A General Two-Stage Initialization for Sag-Free Deformable Simulations

JERRY HSU and NGHIA TRUONG, University of Utah, USA
CEM YUKSEL, University of Utah & Cyber Radiance, USA
KUI WU, Lightspeed & Quantum Studios, Tencent America, USA

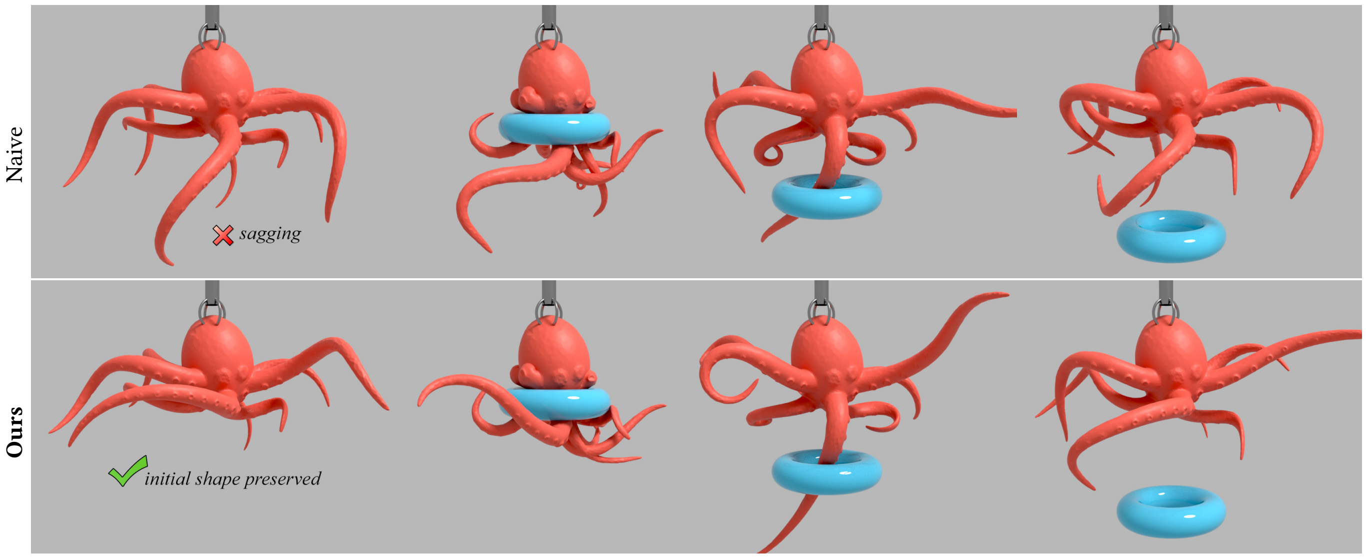

Fig. 1. *An example deformable object simulation prepared using (top-row) naive initialization that treats the given initial shape as the rest shape, which leads to sagging with gravity, and (bottom-row) our initialization that preserves the given initial shape by treating it as the intended shape in static equilibrium under gravity. The two initialization methods produce qualitatively similar animations, while ours maintains the initial shape prior to collisions with the torus. Simulations are generated using FEM with corotated linear elasticity material [Sifakis and Barbic 2012].*

Initializing simulations of deformable objects involves setting the rest state of all internal forces at the rest shape of the object. However, often times the rest shape is not explicitly provided. In its absence, it is common to initialize by treating the given initial shape as the rest shape. This leads to sagging, the undesirable deformation under gravity as soon as the simulation begins. Prior solutions to sagging are limited to specific simulation systems and material models, most of them cannot handle frictional contact, and they require solving expensive global nonlinear optimization problems.

We introduce a novel solution to the sagging problem that can be applied to a variety of simulation systems and materials. The key feature of our approach is that we avoid solving a global nonlinear optimization problem by performing the initialization in two stages. First, we use a global linear optimization for static equilibrium. Any nonlinearity of the material definition is handled in the local stage, which solves many small local problems efficiently and in parallel. Notably, our method can properly handle frictional contact orders of magnitude faster than prior work. We show that our approach can be applied to various simulation systems by presenting examples with mass-spring systems, cloth simulations, the finite element method, the material point method, and position-based dynamics.

CCS Concepts: • **Computing methodologies** → **Physical simulation**;

Additional Key Words and Phrases: deformable simulation, mass-spring system, FEM, MPM, PBD, inverse problem, inverse simulation

ACM Reference Format:
Jerry Hsu, Nghia Truong, Cem Yuksel, and Kui Wu. 2022. A General Two-Stage Initialization for Sag-Free Deformable Simulations. *ACM Trans. Graph.* 41, 4, Article 64 (July 2022), 13 pages. https://doi.org/10.1145/3528223.3530165

Authors' addresses: Jerry Hsu, jerry060599@gmail.com; Nghia Truong, University of Utah, Salt lake city, UT, USA; Cem Yuksel, cem@cemyuksel.com, University of Utah & Cyber Radiance, Salt lake city, UT, USA; Kui Wu, kwwu@tencent.com, Lightspeed & Quantum Studios, Tencent America, Los Angeles, CA, USA.

1 INTRODUCTION

Deformable objects are a primary target for physically-based simulations in computer graphics. These simulations can be initialized using an explicitly provided *rest shape*, where the object is stationary without any internal or external forces. Yet, oftentimes such a rest shape is not provided. This is because artists typically model assets by implicitly considering gravity and contact. Therefore, such models must contain internal forces to preserve their shapes. Nonetheless, in the absence of a user-provided rest shape, it is a common practice to treat the given *initial shape* (i.e. the shape of the model at the beginning of the simulation) as its rest shape. Unfortunately, this leads to the well-known problem of *sagging*, the undesired deformation of the object as soon as the simulation begins to apply external forces, such as gravity (see

Figure 1). Thus, the simulated deformable object deviates from its intended initial shape, collapsing under its own weight.

Existing methods for sag-free initialization have to construct and solve large nonlinear optimization problems. This is because typical material models used in deformable simulations (e.g. hyperelastic Neo-Hookean) are highly nonlinear. Furthermore, coupling with frictional contacts subject to cone constraints in the optimization makes the problem even more difficult and expensive to solve.

That is why, with few exceptions [Derouet-Jourdan et al. 2013; Ly et al. 2018], external contacts are either completely ignored or treated as fixed position constraints that are prone to producing incorrect initialization. In addition, they are all specific to a particular simulation method and/or material model that can significantly limit their potential applications.

In this paper, we present a novel approach for initializing deformable simulations that prevent sagging. Unlike some prior methods that compute a rest shape, our approach simply solves for the internal configurations at the given initial shape. A key insight of our approach is the realization that the construction of an expensive nonlinear system can be entirely avoided by performing initialization in two stages: a *global stage* that ensures static equilibrium and a *local stage* that converts the computed static equilibrium conditions into simulation parameters.

The global stage can be handled with a linear system, which allows using significantly faster solvers, by separating the material definition from the static equilibrium conditions. Our global stage remains linear, even in the presence of highly-nonlinear materials and frictional contacts. Any nonlinearity is handled in the local stage that solves much smaller local problems. Also, this local stage can be trivially parallelized. Therefore, it can be computed quickly.

Unlike all prior methods that are limited to a single simulation system [Chen et al. 2014; Ly et al. 2018; Mukherjee et al. 2018; Wang 2015], our approach can be implemented with a variety of simulation techniques. As a result, we present the first sag-free initialization approaches for the material point method (MPM) and position-based dynamics (PBD) to our knowledge. We also show that our method can be applied to cloth simulation to achieve orders of magnitude faster initialization than prior work with contact and friction. Furthermore, our method can initialize mass-spring systems and the finite element method (FEM) in the presence of frictional contact and is orders of magnitude faster than prior work for similar simulations [Ly et al. 2018; Twigg and Kačić-Alesić 2011].

Because we do not explicitly compute a rest shape, our approach is less suitable for applications that require a rest shape, such as 3D printing. On the other hand, a rest shape does not always exist in reality either. For example, there exists no shape with no internal forces for the internal structures of plants, concrete bonded with tensioned rods, or tempered glass (with inner layers in tension and outer layers in compression). In fact, a rest shape does not always exist for deformable simulations either.

In summary, we present a sag-free initialization method for deformable simulations that

- Supports various, significantly different simulation methods,
- Provides an efficient solution with a linear global system,
- Handles nonlinear materials by solving small local problems,
- And properly incorporates frictional contact.

2 RELATED WORK

Though we target simulations of deformable objects in this paper, the problem of simulation initialization is not specific to deformable objects. For example, it is possible to use backward-in-time integration to determine the initial state of a rigid body simulation that would result in a desired target state [Twigg and James 2008].

There is a large body of work on simulating deformable objects in graphics. Earlier methods relied on mass-spring systems [Chen et al. 1998; Terzopoulos 1995; Terzopoulos et al. 1987]. Then, finite element method [Chen et al. 1998; Debunne et al. 2001; Müller et al. 2002; Sifakis and Barbic 2012] and, more recently, material point method [Fang et al. 2019; Gao et al. 2018; Jiang et al. 2017, 2015; Stomakhin et al. 2013] became more popular, as they are adopted from continuum mechanics and can handle various materials and physical phenomena. For high-performance simulations, position-based dynamics (PBD) provides a popular alternative [Bender et al. 2014; Macklin and Muller 2021; Macklin et al. 2016].

Initializing simulations of deformable objects has been an important problem, since *naive initialization* that treats the initial shape as the rest shape leads to sagging when simulation begins.

This problem received particular interest in hair simulation. Earlier work circumvented the sagging problem by modeling hair in the presence of gravity [Lee and Ko 2001]. The multi-body chain representation [Hadap 2006] allowed initializing each strand using inverse dynamics. The super-helix model for hair simulation [Bertails et al. 2006] was adapted for simulating 2D curves [Derouet-Jourdan et al. 2010], which can be initialized to maintain static equilibrium under gravity. All of these approaches initialize curves in isolation without considering contact. The only complete solution for initializing hair simulations is limited to the super-helix representation and requires a global nonlinear quadratic optimization in a conic domain for formulating inverse dynamics with frictional contact [Derouet-Jourdan et al. 2013]. Hair simulation methods used in production today still rely on ad hoc solutions, such as adding spring forces and constraints to minimize sagging [Iben et al. 2019].

