Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2016)
Ladislav Kavan and Chris Wojtan (Editors)

ADMM $\supseteq$ Projective Dynamics: Fast Simulation of General Constitutive Models

Rahul Narain   Matthew Overby   George E. Brown

University of Minnesota

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 demonstrate these benefits on several examples that include cloth, collisions, and volumetric deformable bodies with nonlinear elasticity.

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 [BBO*09, WOR11, MBT*12]. However, general nonlinear materials are difficult to simulate robustly in real time. Recent algorithms for real-time simulation, such as position-based dynamics [MHHR07, BMO*14] and projective dynamics [BML*14] 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, we apply the *alternating direction method of multipliers* (ADMM) optimization algorithm [BPC*11] to the time integration of nonlinear elastic forces, and derive a novel simulation algorithm. Our method generalizes projective dynamics to arbitrary conservative forces, including nonlinear elasticity and hard constraints, while retaining its speed, parallelizability, and robustness (see Figure 1). 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.

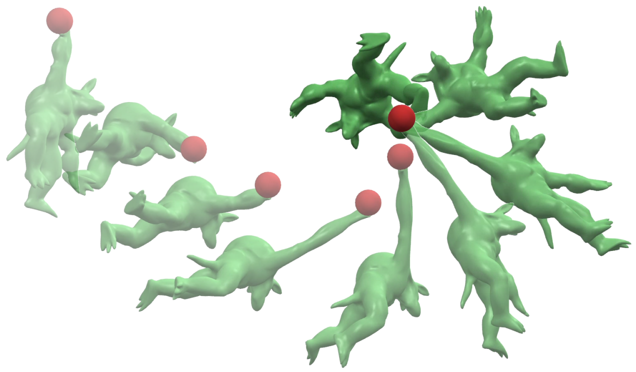

Figure 1: *A neo-Hookean armadillo with 2.8k tetrahedra simulated interactively at 15 fps using our method.*

2. Related work

Nonlinear constitutive 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 [BBO*09] and cloth [WOR11, MBT*12] from observed data; their results show that nonlinearity leads to visually more realistic behavior than linear elastic models. Furthermore, Xu et al. [XSZB15] recently showed that allowing artists to directly specify the nonlinear elastic response of a simulated material enables desirable animation effects to be achieved.

Parallel time integration: Implicit time integration techniques such as the backward Euler method have long been popular in physics-based animation [TPBF87, BW98] 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 of backward Euler integration [KYT*06, MTGG11, GSS*15], and perform the optimization by alternating between local and global steps. Liu et al. [LBOK13] proposed a block coordinate descent approach for mass-spring systems. The projective dynamics algorithm of Bouaziz et al. [BML*14] 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 [BDS*12] for geometry manipulation using shape constraints. However, both aforementioned integration techniques [LBOK13, BML*14] 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 [MHHR07, BMO*14], which models all forces as hard constraints and solves them using Gauss-Seidel iterations. Recently, Wang [Wan15] proposed the use of the Chebyshev semi-iterative technique to accelerate the convergence of both projective dynamics and position-based dynamics. Their technique is orthogonal to our work, and could potentially be applied to our algorithm as well.

Concurrently to our work, Liu et al. [LBK16] 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 [NW06].

Primal-dual algorithms for optimization: In the convex optimization literature, primal-dual methods [EZC10, OV14] such as ADMM [BPC*11] and PDHG [ZC08] 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.

Such algorithms have begun to find use in graphics as well. In their fluid tracking algorithm, Gregson et al. [GITH14] 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 [NSO12]. Another primal-dual algorithm known as PDHG was used by Narain et al. [NAB*15] 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

We consider a physical system with positions $\mathbf{x} \in \mathbb{R}^d$, velocities $\mathbf{v} \in \mathbb{R}^d$, and inertia matrix $\mathbf{M}$. The forces acting on the system can be separated into conservative forces $\mathbf{f} = -\nabla U(\mathbf{x})$ induced by a potential energy $U(\mathbf{x})$, and other potentially non-conservative external forces $\mathbf{f}_{\text{ext}}$. We will perform backward Euler integration on the conservative forces $\mathbf{f}$ and treat the other forces $\mathbf{f}_{\text{ext}}$ (including gravity) explicitly. In the minimization form of backward Euler time integration [MTGG11, GSS*15], the state of the system $\mathbf{x}$ at the next time step $t+\Delta t$ is given by

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

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

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

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 [BML*14], 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 $\mathbf{D}_i\mathbf{x}$. Then we have

$$U(\mathbf{x}) = \sum_{i=1}^{m} U_i(\mathbf{D}_i\mathbf{x}), \tag{3}$$

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 could 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

$$U_*\left(\begin{bmatrix} \mathbf{D}_1\mathbf{x} \\ \mathbf{D}_2\mathbf{x} \\ \vdots \\ \mathbf{D}_m\mathbf{x} \end{bmatrix}\right) = \sum_{i=1}^{m} U_i(\mathbf{D}_i\mathbf{x}), \tag{4}$$

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

$$\min_{\mathbf{x}} \frac{1}{2\Delta t^2} \|\mathbf{M}^{1/2}(\mathbf{x} - \tilde{\mathbf{x}})\|^2 + U_*(\mathbf{D}\mathbf{x}). \tag{5}$$

3.1. ADMM

The alternating direction method of multipliers [BPC*11] is a method for solving optimization problems of the form

