Constrained Projective Dynamics: Real-Time Simulation of Deformable Objects with Energy-Momentum Conservation

MIN HYUNG KEE, Korea University, Korea
KIWON UM, LTCI, Telecom Paris, IP Paris, France
WOOSEOK JEONG, Korea University, Korea
JUNGHYUN HAN, Korea University, Korea

Fig. 1. Our constrained Projective Dynamics method generates vivid motions efficiently.

This paper proposes a novel energy-momentum conserving integration method. Adopting Projective Dynamics, the proposed method extends its unconstrained minimization for time integration into the constrained form with the position-based energy-momentum constraints. This resolves the well-known problem of unwanted dissipation of energy and momenta without compromising the real-time performance and simulation stability. The proposed method also enables users to directly control the energy and momenta so as to easily create the vivid deformable and global motions they want, which is a fascinating feature for many real-time applications such as virtual/augmented reality and games.

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

Additional Key Words and Phrases: Energy-momentum conservation, constrained optimization, projective dynamics, deformable objects simulation, real-time simulation

ACM Reference Format:
Min Hyung Kee, Kiwon Um, WooSeok Jeong, and JungHyun Han. 2021. Constrained Projective Dynamics: Real-Time Simulation of Deformable Objects with Energy-Momentum Conservation. *ACM Trans. Graph.* 40, 4, Article 160 (August 2021), 12 pages. https://doi.org/10.1145/3450626.3459878

Authors' addresses: Min Hyung Kee, Computer Science Department, Korea University, Seoul, Korea; Kiwon Um, IDS Department, LTCI, Telecom Paris, IP Paris, Palaiseau, France; WooSeok Jeong, Computer Science Department, Korea University, Seoul, Korea; JungHyun Han, Computer Science Department, Korea University, Seoul, Korea.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
0730-0301/2021/8-ART160 $15.00
https://doi.org/10.1145/3450626.3459878

1 INTRODUCTION

In computer graphics, simulating elastic motions of deformable objects has been a popular research topic since the seminal work of Terzopoulous et al. [1987]. Real-time applications such as video games and virtual/augmented reality environments involve stringent requirements for both stability and efficiency of simulation. Consequently, the solvers for such applications are often limited to implicit numerical integration methods such as backward Euler, which produces stable results even with large time steps.

The most popular backward Euler-based solvers include Position Based Dynamics [Müller et al. 2007] and Projective Dynamics (henceforth, abbreviated to PD) [Bouaziz et al. 2014]. They reformulate the integration problem as an optimization form and fulfill the requirements for stability and efficiency. Yet, at the same time, they suffer from unwanted dissipation of energy and momenta, i.e., *artificial damping*, which is inherited from the backward Euler scheme. To address this problem, Dinev et al. [2018b] proposed a solution referred to as FEPR. It first takes the full integration steps of the underlying solver and then projects the results onto a constant energy-momentum manifold.

From the previous studies, we made two key observations: (1) the generality of PD, which allows us to extend its numerical integrator to incorporate additional constraints without sacrificing the simulation stability, and (2) the importance of energy-momentum conservation demonstrated in FEPR, which enables vivid motions

addressing unwanted numerical dissipation of energy and momenta. Inspired by the observations, we designed a new constrained optimization formulation.

The main contribution of our work is a new energy-momentum conserving integration method, *constrained Projective Dynamics*. It extends the unconstrained minimization problem of PD's time integration into a constrained one with *position-based energy-momentum constraints*. Our "solver-integrated constraint approach" is distinguished from the "projection-after-solve approach" such as FEPR. The experimental results show that our method produces more visually-appealing elastic motions in a faster way.

An additional strength of our method is that it enables intuitive and easy control of the energy and momenta such that the changes can immediately appear in the simulated motions. This is due to the energy-momentum constraints that are handled within the solver. In contrast, backward Euler solvers make it difficult to control the desirable motions, especially due to the elusive nature of damping that often depends on the number of solver iterations and time step size.