Methods for sag-free initialization have been explored for simulating other deformable objects as well. Example-based deformable simulations can be initialized by leveraging explicitly-provided rest shapes [Martin et al. 2011; Schumacher et al. 2012]. In the absence of a given rest shape, Twigg and Kačić-Alesić [2011] proposed a force-based formulation of static equilibrium for deformable objects simulated using mass-spring systems. This method bears similarities to our approach, but differs starkly in its complex nonlinear global optimization, lack of frictional contact handling, and, for some problems, the requirement for changing certain constraints' stiffness in order to generate a valid solution.

Static equilibrium problems have been of interest for engineering applications as well. In particular, Whiting et al. [2009], Deuss et al. [2014], Shin et al. [2016], and Yao et al. [2017] proposed linear systems for static equilibrium, similar to our global stage.

More recently, Ly et al. [2018] presented a specialized sag-free initialization solution for simulating elastic shells, which is capable of handling frictional contact using a two-step algorithm. The first step solves a nonlinear static equilibrium problem by constraining the positions of all vertices that are in contact. Then, the second step projects the results onto a convex domain of valid frictional contact, using a nonlinear least squares minimization. In comparison, our approach can initialize similar simulation scenarios with orders of magnitude faster computation times.

A closely-related problem to sag-free initialization of deformable simulations is inverse elastic shape design, which is motivated by the intention of fabricating soft objects with a desired shape under gravity. In this case, the goal is computing a rest shape for the object from a given desired shape, such that the object fabricated using the rest shape deforms into the desired shape under gravity. This problem has been investigated for fabricating balloons [Skouras et al. 2012], soft characters with actuators [Skouras et al. 2013], 3D printed soft objects [Chen et al. 2014; Mukherjee et al. 2018; Wang et al. 2015], flexible rod meshes [Pérez et al. 2015], garments [Bartle et al. 2016], and planar-rod structures [Miguel et al. 2016]. All of these solutions are limited to a particular simulation system and involve a global nonlinear optimization problem. More importantly, since these methods are not targeted for initializing simulations, they cannot handle frictional contact, which is essential for various simulation scenarios.

3 DEFORMABLE SIMULATION INITIALIZATION

Our sag-free initialization method can be applied to various deformable simulation systems. To keep our descriptions here in general, we use an abstraction of the simulation system, assuming that the deformable object is discretized into a number of *masses* and *elements*. The elements apply internal forces on the masses, based on the deformation of the object.

How these elements and masses are defined can vary depending on the simulation method. Graphics simulations typically use point masses at particles or vertices of a mesh. The elements in FEM with a tetrahedral mesh would be the tetrahedra, and, in a mass-spring system, it would be the springs. In Section 4, we provide a more detailed discussion of how to interpret them in other simulation methods, including PBD, which does not directly use forces.

An element's forces depend on the element's *state*, a local measure of the object's current shape. An element applies no forces at its *rest state* and its difference from the current state determines its forces.

Our goal is to find a rest state for each element in the presence of external forces (like gravity and contact), such that the deformable object minimizes its deformation when the simulation begins. Our solution is to divide the problem into a *global stage* and a *local stage*. The global stage solves a static equilibrium problem for the entire deformable object to determine the internal forces that should be generated by all elements to maintain the initial shape under external forces. Then, the subsequent local stage initializes the elements by computing their rest states from their current states and their target forces computed in the global stage.

A key insight of our method is the realization that this two-stage process simplifies both stages. The global stage can be constructed as a linear system, even when the elements have nonlinear force responses. Any nonlinearity is contained in the local stage, which requires solving much smaller (i.e. local) problems that can be handled efficiently and parallelized trivially.

3.1 Static Equilibrium

Static equilibrium for a deformable object means that all masses have zero acceleration. Thus, the forces acting on each mass must add up to zero. Let $\mathbf{f}_{ij}$ be the force generated by element i acting on the mass j. We can write the static equilibrium condition as

$$\forall j, \quad \sum_i \mathbf{f}_{ij} = -\mathbf{f}_j^{\text{ext}} , \tag{1}$$

where $\mathbf{f}_j^{\text{ext}}$ is the external force acting on mass j. Any set of internal forces that satisfies Equation 1 would lead to static equilibrium.

Equation 1 is sufficient for defining a static equilibrium for point masses. However, if masses are not points and the forces acting on them are not necessarily applied to their centers, we must ensure that their angular accelerations are zero as well. Thus, the net torque generated by all elements acting on a mass must add up to zero, such that

$$\forall j, \quad \sum_i \left(\mathbf{r}_{ij} \times \mathbf{f}_{ij}\right) = -\boldsymbol{\tau}_j^{\text{ext}} , \tag{2}$$

where $\mathbf{r}_{ij}$ is the vector from the center of the mass to the point that the force $\mathbf{f}_{ij}$ is applied and $\boldsymbol{\tau}_j^{\text{ext}}$ is the torque applied by external forces. Note that this condition is not needed for point masses, since $\mathbf{r}_{ij} = \mathbf{0}$ and $\boldsymbol{\tau}_j^{\text{ext}} = \mathbf{0}$ for all elements i and point masses j.

3.2 Constraints on Internal Forces

We must also ensure that the set of internal forces is within the subspace of forces that the elements can produce. This subspace can vary depending on the specific force formulation used, but there are constraints that broadly apply to all physical force models. For example, elements must produce no net force (based on Newton's third law), satisfying

$$\forall\, i, \quad \sum_j \mathbf{f}_{ij} = \mathbf{0} \, . \tag{3}$$

Note that if the element is connected to a stationary external body (such as a cantilever beam with one side fixed on the wall), while the net force it generates would still add up to zero, Equation 3 would not apply to that specific element.

Also, elements would produce no net torque (to conserve angular momentum). Thus, we can write

$$\forall\, i, \quad \sum_j \left((\mathbf{x}_j - \mathbf{p}) \times \mathbf{f}_{ij}\right) = \mathbf{0} \, , \tag{4}$$

where $\mathbf{x}_j$ is the position of the mass j and $\mathbf{p}$ is an arbitrary point in the space.

Besides these general constraints, the formulation of an element may constrain the subspace of forces it can produce. For example, a spring applies forces only along the spring direction. Such constraints, if any, must be considered when determining the set of forces that must be produced by each element to maintain static equilibrium. We present example force formulations that require different sets of constraints when we discuss applications of our initialization process to specific simulation systems in Section 4.

3.3 The Global Stage

Our goal in the global stage is to find the set of forces $\mathbf{f}_{ij}$ that can be produced by the elements and would maintain static equilibrium. Let m be the number of internal forces $\mathbf{f}_{ij}$ generated by all elements and $\mathbf{f}$ be a vector of length $3m$, containing all external forces $\mathbf{f}_{ij}$ acting on n masses. Using the static equilibrium condition for forces (Equation 1) and the internal force constraints (Equations 3 and 4) discussed above, along with the constraints imposed by the force formulations (if any), we can form a linear system of equations

$$\mathbf{A}\,\mathbf{f} = -\mathbf{f}^{\text{ext}}\ , \tag{5}$$

where $\mathbf{A}$ is a $(3n + n_f) \times 3m$ sparse matrix and n_f is the number of internal force constraints (i.e. Equations 3 and 4). Note that the constraints on the internal forces can be implemented as hard constraints, considerably reducing the size of the linear system in the global stage (as we discuss in Section 4).

In the case of non-point masses, we must also consider the net torque condition (Equation 2). This adds another set of equations to be satisfied per mass, resulting in

$$\begin{bmatrix}\mathbf{A}\\ \mathbf{A}_{\boldsymbol{\tau}}\end{bmatrix}\ \mathbf{f} = -\begin{bmatrix}\mathbf{f}^{\text{ext}}\\ \boldsymbol{\tau}^{\text{ext}}\end{bmatrix}\ , \tag{6}$$

where $\mathbf{A}_{\boldsymbol{\tau}}$ is a $3n \times 3m$ sparse matrix, representing the torque conditions (Equation 2). For simplicity, we assume point masses in the rest of the paper, omitting the torque conditions in our equations.

Typically, we have many more elements than masses (i.e. $n > m$). As such, $\mathbf{A}$ is often an under-determined system, which may lead to infinitely many solutions. Out of all possible solutions, we would often prefer the solution that would minimize the internal stress of the object in the initial shape. In fact, minimizing the internal stress might even be more favorable over strictly preventing all sagging. Thus, we solve

$$\min_{\mathbf{f}} \left\|\mathbf{A}\,\mathbf{f} + \mathbf{f}^{\text{ext}}\right\|_2^2 + \rho\,\|\mathbf{f}\|_2^2\ , \tag{7}$$

where ρ is a small regularization coefficient that steers the solution towards minimal internal stress ($\rho = 10^{-7}$ in our examples). Note that using $\rho = \infty$ would correspond to naive initialization with $\mathbf{f} = \mathbf{0}$. The resulting least squares problem can be written as

$$\left(\mathbf{A}^T\mathbf{A} + \rho\mathbf{I}\right)\ \mathbf{f} = -\mathbf{A}^T\mathbf{f}^{\text{ext}} \tag{8}$$

Since $\mathbf{A}^T\mathbf{A} + \rho\mathbf{I}$ is positive definite, we can solve this minimization using conjugate gradient or any sparse linear system solver. Note that if Equation 5 does not have a solution, we get $\mathbf{f}$ that would minimize the motion.

3.4 Contacts

For a deformable object to be in static equilibrium under gravity, it must either be attached to a stationary object or rest in contact on a surface. To properly handle contacts, we define normal forces acting on masses in contact with stationary objects, along the surface normal $\mathbf{n}_j$ at the contact point. Thus, the normal force can be written as

$$\mathbf{f}_j^{\mathbf{n}} = c_j\ \mathbf{n}_j\ , \tag{9}$$

where c_j is a scalar indicating the magnitude of the normal force.

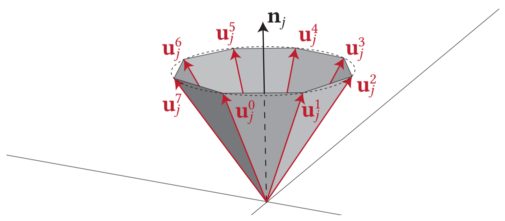

Fig. 2. *Conservative polygonal approximation of the friction, using 8 frictional contact directions $\mathbf{u}_j^k$ around the collision normal $\mathbf{n}_j$.*

We add the normal force $\mathbf{f}_j^{\mathbf{n}}$ to $\mathbf{f}$ in Equation 5 to treat as just another unknown force, which is only solved in the global stage to maintain the contact. In fact, since $\mathbf{n}_j$ is known, we simply need to solve for the magnitude of the normal force c_j. However, $\mathbf{f}_j^{\mathbf{n}}$ is subject to a special constraint that c_j must not be negative.

