ADMM ⊇ Projective Dynamics: Fast Simulation of Hyperelastic Models with Dynamic Constraints

Matthew Overby, George E. Brown, Jie Li, and Rahul Narain

Abstract—We apply the alternating direction method of multipliers (ADMM) optimization algorithm to implicit time integration of elastic bodies, and show that the resulting method closely relates to the recently proposed projective dynamics algorithm. However, as ADMM is a general purpose optimization algorithm applicable to a broad range of objective functions, it permits the use of nonlinear constitutive models and hard constraints while retaining the speed, parallelizability, and robustness of projective dynamics. We further extend the algorithm to improve the handling of dynamically changing constraints such as sliding and contact, while maintaining the benefits of a constant, prefactored system matrix. We demonstrate the benefits of our algorithm on several examples that include cloth, collisions, and volumetric deformable bodies with nonlinear elasticity and skin sliding effects.

1 Introduction

Animating realistic deformable objects at interactive rates has been a long-standing goal of computer graphics research. To achieve truly realistic behavior, simulation techniques must be able to support the complex, nonlinear deformation behavior of real materials. Soft materials such as cloth, rubber, and biological soft tissue exhibit significantly nonlinear elastic behavior, and much work has gone into acquiring their constitutive properties for use in computer graphics [1, 2, 3, 4]. However, general nonlinear materials are difficult to robustly simulate in real time. Recent algorithms for real-time simulation, such as position-based dynamics [5, 6] and projective dynamics [7] are fast, parallel, and robust, but only support a restricted subset of elastic models, namely those that can be expressed as a combination of soft constraints.

In this work, which is an extended version of a paper presented at SCA 2016 [8], we apply the *alternating direction method of multipliers* (ADMM) optimization algorithm [9] to the time integration of nonlinear elastic forces, and derive a novel simulation algorithm. Our method generalizes projective dynamics to arbitrary conservative forces, including general hyperelastic materials and hard constraints, while retaining its speed, parallelizability, and robustness (see Fig. 1). We further describe techniques for improving the convergence of non-constant constraints, without requiring the factorization of the system matrix to be recomputed.

Our ADMM-based implicit integration algorithm has the following attractive properties:

- it is a generic method that supports arbitrary conservative forces,
- it is highly parallelizable,
- it is robust to large time steps and nonsmooth energies, and
- in the special case of linear forces, it is identical to projective dynamics.

- *M. Overby, G.E. Brown, J. Li, and R. Narain are with the University of Minnesota.*

2 Related work

2.1 Nonlinear elastic models

Soft materials of interest in graphics, like cloth, rubber, muscle, and skin, exhibit a strongly nonlinear elastic response when undergoing large deformations. Much previous work in graphics has focused on acquiring nonlinear constitutive models of volumetric soft tissue [1, 4] and cloth [2, 3] from observed data; their results show that nonlinearity leads to more realistic visual behavior than linear elastic models. Furthermore, Xu et al. [10] recently showed that allowing artists to directly specify the nonlinear elastic response of a simulated material enables desirable animation effects to be achieved.

Another important aspect of character animation is the realistic motion of skin. Effects such as wrinkling and stretching of skin and appropriate deformation of surface texture are desirable. Wrinkles can be achieved with volumetric interiors enclosed by a thin shell [11] or adaptive layered meshes [12]. Simulation of thin elastic sheets sliding on deforming surfaces can be used to represent the behavior of skin or tight clothing on a character. Li et al. [13] simulate the sliding of hyperelastic skin using an Eulerian approach. Müller et al. [14] show that strain based constraints can be used in position based dynamics to improve skin sliding behavior, although not for hyperelastic materials.

2.2 Parallel time integration

Implicit time integration techniques such as the backward Euler method have long been popular in physics-based animation [15, 16] due to their robustness and unconditional stability. However, their principal drawback is that they require solving a large system of nonlinear equations. Even a single Newton iteration is often too expensive for real-time use, while cheaper methods like Gauss-Seidel or Jacobi iterations can be extremely slow to converge to acceptable accuracy.

Recent years have seen the development of a number of fast and parallel techniques for performing implicit integration. These techniques employ the optimization form

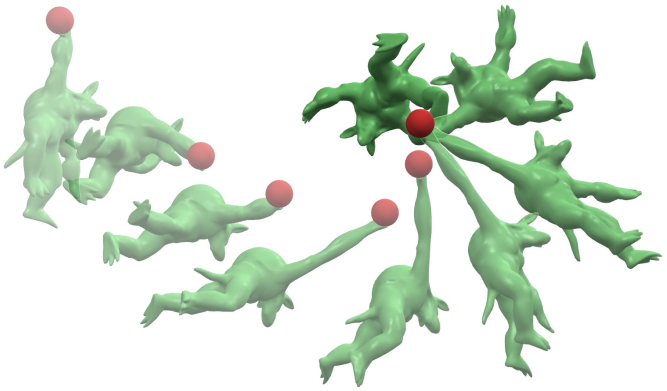


Fig. 1: A neo-Hookean armadillo with 2.8k tetrahedra simulated interactively at 27 fps using our method.

LIST OF SYMBOLS	
$\mathbf{D}$	Reduction matrix
Δt	Time step
$\mathbf{M}$	Mass matrix
$\bar{\mathbf{u}}$	ADMM dual variable
$\mathbf{v}$	Node velocity
$\mathbf{C}$	Constraint matrix
$\mathbf{n}$	Surface normal
$\mathbf{p}$	Constraint projection
$\mathbf{W}$	Weight matrix
$\mathbf{x}$	Node position
$\tilde{\mathbf{x}}$	Predicted $\mathbf{x}$ without forces
$\mathbf{z}$	ADMM update variable
U	Potential energy

TABLE 1: List of symbols used in our paper.

of backward Euler integration [17, 18, 19], and perform the optimization by alternating between local and global steps. Liu et al. [20] proposed a block coordinate descent approach for mass-spring systems. The projective dynamics algorithm of Bouaziz et al. [7] generalized this approach beyond springs to finite element models and other forms of constrained dynamics. This approach is closely related to the Shape-Up algorithm [21] for geometry manipulation using shape constraints. Weiler et al. [22] extend projective dynamics to fluid simulation using a parallel matrix-free conjugate gradient solver. Fratarcangeli et al. [23] perform the linear solve of projective dynamics with parallel graph-colored Gauss-Seidel to permit GPU acceleration and the dynamic addition and removal of constraints. However, the aforementioned integration techniques [20, 7, 22, 23] assume a simplified elastic model, in which the force is always proportional to the distance from a constraint manifold.

Another related method is the position-based dynamics approach [5, 6], which models all forces as hard constraints and solves them using Gauss-Seidel iterations. This is generalized by XPBD [24] which permits constraints to have a finite stiffness, for better modeling of elastic forces. Wang [25] proposed the use of the Chebyshev semi-iterative technique to accelerate the convergence of both projective dynamics and position-based dynamics. This technique is orthogonal to our work, and could potentially be applied to our algorithm as well.

Concurrently to our work, Liu et al. [26] also generalize projective dynamics to nonlinear materials by interpreting it as a quasi-Newton optimization algorithm. They further accelerate convergence using the L-BFGS method [27].

2.3 Primal-dual algorithms for optimization

In the convex optimization literature, primal-dual methods [28, 29] such as ADMM [9], the alternating direction algorithm (AMA) [30], and the primal-dual hybrid gradient method (PDHG) [31] have been growing in popularity. Such methods rely on decomposing the objective function into two terms and perform alternating optimization steps on each in turn. This is particularly useful if the two terms have simple but incompatible structures, so that efficient optimization routines can be implemented for each but not for their sum. Recently, accelerated versions of ADMM for problems with additional regularity have appeared in the literature [32, 33, 34]. Furthermore, while the theoretical basis of ADMM arises from convex optimization, it has been successfully applied to many nonconvex problems including distributed clustering [35], phase retrieval [36], and nonnegative matrix factorization [37], and there is a growing body of work proving convergence of the algorithm on certain classes of nonconvex problems [38, 39, 40].

Such primal-dual algorithms have begun to find use in graphics as well. In their fluid tracking algorithm, Gregson et al. [41] used ADMM to impose a divergence-free constraint onto a velocity estimation problem. ADMM is a highly parallelizable variant of the *method of multipliers*, or the augmented Lagrangian method, which has previously been applied to strain limiting in cloth simulation [42]. The PDHG algorithm was used by Narain et al. [43] to optimize the reproduction of defocus cues on 3D displays. However, to our knowledge, the connection between primal-dual optimization algorithms and recent techniques for parallel implicit integration has not yet been observed.

3 IMPLICIT INTEGRATION USING ADMM

Consider a physical system with positions $\mathbf{x} \in \mathbb{R}^d$, velocities $\mathbf{v} \in \mathbb{R}^d$, and a constant inertia matrix $\mathbf{M}$. We will perform backward Euler integration on some of the forces acting on the system, denoted $\mathbf{f}$, while treating others, denoted $\mathbf{f}_{\text{ext}}$, explicitly. This yields the system of equations

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \mathbf{v}(t+\Delta t)\Delta t, \tag{1a}$$

$$\mathbf{M}\mathbf{v}(t+\Delta t) = \mathbf{M}\mathbf{v}(t) + \mathbf{f}_{\text{ext}}(t)\Delta t + \mathbf{f}(t+\Delta t)\Delta t, \tag{1b}$$

or equivalently,

$$\frac{1}{\Delta t^2}\mathbf{M}\big(\mathbf{x}(t+\Delta t) - \tilde{\mathbf{x}}(t+\Delta t)\big) = \mathbf{f}(t+\Delta t) \tag{2a}$$

$$\mathbf{v}(t+\Delta t) = \frac{\mathbf{x}(t+\Delta t) - \mathbf{x}(t)}{\Delta t} \tag{2b}$$

where $\tilde{\mathbf{x}}(t+\Delta t) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t + \mathbf{M}^{-1}\mathbf{f}_{\text{ext}}(t)\Delta t^2$ is the predicted state at $t+\Delta t$ in the absence of the implicit forces $\mathbf{f}$.

In general, (2a) is a large, nonlinear system of equations, as the forces $\mathbf{f}$ are typically nonlinear in $\mathbf{x}$ and/or $\mathbf{v}$. A common solution in graphics applications is to linearize the system about the current state, effectively applying one iteration of Newton's method. However, this strategy can be inadequate when $\mathbf{f}$ is strongly nonlinear, for example in the

bending of stiff strands [44], necessitating multiple Newton solves and a commensurate increase in computation time.

