Quasi-Newton Methods for Real-Time Simulation of Hyperelastic Materials

TIANTIAN LIU
University of Pennsylvania
SOFIEN BOUAZIZ
École polytechnique fédérale de Lausanne
and
LADISLAV KAVAN
University of Utah

We present a new method for real-time physics-based simulation supporting many different types of hyperelastic materials. Previous methods such as Position-Based or Projective Dynamics are fast but support only a limited selection of materials; even classical materials such as the Neo-Hookean elasticity are not supported. Recently, Xu et al. [2015] introduced new "spline-based materials" that can be easily controlled by artists to achieve desired animation effects. Simulation of these types of materials currently relies on Newton's method, which is slow, even with only one iteration per timestep. In this article, we show that Projective Dynamics can be interpreted as a quasi-Newton method. This insight enables very efficient simulation of a large class of hyperelastic materials, including the Neo-Hookean, spline-based materials, and others. The quasi-Newton interpretation also allows us to leverage ideas from numerical optimization. In particular, we show that our solver can be further accelerated using L-BFGS updates (Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm). Our final method is typically more than 10 times faster than one iteration of Newton's method without compromising quality. In fact, our result is often more accurate than the result obtained with one iteration of Newton's method. Our method is also easier to implement, implying reduced software development costs.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*

General Terms: Physics-based Animation

Additional Key Words and Phrases: Physics-based animation, material models, numerical optimization

Authors' addresses: L. Kavan (correspondence author), School of Computing, University of Utah, Salt Lake City, UT; email: ladislav.kavan@gmail.com.

DOI: http://dx.doi.org/10.1145/2990496

ACM Reference Format:

T. Liu, S. Bouaziz, and L. Kavan. 2017. Quasi-newton methods for real-time simulation of hyperelastic materials. ACM Trans. Graph. 36, 3, Article 23 (May 2017), 16 pages.
DOI: http://dx.doi.org/10.1145/2990496

1. INTRODUCTION

Physics-based animation is an important tool in computer graphics, even though creating visually compelling simulations often requires a lot of patience. Waiting for results is not an option in real-time simulations, which are necessary in applications such as computer games and training simulators (e.g., surgery simulators). Previous methods for real-time physics such as Position-Based Dynamics [Müller et al. 2007] or Projective Dynamics [Bouaziz et al. 2014] have been successfully used in many applications and commercial products, despite the fact that these methods support only a restricted set of material models. Even classical models from continuum mechanics, such as the Neo-Hookean, St. Venant-Kirchoff, or Mooney-Rivlin materials, are not supported by Projective Dynamics. We tried to emulate their behavior with Projective Dynamics, but despite our best efforts, there are still obvious visual differences when compared to simulations with the original nonlinear materials.

The advantages of more general material models were nicely demonstrated in the recent work of Xu et al. [2015], who proposed a new class of spline-based materials particularly suitable for physics-based animation. Their user-friendly spline interface enables artists to easily modify material properties in order to achieve desired animation effects. However, their system relies on Newton's method, which is slow, even if the number of Newton's iterations per frame is limited to one. Our method enables fast simulation of spline-based materials, combining the benefits of artist-friendly material interfaces with the advantages of fast simulation, such as rapid iterations and/or higher resolutions.

Physics-based simulation can be formulated as an optimization problem where we minimize a multivariate function g. Newton's method minimizes g by performing descent along direction $-(\nabla^2 g)^{-1}\nabla g$, where $\nabla^2 g$ is the Hessian matrix, and ∇g is the gradient. One problem of Newton's method is that the Hessian $\nabla^2 g$ can be indefinite, in which case Newton's direction could erroneously *increase* g. This undesired behavior can be prevented by so-called definiteness fixes [Teran et al. 2005; Nocedal and Wright 2006]. While definiteness fixes require some computational overheads, the slow speed of Newton's method is mainly caused by the fact that

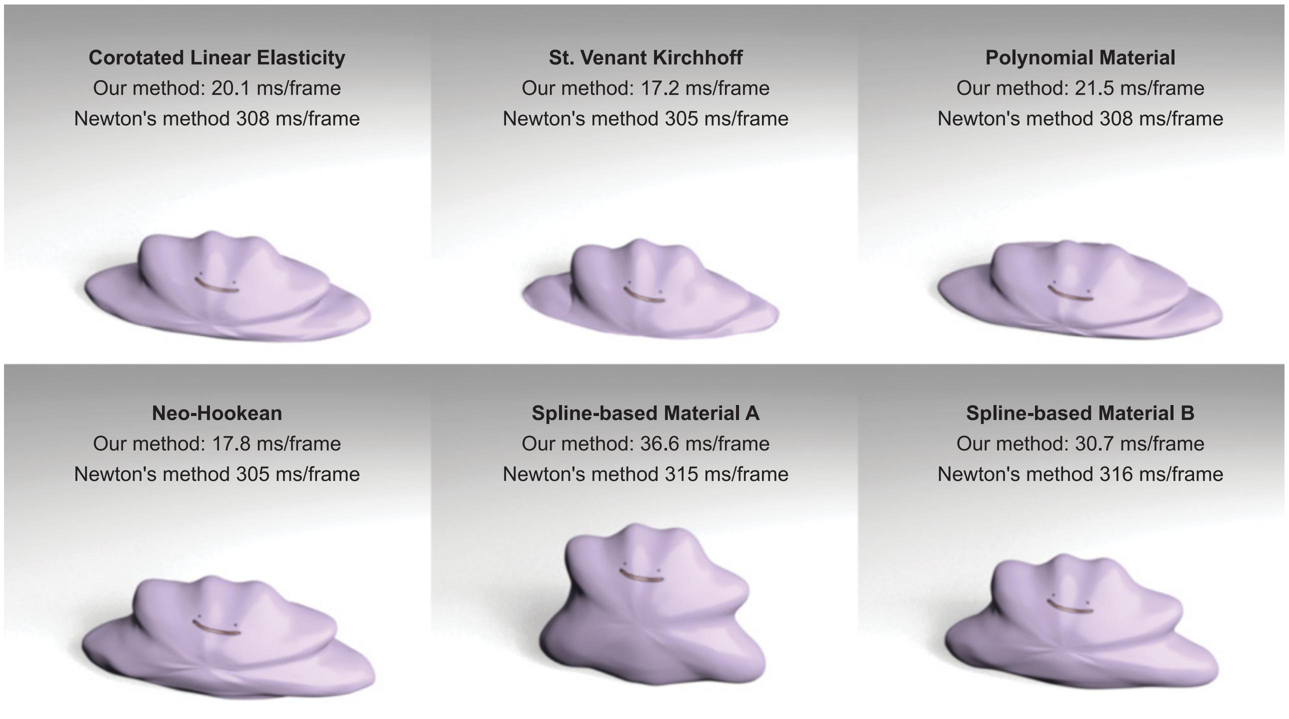
Corotated Linear Elasticity
Our method: 20.1 ms/frame
Newton's method 308 ms/frame
St. Venant Kirchhoff
Our method: 17.2 ms/frame
Newton's method 305 ms/frame
Polynomial Material
Our method: 21.5 ms/frame
Newton's method 308 ms/frame
Neo-Hookean
Our method: 17.8 ms/frame
Newton's method 305 ms/frame
Spline-based Material A
Our method: 36.6 ms/frame
Newton's method 315 ms/frame
Spline-based Material B
Our method: 30.7 ms/frame
Newton's method 316 ms/frame

Fig. 1. Our method enables fast simulation of many different types of hyperelastic materials. Compared to the commonly applied Newton's method, our method is about 10 times faster, while achieving even higher accuracy and being simpler to implement. The polynomial and spline-based materials are models recently introduced by Xu et al. [2015]. Spline-based material A is a modified Neo-Hookean material with stronger resistance to compression; spline-based material B is a modified Neo-Hookean material with stronger resistance to tension.

the Hessian changes at every iteration; that is, we need to solve a *new* linear system for every Newton step.

The point of departure for our method is the insight that Projective Dynamics can be interpreted as a special type of quasi-Newton method. In general, quasi-Newton methods [Nocedal and Wright 2006] work by replacing the Hessian $\nabla^2 g$ with a linear operator $\mathbf{A}$, where $\mathbf{A}$ is positive definite and solving linear systems $\mathbf{Ax} = \mathbf{b}$ is fast. The descent directions are then computed as $-\mathbf{A}^{-1}\nabla g$ (where the inverse is of course not explicitly evaluated—in fact, $\mathbf{A}$ is often not even represented with a matrix). The tradeoff is that if $\mathbf{A}$ is a poor approximation of the Hessian, the quasi-Newton method may converge slowly. Unfortunately, coming up with an effective approximation of the Hessian is not easy. We tried many previous quasi-Newton methods, but even after boosting their performance with L-BFGS [Nocedal and Wright 2006], we were unable to obtain an effective method for real-time physics. We show that Projective Dynamics can be reformulated as a quasi-Newton method with some remarkable properties; in particular, the resulting $\mathbf{A}_{\mathtt{our}}$ matrix is *constant* and *positive definite*. This reformulation enables us to generalize the method to hyperelastic materials not supported by Projective Dynamics, such as the Neo-Hookean or spline-based materials. Even though the resulting solver is slightly more complicated than Projective Dynamics (in particular, we must employ a line search to ensure stability), the computational overhead required to support more general materials is rather small.

The quasi-Newton formulation also allows us to further improve convergence of our solver. We propose using L-BFGS, which uses curvature information estimated from a certain number of previous iterates to improve the accuracy of our Hessian approximation $\mathbf{A}_{\mathtt{our}}$. Adding the L-BFGS Hessian updates introduces only a small computational overhead while accelerating the convergence of our method. However, the performance of L-BFGS highly depends on the quality of the initial Hessian approximation. With previous quasi-Newton methods, we observed rather disappointing convergence properties (see Figure 7). The combination of our Hessian approximation $\mathbf{A}_{\mathtt{our}}$ with L-BFGS is quite effective and can be interpreted as a generalization of the recently proposed Chebyshev Semi-Iterative method for accelerating Projective Dynamics [Wang 2015].

The L-BFGS convergence boosting is compatible with our first contribution, that is, fast simulation of complex non-linear materials. Specifically, we can simulate any materials satisfying the Valanis-Landel assumption [Valanis and Landel 1967], which includes many classical materials, such as St. Venant-Kirchhoff, Neo-Hookean, Mooney-Rivlin, and also the recently proposed spline-based materials [Xu et al. 2015] (none of which is supported by Projective Dynamics). In summary, our final method achieves faster convergence than Projective Dynamics while being able to simulate a large variety of hyperelastic materials.

2. RELATED WORK

The work of Terzopoulos et al. [1987] pioneered physics-based animation, nowadays an indispensable tool in feature animation and visual effects. Real-time physics became widespread only more recently, with the first success stories represented by real-time rigid body simulators, commercially offered by companies such as Havok since early 2000s. Fast simulation of deformable objects is more challenging because they feature many more degrees of freedom than rigid bodies. Fast simulations of deformable objects using shape matching [Müller et al. 2005; Rivers and James 2007] paved the way toward more general Position-Based Dynamics methods [Müller et al. 2007; Stam 2009]. The past decade witnessed rapid development of Position-Based methods, including improvements of the convergence [Müller 2008; Kim et al. 2012], robust simulation of elastic models [Müller and Chentanez 2011], generalization to fluids [Macklin and Müller 2013] and continuum-based materials [Müller et al. 2014; Bender et al. 2014a], unified solvers including multiple phases of matter [Macklin et al. 2014], and, most recently, methods to avoid element inversion [Müller et al. 2015]. We refer to a recent survey [Bender et al. 2014b] for a more detailed summary of Position-Based methods.

A new interpretation of Position-Based methods was offered by Liu et al. [2013], observing that Position-Based Dynamics can be interpreted as an approximate solver for Implicit Euler time-stepping. The same paper introduces a fast local/global solver for mass-spring systems integrated using Implicit Euler. This method was later generalized to Projective Dynamics [Bouaziz et al. 2014] by combining the ideas of Liu et al. [2013] with a shape editing system "Shape-Up" [Bouaziz et al. 2012]. Recently, a Chebyshev Semi-Iterative method [Wang 2015] has been proposed to accelerate convergence of Projective Dynamics while also exploring highly parallel GPU implementations of real-time physics. Concurrently to our work, Narain et al. [2016] interpreted Projective Dynamics as a special case of the Alternating Direction Method of Multipliers (ADMM), leading to another way to enable simulation of more general elastic materials.

Multigrid methods represent another approach to accelerate physics-based simulations [Georgii and Westermann 2006; Müller 2008; Wang et al. 2010; McAdams et al. 2011; Tamstorf et al. 2015]. Multigrid methods are attractive especially for highly detailed meshes where sparse direct solvers become hindered by high memory requirements. However, constructing multiresolution data structures and picking suitable parameters is not a trivial task. Another way to speed up FEM is by using subspace simulation, where the nodal degrees of freedom are replaced with a low-dimensional linear subspace [Barbič and James 2005; An et al. 2008; Li et al. 2014]. These methods can be very efficient; however, deformations that were not accounted for during the subspace construction may not be well represented. A variety of approaches have been designed to address this limitation while trying to preserve efficiency [Harmon and Zorin 2013; Teng et al. 2014, 2015]. Simulating at coarser resolutions is also possible, while crafting special data-driven materials that avoid the loss of accuracy typically is associated with lower resolutions [Chen et al. 2015].