Fortunately, this boundary constraint (i.e. $c_j \geq 0$) can be enforced without requiring nonlinear optimization, such as using the boundary-constrained conjugate gradient (BCCG) method [Vollebregt 2014]. Though any other constrained optimization solver would work as well, keeping the problem linear provides major improvements in performance, which we demonstrate in Section 5.7.

Notice that our contact handling mechanism does not consider how the simulation system handles contacts. Indeed, it is not required that the simulation system uses a similar or even a force-based contact handling method. Any method for contact handling that would prevent excessive penetration between the deformable object and the collision objects would be effectively equivalent to applying corresponding forces to balance the deformable object.

3.5 Friction

Friction can be modeled in our formulation as forces acting on contact points in orthogonal directions to the surface normal. Thus, the friction force $\mathbf{f}_j^{\mu}$ acting on mass j must satisfy the orthogonality condition $\mathbf{f}_j^{\mu} \cdot \mathbf{n}_j = 0$, which would be easy to add to our system as a constraint. Yet, the magnitude of the friction force is also bounded by the magnitude of the normal force, such that

$$\left\|\mathbf{f}_j^{\mu}\right\| \leq \mu\ c_j\ , \tag{10}$$

where μ is the static friction coefficient. This is a nonlinear constraint that cannot be directly inserted into our linear system.

Geometrically, the static friction constraint in Equation 10 forms a cone that contains all valid frictional contact forces. To avoid this nonlinear constraint, we use a conservative polygonal approximation of the friction cone, similar to Kaufman et al. [2008], as shown in Figure 2. Using a K-sided prism, formed by unit vectors $\mathbf{u}_j^k$ with $k \in \{0 \ldots K-1\}$ around the collision normal $\mathbf{n}_j$ and on the surface of the collision cone, we can represent the frictional-contact force within the collision cone as a weighted combination

$$\mathbf{f}_j^{\mu} = \sum_{k=0}^{K-1} c_j^k\ \mathbf{u}_j^k\ , \tag{11}$$

where $c_j^k \geq 0$ are the magnitudes of the frictional-contact forces. Note that using sufficiently large K, we can represent almost all possible frictional-contact forces within the friction cone (we use $K = 8$ for all examples in this paper). More importantly, this forms a conservative approximation and the resulting collision contact force $\mathbf{f}_j^\mu$ is guaranteed to remain within the collision cone as long as $c_j^k \geq 0$. Therefore, we can expect the simulation system to produce the computed friction force (or use an equivalent static friction handling mechanism) to maintain static equilibrium.

Again, the simulation system does not need to explicitly generate the exact friction forces we compute in our global stage. Any form of enforcing static friction that maintains the positions should be sufficient for preserving the static equilibrium we compute. Because the role of the static friction force is to simply prevent sliding, the rest of the internal forces we compute can preserve the object's shape as long as the simulation system can prevent sliding (either by applying a similar friction force or using an analogous constraint). Indeed, the simulation examples we present in Section 5 use different contact and friction handling mechanisms than the ones in our global stage.

3.6 The Local Stage

The goal of the local stage is determining the rest state of each element. Let $\mathbf{q}_i$ represent the rest state parameters of element i, and $\mathbf{G}_i$ be its force function that depends on $\mathbf{q}_i$ and the deformable object's shape $\mathbf{x}$. We can write

$$\mathbf{H}_i = \mathbf{G}_i(\mathbf{x}, \mathbf{q}_i) \,, \tag{12}$$

where $\mathbf{H}_i = \left[\mathbf{f}_{ij} \ldots\right]$ is the matrix of all forces generated by element i. If the force function $\mathbf{G}_i$ is invertible (with respect to $\mathbf{q}_i$) and has a closed form solution, such that

$$\mathbf{q}_i = \mathbf{G}_i^{-1}(\mathbf{H}_i)\big|_{\mathbf{x}} \,, \tag{13}$$

the computation of rest configuration $\mathbf{q}_i$ becomes trivial.

However, this is not the case for all possible force formulations that can be used with our method. For example, most FEM material models use nonlinear functions $\mathbf{G}_i$ with respect to the rest shape that is not easy to invert. In such cases, we can numerically solve for $\mathbf{q}_i$ using Newton or quasi-Newton iterations. Though this requires solving a nonlinear system, since each element is handled independently, we end up with many small systems of nonlinear equations that can be solved efficiently and in parallel.

Note that within this local stage, we can also solve for the stiffness of the force model to enable local stiffening of weak points. This allows dynamically tuning the stiffness of the object without recomputing the global stage.

4 EXAMPLE SIMULATION SYSTEMS

Our method can be used with various simulation systems that have different integration methods, material models, and force formulations. We only require that the set of equations in the global stage includes the necessary constraints for defining the subspace of forces $\mathbf{f}_{ij}$ that can be produced by each element i. As we show in this section, this subspace, even for various nonlinear force models, can be defined using linear constraints. Thus, our global stage remains linear and any nonlinearity of the force formulation is reserved for the local stage. In the rest of this section, we discuss the details of using our method with example simulation systems.

4.1 Forces with Known Directions

Let $\mathbf{x}$ represent the current shape of the object, containing all positions $\mathbf{x}_j$ of its masses. Various force formulations used in computer graphics have directions defined entirely by the current shape $\mathbf{x}$.

A good example of this is the spring force. Any spring i, linear or nonlinear, connecting two masses $\mathbf{x}_j$ and $\mathbf{x}_k$ applies equal and opposite forces along the spring direction, such that

$$\mathbf{f}_{ij} = -\mathbf{f}_{ik} = -\kappa_i \, \mathbf{d}_{jk} T_i \,, \tag{14}$$

where κ_i is the stiffness, $\mathbf{d}_{ij} = (\mathbf{x}_j - \mathbf{x}_k)/\|\mathbf{x}_j - \mathbf{x}_k\|$ denotes the spring direction, and T_i is the spring tension. Notice that here the only unknown during initialization is T_i. It is trivial to enforce the force direction as a hard constraint by solving for T_i values, instead of the individual forces $\mathbf{f}_{ij}$. Also, with this constraint Equations 3 and 4 are automatically satisfied.

Forces with known directions are not limited to springs that connect two points. More generally, many such force formulations can be written as derivatives of a scalar condition function [Baraff and Witkin 1998], such that the resulting force is

$$\mathbf{f}_{ij} = -\kappa_i \frac{\partial C_i(\mathbf{x})}{\partial \mathbf{x}_j} C_i(\mathbf{x}) \,, \tag{15}$$

where $C_i(\mathbf{x})$ is the scalar condition function. Here, the only unknown during initialization is the value of $C_i(\mathbf{x})$ at the initial shape. Thus, any force formulation that is derived from a scalar condition can be easily used with our method, including all types of springs [Choi and Ko 2002; Selle et al. 2008; Wu and Yuksel 2016] and cloth forces [Baraff and Witkin 1998].

Note that, unlike prior methods, our global stage is entirely independent of the material model. This makes it trivial to keep the static equilibrium equations linear, when the force directions are known, even when using highly-nonlinear materials. If the material has limits on the force magnitude, however, they must be incorporated as boundary constraints, similar to our contact forces.

4.2 Finite Element Method (FEM)

In FEM, we have tetrahedra as elements applying forces to their vertices acting as masses. These force directions are not purely defined by the vertex positions $\mathbf{x}_j$. In fact, a tetrahedron can apply any set of forces, as long as it obeys the physical constraints of no net force and no net torque, satisfying Equations 3 and 4, respectively. Therefore, with FEM we simply need to incorporate these equations into the linear system of our global stage.

We can satisfy these equations using hard constraints. Consider a single tetrahedron applying forces $\mathbf{f}_j$ onto its vertices j, where $j \in \{0, 1, 2, 3\}$. Equation 3 can be enforced by simply replacing $\mathbf{f}_3$ in the linear system with

$$\mathbf{f}_3 = -\mathbf{f}_0 - \mathbf{f}_1 - \mathbf{f}_2 \,. \tag{16}$$

To enforce Equation 4, we compute the torque at one of the vertices, say $\mathbf{p} = \mathbf{x}_3$, and the net torque condition simplifies to

$$\boldsymbol{\tau} = \sum_{j=0}^{2} \mathbf{e}_j \times \mathbf{f}_j = \mathbf{0} \,, \tag{17}$$

where $\mathbf{e}_j = \mathbf{x}_j - \mathbf{x}_3$. Then, $\mathbf{f}_2$ can be cancelled out in this equation using $\mathbf{e}_2 \cdot \boldsymbol{\tau} = 0$, resulting

$$(\mathbf{e}_2 \times \mathbf{e}_0) \cdot \mathbf{f}_0 + (\mathbf{e}_2 \times \mathbf{e}_1) \cdot \mathbf{f}_1 = 0 \,. \tag{18}$$

Based on this equation, any one component of $\mathbf{f}_0$ or $\mathbf{f}_1$ can be represented using the other five components. Considering $\mathbf{e}_2 \times \boldsymbol{\tau} = \mathbf{0}$, $\mathbf{f}_2$ can be represented in the form of $\mathbf{f}_0$, $\mathbf{f}_1$, and a scalar value $\mathbf{e}_2 \cdot \mathbf{f}_2$

$$\mathbf{f}_2 = \frac{\mathbf{e}_2}{\mathbf{e}_2 \cdot \mathbf{e}_2} (\mathbf{e}_2 \cdot \mathbf{f}_2) + \frac{\mathbf{e}_2}{\mathbf{e}_2 \cdot \mathbf{e}_2} \times (\mathbf{e}_0 \times \mathbf{f}_0 + \mathbf{e}_1 \times \mathbf{f}_1) \,. \tag{19}$$

Therefore, only $\mathbf{f}_0$, two components of $\mathbf{f}_1$, and $\mathbf{e}_2 \cdot \mathbf{f}_2$ are unknowns in the linear system in Equation 5.

