated in sub-iterates (indexed by k) which we denote as $\mathbf{x}^{n+1,l}_{(k)} \in \mathbb{R}^{dN^V}$ with $0 \leq k < N^{GS}$. For example, $\mathbf{x}^{n+1,l}_{(0)} = \mathbf{x}^{n+1,l}$ and $\mathbf{x}^{n+1,l}_{(N^V - N^D - 1)} = \mathbf{x}^{n+1,l+1}$ for PBNG. To further clarify, with PBD/XPBD in the k^{th} sub-iterate, the nodes in the k^{th} constraint will be projected/solved for and so N^{GS} will be equal to the total number of constraints. In our position-based approach, in the k^{th} sub-iterate, only the d components of a single node i_k will be updated. It is important to introduce this notation, since unlike with Jacobi-based approaches, the update of the k^{th} sub-iterate will depend on the contents of the $k-1^{\text{th}}$ sub-iterate.

6 POSITION-BASED DYNAMICS: CONSTRAINT-BASED NONLINEAR GAUSS-SEIDEL

Macklin et al. [2016] show that PBD [Müller et al. 2007] can be seen to be the extreme case of a numerical method for the approximation of the backward Euler temporal discretization of the FEM spatial discretization of Equation (1)

$$\sum_{j=0}^{N^V-1} m_{ij}\left(\frac{\mathbf{x}_j^{n+1} - 2\mathbf{x}_j^n + \mathbf{x}_j^{n-1}}{\Delta t^2}\right) = \mathbf{f}_i(\mathbf{x}^{n+1}) + \mathbf{f}_i^{\text{ext}},\ \mathbf{X}_i \notin \Omega_D^0. \quad (20)$$

Here $m_{ii} = \int_{\Omega^0} R^0 \chi_i d\mathbf{X}$ and $m_{ij} = 0,\ j \neq i$ are entries in the mass matrix. However, Macklin et al. [2016] require that the discrete potential energy in Equation (15) is of the form

$$\hat{PE}^{\Psi}(\mathbf{y}) = \sum_{c=0}^{2N^E-1} \frac{1}{2\alpha_c} C_c^2(\mathbf{y}),\ \mathbf{y} \in \mathbb{R}^{dN^E}. \quad (21)$$

To demonstrate the connection between Equation (20) and PBD, Macklin et al. [2016] develop XPBD. It is based on the total Lagrange multiplier formulation

$$\sum_{j=0}^{N^V-1} m_{ij}\left(\mathbf{x}_j^{n+1} - \hat{\mathbf{x}}_j\right) - \sum_{c=0}^{P-1} \frac{\partial C_c}{\partial \mathbf{x}_i}(\mathbf{x}^{n+1})\lambda_c^{n+1} = 0,\ \mathbf{X}_i \notin \Omega_D^0 \quad (22)$$

$$C_c(\mathbf{x}^{n+1}) + \frac{\alpha_c}{\Delta t^2}\lambda_c^{n+1} = 0,\ 0 \leq c < P \quad (23)$$

where $\hat{\mathbf{x}}_j = 2\mathbf{x}_j^n - \mathbf{x}_j^{n-1} - \frac{\Delta t^2}{m_{jj}}\mathbf{f}_j^{\text{ext}}$ and $\boldsymbol{\lambda}^{n+1} \in \mathbb{R}^P$ is introduced as an additional unknown. The $\mathbf{x}^{n+1} \in \mathbb{R}^{dN^V}$ in Equations (22)-(23) is the same in the solution to Equation (20). P is the number of constraints. Macklin et al. [Macklin et al. 2016] use a per-constraint Gauss-Seidel update of Equations (22)-(23)

$$\mathbf{x}^{n+1,l}_{i(k+1)} = \mathbf{x}^{n+1,l}_{i(k)} + \Delta\mathbf{x}^{n+1,l}_{i(k+1)},\ \mathbf{X}_i \notin \Omega_D^0 \quad (24)$$

$$\Delta\mathbf{x}^{n+1,l}_{i(k+1)} = \frac{\Delta\lambda^{n+1,l}_{(k+1)c_k}}{m_{ii}}\frac{\partial C_{c_k}}{\partial \mathbf{x}_i}(\mathbf{x}^{n+1,l}_{(k)}) \quad (25)$$

$$\Delta\lambda^{n+1,l}_{(k+1)c_k} = \frac{-C_{c_k}(\mathbf{x}^{n+1,l}_{(k)}) + \frac{\alpha_{c_k}}{\Delta t^2}C_{c_k}(\mathbf{x}^{n+1,l}_{(k)})}{\sum_{j=0}^{N^V-1}\frac{1}{m_{jj}}\sum_{\beta=0}^{d-1}\left(\frac{\partial C_{c_k}}{\partial x_{j\beta}}(\mathbf{x}^{n+1,l}_{(k)})\right)^2 + \frac{\alpha_{c_k}}{\Delta t^2}}. \quad (26)$$

Here the $k+1^{\text{th}}$ sub-iterate in iteration l is generated by solving for the change in a single Lagrange multiplier $\Delta\lambda^{n+1,l}_{(k+1)c_k}$ associated with a constraint c_k that varies from sub-iteration to sub-iteration. However, as pointed out in [Chen et al. 2023], this Gauss-Seidel procedure does not converge to a solution of Equation (20). [Chen et al. 2023] isolates the root cause of this as the omission of the residual of Equation (22) in the update of the Lagrange multiplier in Equation (26) and moreover that inclusion of the residual in the update leads to unstable behavior. We demonstrate this behavior and contrast with our approach in Figure 3.

6.1 Quasistatics

As noted by Macklin et al. [2016], the XPBD update in Equations (24)-(26) is the same as in the original PBD [Müller et al. 2007] in the limit $\alpha_c \to 0$. By choosing a stiffness inversely proportionate to a parameter $s \geq 0$ and examining the limiting behavior of the equations being approximated, we see that the original PBD approach generates an approximation to the quasistatic problem (Equations (5)), albeit with the external forcing terms omitted. More precisely, define $\boldsymbol{\phi}_s$ to be a solution of the problem

$$sR^0\frac{\partial^2\boldsymbol{\phi}_s}{\partial t^2} = \nabla^{\mathbf{X}}\cdot\mathbf{P} + s\mathbf{f}^{\text{ext}}. \quad (27)$$

subject to the same boundary conditions in Equations (2)-(3). This is equivalent to scaling the α_c that would appear in Equation (1) (through $\mathbf{P}$) by s. The α_c are inversely proportionate to the Lamé parameters, so as $s \to 0$, the material stiffness increases. Since the inertia and external force terms in Equation (27) vanish as $s \to 0$, it is clear then that the original PBD formulation generates an approximation to the solution of a quasistatic problem with the external forcing $\mathbf{f}^{\text{ext}}$ omitted. Note that PBD does include the external forcing term in its initial guess $\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \Delta t\mathbf{v}_i^n + \frac{\Delta t^2}{m_{ii}}\mathbf{f}_i^{\text{ext}}$. However, the effect of the initial guess vanishes as the iteration count is increased. We demonstrate this in Section 9.3 of the paper. Also, note that this is not the case in the XPBD formulation where $\alpha_c > 0$, as it is the inverse stiffness term.

Unfortunately, XPBD cannot be trivially modified to run quasistatic problems. For example, omitting the mass terms on the left-hand side of Equation (22) makes the Gauss-Seidel update in Equations (24)-(26) impossible since there would be a division by zero. The simplest fix for quasistatic problems with XPBD is to run to steady state using a pseudo-time iteration as in [Chentanez et al. 2020]. This prevents the need for scaling the α_c which inherently removes the external forcing terms and does not introduce a divide by zero in Equation (25). However, this is very costly since each quasistatic time step is essentially the cost of an entire XPBD simulation.

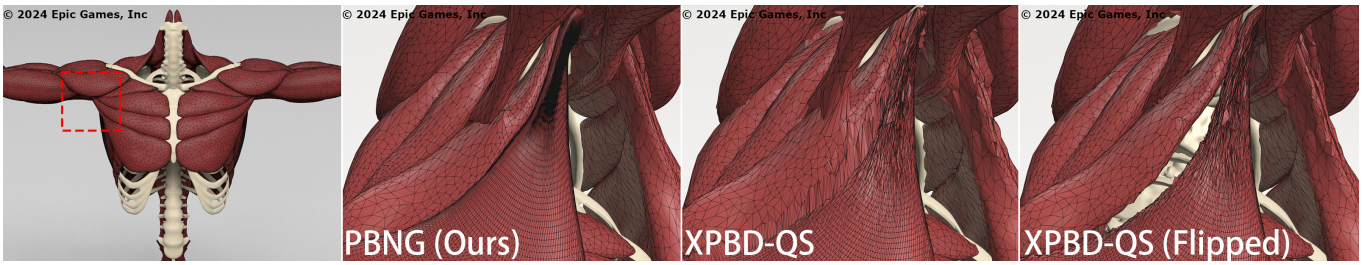

Fig. 2. **PBNG vs XPBD**. Muscle simulation demonstrates iteration-order-dependent behavior with XPBD and quasistatics. A zoom-in view under the right armpit region is provided. Each method is run for 130 iterations. PBNG converges to the desired solution, binding the muscles closely together. XPBD-QS and XPBD-QS (Flipped) fail to converge, leaving either artifacts or gaps between the muscles. Details on XPBD-QS and XPBD-QS (Flipped) can be found in Section 11.