On the other hand, when all the implicit forces $\mathbf{f}$ are conservative, that is, $\mathbf{f} = -\nabla U(\mathbf{x})$ for some potential energy $U(\mathbf{x})$, the system of equations (2a) can be expressed as an optimization problem [18, 19],

$$\mathbf{x}(t+\Delta t) = \arg\min_{\mathbf{x}} \left(\frac{1}{2\Delta t^2} \|\mathbf{x} - \tilde{\mathbf{x}}(t+\Delta t)\|_{\mathbf{M}}^2 + U(\mathbf{x}) \right), \tag{3}$$

where $\|\mathbf{x}\|_{\mathbf{M}} = \sqrt{\mathbf{x}^T \mathbf{M} \mathbf{x}}$. To simplify the notation, from here on we will write $\mathbf{x}(t+\Delta t)$ and $\tilde{\mathbf{x}}(t+\Delta t)$ simply as $\mathbf{x}$ and $\tilde{\mathbf{x}}$. The equivalence of the two formulations can be verified by observing that the gradient of the objective in (3) is zero precisely when (2a) is satisfied. The advantage of an optimization-based formulation is that it is numerically more convenient to work with, as pointed out by [17], and permits a broader range of efficient solvers.

Typically, U is the sum of many different energy terms, each of which affect only a small subset of nodes. Following the projective dynamics approach [7], we define for each energy term a "reduction matrix" $\mathbf{D}_i$ such that the energy only depends on a small vector of local coordinates $\mathbf{D}_i\mathbf{x}$, for example, the edge vector for a spring, or the deformation gradient for a finite element. The use of such matrices is motivated in [21]: in effect, $\mathbf{D}_i$ improves convergence by projecting out the translational null space of the energy. Then we have

$$U(\mathbf{x}) = \sum_{i=1}^{m} U_i(\mathbf{D}_i\mathbf{x}), \tag{4}$$

where the function U_i maps the local coordinates to the corresponding energy. For example, for a spring of length ℓ and stiffness k between nodes a and b, we choose $\mathbf{D}_i$ so that $\mathbf{D}_i\mathbf{x} = [\mathbf{x}_b - \mathbf{x}_a]$, and define $U_i(\mathbf{D}_i\mathbf{x}) = \frac{1}{2}k(\|\mathbf{D}_i\mathbf{x}\| - \ell)^2$. For convenience, we concatenate all the local coordinates into a single vector, and define the function

$$U_*\left(\begin{bmatrix} \mathbf{D}_1\mathbf{x} \\ \mathbf{D}_2\mathbf{x} \\ \vdots \\ \mathbf{D}_m\mathbf{x} \end{bmatrix}\right) = \sum_{i=1}^{m} U_i(\mathbf{D}_i\mathbf{x}), \tag{5}$$

so that $U(\mathbf{x}) = U_*(\mathbf{D}\mathbf{x})$ where $\mathbf{D} = \begin{bmatrix} \mathbf{D}_1^T & \mathbf{D}_2^T & \cdots & \mathbf{D}_n^T \end{bmatrix}^T$.

Thus, our key task is to solve the optimization problem

$$\min_{\mathbf{x}} \ \frac{1}{2\Delta t^2} \|\mathbf{x} - \tilde{\mathbf{x}}\|_{\mathbf{M}}^2 + U_*(\mathbf{D}\mathbf{x}). \tag{6}$$

Given the special structure of the problem, we will employ the ADMM algorithm in order to solve it efficiently. First, we give some background on ADMM and its relevant properties.

3.1 ADMM

The alternating direction method of multipliers [9] is a method for solving optimization problems of the form

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{z}} \ f(\mathbf{x}) + g(\mathbf{z}) \\ &\text{s.t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}. \end{aligned} \tag{7}$$

The algorithm works by introducing a (scaled) dual variable $\mathbf{u}$ and iterating the following update rules:

$$\mathbf{x}^{n+1} = \arg\min_{\mathbf{x}} \Big(f(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}^n - \mathbf{c} + \mathbf{u}^n\|^2 \Big), \tag{8}$$

$$\mathbf{z}^{n+1} = \arg\min_{\mathbf{z}} \Big(g(\mathbf{z}) + \frac{\rho}{2} \|\mathbf{A}\mathbf{x}^{n+1} + \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}^n\|^2 \Big), \tag{9}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + (\mathbf{A}\mathbf{x}^{n+1} + \mathbf{B}\mathbf{z}^{n+1} - \mathbf{c}) \tag{10}$$

for some step length parameter $\rho > 0$. Here superscripts on the variables denote the iteration number.

When f and g are convex, ADMM is guaranteed to converge to the optimal solution for any $\rho > 0$ under very mild assumptions [9]. In particular, neither f or g needs to be differentiable in the convex case. While the energy functions U_i used in physics-based animation are typically not convex due to the presence of rotations, ADMM has in practice also been observed to perform well on many problems where f and/or g are nonconvex (see references in Sec. 2.3).

Interestingly, in contrast to methods like gradient descent and conjugate gradient, ADMM is invariant to rescaling the optimization variables $\mathbf{x}$ and $\mathbf{z}$, in the sense that if we introduce rescaled variables $\bar{\mathbf{x}} = \mathbf{P}\mathbf{x}, \bar{\mathbf{z}} = \mathbf{Q}\mathbf{z}$ for invertible matrices $\mathbf{P}$ and $\mathbf{Q}$ and then apply ADMM to the transformed problem

$$\begin{aligned} &\min_{\bar{\mathbf{x}},\bar{\mathbf{z}}} \ f(\mathbf{P}^{-1}\bar{\mathbf{x}}) + g(\mathbf{Q}^{-1}\bar{\mathbf{z}}) \\ &\text{s.t. } \mathbf{A}\mathbf{P}^{-1}\bar{\mathbf{x}} + \mathbf{B}\mathbf{Q}^{-1}\bar{\mathbf{z}} = \mathbf{c}, \end{aligned} \tag{11}$$

we get exactly the same sequence of iterates (rescaled by $\mathbf{P}$ and $\mathbf{Q}$) as we would for the original problem. However, ADMM is *not* invariant to rescaling the linear constraint: modifying the problem to the equivalent

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{z}} \ f(\mathbf{x}) + g(\mathbf{z}) \\ &\text{s.t. } \mathbf{W}\mathbf{A}\mathbf{x} + \mathbf{W}\mathbf{B}\mathbf{z} = \mathbf{W}\mathbf{c} \end{aligned} \tag{12}$$

for some invertible matrix $\mathbf{W}$ does significantly affect the iterates and the convergence rate. To motivate a rule for choosing $\mathbf{W}$, we consider the behavior of the algorithm on a purely quadratic problem, which is easier to analyze. In Appendix A of the supplemental document, we show that when both f and g are convex quadratic functions and $\mathbf{B}$ is an invertible matrix, if we choose $\rho = 1$ and $\mathbf{W}$ such that $g(\mathbf{z}) = \frac{1}{2}\|\mathbf{W}\mathbf{B}\mathbf{z}\|^2$, each ADMM iteration reduces the error by a constant factor of $\frac{1}{2}$. We find this simple heuristic to be effective for general non-quadratic problems as well, as long as g can be approximated by a quadratic function.

3.2 Application of ADMM to implicit integration

Our problem (6) can be expressed in the form (7) as

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{z}} \ \frac{1}{2\Delta t^2} \|\mathbf{x} - \tilde{\mathbf{x}}\|_{\mathbf{M}}^2 + U_*(\mathbf{z}) \\ &\text{s.t. } \mathbf{W}(\mathbf{D}\mathbf{x} - \mathbf{z}) = \mathbf{0}, \end{aligned} \tag{13}$$

that is, by choosing

$$f(\mathbf{x}) = \frac{1}{2\Delta t^2} \|\mathbf{x} - \tilde{\mathbf{x}}\|_{\mathbf{M}}^2, \tag{14a}$$

$$g(\mathbf{z}) = U_*(\mathbf{z}), \tag{14b}$$

$$\mathbf{A} = \mathbf{W}\mathbf{D}, \tag{14c}$$

$$\mathbf{B} = -\mathbf{W}, \tag{14d}$$

$$\mathbf{c} = \mathbf{0}. \tag{14e}$$

Here $\mathbf{z}$ is a new variable that satisfies $\mathbf{z} = \mathbf{D}\mathbf{x}$ upon convergence. In keeping with the separability of U_*, we further divide $\mathbf{z}$ into segments $\mathbf{z}_i$ of the same size as the corresponding $\mathbf{D}_i$, and take the constraint weighting matrix $\mathbf{W}$ to be block diagonal with blocks of the corresponding size:

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_m \end{bmatrix}, \qquad \mathbf{W} = \begin{bmatrix} \mathbf{W}_1 & & & \\ & \mathbf{W}_2 & & \\ & & \ddots & \\ & & & \mathbf{W}_m \end{bmatrix}. \tag{15}$$

Thus we have

$$U_*(\mathbf{z}) = \sum_{i=1}^{m} U_i(\mathbf{z}_i) \tag{16}$$

and the constraint is equivalently

$$\mathbf{W}_i(\mathbf{D}_i\mathbf{x} - \mathbf{z}_i) = \mathbf{0} \text{ for all } i = 1, \ldots, m. \tag{17}$$

Typically, each block is a multiple of the identity, $\mathbf{W}_i = w_i\mathbf{I}$ for some scalar weight w_i.

As motivated above, we fix $\rho = 1$, and choose $\mathbf{W}_i$ for each energy term U_i to approximate it by a quadratic function. That is, we seek an invertible matrix $\mathbf{W}_i$ such that $U_i(\mathbf{z}_i) \approx \frac{1}{2}\|\mathbf{W}_i\mathbf{z}_i\|^2$ over the likely range of values of $\mathbf{z}_i$. Similar to what was observed by Liu et al. [26], this is preferable to using the Hessian of U_i at a single point, as the latter may vanish for some nonlinear energies. For example, for a spring energy $U(\mathbf{z}_i) = \frac{1}{2}k_i(\|\mathbf{z}_i\| - \ell_i)^2$, the Hessian is undefined at $\mathbf{z}_i = \mathbf{0}$, and has rank 1 at the rest length $\|\mathbf{z}_i\| = \ell_i$; nevertheless, we may take $U_i(\mathbf{z}_i) \approx \frac{1}{2}k_i\|\mathbf{z}_i\|^2$ as a reasonable quadratic approximation of the global behavior of the spring energy. Similarly, for other elastic models, we choose a single scalar k_i to characterize the growth of U_i with respect to $\|\mathbf{z}_i\|$, and set $\mathbf{W}_i = w_i\mathbf{I} = \sqrt{k_i}\mathbf{I}$. Note that a poor choice of $\mathbf{W}_i$ does not change the solution, though it can adversely affect the convergence rate.