After all forces are determined in the global stage, the local stage solves for the the *reference shape matrix* $\mathbf{D}_m$ of each tetrahedron, which represents its rest configuration [Sifakis and Barbic 2012]. The force matrix $\mathbf{H} = \begin{bmatrix} \mathbf{f}_0 & \mathbf{f}_1 & \mathbf{f}_2 \end{bmatrix}$ of a tetrahedral element is computed from $\mathbf{D}_m$ using

$$\mathbf{H} = -w(\mathbf{D}_m)\, \mathbf{P}(\mathbf{D}_m)\, \mathbf{D}_m^{-T} \,, \tag{20}$$

where $w(\mathbf{D}_m)$ is the volume of the element and $\mathbf{P}(\mathbf{D}_m)$ is the first Piola-Kirchhoff stress tensor that defines the material behavior. Given $\mathbf{H}$ computed in the global stage, we can find $\mathbf{D}_m$ by solving a small nonlinear optimization problem for each tetrahedral element

$$\min_{\mathbf{D}_m} \left\| \mathbf{H} + w(\mathbf{D}_m)\, \mathbf{P}(\mathbf{D}_m)\, \mathbf{D}_m^{-T} \right\|_2^2 \,. \tag{21}$$

Note that the target forces in $\mathbf{H}$ can only be produced by the material, if it is sufficiently stiff. Otherwise, we must bound the forces in the global stage. Alternatively, we can easily solve for the minimal stiffness needed to produce the target forces in the local stage. This provides an efficient and convenient way of introducing *local stiffening* as a part of our initialization.

4.3 Material Point Method (MPM)

The way we incorporate our approach into MPM is somewhat counterintuitive. We treat particles (a.k.a. *material points*) as the elements and the grid vertices as the masses, because we solve the static equilibrium on the grid. Though particles carry mass in MPM, the forces are applied on the grid vertices, and while in static equilibrium, each grid vertex has a corresponding mass, defined by the surrounding particles.

Below we show how to apply our approach to two examples of the moving least squares MPM (MLS-MPM) [Hu et al. 2018] with quadratic interpolation kernel, though any other MPM formulation can be used with our method as well. We first describe the details of how to incorporate hyper-elastic MPM, and then we discuss MPM fluid simulation.

4.3.1 Hyper-elastic MPM. We consider hyper-elastic MPM simulations that do not incorporate plasticity, because plasticity can simply overwrite the internal deformations we compute for maintaining static equilibrium.

Let $\mathbf{f}_{ij}$ represent the force applied by particle i with position $\mathbf{p}_i$ onto the grid node j with position $\mathbf{x}_j$. It is computed using

$$\mathbf{f}_{ij} = -\frac{4}{\Delta x^2} V_i\, \boldsymbol{\sigma}_i\, (\mathbf{x}_j - \mathbf{p}_i)\, \omega_{ij} \tag{22}$$

where $\boldsymbol{\sigma}_i$ is a 3×3 symmetric matrix representing the particle's Cauchy stress tensor, V_i is particle's volume at the current shape $\mathbf{x}$, Δx is the grid cell size, and ω_{ij} is the interpolation weight computed based on the distance from particle i to grid node j. The only unknown in Equation 22 is $\boldsymbol{\sigma}_i$ and the rest can be directly taken from the initial shape and the simulation parameters.

While forming the linear system in our global stage, we must include the necessary constraints to make sure that the $\mathbf{f}_{ij}$ values are within the subspace of forces that can be generated by Equation 22. We can easily achieve this using hard constraints and solving for $\boldsymbol{\sigma}_i$ in the global stage, instead of the individual $\mathbf{f}_{ij}$ values. We enforce that the resulting $\boldsymbol{\sigma}_i$ are symmetric matrices by solving for their upper triangular parts. Note that this solution automatically satisfies Equation 3 (i.e. no net force), because $\sum_j (\mathbf{x}_j - \mathbf{p}_i)\omega_{ij} = \mathbf{0}$, and Equation 4 (i.e. no net torque), because $\boldsymbol{\sigma}_i$ is symmetric.

The Cauchy stress tensor $\boldsymbol{\sigma}_i$ is a function of particle's deformation gradient $\mathbf{F}_i$, such that

$$\boldsymbol{\sigma}_i = \frac{1}{\det(\mathbf{F}_i)} \mathbf{P}(\mathbf{F}_i)\, \mathbf{F}_i^T \,, \tag{23}$$

where $\mathbf{P}(\mathbf{F}_i)$ is the first Piola-Kirchhoff stress tensor, the definition of which varies depending on the material model. $\mathbf{F}_i$ is stored at each particle to record how the material has been deformed. Therefore, in the local stage, after having $\boldsymbol{\sigma}_i$ from the global stage, we compute $\mathbf{F}_i$ of each particle. This is achieved by solving the small nonlinear optimization problems

$$\min_{\mathbf{F}_i} \left\| \boldsymbol{\sigma}_i - \frac{1}{\det(\mathbf{F}_i)} \mathbf{P}(\mathbf{F}_i)\, \mathbf{F}_i^T \right\|_2^2 \,. \tag{24}$$

Once we have $\mathbf{F}_i$, we can directly compute the particle's volume at the rest shape using $V_i^R = V_i / \det(\mathbf{F}_i)$, completing the initialization.

4.3.2 MPM Fluid Simulation. We use the nearly incompressible fluid model [Tampubolon et al. 2017] to simulate water using MPM. In this case, the Cauchy stress tensor is defined using water pressure p, such that

$$\boldsymbol{\sigma}_i = -\mathbf{I} p_i \qquad \text{with} \qquad p_i = k\left(\frac{1}{J_i^{\gamma}} - 1\right), \tag{25}$$

where p_i is designed to penalize the volume change of the water tensor and computed based on the determinant of the water deformation gradient $J_i = \det(\mathbf{F}_i)$, k is the bulk modulus of the water, and γ is a term to penalize deviation from incompressibility.

Similar to hyper-elastic MPM, we can solve $\boldsymbol{\sigma}_i$ in the global stage. The only difference is that $\boldsymbol{\sigma}_i$ is a 3×3 scalar matrix, so our linear system is formed using only the p_i terms. Once the global stage computes the p_i values, we can directly evaluate $\boldsymbol{\sigma}_i$ and solve for the $\mathbf{F}_i$ terms of each particle in the local stage. Then, we can easily evaluate J_i to initialize each water particle.

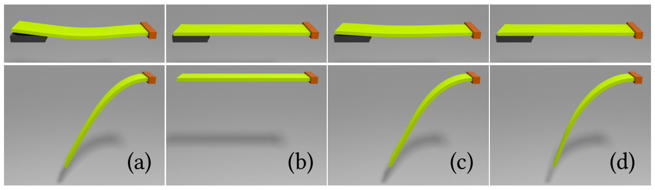

Fig. 3. *A thin elastic beam that is fixed on the right side and resting on contact on a rigid object on the left side, simulated using FEM with corotated linear elasticity material. The bottom row shows the final simulated shape after the rigid object is removed. The simulations are initialized using (a) naive initialization, (b) ours without contact handling, (c) ours by treating contacts as position constraints, and (d) ours with proper contact handling. Notice that proper contact handling is important for both maintaining the initial shape and allowing deformation when the contact is removed.*

4.4 Position-Based Dynamics (PBD)

PBD [Bender et al. 2014] is a significantly different simulation system than the ones described above. Nonetheless, we can still use our method with PBD, though we must redefine certain terms.

First of all, PBD does not directly use forces. Instead, it computes position updates from a set of *position constraints*. We treat these position constraints as our elements, acting on the vertices/particles of the system that correspond to our masses. Matching our notation in Section 3, let $\mathbf{f}_{ij}$ represent the total position update of constraint i on mass j and $\mathbf{f}_j^{\text{ext}}$ be the position update due to external forces. Then, the static equilibrium condition for PBD using position updates can be written exactly as in Equation 1.

In addition, PBD uses iterative solvers, typically with a fixed number of iterations N per frame. The total position update $\mathbf{f}_{ij}$ of a constraint i acting on mass j depends on N and its scalar constraint function $C_i(\mathbf{x})$, along with the inverse mass w_j and the inverse stiffness α_i. The resulting static equilibrium condition for mass j to be used in our global stage can be written as

$$\sum_i \mathbf{g}_{ij}(\mathbf{x})\, C_i(\mathbf{x}) = -\mathbf{f}_j^{\text{ext}} \,, \tag{26}$$

where $\mathbf{g}_{ij}(\mathbf{x})$ can be directly computed from the current positions $\mathbf{x}$ and the parameters, such that

$$\mathbf{g}_{ij}(\mathbf{x}) = -N \frac{w_j\, \nabla_j C_i(\mathbf{x})}{\alpha_i\, \sigma_i(\mathbf{x})} \,, \tag{27}$$

with

$$\sigma_i(\mathbf{x}) = \sum_j w_j \|\nabla_j C_i(\mathbf{x})\|^2 \,. \tag{28}$$

We present the derivation of $\mathbf{g}_{ij}(\mathbf{x})$ in Appendix A. The only unknown here during initialization is $C_i(\mathbf{x})$, which we solve for in our global stage.

XPBD [Macklin et al. 2016], on the other hand, uses slightly different position updates to make sure that the effective stiffness of the constraints is independent of the number of iterations N. As described in Appendix A, with XPBD we get

$$\mathbf{g}_{ij}(\mathbf{x}) = -\left(\frac{1-s^N}{1-s}\right) \frac{w_j\, \nabla_j C_i(\mathbf{x})}{\alpha_i + \sigma_i(\mathbf{x})} \,, \tag{29}$$

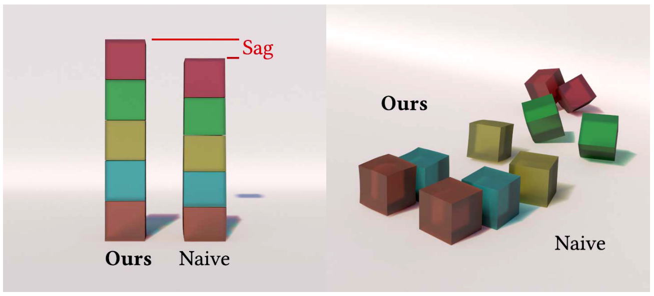

Fig. 4. *Jelly cubes are stacked with self-contacts. The state just after the simulation is shown on the left, and the state after toppling is to the right. Notice the sagging due to naive initialization and the difference in rest shape after toppling.*

where

$$s = \frac{\sigma_i(\mathbf{x})}{\alpha_i + \sigma_i(\mathbf{x})} \,. \tag{30}$$

It is important to note that our derivation in Appendix A assumes that PBD and XPBD use a sufficient number of iterations N to converge to a solution that properly satisfies all constraints. Though we have not observed this in our experiments, we would expect that using too small N that terminates the iterations prior to convergence may lead to some minor sagging due to the remaining error.

5 RESULTS

We evaluate our method by initializing deformable simulations using mass-spring systems, FEM, MPM, cloth, and XPBD. In all our tests, our method successfully produces sag-free simulations under external forces and contacts, and the motion of the deformable objects closely matches simulations using *naive initialization* (i.e. treating the initial shape as the rest shape).