We refer to this technique as XPBD-QS (see 11.3 for example). In addition to the excessive cost of this approach, we also observe severe iteration-order dependent behavior of XPBD-QS in the presence of spatially varying constraints and where constraints of different types affect the same vertices (see Figure 2). We believe the omission of the primary residual noted by Chen et al. [2023] is the cause of this iteration-dependent behavior. Intuitively, the Gauss-Seidel update would have information about adjacent constraints if it could be added stably.

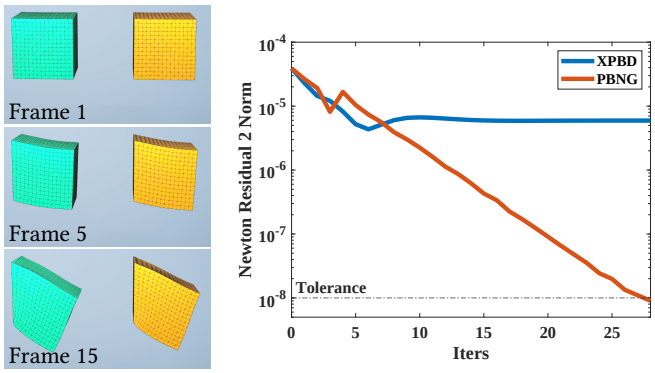

Fig. 3. **Left**. Clamped blocks under gravity. The green block is XPBD, and the yellow one is PBNG. **Right**. PBNG is able to reduce the Newton residual to the tolerance, whereas XPBD's residual stagnates.

7 POSITION-BASED NONLINEAR GAUSS-SEIDEL

To fix the issues with PBD/XPBD and quasistatics, we abandon the Lagrange-multiplier formulation and approximate the solution of Equation (11) using position-centric, rather than constraint-centric nonlinear Gauss-Seidel. This update takes into account each constraint that the position participates in. Visual intuition for this is illustrated in Figure 6(a). More specifically, in the k^{th} sub-iterate of iteration l, we modify a single node i_k with $\mathbf{X}_{i_k} \notin \partial\Omega_D^0$ as

$$\begin{aligned} \mathbf{x}^{n+1,l}_{(k+1)i_k} &= \mathbf{x}^{n+1,l}_{(k)i_k} + \Delta\mathbf{x}^{n+1,l}_{(k+1)i_k} \\ \Delta\mathbf{x}^{n+1,l}_{(k+1)i_k} &= \underset{\Delta\mathbf{y}\in\mathbb{R}^d}{\text{argmin}} \; \hat{PE}(\mathbf{x}^{n+1,l}_{(k)} + \tilde{\mathbf{C}}^{i_k}\Delta\mathbf{y}) - \Delta\mathbf{y}\cdot\hat{\mathbf{f}}^{\text{ext}}_{i_k}. \end{aligned} \tag{28}$$

Here $\tilde{\mathbf{C}}^{i_k} \in \mathbb{R}^{dN^V\times d}$ is a selection matrix that isolates the degrees of freedom on the node i_k and has entries $\tilde{C}^{i_k}_{j\alpha\beta} = \delta_{ji_k}\delta_{\alpha\beta}$. We solve this minimization by setting the gradient to zero

$$\mathbf{0} = \mathbf{f}_{i_k}(\mathbf{x}^{n+1,l}_{(k)} + \tilde{\mathbf{C}}^{i_k}\Delta\mathbf{x}^{n+1,l}_{(k+1)i_k}) + \hat{\mathbf{f}}^{\text{ext}}_{i_k}. \tag{29}$$

We use a single step of a modified Newton's method to approximate the solution of Equation (29) for $\Delta\mathbf{x}^{n+1,l}_{(k+1)i_k} \in \mathbb{R}^d$. We use $\Delta\mathbf{x}^{n+1,l}_{(k+1)i_k} = \mathbf{0}$ as the initial guess. We found that using more than one iteration did not significantly improve robustness or convergence. Our update is of the form

$$\Delta\mathbf{x}^{n+1,l}_{(k+1)i_k} = \left(\mathbf{A}^{n+1,l}_{(k+1)i_k}\right)^{-1}\left(\mathbf{f}_{i_k}(\mathbf{x}^{n+1,l}_{(k)}) + \hat{\mathbf{f}}^{\text{ext}}_{i_k}\right). \tag{30}$$

Here $\mathbf{A}^{n+1,l}_{(k+1)i_k} \approx -\frac{\partial\mathbf{f}_{i_k}}{\partial\mathbf{y}_{i_k}}(\mathbf{x}^{n+1,l}_{(k)}) \in \mathbb{R}^{d\times d}$ is an approximation to the potential energy Hessian/negative force gradient.

7.1 Modified Hessian

We choose the modified energy Hessian $\mathbf{A}^{n+1,l}_{(k+1)i_k}$ to minimize its computational cost. The true Hessian $\frac{\partial\mathbf{f}_{i_k}}{\partial\mathbf{y}_{i_k}} \in \mathbb{R}^{d\times d}$ has entries

$$\begin{aligned} \frac{\partial f_{i_k\alpha}}{\partial y_{i_k\beta}}(\mathbf{y}) = &-\sum_{e=0}^{N^E-1}\sum_{\delta,\gamma=0}^{d-1} C^e_{\alpha\gamma\beta\delta}(\mathbf{y})\frac{\partial N^e_{i_k}}{\partial X_\gamma}\frac{\partial N^e_{i_k}}{\partial X_\delta}V^0_e - \\ &\sum_{c=0}^{N^{\text{wc}}-1}\left(w^c_{0i_k} - w^c_{1i_k}\right)^2 K_{c\alpha\beta},\ 0\le\alpha,\beta<d \end{aligned} \tag{31}$$

where $C^e_{\alpha\gamma\beta\delta}(\mathbf{y}) = \frac{\partial^2\Psi}{\partial F_{\beta\delta}\partial F_{\alpha\gamma}}(\sum_{j=0}^{N^V-1}\mathbf{y}_j\frac{\partial N^e_j}{\partial\mathbf{X}})$ is the Hessian of the potential energy density evaluated at the deformation gradient in element e and $K_{c\alpha\beta}$ is the stiffness tensor associated with weak-constraints.

The primary cost in Equation (31) is the evaluation of the Hessian of the energy density $C^e_{\alpha\gamma\beta\delta}(\mathbf{y})$ which is a symmetric fourth order tensor with $d^2\times d^2$ entries. Furthermore, this tensor can be indefinite, which would complicate the convergence of the Newton procedure. We use a definiteness projection as in [Teran et al. 2005] and [Smith et al. 2019]. However, we use a very simple symmetric positive definite approximation that (unlike [Smith et al. 2019; Teran et al. 2005]) does not require the singular value decomposition of the

element deformation gradient $\sum_{j=0}^{N^V-1} \mathbf{y}_j \frac{\partial N_j^e}{\partial \mathbf{X}}$. Also note that Teran et al. [2005] also require the solution of a 3×3 and three 2×2 symmetric eigenvalue problems; our approach does not require this. Our simple approximation is

$$\tilde{C}^e_{\alpha\gamma\beta\delta}(\mathbf{y}) = 2\mu\delta_{\alpha\beta}\delta_{\gamma\delta} + \lambda J(F^e)^{-1}_{\alpha\gamma}(\mathbf{y}) J(F^e)^{-1}_{\beta\delta}(\mathbf{y}). \tag{32}$$

Here $J\mathbf{F}^e(\mathbf{y}) = \det(\mathbf{F}^e(\mathbf{y})(\mathbf{F}^e)^{-T}(\mathbf{y})$ is the cofactor matrix of the element deformation gradient $\mathbf{F}^e(\mathbf{y}) = \sum_{j=0}^{N^V-1} \mathbf{y}_j \frac{\partial N_j^e}{\partial \mathbf{X}}$. We note that the cofactor matrix is defined for all deformation gradients $\mathbf{F}^e$, singular, inverted (negative determinant) or otherwise. This is essential for robustness to large deformation. We note that this approximation works for any isotropic potential energy density Ψ where μ and λ are the associated Lamé parameters computed from Young's modulus E. We discuss the motivation for this simplification in [Anonymous 2023], but note here that it is clearly positive definite since it is a scaled version of the identity with a rank-one update from the cofactor matrix. With this convention, our symmetric positive definite modified nodal Hessian is of the form

$$A^{n+1}_{(k+1)i_k\alpha\beta} = \sum_{e=0}^{N^E-1} \sum_{\delta,\gamma=0}^{d-1} \tilde{C}^e_{\alpha\gamma\beta\delta}(\mathbf{x}^{n+1,l}_{(k)}) \frac{\partial N^e_{i_k}}{\partial X_\gamma} \frac{\partial N^e_{i_k}}{\partial X_\delta} V^0_e + \tag{33}$$

$$\sum_{c=0}^{N^{\text{wc}}-1} \left(w^c_{0i_k} - w^c_{1i_k}\right)^2 K_{c\alpha\beta},\ 0 \le \alpha, \beta < d. \tag{34}$$

7.2 Collision against kinematic bodies

We add support for hard collision constraints against kinematic geometry (collision bodies that do not deform). At the beginning of each time step, each vertex $\mathbf{x}_i$ detects its closest point $\bar{\mathbf{x}}_i$ on the kinematic body. We use $\mathbf{n}_i$ to denote the unit outward normal to the collision body at the closest point. $\mathbf{x}_i$ is then classified as penetrating if $(\mathbf{x}_i - \bar{\mathbf{x}}_i) \cdot \mathbf{n}_i < 0$. For each penetrating $\mathbf{x}_i$, we project it to $\bar{\mathbf{x}}_i$ before the simulation. Then for each PBNG iteration, we check if $\Delta\mathbf{x}^{n,l}_{(k)i} \cdot \mathbf{n}_i < 0$. If so, we project $\Delta\mathbf{x}^{n,l}_{(k)i}$ to $\Delta\bar{\mathbf{x}}^{n,l}_{(k)i} = (\mathbf{I} - \mathbf{n}_i\mathbf{n}_i^T)\Delta\mathbf{x}^{n,l}_{(k)i}$ to allow for sliding tangential to the constraint surface.