The concept of constraint projection, which appears in both Position-Based and Projective Dynamics, is also central to the Fast Projection method [Goldenthal et al. 2007] and strain-limiting techniques [Thomaszewski et al. 2009; Narain et al. 2012]. The Fast Projection method and Position-Based Dynamics formulate physics simulation as a constrained optimization problem that is solved by linearizing the constraints in the spirit of sequential quadratic programming [Macklin et al. 2014]. The resulting Karush-Kuhn-Tucker (KKT) equation system is then solved using a direct solver [Goldenthal et al. 2007] or an iterative method such as Gauss-Seidel [Müller et al. 2007; Stam 2009; Fratarcangeli and Pellacini 2015], Jacobi [Macklin and Müller 2013], or their under-/overrelaxation counterparts [Macklin et al. 2014]. By using a constrained optimization formulation, the Fast Projection method and Position-Based Dynamics are designed for solving infinitely stiff systems but are not appropriate to handle soft materials. This problem can be overcome by regularizing the KKT system [Servin et al. 2006; Tournier et al. 2015], leading to approaches that can accurately handle extremely high tensile forces (e.g., string of a bow) but also support soft (compliant) constraints. However, these methods are slower than Projective Dynamics because a new linear system has to be solved at each iteration.

The idea of quasi-Newton methods in elasticity is not new and was studied for a long time before real-time simulations were feasible. Several research papers have been done to accelerate Newton's method in FEM simulations by updating the Hessian approximation only once every frame [Bathe and Cimento 1980; Fish et al. 1995]. However, even one Hessian update is usually so computationally expensive that it cannot fit into the computing time limit of real-time applications. Deuflhard [2011] minimizes the number of Hessian factorizations by carefully scheduled Hessian updates. But the update rate will heavily depend on the deformation. A good Hessian approximation suitable for real-time applications should be easy to refactorize or capable of prefactorization. One straightforward constant approximation that is good for prefactorization is the Hessian evaluated at the rest pose (undeformed configuration). The rest-pose Hessian is positive semidefinite and its use at any configuration enables prefactorization. Unfortunately, the actual Hessian of deformed configurations is often very different from the rest-pose Hessian, and this approximation is therefore not satisfactory for larger deformations [Müller et al. 2002].

To improve on this, Müller et al. [2002] introduced per-vertex "stiffness warping" of the rest-pose Hessian, which is more accurate and can still leverage prefactorized rest-pose Hessian. Unfortunately, the per-vertex stiffness warping approach can introduce nonphysical ghost forces that violate momentum conservation and can lead to instabilities [Müller and Gross 2004]. This problem was addressed by *per-element* stiffness warping [Müller and Gross 2004], which avoids the ghost forces, but unfortunately, the per-element-warped stiffness matrices need to be refactorized, introducing computational overheads that are prohibitive in real-time simulation. For corotated elasticity, Hecht et al. [2012] proposed an improved method that can reuse previously computed Hessian factorization. Specifically, the sparse Cholesky factors are updated only *when* necessary and also only *where* necessary. This spatiotemporal staging of Cholesky updates improves runtime performance; however, the Cholesky updates are still costly and their scheduling can be problematic, especially in real-time applications, which require approximately constant per-frame computing costs. Also, the frequency of Cholesky updates depends on the simulation: fast motion with large deformations will require more frequent updates and thus more computation or risk ghost forces and potential instabilities. Neither is an option in real-time simulators. Recently, Kovalsky et al. [2016] introduced the idea of quadratic proxies to accelerate optimization problems arising in geometry processing. The "quadratic proxy" can be seen as replacing the exact Hessian matrix with the Laplacian matrix of the mesh; their method can therefore also be categorized as a quasi-Newton method, closely related to our method.

Our reformulation of Projective Dynamics as a quasi-Newton method reveals relationships to so-called Sobolev gradient methods, which have been studied since the 1980s in the continuous setting [Neuberger 1983]; see also the more recent monograph [Neuberger 2009]. The idea of quasi-Newton methods appears already in Desbrun et al. [1999] and Hauth and Etzmuss [2001] in the context of mass-spring systems and, more recently, in Martin et al. [2013] in the context of geometry processing. Martin et al. [2013] also propose multiscale extensions and discuss an application in physics-based simulation, but they consider only the case of thin shells and their numerical method alters the physics of the simulated system. Quasi-Newton methods are also useful in situations where computation of the Hessian would be expensive or impractical [Nocedal and Wright 2006]. In character animation, Hahn et al. [2012] used BFGS to simulate physics in "Rig Space," which is challenging because the rig is a black-box function and its derivatives are approximated using finite differences.

3. BACKGROUND

Projective Dynamics. We start by introducing our notation and recapitulating the key concepts of Projective Dynamics. Let $\mathbf{x} \in \mathbb{R}^{n\times 3}$ be the current (deformed) state of our system containing n nodes, each with three spatial dimensions. Projective Dynamics requires a special form of elastic potential energies, based on the concept of constraint projection. Specifically, Projective Dynamics energy for element number i is defined as

$$E_i(\mathbf{x}) = \min_{\mathbf{p}_i \in \mathcal{M}_i} \tilde{E}_i(\mathbf{x}, \mathbf{p}_i), \quad \tilde{E}_i(\mathbf{x}, \mathbf{z}) = \|\mathbf{G}_i\mathbf{x} - \mathbf{z}\|_F^2, \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm, $\mathcal{M}_i$ is a constraint manifold, $\mathbf{p}_i$ is an auxiliary "projection variable," and $\mathbf{G}_i$ is a discrete differential operator represented, for example, by a sparse matrix. For example, if element number i is a tetrahedron, $\mathcal{M}_i$ is $SO(3)$, and $\mathbf{G}_i$ is a deformation gradient operator [Sifakis and Barbič 2012], we obtain the well-known as-rigid-as-possible material model [Chao et al. 2010]. Another elementary example is a spring, where the element is an edge, $\mathcal{M}_i$ is a sphere, and $\mathbf{G}_i$ subtracts two endpoints of the spring. If all elements are springs, Projective Dynamics becomes equivalent to the work of Liu et al. [2013]. The key property of $\mathbf{G}_i$ is that constant vectors are in its nullspace, which makes E_i translation invariant. The total energy of the system is

$$E(\mathbf{x}) = \sum_i w_i E_i(\mathbf{x}), \tag{2}$$

where i indexes elements and $w_i > 0$ is a positive weight, typically defined as the product of undeformed volume and stiffness.

Time integration. As discussed by Martin et al. [2011], Backward Euler time integration can be expressed as a minimization of

$$g(\mathbf{x}) = \frac{1}{2h^2}\underbrace{\mathrm{tr}((\mathbf{x}-\mathbf{y})^\mathsf{T}\mathbf{M}(\mathbf{x}-\mathbf{y}))}_{\text{inertia}} + \underbrace{E(\mathbf{x})}_{\text{elasticity}}, \tag{3}$$

where $\mathbf{y}$ is a constant depending only on previously computed states, $\mathbf{M}$ is a positive definite mass matrix (typically diagonal—mass lumping), and $h > 0$ is the time step (we use fixed h corresponding to the frame rate of $30fps$, i.e., $h = 1/30s$). The trace (tr) reflects the fact that there are no dependencies between the x, y, z coordinates, which enables us to work only with $n\times n$ matrices (as opposed to more general $3n \times 3n$ matrices). This is somewhat moot in the context of the mass matrix $\mathbf{M}$, but it will be more important in the following. The constant $\mathbf{y}$ is defined as $\mathbf{y} := 2\mathbf{q}_l - \mathbf{q}_{l-1} + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$, where $\mathbf{q}_l \in \mathbb{R}^{n\times 3}$ is the current state, $\mathbf{q}_{l-1}$ is the previous state, and $\mathbf{f}_{\text{ext}} \in \mathbb{R}^{n\times 3}$ are external forces such as gravity. The minimizer of $g(\mathbf{x})$ will become the next state, $\mathbf{q}_{l+1}$. Intuitively, the first term in Equation (3) can be interpreted as "inertial potential," attracting $\mathbf{x}$ toward $\mathbf{y}$, where $\mathbf{y}$ corresponds to state predicted by Newton's first law—motion without the presence of any internal forces. The second term penalizes states $\mathbf{x}$ with large elastic deformations. Minimization of $g(\mathbf{x})$ corresponds to finding a balance between the two terms. Note that many other implicit integration schemes can also be expressed as minimization problems similar to Equation (3). In particular, we have implemented Implicit Midpoint, which has the desirable feature of being symplectic [Hairer et al. 2002; Kharevych et al. 2006]. Unfortunately, in our experiments, we found Implicit Midpoint to be markedly less stable than Backward Euler and, therefore, we continue to use Backward Euler despite its numerical damping.

Local/global solver. The key idea of Projective Dynamics is to expose the auxiliary projection variables $\mathbf{p}_i$, taking advantage of the special energy form according to Equation (1). To simplify notation, we stack all projection variables into $\mathbf{p} \in \mathbb{R}^{c\times 3}$ and define binary selector matrices $\mathbf{S}_i$ such that $\mathbf{p}_i = \mathbf{S}_i\mathbf{p}$, where c is the dimensionality of each constraint; for example, a spring corresponds to $c = 1$ and a tetrahedron to $c = 3$. Projective Dynamics uses the augmented objective

$$\tilde{g}(\mathbf{x}, \mathbf{p}) = \frac{1}{2h^2}\mathrm{tr}((\mathbf{x}-\mathbf{y})^\mathsf{T}\mathbf{M}(\mathbf{x}-\mathbf{y})) + \sum_i w_i \tilde{E}(\mathbf{x}, \mathbf{S}_i\mathbf{p}), \tag{4}$$

which is minimized over both $\mathbf{x}$ and $\mathbf{p}$, subject to the constraint $\mathbf{p} \in \mathcal{M}$, where $\mathcal{M}$ is a Cartesian product of the individual constraint manifolds. The optimization is solved using an alternating (local/global) solver. In the local step, $\mathbf{x}$ is assumed to be fixed; the optimal $\mathbf{p}$ are given by projections on individual constraint manifolds, for example, projecting each deformation gradient (a 3×3 matrix) on $SO(3)$. In the global step, $\mathbf{p}$ is assumed to be fixed and we rewrite the objective $\tilde{g}(\mathbf{x}, \mathbf{p})$ in matrix form:

$$\frac{1}{2h^2}\mathrm{tr}((\mathbf{x}-\mathbf{y})^\mathsf{T}\mathbf{M}(\mathbf{x}-\mathbf{y})) + \frac{1}{2}\mathrm{tr}(\mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x}) - \mathrm{tr}(\mathbf{x}^\mathsf{T}\mathbf{J}\mathbf{p}) + C, \tag{5}$$

where $\mathbf{L} := \sum w_i \mathbf{G}_i^\mathsf{T}\mathbf{G}_i$, $\mathbf{J} := \sum w_i \mathbf{G}_i^\mathsf{T}\mathbf{S}_i$, and $\mathbf{G}_i$ is a linear mapping from state vector $\mathbf{x}$ to an element-wise deformation representation, for example, deformation gradient in finite element methods. $\mathbf{G}_i$ only depends on the mesh topology and the rest-pose shapes of all elements. The constant C is irrelevant for optimization. For a fixed $\mathbf{p}$, the minimization of $\tilde{g}(\mathbf{x}, \mathbf{p})$ can be accomplished by finding $\mathbf{x}$ with a vanishing gradient, that is, $\nabla_\mathbf{x}\tilde{g}(\mathbf{x}, \mathbf{p}) = 0$. Computing the gradient yields some convenient simplifications (the traces disappear):

$$\nabla_\mathbf{x}\tilde{g}(\mathbf{x}, \mathbf{p}) = \frac{1}{h^2}\mathbf{M}(\mathbf{x}-\mathbf{y}) + \mathbf{L}\mathbf{x} - \mathbf{J}\mathbf{p}. \tag{6}$$

Equating the gradient to zero leads to the solution

$$\mathbf{x}^* = (\mathbf{M}/h^2 + \mathbf{L})^{-1}(\mathbf{J}\mathbf{p} + \mathbf{M}\mathbf{y}/h^2). \tag{7}$$

The matrix $\mathbf{M}/h^2 + \mathbf{L}$ is symmetric positive definite and therefore $\mathbf{x}^*$ is a global minimum (for fixed $\mathbf{p}$). The key computational advantage of Projective Dynamics is that $\mathbf{M}/h^2 + \mathbf{L}$ does not depend on $\mathbf{x}$, which allows us to precompute and repeatedly reuse its sparse Cholesky factorization to quickly solve for $\mathbf{x}^*$, which is the result after one local and global step. The local and global steps are repeated for a fixed number of iterations (typically 10 or 20).