5.1 FEM with Frictional Contacts

In our tests, we use the codebase of IPC [Li et al. 2020] for simulating our FEM examples initialized with our method.

Figure 3 shows the importance of proper contact handling. The thin rectangular elastic beam is fixed on one side and rested on contact with a rigid obstacle on the other side. Suppose the contacts are ignored (Figure 3b), the beam fails to deform when the rigid object is removed. On the other hand, treating contacts as position constraints (similar to the fixed side of the beam), as shown in Figure 3c, leads to incorrect initialization owing to erroneous negative contact normal forces. Proper contact handling with our method is able to avoid all such failure cases.

Our method is able to handle self-contacts during initialization, as shown in Figure 4, in which five jelly cubes are stacked on top of each other. In this case, there can be no sag-free state if contacts are not properly considered. Without contacts that indirectly connect the upper cubes to the ground, it would be impossible to keep them stationary in the air.

A common way of contact handling in prior work is to define (infinitely stiff) fixed position constraints on colliding vertices [Chen et al. 2014; Twigg and Kačić-Alesić 2011]. However, fixed position

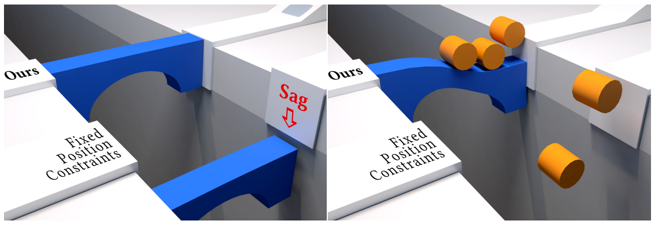

Fig. 5. *A soft bridge using our initialization and fixed position constraints. In the latter case, contacts are treated as static and friction is not handled. Our initialization is able to correctly solve for friction and wedge itself between the two cliffs and (left) maintain the initial state until (right) the barrels overwhelm the bridge's friction. Initializing using fixed position constraints, on the other hand, causes the soft bridge to slide down as soon as the simulation begins.*

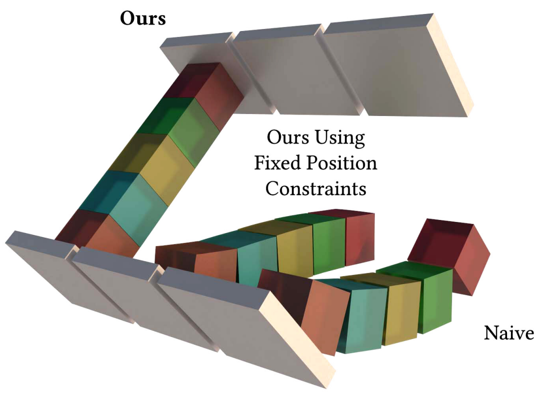

Fig. 6. *Tilted jelly cube stacks sandwiched between opposing rigid obstacles. From left to right is our initialization, position constraint initialization, and naive initialization. Treating contacts as position constraints completely fails to provide a sag-free state as doing so produces physically inaccurate contacts with negative normal force.*

constraints cannot handle frictional contacts properly, as shown in Figure 5. Without proper initialization, the soft bridge between two adjacent cliffs falls under gravity. In this case, our method automatically recognizes the need to expand into the cliffs in order to avoid exceeding the static friction coefficient, leading to a static equilibrium held purely through friction.

Figure 6 demonstrates a complex example with contacts, self-contacts, and frictional contacts by sandwiching tilted jelly cubes between two rigid obstacles. This demonstrates the proper enforcement of non-negative normal forces, static friction constraints, and self-contacts between cubes. If non-negative normal forces were not enforced, the top most cube would most certainly attempt to hang off of the top platform, leading to an invalid solution. If friction constraints were not enforced, the stack would most certainly slip and collapse, leading to an invalid solution. Our method perfectly balances the internal forces, frictional forces, and normal forces to provide a valid rest configuration. For further demonstration, we initialize a truck of soft hay bales in Figure 7 that produces visually similar results during simulation, while preventing sagging on the truck.

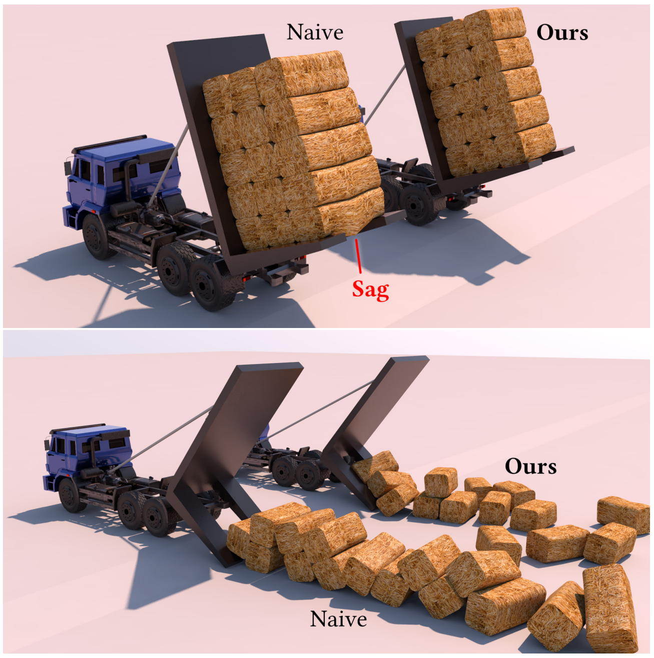

Fig. 7. *The (top) before and (bottom) after of two trucks of soft FEM hey bale. The left truck using naive initialization causes excessive bulging of the hey bales compared to the right truck using our initialization. Again, a difference in rest configuration can be found in the after state on the right.*

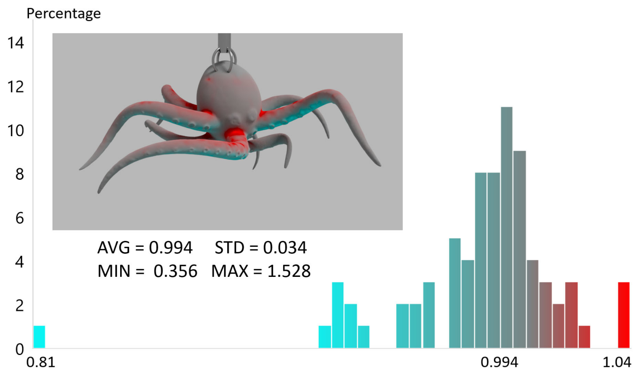

Fig. 8. *Visualization and histogram of volume compression ratios (measured by* $\det(\mathbf{F}_i)$*) of the tetrahedra in the octopus models shown in Figure 1, initialized using our method.*

We show a large FEM example in Figure 1. When initialized using our method, in comparison to naive initialization, Figure 8 shows the computed internal stresses at the initial shape, visualized as compression ratios of the tetrahedra volumes. Notice that most tetrahedra have minor compression/stretching and only a small percentage of them have slightly higher compression/stretching to maintain the initial shape with the necessary internal forces. This indicates that our method can produce stable initialization without excessive internal force.

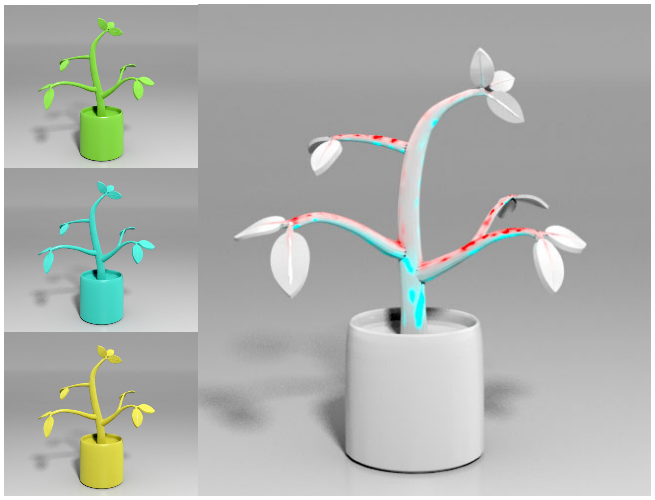

Fig. 9. ***A plant model simulated using FEM with different materials:*** *(top) Neo-Hookean, (middle) Corotated Linear Elasticity, and (bottom) Saint Venant-Kirchhoff, deforming with an external wind force, producing visually similar motion, all initialized using our method. The visualization on the right shows how the tetrahedra in the initial shape are compressed (cyan color) or stretched (red color) after initializing with the Corotated Linear Elasticity material. The average compression ratio is* 0.999 *and the standard deviation is* 0.038. *The other two materials produce similar visualizations.*

Finally, Figure 9 shows FEM simulations with different material models for internal force formulation, including Neo-Hookean, Corotated Linear Elasticity, and Saint Venant-Kirchhoff. Our method is able to handle all three of the common FEM material models we implemented.

5.2 Thin Shell with Contacts

We implement a variety of thin shell examples using spring energies and bending energies defined by the dihedral angle between neighboring triangles. Collisions with the static signed distance function collider, ϕ, are handled by projecting the vertex position, $\mathbf{x}_j$, such that $\phi(\mathbf{x}_j) = 0$ and zeroing the velocity along the projection direction. Vertex-face and edge-edge self-collisions are handled by constraint projection similar to that proposed by Bender et al. [2014]. Note the absence of a force-based collision handling method here. Despite this, our experiments show that our initialization method can still produce non-sagging results with collision. As in Figure 10, our initialization preserves the given shape of the page, while naive initialization leads to the page falling flat. Friction between the page's edge and the bottom page is successfully accounted for, and the pageâĂŹs behavior remains similar under blowing wind. Figure 11 shows an example of a soft blanket wrapped around a sphere with a large number of self-contacts. Initialized by our method, the blanket preserves its initial shape, while naive initialization exhibits a substantial amount of sagging.