Applying the ADMM update rules (8)–(10) to (14a)–(14e) and introducing the variable $\bar{\mathbf{u}} = \mathbf{W}^{-1}\mathbf{u}$ to simplify the notation, we obtain the algorithm

$$\begin{aligned} \mathbf{x}^{n+1} &= \arg\min_{\mathbf{x}}\Big(\frac{1}{2\Delta t^2}\|\mathbf{x}-\tilde{\mathbf{x}}\|_{\mathbf{M}}^2 + \frac{1}{2}\|\mathbf{W}(\mathbf{D}\mathbf{x}-\mathbf{z}^n+\bar{\mathbf{u}}^n)\|^2\Big) \\ &= (\mathbf{M} + \Delta t^2\mathbf{D}^T\mathbf{W}^T\mathbf{W}\mathbf{D})^{-1} \\ &\quad \Big(\mathbf{M}\tilde{\mathbf{x}} + \Delta t^2\mathbf{D}^T\mathbf{W}^T\mathbf{W}(\mathbf{z}^n - \bar{\mathbf{u}}^n)\Big), \end{aligned} \tag{18}$$

$$\mathbf{z}^{n+1} = \arg\min_{\mathbf{z}}\Big(U_*(\mathbf{z}) + \frac{1}{2}\|\mathbf{W}(\mathbf{D}\mathbf{x}^{n+1} - \mathbf{z} + \bar{\mathbf{u}}^n)\|^2\Big), \tag{19}$$

$$\bar{\mathbf{u}}^{n+1} = \bar{\mathbf{u}}^n + \mathbf{D}\mathbf{x}^{n+1} - \mathbf{z}^{n+1}. \tag{20}$$

As the algorithm proceeds, $\mathbf{x}$ and $\mathbf{z}$ converge to the optimal solution $\mathbf{x}^*$ and the corresponding local coordinates $\mathbf{z}^* = \mathbf{D}\mathbf{x}^*$ respectively. To better understand the meaning of the update rules and the role of $\bar{\mathbf{u}}$, we give a physical interpretation of the equations in the following subsection.

For interactive applications, using a small fixed number of iterations is generally sufficient. If it is necessary to measure the progress towards convergence, one can use the *primal* and *dual residuals* [9],

$$\mathbf{r}^{n+1} = \mathbf{W}(\mathbf{D}\mathbf{x}^{n+1} - \mathbf{z}^{n+1}), \tag{21}$$

$$\mathbf{s}^{n+1} = \mathbf{D}^T\mathbf{W}^T\mathbf{W}(\mathbf{z}^{n+1} - \mathbf{z}^n), \tag{22}$$

terminating when both $\|\mathbf{r}^n\|$ and $\|\mathbf{s}^n\|$ are sufficiently small.

3.3 Physical interpretation of the ADMM equations

A conservative energy of the form $U_i(\mathbf{D}_i\mathbf{x})$ gives rise to a force $\mathbf{f}_i = -\mathbf{D}_i^T\nabla U(\mathbf{D}_i\mathbf{x})$. We may therefore think of $\mathbf{g}_i = -\nabla U_i$ as the generalized force due to U_i in its local coordinates, corresponding to the actual force $\mathbf{f}_i = \mathbf{D}_i^T\mathbf{g}_i$.

After the $\mathbf{z}$-update, we have

$$\nabla U_*(\mathbf{z}^{n+1}) - \mathbf{W}^T\mathbf{W}(\mathbf{D}\mathbf{x}^{n+1} - \mathbf{z}^{n+1} + \bar{\mathbf{u}}^n) = \mathbf{0}. \tag{23}$$

The quantity in parentheses is precisely the value assigned to $\bar{\mathbf{u}}^{n+1}$; in other words, the updated $\bar{\mathbf{u}}$ is directly proportional to the generalized force at $\mathbf{z}^{n+1}$:

$$\begin{aligned} \bar{\mathbf{u}}^{n+1} &= \bar{\mathbf{u}}^n + \mathbf{D}\mathbf{x}^{n+1} - \mathbf{z}^{n+1} \\ &= (\mathbf{W}^T\mathbf{W})^{-1}\nabla U_*(\mathbf{z}^{n+1}) \\ &= -(\mathbf{W}^T\mathbf{W})^{-1}\mathbf{g}(\mathbf{z}^{n+1}). \end{aligned} \tag{24}$$

Thus $\bar{\mathbf{u}}^n$ encodes a running estimate of all the generalized forces in the system. Below, we reinterpret the ADMM iterations as a procedure for updating the generalized forces themselves, $\mathbf{g}^n = \nabla U_*(\mathbf{z}^n)$.

The $\mathbf{x}$-step of Eq. (18) is equivalent to performing a backward Euler step in which the energy U is replaced with a quadratic approximation,

$$\begin{aligned} \tilde{U}(\mathbf{x}) &= \frac{1}{2}\|\mathbf{W}(\mathbf{D}\mathbf{x} - \mathbf{z}^n + \bar{\mathbf{u}}^n)\|^2 \\ &= \frac{1}{2}\|\mathbf{W}(\mathbf{D}\mathbf{x} - \mathbf{z}^n)\|^2 - \mathbf{x}^T\mathbf{D}^T\mathbf{g}^n + \text{const.} \end{aligned} \tag{25}$$

This approximation corresponds to a force that equals the estimated total $\mathbf{f}^n = \mathbf{D}^T\mathbf{g}^n$ when $\mathbf{x}$ is consistent with $\mathbf{z}^n$, and increases according to $\mathbf{W}^T\mathbf{W}$ elsewhere.

The $\mathbf{z}$-step then computes the equilibrium local coordinates for a superposition of the true nonlinear energy U_* and a similar quadratic function,

$$\mathbf{z}^{n+1} = \arg\min_{\mathbf{z}}\Big(U_*(\mathbf{z}) + \mathbf{z}^T\mathbf{g}^n + \frac{1}{2}\|\mathbf{W}(\mathbf{D}\mathbf{x}^{n+1} - \mathbf{z})\|^2\Big). \tag{26}$$

The second term cancels out the gradient of U_* at $\mathbf{z}^n$, allowing the third term to pull $\mathbf{z}$ towards consistency with $\mathbf{x}^{n+1}$. If the quadratic approximation is exact ($U_*(\mathbf{z}) = \frac{1}{2}\|\mathbf{W}\mathbf{z}\|^2$), $\mathbf{z}$ will move exactly halfway towards $\mathbf{D}\mathbf{x}^{n+1}$. Finally, as shown above, the $\bar{\mathbf{u}}$-step amounts to updating the force estimate $\mathbf{g}^{n+1}$.

3.4 Implementation

Broadly, the algorithm is composed of two main steps: solving local energies with Eqs. (19)-(20) and the global linear solve, Eq. (18).

The separable structure of U_* and $\mathbf{W}$ makes the $\mathbf{z}$- and $\bar{\mathbf{u}}$-updates trivially parallelizable. The dual variable $\bar{\mathbf{u}}$ is of the same size as $\mathbf{z}$ and can be decomposed into subvectors $\bar{\mathbf{u}}_i$ corresponding to each $\mathbf{z}_i$. Then the $\mathbf{z}$- and $\bar{\mathbf{u}}$-updates for each energy term can be performed independently.

In the $\mathbf{x}$-update (18), the matrix $\mathbf{M} + \Delta t^2\mathbf{D}^T\mathbf{W}^T\mathbf{W}\mathbf{D}$ has the same structure as the global matrix used in projective dynamics. In particular, it is constant, symmetric, and

Algorithm 1 Implicit time integration with ADMM.

```
Initialize x = x̃ and u = 0
loop
    for all energy terms i in parallel do        ▷ Local step
        Compute D_i x + u_i
        Update z_i using (27)
        Update ū_i using (28)
    end for
    Update x using (18)                          ▷ Global step
end loop
```

positive definite. We precompute and store its Cholesky factorization so that solving Eq. (18) is extremely fast.

The entire procedure for implicit integration using ADMM is summarized in Alg. 1. Note that as the algorithm proceeds by alternating between global and local steps, we can list them in either order without substantially changing the algorithm. In practice, it is preferable to perform the local step first, as that is where the forces are computed.

We discuss different types of forces that can be used with our algorithm in Sec. 4. These include both linear and nonlinear elastic forces, as well as hard constraints such as strain limiting and collisions. In Sec. 5, we show how hard constraints can be integrated with the $\mathbf{x}$-update instead of as an energy force, allowing faster convergence for constrained systems. This is ideal for collision and skin-slide constraints, in which the constraint is satisfied even if the solver is not run to convergence.

4 Forces

To apply our algorithm to a specific simulation problem, the main task is to implement a solver for (19)-(20) for each energy term U_i in the simulation:

$$\mathbf{z}_i^{n+1} = \arg\min_{\mathbf{z}_i} \Big(U_i(\mathbf{z}_i) + \frac{1}{2} \|\mathbf{W}_i(\mathbf{D}_i \mathbf{x}^{n+1} - \mathbf{z}_i + \bar{\mathbf{u}}_i^n)\|^2 \Big), \tag{27}$$

$$\bar{\mathbf{u}}_i^{n+1} = \bar{\mathbf{u}}_i^n + \mathbf{D}_i \mathbf{x}^{n+1} - \mathbf{z}_i^{n+1}. \tag{28}$$

As we choose $\mathbf{W}_i$ to be a multiple of the identity, (19) can be interpreted as a *proximal operator* [45] for U_i. Intuitively, it amounts to minimizing U_i subject to a quadratic penalty for moving away from $\mathbf{D}_i \mathbf{x}^{n+1} + \bar{\mathbf{u}}_i^n$. While this is still a nonlinear optimization problem that generally does not have a closed-form solution, for typical energy terms it is very low-dimensional and therefore feasible to solve numerically or using precomputation. In Sec. 4.1, we show how an existing projective dynamics implementation can easily be adapted to perform the $\mathbf{z}$- and $\bar{\mathbf{u}}$-update in our algorithm. More complex energies that do not fit into the projective dynamics framework are discussed in Sec. 4.2, and we have found a few ($\leq$ 10) iterations of L-BFGS to be sufficient for evaluating the proximal operator for them.

4.1 Projective dynamics energies

Projective dynamics solves the optimization problem (6) under the assumption that the energy terms are proportional to the squared distance to some constraint manifolds $\mathcal{C}_i$,

$$U_i(\mathbf{D}_i \mathbf{x}) = \min_{\mathbf{p}_i \in \mathcal{C}_i} \frac{k_i}{2} \|\mathbf{D}_i \mathbf{x} - \mathbf{p}_i\|^2 \tag{29}$$

for some stiffnesses k_i. Thus they act as soft constraints or penalty forces pulling the system towards satisfying the constraints. The projective dynamics algorithm consists of alternating between a local step (performed in parallel for each energy term),

$$\mathbf{p}_i \leftarrow \mathrm{proj}_{\mathcal{C}_i}(\mathbf{D}_i \mathbf{x}), \tag{30}$$

and a global step,

$$\mathbf{x} \leftarrow \left(\mathbf{M} + \Delta t^2 \sum_{i=1}^m k_i \mathbf{D}_i^T \mathbf{D}_i \right)^{-1} \left(\mathbf{M}\tilde{\mathbf{x}} + \Delta t^2 \sum_{i=1}^m k_i \mathbf{D}_i^T \mathbf{p}_i \right) \tag{31}$$

where $\mathrm{proj}_{\mathcal{C}_i}$ gives the nearest point on $\mathcal{C}_i$. The user is required to implement the projection step $\mathrm{proj}_{\mathcal{C}_i}$ for the constraint manifolds of interest.

In Appendix B in the supplemental document, we show that this algorithm is nearly identical to a special case of the ADMM algorithm applied to (29), if the weights are chosen to be $\mathbf{W}_i = \sqrt{k_i}\mathbf{I}$ as usual. In fact, it is exactly identical if the constraint manifolds $\mathcal{C}_i$ are affine, for example in linear (non-corotated) elasticity.