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{z}} \; f(\mathbf{x}) + g(\mathbf{z}) \\ &\text{s.t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}. \end{aligned} \tag{6}$$

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

$$\mathbf{x}^{n+1} = \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}^n - \mathbf{c} + \mathbf{u}^n\|^2 \right), \tag{7}$$

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

$$\mathbf{u}^{n+1} = \mathbf{u}^n + (\mathbf{A}\mathbf{x}^{n+1} + \mathbf{B}\mathbf{z}^{n+1} - \mathbf{c}) \tag{9}$$

for some step length parameter $\rho > 0$. Here superscripts on the variables denote the iteration number. When f and g are convex, ADMM is known to converge to the optimal solution for any $\rho > 0$ under very mild assumptions [BPC*11]. While the energy functions U_i used in physics-based animation are typically not convex, ADMM has in practice also been observed to perform well on nonconvex problems; we refer the reader to [BPC*11, Sec. 9] and references in [HLR15] for more details.

Our problem (5) is of the form (6) if we set

$$f(\mathbf{x}) = \frac{1}{2\Delta t^2}\|\mathbf{M}^{1/2}(\mathbf{x}-\tilde{\mathbf{x}})\|^2, \tag{10}$$
$$g(\mathbf{z}) = U_*(\mathbf{z}), \tag{11}$$
$$\mathbf{A} = \mathbf{WD}, \tag{12}$$
$$\mathbf{B} = -\mathbf{W}, \tag{13}$$
$$\mathbf{c} = \mathbf{0}, \tag{14}$$

for any invertible matrix $\mathbf{W}$. Thus the constraint is $\mathbf{W}(\mathbf{Dx}-\mathbf{z}) = \mathbf{0}$, with $\mathbf{W}$ acting as a weighting parameter that does not change the solution but can be used to improve the convergence rate (see Section 5). To remain compatible with the separability of U_*, we will take $\mathbf{W}$ to be block diagonal, with blocks $\mathbf{W}_i$ the same size as the corresponding $\mathbf{z}_i$. In practice, it is convenient to use blocks of the form $\mathbf{W}_i = w_i\mathbf{I}$, and we will assume the same in the remainder of the paper.

After introducing $\bar{\mathbf{u}} = \mathbf{W}^{-1}\mathbf{u}$, the ADMM update rules (7)–(9) applied to (10)–(14) become

$$\mathbf{x}^{n+1} = (\mathbf{M}+\rho\Delta t^2\mathbf{D}^T\mathbf{W}^T\mathbf{WD})^{-1}\left(\mathbf{M}\tilde{\mathbf{x}}+\rho\Delta t^2\mathbf{D}^T\mathbf{W}^T\mathbf{W}(\mathbf{z}^n-\bar{\mathbf{u}}^n)\right), \tag{15}$$
$$\mathbf{z}^{n+1} = \arg\min_{\mathbf{z}}\left(U_*(\mathbf{z})+\frac{\rho}{2}\|\mathbf{W}(\mathbf{Dx}^{n+1}-\mathbf{z}+\bar{\mathbf{u}}^n)\|^2\right), \tag{16}$$
$$\bar{\mathbf{u}}^{n+1} = \bar{\mathbf{u}}^n+\mathbf{Dx}^{n+1}-\mathbf{z}^{n+1}. \tag{17}$$

As ρ can be absorbed into $\mathbf{W}$ via $\mathbf{W} \leftarrow \sqrt{\rho}\mathbf{W}$, we will omit it from here on.

As the algorithm proceeds, $\mathbf{x}^n$ and $\mathbf{z}^n$ converge to the optimal solution $\mathbf{x}^*$ and the corresponding local coordinates $\mathbf{z}^* = \mathbf{Dx}^*$ respectively. To understand the role of $\bar{\mathbf{u}}^n$, we observe that if U_* is smooth, differentiating (16) yields

$$\nabla U_*(\mathbf{z}^{n+1}) = \mathbf{W}^T\mathbf{W}(\bar{\mathbf{u}}^n+\mathbf{Dx}^{n+1}-\mathbf{z}^{n+1}) \tag{18}$$
$$= \mathbf{W}^T\mathbf{W}\bar{\mathbf{u}}^{n+1}. \tag{19}$$

Thus $\bar{\mathbf{u}}^n$ encodes a (weighted) running estimate of all the forces in the system. To be precise, at convergence the net force is nothing but

$$-\nabla U(\mathbf{x}^*) = -\mathbf{D}^T\nabla U_*(\mathbf{z}^*) \tag{20}$$
$$= -\mathbf{D}^T\mathbf{W}^T\mathbf{W}\bar{\mathbf{u}}^*. \tag{21}$$

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*

Algorithm 1 The ADMM algorithm for implicit time integration.

```
1: Initialize $\mathbf{x} = \tilde{\mathbf{x}}$ and $\mathbf{u} = \mathbf{0}$
2: loop
3:     for all energy terms $i$ in parallel do
4:         Compute $\mathbf{D}_i\mathbf{x}+\mathbf{u}_i$
5:         Update $\mathbf{z}_i$ using (24)
6:         Update $\mathbf{u}_i$ using (25)
7:     end for
8:     Update $\mathbf{x}$ using (15)
9: end loop
```

residuals [BPC*11],

$$\mathbf{r}^{n+1} = \mathbf{W}(\mathbf{Dx}^{n+1}-\mathbf{z}^{n+1}), \tag{22}$$
$$\mathbf{s}^{n+1} = \mathbf{D}^T\mathbf{W}^T\mathbf{W}(\mathbf{z}^{n+1}-\mathbf{z}^n), \tag{23}$$

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

3.2. Implementation

In this section we provide some notes on our implementation, and highlight the conceptual simplicity and parallelizability of our algorithm.