2 RELATED WORK

Implicit integration methods: Due to the well-known instability problems in forward integration methods such as forward Euler, many real-time simulation applications have adopted implicit integration methods since the seminal work of Terzopoulos et al. [1987]. Baraff and Witkin [1998] simulated clothes in real time by employing the first order backward Euler approximation. Choi and Ko [2005] extended it to the second order approximation to increase accuracy.

Implicit integration methods can be reformulated as optimization problems [Gast et al. 2015; Martin et al. 2011; Stuart and Humphries 1998], where the objective functions lead to the constrained motions of simulated objects. For example, Position Based Dynamics (PBD) reformulated the backward Euler scheme as a constrained optimization problem, where the elastic energy is considered as hard constraints and resolved iteratively [Müller et al. 2007]. Macklin et al. [2016] alleviated the problem of iteration-dependent stiffness of PBD by adjusting the Lagrange multiplier for each iteration along with the positions. Liu et al. [2013] presented a method for efficient simulation of mass-spring systems by introducing auxiliary variables that allow two-step optimization approach. It alternately finds the auxiliary variables and the positions. This was generalized to PD [Bouaziz et al. 2014], which is capable of simulating many kinds of deformable objects including rods, clothes, and jellies. PD is not limited to simulating deformable objects. Weiler et al. [2016] used PD for fluid simulation, whereas Li et al. [2019] and Komaritzan and Botsch [2019] extended PD to support character animations. Liu et al. [2017] interpreted PD as a quasi-Newton scheme, whereas Overby et al. [2017] showed that PD can be interpreted as an alternating direction method of multipliers. Both allowed supporting more general materials. Wang et al. [2015] accelerated PD and PBD using a Chebyshev semi-iterative approach, and Brandt et al. [2018] proposed a fast hyper-reduction technique for PD. Kugelstadt et al. [2018] introduced the operator splitting approach by which each term of the time integration scheme can be efficiently solved individually with different numerical methods.

Energy-momentum conservation: Study of energy and momentum conserving methods has been of great interest for physics-based simulation to remedy the stability and numerical dissipation of underlying integrators [Hairer 2006; Simo et al. 1992]. The early studies of Hughes et al. [1978] and LaBudde et al. [1975] applied an energy projection scheme to enforce the energy conservation. However, projecting only the energy is undesirable since the conservation of other quantities such as angular momentum is important as well. Su et al. [2012] enforced energy-momentum conservation by introducing extra forces, depending on the change of energy and momenta.

To achieve better energy-momentum conserving time integration, different integration methods have been studied. For the stiff elastodynamics problems, the exponential integrators have been investigated [Chen et al. 2020; Michels et al. 2017, 2014]. Xu and Barbič [2017] proposed an accurate and stable implicit integration method that combines the trapezoidal rule and second-order implicit backward Euler. Rojas et al. [2019] introduced the Average Vector Field integration, which achieves exact energy conservation for the St. Venant-Kirchhoff material. Löschner et al. [2020] studied the higher-order time integration, which can address the numerical damping of typical first-order integration methods.

Variational integrators, which include the implicit midpoint method and the implicit Newmark type methods [Kane et al. 2000] are known to have good energy-preserving properties [Kharevych et al. 2006; Stern and Desbrun 2006; West 2004]. However, variational integrators become unstable with large time step sizes, making them unsuitable for real-time applications.

Dinev et al. [2018a] proposed a blending technique to satisfy the energy conservation without severely compromising the visual quality. It runs multiple simulators and interpolates the results depending on the energy fluctuation. Brown et al. [2018] proposed a method that accurately simulate dissipative forces when using optimization-based integrators such that it can alleviate the artificial damping. Dahl and Bargteil [2019] introduced an efficient approach that tracks the quantities of global linear/angular momenta of the simulated bodies and adjusts their velocities such that both momenta are preserved in PBD. Dinev et al. [2018b] proposed FEPR, which is a post-processing procedure. It enforces the conservation of energy and momenta by projecting the result of any integration methods onto a constant energy-momentum manifold.