We also show an example of local stiffening in cloth, which is especially important here owing to the cloth's soft nature. In

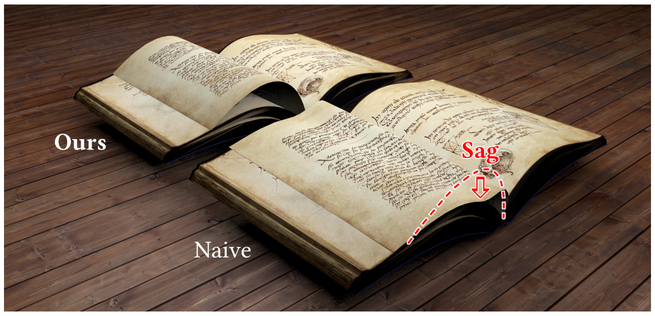

Fig. 10. *Our initialization manages to preserve the initial shape by resting on its edge with friction while naive initialization falls flat. At the start of the simulation, the static friction is immediately overwhelmed and the page collapses.*

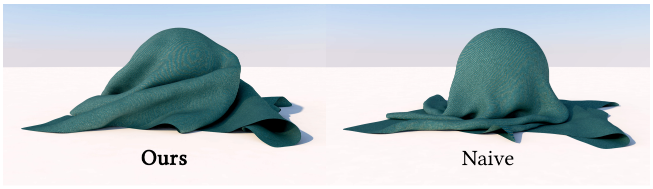

Fig. 11. *A blanket wrapped around a sphere, demonstrating a case with a large number of self-contacts: (left) our initialization successfully preserves the initial shape by utilizing self-contacts for support, but (right) with naive initialization, the folds collapse in on themselves.*

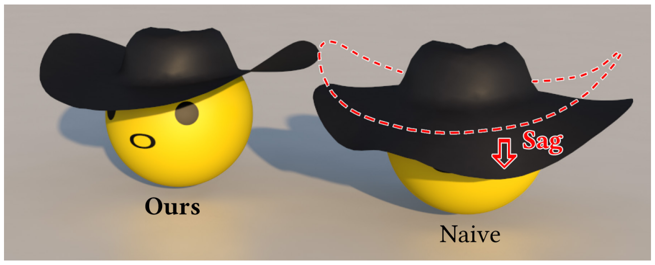

Fig. 12. *Two hats are placed onto spheres and held on through friction. While naive initialization (right) can stay on without slipping off if placed correctly, it fails to maintain the target shape as with our initialization (left). The area around the folds and contacts are automatically stiffened with local stiffening where needed.*

Figure 12, we place a hat on sphere, where friction is used to prevent slippage. With our method, the ridges around the central bulge are automatically stiffened with the edges remaining soft and pliable. This is analogous to the stiffening of cloth when folded or at seams.

5.3 Tension Shifting for Mass-Spring Cloth

Buckling, toppling, and crumbling can be a major concern for poorly supported models with soft materials like cloth. Extra care must be taken to produce solutions that do not lead to runaway collapses under arbitrarily small perturbations. As Equation 7 only considers a solution of minimal internal stress, it may arrive at solutions

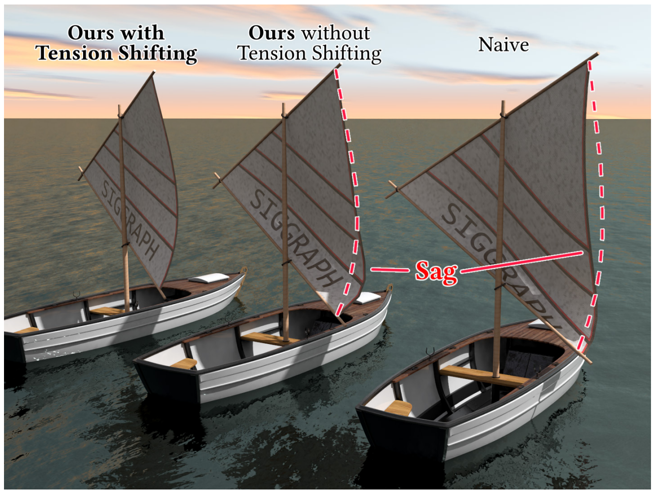

Fig. 13. *We show the difference between our initialization with tension shifting (a) and our initialization without tension shifting (b). While both a and b preserve the initial shape when compared to naive initialization, applying a light breeze quickly causes (b) to collapse. In contrast, our initialization with tension shifting returns to the initial shape even after large perturbations.*

prone to buckling. Mathematically, this instability is characterized by a non-positive definite hessian of the potential energy. However, this is computationally expensive to verify, much less to enforce as constraints. Instead of tackling the problem directly, we embed the shift vector, $\mathbf{s}$, in our optimization problem:

$$\min_{\mathbf{f}} \left\| \mathbf{A}\,\mathbf{f} + \mathbf{f}^{\text{ext}} \right\|_2^2 + \alpha \left\| \mathbf{f} - \mathbf{s} \right\|_2^2 \ . \tag{31}$$

We use the shift vector $\mathbf{s}$ to guide solutions toward states in which the springs are in tension, which we observe to be more stable. The resulting least squares problem in Equation 8 can then be rewritten as

$$\left(\mathbf{A}^T \mathbf{A} + \alpha \mathbf{I} \right) \mathbf{f} = \alpha \mathbf{s} - \mathbf{A}^T \mathbf{f}^{\text{ext}} \tag{32}$$

More specifically, we define the shift vector, $\mathbf{s}$, as following:

$$\mathbf{s}_{ij} = \beta \frac{\partial \mathbf{f}_{ij}}{\partial C_i(\mathbf{x})} = \beta E_i \frac{\mathbf{x}_j - \mathbf{x}_k}{\|\mathbf{x}_j - \mathbf{x}_k\|} \ , \tag{33}$$

where $C_i(\mathbf{x}) = (\|\mathbf{x}_j - \mathbf{x}_k\| - l_i)/l_i$ is the relative strain of the i'th spring (between masses j and k), l_i is the rest length, β is a desired relative strain specified by the user, and E_i is the Young's modulus.

We found this to work well in practice, and Figure 13 demonstrates the importance of tension shifting for cloth. While our method arrives at a state of static equilibrium without tension shifting, it quickly buckles under the wind. Contrastly, the result generated with tension shifting can reliably return to the target shape even after introducing large perturbations.

5.4 Inverse Elastic Shape Design

Our method does not explicitly compute a rest shape for the given deformable object. Therefore, it is not an ideal solution for inverse elastic shape design. Yet, we can still use our method for estimating a rest shape. An example of this is shown in Figure 14. We begin with initializing the deformable object in the presence of external forces (Figure 14a). After initialization, we simulate the deformable object without any external force. This makes the object deform in a direction that reduces the internal forces. After the object reaches a steady state, its shape forms our rest shape estimation (Figure 14b). It is important to note that this is not a true rest shape, since the object can still have internal forces, based on our initialization.

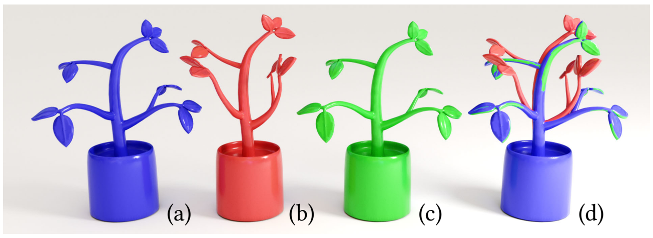

Fig. 14. ***Inverse elastic shape design:*** *(a) the given initial shape, used for initializing our method with gravity, (b) the generated rest shape after simulation without gravity, (c) the final shape after initializing using the generated rest shape and simulating with gravity, and (d) all models. Notice that the initial and the final shapes closely match. Simulated using FEM with corotated linear elasticity material.*

To test the validity of this rest shape estimation, we reinitialize the simulation using it as a rest shape. Then, we simulate the object's deformation with external forces. This is expected to bring the object's shape closer to its intended initial shape (Figure 14c). For the example in Figure 14, this brings the shape remarkably close to the initial shape, verifying that the estimated rest shape is a reasonable approximation (Figure 14d).

5.5 Position-Based Dynamics

Figure 15 shows an example hair mesh simulation using XPBD. Sagging is particularly an important problem for hair simulation. In this example, the hair mesh model is in frictional contact at the given initial shape. Naive initialization deforms the hair model when the simulation begins. Our method, on the other hand, completely prevents sagging, though a subtle motion can be observed in the beginning when XPBD's collision constraints resolve the minor penetrations with the collision objects in the given initial shape.

5.6 Material Point Method (MPM)

Figure 16 shows an example MPM simulation, in which water is simulated using MPM with a nearly incompressible fluid model [Tampubolon et al. 2017], while the bunny is simulated using the Corotated linear elasticity model [Sifakis and Barbic 2012]. Naive initialization leads to bouncing at the beginning while the dam holds the water back. Since there is no interaction between water and bunny during initialization, we initialize them separately. Regarding contacts, we use a slip boundary condition in the forward simulation, which clamps the velocity on the grid perpendicular to the boundary plane. For initialization, we simply remove the degrees of freedom (Dofs) corresponding to components perpendicular to the boundary plane.

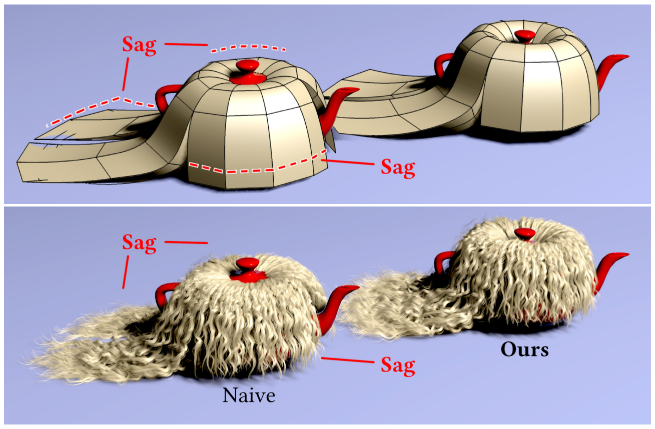

Fig. 15. *An example hair mesh simulation using position-based dynamics (XPBD), exhibiting sagging with naive initialization.*