In the $\mathbf{x}$-update (15), the matrix $\mathbf{M}+\Delta t^2\mathbf{D}^T\mathbf{W}^T\mathbf{WD}$ has the same structure as the global matrix used in projective dynamics. In particular, it is constant, symmetric, and positive definite. We precompute and store its Cholesky factorization, rendering the global step extremely fast.

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:

$$\mathbf{z}_i^{n+1} = \arg\min_{\mathbf{z}_i}\left(U_i(\mathbf{z}_i)+\frac{1}{2}\|\mathbf{W}_i(\mathbf{D}_i\mathbf{x}^{n+1}-\mathbf{z}_i+\bar{\mathbf{u}}_i^n)\|^2\right), \tag{24}$$
$$\bar{\mathbf{u}}_i^{n+1} = \bar{\mathbf{u}}_i^n+\mathbf{D}_i\mathbf{x}^{n+1}-\mathbf{z}_i^{n+1}. \tag{25}$$

To apply our algorithm to a specific simulation problem, the main task is to implement a solver for (24) for each energy term U_i in the simulation. When $\mathbf{W}_i = w_i\mathbf{I}$ is a multiple of the identity, (24) can be interpreted as a *proximal operator* [PB14] for U_i, and 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 does not in general have a closed-form solution, for typical energy terms it is very low-dimensional and therefore feasible to solve numerically or using precomputation. In Section 4.1, we show how an existing projective dynamics implementation can easily be adapted to perform the $\mathbf{z}$-update in our algorithm. For more complex energies that do not fit into the projective dynamics framework, we have found a few (≤ 10) iterations of L-BFGS [NW06] to be sufficient.

The entire procedure for implicit integration using ADMM is summarized in Algorithm 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.

4. Applications and results

In this section, we describe some applications of our ADMM-based algorithms to specific energy functions, including both the simplified energies of the form used in projective dynamics, general constitutive models applied to finite elements, and hard constraints.

4.1. Projective dynamics as a special case of ADMM

Projective dynamics solves the optimization problem (5) under the assumption that the energy terms are of the form

$$U_i(\mathbf{D}_i\mathbf{x}) = \min_{\mathbf{p}_i\in\mathcal{C}_i} \frac{k_i}{2}\|\mathbf{D}_i\mathbf{x}-\mathbf{p}_i\|^2 \tag{26}$$

for some constraint manifolds $\mathcal{C}_i$ and 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),

$$\mathbf{p}_i \leftarrow \operatorname{proj}_{\mathcal{C}_i}(\mathbf{D}_i\mathbf{x}), \tag{27}$$

and a global step,

$$\mathbf{x} \leftarrow \left(\mathbf{M}+\Delta t^2\sum_{i=1}^m k_i\mathbf{D}_i^T\mathbf{D}_i\right)^{-1}\left(\mathbf{M}\tilde{\mathbf{x}}+\Delta t^2\sum_{i=1}^m k_i\mathbf{D}_i^T\mathbf{p}_i\right) \tag{28}$$

$$= \left(\mathbf{M}+\Delta t^2\mathbf{D}^T\mathbf{K}\mathbf{D}\right)^{-1}\left(\mathbf{M}\tilde{\mathbf{x}}+\Delta t^2\mathbf{D}^T\mathbf{K}\mathbf{p}\right). \tag{29}$$

where $\operatorname{proj}_{\mathcal{C}_i}$ gives the nearest point on $\mathcal{C}_i$, and $\mathbf{K}$ is the matrix with diagonal blocks $k_i\mathbf{I}$ of the appropriate size. The user is required to implement the projection step $\operatorname{proj}_{\mathcal{C}_i}$ for the constraint manifolds of interest.

In Appendix A, we show that this algorithm is nearly identical to a special case of the ADMM algorithm applied to (26), with $w_i = \sqrt{k_i}$. 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 A reduces our $\mathbf{z}$-step to

$$\mathbf{p}_i^{n+1} = \operatorname{proj}_{\mathcal{C}_i}(\mathbf{D}_i\mathbf{x}^{n+1}+\bar{\mathbf{u}}_i^n), \tag{30}$$

$$\mathbf{z}_i^{n+1} = \frac{k_i\mathbf{p}_i^{n+1}+w_i^2(\mathbf{D}_i\mathbf{x}^{n+1}+\bar{\mathbf{u}}_i^n)}{k_i+w_i^2}, \tag{31}$$

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

As further validation, in Figure 2 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 [DDB*15]. While our proof 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