3 PROJECTIVE DYNAMICS

This section reviews the variational form of implicit Euler integration [Martin et al. 2011] and PD [Bouaziz et al. 2014]. Given a mesh composed of m vertices, its state is described as a set of the vertex positions $\mathbf{x}(\in \mathbb{R}^{3m})$ and velocities $\mathbf{v}(\in \mathbb{R}^{3m})$ in three-dimensional space. The state evolves in time according to the physics laws (i.e., Newton's laws of motion), and we can record the evolution at discrete time samples $\{t_1, t_2, \cdots, t_N\}$.

Given a state at t_n, the implicit Euler integration approximates the next state at t_{n+1} as follows:

$$\begin{aligned}\mathbf{x}_{n+1} &= \mathbf{x}_n + h\mathbf{v}_{n+1}\\ \mathbf{v}_{n+1} &= \mathbf{v}_n + h\mathbf{M}^{-1}(\mathbf{f}_{\text{ext}} + \mathbf{f}_{\text{int}}(\mathbf{x}_{n+1}))\end{aligned} \tag{1}$$

where h is the time step size, $\mathbf{M}$ is the mass matrix, $\mathbf{f}_{\text{ext}}$ is the external forces, and $\mathbf{f}_{\text{int}}$ is the internal forces. As $\mathbf{f}_{\text{int}}(\mathbf{x}) = -\sum_j \nabla W_j(\mathbf{x})$, where $W_j(\mathbf{x})(\in \mathbb{R})$ denotes the potential energy of the j-th finite element, Eq. (1) is rewritten as follows:

$$\frac{\mathbf{M}}{h^2}(\mathbf{x}_{n+1} - \mathbf{y}) + \sum_j \nabla W_j(\mathbf{x}_{n+1}) = \mathbf{0} \tag{2}$$

where $\mathbf{y} = \mathbf{x}_n + h\mathbf{v}_n + h^2\mathbf{M}^{-1}\mathbf{f}_{\text{ext}}$.

Solving for the new state $\mathbf{x}_{n+1}$ in Eq. (2) can be converted to an optimization problem finding $\mathbf{x}_{n+1}$ that minimizes the following objective function $\mathcal{G}(\mathbf{x}_{n+1}) : \mathbb{R}^{3m} \rightarrow \mathbb{R}$:

$$\mathcal{G}(\mathbf{x}_{n+1}) = \frac{1}{2h^2}\|\mathbf{x}_{n+1} - \mathbf{y}\|^2_{\mathbf{M}} + \sum_j W_j(\mathbf{x}_{n+1}) \tag{3}$$

For the sake of notational simplicity, we will henceforth use $\mathbf{x}$ instead of $\mathbf{x}_{n+1}$.

One of the novelties in PD lies in the choice for the potential function W_j that uses an auxiliary *projection variable* $\mathbf{z}_j$ and the quadratic distance measure:

$$W_j(\mathbf{x}, \mathbf{z}_j) = \frac{w_j}{2}\|\mathbf{G}_j\mathbf{x} - \mathbf{z}_j\|^2_F \tag{4}$$

where w_j is a positive weight and $\mathbf{G}_j$ is a sparse matrix for a discrete differential operator, which typically employs the deformation gradient operator for an edge (i.e., spring) or a tetrahedron [Sifakis and Barbic 2012]. Replacing W_j of Eq. (3) with Eq. (4) leads to the augmented objective function $g(\mathbf{x})$:

$$g(\mathbf{x}) = \frac{1}{2h^2}\|\mathbf{x} - \mathbf{y}\|^2_{\mathbf{M}} + \sum_j \frac{w_j}{2}\|\mathbf{G}_j\mathbf{x} - \mathbf{z}_j\|^2_F \tag{5}$$

Intuitively, the first term of Eq. (5) acts as the *inertia*, which moves $\mathbf{x}$ toward $\mathbf{y}$, whereas the second term acts as the *elasticity*, which penalizes the deformations of $\mathbf{x}$.