4. METHOD

As described in the previous section, Projective Dynamics relies on the special type of elastic energies according to Equation (1). Let us now describe how Projective Dynamics can be interpreted as a quasi-Newton method. The first step is to compute the gradient of the objective $g(\mathbf{x})$ from Equation (3). The energy $E(\mathbf{x})$ used in this objective contains constrained minimization over the projection variables $\mathbf{p}_i \in \mathcal{M}_i$ (see Equation (1) and Equation (2)). Equivalently, we can interpret the $\mathbf{p}_i$ as functions of $\mathbf{x}$ realizing the projections, according to Equation (1). Nevertheless, the gradient

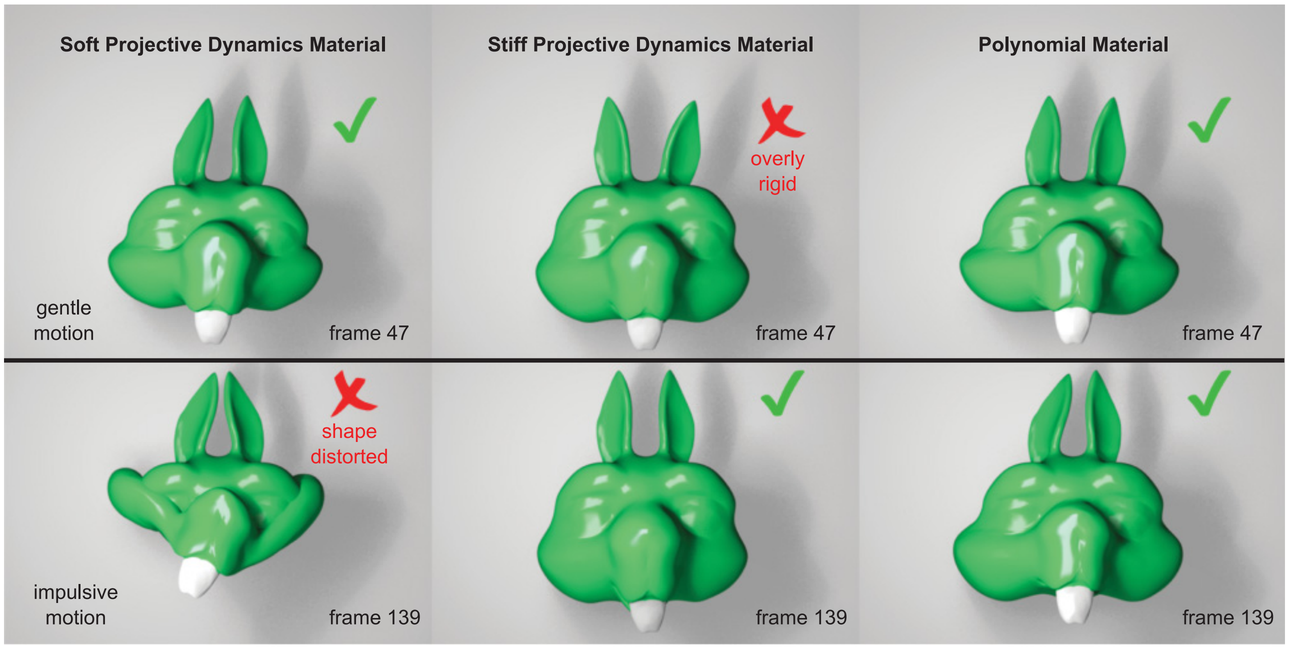

Fig. 2. Animating jiggly squirrel head. The squirrel head is driven by a gentle keyframed motion in the top row, and by a faster, impulsive motion in the bottom row. Soft Projective Dynamics material (left column) creates nice secondary motion but does not prevent large distortions of the shape. If we stiffen the Projective Dynamics material (middle column), we prevent the distortions but also kill the secondary motion. Our polynomial material $a(x) = \mu(x-1)^4$, $b(x) = 0$, $c(x) = 0$ (right column) achieves the desired effect of jiggling without large shape distortions.

$\nabla g(\mathbf{x})$ can still be computed easily—in fact, it is exactly equivalent to $\nabla_{\mathbf{x}}\tilde{g}(\mathbf{x}, \mathbf{p})$ from Equation (6), where we assumed that $\mathbf{p}$ is constant. This at first surprising fact has been observed in previous work [Chao et al. 2010; Bouaziz et al. 2012]. Intuitively, the reason is that if we infinitesimally perturb $\mathbf{x}$, its projection $\mathbf{p}_i(\mathbf{x})$ can move only in the tangent space of $\mathcal{M}_i$, and therefore, the differential $\delta\mathbf{p}_i(\mathbf{x})$ has no effect on $\delta\|\mathbf{x}-\mathbf{p}_i(\mathbf{x})\|^2$. As an intuitive explanation, imagine that $\mathbf{x}$ is a space shuttle projected to its closest point on Earth $\mathbf{p}_i(\mathbf{x})$; to first order, the distance of the space shuttle from Earth does not depend on the tangent motion $\delta\mathbf{p}_i(\mathbf{x})$. Please see the appendix for a more formal discussion. In summary, the gradient of Equation (3) is

$$\nabla g(\mathbf{x}) = \frac{1}{h^2}\mathbf{M}(\mathbf{x}-\mathbf{y}) + \mathbf{L}\mathbf{x} - \mathbf{J}\mathbf{p}(\mathbf{x}), \tag{8}$$

where $\mathbf{p}(\mathbf{x})$ is a function stacking all of the individual projections $\mathbf{p}_i(\mathbf{x})$. Newton's method would proceed by computing second derivatives, that is, the Hessian matrix $\nabla^2 g(\mathbf{x})$, and use it to compute a descent direction $-(\nabla^2 g(\mathbf{x}))^{-1}\nabla g(\mathbf{x})$. Note that definiteness fixes may be necessary to guarantee that this will really be a *descent* direction [Gast et al. 2015].

What happens if we modify Newton's method by using $\mathbf{M}/h^2+\mathbf{L}$ instead of the Hessian $\nabla^2 g(\mathbf{x})$? Simple algebra reveals

$$(\mathbf{M}/h^2+\mathbf{L})^{-1}\nabla g(\mathbf{x}) = \mathbf{x} - (\mathbf{M}/h^2+\mathbf{L})^{-1}(\mathbf{J}\mathbf{p}(\mathbf{x}) + \mathbf{M}\mathbf{y}/h^2).$$

However, the latter term is equivalent to the result of one iteration of the local/global steps of Projective Dynamics; see Equation (7). Therefore, $(\mathbf{M}/h^2+\mathbf{L})^{-1}\nabla g(\mathbf{x}) = \mathbf{x}-\mathbf{x}^*$, and we can interpret $\mathbf{d}_{\mathrm{PD}} := -(\mathbf{M}/h^2+\mathbf{L})^{-1}\nabla g(\mathbf{x})$ as a descent direction (this time there is no need for any definiteness fixes). Projective Dynamics can be therefore understood as a quasi-Newton method that computes the next iterate as $\mathbf{x}+\mathbf{d}_{\mathrm{PD}}$. Typically, quasi-Newton methods use line search techniques [Nocedal and Wright 2006] to find parameter $\alpha > 0$ such that $\mathbf{x}+\alpha\mathbf{d}_{\mathrm{PD}}$ reduces the objective as much as possible. However, with Projective Dynamics energies according to Equation (1), the optimal value is always $\alpha = 1$.

4.1 More General Materials

The interpretation of Projective Dynamics as a quasi-Newton method suggests that a similar optimization strategy might be effective for more general elastic potential energies. First, let us focus on isotropic materials, deferring the discussion of anisotropy to Section 4.4. The assumption of isotropy (material-space rotation invariance) together with world-space rotation invariance means that we can express the elastic energy density function Ψ as a function of singular values of the deformation gradient [Irving et al. 2004; Sifakis and Barbič 2012]. In the volumetric case, we have three singular values $\sigma_1, \sigma_2, \sigma_3 \in \mathbb{R}$, also known as "principal stretches." The function $\Psi(\sigma_1, \sigma_2, \sigma_3)$ must be invariant to any permutation of the principal stretches, for example, $\Psi(\sigma_1, \sigma_2, \sigma_3) = \Psi(\sigma_2, \sigma_1, \sigma_3)$ and so forth. Because directly working with such functions Ψ could be cumbersome, we instead use the Valanis-Landel hypothesis [Valanis and Landel 1967], which assumes that

$$\begin{aligned}\Psi(\sigma_1, \sigma_2, \sigma_3) &= a(\sigma_1) + a(\sigma_2) + a(\sigma_3) \\ &\quad + b(\sigma_1\sigma_2) + b(\sigma_2\sigma_3) + b(\sigma_1\sigma_3) + c(\sigma_1\sigma_2\sigma_3),\end{aligned} \tag{9}$$

where $a, b, c : \mathbb{R} \to \mathbb{R}$. Many material models can be written in the Valanis-Landel form, including linear corotated material [Sifakis and Barbič 2012], St. Venant-Kirchhoff, Neo-Hookean, and Mooney-Rivlin. The recently proposed spline-based materials [Xu et al. 2015] are also based on the Valanis-Landel assumption. How can we generalize Projective Dynamics to these types of materials? Invoking the quasi-Newton interpretation discussed previously, our method will minimize the objective g by performing descent along direction $\mathbf{d}(\mathbf{x}) := -(\mathbf{M}/h^2 + \mathbf{L})^{-1}\nabla g(\mathbf{x})$. The mass matrix $\mathbf{M}$ and step size h are defined as before, and computing $\nabla g(\mathbf{x})$ is straightforward. But how to define a matrix $\mathbf{L}$ for a given material model? This matrix can still have the form $\mathbf{L} := \sum w_i \mathbf{G}_i^\mathsf{T}\mathbf{G}_i$, but the question is how to choose the weights w_i. In Projective Dynamics, we assumed the weights are given as $w_i = V_i k_i$, where $V_i > 0$ is rest-pose volume of ith element, and $k_i > 0$ is a stiffness parameter provided by the user. In our case, the user instead specifies a material model according to Equation (9) from which we have to infer the appropriate k_i value. In the following, we drop the subscript i for ease of notation.

For linear materials (Hooke's law), stiffness is given as the second derivative of elastic energy. Therefore, it would be tempting to set k equal to the second derivative of Ψ at the rest pose (corresponding to $\sigma_1 = \sigma_2 = \sigma_3 = 1$), which evaluates to $a''(1) + 2b''(1) + c''(1)$, regardless of whether we differentiate with respect to σ_1, σ_2, or σ_3. Even though this method would produce suitable k for some materials (such as corotated elasticity), it does not work, for example, for a polynomial material defined as $a(x) = \mu(x-1)^4$, $b(x) = 0$, $c(x) = 0$. Already this relatively simple material can facilitate certain animation tasks, such as creating a cartoon squirrel head that jiggles, but does not overly distort its shape; see Figure 2. However, with this material, the second derivatives at $x = 1$ evaluate to zero regardless of the value of μ, which would lead to zero stiffness, which is obviously not a good approximation. The problem is that the second derivative takes into account only an infinitesimally small neighborhood of $x = 1$, that is, the rest pose. However, we need a *single* value of k that will work well in the entire range of deformations expected in our simulations. To capture this requirement, we define an interval $[x_\text{start}, x_\text{end}]$ where we expect our principal stretches to be. We consider the following stress function:

$$f(\sigma_1) = \left.\frac{\partial \Psi}{\partial \sigma_1}\right|_{\sigma_2=1,\sigma_3=1} = a'(\sigma_1) + 2b'(\sigma_1) + c'(\sigma_1), \tag{10}$$

and define our k as the slope of the best linear approximation of Equation (10) at $[x_\text{start}, x_\text{end}]$. Formally:

$$k := \operatorname*{argmin}_k \int_{x_\text{start}}^{x_\text{end}} (k(x-1) - f(x))^2 dx. \tag{11}$$

Note that due to the symmetry of the Valanis-Landel assumption, we would obtain exactly the same result if we differentiated with respect to σ_2 or σ_3 (instead of σ_1 as earlier). We study different choices of $[x_\text{start}, x_\text{end}]$ intervals in Section 5. In summary, the results are not very sensitive on the particular choice of x_start and x_end. The key fact is that regardless of the specific setting of x_start and x_end, spatial variations of μ are correctly taken into account; that is, softer and stiffer parts of the simulated object will have different μ coefficients (e.g., in our squirrel head we made the teeth more stiff). Even though all elements have the same $[x_\text{start}, x_\text{end}]$ interval, the resulting matrices $\mathbf{L}$ and $\mathbf{J}$ properly reflect the spatially varying stiffness.

Line search. With Projective Dynamics materials (Equation (1)), the line search parameter $\alpha = 1$ is always guaranteed to decrease the objective g (Equation (3)). Unfortunately, this is no longer true in our generalized quasi-Newton setting, where it is easy to find examples where $g(\mathbf{x}+\mathbf{d}(\mathbf{x})) > g(\mathbf{x})$; that is, taking a step of size one actually *increases* the objective. This can lead to erroneous energy accumulation, potentially resulting in catastrophic failure of the simulation ("explosions"), as shown in Figure 3. Fortunately, thanks to the fact that $\mathbf{M}/h^2 + \mathbf{L}$ is positive definite, $\mathbf{d}(\mathbf{x})$ is guaranteed to be a descent direction. Therefore, there exists $\alpha > 0$ such that $g(\mathbf{x} + \alpha\mathbf{d}(\mathbf{x})) \leq g(\mathbf{x})$ (unless we are already at a critical point $\nabla g(\mathbf{x}) = \mathbf{0}$, at which point the optimization is finished). In fact, we can ask for even more; that is, we can always find $\alpha > 0$ such that $g(\mathbf{x} + \alpha\mathbf{d}(\mathbf{x})) \leq g(\mathbf{x}) + \gamma\alpha\, \mathrm{tr}((\nabla g(\mathbf{x}))^\mathsf{T}\mathbf{d}(\mathbf{x}))$ for some constant $\gamma \in (0, 1)$ (we use $\gamma = 0.3$). This is known as the Armijo condition, which expresses the requirement of "sufficient decrease" [Nocedal and Wright 2006], preventing the line search algorithm from reducing the objective only by a negligible amount. Another requirement for robust line search is to avoid too small steps α even though they might satisfy the Armijo condition. We tested two possible strategies: (1) backtracking line search algorithm that satisfies only the Armijo condition and (2) line search algorithm that satisfies both the Armijo condition and the "curvature condition" (collectively known as "Wolfe conditions"). The details of our experiments can be found in Section 5; in summary, we found that both methods lead to comparable error reduction, but the backtracking line search is faster. In our final algorithm, we therefore use the backtracking line search. Specifically, we set the initial α to 1 and multiply it by 0.5 after every failed attempt. This line search strategy is used in all our experiments.