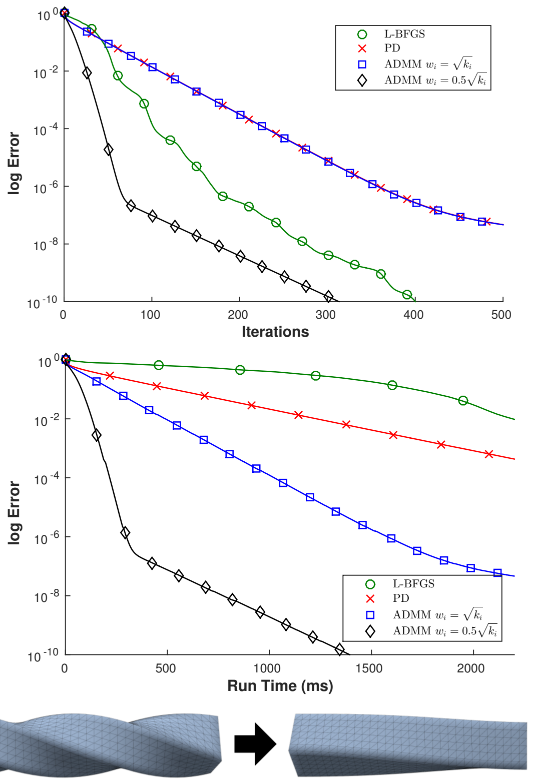

Figure 2: *Convergence of projective dynamics, ADMM, and L-BFGS in the presence of rotations. A deformed box partially untwists over a time step of $\Delta t = 0.04\,$s. On the y-axis we plot relative error $(\varepsilon(\mathbf{x}^n)-\varepsilon(\mathbf{x}^*))/(\varepsilon(\mathbf{x}^0)-\varepsilon(\mathbf{x}^*))$, where $\varepsilon(\mathbf{x}) = f(\mathbf{x})+g(\mathbf{D}\mathbf{x})$, and $\varepsilon(\mathbf{x}^*)$ is the objective value after 1000 L-BFGS iterations. For L-BFGS, we used a history window of size 10.*

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. 5. The L-BFGS method applied directly to the original problem (5) is slower to converge than both projective dynamics and ADMM in terms of computation time, because each iteration is more expensive.

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,

$$\mathbf{F} = \begin{bmatrix}\mathbf{x}_b-\mathbf{x}_a & \mathbf{x}_c-\mathbf{x}_a & \mathbf{x}_d-\mathbf{x}_a\end{bmatrix}\mathbf{B}^{-1}, \tag{32}$$

which is a 3×3 matrix. We define $\mathbf{D}_i$ to be the matrix mapping $\mathbf{x}$ to $\operatorname{vec}(\mathbf{F})$, 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

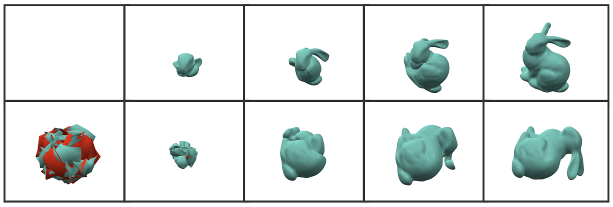

Figure 3: *A low-stiffness Saint Venant–Kirchhoff bunny recovers from having its vertices (top) collapsed to a point, or (bottom) randomized.*

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

$$U_i(\mathbf{z}_i) = V_i \Psi(\text{reshape}_{3\times 3}(\mathbf{z}_i)). \tag{33}$$

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

$$\text{prox}_\Psi(\tilde{\mathbf{F}}, k) = \arg\min_{\mathbf{F}} \left(\Psi(\mathbf{F}) + \frac{k}{2}\|\mathbf{F} - \tilde{\mathbf{F}}\|_F^2 \right), \tag{34}$$

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

$$\tilde{\mathbf{F}}_i^{n+1} = \text{reshape}_{3\times 3}(\mathbf{D}_i\mathbf{x}^{n+1} + \bar{\mathbf{u}}_i^n) \tag{35}$$

$$\mathbf{F}_i^{n+1} = \text{prox}_\Psi(\tilde{\mathbf{F}}_i^{n+1}, w_i^2/V_i) \tag{36}$$

$$\mathbf{z}_i^{n+1} = \text{vec}(\mathbf{F}_i^{n+1}). \tag{37}$$

The formulation for triangle strains is analogous.

For isotropic materials, whose strain energy depends only on the singular values of $\mathbf{F}$, it is sufficient to apply the proximal operator to the singular values alone while preserving the singular vectors. This reduces the problem to a 3-dimensional optimization problem. To handle inverted tetrahedra, we use the IFE convention for the singular value decomposition [ITF04, SHST12], and constrain prox_Ψ to only return nonnegative singular values. Figure 3 demonstrates the robustness of our approach.

Hard constraints: In Section 4.1 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 $\mathcal{S}$,

$$U_i(\mathbf{z}_i) = \begin{cases} 0 & \text{if } \mathbf{z}_i \in \mathcal{S}, \\ \infty & \text{otherwise.} \end{cases} \tag{38}$$

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

$$\mathbf{z}_i^{n+1} = \text{proj}_{\mathcal{S}}(\mathbf{D}_i\mathbf{x}^{n+1} + \bar{\mathbf{u}}_i^n). \tag{39}$$

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.