The PD algorithm uses an iterative alternating optimization technique, where $\mathbf{x}^0 = \mathbf{y}$ (initialization) and then $\mathbf{z}_j^k$ and $\mathbf{x}^{k+1}$ are minimized in the *local* and *global* steps, respectively. In the local step, $\mathbf{x}^k$ is fixed and $\mathbf{z}_j^k$ is solved:

$$\underset{\mathbf{z}_j^k \in \mathcal{M}_j}{\arg\min} \|\mathbf{G}_j\mathbf{x}^k - \mathbf{z}_j^k\|^2_F \tag{6}$$

where $\mathcal{M}_j$ is the constraint manifold, which is defined depending on the element type, e.g., tetrahedron or spring. In the global step, $\mathbf{z}_j^k$ is fixed and $\mathbf{x}^{k+1}$ is solved by minimizing the objective function of Eq. (5). This leads to the following system of linear equations:

$$\mathbf{A}_{\text{PD}}\mathbf{x}^{k+1} = \frac{\mathbf{M}}{h^2}\mathbf{y} + \sum_j w_j\mathbf{G}_j^T\mathbf{z}_j^k \tag{7}$$

where $\mathbf{A}_{\text{PD}} = \frac{\mathbf{M}}{h^2} + \sum_j w_j\mathbf{G}_j^T\mathbf{G}_j$. Note that the system matrix $\mathbf{A}_{\text{PD}}$ is symmetric positive-definite and remains constant during the local/global iterations.

According to Liu et al. [2017], PD can be understood as a quasi-Newton method that minimizes Eq. (5) using an approximated Hessian matrix (i.e., $\mathbf{A}_{\text{PD}} \approx \nabla^2 g(\mathbf{x}^k)$). Therefore, a single iteration of local/global steps in PD can be rewritten as the following optimization problem with respect to $\Delta\mathbf{x}^k$:

$$\underset{\Delta\mathbf{x}^k}{\arg\min}(\nabla g(\mathbf{x}^k))^T\Delta\mathbf{x}^k + \frac{1}{2}(\Delta\mathbf{x}^k)^T\mathbf{A}_{\text{PD}}\Delta\mathbf{x}^k \tag{8}$$

where $\Delta\mathbf{x}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$.

Finding $\Delta\mathbf{x}^k$ is equivalent to solving the following linear system:

$$\mathbf{A}_{\text{PD}}\Delta\mathbf{x}^k = -\nabla g(\mathbf{x}^k) \tag{9}$$

Note that computing $\nabla g(\mathbf{x}^k)$ involves the local step and updating $\mathbf{x}^{k+1}$ with the solution of Eq. (9) is equivalent to the global step.

4 CONSTRAINED PROJECTIVE DYNAMICS

Due to its nature of the underlying backward Euler scheme, the PD solver is stable but suffers from the lack of ability to conserve the angular momentum and total energy, which often leads to excessive artificial damping in rotational and elastic motions. This section describes our method that tackles the artificial damping problem. The key idea is to introduce new energy-momentum constraints to the objective function of PD and formulate a constrained optimization problem such that the results are bound to the constant energy-momentum manifold. Not only is our method stable thanks to the nature of the PD solver, but it also generates vivid motions due to its capability of preventing undesirable damping.

Moreover, our formulation provides intuitive control channels for linear, angular and elastic motions. Controlling the energy-momentum constraints, users can easily create a variety of effects related to the physical relationships among the conserved quantities.

4.1 Position-based Energy and Momenta

We consider three important physical quantities: total linear momentum $P(\in \mathbb{R}^3)$, total angular momentum $L(\in \mathbb{R}^3)$ and total energy $H(\in \mathbb{R})$. In our mesh configuration, these quantities are defined as follows:

$$\begin{aligned}P(\mathbf{v}) &= \sum_i m^i\mathbf{v}^i\\ L(\mathbf{x}, \mathbf{v}) &= \sum_i \mathbf{x}^i \times m^i\mathbf{v}^i\\ H(\mathbf{x}, \mathbf{v}) &= \frac{1}{2}\|\mathbf{v}\|^2_{\mathbf{M}} + \sum_j W_j(\mathbf{x})\end{aligned} \tag{10}$$

where m^i, $\mathbf{v}^i$ and $\mathbf{x}^i$ represent respectively the mass, velocity and position of the i-th vertex.