For general constraint manifolds, a similar argument to that in Appendix B reduces our $\mathbf{z}$-step to

$$\mathbf{p}_i^{n+1} = \mathrm{proj}_{\mathcal{C}_i}(\mathbf{D}_i \mathbf{x}^{n+1} + \bar{\mathbf{u}}_i^n), \tag{32}$$

$$\mathbf{z}_i^{n+1} = \frac{1}{2}(\mathbf{p}_i^{n+1} + \mathbf{D}_i \mathbf{x}^{n+1} + \bar{\mathbf{u}}_i^n), \tag{33}$$

while the other steps remain the same. Thus any energy terms from a projective dynamics implementation can be readily incorporated into our algorithm.

4.2 Other elastic models

In our algorithm, any conservative force acting on the system is specified by two things: the reduction matrix $\mathbf{D}_i$, and the local energy function $U_i(\mathbf{z}_i)$. Here we give examples for some common forces and constraints.

Finite elements and nonlinear elasticity For a tetrahedron, we assume a rest shape with vertex positions $\mathbf{X}_a, \mathbf{X}_b, \mathbf{X}_c, \mathbf{X}_d$ in reference space, and an associated reference shape matrix $\mathbf{B} = \begin{bmatrix} \mathbf{X}_b - \mathbf{X}_a & \mathbf{X}_c - \mathbf{X}_a & \mathbf{X}_d - \mathbf{X}_a \end{bmatrix}$. Its deformed configuration is characterized by the deformation gradient,

$$\mathbf{F} = \begin{bmatrix} \mathbf{x}_b - \mathbf{x}_a & \mathbf{x}_c - \mathbf{x}_a & \mathbf{x}_d - \mathbf{x}_a \end{bmatrix} \mathbf{B}^{-1}, \tag{34}$$

which is a 3×3 matrix. We define $\mathbf{D}_i$ to be the matrix mapping $\mathbf{x}$ to $\mathrm{vec}(\mathbf{F})$, the 9-dimensional vector that contains the columns of $\mathbf{F}$. We do the same for triangles, except that $\mathbf{X}_a, \mathbf{X}_b, \mathbf{X}_c$ lie in a 2D reference space, and so the deformation gradient is a 3×2 matrix and $\mathbf{D}_i \mathbf{x}$ lies in $\mathbb{R}^6$.

Any hyperelastic constitutive model can be specified as a strain energy density function $\Psi(\mathbf{F})$, so that the elastic energy of a tetrahedron of volume $V_i = \frac{1}{6} \det \mathbf{B}$ is

$$U_i(\mathbf{z}_i) = V_i \Psi(\mathrm{reshape}_{3\times 3}(\mathbf{z}_i)). \tag{35}$$

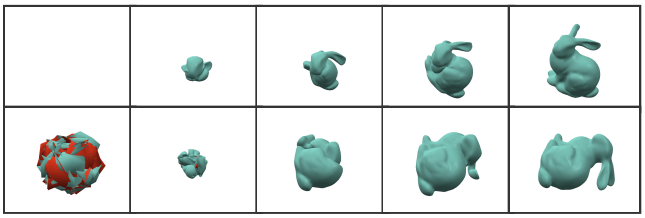


Fig. 2: A low-stiffness Saint Venant-Kirchhoff bunny recovers from having its vertices (top) collapsed to a point, or (bottom) randomized.

For a given constitutive model, one needs to implement a single function

$$\text{prox}_{\Psi}(\tilde{\mathbf{F}}, \tau) = \arg\min_{\mathbf{F}} \Big(\Psi(\mathbf{F}) + \frac{\tau}{2} \|\mathbf{F} - \tilde{\mathbf{F}}\|_F^2 \Big), \tag{36}$$

after which the $\mathbf{z}$-step for a tetrahedron of any shape can be performed as follows:

$$\tilde{\mathbf{F}}_i^{n+1} = \text{reshape}_{3\times 3}(\mathbf{D}_i \mathbf{x}^{n+1} + \bar{\mathbf{u}}_i^n) \tag{37}$$

$$\mathbf{F}_i^{n+1} = \text{prox}_{\Psi}(\tilde{\mathbf{F}}_i^{n+1}, w_i^2/V_i) \tag{38}$$

$$\mathbf{z}_i^{n+1} = \text{vec}(\mathbf{F}_i^{n+1}). \tag{39}$$

The formulation for triangle strains is analogous.

For isotropic materials, where strain energy depends only on the singular values of $\mathbf{F}$, it can be shown that the proximal $\mathbf{F}$ has the same singular vectors as $\tilde{\mathbf{F}}$, and it is sufficient to apply the proximal operator to the singular values alone. This reduces the problem to a 3-dimensional optimization problem. To handle inverted tetrahedra, we use the IFE convention for the singular value decomposition [46, 47], and constrain prox_{Ψ} to only return nonnegative singular values. Fig. 2 demonstrates the robustness of our approach.

Hyperelastic skin Biological tissues like skin are nearly incompressible because they contain a large amount of water, and are highly resistant to large deformations. Variations of the Fung model [48, Ch. 7.10] are often used to represent this behavior. These models contain an exponential term which allows the material to undergo small deformations but makes it extremely resistant to larger ones. Specifically, we model skin as an incompressible material with the energy density function

$$\Psi_{\text{Fung}}(\mathbf{F}) = \frac{\mu}{2b}\big(e^{b(I_1-3)} - 1\big), \tag{40}$$

where μ and b are model parameters, and $I_1 = \text{tr}(\mathbf{F}^T\mathbf{F})$ is the first strain invariant, i.e. the sum of the squared principal stretches $\sigma_1^2 + \sigma_2^2 + \sigma_3^2$. We apply this mode to a thin sheet composed of triangles by letting $\sigma_3 = 1/(\sigma_1\sigma_2)$ to satisfy incompressibility.

4.3 Energy based hard constraints

So far, we have shown how simple elastic models that act as soft constraints can be incorporated in our algorithm. We can also model hard constraints that the system is not permitted to violate, by setting $U = \infty$ for invalid configurations. Given a set $\mathcal{S}$ of allowed values of $\mathbf{D}_i\mathbf{x}$, we simply add an energy term equal to the characteristic function of $\mathcal{S}$,

$$U_i(\mathbf{z}_i) = \begin{cases} 0 & \text{if } \mathbf{z}_i \in \mathcal{S}, \\ \infty & \text{otherwise.} \end{cases} \tag{41}$$

In this case, the $\mathbf{z}$-step simply amounts to a projection to the nearest configuration on $\mathcal{S}$,

$$\mathbf{z}_i^{n+1} = \text{proj}_{\mathcal{S}}(\mathbf{D}_i\mathbf{x}^{n+1} + \bar{\mathbf{u}}_i^n). \tag{42}$$

We note that while all the $\mathbf{z}_i$ are always projected to the constraint set, the full system state $\mathbf{x}$ may not exactly satisfy the constraints before convergence. Furthermore, this approach requires an arbitrary choice of w_i, which may need to be tuned depending on the magnitude of other forces acting on the constrained nodes. In Sec. 5 we describe an alternative approach that does not require such tuning.

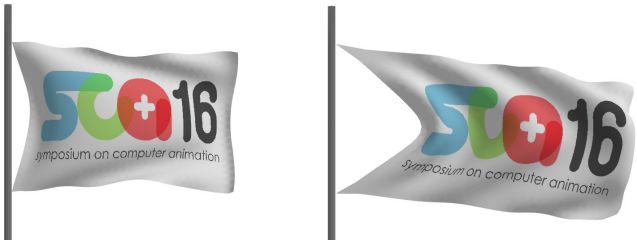


Fig. 3: Strain limiting for cloth using (left) our algorithm, and (right) projective dynamics. Projective dynamics only models soft constraints, and therefore cannot enforce the strain limits in the presence of large forces such as high winds.

Strain limiting Rigid rods and inextensible ropes can easily be modeled with this approach, where the projection becomes a normalization of the edge vector to exactly the desired length. In Fig. 3, we show a more complex example: a cloth flag with strain limits applied to each triangle. For woven cloth it is appropriate to limit strain primarily in the warp and weft directions, which we do by clamping the columns of the deformation gradient $\mathbf{F}$ so their lengths lie in $[0.95, 1.05]$. Isotropic strain limiting can be applied by clamping the singular values of $\mathbf{F}$ instead.

Collisions Objects colliding with static obstacles can be handled with Eq. (41). We add a single non-penetration energy U_{np} to the system, which is infinite if any vertex is inside any obstacle. Thus $\mathbf{D}_{\text{np}} = \mathbf{I}$, and the $\mathbf{z}$-step simply projects each vertex to the nearest non-penetrating location. Unlike the projective dynamics approach, the system matrix used in the global $\mathbf{x}$-step does not need to change when collisions occur.

Friction Since friction is a non-conservative force and is not associated with a potential function, it cannot be directly incorporated into our optimization-based framework. As a preliminary approach, we treat it as an external force that is updated as the solver proceeds. In each iteration, after performing the global solve, we obtain the updated velocity estimate $\mathbf{v}^{n+1} = \big(\mathbf{x}^{n+1} - \mathbf{x}(t)\big)/\Delta t$. For each vertex in contact with an obstacle, we obtain its contact normal force

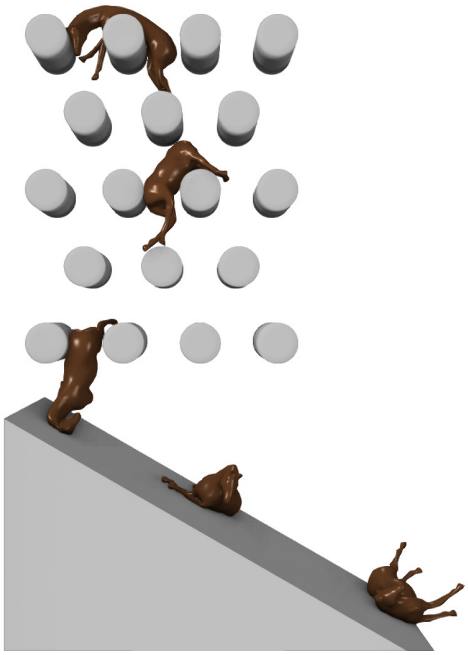


Fig. 4: An elastic horse is dropped onto a series of cylinders, demonstrating collision handling. When the horse hits the inclined plane, friction causes it to roll and flail its limbs.

f_N from the corresponding entry of $\bar{\mathbf{u}}_{\text{np}}$, use the Coulomb law to compute the friction force using f_N and the vertex's tangential relative velocity, and add it as an external force in $\mathbf{f}_{\text{ext}}$.

Strictly speaking, modifying $\mathbf{f}_{\text{ext}}$ (and correspondingly $\tilde{\mathbf{x}}$) as the solver proceeds means that the problem no longer corresponds to a fixed optimization problem (6), and the ADMM algorithm is not strictly applicable. Nevertheless, we have observed the approach to work well in practice, as illustrated in Fig. 4. In future work, we hope to explore alternative approaches that retain the optimization formulation, perhaps through the use of a dissipation function as advocated by recent work [49].