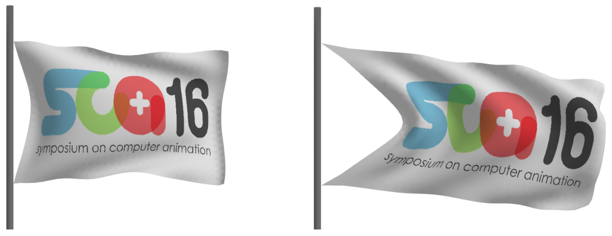
SC+16
symposium on computer animation
SC+16
symposium on computer animation

Figure 4: *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.*

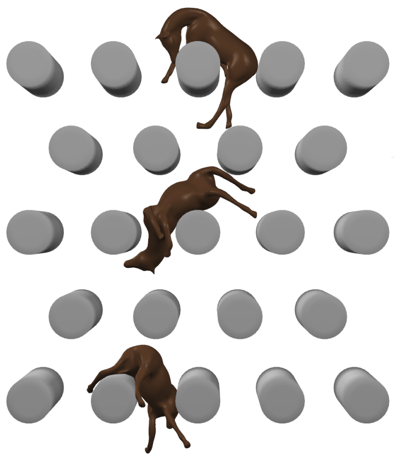

Figure 5: *An elastic horse is dropped through a series of cylinders, demonstrating collision handling.*

Rigid rods (and inextensible ropes) can easily be modeled with this approach, where the projection becomes normalization of the vector to exactly (or at most) the desired length. In Figure 4, we show a more complex example: a cloth flag with strain limits applied to each triangle. For cloth it is appropriate to limit strain primarily in the horizontal and vertical directions, which we do by projecting the columns of the deformation gradient $\mathbf{F}$ so their lengths lie in $[0.95, 1.05]$. Isotropic strain limiting can be applied by projecting the singular values of $\mathbf{F}$ instead.

Collisions: Collisions with static obstacles can be handled in exactly the same way as hard constraints. 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. An example of an elastic body experiencing collisions with static obstacles is shown in Figure 5.

4.3. Performance

We tested the performance of our unoptimized implementation on a 3.5GHz Intel Xeon E5-1650 with 6 cores and hyperthreading. The

Example	#nodes	#elems	#iters	Local step (ms)	Global step (ms)	Time per frame (ms)
Armadillo (Fig. 1)	919	2761	7	7.9	0.45	64
Bunny (Fig. 3)	777	2510	10	4.3	0.41	47
Flag (Fig. 4)	1251	2400	20	0.85	0.58	30
Horse (Fig. 5)	962	3221	13	0.96	0.54	19
Untwist (Fig. 2) — low res, linear	367	1562	20	0.57	0.26	16
— medium res, linear	868	4170	20	1.5	0.81	47
— high res, linear	1727	9031	20	3.4	1.9	107
— high res, neo-Hookean	1727	9031	20	29	2.0	633

Table 1: *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. The times listed for local and global steps are per iteration.*

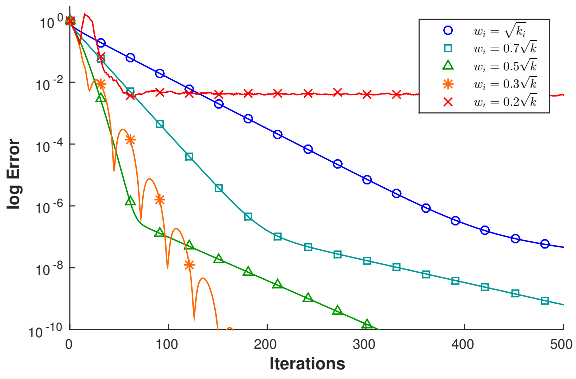

Figure 6: *Convergence of the untwisting box example from Figure 2, using ADMM with different values of w_i. Reducing w_i can improve the convergence rate, but if it is too low, ADMM may fail to converge.*

local step was parallelized with OpenMP, and the global step was solved using a precomputed sparse Cholesky factorization of the system matrix. Run-time performance numbers of all our examples are presented in Table 1. Figures 1, 3, and 5 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 (34). 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.

5. Discussion and future work

One 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 (26), 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 **z**-step (24) have similar curvature. Figure 6 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.

It is known that for convex, potentially nonsmooth functions, ADMM has an $O(1/n)$ convergence rate [WB13]. Recently work in optimization has proposed variations on ADMM that have faster convergence rates [GMS12, GOSB14, KCSB15]. 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 [Wan15] to our algorithm.

An important application of nonlinear constitutive models is the use of data-driven materials acquired from real objects. Unfortunately, existing techniques [BBO*09, WOR11, MBT*12] 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 the acquired data would allow such materials to be used in optimization-based integration techniques such as ours.

Non-conservative forces such as friction and damping are not yet supported in our implicit integration method. Rayleigh damping may be incorporated using the lagged approach proposed by Gast et al. [GSS*15], which expresses each damping force as an additional term in the objective function. Friction and other non-conservative forces are more challenging, and we plan to address them in future work. One potential approach may be to interpret the local step (24) as the task of finding the equilibrium of the force $-\nabla U_i$ plus a spring connecting $\mathbf{z}_i$ to $\mathbf{D}_i\mathbf{x} + \bar{\mathbf{u}}_i$. This task can equally well be performed for non-conservative forces, although we may lose the convergence and robustness guarantees of optimization-based integration.

6. 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 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. This makes our method an effective technique for highly parallel simulation of realistic deformable objects.

Acknowledgements

We would like to thank Arindam Banerjee and Ioannis Karamouzas for insightful discussions, and Jie Li for help with the rendering.