7.3 SOR and Chebyshev Iteration

PBNG is remarkably stable and gives visually plausible results when the computational budget is limited, but it is also capable of producing numerically accurate results as the budget is increased. However, as with most Gauss-Seidel approaches the convergence rate of PBNG may decrease with iteration count (see Figure 12 for details). We investigated two simple acceleration techniques to help mitigate this: the Chebyshev semi-iterative method (as in [Wang 2015]) and SOR (as in [Fratarcangeli et al. 2016]). The Chebyshev method uses the update $\mathbf{x}^{n+1,l+1} = \omega_{l+1}(\gamma(\mathbf{x}^{n+1,l+1}_{\text{PBNG}} - \mathbf{x}^{n+1,l}) + \mathbf{x}^{n+1,l} - \mathbf{x}^{n+1,l-1}) + \mathbf{x}^{n+1,l-1}$. where $\mathbf{x}^{n+1,l+1}$ denotes the accelerated update and $\mathbf{x}^{n+1,l+1}_{\text{PBNG}}$ denotes the standard PBNG update. Here $\omega_{l+1} = \frac{4}{4-\rho^2\omega_l}$ for $l > 2$, $\frac{2}{2-\rho^2}$ for $l = 2$ and 1 for $l < 2$. γ is an under-relaxation parameter that stabilizes the algorithm. For our examples, we set $\rho = .95$. PBNG is very stable, and this allows for the use of over-relaxation as well. We set $\gamma = 1.7$. The SOR method uses a similar, but simpler update $\mathbf{x}^{n+1,l+1} = \omega(\mathbf{x}^{n+1,l+1}_{\text{PBNG}} - \mathbf{x}^{n+1,l-1}) + \mathbf{x}^{n+1,l-1}$. We use $\omega = 1.7$ for this under-relaxation parameter. Chebyshev and SOR behave similarly in terms of residual reduction and visual appearance (see Figure 12).

8 CLOTH SIMULATION

Our method can also naturally handle cloth simulation by adding a surface mesh contribution $\tilde{\text{PE}}(\mathbf{y})$ directly to the potential energy in Equation (17). We use the sum of a membrane hyperelastic potential and a bending term:

$$\tilde{\text{PE}}(\mathbf{y}) = \sum_{\hat{t}} \Psi^{\text{cm}}(\mathbf{F}^{\text{mem}}_{\hat{t}}(\mathbf{y})) A_{\hat{t}} + \frac{1}{2} k_b \sum_e \theta_e(\mathbf{y})^2. \tag{35}$$

The membrane term is a simple generalization of the fixed corotated model [Stomakhin et al. 2012] to the case of surfaces

$$\Psi^{\text{cm}}(\mathbf{F}^{\text{mem}}) = \mu^{\text{cm}} |\mathbf{F}^{\text{mem}} - \mathbf{R}(\mathbf{F}^{\text{mem}})|^2_F + \frac{\lambda^{\text{cm}}}{2} (J(\mathbf{F}^{\text{mem}}) - 1)^2. \tag{36}$$

Here μ^{cm} and λ^{cm} are derived from Young's modulus E^{cm} in a similar fashion as μ and λ in Section 7. $\mathbf{F}^{\text{mem}}_{\hat{t}}(\mathbf{y}) = \sum_i \mathbf{y}_i \frac{\partial \hat{\chi}_i}{\partial \mathbf{X}}(\mathbf{X}_{\hat{t}}) \in \mathbb{R}^{3\times 2}$ is the deformation gradient computed over the triangle $\hat{t}$ where $\hat{\chi}_i$ are piecewise linear interpolating functions over the triangles. $A_{\hat{t}}$ is the reference area of the triangle $\hat{t}$ and $J(\mathbf{F}^{\text{mem}}) = \sqrt{\det(\mathbf{F}^{\text{mem}T}\mathbf{F}^{\text{mem}})}$ denotes the multiplicative change in the triangle area under motion defined by $\mathbf{y}$. The term $\mathbf{R}(\mathbf{F}^{\text{mem}})$ in Equation (36) is the rotational part of the polar decomposition of $\mathbf{F}^{\text{mem}} = \mathbf{R}(\mathbf{F}^{\text{mem}})\mathbf{S}(\mathbf{F}^{\text{mem}})$ with the convention that $\mathbf{R}(\mathbf{F}^{\text{mem}}) \in \mathbb{R}^{3\times 2}$ has orthogonal columns and $\mathbf{S}(\mathbf{F}^{\text{mem}}) \in \mathbb{R}^{2\times 2}$ is symmetric.

For bending resistance in Equation (35), we adopt a similar approach to Baraff and Witkin [1998]. For each edge e with vertices $\mathbf{y}^0_e, \mathbf{y}^1_e$ that is incident to two triangles with unit normals $\mathbf{n}^1_e(\mathbf{y}), \mathbf{n}^2_e(\mathbf{y})$, we define $\theta_e(\mathbf{y}) \in [0, \pi)$ as the bending angle where $\theta_e(\mathbf{y}) = \text{atan}(\frac{(\mathbf{n}^1_e(\mathbf{y}) \times \mathbf{n}^2_e(\mathbf{y})) \cdot (\mathbf{y}^1_e - \mathbf{y}^0_e)}{\mathbf{n}^1_e(\mathbf{y}) \cdot \mathbf{n}^2_e(\mathbf{y})})$. k_b is the bending stiffness parameter. Note that as in [Baraff and Witkin 1998], we do not use an area weighting on the bending term.

8.1 Modified Hessian

Similar to Section 7.1, we modify the Hessians of the above models to ensure positive semi-definiteness. We make the simple approximation $\frac{\partial^2 \Psi^{\text{cm}}}{\partial F^{\text{mem}}_{\alpha\gamma} \partial F^{\text{mem}}_{\beta\delta}}(\mathbf{y}) \approx 2\mu^{\text{cm}}\delta_{\alpha\beta}\delta_{\gamma\delta} + \frac{\lambda^{\text{cm}}}{4} J^2 L^{\text{mem}}_{\alpha\gamma} L^{\text{mem}}_{\beta\delta}$ where $\mathbf{L}^{\text{mem}} = \mathbf{F}^{\text{mem}}((\mathbf{F}^{\text{mem}})^T\mathbf{F}^{\text{mem}})^{-T} + \mathbf{F}^{\text{mem}}((\mathbf{F}^{\text{mem}})^T\mathbf{F}^{\text{mem}})^{-1}$. For the bending model, we use the rank one approximation

$$\frac{1}{2} \frac{\partial^2 \theta_e^2}{\partial \mathbf{x}^2} \approx \frac{\partial \theta_e}{\partial \mathbf{x}} \otimes \frac{\partial \theta_e}{\partial \mathbf{x}}. \tag{37}$$

The bending potential in Equation (35) is quadratic in θ_e so a rank-one term consisting of the outer-product in Equation (37) is a simple PSD Hessian approximation. See the supplemental technical document for more details [Anonymous 2023].

8.2 Multiresolution Acceleration

When a solid and a piece of cloth have the same particle count, the max topological distance between the particles in the cloth is bigger than that in the solid because a cloth is essentially a 2D object. Since Gauss-Seidel only does local updates, it needs to propagate information further for cloth to converge. In practice, cloth will look

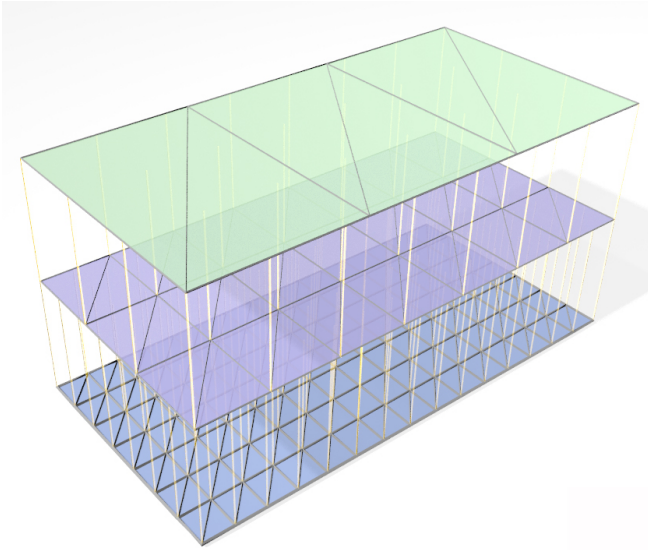

Fig. 4. **MPBNG Illustration**. We visualize the cloth multi-resolution hierarchy. Straight lines illustrate constraints between vertices on the finer level to their targets on the coarser level.

overly stretchy when propagation has not proceeded sufficiently (see Section 11.5.). This slowed convergence with increased resolution is the motivation for techniques like multigrid [Brandt 1977; Wang et al. 2018]. We develop a multiresolution techniques along these lines: MPBNG (or multilayer PBNG). For a given piece of cloth χ_0, we remesh it to create a nested mesh hierarchy $\chi_0 \supset \chi_1 \supset \chi_2 \supset \chi_n$. Usually the number of vertices is reduced by approximately a factor of 4. These layers equally divide the stiffness and mass density of the original material as in a multi-species continua model [Atkin and Craine 1976]. We then bind each two successive layers using weak constraints (Equation (15)) and define a multi-layer iteration as follows: 4-6 PBNG iterations are first applied on the coarsest level. Then all particles on $\cup_{i\neq n}\chi_i$ are iterated over in a standard PBNG manner. This is done in parallel, utilizing coloring that takes into account all layers and the constraints between them. After the multi-layer iterations have been applied, we do a final pass of 6-8 PBNG iterations on the finest layer. The weak constraints are visualized in Figure 4. Note that we found this three-pass approach to be more effective than a standard multigrid V-cycle in practice.