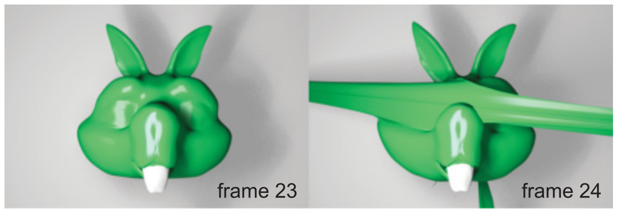

Fig. 3. Without line search, the squirrel head animation using our polynomial material (as in Figure 2) quickly becomes unstable.

ALGORITHM 1: Quasi-Newton Solver

```
1  $\mathbf{x}_1 := \mathbf{y}$; $g(\mathbf{x}_1) :=$ evalObjective($\mathbf{x}_1$)
2  for $k = 1, \ldots,$ numIterations do
3      $\nabla g(\mathbf{x}_k) :=$ evalGradient($\mathbf{x}_k$)
4      $\mathbf{d}(\mathbf{x}_k) := -(\mathbf{M}/h^2 + \mathbf{L})^{-1}\nabla g(\mathbf{x}_k)$
5      $\alpha := 2$
6      repeat
7          $\alpha := \alpha/2$
8          $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha\mathbf{d}(\mathbf{x}_k)$
9          $g(\mathbf{x}_{k+1}) :=$ evalObjective($\mathbf{x}_{k+1}$)
10     until $g(\mathbf{x}_{k+1}) \leq g(\mathbf{x}_k) + \gamma\alpha\, \mathrm{tr}((\nabla g(\mathbf{x}_k))^\mathsf{T}\mathbf{d}(\mathbf{x}_k))$;
11 end
```

Algorithm 1 summarizes the process of computing one frame of our simulation. The outer loop (lines 2–11) performs quasi-Newton iterations, and the inner loop (lines 6–10) implements the line search. What is the extra computational cost required to support more general materials? With Projective Dynamics energies (Equation (1)), we do not need the line search, because $\alpha = 1$ always works. Indeed, if we drop the line search from Algorithm 1, the algorithm becomes equivalent to a generalized local/global process, as discussed in Section 3 (which is unstable for non-Projective-Dynamics energies). Rejected line search attempts, that is, additional iterations of the line search, represent the main computational overhead of our method. Fortunately, we found that in practical simulations, the number of extra line search iterations is relatively small. For example, in the squirrel head example in Figure 2 using the polynomial material, we need only 4,280 line search iterations for the entire sequence with 400 frames, 10 quasi-Newton iterations per frame; that is, the *average* number of line search iterations per quasi-Newton iteration is only 1.07. Even though in most cases the full step ($\alpha = 1$) succeeds, the Armijo safeguard is essential for stability; if we drop it, the simulation can quickly explode, as shown in Figure 3.

4.2 Accelerating Convergence

The connection between Projective Dynamics and quasi-Newton methods allows us to take advantage of further mathematical optimization techniques. In this section, we discuss how to accelerate convergence of our method using L-BFGS (Limited-memory BFGS). The BFGS algorithm (Broyden-Fletcher-Goldfarb-Shanno) is one of the most popular general-purpose quasi-Newton methods; its key idea is to approximate the Hessian using curvature information calculated from previous iterates, that is, $\mathbf{x}_1, \ldots, \mathbf{x}_{k-1}$. The L-BFGS modification means that we will use only the most recent m iterates, that is, $\mathbf{x}_{k-m}, \ldots, \mathbf{x}_{k-1}$, the rationale being that too distant iterates are less relevant in estimating the Hessian at $\mathbf{x}_k$.

In Algorithm 1, the matrix $\mathbf{M}/h^2 + \mathbf{L}$ in line 4 can be interpreted as our initial approximation of the Hessian. This matrix is constant, which on one hand enables its prefactorization, but on the other hand, $\mathbf{M}/h^2 + \mathbf{L}$ may be far from the Hessian $\nabla^2 g(\mathbf{x}_k)$, which is the reason for slower convergence compared to Newton's method [Bouaziz et al. 2014]. L-BFGS allows us to develop a more accurate, state-dependent Hessian approximation, leading to faster convergence without too much computational overhead (in our experiments the overhead is typically less than 1% of the simulation time, see Table I). The key to fast iterations of L-BFGS is the fact that the progressively updated approximate Hessian $\mathbf{A}_k$ is not stored explicitly, which would require us to solve a new linear system $\mathbf{A}_k\mathbf{d}(\mathbf{x}_k) = -\nabla g(\mathbf{x}_k)$ each iteration, implying high computational costs. Instead, L-BFGS implicitly represents the *inverse* of $\mathbf{A}_k$, that is, linear operator $\mathbf{B}_k$, such that the desired descent direction can be computed simply as $\mathbf{d}(\mathbf{x}_k) = -\mathbf{B}_k\nabla g(\mathbf{x}_k)$. The linear operator $\mathbf{B}_k$ is not represented using a matrix (which would have been dense), but instead as a sequence of dot products, known as the L-BFGS two-loop recursion; see Algorithm 2. For a more detailed discussion of BFGS and its variants, we refer to Chapters 6 and 7 of Nocedal and Wright [2006].

Algorithm 2 requires us to provide an initial Hessian approximation $\mathbf{A}_0$, ideally such that the linear system $\mathbf{A}_0\mathbf{r} = \mathbf{q}$ can be solved efficiently (line 7). In our method, we use $\mathbf{M}/h^2 + \mathbf{L}$ as the initial Hessian approximation. At first, it may seem the initialization of the Hessian is perhaps not too important and the L-BFGS iterations quickly approach the exact Hessian. However, this intuition is not true. In Section 5, we experiment with different possible initializations of the Hessian and show that our particular choice of $\mathbf{M}/h^2 + \mathbf{L}$ outperforms alternatives such as Hessian of the rest pose and many others. In short, the reason is that the L-BFGS updates use only a few gradient samples, which provide only a limited amount of information about the exact Hessian. The appeal of the L-BFGS strategy is that it is very fast—the compute cost of the two for-loops in Algorithm 2 is negligible compared to the cost of solving the linear system in line 7 with our choice of $\mathbf{A}_0 = \mathbf{M}/h^2 + \mathbf{L}$. This is true even for high values of m. In other words, the linear solve using $\mathbf{M}/h^2 + \mathbf{L}$ (line 7) is still doing the "heavy lifting," while the L-BFGS updates provide an additional convergence boost at the cost of minimal computational overhead.

Upgrading our method with L-BFGS is simple: we only need to replace line 4 in Algorithm 1 with a call of Algorithm 2. Note that for $m = 0$, Algorithm 2 returns exactly the same descent direction as before, that is, $\mathbf{d}(\mathbf{x}_k) := -(\mathbf{M}/h^2 + \mathbf{L})^{-1}\nabla g(\mathbf{x}_k)$. What is the optimal m, that is, the size of the history window? Too small m will not allow us to unlock the full potential of L-BFGS. The main problem with too high m is not the higher computational cost of the two loops in Algorithm 2, but the fact that too distant iterates (such as $\mathbf{x}_{k-100}$) may contain information irrelevant for the Hessian at $\mathbf{x}_k$ and the result can be even worse than with a shorter window. We found that $m = 5$ is typically a good value in our experiments.

The recently proposed Chebyshev Semi-Iterative methods for Projective Dynamics [Wang 2015] can also be interpreted as a special type of quasi-Newton method that utilizes two previous iterates, that is, corresponding to $m = 2$. Indeed, in our experiments, L-BFGS with $m = 2$ exhibits a similar convergence rate as the Chebyshev method; see Figure 7 and further discussion in Section 5. Finally, we note that even though the Wolfe conditions are the recommended line search strategy for L-BFGS, we did not observe any significant convergence benefit compared to our backtracking strategy. However, evaluating the Wolfe conditions increases the computational cost per iteration, and therefore, we continue to rely on the backtracking strategy as described in Algorithm 1.

ALGORITHM 2: Descent Direction Computation with L-BFGS

```
1  $\mathbf{q} := -\nabla g(\mathbf{x}_k)$
2  for $i = k-1, \ldots, k-m$ do
3      $\mathbf{s}_i := \mathbf{x}_{i+1} - \mathbf{x}_i$; $\mathbf{t}_i := \nabla g(\mathbf{x}_{i+1}) - \nabla g(\mathbf{x}_i)$; $\rho_i := \mathrm{tr}(\mathbf{t}_i^\mathsf{T}\mathbf{s}_i)$
4      $\zeta_i := \mathrm{tr}(\mathbf{s}_i^\mathsf{T}\mathbf{q})/\rho_i$
5      $\mathbf{q} := \mathbf{q} - \zeta_i\mathbf{t}_i$
6  end
7  $\mathbf{r} := \mathbf{A}_0^{-1}\mathbf{q}$ // $\mathbf{A}_0$ is initial Hessian approximation
8  for $i = k-m, \ldots, k-1$ do
9      $\eta := \mathrm{tr}(\mathbf{t}_i^\mathsf{T}\mathbf{r})/\rho_i$
10     $\mathbf{r} := \mathbf{r} + \mathbf{s}_i(\zeta_i - \eta)$
11 end
12 $\mathbf{d}(\mathbf{x}_k) := \mathbf{r}$ // resulting descent direction
```

4.3 Collisions

A classical approach to enforcing nonpenetration constraints between deformable solids is to (1) detect collisions and (2) resolve them using temporarily instantiated repulsion springs, which bring the volume of undesired overlaps to zero [McAdams et al. 2011; Harmon et al. 2011]. However, in Projective Dynamics, the primary emphasis is on computational efficiency and therefore only simplified collision resolution strategies have been proposed by Bouaziz et al. [2014]. Specifically, Projective Dynamics offers two possible strategies. The first strategy is a two-phase method, where collisions are resolved in a separate postprocessing step using projections, similar to Position-Based Dynamics. The same strategy was employed also by Liu et al. [2013]. The drawback of this approach is the fact that collision projections are oblivious to elasticity and inertia of the simulated objects. The second approach used in Projective Dynamics is more physically realistic but introduces additional computational overhead. Specifically, temporarily instantiated repulsion springs are added to the total energy. This leads to physically realistic results, but the drawback is that the matrix $\mathbf{M}/h^2 + \mathbf{L}$ needs to be refactorized whenever the set of repulsion springs is updated—typically at the beginning of each frame.

Our quasi-Newton interpretation invites a new approach to collision response that is physically realistic but avoids expensive

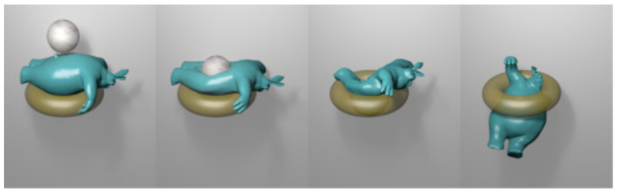

Fig. 4. Our method is capable of simulating complex collision scenarios, such as squeezing the Big Bunny through a torus. The Big Bunny uses corotated elasticity with $\mu = 5$ and $\lambda = 200$.

refactorizations. Specifically, for each interpenetration found by collision detection, we introduce an energy term $E_{\text{collision}}(\mathbf{x}) = ((\mathbf{Sx} - \mathbf{t})^{\mathsf{T}}\mathbf{n})^2$, where $\mathbf{S}$ is a selector matrix of the collided vertex, $\mathbf{t}$ is its projection on the surface, and $\mathbf{n}$ is the surface normal. This constraint pushes the collided vertex to the tangent plane. It is important to add this term to our total energy $E(\mathbf{x})$ only if the vertex is in collision or contact. Whenever the relative velocity between the vertex and the collider indicates separation, the $E_{\text{collision}}(\mathbf{x})$ term is discarded (otherwise, it would correspond to unrealistic "glue" forces). This is done once at the beginning of each iteration (just before line 3 in Algorithm 1). The rest of our algorithm (lines 6–10 of Algorithm 1) is unaffected by these updates; that is, the unilateral nature of the collision constraints is handled correctly without any further processing.

The key idea of our approach is to leverage the quasi-Newton approximation for collision processing. In particular, we account for the $E_{\text{collision}}(\mathbf{x})$ terms when evaluating the energy and its gradients, but we ignore their contributions to the $\mathbf{M}/h^2 + \mathbf{L}$ matrix. This means that we form a somewhat more aggressive approximation of the Hessian, with the benefit that the system matrix will never need to be refactorized. The line search process (lines 6–10 in Algorithm 1) guarantees that energy will decrease in spite of this more aggressive approximation. The only tradeoff we observed in our experiments is that the number of line search iterations may increase, which is a small cost to pay for avoiding refactorizations. We observed that even in challenging collision scenarios, such as when squeezing a Big Bunny through a torus, the approach behaves robustly and successfully resolves all collisions; see Figure 4.