5 Constrained Global Step

In many applications, the prefactored linear solve with energy-based hard constraints is sufficient to robustly simulate a variety of constrained dynamics effects. However, this approach suffers from two limitations, which arise because the solver is not run to convergence in practice. First, while the local coordinate $\mathbf{z}$ is always projected to the constraint set, the node positions in $\mathbf{x}$ may not actually satisfy the constraint. Second, the solver's ability to resolve energy-based constraints is largely influenced by the choice of the associated w_i. Increasing w_i causes the solver to prioritize resolving the constraints, at the expense of other forces acting on the same nodes. For example, Fig. 5 shows how energy based sliding constraints with different values of w_i change the elastic behavior. When the constraint weight is increased, the elastic energy of the triangles does not get as much priority, and the shirt sags undesirably. This is especially noticeable around the lower torso and collar.

In the next section, we overcome these limitations by applying the constraints in the global step instead, and

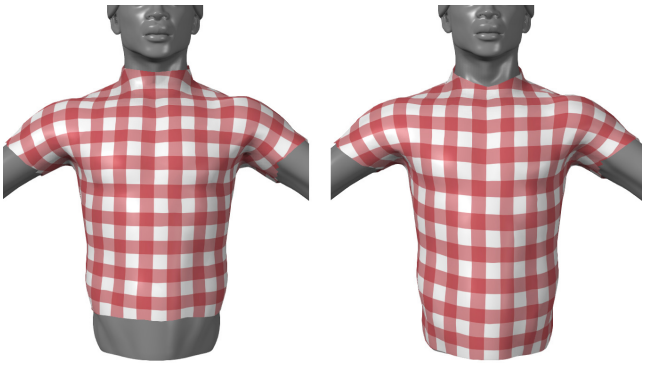


Fig. 5: Choice of weights w_i affects convergence when two forces are applied to the same element. A well chosen w_i with energy-based sliding constraints is shown on the left. Increasing this w_i (right) makes other forces behave less stiff for the same number of iterations. Constraining $f(\mathbf{x})$ instead of $g(\mathbf{z})$ does not suffer from this problem.

discuss how skin sliding and contact constraints can be represented in this formulation. While other constraints can be defined this way, we note that in the following examples only sliding and contact constraints are handled in this manner.

Two solvers for for handling these hard constraints, Uzawa conjugate gradient and graph-colored Gauss-Seidel, are presented in Sec. 5.2 and Sec. 5.3 respectively. The former is used for handling general dynamic constraints, while the latter only works for nodal constraints but is highly parallelizable.

5.1 The saddle point system

We move the constraint term U_c to act on $\mathbf{x}$ instead of $\mathbf{z}$, modifying Eq. (14a) to

$$f(\mathbf{x}) = \frac{1}{2\Delta t^2}\|\mathbf{x} - \tilde{\mathbf{x}}\|^2_{\mathbf{M}} + U_c(\mathbf{x}). \tag{43}$$

That is, we must minimize a quadratic function subject to nonlinear constraints in the global step. The constraints are linearized about the current state to obtain a linear constraint $\mathbf{Cx} = \mathbf{d}$, with $\mathbf{C}$ being an $m \times 3n$ matrix for n nodes and m constrained directions. Consequently, the $\mathbf{x}$-update becomes a linearly constrained quadratic minimization with the same system matrix in (18). For convenience, we denote

$$\mathcal{A} = \Delta t^{-2}\mathbf{M} + \mathbf{D}^T\mathbf{W}^T\mathbf{W}\mathbf{D}, \tag{44a}$$

$$\mathbf{b} = \Delta t^{-2}\mathbf{M}\tilde{\mathbf{x}} + \mathbf{D}^T\mathbf{W}^T\mathbf{W}(\mathbf{z} - \mathbf{u}) \tag{44b}$$

so that the goal of the $\mathbf{x}$-update is to minimize $\frac{1}{2}\|\mathbf{x}\|_{\mathcal{A}} - \mathbf{b}^T\mathbf{x}$ subject to $\mathbf{Cx} = \mathbf{d}$. Introducing Lagrange multipliers $\boldsymbol{\lambda}$, the solution is given by the saddle point system

$$\begin{bmatrix} \mathcal{A} & \mathbf{C}^T \\ \mathbf{C} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{d} \end{bmatrix}. \tag{45}$$

Our goal is now to solve this system in the global step of each ADMM iteration, replacing the prefactored linear solve of Eq. (18). Before moving on to solution techniques, we first give two examples of constraints using this formulation.

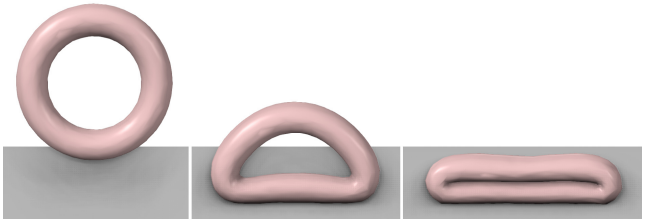


Fig. 6: A torus is dropped on a plane and temporarily flattens, demonstrating self-contact. Collisions with the ground plane and itself are represented as hard constraints.

Sliding constraints An important type of equality constraint is a sliding constraint, in which a node is constrained to lie on a surface. This is useful for modeling behavior such as skin and tight-fitting clothing. For one-way sliding on static objects, the linearization for each constrained vertex $\mathbf{x}_i$ is simply $\mathbf{n}^T(\mathbf{x}_i - \mathbf{p}_i) = \mathbf{0}$, where $\mathbf{p}_i$ is the nearest point on the object surface and $\mathbf{n}$ is its normal. Thus the ith row of $\mathbf{C}$ contains $\mathbf{n}^T$ in the appropriate location, and the ith entry of $\mathbf{d}$ is $\mathbf{n}^T\mathbf{p}_i$. To allow two-way interaction for muscle-skin simulation, we find the barycentric coordinates $\beta_1, \beta_2, \beta_3$ of $\mathbf{p}_i$ on the face j it projects to, and define the linearized constraint as $\mathbf{n}_j^T(\mathbf{x}_i - \beta_1\mathbf{x}_1 - \beta_2\mathbf{x}_2 - \beta_3\mathbf{x}_3) = \mathbf{0}$ where $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ are the vertices of the face and $\mathbf{n}_j$ is its normal. Then the rows of the constraint matrix are of the form

$$\mathbf{C}_i = [0, \ldots, \mathbf{n}_j, \ldots, -\beta_1\mathbf{n}_j, \ldots, -\beta_2\mathbf{n}_j, \ldots, -\beta_3\mathbf{n}_j, \ldots, 0] \tag{46}$$

and the corresponding entries of $\mathbf{d}$ are zero.

Collisions This approach can handle collisions with both static and dynamic obstacles in the same manner as sliding constraints. When a vertex is not penetrating an object, we set the corresponding row of $\mathbf{C}$ to zero, otherwise we set a sliding constraint to the nearest position on the surface of the object. This is not a perfect solution because collisions are better modeled via unilateral rather than equality constraints. Indeed, when objects are at rest, but in contact, the contact constraints may oscillate between active and inactive, sometimes resulting in visible jitter. Properly linearizing the inequality constraints would result in a quadratic programming problem, which is expensive to solve. Instead, we opt to alleviate the problem by treating vertices within some small distance ϵ of the object as colliding. When handling collisions in the global step, the contact normal forces are given by $f_N = -\mathbf{C}^T\boldsymbol{\lambda}$ and can be used for friction as before.

Self-collisions can be handled in the same manner as dynamic obstacles. A self-collision is detected using node-in-tetrahedron tests, and the node is projected to the nearest surface triangle of the deformed mesh. Both the collision detection and surface projection steps exclude the triangles and tetrahedra that the node belongs to. A example of this approach is shown in Fig. 6, in which a torus is dropped on a static plane and collides with itself.

Algorithm 2 Uzawa conjugate gradient algorithm

$\mathbf{x} = \mathcal{A}^{-1}(\mathbf{b} - \mathbf{C}^T\boldsymbol{\lambda})$
$\mathbf{r}_\lambda = \mathbf{C}\mathbf{x} - \mathbf{d}$ ▷ Residuals for $\boldsymbol{\lambda}$
$\mathbf{s}_\lambda = \mathbf{r}_\lambda$ ▷ Search direction for $\boldsymbol{\lambda}$
while $\mathbf{r}_\lambda > \varepsilon$ **do**
  $\mathbf{s}_\mathbf{x} = \mathcal{A}^{-1}\mathbf{C}^T\mathbf{s}_\lambda$ ▷ Search direction for $\mathbf{x}$
  $\mathbf{a}_\lambda = \mathbf{C}\mathbf{s}_\mathbf{x}$
  $\alpha = {\mathbf{s}_\lambda}^T\mathbf{r}_\lambda / {\mathbf{s}_\lambda}^T\mathbf{a}_\lambda$
  $\mathbf{x} \mathrel{-}= \alpha\mathbf{s}_\mathbf{x}$ ▷ Update $\mathbf{x}$
  $\boldsymbol{\lambda} \mathrel{+}= \alpha\mathbf{s}_\lambda$ ▷ Update $\boldsymbol{\lambda}$
  $\mathbf{r}_\lambda \mathrel{-}= \alpha\mathbf{a}_\lambda$ ▷ Update residuals
  $\beta = {\mathbf{r}_\lambda}^T\mathbf{a}_\lambda / {\mathbf{s}_\lambda}^T\mathbf{a}_\lambda$
  $\mathbf{s}_\lambda = \mathbf{r}_\lambda - \beta\mathbf{s}_\lambda$ ▷ Update search direction
end while

5.2 Uzawa conjugate gradient for general constraints

In our saddle point system (45), the constraint blocks $\mathbf{C}$ are dynamically varying, but the upper-left block $\mathcal{A}$ is constant and its Cholesky factorization is already available. We exploit this structure to solve the saddle point system by using the conjugate gradient form of the Uzawa algorithm [50]. Uzawa conjugate gradient (UCG) is commonly used for solving saddle point problems related to fluid dynamics, in particular the Stokes equations (e.g. [51]). Chen [52, Ch. 3.7.2] discusses UCG and related variants. For broader details on the Uzawa algorithm we refer the reader to [53, 54] and references therein.

The full UCG algorithm is shown in Alg. 2. Using this approach to solve Eq. (45) allows the system to reach the constrained minimum objective value much faster than energy-based hard constraints, even with a well chosen w_i, as we discuss further in Sec. 6.2. UCG allows general linear equality constraints to be added and removed dynamically without requiring the factorization of the system matrix $\mathcal{A}$ to be recomputed. However, this does come at the price of additional computational overhead: UCG requires multiple prefactored linear solves in each global step (one per conjugate gradient iteration).

To demonstrate two-way coupling constraints using UCG, multiple objects with different elastic forces are dropped on a cloth and collide in Fig. 7. Additionally, coupled muscle-skin dynamics can be represented with sliding constraints to simulate wrinkles and buckling. In Fig. 8, a hard thin shell using the Fung model is constrained to the surface of a lower stiffness neo-Hookean tetrahedral mesh. A sphere collides with the top and results natural looking wrinkles.