Appendix A: Proof that projective dynamics ≈ ADMM

We will apply ADMM to the projective dynamics energy (26)

$$U_i(\mathbf{z}_i) = \min_{\mathbf{p}_i \in \mathcal{C}_i} \frac{k_i}{2} \|\mathbf{z}_i - \mathbf{p}_i\|^2. \tag{40}$$

In our formulation of ADMM, we have one parameter $\mathbf{W}$. We define $\mathbf{W}$ via $\mathbf{W}_i = k_i \mathbf{I} = \sqrt{w_i}\mathbf{I}$, so that $\mathbf{W}^T\mathbf{W} = \mathbf{K}$. Then the energy can be conveniently expressed in terms of a single constraint manifold, $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_m$:

$$U_*(\mathbf{z}) = \min_{\mathbf{p} \in \mathcal{C}} \frac{1}{2}(\mathbf{z}-\mathbf{p})^T\mathbf{K}(\mathbf{z}-\mathbf{p}) \tag{41}$$

$$= \min_{\mathbf{p} \in \mathcal{C}} \frac{1}{2}\|\mathbf{W}(\mathbf{z}-\mathbf{p})\|^2. \tag{42}$$

Now the $\mathbf{z}$-update of ADMM, i.e. (16) with $\rho = 1$, becomes

$$\mathbf{z}^{n+1} = \arg\min_{\mathbf{z}} \left(U_*(\mathbf{z}) + \frac{1}{2}\|\mathbf{W}(\mathbf{D}\mathbf{x}^{n+1} - \mathbf{z} + \bar{\mathbf{u}}^n)\|^2 \right) \tag{43}$$

$$= \arg\min_{\mathbf{z}} \left(\min_{\mathbf{p} \in \mathcal{C}} \frac{1}{2}\|\mathbf{W}(\mathbf{z}-\mathbf{p})\|^2 + \frac{1}{2}\|\mathbf{W}(\mathbf{z}-\mathbf{y})\|^2 \right) \tag{44}$$

where $\mathbf{y} = \mathbf{D}\mathbf{x}^{n+1} + \bar{\mathbf{u}}^n$. Consider the underlying minimization

$$\min_{\mathbf{z},\mathbf{p} \in \mathcal{C}} \frac{1}{2}\|\mathbf{W}(\mathbf{z}-\mathbf{p})\|^2 + \frac{1}{2}\|\mathbf{W}(\mathbf{z}-\mathbf{y})\|^2. \tag{45}$$

For any fixed $\mathbf{p} \in \mathcal{C}$, the minimum is attained at $\mathbf{z} = \frac{1}{2}(\mathbf{p}+\mathbf{y})$ and its value is $\frac{1}{4}\|\mathbf{W}(\mathbf{p}-\mathbf{y})\|^2$. Therefore, the optimal $\mathbf{p}$ must minimize $\|\mathbf{W}(\mathbf{p}-\mathbf{y})\|^2$. For our choice of $\mathbf{W}$ and $\mathcal{C}$ this amounts to minimizing $w_i\|\mathbf{p}_i - \mathbf{y}_i\|^2$ independently for each i, that is, choosing $\mathbf{p}_i = \mathrm{proj}_{\mathcal{C}_i}\mathbf{y}_i = \mathrm{proj}_{\mathcal{C}_i}(\mathbf{D}_i\mathbf{x}^{n+1} + \bar{\mathbf{u}}^n)$. So in fact we have

$$\mathbf{p}^{n+1} = \mathrm{proj}_{\mathcal{C}}(\mathbf{D}\mathbf{x}^{n+1} + \bar{\mathbf{u}}^n), \tag{46}$$

$$\mathbf{z}^{n+1} = \frac{1}{2}\left(\mathbf{p}^{n+1} + \mathbf{D}\mathbf{x}^{n+1} + \bar{\mathbf{u}}^n\right). \tag{47}$$

Armed with (46)–(47), we will now eliminate $\mathbf{z}$ from the ADMM update rules in favour of $\mathbf{p}$. The $\mathbf{u}$-update becomes

$$\bar{\mathbf{u}}^{n+1} = \bar{\mathbf{u}}^n + \mathbf{D}\mathbf{x}^{n+1} - \mathbf{z}^{n+1} \tag{48}$$

$$= \bar{\mathbf{u}}^n + \mathbf{D}\mathbf{x}^{n+1} - \frac{1}{2}\left(\mathbf{p}^{n+1} + \mathbf{D}\mathbf{x}^{n+1} + \bar{\mathbf{u}}^n\right) \tag{49}$$

$$= \frac{1}{2}\left(\mathbf{D}\mathbf{x}^{n+1} + \bar{\mathbf{u}}^n - \mathbf{p}^{n+1}\right). \tag{50}$$

Conveniently, this also means that after the $\bar{\mathbf{u}}$-update,

$$\mathbf{z}^{n+1} - \bar{\mathbf{u}}^{n+1} = \frac{1}{2}\left(\mathbf{p}^{n+1} + \mathbf{D}\mathbf{x}^{n+1} + \mathbf{u}^n\right) - \frac{1}{2}\left(\mathbf{D}\mathbf{x}^{n+1} + \bar{\mathbf{u}}^n - \mathbf{p}^{n+1}\right) \tag{51}$$

$$= \mathbf{p}^{n+1}. \tag{52}$$

The $\mathbf{x}$-update is now

$$\mathbf{x}^{n+1} = \left(\mathbf{M} + \Delta t^2\mathbf{D}^T\mathbf{W}^T\mathbf{W}\mathbf{D}\right)^{-1}\left(\mathbf{M}\tilde{\mathbf{x}} + \Delta t^2\mathbf{D}^T\mathbf{W}^T\mathbf{W}(\mathbf{z}^n - \bar{\mathbf{u}}^n)\right) \tag{53}$$

$$= \left(\mathbf{M} + \Delta t^2\mathbf{D}^T\mathbf{W}^T\mathbf{W}\mathbf{D}\right)^{-1}\left(\mathbf{M}\tilde{\mathbf{x}} + \Delta t^2\mathbf{D}^T\mathbf{W}^T\mathbf{W}\mathbf{p}^n\right), \tag{54}$$