4.4 Anisotropy

Our numerical methods, including the L-BFGS acceleration, can be directly applied also to anisotropic material models. We verified this on an elastic cube model with corotated base material ($\mu = 10, \lambda = 100$, referring to the notation of Sifakis and Barbič [2012]) enhanced with the additional anisotropic stiffness term $\frac{\kappa}{2}(\|\mathbf{Fd}\| - 1)^2$, where $\mathbf{F}$ is the deformation gradient and $\mathbf{d}$ is the (rest-pose) direction of anisotropy. This corresponds to the directional reinforcement of the material that is common, for example, in biological soft tissues containing collagenous fibers. The result of our method with $\kappa = 50$ can be seen in Figure 5.

5. RESULTS

Our method supports standard elastic materials, such as corotated linear elasticity, St. Venant-Kirchhoff, and the Neo-Hookean model; see Figure 1. None of these materials is supported by Projective Dynamics (note that Projective Dynamics supports a special subclass of corotated linear materials, specifically ones with $\lambda = 0$). Our method also supports the recently introduced spline-based materials proposed by Xu et al. [2015], as shown in Figure 1 and Figure 6.

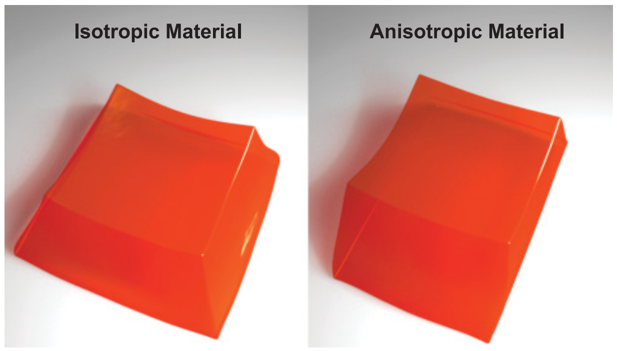

Fig. 5. Dropping an elastic cube on the ground. Left: Deformation using isotropic elasticity (linear corotated model). Right: The result after adding anisotropic stiffness.

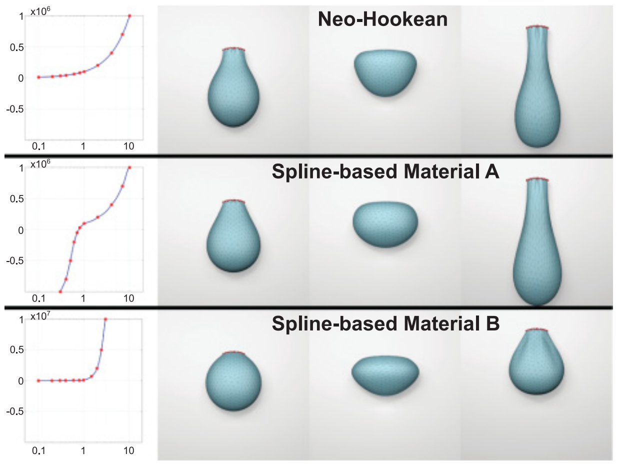

Fig. 6. Elastic sphere with spline-based materials [Xu et al. 2015], simulated using our method. Spline-based material A is a modified Neo-Hookean material that resists compression more; material B is a modified Neo-Hookean material that resists tension more. The strain-stress curves are shown on the left.

Table I reports our testing scenarios and compares the runtime of our method with Newton's method, both executed on an Intel i7-4910MQ CPU at 2.90GHz. All scenarios are produced with a fixed timestep of 1/30 seconds. Because Newton's method is not guaranteed to work with indefinite Hessians, we employ the standard definiteness fix [Teran et al. 2005]; that is, we project the Hessian of each element to its closest positive definite component. We found that this method works better than other definiteness fixes, such as adding a multiple of the identity matrix [Martin et al. 2011], which affects the *entire* simulation even if there are just a few problematic elements. The approximately 100 times faster runtime of one iteration of our method compared to one iteration of Newton's method is due to the following facts: (1) we use precomputed sparse Cholesky factorization, because our matrix $\mathbf{M}/h^2 + \mathbf{L}$ is constant; (2) the size of our matrix is $n \times n$, whereas the Hessian used in Newton's method is a $3n \times 3n$ matrix—that is, the x, y, z coordinates are no longer decoupled; (3) the computation of SVD derivatives, necessary to evaluate the Hessians of materials based on principal stretches [Xu et al. 2015], is expensive. Note that our method is also simpler to implement, as no SVD derivatives or definiteness fixes are necessary.

Table I. Performance of All Testing Senarios

				Our Method (10 Iterations)				Newton (1 Iteration)	
Model	#Ver.	#Ele.	Material Model	Line Search Iterations	L-BFGS Overhead	Per-Frame Time	Relative Error	Per-Frame Time	Relative Error
Thin sheet	660	1932	Polynomial	10.8	0.026ms	4.4ms	2.7×10^{-8}	184ms	8.8×10^{-4}
Sphere	889	1821	Spline-based A	24.5[†]	0.155ms	21.2ms	2.7×10^{-7}	188ms	6.9×10^{-4}
Sphere	889	1821	Spline-based B	21.8	0.156ms	19.7ms	6.9×10^{-6}	187ms	2.5×10^{-4}
Shaking bar	574	1647	Corotated	10.1	0.193ms	7.2ms	1.6×10^{-4}	171ms	4.4×10^{-3}
Ditto	1454	4140	Neo-Hookean	11.7	0.203ms	17.8ms	3.0×10^{-5}	305ms	1.6×10^{-3}
Hippo	2387	8406	Corotated	11.9	0.555ms	40.6ms	2.2×10^{-3}	640ms	3.7×10^{-2}
Twisting bar	3472	10441	Neo-Hookean	10.6	0.945ms	45.6ms	9.4×10^{-5}	681ms	7.9×10^{-3}
Cloth	6561	32160	Mass-Springs	10.0	1.20ms	42.3ms	9.3×10^{-4}	798ms	1.2×10^{-2}
Big Bunny	6308	26096	Corotated	49.2[‡]	2.19ms	623ms	9.8×10^{-2}	2700ms	2.8×10^{-1}
Squirrel	8395	23782	Polynomial	10.7	1.41ms	153ms	8.3×10^{-8}	2400ms	9.1×10^{-6}
Squirrel	33666	125677	Polynomial	10.5	6.38ms	706ms	1.5×10^{-5}	15800ms	5.4×10^{-5}

In all examples, we execute 10 iterations of our method per frame, accelerated with L-BFGS with history window $m = 5$. Newton's method uses one iteration per frame. The "line search iterations" reports the average number of line search iterations per frame. The "L-BFGS overhead" is the runtime overhead of L-BFGS, that is, timing of Algorithm 2 without line 7 ($m = 5$). The reported per-frame time for our method accounts for *all 10 iterations*. One iteration of our method is approximately 100 times faster than one iteration of Newton's method. We use 10 iterations of our method, which reduces the error more than one iteration of Newton's method while being about 10 times faster. [†]The higher number of line search iterations is due to the high nonlinearity of the spline-based materials and large deformations of the sphere. [‡]In this case, the higher number of line search iterations is caused by nonlinearities due to collisions (Section 4.3).

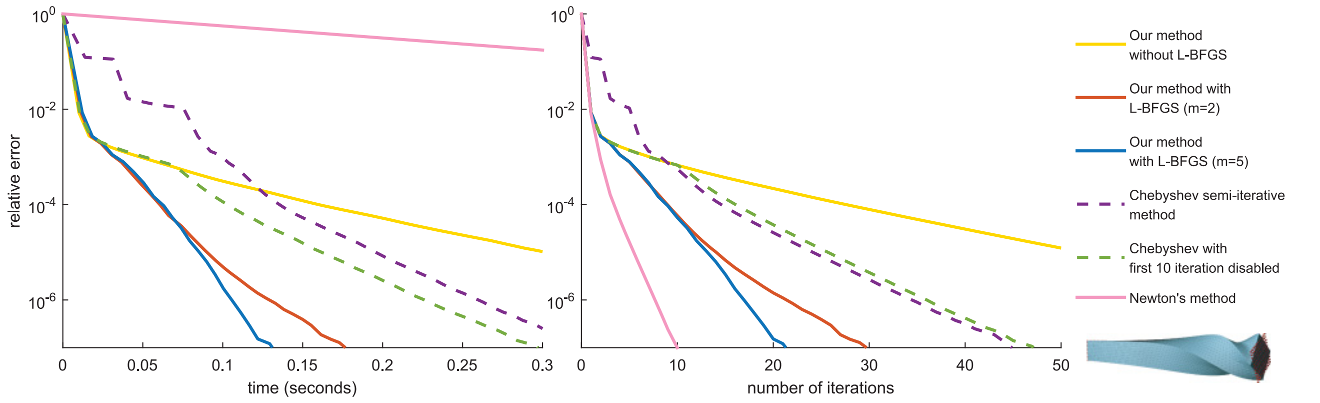

Fig. 7. Convergence of our method with different L-BFGS history settings, compared to Chebyshev Semi-Iterative method and Newton's method (baseline). The model is "Twisting bar" with Neo-Hookean elasticity, which is a representative example of large deformations with the nonlinear material model. For consistency, we use the exact same model in our following experiments.

Comparison to Chebyshev Semi-Iterative method. We compared the convergence of our method with various lengths of the L-BFGS window to the recently introduced Chebyshev Semi-Iterative method [Wang 2015]. We also plot results obtained with Newton's method as a baseline; see Figure 7.

Even though the Chebyshev method was originally proposed only for Projective Dynamics energies, our generalization to arbitrary materials is compatible with the Chebyshev Semi-Iterative acceleration; see Algorithm 3. Algorithm 3 computes a descent direction that can be used in line 4 of Algorithm 1. As discussed by Wang [2015], the Chebyshev acceleration should be disabled during the first S iterations, where the recommended value is $S = 10$. Another parameter that is essential for the Chebyshev method is an estimate of spectral radius ρ, which is calculated from training simulations [Wang 2015]. This parameter must be estimated carefully, because underestimated ρ can lead to the Chebyshev method producing *ascent* directions (as opposed to descent directions). Without line search, the ascent directions manifest themselves as oscillations [Wang 2015]. For the purpose of comparisons, we implemented the Chebyshev method with a direct solver, which is the fastest method on the CPU [Wang 2015].

ALGORITHM 3: Descent Direction Computation Using Chebyshev Semi-Iterative Method [Wang 2015]

```
1 // S ... Chebyshev disabled for the first S iterates, default S = 10
2 // ρ ... approximated spectral radius
3 $\mathbf{q} := -(\mathbf{M}/h^2 + \mathbf{L})^{-1}\nabla g(\mathbf{x}_k)$
4 $\hat{\mathbf{x}}_{k+1} := \mathbf{x}_k + \mathbf{q}$
5 if $k < S$ then $\omega_{k+1} := 1$;
6 if $k = S$ then $\omega_{k+1} := 2/(2-\rho^2)$;
7 if $k > S$ then $\omega_{k+1} := 4/(4-\rho^2\omega_k)$;
8 $\mathbf{d}(\mathbf{x}_k) := \omega_{k+1}(\hat{\mathbf{x}}_{k+1} - \mathbf{x}_{k-1}) + \mathbf{x}_{k-1} - \mathbf{x}_k$
```

We compare the convergence of all methods using relative error, defined as

$$\frac{g(\mathbf{x}_k) - g(\mathbf{x}^*)}{g(\mathbf{x}_0) - g(\mathbf{x}^*)}, \tag{12}$$

where $\mathbf{x}_0$ is the initial guess (we use $\mathbf{x}_0 := \mathbf{y}$ for all methods), $\mathbf{x}_k$ is the kth iterate, and $\mathbf{x}^*$ is the exact solution computed using Newton's method (iterated until convergence). The decrease of relative error

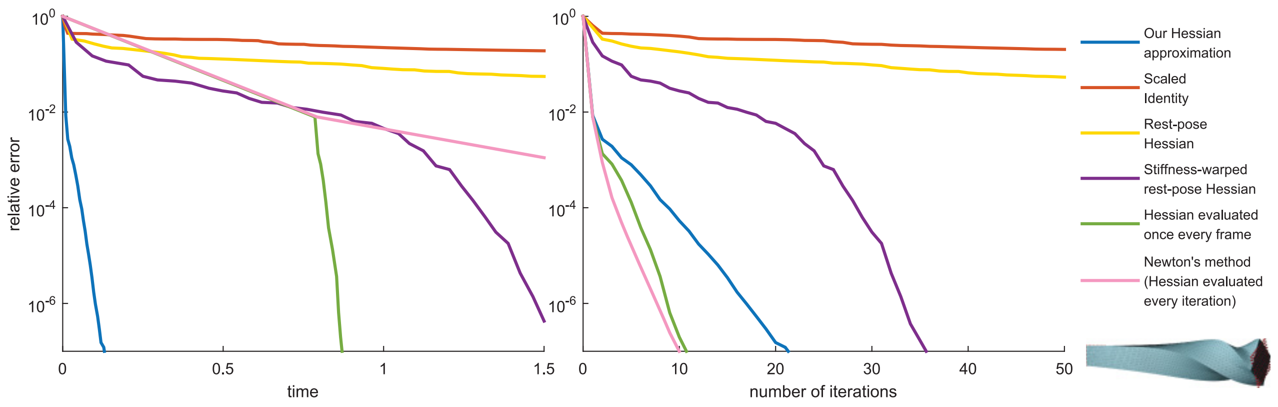

Fig. 8. Convergence comparison of L-BFGS methods (all using $m = 5$) initialized with different Hessian approximations, along with Newton's method (baseline). The model is "Twisting bar" with Neo-Hookean elasticity.

for one example frame is shown in Figure 7, where all methods are using the backtracking line search outlined in Algorithm 1. As expected, descent directions computed using Newton's method are the most effective ones, as can be seen in Figure 7 (right). However, each iteration of Newton's method is computationally expensive, and therefore other methods can realize faster error reduction with respect to computational time, as shown in Figure 7 (left). All of the remaining methods are based on the constant Hessian approximation $\mathbf{M}/h^2 + \mathbf{L}$, which leads to much faster convergence. Out of these methods, classical Projective Dynamics converges the slowest. The Chebyshev Semi-Iterative method improves the convergence; we also confirmed that disabling the Chebyshev method during the first 10 iterations indeed helps, as recommended by Wang [2015]. Our method aided with L-BFGS improves convergence even further. Already with $m = 2$ (where m is the size of the history window), we obtain slightly faster convergence than with the Chebyshev method. One reason is that it is not necessary to disable L-BFGS in the first several iterates, because L-BFGS is effective as soon as the previous iterates become available. Also, we do not have to estimate the spectral radius, which is required by the Chebyshev method. With L-BFGS, we can also increase the history window, for example, to $m = 5$, obtaining even more rapid convergence.