After each multiresolution iteration l, we reduce the stiffness of the weak constraints and the mass on the coarse layers. We define a scaling factor $\kappa_l = \sqrt{1 - \frac{l^2}{r_{\text{des}}^2}}$ where r_{des} is a user-specified radius of descent. We choose $r_{\text{des}} = k_{\max} + r_{\text{pad}}$ where $k_{\max}$ is the maximal number of iterations and $r_{\text{pad}} \in [.05, 2]$. In practice, r_{pad} does require much tuning and $r_{\text{pad}} = .05$ typically suffices. This factor is multiplied to the mass of the nodes on the coarse layers (i.e. all layers except χ_0) and k_c of all weak constraints. As l becomes larger, κ_l decreases to a small number, which means the problems converges to the original one. The same approach can be applied to solid simulation as well, however we observed that MPBNG achieves less residual reduction than PBNG in this case (with a fixed computational budget).

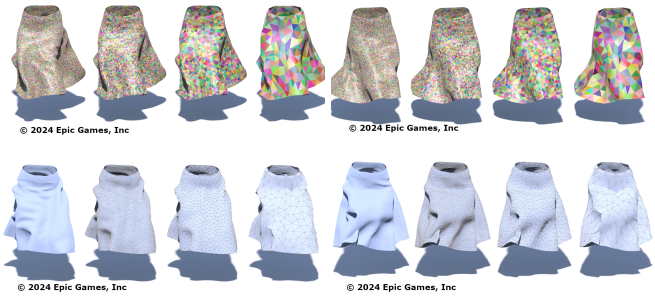

Fig. 5. **Multiresolution Dress**. We illustrate the multiresolution meshes used with our MPBNG approach in a representative clothing simulation.

9 LAMÉ COEFFICIENTS

The parameters of an isotropic constitutive model are often based on Lamé coefficients μ and λ which are themselves set from Young's modulus E and Poisson's ratio ν according to Equation (9). This relationship is based on the assumption of linear dependence of stress on strain, or quadratic potential energy density

$$\Psi^{\text{le}}(\mathbf{F}) = \mu \text{tr}(\boldsymbol{\epsilon}^2(\mathbf{F})) + \frac{\lambda}{2}\text{tr}(\boldsymbol{\epsilon}(\mathbf{F}))^2 \tag{38}$$

$$\boldsymbol{\epsilon} = \frac{1}{2}(\mathbf{F} + \mathbf{F}^T) - \mathbf{I}. \tag{39}$$

Furthermore, Equation (9) is derived from the model in Equation (38) by holding one end of a cuboidal domain fixed and applying a displacement at its opposite end. The remaining faces of the domain are assumed to be traction-free. Young's modulus is the scaling in a linear relationship between the traction exerted by the material in resistance to the displacement. The Poisson's ratio correlates with the degree of volume preservation via deformation in the directions orthogonal to the applied displacement.

The use of Lamé coefficients with nonlinear models is not directly analogous since the relation between displacement and traction is not a linear scaling in the cuboid example. When using Lamé coefficients with nonlinear problems, the cuboid derivation should hold if the model were linearized around $\mathbf{F} = \mathbf{I}$. All isotropic hyperelastic constitutive models can be written in terms of the isotropic invariants $I_\alpha : \mathbb{R}^{d\times d} \to \mathbb{R}$, $0 \le \alpha < d$

$$I_0(\mathbf{F}) = \text{tr}(\mathbf{F}^T\mathbf{F}),\ I_1(\mathbf{F}) = \text{tr}((\mathbf{F}^T\mathbf{F})^2),\ I_2(\mathbf{F}) = \det(\mathbf{F}) \tag{40}$$

$$\Psi(\mathbf{F}) = \hat{\Psi}(I_0(\mathbf{F}), I_1(\mathbf{F}), I_2(\mathbf{F})). \tag{41}$$

See [Gonzalez and Stuart 2008] for more detailed derivation. Note, when $d = 2$, $I_1(\mathbf{F}) = \text{tr}((\mathbf{F}^T\mathbf{F})^2)$ is not used. With this convention, the Hessian of the potential energy density is of the form

$$\frac{\partial^2\Psi}{\partial\mathbf{F}^2} = \sum_{\alpha=0}^{d-1}\frac{\partial\hat{\Psi}}{\partial I_\alpha}\frac{\partial^2 I_\alpha}{\partial\mathbf{F}^2} + \sum_{\alpha,\beta=0}^{d-1}\frac{\partial^2\hat{\Psi}}{\partial I_\alpha\partial I_\beta}\frac{\partial I_\alpha}{\partial\mathbf{F}}\otimes\frac{\partial I_\beta}{\partial\mathbf{F}}. \tag{42}$$

If Lamé parameters are to be used with a nonlinear model, the Hessian $\frac{\partial^2\Psi}{\partial\mathbf{F}^2}(\mathbf{F})$ should match that of linear elasticity when evaluated at $\mathbf{F} = \mathbf{I}$. For example, this is why we adjust the Lamé parameters used

in [Macklin and Muller 2021] in Equation (8). See the supplemental technical document for derivation details [Anonymous 2023].

We choose our approximate Hessian in Equation 9 in the paper based on this fact. That is, by omitting all but the first and last terms in Equation (42), our approximate Hessian is both symmetric positive definite and consistent with any model that is set from Lamé coefficients (e.g. from Young's modulus and Poisson's ratio)

$$\tilde{C} = \mu \frac{\partial^2 I_0}{\partial \mathbf{F}^2} + \lambda \frac{\partial I_{d-1}}{\partial \mathbf{F}} \otimes \frac{\partial I_{d-1}}{\partial \mathbf{F}}. \tag{43}$$

Again, see the supplemental technical document for derivation details [Anonymous 2023].

10 COLORING AND PARALLELISM

Parallel implementation of Gauss-Seidel techniques is complicated by data dependencies in the updates. This can be alleviated by careful ordering of sub-iterate position updates. We provide simple color-based orderings for both PBD and PBNG techniques. For PBD, colors are assigned to constraints so that those in the same color do not share incident nodes. Constraints in the same color can then be solved at the same time with no 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 constraints. For each constraint c, we find the minimal color as the least integer that is not contained in the set $\cup_{\mathbf{x}_i \in c} S_{\mathbf{x}_i}$. We then register the color by adding it into $S_{\mathbf{x}_i}$ for each $\mathbf{x}_i$ in constraint c. With PBNG, we color the nodes so that those in the same color do not share any mesh element or weak constraint. For each element or weak constraint c, we maintain a set S_c that stores the colors used by its incident nodes. For a position $\mathbf{x}_i$, we compute its color as the minimal one not contained in the set $\cup_{\mathbf{x}_i \in c} S_c$. Then we register the color by adding it into S_c for each element or weak constraint $\mathbf{x}_i$ is incident to. We observe that coloring the nodes instead of the constraints gives fewer colors. This makes simulations run faster since more work can be done without race conditions. We provide more details on the coloring process in Figure 6. The performance gain is listed in Table 1. Note that we use the omp parallel directive from Intel's OpenMP library for parallelizing the updates.

10.1 Collision Coloring

For simulations with static weak constraints, the coloring is a one-time cost. Otherwise, the colors have to be updated every time the weak constraint structure changes, e.g. from self-collision (see [Anonymous 2023]). We propose a simple coloring scheme for this purpose: only nodes incident to the newly added weak constraints need recoloring. We first compute all nodes $\mathbf{x}_i^{\text{extra}}$ that are incident to newly added weak constraints. For each $\mathbf{x}_i^{\text{extra}}$, we compute the used color set $\cup_{\mathbf{x}_i^{\text{extra}} \in c} S_c$. We use the color of $\mathbf{x}_i^{\text{extra}}$ from the previous time step as an initial guess. If it already exists in the used color set, then we find the minimal color that is not used. This is generally of moderate cost, e.g. in the muscle examples with collisions (Figures 1, 2 and 15), our algorithm takes less than 680ms/frame for recoloring, while the actual simulation takes a total of 67s to run.