exactly like the $\mathbf{x}$-update in projective dynamics. Instead of the $\mathbf{z}$-update we have

$$\mathbf{p}^{n+1} = \mathrm{proj}_{\mathcal{C}}(\mathbf{D}\mathbf{x}^{n+1} + \bar{\mathbf{u}}^n) \tag{55}$$

$$\iff \mathbf{p}_i^{n+1} = \mathrm{proj}_{\mathcal{C}_i}(\mathbf{D}_i\mathbf{x}^{n+1} + \bar{\mathbf{u}}_i^n), \tag{56}$$

which is almost exactly like the $\mathbf{p}$-update in projective dynamics, except for the presence of the dual variables $\bar{\mathbf{u}}_i$. Finally, the $\mathbf{u}$-update remains

$$\bar{\mathbf{u}}^{n+1} = \frac{1}{2}\left(\mathbf{D}\mathbf{x}^{n+1} + \bar{\mathbf{u}}^n - \mathbf{p}^{n+1}\right) \tag{57}$$

$$\iff \bar{\mathbf{u}}_i^{n+1} = \frac{1}{2}\left(\mathbf{D}_i\mathbf{x}^{n+1} + \bar{\mathbf{u}}_i^n - \mathbf{p}_i^{n+1}\right) \tag{58}$$

which has no counterpart in projective dynamics.

So far we have seen that for a general constraint manifolds $\mathcal{C}_i$, projective dynamics and ADMM are extremely similar, with the only difference being the presence of the $\bar{\mathbf{u}}_i$ variables and their corresponding update rules. In the special case when the constraints are *linear*, that is, the manifolds $\mathcal{C}_i$ are affine, we will show that the two algorithms become identical.

Let $\mathcal{C}_i$ be an affine subspace with normal space $\mathcal{N}_i$. Then the projection operator $\mathrm{proj}_{\mathcal{C}_i}$ has the properties that

$$\mathbf{z}_i - \mathrm{proj}_{\mathcal{C}_i}\mathbf{z}_i \in \mathcal{N}_i, \tag{59}$$

$$\forall \mathbf{n} \in \mathcal{N}_i : \mathrm{proj}_{\mathcal{C}_i}(\mathbf{z}_i + \mathbf{n}) = \mathrm{proj}_{\mathcal{C}_i}\mathbf{z}_i. \tag{60}$$

We can see that

$$\bar{\mathbf{u}}_i^{n+1} = \frac{1}{2}\left(\mathbf{D}_i\mathbf{x}^{n+1} + \bar{\mathbf{u}}_i^n - \mathbf{p}_i^{n+1}\right) \tag{61}$$

$$= \frac{1}{2}\left((\mathbf{D}_i\mathbf{x}^{n+1} + \bar{\mathbf{u}}_i^n) - \mathrm{proj}_{\mathcal{C}}(\mathbf{D}_i\mathbf{x}^{n+1} + \bar{\mathbf{u}}_i^n)\right) \tag{62}$$

$$\in \mathcal{N}_i, \tag{63}$$

and so

$$\mathbf{p}_i^{n+1} = \mathrm{proj}_{\mathcal{C}_i}\left(\mathbf{D}_i\mathbf{x}^{n+1} + \mathbf{u}_i^n\right) \tag{64}$$

$$= \mathrm{proj}_{\mathcal{C}_i}\mathbf{D}_i\mathbf{x}^{n+1} \tag{65}$$

as long as $\mathbf{u}_i^0 \in \mathcal{N}_i$ (for example, if we initialize $\mathbf{u}_i^0 = \mathbf{0}$).

This proves the equivalence of projective dynamics and ADMM for linear constraints. Furthermore, nonlinear constraints that are smooth can be well approximated by a linearization in the neighbourhood of the solution, so both algorithms should behave similarly as they approach convergence.

References

[BBO*09] BICKEL B., BÄCHER M., OTADUY M. A., MATUSIK W., PFISTER H., GROSS M.: Capture and modeling of non-linear heterogeneous soft tissue. *ACM Trans. Graph. 28*, 3 (July 2009), 89:1–89:9. 1, 6

[BDS*12] BOUAZIZ S., DEUSS M., SCHWARTZBURG Y., WEISE T., PAULY M.: Shape-up: Shaping discrete geometry with projections. *Comp. Graph. Forum 31*, 5 (Aug. 2012), 1657–1667. 2

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. 33*, 4 (July 2014), 154:1–154:11. 1, 2

[BMO*14] BENDER J., MÜLLER M., OTADUY M. A., TESCHNER M., MACKLIN M.: A survey on position-based simulation methods in computer graphics. *Comp. Graph. Forum 33*, 6 (2014), 228–251. 1, 2

[BPC*11] BOYD S., PARIKH N., CHU E., PELEATO B., ECKSTEIN J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn. 3*, 1 (Jan. 2011), 1–122. 1, 2, 3

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proc. Conf. on Comp. Graph. and Interactive Tech.* (1998), SIGGRAPH, ACM, pp. 43–54. 2

[DDB*15] DEUSS M., DELEURAN A. H., BOUAZIZ S., DENG B., PIKER D., PAULY M.: Shapeop—a robust and extensible geometric modelling paradigm. In *Modelling Behaviour: Design Modelling Symposium 2015*, Thomsen R. M., Tamke M., Gengnagel C., Faircloth B., Scheurer F., (Eds.). Springer International Publishing, 2015, pp. 505–515. 4