L-BFGS with different initial Hessian estimates. Our method can be interpreted as providing a particularly good initial estimate of the Hessian for L-BFGS. Therefore, it is important to compare to other possible Hessian initializations. In a general setting, Nocedal and Wright [2006] recommend to bootstrap L-BFGS using a scaled identity matrix:

$$\mathbf{A}_0 := \frac{\operatorname{tr}(\mathbf{s}_{k-1}^{\mathsf{T}}\mathbf{y}_{k-1})}{\operatorname{tr}(\mathbf{y}_{k-1}^{\mathsf{T}}\mathbf{y}_{k-1})}\mathbf{I}. \tag{13}$$

We experimented with this approach, but we found that our choice $\mathbf{A}_0 := \mathbf{M}/h^2 + \mathbf{L}$ leads to much faster convergence, trumping the computational overhead associated with solving the prefactorized system $\mathbf{A}_0\mathbf{r} = \mathbf{q}$ (see Figure 8, blue graph).

Another possibility would be to set $\mathbf{A}_0$ equal to the rest-pose Hessian (formally, $\mathbf{A}_0 := \mathbf{M}/h^2 + \nabla^2\mathbf{g}(\mathbf{x}_0)$), which is also a constant matrix that can be prefactorized. As shown in Figure 8 (yellow graph), this is a slightly better approximation than scaled identity, but still not very effective. This is because the actual Hessian depends on world-space rotations of the model, deviating significantly from the rest-pose Hessian. Figure 9 shows an example illustrating the drawbacks of the rest-pose Hessian. Configuration 1 shows an elastic cube released from a slightly stretched state.

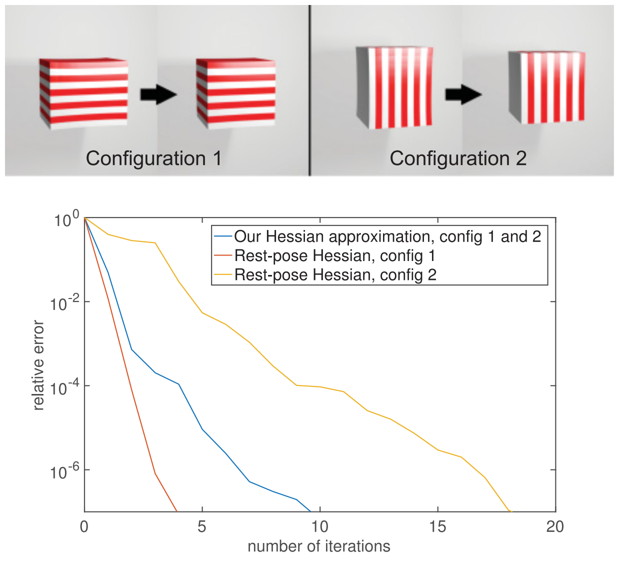

Fig. 9. Convergence comparison of L-BFGS methods with different configurations. Configuration 1 is the simulation of an elastic cube with Neo-Hookean elasticity, released from a horizontally stretched pose, close to the rest pose. Configuration 2 is the same configuration rotated by 90 degrees. Using the rest-pose Hessian as an initial Hessian approximation does not work well in Configuration 2. Our method is rotation invariant and therefore performs equally well in both configurations.

In this configuration, setting $\mathbf{A}_0$ to the rest-pose Hessian results in faster error reduction than our method (red graph), because the initial configuration is close to the rest pose and therefore the exact Hessian is close to the rest-pose Hessian. Unfortunately, when we rotate the initial configuration by 90 degrees (Configuration 2), the rest-pose Hessian becomes ineffective, as it is far away from the exact Hessian (yellow graph). Our Hessian approximation is invariant to rigid body transformations and therefore leads to the same error reduction in both Configurations 1 and 2 (blue graph). To further analyze this effect, we also computed the condition number of $\mathbf{A}^{-1}\nabla^2\mathbf{g}(\mathbf{x})$, where $\mathbf{A}$ is an approximate Hessian. The condition numbers reported in Table 8 confirm this observation.

Another interpretation of our Hessian approximation can be derived from the energy density function of the Neo-Hookean material

Table II. Condition Number of $\mathbf{A}^{-1}\nabla^2\mathbf{g}(\mathbf{x})$ in Figure 9

A	Rest-Pose Hessian		Our Hessian Approximation	
Configuration	1	2	1	2
Condition number	2.45	45.5	9.94	9.94

Condition number of $\mathbf{A}^{-1}\nabla^2\mathbf{g}(\mathbf{x})$ in Configurations 1 and 2 as in Figure 9, where $\nabla^2\mathbf{g}(\mathbf{x})$ is the exact Hessian matrix evaluated at the beginning of the frame and $\mathbf{A}$ is an approximate Hessian, computed (1) at the rest pose and (2) using our method.

[Sifakis and Barbič 2012]:

$$\Psi(\mathbf{F}) = \frac{\mu}{2}(\mathrm{tr}(\mathbf{F}^T\mathbf{F}) - 3) - \mu \log(\det(\mathbf{F})) + \frac{\lambda}{2}\log^2(\det(\mathbf{F})), \quad (14)$$

where $\mathbf{F}$ is the deformation gradient and μ and λ are Lamé coefficients. In this case, our Hessian approximation corresponds to the first term, that is, $\mathrm{tr}(\mathbf{F}^T\mathbf{F})$, which is indeed rotation invariant. The rest-pose Hessian is not rotation invariant and thus produces worse approximation of the exact Hessian as shown in Figure 9.

This issue of the rest-pose Hessian was observed by Müller et al. [2002], who proposed per-vertex *stiffness warping* as a possible remedy. Per-vertex stiffness warping still allows us to leverage prefactorization of the rest-pose Hessian and results in better convergence than pure rest-pose Hessian; see Figure 8 (purple graph). However, per-vertex stiffness warping may introduce ghost forces, because stiffness warping uses different rotation matrices for each vertex, which means that internal forces in one element no longer have to sum to zero. The ghost forces disappear in a fully converged solution; however, this would require a prohibitively high number of iterations.

Yet another possibility is to completely re-evaluate the Hessian at the beginning of each frame. This requires refactorization; however, the remaining 10 (or so) iterations can reuse the factorization, relying only on L-BFGS updates. When measuring convergence with respect to the number of iterations, this approach is very effective, as shown in Figure 8 (right, green graph). However, the cost of the initial Hessian factorization is significant, as can be seen in Figure 8 (left, green graph). Our method uses the same Hessian factorization for all frames, avoiding the per-frame factorization costs while featuring excellent convergence properties; see Figure 8 (blue graph).

The overhead of per-frame Hessian factorizations can be mitigated by carefully scheduled Hessian updates. In particular, the Hessian can be reused for multiple subsequent frames if the state is not changing too much [Deuflhard 2011]. Assuming the corotated elastic model, Hecht et al. [2012] push this idea even further by proposing a warp-canceling form of the Hessian that allows not only for *temporal* schedule but also for *spatially* localized updates. Specifically, a nested dissection tree allows for recomputing only parts of the mesh, which is particularly advantageous in situations where only a small part of the object is undergoing large deformations. However, the updates are still costly, and the frequency of the updates depends on the simulation. Similarly to per-vertex stiffness warping, insufficiently frequent updates may produce ghost forces and consequent instabilities. This can be a problem when simulating quickly moving elastic objects. To illustrate this, in Figure 10, we show a simulation of shaking an elastic bar. Even if we schedule the Hessian updates every other frame and recompute the entire domain, this method still generates too large ghost forces and becomes unstable. In contrast, our method remains stable and does not require *any* runtime Hessian updates.

Comparison to Projective Dynamics. One possible alternative to our method would be to apply regular Projective Dynamics with additional strain-limiting constraints [Bouaziz et al. 2014], enabling us to construct piece-wise linear approximations of the strain-stress curves of more general materials. We tried to use this approach to approximate the polynomial material ($a(x) = \mu(x-1)^4, b(x) = 0, c(x) = 0$) discussed in Section 4.1; see Figure 11. Even though we obtain similar overall behavior, there are two types of artifacts associated with this approximation. First, the strain-limiting constraints introduce damping when they are *not* activated. This is because the projection terms still exist in our constant matrix $\mathbf{M}/h^2 + \mathbf{L}$; if the strain limiting is not activated, the deformation gradients project to their current values, which produces the undesired damping. The second problem is due to the nonsmooth nature of the piece-wise linear approximation; that is, the stiffness of the simulated object is abruptly changed when the strain-limiting constraints become activated. As shown in the accompanying video, our method avoids both of these issues.

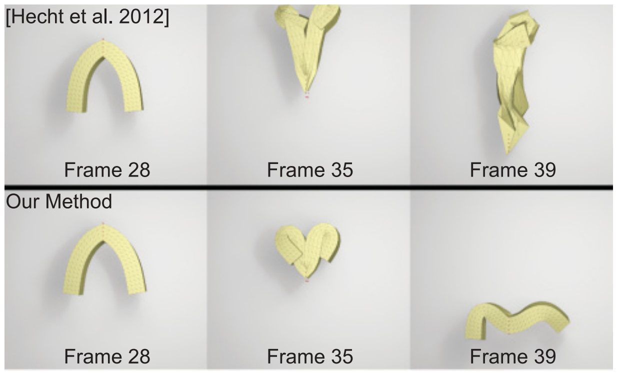

Fig. 10. Simulation of a bar with corotated elasticity, constrained in the middle and rapidly shaken. The method of Hecht et al. [2012] with full Hessian updates every *other* frame explodes due to large ghost forces (top). Our method does not introduce any ghost forces and remains stable (bottom).

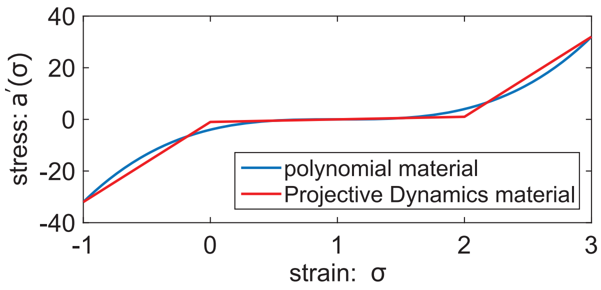

Fig. 11. The strain-stress curve of a polynomial material can be approximated piece-wise linearly with two Projective Dynamics constraints.

The L-BFGS acceleration also benefits simulations that use only Projective Dynamics materials (Equation (1)). The most elementary examples of these materials are mass-spring systems. In Figure 12, we can see that the L-BFGS acceleration applied to a mass-spring system simulation results in more realistic wrinkles with negligible computational overhead; see Table I.

Choice of L-BFGS history window size. In order to find the best history window size (m), we experimented with different values of m; see Figure 13. Too large m takes into account too distant iterates, which can lead to worse approximation of the Hessian. In Figure 13, we see that the optimal value is $m = 5$, which is also our recommended default setting. However, it is comforting that the algorithm is not particularly sensitive to the setting of m—even large values such as $m = 100$ produce only slightly worse

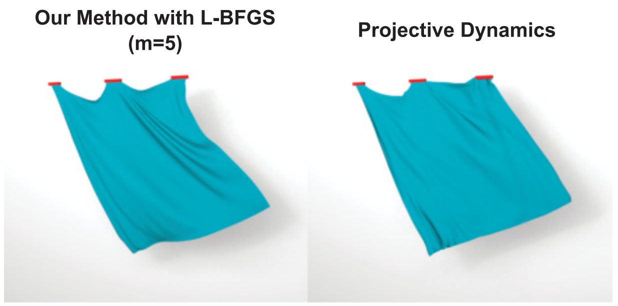

Fig. 12. Mass-spring system simulation using our method with L-BFGS (left) and without, that is, using pure Projective Dynamics (right). The L-BFGS acceleration results in more realistic wrinkles.

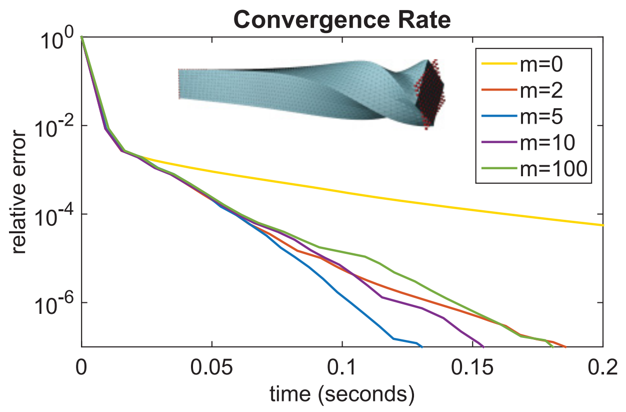

Fig. 13. Comparison of L-BFGS convergence rate with different history window sizes (m).

convergence. We also observed that in scenarios with frequent collisions, the history becomes less useful. In these cases, reducing the window size according to the number of newly instantiated collision constraints may be beneficial. In Figure 7, we also notice that the convergence rate of the Chebyshev method is similar to our method with L-BFGS using $m = 2$. We believe this is not a coincidence, because the Chebyshev method uses *two* previous iterates, just like L-BFGS with $m = 2$.