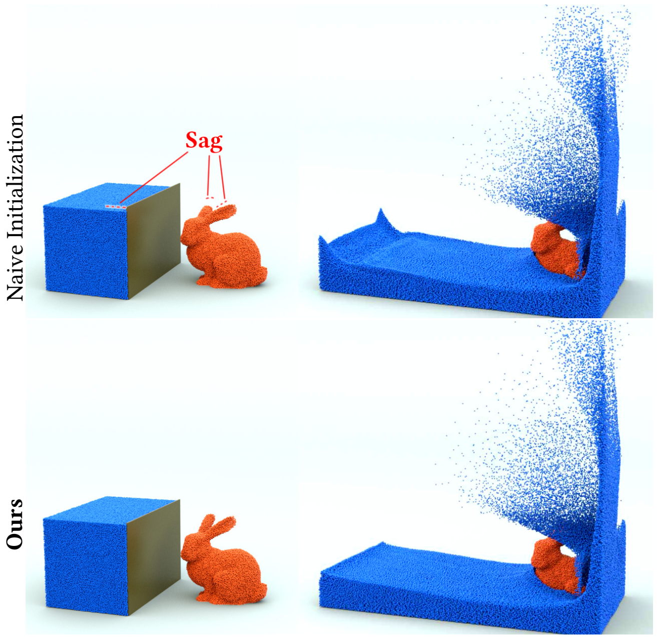

Fig. 16. *An example of MPM simulation with deformable bunny coupling with nearly incompressible MPM water. Before the brown wall on the right of the water is removed, our initialization totally avoids volume loss and sagging on the bunny's ear due to gravity.*

Table 1. *Computation times of our method.*

Example		Simulation	Elements	Masses	Contacts	Global S.	Local S.
Octopus†	Fig. 1	FEM	102,200	23,640	0	221 s	1.5 s
Box Stack*	Fig. 4	FEM	2,802	1,104	245	3.5 s	10 ms
Bridge*	Fig. 5	FEM	266	121	18	0.2 s	1 ms
Tilted Stack*	Fig. 6	FEM	2,879	1,132	294	11 s	12 ms
Heybales*	Fig. 7	FEM	4,390	1,920	294	4.1 s	13 ms
Plant*	Fig. 9	FEM	47,077	14,842	0	34 s	147 ms
Book*	Fig. 10	Cloth	483	180	9	0.08 s	0.03 ms
Blanket*	Fig. 11	Cloth	5338	2750	300	1.13 s	0.14 ms
Hat*	Fig. 12	Cloth	3,156	1,087	27	0.56 s	0.2 ms
Sail*	Fig. 13	Cloth	360	136	0	0.006 s	0.02 ms
Hair Mesh**	Fig. 15	XPDB	3,234	162	89	2.5 s	<0.01 ms
Bunny‡	Fig. 16	MPM	86,819	18,565	-	29 s	36 s
Fluid‡	Fig. 16	MPM	686,937	309,048	-	96 s	15 s
Dragon*	Fig. 17	Mass-Spring	35,874	7,212	0	0.4 s	0.2 ms

*The performance results are measured on computers with different configurations: * an AMD 5950X CPU (16 cores) and 32 GB RAM, †an Intel Core i9-9980XE CPU (18 cores) and 64GB RAM, ‡an AMD Ryzen Threadripper 3970X CPU (32 cores) and 256 GB RAM and computed in Python, and ** dual Intel Xeon E5-2643 v3 CPUs (24 cores) with 64 GB RAM.*

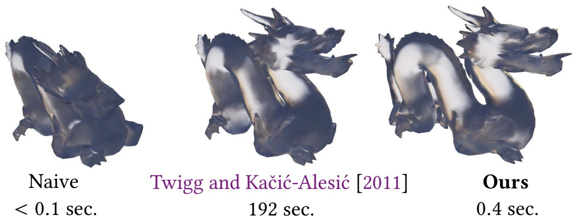

Fig. 17. *An example mass-spring simulation of a tetrahedral mesh.*

5.7 Performance and Comparisons

We summarize the performance results in Table 1. Since the global stage involves linear optimization, it can be computed efficiently. On the other hand, the local stage that contains many small nonlinear optimization problems is computed in a fraction of the time.

The local stage is trivial in mass-spring simulations. For FEM, all three material models in Figure 9 are computed in identical times. Both the simulator and our initialization for MPM are implemented in Python with Taichi programming language [Hu et al. 2018] and SciPy [Virtanen et al. 2020]. Notice that the global stage for MPM simulation is more expensive than the other simulation systems. This is because each element (i.e., particle) in MPM applies forces to 64 masses (i.e., grid vertices), forming a denser matrix in the global linear system. The local stage for the MPM bunny is slower than that for MPM fluid, because each iteration in the local stage for the deformable bunny needs to perform 3×3 matrix SVD.

In Figure 17 we provide a direct comparison to the primal-dual method introduced by [Twigg and Kačić-Alesić 2011] using IPOPT [Wächter and Biegler 2006]. As our method allows for handling rest length positivity constraints in the local stage, our optimization remains linear and is 480× faster than the method of Twigg and Kačić-Alesić [2011] in this example.

Our method also provides significantly faster initialization than Ly et al. [2018], the only prior work that can handle frictional contact with thin shells. For example, our initialization in Figure 12 is completed in half a second, in comparison to 50 minutes reported by Ly et al. [2018] for a model with a similar shape and complexity.

6 DISCUSSION AND LIMITATIONS

In comparison to prior work, our approach is similar to methods that compute internal force parameters [Derouet-Jourdan et al. 2013; Twigg and Kačić-Alesić 2011] as opposed to ones that explicitly solve for the global rest shape of the deformable object [Chen et al. 2014; Ly et al. 2018; Mukherjee et al. 2018]. A clear distinction of our approach to all prior work, however, is that we avoid solving a global nonlinear system.

Another important distinction of our approach from prior work is that our solution is not limited to a specific simulation system or material model. As we have shown in this paper, we can apply our solution to a variety of simulation methods and materials. Most notably, we present the first sag-free initialization method that can be used with MPM and PBD to our knowledge.

The main challenge for applying our solution to a simulation system is determining the subspace of internal forces that can be generated by the force elements. For example, the forces in FEM cover the entire subspace of forces that satisfy the zero net force and zero net torque constraints (Equations 3 and 4), force formulations with known directions cover a portion of this subspace that have force components along with these directions, and MPM forces cover the subspace that can be represented by Equation 22. If the necessary constraints are not enforced, the global step can generate a set of forces for an element that is outside of its subspace. In that case, the local step would fail to initialize the element, since it cannot produce the given set of forces. As long as the subspace of the force formulation can be represented using linear constraints, our global step remains linear. However, if defining this subspace requires nonlinear constraints, the optimization problem in our global step would become nonlinear, since it must include these constraints.

Though our method is general, it is not a universal solution. First of all, our method is only applicable to simulation systems that can be abstracted as elements and masses. Also, we require that the internal force formulation allows computing its rest configuration from its output. Though typical force formulations would permit such solutions, it is theoretically possible to come up with a function that might be challenging.

7 CONCLUSION AND FUTURE WORK

We have introduced a two-stage solution for initializing deformable simulations to achieve sag-free animations with frictional contact. By splitting the problem into a global and a local stage, we can efficiently solve the static equilibrium problem using a linear global optimization and handle any nonlinearity of the force formulations in the local stage. With various examples, we show that our method provides an effective approach for producing sag-free simulations of deformable objects and it can be applied to a variety of simulation systems and material models.

In particular, we have also shown examples with FEM and mass-spring systems, and, notably, we present the first sag-free initialization method for MPM and PBD to our knowledge. More generally, we have explained how any force with a known direction can be initialized using our method.

An obvious future direction would be applying our solution to other simulation systems. This requires investigating their internal force formulations and identifying the set of constraints needed for defining the subspace of forces they can produce.

Existing methods for plasticity in MPM modify the deformation gradients of particles. When used with sag-free initialization, plasticity would simply overwrite the computed deformation gradients. Therefore, another interesting future direction would be incorporating plasticity in MPM with sag-free initialization.

ACKNOWLEDGMENTS

We would like to thank Ming Gao and Yuanming Hu for helpful discussions on our fluid simulation example. This project was supported in part by NSF grant #1956085.

REFERENCES

David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. ACM, New York, NY, USA, 43–54.

Aric Bartle, Alla Sheffer, Vladimir G. Kim, Danny M. Kaufman, Nicholas Vining, and Floraine Berthouzoz. 2016. Physics-Driven Pattern Adjustment for Direct 3D Garment Editing. *ACM Trans. Graph.* 35, 4, Article 50 (July 2016), 11 pages.

Jan Bender, Matthias Müller, Miguel A. Otaduy, Matthias Teschner, and Miles Macklin. 2014. A Survey on Position-Based Simulation Methods in Computer Graphics. *Comput. Graph. Forum* 33, 6 (sep 2014), 228–251.

Florence Bertails, Basile Audoly, Marie-Paule Cani, Bernard Querleux, Frédéric Leroy, and Jean-Luc Lévundefinedque. 2006. Super-Helices for Predicting the Dynamics of Natural Hair. *ACM Trans. Graph.* 25, 3 (July 2006), 1180–1187.

Xiang Chen, Changxi Zheng, Weiwei Xu, and Kun Zhou. 2014. An Asymptotic Numerical Method for Inverse Elastic Shape Design. *ACM Trans. Graph.* 33, 4, Article 95 (July 2014), 11 pages.

Y. Chen, Q. Zhu, A. Kaufman, and S. Muraki. 1998. Physically-Based Animation of Volumetric Objects. In *Proceedings of the Computer Animation (CA '98)*. IEEE Computer Society, USA, 154.

Kwang-Jin Choi and Hyeong-Seok Ko. 2002. Stable but Responsive Cloth. *ACM Trans. Graph.* 21, 3 (July 2002), 604–611.

Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. 2001. Dynamic Real-Time Deformations Using Space Time & Adaptive Sampling. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. ACM, New York, NY, USA, 31–36.

Alexandre Derouet-Jourdan, Florence Bertails-Descoubes, Gilles Daviet, and Joëlle Thollot. 2013. Inverse Dynamic Hair Modeling with Frictional Contact. *ACM Trans. Graph.* 32, 6, Article 159 (Nov. 2013), 10 pages.

Alexandre Derouet-Jourdan, Florence Bertails-Descoubes, and Joëlle Thollot. 2010. Stable Inverse Dynamic Curves. *ACM Trans. Graph.* 29, 6, Article 137 (Dec. 2010), 10 pages.

Mario Deuss, Daniele Panozzo, Emily Whiting, Yang Liu, Philippe Block, Olga Sorkine-Hornung, and Mark Pauly. 2014. Assembling Self-Supporting Structures. *ACM Trans. Graph.* 33, 6, Article 214 (nov 2014), 10 pages.

Yu Fang, Minchen Li, Ming Gao, and Chenfanfu Jiang. 2019. Silly Rubber: An Implicit Material Point Method for Simulating Non-Equilibrated Viscoelastic and Elastoplastic Solids. *ACM Trans. Graph.* 38, 4, Article 118 (July 2019), 13 pages.