[EZC10] ESSER E., ZHANG X., CHAN T. F.: A general framework for a class of first order primal-dual algorithms for convex optimization in imaging science. *SIAM J. Img. Sci. 3*, 4 (Dec. 2010), 1015–1046. 2

[GITH14] GREGSON J., IHRKE I., THUEREY N., HEIDRICH W.: From capture to simulation: Connecting forward and inverse problems in fluids. *ACM Trans. Graph. 33*, 4 (July 2014), 139:1–139:11. 2

[GMS12] GOLDFARB D., MA S., SCHEINBERG K.: Fast alternating linearization methods for minimizing the sum of two convex functions. *Math. Programming 141*, 1 (2012), 349–382. 6

[GOSB14] GOLDSTEIN T., O'DONOGHUE B., SETZER S., BARANIUK R.: Fast alternating direction optimization methods. *SIAM J. Img. Sci. 7*, 3 (2014), 1588–1623. 6

[GSS*15] GAST T. F., SCHROEDER C., STOMAKHIN A., JIANG C., TERAN J. M.: Optimization integrator for large time steps. *IEEE Trans. on Vis. and Comp. Graph. 21*, 10 (Oct 2015), 1103–1115. 2, 6

[HLR15] HONG M., LUO Z.-Q., RAZAVIYAYN M.: 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. 3

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *Proc. ACM SIGGRAPH/Eurographics SCA* (2004), SCA, Eurographics Association, pp. 131–140. 5

[KCSB15] KADKHODAIE M., CHRISTAKOPOULOU K., SANJABI M., BANERJEE A.: Accelerated alternating direction method of multipliers. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* (2015), KDD, ACM, pp. 497–506. 6

[KYT*06] KHAREVYCH L., YANG W., TONG Y., KANSO E., MARSDEN J. E., SCHRÖDER P., DESBRUN M.: Geometric, variational integrators for computer animation. In *Proc. ACM SIGGRAPH/Eurographics SCA* (2006), SCA, Eurographics Association, pp. 43–51. 2

[LBK16] LIU T., BOUAZIZ S., KAVAN L.: Towards real-time simulation of hyperelastic materials. *arXiv preprint arXiv:1604.07378* (2016). 2

[LBOK13] LIU T., BARGTEIL A. W., O'BRIEN J. F., KAVAN L.: Fast simulation of mass-spring systems. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 214:1–214:7. 2

[MBT*12] MIGUEL E., BRADLEY D., THOMASZEWSKI B., BICKEL B., MATUSIK W., OTADUY M. A., MARSCHNER S.: Data-driven estimation of cloth simulation models. *Comp. Graph. Forum 31*, 2.2 (2012), 519–528. 1, 6

[MHHR07] MÜLLER M., HEIDELBERGER B., HENNIX M., RATCLIFF J.: Position based dynamics. *J. Vis. Comm. and Img. Rep. 18*, 2 (2007), 109 – 118. 1, 2

[MTGG11] MARTIN S., THOMASZEWSKI B., GRINSPUN E., GROSS M.: Example-based elastic materials. *ACM Trans. Graph. 30*, 4 (July 2011), 72:1–72:8. 2

[NAB*15] NARAIN R., ALBERT R. A., BULBUL A., WARD G. J., BANKS M. S., O'BRIEN J. F.: Optimal presentation of imagery with focus cues on multi-plane displays. *ACM Trans. Graph. 34*, 4 (July 2015), 59:1–59:12. 2

[NSO12] NARAIN R., SAMII A., O'BRIEN J. F.: Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 152:1–152:10. 2

[NW06] NOCEDAL J., WRIGHT S.: *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. 2, 3

[OV14] O'CONNOR D., VANDENBERGHE L.: Primal-dual decomposition by operator splitting and applications to image deblurring. *SIAM J. Img. Sci. 7*, 3 (2014), 1724–1754. 2

[PB14] PARIKH N., BOYD S.: Proximal algorithms. *Found. Trends Optim. 1*, 3 (Jan. 2014), 127–239. 3

[SHST12] STOMAKHIN A., HOWES R., SCHROEDER C., TERAN J. M.: Energetically consistent invertible elasticity. In *Proc. ACM SIGGRAPH/Eurographics SCA* (2012), SCA, Eurographics Association, pp. 25–32. 5

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Proc. Conf. on Comp. Graph. and Interactive Tech.* (1987), SIGGRAPH, ACM, pp. 205–214. 2

[Wan15] WANG H.: A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 246:1–246:9. 2, 6

[WB13] WANG H., BANERJEE A.: Online alternating direction method. In *Proc. Intl. Conf. Mach. Learn.* (2013). 6

[WOR11] WANG H., O'BRIEN J. F., RAMAMOORTHI R.: Data-driven elastic models for cloth: Modeling and measurement. *ACM Trans. Graph. 30*, 4 (July 2011), 71:1–71:12. 1, 6

[XSZB15] XU H., SIN F., ZHU Y., BARBIČ J.: Nonlinear material design using principal stretches. *ACM Trans. Graph. 34*, 4 (July 2015), 75:1–75:11. 1

[ZC08] ZHU M., CHAN T.: *An efficient primal-dual hybrid gradient algorithm for total variation image restoration*. Tech. rep., University of California, Los Angeles, 2008. 2