Line search conditions. The purpose of a line search algorithm is to ensure a sufficient decrease in a given descent direction. Line 4 in Algorithm 1 is known as the Armijo condition. This condition prevents overshooting, but it is not enough to ensure that the algorithm keeps making reasonable progress, because it is satisfied for all sufficiently small values of α. In order to rule out unacceptably small steps, the popular Wolfe conditions use an additional requirement: $\mathrm{tr}((\nabla g(\mathbf{x}+\alpha\mathbf{d}(\mathbf{x})))^{\mathsf{T}}\mathbf{d}(\mathbf{x})) \geq \gamma_2\, \mathrm{tr}((\nabla g(\mathbf{x}))^{\mathsf{T}}\mathbf{d}(\mathbf{x}))$, where $\gamma_2 \in (\gamma, 1)$, and γ is the constant in line 4 of Algorithm 1. This requirement is known as a "curvature condition." Intuitively, this condition requires the gradient at the new iterate to be sufficiently "flat," that is, close to a critical point.

We implemented two algorithms: (1) backtracking line search starting with $\alpha = 1$ and using only the Armijo condition and (2) line search using both of the Wolfe conditions (according to Algorithm 3.5 and 3.6 in Nocedal and Wright [2006]). The line search step sizes for one example frame are compared in Figure 14. With descent directions computed with our method, we observed that 1 is an excellent initial guess that usually satisfies both of the Wolfe conditions. In these cases, both algorithms return $\alpha = 1$. In some

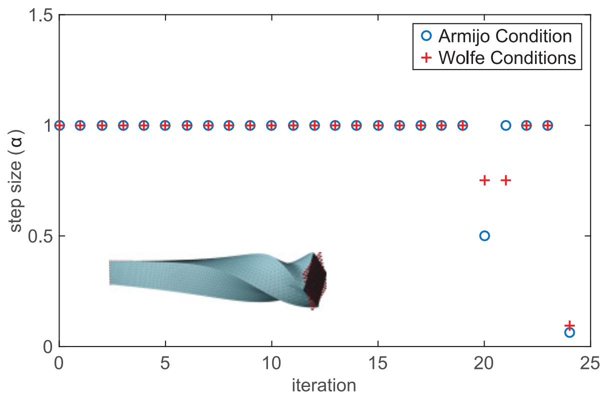

Fig. 14. Step size of each quasi-Newton iteration in the twisting bar example. The blue circles are step sizes chosen by backtracking line search with the Armijo condition. The red crosses are step sizes chosen by line search satisfying the Wolfe conditions.

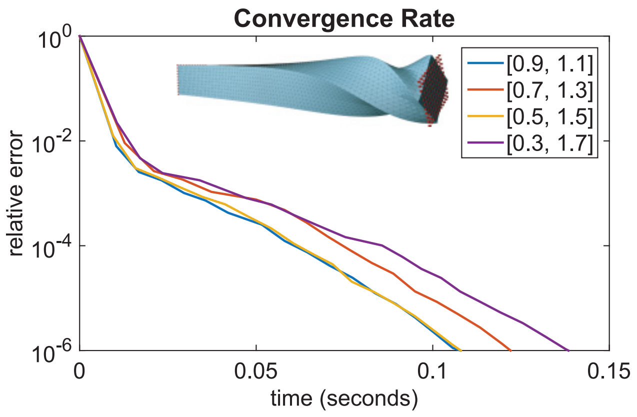

Fig. 15. The convergence rate for different stiffness parameters chosen from different $[x_{\mathrm{start}}, x_{\mathrm{end}}]$ intervals.

iterations, for example, in iteration 20 in Figure 14, the curvature condition enforces a larger step than the backtracking line search. However, at the beginning of iteration 20, the relative error has been already reduced to 10^{-7}, and the different step size does not have a significant effect on the result. We note the Wolfe conditions are usually recommended when using L-BFGS methods [Nocedal and Wright 2006] initialized with a scaled identity matrix, but this can be a poor initial guess. Our Hessian approximation provides a better initial guess and therefore careful line search becomes less critical. In practice, we observed that both line search approaches lead to comparable error reduction when using our Hessian approximation, and therefore we recommend the computationally less expensive backtracking line search strategy.

Choice of stiffness parameters. As discussed in Section 4.1, we use Equation (11) to define our stiffness parameter k as the slope of the best linear approximation of Equation (10) at $\in [x_{\mathrm{start}}, x_{\mathrm{end}}]$. What is the best $[x_{\mathrm{start}}, x_{\mathrm{end}}]$ interval to use? In the limit, with $[x_{\mathrm{start}}, x_{\mathrm{end}}] \to [1, 1]$, our k would converge to the second derivative. However, a finite interval $[x_{\mathrm{start}}, x_{\mathrm{end}}]$ guarantees that our k is meaningful even for materials such as the polynomial material $a(x) = \mu(x-1)^4$, $b(x) = 0$, $c(x) = 0$; in this case, we obtain a k that depends linearly on μ. We argue that the convergence of our algorithm is not very sensitive to a particular choice of the $[x_{\mathrm{start}}, x_{\mathrm{end}}]$ interval. In Figure 15, we show convergence graphs of a twisting

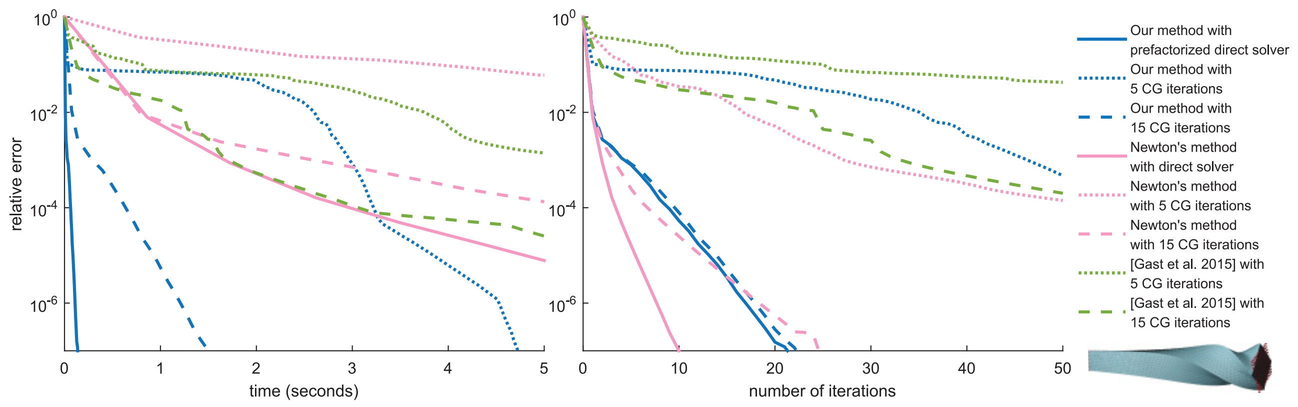

Fig. 16. Convergence comparison of various methods using sparse direct solvers and conjugate gradients. The model is "Twisting bar" with Neo-Hookean elasticity.

bar with Neo-Hookean material using different intervals to compute the stiffness parameter k. Although Neo-Hookean material is highly nonlinear, the convergence rates for different interval choices are quite similar. Therefore, we decided not to investigate more sophisticated strategies and we set $x_{\text{start}} = 0.5, x_{\text{end}} = 1.5$ in all of our simulations.

Comparison with iterative solvers. Sparse iterative solvers do not require expensive factorizations and are therefore attractive in interactive applications. A particularly popular iterative solver is the Conjugate Gradient method (CG) [Hestenes and Stiefel 1952]. An additional advantage is that CG can be implemented in a matrix-free fashion, that is, without explicitly forming the sparse system matrix. Gast et al. [2015] further accelerate the CG solver used in Newton's method by proposing a CG-friendly definiteness fix. Specifically, the CG iterations are terminated whenever the maximum number of iterations is reached or indefiniteness of the Hessian matrix is detected.

While iterative methods can be the only possible choice in high-resolution simulations (e.g., in scientific computing), in real-time simulation scales, sparse direct solvers with precomputed factorization are hard to beat, as we show in Figure 16. Specifically, we test Newton's method with linear systems solved using CG with five and 15 iterations, using a Jacobi preconditioner. Even with 15 CG iterations, the accuracy is still not the same as with the direct solver, while the computational cost becomes high. If we use only five CG iterations, the linear system solving time improves, but the convergence rate suffers because the descent directions are not sufficiently effective. The method of Gast et al. [2015] initially outperforms Newton with CG; however, the convergence slows down in subsequent iterations. We also tried to apply CG to our method, in lieu of the direct solver. With 15 CG iterations, the convergence is competitive; however, the CG solver is slower.

A carefully chosen preconditioner usually helps the convergence of a CG solver. Figure 17 shows an example of the effect of different preconditioners. The green graph is L-BFGS initialized with the Hessian matrix evaluated at the beginning of every frame, solved by a direct solver. This Hessian approximation provides a very nice initial guess for L-BFGS, but its evaluation and factorization are too expensive to execute once per frame. The two yellow graphs use the same Hessian approximation as the green graph but are solved using preconditioned conjugate gradients (PCGs) with five and 15 iterations, using incomplete Cholesky factorization of the rest-pose Hessian as a preconditioner. Compared with the green

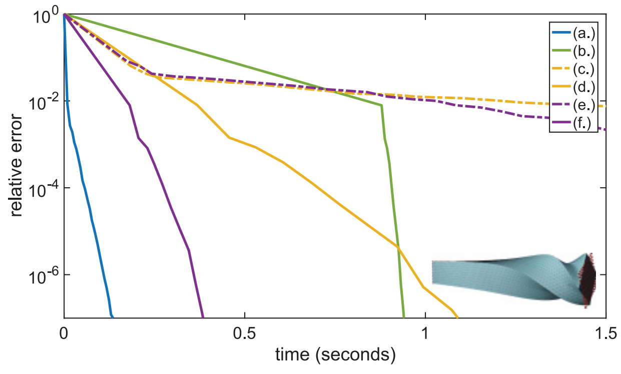

Fig. 17. Comparison to preconditioned conjugate gradients (PCGs) with different preconditioners, tested on the "Twisting bar" example with Neo-Hookean elasticity. Experiment (a) is L-BFGS using our Hessian approximation as the initial guess. The rest of the experiments (b–f) are L-BFGS using the Hessian matrix evaluated at the beginning of each frame solved by the following options: (b) direct solver; (c) five PCG iterations with incomplete Cholesky (ichol) of the rest-pose Hessian; (d) 15 PCG iterations with ichol of the rest-pose Hessian; (e) five PCG iterations with ichol of our Hessian approximation; (f) 15 PCG iterations with ichol of our Hessian approximation.

graph, we can see that the PCG solver is more efficient especially at the first several iterations, since it does not require the expensive factorization of the Hessian matrix. However, the convergence starts to slow down at later iterates. The two purple graphs are using the exact same configuration as the yellow graphs, but use incomplete Cholesky factorization of our Hessian approximation, $\mathbf{M}/h^2 + \mathbf{L}$, as a preconditioner. We can see that our Hessian approximation is a better preconditioner than the rest-pose Hessian.

Robustness. We demonstrate that our proposed extensions to more general materials and the L-BFGS solver upgrade do not compromise simulation robustness. In Figure 18, we show an elastic hippo that recovers from an extreme (randomized) deformation with many inverted elements. Specifically, the hippo model uses L-BFGS with $m = 5$ and corotated linear elasticity with $\mu = 20$ and $\lambda = 100$ (note that Projective Dynamics supports only corotated materials with $\lambda = 0$).

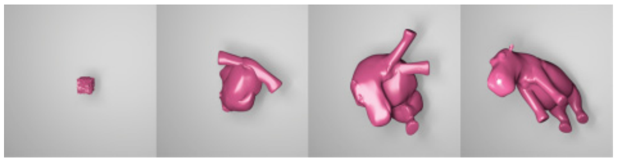

Fig. 18. Our method is robust despite extreme initial conditions: a randomly initialized hippo returns back to its rest pose. This example does not contain any explicit inversion handling constraints, only the standard penalization of inverted elements due to corotated elasticity.

6. LIMITATIONS AND FUTURE WORK

Our method is currently limited only to hyperelastic materials satisfying the Valanis-Landel assumption. Even though this assumption covers many practical models, including the recently proposed spline-based materials [Xu et al. 2015], it would be interesting to study the further generalization of our method. Perhaps even more interesting would be to remove the assumption of hyperelasticity. Can we develop fast algorithms for simulating non-hyperelastic materials, including effects such as relaxation, creep, and hysteresis [Bargteil et al. 2007]? Our method assumes linear FEM; it would be a great topic to find a good Hessian approximation to nonlinear shape functions, such as Quadratic Bézier Finite Elements [Bargteil and Cohen 2014]. Inspired by the recent work of Wang [2015], we would like to explore GPU implementations of physics-based simulations. Our current method is derived from the Implicit Euler time integration method and therefore inherits its artificial damping drawbacks. We experimented with Implicit Midpoint—a symplectic integrator that does not suffer from this problem. However, we found that Implicit Midpoint is much less stable. In the future, we would like to explore fast numerical solvers for symplectic yet stable integration methods. Finally, we plan to investigate specific physics-based applications that require *both* high accuracy and speed, such as interactive surgery simulation.