Ming Gao, Xinlei Wang, Kui Wu, Andre Pradhana, Eftychios Sifakis, Cem Yuksel, and Chenfanfu Jiang. 2018. GPU Optimization of Material Point Methods. *ACM Trans. Graph.* 37, 6, Article 254 (Dec. 2018), 12 pages.

Sunil Hadap. 2006. Oriented Strands: Dynamics of Stiff Multi-Body System. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Vienna, Austria) *(SCA '06)*. Eurographics Association, Goslar, DEU, 91–100.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018. A Moving Least Squares Material Point Method with Displacement Discontinuity and Two-Way Rigid Body Coupling. *ACM Trans. Graph. (TOG)* 37, 4 (2018), 150.

Hayley Iben, Jacob Brooks, and Christopher Bolwyn. 2019. Holding the Shape in Hair Simulation. In *ACM SIGGRAPH 2019 Talks* (Los Angeles, California) *(SIGGRAPH '19)*. ACM, New York, NY, USA, Article 59, 2 pages.

Chenfanfu Jiang, Theodore Gast, and Joseph Teran. 2017. Anisotropic Elastoplasticity for Cloth, Knit and Hair Frictional Contact. *ACM Trans. Graph.* 36, 4, Article 152 (July 2017), 14 pages.

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The Affine Particle-in-Cell Method. *ACM Trans. Graph.* 34, 4, Article 51 (July 2015), 10 pages.

Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. 2008. Staggered Projections for Frictional Contact in Multibody Systems. *ACM Trans. Graph.* 27, 5, Article 164 (dec 2008), 11 pages.

Doo-Won Lee and Hyeong-Seok Ko. 2001. Natural Hairstyle Modeling and Animation. *Graph. Models* 63, 2 (March 2001), 67–85.

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. 2020. Incremental Potential Contact: Intersection-and Inversion-Free, Large-Deformation Dynamics. *ACM Trans. Graph.* 39, 4, Article 49 (jul 2020), 20 pages.

Mickaël Ly, Romain Casati, Florence Bertails-Descoubes, Mélina Skouras, and Laurence Boissieux. 2018. Inverse Elastic Shell Design with Contact and Friction. *ACM Trans. Graph.* 37, 6, Article 201 (dec 2018), 16 pages.

Miles Macklin and Matthias Muller. 2021. A Constraint-Based Formulation of Stable Neo-Hookean Materials. In *Motion, Interaction and Games* (Virtual Event, Switzerland) *(MIG '21)*. ACM, New York, NY, USA, Article 12, 7 pages.

Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: Position-Based Simulation of Compliant Constrained Dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (Burlingame, California) *(MIG '16)*. ACM, New York, NY, USA, 49–54.

Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-Based Elastic Materials. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. ACM, New York, NY, USA, Article 72, 8 pages.

Eder Miguel, Mathias Lepoutre, and Bernd Bickel. 2016. Computational Design of Stable Planar-rod Structures. *ACM Trans. Graph.* 35, 4, Article 86 (July 2016), 11 pages.

Rajaditya Mukherjee, Longhua Wu, and Huamin Wang. 2018. Interactive Two-Way Shape Design of Elastic Bodies. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 1, Article 11 (July 2018), 17 pages.

Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. 2002. Stable Real-Time Deformations. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Antonio, Texas) *(SCA '02)*. ACM, New York, NY, USA, 49–54.

Jesús Pérez, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, José A. Canabal, Robert Sumner, and Miguel A. Otaduy. 2015. Design and Fabrication of Flexible Rod Meshes. *ACM Trans. Graph.* 34, 4, Article 138 (July 2015), 12 pages.

Christian Schumacher, Bernhard Thomaszewski, Stelian Coros, Sebastian Martin, Robert Sumner, and Markus Gross. 2012. Efficient Simulation of Example-Based Materials. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Lausanne, Switzerland) *(SCA '12)*. Eurographics Association, Goslar, DEU, 1–8.

Andrew Selle, Michael Lentine, and Ronald Fedkiw. 2008. A Mass Spring Model for Hair Simulation. *ACM Trans. Graph.* 27, 3 (aug 2008), 1–11.

Hijung V. Shin, Christopher F. Porst, Etienne Vouga, John Ochsendorf, and Frédo Durand. 2016. Reconciling Elastic and Equilibrium Methods for Static Analysis. *ACM Trans. Graph.* 35, 2, Article 13 (feb 2016), 16 pages.

Eftychios Sifakis and Jernej Barbic. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses* (Los Angeles, California) *(SIGGRAPH '12)*. ACM, New York, NY, USA, Article 20, 50 pages.

Mélina Skouras, Bernhard Thomaszewski, Bernd Bickel, and Markus Gross. 2012. Computational Design of Rubber Balloons. *Comput. Graph. Forum* 31, 2pt4 (May 2012), 835–844.

Mélina Skouras, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, and Markus Gross. 2013. Computational Design of Actuated Deformable Characters. *ACM Trans. Graph.* 32, 4, Article 82 (July 2013), 10 pages.

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A Material Point Method for Snow Simulation. *ACM Trans. Graph.* 32, 4, Article 102 (July 2013), 10 pages.

Andre Pradhana Tampubolon, Theodore Gast, Gergely Klár, Chuyuan Fu, Joseph Teran, Chenfanfu Jiang, and Ken Museth. 2017. Multi-species simulation of porous sand and water mixtures. *ACM Trans. Graph. (TOG)* 36, 4 (2017), 1–11.

Demetri Terzopoulos. 1995. Heating and melting deformable models (from goop to glop). In *Graphics interface*, Vol. 89. Canadian Information Processing Society, Toronto, Ontario, Canada, 219–226.

Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically Deformable Models. *SIGGRAPH Comput. Graph.* 21, 4 (Aug. 1987), 205–214.

Christopher D. Twigg and Doug L. James. 2008. Backward Steps in Rigid Body Simulation. *ACM Trans. Graph.* 27, 3, Article 25 (Aug. 2008), 10 pages.

Christopher D. Twigg and Zoran Kačić-Alesić. 2011. Optimization for Sag-Free Simulations. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Vancouver, British Columbia, Canada) *(SCA '11)*. ACM, New York, NY, USA, 225–236.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272.

Edwin A. H. Vollebregt. 2014. The Bound-Constrained Conjugate Gradient Method for Non-negative Matrices. *Journal of Optimization Theory and Applications* 162, 3 (01 Sep 2014), 931–953.

Andreas Wächter and Lorenz T. Biegler. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming* 106 (2006), 25–57.

Bin Wang, Longhua Wu, KangKang Yin, Uri Ascher, Libin Liu, and Hui Huang. 2015. Deformation Capture and Modeling of Soft Objects. *ACM Trans. Graph.* 34, 4, Article 94 (July 2015), 12 pages.

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

Emily Whiting, John Ochsendorf, and Frédo Durand. 2009. Procedural Modeling of Structurally-Sound Masonry Buildings. *ACM Trans. Graph.* 28, 5 (dec 2009), 1–9.

Kui Wu and Cem Yuksel. 2016. Real-Time Hair Mesh Simulation. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Redmond, Washington) *(I3D '16)*. ACM, New York, NY, USA, 59–64.

Jiaxian Yao, Danny M. Kaufman, Yotam Gingold, and Maneesh Agrawala. 2017. Interactive Design and Stability Analysis of Decorative Joinery for Furniture. *ACM Trans. Graph.* 36, 2, Article 20 (mar 2017), 16 pages.

A DERIVATION OF PBD AND XPBD TOTAL UPDATES

PBD and XPBD use iterative solvers that directly operate on positions and update the positions every time a constraint is applied. Let $\mathbf{x}_j^t$ represent the position of mass j at iteration t, $t \in \{0, \ldots, N\}$. The PBD position update due to constraint i can be written as

$$\mathbf{f}_{ij}^t = -\frac{w_j \, \nabla_j C_i(\mathbf{x}^{t-1})}{\alpha_i \, \sigma_i(\mathbf{x}^{t-1})} C_i(\mathbf{x}^{t-1}) \ . \tag{34}$$

To simplify this and because we are solving for a static equilibrium, we assume that the position of mass j is always at $\mathbf{x}_j$ prior to applying any of its constraints (i.e. $\mathbf{x}^t = \mathbf{x}^{t-1}$). With this simplification, we get the same position update at each iteration (i.e. $\mathbf{f}_{ij}^t = \mathbf{f}_{ij}^{t-1}$) and the resulting total position update becomes

$$\mathbf{f}_{ij} = \sum_{t=1}^{N} \mathbf{f}_{ij}^t = N \, \mathbf{f}_{ij}^1 = \mathbf{g}_{ij}(\mathbf{x}) \, C_i(\mathbf{x}) \ . \tag{35}$$

In XPBD, however, the position update at iteration t depends on the position update of the previous iterations. This is formulated using Lagrange multipliers λ_i^t that vary at each iteration, such that $\lambda_i^t = \lambda_i^{t-1} + \Delta\lambda_i^t$ and $\lambda_i^0 = 0$. Again, using the same simplification (i.e. $\mathbf{x}^t = \mathbf{x}^{t-1}$), we can write

$$\mathbf{f}_{ij}^t = w_j \, \Delta\lambda_i^t \, \nabla_j C_i(\mathbf{x}) \ , \tag{36}$$

where

$$\Delta\lambda_i^t = -\frac{C_i(\mathbf{x}) + \alpha_i \lambda^{t-1}}{\alpha_i + \sigma_i(\mathbf{x})} \ . \tag{37}$$

This can be used for defining a recursive function

$$\lambda_i^t = \lambda_i^1 + s\lambda_i^{t-1} \qquad \text{where} \qquad \lambda_i^1 = -\frac{C_i(\mathbf{x})}{\alpha_i + \sigma_i(\mathbf{x})} \ . \tag{38}$$

The corresponding geometric series for N iterations has a closed-form solution

$$\lambda_i^N = \lambda_i^1 \sum_{t=1}^{N} s^{t-1} = \lambda_i^1 \left(\frac{1 - s^N}{1 - s} \right) \ . \tag{39}$$

The resulting total position update for XPBD can be written as

$$\mathbf{f}_{ij} = \sum_{t=1}^{N} \mathbf{f}_{ij}^t = w_j \, \nabla_j C_i(\mathbf{x}) \sum_{t=1}^{N} \Delta\lambda_i^t = w_j \, \nabla_j C_i(\mathbf{x}) \, \lambda_i^N \ . \tag{40}$$

Combining them with $\mathbf{f}_{ij} = \mathbf{g}_{ij}(\mathbf{x}) \, C_i(\mathbf{x})$, we get Equation 29.