11 EXAMPLES

We demonstrate the versatility and robustness of PBNG with a number of representative simulations of quasistatic (and dynamic) hyperelasticity. Examples run with the corotated model used in the algorithm from [Gast et al. 2016] for its accuracy and efficiency. Example 11.2.4, 11.5 and 11.6 are dynamic simulations. The rest are quasistatic simulations. All the examples use Poisson's ratio $\nu = 0.3$. We compare PBNG, PBD, XPBD, XPBD-QS and XPBD-QS (Flipped). For XPBD-QS we do the hyperelastic constraints first, followed by weak constraints. For XPBD-QS (Flipped) the order is swapped. All the examples were run on an AMD Ryzen Threadripper PRO 3995WX CPU using 8 threads. In Table 3, we provide comprehensive performance statistics for PBNG. In Table 2, we provide runtime comparisons between PBNG and the other methods. Note that for efficiency we did not use a line search with Newton's method in our experiments. We note adding line search requires evaluating Newton residuals multiple times, which would further increase cost. As in Figure 7, line search would further reduce number of Newton iterations, making it less stable.

11.1 Stretching Block

We stretch and twist a simple block in a simple scenario. The block has 32K particles and 150K elements. Both ends of the block are clamped. They are stretched, squeezed and twisted in opposite directions. The block has $R^0 = 10kg/m^3$ and Young's modulus $E = 10^5 Pa$. There is no gravity. The simulation is quasistatic. We compare performance between Newton's method, PBD, PBNG and XPBD as described in Section 6. In Figure 7, these methods are run under a fixed budget. Every method has a runtime of 1.3s/frame. With an ample budget, PBNG converges to ground truth, while PBD and XPBD do not. In Figure 7, we show a simulation where every method has a runtime of 170ms/frame. Newton's method is remarkably unstable. PBNG looks visually plausible. PBD and XPBD-QS have visual artifacts and fail to converge. Residual plots vs. time are shown at the bottom of Figure 7.

11.1.1 Resolution Comparison. In this example, we demonstrate PBNG's versatility by running the block stretching and twisting with various resolutions. As shown in Figure 8, the top block has 32K particles and 150K elements. The middle block has 260K particles and 1250K elements. The bottom block has 2097K particles and 10242K elements. Even at high-resolution (bottom block), PBNG is visually plausible after only 40 iterations and 61 seconds/frame of runtime.

11.1.2 Constitutive Model Comparison. In this example, we apply PBNG to various constitutive models on the same block examples. All three blocks have 32K particles and 150K elements. Frames are shown in Figure 9. The blocks from top to bottom are run with corotated (Equation 7), stable Neo-Hookean (Equation 10) and Neo-Hookean (Equation 8) models respectively. With 40 iterations per frame, they are all visually plausible.

11.1.3 Comparison with Linear Gauss-Seidel. In this example, we show the superior performance of the nonlinear Gauss-Seidel strategy in PBNG against Newton's method with linear Gauss-Seidel used at each iteration (see Figure 10). We compare on a representative block example with 32K particles and 150K elements. The simulation is run with both low iteration counts and a high iteration counts. Note that we match the iteration count instead of runtime,

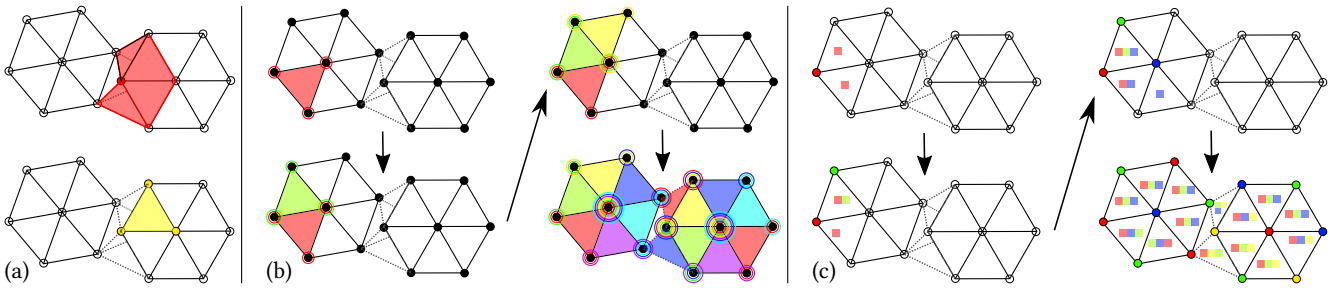

Fig. 6. **(a) Dual Coloring** . Node based coloring (top) is contrasted with constraint based coloring (bottom). When a node is colored as red, its incident elements register red as used colors. When a constraint is colored yellow, its incident particles register yellow as used colors. **(b) Constraints-Based Coloring**. A step-by-step constraint mesh coloring scheme is shown. The dotted line indicates two weak constraints between the elements. The first constraint is colored red, all its incident points will register red as a used color. Other constraints incident to the first constraint have to choose other colors. **(c) Node-Based Coloring**. A step-by-step node coloring scheme is shown. The constraint register the colors used by its incident particles. The first particle is colored red, so all its incident constraints will register red as used. Other particles incident to the constraints have to choose other colors.

Table 1. **Number of Colors Comparison.** Runtime is measured per iteration (averaged over the first 200 iterations). PBNG does more work per-iteration than PBD, but has comparable speed due to improved scaling resulting from a smaller number of colors.

Example	# Vertices	# Elements.	# Particle Colors	# Constraint Colors	PBNG Runtime/Iter	PBD Runtime/Iter
Res 32 Box Stretching	32K	150K	5	39	28ms	26.8ms
Muscles Without Collisions	284k	1097K	13	82	131ms	140ms
Res 64 Box Stretching	260K	1250K	5	39	65ms	137ms
Res 128 Box Stretching	2097K	10242K	5	40	1520ms	1080ms
Dropping Simple Shapes Into Box	256K	1069K	11	52	270ms	140ms
Res 16 Box Dropping	4.1K	17K	5	39	3.6ms	4.1ms

Table 2. **Methods Comparisons**. We show runtime per frame for different methods. Each frame is $\frac{1}{30}$ seconds.

Example	# Vertices	# Elements.	PBNG Runtime	Newton Runtime	PBD Runtime	PBNG # iter	PBD # iter	Newton # iter
Box Stretching (low budget)	32K	150K	170ms	170ms	170ms	6	6	2 (7 CGs)
Box Stretching (big budget)	32K	150K	1.3s	1.3s	1.3s	40	40	11 (10 CGs)
Muscle with collisions	284k	1097K	67s	430s	-	510	-	34 (200CGs)

Table 3. **Performance Table of PBNG**. Runtime is measured for each frame (averaged over the course of the simulation). Each frame is written after advancing time .033.

Example	# Vertices	# Elements	# Triangles	PBNG Runtime / Frame	PBNG # Iter/Frame	# Substeps	Model
Box Stretching (low budget)	32K	150K	0	170ms	6	1	Corotated
Box Stretching (big budget)	32K	150K	0	1300ms	40	1	Corotated
Muscle with Collisions	284k	1097K	0	67000ms	510	17	Corotated
Res 64 Box Stretching	260K	1250K	0	1300ms	20	1	Corotated
Res 128 Box Stretching	2097K	10242K	0	61000ms	40	1	Corotated
Dropping Simple Shapes Into Box	256K	1069K	0	49800ms	136	17	Corotated
Two Moving Blocks Colliding	8.2K	33K	0	1630ms	136	17	Corotated
Box Stretching	32K	150K	0	1300ms	40	1	Stable Neo-Hookean
Box Stretching	32K	150K	0	825ms	40	1	Neo-Hookean
Armadillos Dropping	344K	1320K	8K	101200ms	360	9	Corotated

because computation of the explicit matrix and residual for linear Gauss-Seidel once (620ms) already exceeds the total simulation cost of PBNG (170ms) in the low iteration count setting. For low iteration counts PBNG runs with 6 iterations/frame and linear Gauss-Seidel uses 2 Newton iterations with 3 Gauss-Seidel iteration each. For high iteration counts PBNG runs with 42 iterations/frame and Newton + linear Gauss-Seidel runs 6 Newton iterations with 7 Gauss-Seidel iteration each.

We observe that PBNG has several advantages. It only uses the diagonal blocks on the Hessian with local solves, resulting in a much

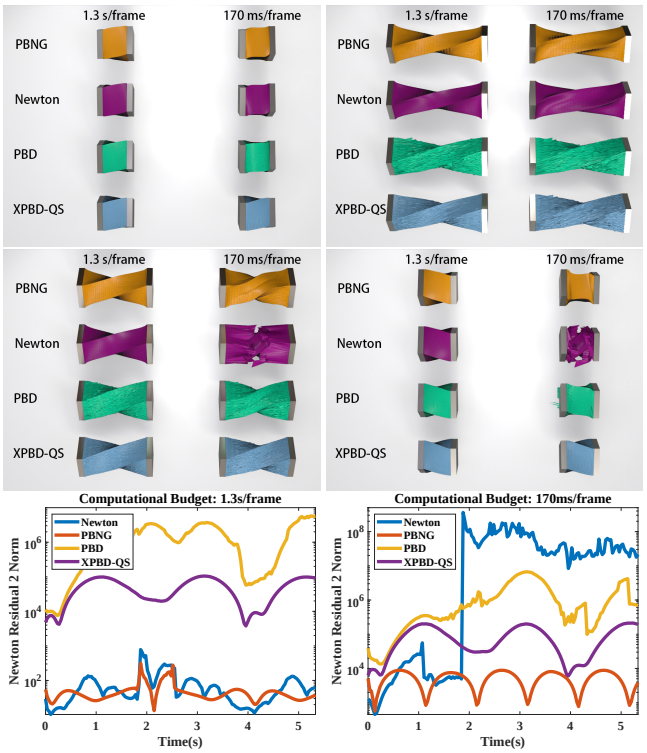

Fig. 7. **Comparisons with Different Computational Budget**. A block is stretched/compressed while being twisted. With a sufficiently large computational budget, Newton's method is stable, but it becomes unstable when the computational budget is small. PBD and XPBD-QS do not significantly reduce the residual in the given computational time, resulting in noisy artifacts on the mesh. PBNG maintains relatively small residuals and generates visually plausible results of the deformable block even if the budget is limited.