5.3 Parallel Gauss-Seidel for nodal constraints

While the UCG algorithm supports general linear constraints, many applications only require nodal constraints, that is, hard constraints that each apply to only one node. For example, animators often prefer direct control of a character's underlying muscles, and only desire the simulation of surface-level skin and clothing whose nodes are constrained to lie on the animated character surface. Per-node collisions with scripted obstacles can also be handled this way. In this section, we show how graph-colored parallel Gauss-Seidel can be used to solve nodal constraints efficiently.

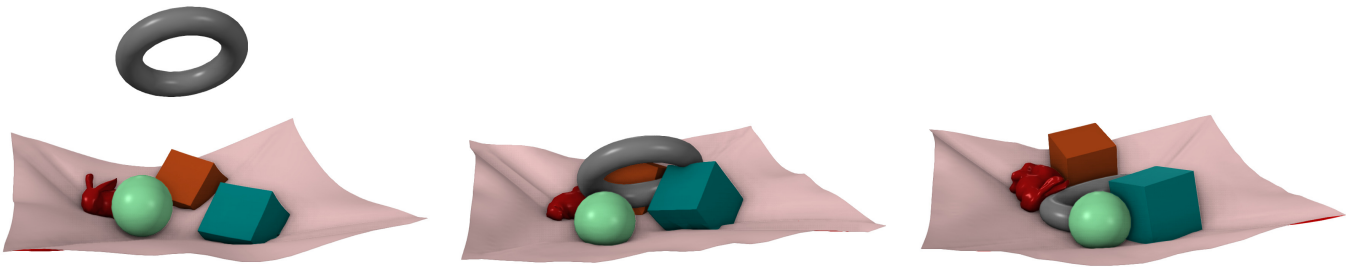


Fig. 7: Several objects with varying elastic models are piled up on a strain limited cloth with constrained corners. The bunny is neo-Hookean, the boxes are St. Venant-Kirchhoff, and rest are linear elastic.

This approach is based on the *Vivace* solver introduced by Fratarcangeli et al. [23]. Their work extends projective dynamics by parallelizing the global step. Instead of using a prefactored matrix solve, it relies on a randomized graph coloring algorithm to partition the vertices into groups. Vertices of the same group (referred to as a color) are independent, and their Gauss-Seidel update can be computed in parallel. However, the Vivace solver only models soft constraints as penalty forces, thus many iterations and high constraint stiffness may be required to avoid penetration for constraints such as sliding or collisions.

We extend Vivace to support nodal constraints. If a node is subject to a constraint of the form $\mathbf{n}_i^T(\mathbf{x}_i - \mathbf{p}_i) = \mathbf{0}$, its Gauss-Seidel update is computed in the tangent space of the constraint. We do so by constructing the 3×2 matrix $\mathbf{G}_i$ whose columns are orthogonal to each other and to $\mathbf{n}_i$. We also note the diagonal 3×3 blocks of $\mathcal{A}$ are of the form $a_{ii}\mathbf{I}$ in our formulation, so the constrained Gauss-Seidel update becomes:

$$\begin{aligned} \mathbf{x}_{\text{tmp}} &= \mathbf{G}_i^T(\mathbf{b}_i/a_{ii} - \mathcal{A}_{i:}\mathbf{x}/a_{ii} + \mathbf{x}_i^n - \mathbf{p}_i), \\ \mathbf{x}_i^{n+1} &= \mathbf{G}_i\mathbf{x}_{\text{tmp}} + \mathbf{p}_i. \end{aligned} \tag{47}$$

Successive over-relaxation can also be applied in the tangent space to speed up convergence.

Fig. 9 shows an example of an artist-animated character with a simulated elastic shirt using this approach. The shirt is able to slide across the surface of the body with appropriate surface texture deformation.

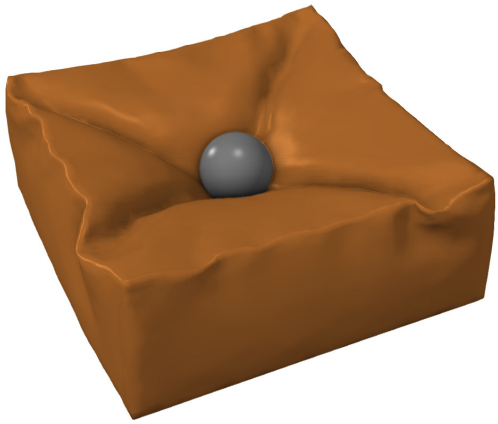


Fig. 8: A block of skin using two-way slide constraints for tissue-muscle interaction wrinkles when a force is applied. The exterior membrane uses the Fung model and the interior is neo-Hookean.

6 Analysis and Discussion

In Sec. 6.1 we compare our method with projective dynamics and LBFG-S. The convergence of nodal constraints using UCG, parallel Gauss-Seidel, and energy based constraints is presented in Sec. 6.2. We define relative error as $\big(\epsilon(\mathbf{x}^n) - \epsilon(\mathbf{x}^*)\big)/\big(\epsilon(\mathbf{x}^0) - \epsilon(\mathbf{x}^*)\big)$, where $\epsilon(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{D}\mathbf{x})$, and the reference value $\epsilon(\mathbf{x}^*)$ is the value of the objective after 1000 iterations.

6.1 Comparison with projective dynamics

We show in Sec. 4.1 and Appendix B that our ADMM formulation is identical to projective dynamics when the constraint manifolds are affine. As further validation, in Fig. 10 (left) we compare the convergence of projective dynamics, ADMM, and L-BFGS on a simple nonlinear problem, namely a corotated linear elastic material with a twisted initial condition. For projective dynamics, we used the implementation in the ShapeOp library [55]. While our proof of equivalence does not apply to this problem because of the presence of rotations, we can see that with the choice $w_i = \sqrt{k_i}$ ADMM and projective dynamics take nearly the same steps towards convergence (the difference in run time is likely an artifact of implementation details). Interestingly, with a lower weight $w_i = \frac{1}{2}\sqrt{k_i}$, ADMM turns out to converge faster than projective dynamics; we discuss this further in Sec. 7. The L-BFGS method applied directly to the original problem (6) is slower to converge than both projective dynamics and ADMM in terms of computation time, because each iteration is more expensive.

6.2 Constrained global step convergence

The convergence of presented methods for handling nodal collisions are shown in Fig. 10 (center) and nodal sliding constraints (right). These include ADMM using a prefactored Cholesky solve, Uzawa conjugate gradient, and parallel Gauss-Seidel. For the energy-based constraints, we used the value of w_i which empirically gave the best convergence. Both iterative solvers were ran for the full number of iterations in which early exit was not used. The measured run time includes collision detection, the local step, and the global step. As the constraints may not be exactly satisfied

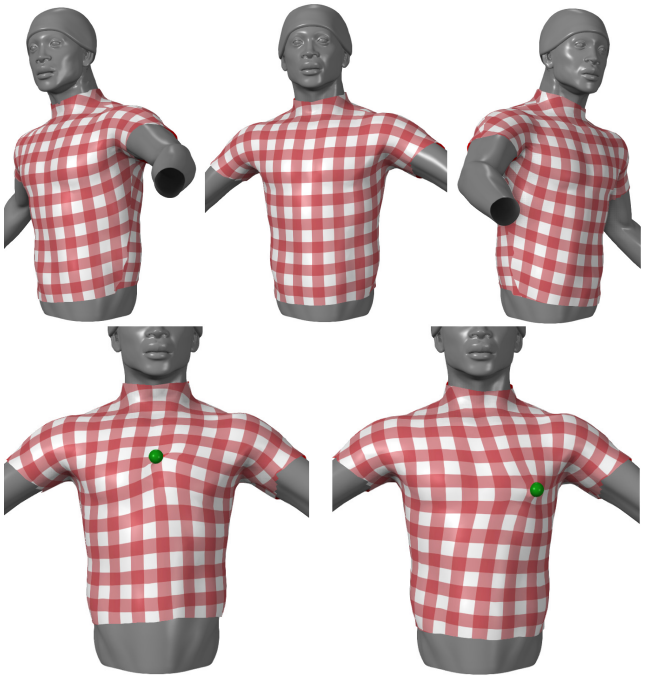


Fig. 9: An artist-animated character with simulated elastic shirt using sliding constraints. As the torso twists and turns (top), the shirt is able to slide over its features. Pulling on the strain limited shirt (bottom) causes sliding throughout the entire shirt mesh.

at the end of an iteration, we temporarily project nodes to a collision-free state when computing the objective.

In the collision example (center), a volumetric sphere mesh with linear tetrahedral forces rests on a frictionless, rigid plane subject to gravity. We take a large time step $\Delta t = 0.2$s, and compute the value of the objective at each ADMM iteration. Despite the overhead of multiple linear solves in the global step, the UCG algorithm is able to reach lower error faster than other approaches. Interestingly, increasing the number of conjugate gradient iterations slowed convergence with respect to run time, and did not have a significant impact overall. After about 20 iterations, the UCG solver stalls and is unable to make significant further progress. This may be attributable to the fact that the UCG algorithm is not designed for unilateral constraints. Parallel constrained Gauss-Seidel with 30 iterations spends a little more time to reach the same relative error as UCG, but is able to continue toward a lower objective. This highlights parallel Gauss-Seidel as an attractive approach for nodal constraints.

The skin slide convergence plot (right) is for a time step using the same configuration as Fig. 9. Like the collision example, both parallel Gauss-Seidel and UCG are able to achieve a low relative error faster than the Cholesky solve with energy-based sliding constraints. However, the convergence of UCG is not monotonic if the number of iterations is too small, as the saddle point system is not solved to sufficient accuracy. With respect to runtime, both UCG and Gauss-Seidel perform similarly. Because graph-colored Gauss-Seidel is highly parallel, we expect the run time performance to increase with additional parallel hardware.

6.3 Performance

We tested the performance of our unoptimized implementation on a 3.5GHz Intel Xeon E5-1650 with 6 cores and hyperthreading. A k-d tree is used for all demos that include sliding and collision constraints. The local step was parallelized with OpenMP, and the global step was either solved using a precomputed sparse Cholesky factorization of the system matrix or the methods described in Sec. 5. Runtime performance numbers of all our examples are presented in Table 2. Figs. 1, 2, and 4 were rendered with one iteration of Loop subdivision.

As the "Untwist" example shows, the cost of the local step is linear in the number of elements, while that of the global step is slightly superlinear due to the prefactored linear solve. It can also be seen that the local step becomes more expensive when using general nonlinear materials, such as the neo-Hookean model, than when using the simpler projective dynamics model. This is mainly due to the cost of computing the proximal operator (36). Nevertheless, as the local step is fully parallel, its performance will improve directly with greater hardware parallelism. Another simple avenue for acceleration would be to precompute the proximal operator into a lookup table for use at runtime.