Collision detection and response is a challenging problem. The classical model of repulsion springs [McAdams et al. 2011], which we adopted in our implementation, is analogous to an active set method that adds/removes constraints during the iterations of the algorithm. It is possible that this approach will end up cycling; however, we have never observed this in practice. One possible workaround is to limit the number of iterations, possibly leaving some collision constraints unresolved. In collision-dominant simulations, more advanced algorithms may be necessary. Another limitation is that in our current implementation, we treat collisions as soft constraints with relatively stronger stiffness compared to the elastic models. One possible way to resolve hard collision constraints is to use Lagrangian multipliers by solving the KKT system using its Schur complement [Ichim et al. 2016]. However, in cases with many collision constraints, the Schur complement becomes impractically large. Another possible approach to treating hard collision constraints is the Augmented Lagrangian method [Deng et al. 2013]. Fast and robust collision resolution in challenging scenarios is a problem that deserves significant attention in future work.

7. CONCLUSIONS

We have presented a method for fast physics-based simulation of a large class of hyperelastic materials. The key to our approach is the insight that Projective Dynamics [Bouaziz et al. 2014] can be reformulated as a quasi-Newton method. Aided with line search, we obtain a robust simulator supporting many practical material models. Our quasi-Newton formulation also allows us to further accelerate convergence by combining our method with L-BFGS. Even though L-BFGS is sensitive to initial Hessian approximation, our method suggests a particularly effective Hessian initialization that yields fast convergence. Most of our experiments use 10 iterations of our method, which is typically more accurate than one iteration of Newton's method, while being about 10 times faster and easier to implement. Traditionally, real-time physics is considered to be approximate but fast, while offline physics is accurate but slow. We hope that our method will help to blur the boundaries between real-time and offline physics-based animation.

ACKNOWLEDGMENTS

We thank Jernej Barbič, Erik Brunvand, Elaine Cohen, Sebastian Martin, James O'Brien, Mark Pauly, Peter Shirley, Cem Yuksel, and Hongyi Xu for many inspiring discussions. We also thank Petr Kadleček for help with rendering, Alec Jacobson for providing the mesh generating tool, Dimitar Dinev for proofreading, and Hannah Swan for narrating the accompanying video. This work was supported by NSF awards IIS-1622360 and IIS-1350330.

REFERENCES

Steven S. An, Theodore Kim, and Doug L. James. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph.* 27 (2008), 165:1–165:10.

Jernej Barbič and Doug L. James. 2005. Real-time subspace integration for St. Venant-kirchhoff deformable models. *ACM Trans. Graph.* 24 (2005), 982–990.

Adam W. Bargteil and Elaine Cohen. 2014. Animation of deformable bodies with quadratic Bézier finite elements. *ACM Trans. Graph.* 33 (2014), 27:1–27:10.

Adam W. Bargteil, Chris Wojtan, Jessica K. Hodgins, and Greg Turk. 2007. A finite element method for animating large viscoplastic flow. *ACM Trans. Graph.* 26 (2007), 16:1–16:8.

Klaus Jürgen Bathe and Arthur P. Cimento. 1980. Some practical procedures for the solution of nonlinear finite element equations. In *Computer Methods in Applied Mechanics and Engineering* 22 (1980), 59–85.

Jan Bender, Dan Koschier, Patrick Charrier, and Daniel Weber. 2014a. Position-based simulation of continuous materials. *Computers & Graphics* 44 (2014), 1–10.

Jan Bender, Matthias Müller, Miguel A. Otaduy, Matthias Teschner, and Miles Macklin. 2014b. A survey on position-based simulation methods in computer graphics. In *Comput. Graph. Forum*. 33 (2014), 228–251.

Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. 2012. Shape-up: Shaping discrete geometry with projections. In *Comput. Graph. Forum*, Vol. 31. 1657–1667.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph.* 33 (2014), 154:1–154:11.

Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. 2010. A simple geometric model for elastic deformations. *ACM Trans. Graph.* 29 (2010), 38:1–38:6.

Desai Chen, David Levin, Shinjiro Sueda, and Wojciech Matusik. 2015. Data-driven finite elements for geometry and material design. *ACM Trans. Graph.* 34 (2015), 74:1–74:10.

Bailin Deng, Sofien Bouaziz, Mario Deuss, Juyong Zhang, Yuliy Schwartzburg, and Mark Pauly. 2013. Exploring local modifications for constrained meshes. In *Comput. Graph. Forum*, Vol. 32. 11–20.

Mathieu Desbrun, Peter Schröder, and Alan Barr. 1999. Interactive animation of structured deformable objects. *Graphics Interface* 99 (1999), 10.

Peter Deuflhard. 2011. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*. Springer Science & Business Media.

Jacob Fish, Murali Pandheeradi, and Vladimir Belsky. 1995. An efficient multilevel solution scheme for large scale non-linear systems. *Internat. J. Numer. Methods Engrg.* 38 (1995), 1597–1610.

Marco Fratarcangeli and Fabio Pellacini. 2015. Scalable partitioning for parallel position based dynamics. In *Comput. Graph. Forum*. 34 (2015), 405–413.

Theodore F. Gast, Craig Schroeder, Alexey Stomakhin, Chenfanfu Jiang, and Joseph M. Teran. 2015. Optimization integrator for large time steps. *IEEE Trans. Visualization Comp. Graph.* 21 (2015), 1103–1115.

Joachim Georgii and Rüdiger Westermann. 2006. A multigrid framework for real-time simulation of deformable bodies. *Comput. Graphics* 30, 3 (2006), 408–415.

Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. 2007. Efficient simulation of inextensible cloth. *ACM Trans. Graph.* 26 (2007), 49:1–49:7.

Fabian Hahn, Sebastian Martin, Bernhard Thomaszewski, Robert Sumner, Stelian Coros, and Markus Gross. 2012. Rig-space physics. *ACM Trans. Graph.* 31 (2012), 72:1–72:8.

Ernst Hairer, Christian Lubich, and Gerhard Wanner. 2002. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer.

David Harmon, Daniele Panozzo, Olga Sorkine, and Denis Zorin. 2011. Interference-aware geometric modeling. *ACM Trans. Graph.* 30 (2011), 137:1–137:10.

David Harmon and Denis Zorin. 2013. Subspace Integration with Local Deformations. *ACM Trans. Graph.* 32 (2013), 107:1–107:10.

Michael Hauth and Olaf Etzmuss. 2001. A high performance solver for the animation of deformable objects using advanced numerical methods. In *Comput. Graph. Forum*. 20 (2001), 319–328.

Florian Hecht, Yeon Jin Lee, Jonathan R. Shewchuk, and James F. O'Brien. 2012. Updated sparse cholesky factors for corotational elastodynamics. *ACM Trans. Graph.* 31 (2012), 123:1–123:13.

Magnus Rudolph Hestenes and Eduard Stiefel. 1952. *Methods of Conjugate Gradients for Solving Linear Systems*. Vol. 49. NBS.

Alexandru-Eugen Ichim, Ladislav Kavan, Merlin Nimier-David, and Mark Pauly. 2016. Building and animating user-specific volumetric face rigs. In *Proc. EG Symp. Computer Animation*. 107–117.

Geoffrey Irving, Joseph Teran, and Ron Fedkiw. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. EG Symp. Computer Animation*. 131–140.

Liliya Kharevych, Weiwei Yang, Yiying Tong, Eva Kanso, Jerrold E Marsden, Peter Schröder, and Matthieu Desbrun. 2006. Geometric, variational integrators for computer animation. In *Proc. EG Symp. Computer Animation*. 43–51.

Tae-Yong Kim, Nuttapong Chentanez, and Matthias Müller-Fischer. 2012. Long range attachments-a method to simulate inextensible clothing in computer games. In *Proc. EG Symp. Computer Animation*. 305–310.

Shahar Z. Kovalsky, Meirav Galun, and Yaron Lipman. 2016. Accelerated quadratic proxy for geometric optimization. *ACM Trans. Graph.* 35 (2016), 134:1–134:11.

Siwang Li, Jin Huang, Fernando de Goes, Xiaogang Jin, Hujun Bao, and Mathieu Desbrun. 2014. Space-time editing of elastic motion through material optimization and reduction. *ACM Trans. Graph.* 33 (2014), 108:1–108:10.

Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. 2013. Fast simulation of mass-spring systems. *ACM Trans. Graph.* 32 (2013), 214:1–214:7.

Miles Macklin and Matthias Müller. 2013. Position based fluids. *ACM Trans. Graph.* 32 (2013), 104:1–104:12.

Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. 2014. Unified particle physics for real-time applications. *ACM Trans. Graph.* 33 (2014), 153:1–153:12.

Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-based elastic materials. *ACM Trans. Graph.* 30 (2011), 72:1–72:8.

Tobias Martin, Pushkar Joshi, Miklós Bergou, and Nathan Carr. 2013. Efficient non-linear optimization via multi-scale gradient filtering. In *Comput. Graph. Forum*. 32 (2013), 89–100.

Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph.* 30 (2011), 37:1–37:12.

Matthias Müller. 2008. Hierarchical position based dynamics. In *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2008)*. Eurographics Association.

Matthias Müller and Nuttapong Chentanez. 2011. Solid simulation with oriented particles. *ACM Trans. Graph.* 30 (2011), 92:1–92:10.

Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. 2014. Strain based dynamics. In *Proc. EG Symp. Computer Animation*, Vol. 2 (2014), 149–157.

Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. 2015. Air meshes for robust collision handling. *ACM Trans. Graph.* 34 (2015), 133:1–133:9.

Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. 2002. Stable real-time deformations. In *Proc. EG Symp. Computer Animation*. 49–54.

Matthias Müller and Markus Gross. 2004. Interactive virtual materials. In *Proceedings of Graphics Interface* (2004), 239–246.

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *J. Vis. Comun. Image Represent.* 18 (2007), 109–118.

Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2005. Meshless deformations based on shape matching. *ACM Trans. Graph.* 24 (2005), 471–478.

Rahul Narain, Matthew Overby, and George E. Brown. 2016. ADMM $\supseteq$ projective dynamics: Fast simulation of general constitutive models. In *Proc. ACM SIGGRAPH/Eurographics Symp. on Computer Animation (SCA'16)*. 21–28.

Rahul Narain, Armin Samii, and James F. O'Brien. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph.* 31 (2012), 152:1–152:10.

J.W. Neuberger. 1983. Steepest descent for general systems of linear differential equations in hilbert space. In *Ordinary Differential Equations and Operators*. Springer.

John Neuberger. 2009. *Sobolev Gradients and Differential Equations*. Springer Science & Business Media.

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization*. Springer Verlag.

Alec R. Rivers and Doug L. James. 2007. FastLSM: Fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph.* 26 (2007), 82:1–82:6.

Martin Servin, C. Lacoursière, and N. Melin. 2006. Interactive simulation of elastic deformable materials. In *The Annual SIGRAD Conference; Special Theme: Computer Games* 19 (2006).

Eftychios Sifakis and Jernej Barbič. 2012. FEM simulation of 3D deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH Courses*. 20:1–20:50.

Jos Stam. 2009. Nucleus: Towards a unified dynamics solver for computer graphics. In *IEEE Int. Conf. on CAD and Comput. Graph.* 1–11.

Rasmus Tamstorf, Toby Jones, and Stephen F. McCormick. 2015. Smoothed aggregation multigrid for cloth simulation. *ACM Trans. Graph.* 34 (2015), 245:1–245:13.

Yun Teng, Mark Meyer, Tony DeRose, and Theodore Kim. 2015. Subspace condensation: Full space adaptivity for subspace deformations. *ACM Trans. Graph.* 34 (2015), 76:1–76:9.

Yun Teng, Miguel A. Otaduy, and Theodore Kim. 2014. Simulating articulated subspace self-contact. *ACM Trans. Graph.* 33 (2014), 106:1–106:9.

Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. 2005. Robust quasistatic finite elements and flesh simulation. In *Proc. EG Symp. Computer Animation*. 181–190.

Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically deformable models. In *Computer Graphics (Proc. SIGGRAPH)*, Vol. 21. 205–214.

Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. 2009. Continuum-based strain limiting. In *Comput. Graph. Forum*. 28 (2009), 569–576.

Maxime Tournier, Matthieu Nesme, Benjamin Gilles, and Francois Faure. 2015. Stable constrained dynamics. *ACM Trans. Graph.* 34 (2015), 132:1–132:10.

K. C. Valanis and Robert F. Landel. 1967. The strain-energy function of a hyperelastic material in terms of the extension ratios. *Journal of Applied Physics* 38 (1967), 2997–3002.

Huamin Wang. 2015. A Chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph.* 34 (2015), 246:1–246:9.

Huamin Wang, James O'Brien, and Ravi Ramamoorthi. 2010. Multi-resolution isotropic strain limiting. *ACM Trans. Graph.* 29 (2010), 156:1–156:10.

Hongyi Xu, Funshing Sin, Yufeng Zhu, and Jernej Barbič. 2015. Nonlinear material design using principal stretches. *ACM Trans. Graph.* 34 (2015), 75:1–75:11.

Received May 2016; revised October 2016; accepted December 2016