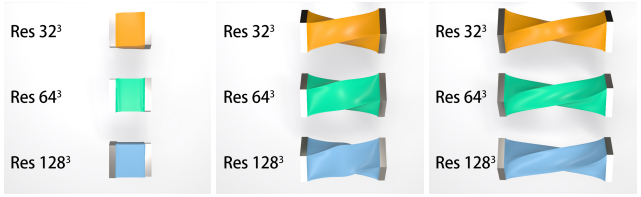

Fig. 8. **Different Mesh Resolution**. PBNG produces consistent results when the mesh is spatially refined. The highest resolution mesh in this comparison has over 2M vertices and only requires 40 iterations to produce visually plausible results.

lower per iteration cost than linear Gauss-Seidel, as shown in Table 4. PBNG does not have the overhead of computing the global Hessian and global residual, which are typically more costly than the entire simulation budget in real-time applications. PBNG achieves clearly superior nonlinear system residual reduction, as shown in Figure 10. Lastly, we observe that linear Gauss-Seidel requires a smaller SOR ω because it is less stable than PBNG in practice. For this example $\omega = 1.3$ for linear Gauss-Seidel and $\omega = 1.7$ for PBNG.

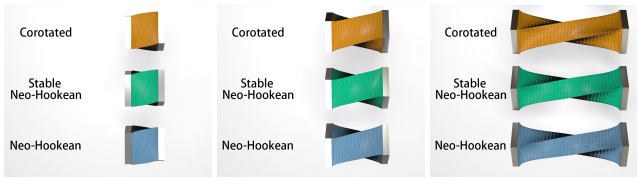

Fig. 9. **Different Constitutive Models**. PBNG works with various constitutive models. We showcase the corotated, Neo-Hookean, and stable Neo-Hoookean models through a block twisting and stretching example.

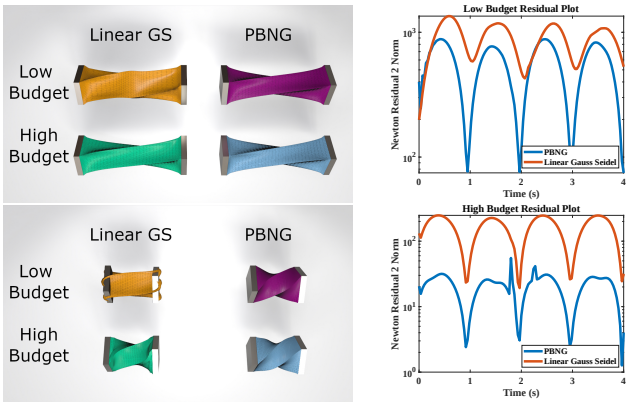

Fig. 10. **Linear Gauss-Seidel vs. PBNG**. PBNG achieves superior residual reduction and visual quality compared to Newton's method with linear Gauss-Seidel.

Fig. 11. **Hessian Comparison**. The top three bars are simulated using Newton's method with different linear solvers (QR, BICGSTAB and linear Gauss-Seidel respectively). The bottom bar is simulated using PBNG. The top bar uses the exact Hessian and becomes unstable. The bottom bar uses our Hessian projection and stays stable.

11.1.4 Approximate Hessian Comparison. In this example we demonstrate the efficacy of our Hessian approximation (Equation 32). All four blocks have 4K particles and 20K elements (see Figure 11). The top three bars are simulated using Newton's method with the exact Hessian and different linear solvers. The top bar uses an exact solve (QR decomposition). The second bar uses an iterative solver (BICGSTAB since the true Hessian is not positive definite) and the third bar uses linear Gauss-Seidel. The bottom bar is simulated using the approximate Hessian in Equation 32. All approaches using the exact Hessian lead to unstable results, while our approximation leads to a correct converged result.

Table 4. **Runtime Breakdown:** We compare the runtimes of linear Gauss-Seidel and PBNG. Newton Overhead refers to the cost of computing the Newton residual and explicit Hessian in each iteration (a cost which PBNG does not require).

Iteration Count	Newton Overhead	Linear GS Runtime/Iter	PBNG Runtime/Iter	Linear GS SOR	PBNG SOR	PBNG Runtime/Frame	Linear GS Runtime/Frame
6	620ms	35ms	27ms	1.3	1.7	170ms	1345ms
40	620ms	35ms	27ms	1.3	1.7	1080ms	5358ms

11.1.5 Acceleration Comparison. In this example, we compare the effects of the Chebyshev semi-iterative method and the SOR method. In Figure 12, we stretch and twist the same block with 32K particles and 150K elements. The top bar is simulated with plain PBNG. The middle bar is simulated with PBNG with Chebyshev semi-iterative method with $\gamma = 1.7, \rho = .95$. The bottom bar is simulated with PBNG with SOR and $\omega = 1.7$. 10 iterations are used for each time step. With a limited budget, plain PBNG is less converged than accelerated techniques. Figure 12 shows the convergence rate of the three methods vs. the number of iterations at the first time step. We can see that the acceleration techniques boost the convergence rate.

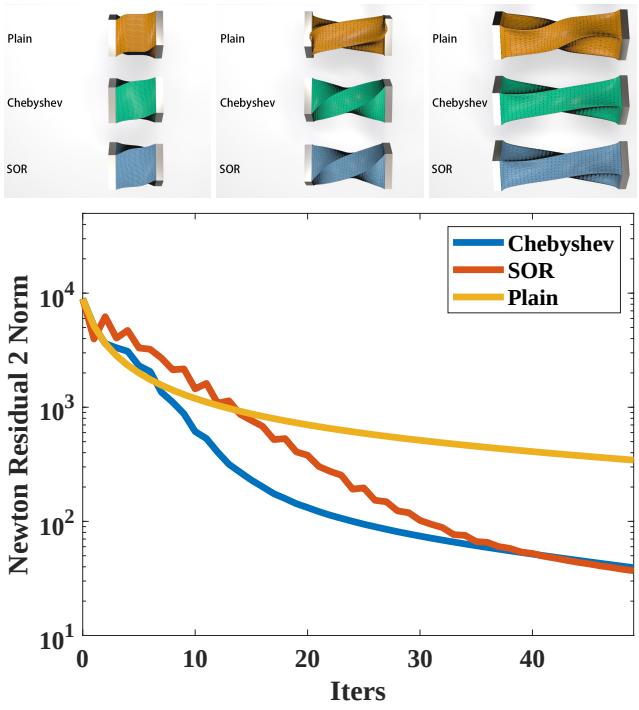

Fig. 12. **Acceleration Techniques**. The convergence rate of PBNG may slow down as the iteration count increases. Chebyshev semi-iterative method and SOR effectively accelerate the Newton residual reduction.

11.2 Collisions

Here we demonstrate our approach with examples that require collision resolution. For all examples in this section, we use a time step of $\Delta t = .002s$ and detect collision every time step. We illustrate how weak constraints are dynamically created for collision with a simple two-block colliding example.

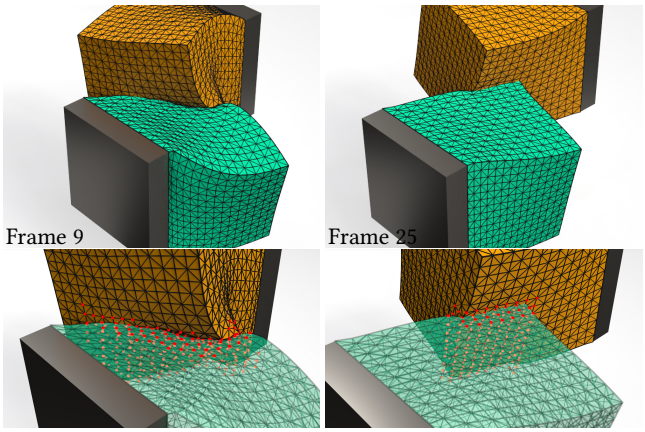

Fig. 13. **Two Blocks Colliding**. Two blocks collide with each other with one face clamped. Red particles indicate that dynamic weak constraints have been built to resolve the collision of corresponding mesh vertices.

11.2.1 Two Blocks Colliding. We demonstrate the generation of dynamic weak constraints with a simple example. We take two blocks with one side fixed and drive them toward each other. This is a dynamic/backward Euler simulation. The blocks have $R^0 = 10kg/m^3$ and Young's modulus $E = 1000Pa$. The weak constraints have stiffness $k_n = 10^8$ and $k_\tau = 0$. The dynamic weak constraints are visualized in Figure 13 as red nodes in the mesh.

11.2.2 Dropping Objects. 40 objects with simple shapes are dropped into a glass box. The objects have a total of 256K particles and 1069K elements. The simulation is run with dynamic/backward Euler. Some frames are shown in Figure 14. We show PBNG's capability of handling collision-intensive scenarios. The example is run with $\Delta t = 0.002s$, $R^0 = 10kg/m^3$, Young's modulus $E = 3000Pa$ and weak constraint stiffness $k_n = 10^8$ and $k_\tau = 0$.