7 Limitations and Future Work

Convergence Proving the convergence of ADMM on nonconvex problems is an area of active research. Recent results have proven convergence for sufficiently large ρ under certain additional conditions [38, 39, 40]. These correlate with our empirical observations that convergence may fail for small ρ, as discussed in Sec. 7, although the assumptions required by existing theoretical results are too restrictive to apply to our application. We hope that future developments in this area will provide more general convergence guarantees.

A limitation of our approach is that while the algorithm converges for a broad range of choices of the weights w_i, in practice the rate of convergence can depend significantly on the choice of w_i. For energy terms of the form (29), choosing $w_i^2 = k_i$ gives convergence nearly identical to that of projective dynamics. For more general energies, we have similarly observed reliable convergence when w_i^2 is close to the effective stiffness of the energy, that is, when both terms in the $\mathbf{z}$-step (27) have similar curvature. Fig. 11 shows that reducing w_i can in fact lead to much faster convergence, but if the weight is extremely small the method can fail to converge because the nonconvexity of the problem becomes too severe. Determining the choice of w_i for reliably fast convergence is a question we hope to address in future work.

Future work It is known that for convex, potentially non-smooth functions, ADMM has an $O(1/n)$ convergence rate [56]. Recently work in optimization has proposed variations on ADMM that have faster convergence rates [32, 33, 34]. Our work opens up the possibility of exploiting these and future advances in optimization to speed up physics-based animation. In the future we also hope to explore applications of Chebyshev acceleration [25] to our algorithm.

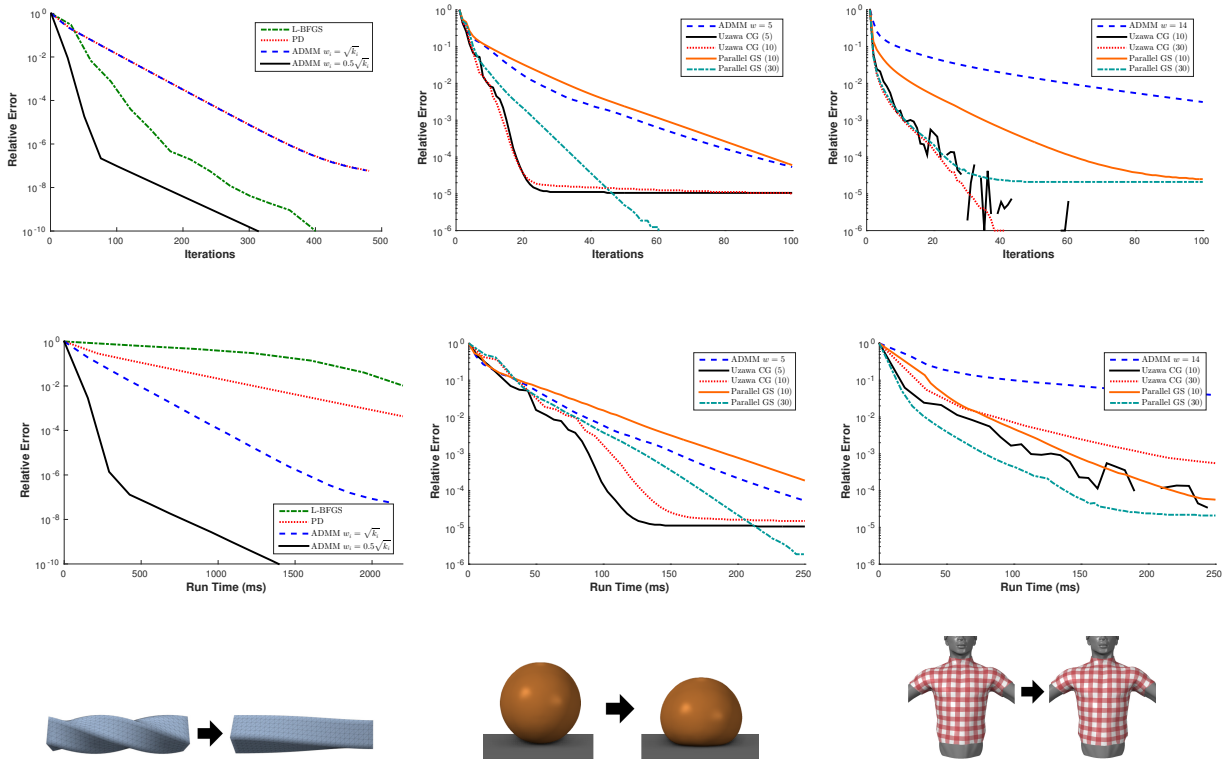


Fig. 10: (Left) Convergence of projective dynamics, ADMM, and L-BFGS in the presence of rotations. A deformed box partially untwists over a time step of $\Delta t = 0.04\,\mathrm{s}$. For L-BFGS, we used a history window of size 10. (Center) Convergence of collision constraints with Uzawa conjugate gradient, parallel Gauss-Seidel, and energy based constraints. A sphere subject to gravity collides with a surface at a large time step of $\Delta t = 0.2\,\mathrm{s}$. The numbers in parentheses represent the number of inner iterations used by each approach. (Right) Convergence of slide constraints for one iteration of the shirt demo in Fig. 9.

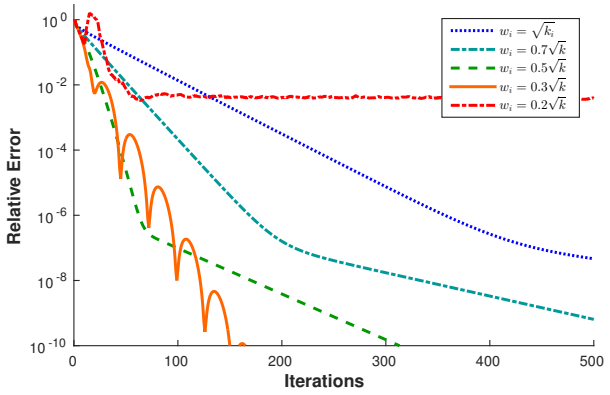


Fig. 11: Convergence of the untwisting box example from Fig. 10 (left), using ADMM with different values of w_i. Reducing w_i can improve the convergence rate, but if it is too low, ADMM will not make progress after a certain number of iterations.

An important application of nonlinear constitutive models is the use of data-driven materials acquired from real objects. Unfortunately, with the exception of the recent work of [4], most existing techniques [1, 2, 3] represent the acquired material as a stress-strain response function, which may not necessarily correspond to a hyperelastic (conservative) material. Finding a hyperelastic model whose stress-strain response best fits a given response function would allow existing databases of acquired nonlinear materials to be used in optimization-based integration techniques such as ours.

The Uzawa conjugate gradient algorithm is one of multiple ways to solve Eq. (45). Its application to unilateral constraints such as contact works well in most cases, but may limit convergence to the true solution as shown in Fig. 10 (center). It can also result in jittering in animations due to constraints oscillating between active and inactive states, which is especially likely to occur in layered and self-colliding cloth. Another limitation with our application of UCG is that it requires linearizing the constraints, which may be a poor approximation for highly curved regions. We hope to address these limitations with improved solvers for Eq. (43). For instance, we would like to compare UCG to optimized saddle point solvers, such as PARDISO [57], and incorporate line search or other step size control techniques to better handle nonlinear constraints.

8 Conclusion

We have shown how ADMM, a simple, versatile, and scalable optimization algorithm, can be fruitfully applied to time integration for physics-based animation. The resulting algorithm reduces to a method nearly identical to projective dynamics for simple energy functions, and generalizes to

Example	#nodes	#elems	#iters	Global solver	Local step (ms)	Global step (ms)	Collision detection (ms)	Time per frame (ms)
Armadillo (Fig. 1)	919	2761	7	PC	2.6	0.82	—	35
Bunny (Fig. 2)	777	2510	10	PC	2.9	0.8	—	38
Flag (Fig. 3)	1251	2400	20	PC	0.68	0.63	—	28
Horse (Fig. 4)	962	3221	13	UCG (10)	0.69	2.3	0.73	49
Torus (Fig. 6)	798	4130	30	UCG (10)	4.1	2.7	1.1	281
Cloth Pile (Fig. 7)	4559	14394	20	UCG (10)	9.2	10.8	7.1	165
Wrinkles (Fig. 8)	7257	17677	40	UCG (10)	48	32	12	3733
Shirt (Fig. 9)	1933	3599	40	PGS (30)	0.43	2.1	3.9	160
Untwist (Fig. 10) — low res, linear	367	1562	20	PC	0.57	0.26	—	17
— medium res, linear	868	4170	20	PC	1.5	0.81	—	47
— high res, linear	1727	9031	20	PC	3.4	1.9	—	107
— high res, neo-Hookean	1727	9031	20	PC	10.8	2.0	—	633

TABLE 2: Run-time performance numbers for our examples. Each example iterates over a time step of 40 ms except Fig. 1, which has a time step of 80 ms and Figs. 7, 6, and 8 which have a time step of 20 ms. The times listed for collisions, local, and global steps are per iteration. Not shown in the table but included in time per frame is the k-d tree update. The global step is computed with a prefactored Cholesky solve (PC), Uzawa conjugate gradient (UCG), or parallel Gauss-Seidel (PGS).

realistic nonlinear energies while retaining its parallelizability and robustness. We have described how arbitrary constitutive models can be incorporated in our approach without modification, requiring only the solution of a small 3-dimensional (for isotropic materials) or 9-dimensional (for anisotropic materials) optimization problem for each element. By resolving constraints in the global linear solve, we can reach a lower objective value much faster than energy based constraints, even when the solver is not run to convergence. This makes our method an effective technique for highly parallel simulation of realistic deformable objects.

ACKNOWLEDGEMENTS

We would like to thank Moses Adeagbo for the character animation used in Fig. 9, and Arindam Banerjee and Ioannis Karamouzas for insightful discussions.

REFERENCES

[1] B. Bickel, M. Bächer, M. A. Otaduy, W. Matusik, H. Pfister, and M. Gross, "Capture and modeling of non-linear heterogeneous soft tissue," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 89:1–89:9, Jul. 2009.

[2] H. Wang, J. F. O'Brien, and R. Ramamoorthi, "Data-driven elastic models for cloth: Modeling and measurement," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 71:1–71:12, Jul. 2011.

[3] E. Miguel, D. Bradley, B. Thomaszewski, B. Bickel, W. Matusik, M. A. Otaduy, and S. Marschner, "Data-driven estimation of cloth simulation models," *Comp. Graph. Forum*, vol. 31, no. 2.2, pp. 519–528, 2012.

[4] E. Miguel, D. Miraut, and M. A. Otaduy, "Modeling and estimation of energy-based hyperelastic objects," *Computer Graphics Forum (Proc. of Eurographics)*, vol. 35, no. 2, 2016.

[5] M. Mller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *J. Vis. Comm. and Img. Rep.*, vol. 18, no. 2, pp. 109 – 118, 2007.

[6] J. Bender, M. Mller, M. A. Otaduy, M. Teschner, and M. Macklin, "A survey on position-based simulation methods in computer graphics," *Comp. Graph. Forum*, vol. 33, no. 6, pp. 228–251, 2014.