11.2.3 Muscles. We quasistatically simulate a large-scale musculature with collision and connective tissue weak constraints. The mesh has a total of 284K particles and 1097K elements. The muscles have $R^0 = 1000kg/m^3$, Young's modulus $E = 10^5Pa$, and the connective tissue (blue) weak constraint stiffness is isotropic: $k_n = k_\tau = 10^8$. Dynamic collision (red) weak constraint stiffness is anisotropic: $k_n = 10^8$ and $k_\tau = 0$. We show several frames of muscles simulated with PBNG and dynamically generated weak constraints in Figure 15. PBNG takes 67 seconds to simulate a frame, while Newton's method takes 430s. In Figure 1, we show that PBNG looks visually the same as Newton, while running 6-7 times faster. We also show that PBD and XPBD-QS fail to converge. In Figure 1,

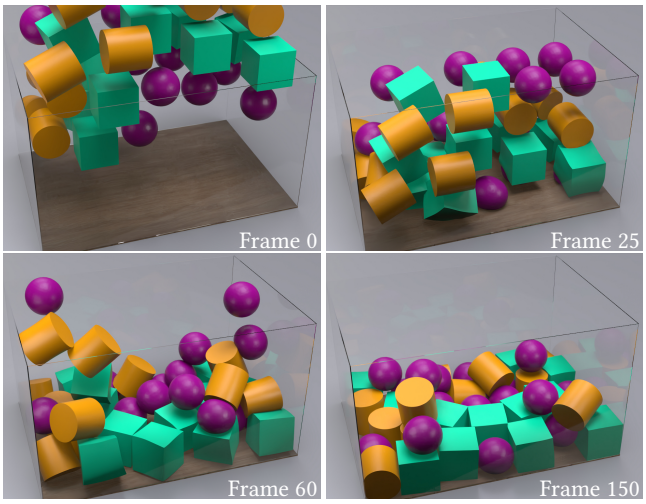

Fig. 14. **Objects Dropping**. A variety of objects drop under gravity. Our method is able to robustly handle collisions between deformable objects through weak constraints.

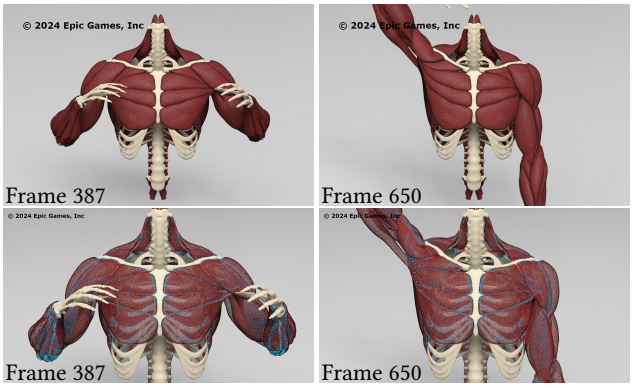

Fig. 15. **PBNG Muscle Simulation**. The top row shows simulation results while the bottom row visualizes the vertex constraint status. *Red* indicates a vertex involved in contact, weak constraints are dynamically built to resolve the collisions. *Blue* represents the vertex positions of connective tissue bindings.

we show PBD becomes unstable. In Figure 2, we demonstrate sub-iteration order-dependent behavior with XPBD-QS. XPBD-QS has weak constraints processed last, which leads to excessive stretching of elements. XPBD-QS (Flipped) has weak constraints processed first, which degrades their enforcement and leaves a gap.

11.2.4 Dropping Armadillos. We showcase the capability of PBNG with a simulation coupling cloth with solids. In this example 20 armadillos are dropped onto a piece of cloth with four corners held fixed. Frames are shown at Figure 16. After 3.33s, one end of the cloth breaks free and armadillos drop into a glass box. Each armadillo has 17K vertices and 66K particles. The rectangular cloth has 4K particles and 8K triangles. We set $\Delta t = 0.004s$, $R^0 = 10kg/m^3$, $E = 1000Pa$. For the rectangular cloth we set $R^0 = 10kg/m^2$, $E^{\text{cod}} = 10000Pa$, $k_b =$

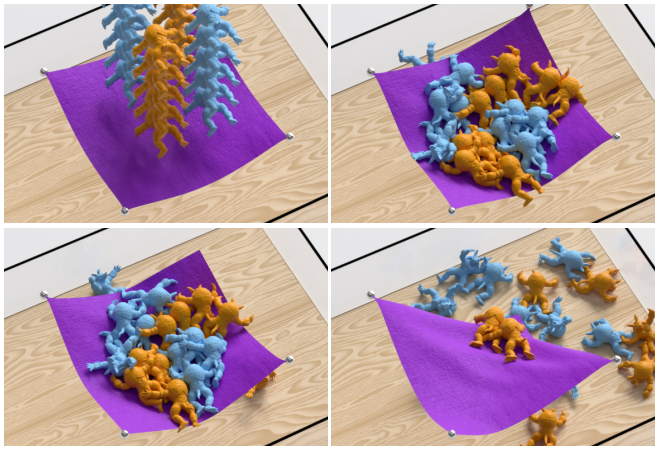

Fig. 16. **Armadillos Dropping**. We demonstrate that PBNG can handle a large-scale simulation involving many collision-driven deformations and with clothing and solids coupled together.

.05. We set Poisson ratio $\nu = 0.3$ for all objects and $k_n = 10^8$ for weak constraints. The average runtime is 101.2s/frame.

11.3 XPBD with Varying Stiffness

In this example, we demonstrate that XPBD-QS fails to resolve quasistatic problems with varied stiffness. In Figure 17, we show the initial setup for the simulation. The simulation is quasistatic. Both block meshes have $R^0 = 10kg/m^3$ and Young's modulus $E = 1000Pa$. The first block mesh has its top boundary constrained. The second block is weakly constrained to the first block. The springs have stiffness $k_n = k_\tau = 10^8$. There is gravity in the scene with acceleration $-9.8m/s^2$ in the $y-$direction. As we show in Figure 17, PBNG converges to a plausible state. XPBD-QS and XPBD-QS (Flipped) fail to converge. Depending on the order of the constraints, it either leaves a gap between the two blocks or a very stretched top layer of the bottom block. This example also serves as a simplified version of the connective bindings on the muscles, which are used in Figure 2. The residual plot is shown on the right of Figure 17.

11.4 Quasistatic PBD/XPBD and External Forcing

In this example, we show how PBD eliminates the effects of external forcing as the number of iterations increases. We clamp the left side of a simple bar mesh. We run a quasistatic simulation with gravity (acceleration $-9.8m/s^2$ in the $y-$direction). The bar has $R^0 = 10kg/m^3$ and Young's modulus $E = 1000Pa$. As shown in Figure 18, PBD converges to a rigid bar configuration. PBNG converges to a plausible solution. XPBD-QS appears to resolve the issues with PBD and quasistatics. However, XPBD-QS with 10 iterations per pseudo-time step appears more converged than XPBD-QS with 1 iteration per pseudo-time step.

11.5 Multiresolution Test: Cloth Stretching

We demonstrate how our multi-layer (MPBNG) approach resolves excessive stretching with PBNG when the iteration count is low. A rectangular shape cloth with 8K vertices is suspended from two

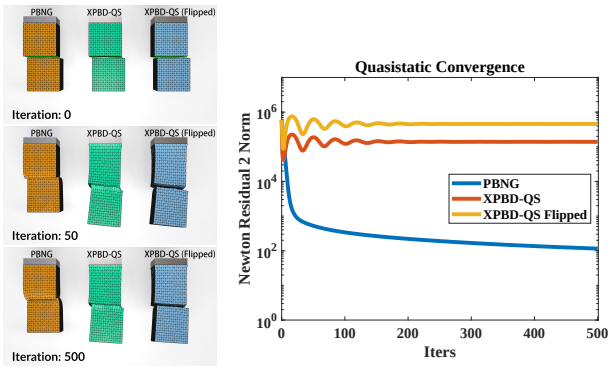

Fig. 17. **Two Blocks Hanging**. Two identical blocks are bound together through weak constraints. Green line segments in iteration 0 indicate weak constraint springs. PBNG is able to reduce the residual by a few orders of magnitude and converges quickly. XPBD-QS methods demonstrate iteration-order-dependent behavior. Residuals oscillate and produce visually incorrect results.

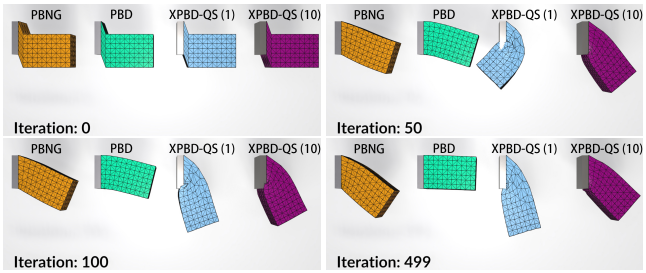

Fig. 18. **Bar under Gravity**. A quasistatic simulation of a bar bending under gravity using different methods. The effect of external forcing vanishes in the PBD example as the number of iterations increases. More local iterations of XPBD-QS produces better results. PBNG converges to visually plausible results within fewer iterations than XPBD-QS.

corners under gravity. The cloth has density $R^0 = 10kg/m^2$, Young's modulus $E = 1000Pa$, $k_b = 0$. We run both PBNG and MPBNG with a budget of 1.8s/frame of computation each. We take $dt = 0.005s$ and use 30 frames per second. PBNG is run with 44 iterations per timestep and MPBNG is run with 13 iterations per timestep with $r_{pad} = 0.05$. As shown in Figure 19, PBNG appears to be much stretchier than the converged simulation, while MPBNG is less stretched and almost indistinguishable from the well-converged ideal solution (despite MPBNG having the same computational run time as PBNG).

11.6 Multiresolution Test: Clothing on Mannequin

We demonstrate the ability of MPBNG to reduce excessive stretching with a dress example. The dress has 24K vertices, density $R^0 = 10kg/m^2$, Young's modulus $E = 1000Pa$ and $k_b = 0.01$. The simulation is run with $\Delta t = .012s$ and a fixed budget of 840ms/frame. PBNG has 84 iterations per frame with this budget and MPBNG has 24 iterations. We create a mesh hierachy of four layers, where the layers have .5K, 1.5K, 6K and 24K particles respectively (see Figure 5). As

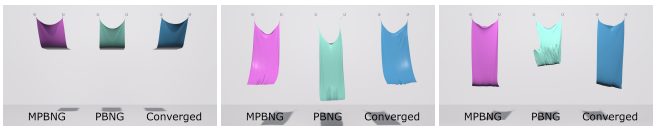

Fig. 19. **Draping Cloth**. A piece of rectangular cloth with two corners clamped swings under gravity. With a fixed computational time of 1.8s/frame, MPBNG is much more similar to the converged look of the cloth than PBNG which appears too stretchy.

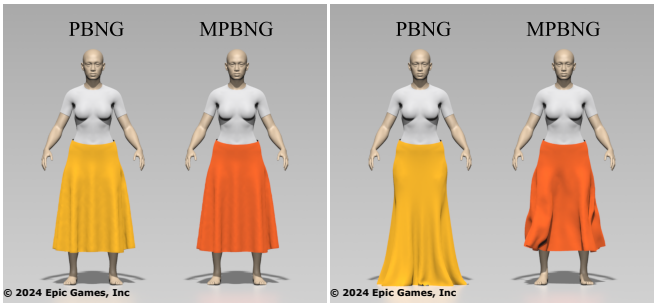

Fig. 20. **Draping Dress**. With a fixed computational time of 840ms/frame, MPBNG cloth appears much less stretchy than PBNG alone.

shown in Figure 20, PBNG is stretchier than MPBNG given the same computational budget. We further demonstrate MPBNG's capabilities when the mannequin runs (see Figure 1). We take $dt = 0.003s$ and use 20 iterations per time step with $r_{\text{pad}} = 2$.

11.7 XPBD

We run a simple dynamics example to show that XPBD cannot reduce the backward Euler system residual in practice, as discussed in Chen et al. [2023]. We take a simple block with the left side clamped. It falls under gravity and oscillates. The simulation scene is shown on the top of Figure 3. The block has 4.1K particles and 17K elements. This simple but representative example demonstrates superior convergence behavior of PBNG over XPBD.

12 DISCUSSION AND LIMITATIONS

We show that a node-based Gauss-Seidel approach for the nonlinear equations of quasistatic and backward Euler time stepping has remarkably stable behavior. We show that it is capable of reducing the nonlinear system residuals in practice, in contrast to PBD/XPBD. Furthermore, we show that our node-based Gauss-Seidel approach resolves fundamental issues with PBD/XPBD for quasistatic problems, particularly for applications that require efficient and reliable creation of training data. However, our approach is mostly tailored to isotropic hyperelasticity. In future work, generalizing to the case of transverse and general anisotropy is of interest.

REFERENCES

Anonymous. 2023. *Supplementary Technical Document* (2023).
R. Atkin and R. Craine. 1976. Continuum theories of mixtures: basic theory and historical development. *Quart J Mech App Math* 29, 2 (1976), 209–244.
S. Bailey, D. Otte, P. Dilorenzo, and J. O'Brien. 2018. Fast and Deep Deformation Approximations. *ACM Trans Graph* 37, 4 (Aug. 2018), 119:1–12. https://doi.org/10.

1145/3197517.3201300
D. Baraff and A. Witkin. 1998. Large Steps in Cloth Simulation. In *Proc ACM SIGGRAPH (SIGGRAPH '98)*. 43–54.
H. Bertiche, M. Madadi, and S. Escalera. 2021. PBNS: Physically Based Neural Simulator for Unsupervised Garment Pose Space Deformation. arXiv:2012.11310 [cs.CV]
D. Bertsekas. 1997. Nonlinear programming. *J Op Res Soc* 48, 3 (1997), 334–334.
J. Bonet and R. Wood. 2008. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press.
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.
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.
A. Brandt. 1977. Multi-level adaptive solutions to boundary-value problems. *Math Comp* 31, 138 (1977), 333–390.
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.
Y. Chen, Y. Han, J. Chen, MS. a, R. Fedkiw, and J. Teran. 2023. Primal Extended Position Based Dynamics for Hyperelasticity. In *Proceedings of the 16th ACM SIGGRAPH Conference on Motion, Interaction and Games (MIG '23)*. Article 21, 10 pages.
N. Chentanez, M. Macklin, M. Müller, S. Jeschke, and T. Kim. 2020. Cloth and Skin Deformation with a Triangle Mesh Based Convolutional Neural Network. *Comp Graph Forum* 39, 8 (2020), 123–134.
O. Etzmuss, J. Gross, and W. Strasser. 2003. Deriving a particle system from continuum mechanics for the animation of deformable objects. *IEEE Trans Vis Comp Graph* 9, 4 (Oct. 2003), 538–550.
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
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.
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.
Z. Geng, D. Johnson, and R. Fedkiw. 2020. Coercing machine learning to output physically accurate results. *J Comp Phys* 406 (2020), 109099. https://doi.org/10.1016/j.jcp.2019.109099
O. Gonzalez and A. Stuart. 2008. *A first course in continuum mechanics*. Cambridge University Press.
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.
N. Jin, Y. Zhu, Z. Geng, and R. Fedkiw. 2020. A Pixel-Based Framework for Data-Driven Clothing. In *Proc ACM SIGGRAPH/Eurographics Symp Comp Anim* (Virtual Event, Canada) *(SCA '20)*. Eurographics Association, Article 13, 10 pages. https://doi.org/10.1111/cgf.14108
Y. Jin, Y. Han, Z. Geng, J. Teran, and R. Fedkiw. 2022. Analytically Integratable Zero-Restlength Springs for Capturing Dynamic Modes Unrepresented by Quasistatic Neural Networks. In *ACM SIGGRAPH 2022 Conf Proc* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. ACM, New York, NY, USA, Article 37, 9 pages. https://doi.org/10.1145/3528233.3530705
S. Kovalsky, M. Galun, and Y. Lipman. 2016. Accelerated Quadratic Proxy for Geometric Optimization. *ACM Trans Graph* 35, 4, Article 134 (2016), 11 pages.
L. Lan, M. Li, C. Jiang, H. Wang, and Y. Yang. 2023. Second-order Stencil Descent for Interior-point Hyperelasticity. *ACM Trans Graph* 42, 4 (2023), 108:1–108:16.
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
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.
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.
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.
R. Luo, T. Shao, H. Wang, W. Xu, X. Chen, K. Zhou, and Y. Yang. 2020. NNWarp: Neural Network-Based Nonlinear Deformation. , 1745-1759 pages. https://doi.org/10.1109/TVCG.2018.2881451
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
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.
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.
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.
M.Tournier, M. Nesme, B. Gilles, and F. Faure. 2015. Stable Constrained Dynamics. *ACM Trans Graph* (2015), 1–10.
M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. 2002. Stable real-time deformations. In *Proc 2002 ACM SIGGRAPH/Eurograph Symp Comp Anim*. 49–54.
M. Müller and M. Gross. 2004. Interactive virtual materials. In *Proc Graph Int*. Canadian Human-Computer Communications Society, 239–246.
M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. 2007. Position based dynamics. *J Vis Comm Im Rep* 18, 2 (2007), 109–118.
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.
J. Neuberger. 1985. Steepest descent and differential equations. *J Math Soc Japan* 37, 2 (1985), 187–195.
J. Nocedal and S. Wright. 2006. Conjugate gradient methods. *Num Opt* (2006), 101–134.
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.
R. Schmedding and M. Teschner. 2008. Inversion handling for stable deformable modeling. *Vis Comp* 24, 7-9 (2008), 625–633.
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.
B. Smith, F. Goes, and T. Kim. 2019. Analytic eigensystems for isotropic distortion energies. *ACM Trans Graph (TOG)* 38, 1 (2019), 1–15.
B. Smith, F. De Goes, and T. Kim. 2018. Stable neo-hookean flesh simulation. *ACM Trans Grap (TOG)* 37, 2 (2018), 1–15.
O. Sorkine and M. Alexa. 2007. As-Rigid-As-Possible Surface Modeling. In *EUROGRAPHICS SYMPOSIUM ON GEOMETRY PROCESSING*.
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.
A. Stomakhin, R. Howes, C. Schroeder, and J. Teran. 2012. Energetically consistent invertible elasticity. In *Proc Symp Comp Anim*. 25–32.
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.
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.
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
Z. Wang, L.Wu, M. Fraftarcangeli, M. Tang, and H. Wang. 2018. Parallel Multigrid for nonlinear cloth simulation. *Computer Graphics Forum* 37, 7 (2018), 131–141. https://doi.org/10.1111/cgf.13554
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.
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.