[7] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly, "Projective dynamics: Fusing constraint projections for fast simulation," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 154:1–154:11, Jul. 2014.

[8] R. Narain, M. Overby, and G. E. Brown, "ADMM $\supseteq$ projective dynamics: Fast simulation of general constitutive models," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '16, Eurographics. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2016, pp. 21–28.

[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[10] H. Xu, F. Sin, Y. Zhu, and J. Barbič, "Nonlinear material design using principal stretches," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 75:1–75:11, Jul. 2015.

[11] O. Rémillard and P. G. Kry, "Embedded thin shells for wrinkle simulation," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 50:1–50:8, Jul. 2013.

[12] P. Li and P. G. Kry, "Multi-layer skin simulation with adaptive constraints," in *Proceedings of the Seventh International Conference on Motion in Games*, ser. MIG '14. New York, NY, USA: ACM, 2014, pp. 171–176.

[13] D. Li, S. Sueda, D. R. Neog, and D. K. Pai, "Thin skin elastodynamics," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 49:1–49:10, Jul. 2013.

[14] M. Müller, N. Chentanez, T.-Y. Kim, and M. Macklin, "Strain based dynamics," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '14, Eurographics. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2014, pp. 149–157.

[15] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," in *Proc. Conf. on Comp. Graph. and Interactive Tech.*, ser. SIGGRAPH. ACM, 1987, pp. 205–214.

[16] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proc. Conf. on Comp. Graph. and Interactive Tech.*, ser. SIGGRAPH. ACM, 1998, pp. 43–54.

[17] L. Kharevych, W. Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun, "Geometric, variational

integrators for computer animation," in *Proc. ACM SIGGRAPH/Eurographics SCA*, ser. SCA. Eurographics Association, 2006, pp. 43–51.

[18] S. Martin, B. Thomaszewski, E. Grinspun, and M. Gross, "Example-based elastic materials," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 72:1–72:8, Jul. 2011.

[19] T. F. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J. M. Teran, "Optimization integrator for large time steps," *IEEE Trans. on Vis. and Comp. Graph.*, vol. 21, no. 10, pp. 1103–1115, Oct 2015.

[20] T. Liu, A. W. Bargteil, J. F. O'Brien, and L. Kavan, "Fast simulation of mass-spring systems," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 214:1–214:7, Nov. 2013.

[21] S. Bouaziz, M. Deuss, Y. Schwartzburg, T. Weise, and M. Pauly, "Shape-up: Shaping discrete geometry with projections," *Comp. Graph. Forum*, vol. 31, no. 5, pp. 1657–1667, Aug. 2012.

[22] M. Weiler, D. Koschier, and J. Bender, "Projective fluids," in *Proceedings of the 9th International Conference on Motion in Games*, ser. MIG '16. New York, NY, USA: ACM, 2016, pp. 79–84.

[23] M. Fratarcangeli, V. Tibaldo, and F. Pellacini, "Vivace: A practical gauss-seidel method for stable soft body dynamics," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 214:1–214:9, Nov. 2016.

[24] M. Macklin, M. Müller, and N. Chentanez, "Xpbd: Position-based simulation of compliant constrained dynamics," in *Proceedings of the 9th International Conference on Motion in Games*, ser. MIG '16. New York, NY, USA: ACM, 2016, pp. 49–54.

[25] H. Wang, "A chebyshev semi-iterative approach for accelerating projective and position-based dynamics," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 246:1–246:9, Oct. 2015.

[26] T. Liu, S. Bouaziz, and L. Kavan, "Quasi-newton methods for real-time simulation of hyperelastic materials," *ACM Trans. Graph.*, vol. 36, no. 3, pp. 23:1–23:16, May 2017.

[27] J. Nocedal and S. Wright, *Numerical Optimization*, ser. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[28] E. Esser, X. Zhang, and T. F. Chan, "A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science," *SIAM J. Img. Sci.*, vol. 3, no. 4, pp. 1015–1046, Dec. 2010.

[29] D. O'Connor and L. Vandenberghe, "Primal-dual decomposition by operator splitting and applications to image deblurring," *SIAM J. Img. Sci.*, vol. 7, no. 3, pp. 1724–1754, 2014.

[30] P. Tseng, "Applications of a splitting algorithm to decomposition in convex programming and variational inequalities," *SIAM Journal on Control and Optimization*, vol. 29, no. 1, pp. 119–138, 1991. [Online]. Available: http://dx.doi.org/10.1137/0329006

[31] M. Zhu and T. Chan, "An efficient primal-dual hybrid gradient algorithm for total variation image restoration," University of California, Los Angeles, Tech. Rep., 2008.

[32] D. Goldfarb, S. Ma, and K. Scheinberg, "Fast alternating linearization methods for minimizing the sum of two convex functions," *Math. Programming*, vol. 141, no. 1, pp. 349–382, 2012.

[33] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM J. Img. Sci.*, vol. 7, no. 3, pp. 1588–1623, 2014.

[34] M. Kadkhodaie, K. Christakopoulou, M. Sanjabi, and A. Banerjee, "Accelerated alternating direction method of multipliers," in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, ser. KDD. ACM, 2015, pp. 497–506.

[35] P. A. Forero, A. Cano, and G. B. Giannakis, "Distributed clustering using wireless sensor networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 707–724, Aug 2011.

[36] Z. Wen, C. Yang, X. Liu, and S. Marchesini, "Alternating direction methods for classical and ptychographic phase retrieval," *Inverse Problems*, vol. 28, no. 11, p. 115010, 2012.

[37] D. L. Sun and C. Fvotte, "Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 6201–6205.

[38] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," in *IEEE Intl. Conf. Acoustics, Speech and Signal Proc. (ICASSP)*, April 2015, pp. 3836–3840.

[39] G. Li and T. K. Pong, "Global convergence of splitting methods for nonconvex composite optimization," *SIAM Journal on Optimization*, vol. 25, no. 4, pp. 2434–2460, 2015.

[40] S. Magnsson, P. C. Weeraddana, M. G. Rabbat, and C. Fischione, "On the convergence of alternating direction lagrangian methods for nonconvex structured optimization problems," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 3, pp. 296–309, Sept 2016.

[41] J. Gregson, I. Ihrke, N. Thuerey, and W. Heidrich, "From capture to simulation: Connecting forward and inverse problems in fluids," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 139:1–139:11, Jul. 2014.

[42] R. Narain, A. Samii, and J. F. O'Brien, "Adaptive anisotropic remeshing for cloth simulation," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 152:1–152:10, Nov. 2012.

[43] R. Narain, R. A. Albert, A. Bulbul, G. J. Ward, M. S. Banks, and J. F. O'Brien, "Optimal presentation of imagery with focus cues on multi-plane displays," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 59:1–59:12, Jul. 2015.

[44] D. M. Kaufman, R. Tamstorf, B. Smith, J.-M. Aubry, and E. Grinspun, "Adaptive nonlinearity for collisions in complex rod assemblies," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 123:1–123:12, Jul. 2014.

[45] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, Jan. 2014.

[46] G. Irving, J. Teran, and R. Fedkiw, "Invertible finite elements for robust simulation of large deformation," in *Proc. ACM SIGGRAPH/Eurographics SCA*, ser. SCA. Eurographics Association, 2004, pp. 131–140.

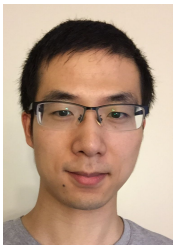
[47] A. Stomakhin, R. Howes, C. Schroeder, and J. M. Teran, "Energetically consistent invertible elasticity," in *Proc. ACM SIGGRAPH/Eurographics SCA*, ser. SCA. Eurographics Association, 2012, pp. 25–32.

[48] Y.-c. Fung, *Biomechanics: mechanical properties of living*

tissues. Springer Science & Business Media, 2013.
[49] I. Karamouzas, N. Sohre, R. Narain, and S. J. Guy, "Implicit crowds: Optimization integrator for robust crowd simulation," *ACM Transactions on Graphics*, vol. 36, no. 4, Jul. 2017.
[50] H. Uzawa, "Iterative methods for concave programming," in *Studies in Linear and Nonlinear Programming*, 1958, pp. 154–165.
[51] F. Bertrand and P. A. Tanguy, "Krylov-based uzawa algorithms for the solution of the stokes equations using discontinuous-pressure tetrahedral finite elements," *Journal of Computational Physics*, vol. 181, no. 2, pp. 617–638, 2002.
[52] Z. Chen, *Finite element methods and their applications*. Springer Science & Business Media, 2005.
[53] M. Benzi, G. H. Golub, and J. Liesen, "Numerical solution of saddle point problems," *Acta Numerica*, vol. 14, pp. 1–137, 2005.
[54] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev, "Analysis of the inexact uzawa algorithm for saddle point problems," *SIAM Journal on Numerical Analysis*, vol. 34, no. 3, pp. 1072–1092, 1997.
[55] M. Deuss, A. H. Deleuran, S. Bouaziz, B. Deng, D. Piker, and M. Pauly, "Shapeop—a robust and extensible geometric modelling paradigm," in *Modelling Behaviour: Design Modelling Symposium 2015*, R. M. Thomsen, M. Tamke, C. Gengnagel, B. Faircloth, and F. Scheurer, Eds. Springer International Publishing, 2015, pp. 505–515.
[56] H. Wang and A. Banerjee, "Online alternating direction method," in *Proc. Intl. Conf. Mach. Learn.*, 2013.
[57] C. G. Petra, O. Schenk, and M. Anitescu, "Real-time stochastic optimization of complex energy systems on high-performance computers," *IEEE Computing in Science & Engineering*, vol. 16, no. 5, pp. 32–42, 2014.

Matthew Overby received the B.S. and M.S. degrees in Computer Science from the University of Minnesota Duluth. He was briefly employed at the University of Utah, where he researched methods for fast simulation of urban heat transfer. He is now pursuing the Ph.D. degree in Computer Science at the University of Minnesota. His area of research is physics-based simulation, with interests in muscle and skin dynamics for character and creature animation.

George Brown received B.S. degrees in Physics and Astrophysics from the University of Minnesota and his M.S. degree in Physics from the University of Florida. He also received his M.S. degree in Computer Science from the University of Minnesota, where he is currently working towards his Ph.D. His research interests include simulation of cloth, contact and friction, fracture, and real-time simulation of hyperelastic constitutive models.



Jie Li received a Bachelor of Engineering in Computer Science from Beijing Forestry University. He is now pursuing his Ph.D. degree in Computer Science at the University of Minnesota. His research interests include physics-based animation and computational optimization. He is currently working on a contact and friction solver and its application in cloth simulation.

Rahul Narain received the B.Tech. degree from the Indian Institute of Technology, Delhi, and the M.S. and Ph.D. degrees in Computer Science from the University of North Carolina at Chapel Hill. He is currently an assistant professor in the Department of Computer Science and Engineering at the University of Minnesota, Twin Cities. His primary area of interest is computer animation, particularly focusing on numerical methods for physics-based simulation, multi-agent navigation, and computational displays.