Taking Eq. (10) as is in our formulation would introduce a new variable, $\mathbf{v}$, to the objective function in addition to $\mathbf{x}$ thus may overburden the solver. To avoid this, we replace $\mathbf{v}$ in Eq. (10) with the backward Euler approximation, $(\mathbf{x} - \mathbf{x}_n)/h$. Then, the momenta

and energy are defined as follows:

$$\begin{aligned} P(\mathbf{x}) &= \frac{1}{h}\sum_i m^i(\mathbf{x}^i - \mathbf{x}_n^i) \\ L(\mathbf{x}) &= \frac{1}{h}\sum_i m^i\mathbf{x}^i \times (\mathbf{x}^i - \mathbf{x}_n^i) \\ H(\mathbf{x}) &= \frac{1}{2h^2}\|\mathbf{x}-\mathbf{x}_n\|_{\mathbf{M}}^2 + \sum_j W_j(\mathbf{x}) \end{aligned} \tag{11}$$

Note that the variable count is reduced by half due to the replacement.

4.2 Constrained Optimization Problem

We aim to constrain the solutions of the optimization problem such that the new state $\mathbf{x}$ conserves the energy and momenta. Let P_{n+1}, L_{n+1} and H_{n+1} denote the physical quantities to be conserved at t_{n+1}. In a closed system where $\mathbf{f}_{\text{ext}} = \mathbf{0}$, $P_{n+1} = P_n$, $L_{n+1} = L_n$, and $H_{n+1} = H_n$. In general, however, considering the external forces such as gravity, we update each quantity with an approximation of the amount of transferred energy (also known as power):

$$\begin{aligned} P_{n+1} &= P_n + h\sum_i \mathbf{f}_{\text{ext}}^i \\ L_{n+1} &= L_n + h\sum_i \mathbf{f}_{\text{ext}}^i \times \mathbf{x}_n^i \\ H_{n+1} &= H_n + h\mathbf{f}_{\text{ext}}^T\mathbf{v}_n \end{aligned} \tag{12}$$

Unfortunately, Eq. (12) does not take into account the dependency between the energy and momenta and therefore is prone to incompatibility. For example, suppose that $P_{n+1} > 0$, $L_{n+1} = 0$, and $H_{n+1} = 0$. P_{n+1} returns a non-zero kinetic energy, but it is incompatible with H_{n+1} that should be zero. This problem and a solution were discussed in FEPR. Inspired by their solution, we have developed our own. First, the energy coming from the momenta is defined:

$$K = \frac{1}{2}P_{n+1}\cdot P_{n+1}/\sum_i m^i + \frac{1}{2}(I^{-1}L_{n+1})\cdot L_{n+1} \tag{13}$$

where I^{-1} represents the inverse of the inertia tensor and is approximated using $\mathbf{x}_n$ [Dinev et al. 2018a]. Then, we specify the constraints as follows:

$$\begin{aligned} P(\mathbf{x}) &= P_{n+1} \\ L(\mathbf{x}) &= L_{n+1} \\ H(\mathbf{x}) &= (1-\alpha)H_{n+1} + \alpha K \end{aligned} \tag{14}$$

where α interpolates H_{n+1} and K. Finally, our constrained optimization problem is given as follows:

$$\begin{aligned} \underset{\mathbf{x},\alpha}{\arg\min} \quad & g(\mathbf{x}) + \frac{1}{2}\epsilon\alpha^2 \\ \text{subject to} \quad & P(\mathbf{x}) = P_{n+1} \\ & L(\mathbf{x}) = L_{n+1} \\ & H(\mathbf{x}) = (1-\alpha)H_{n+1} + \alpha K \end{aligned} \tag{15}$$

where $g(\mathbf{x})$ is defined in Eq. (5) and ϵ is the regularization weight. (We name $\frac{1}{2}\epsilon\alpha^2$ the *regularization* term.)

Note that α is solved via optimization. When P_{n+1}, L_{n+1} and H_{n+1} are compatible with each other, α will end up with zero, abandoning K. Otherwise, α guarantees that the energy becomes compatible with the momenta. In the above example, where $P_{n+1} > 0$, $L_{n+1} = 0$, and $H_{n+1} = 0$, α will be close to one.

4.3 Numerical Solution

Our constrained optimization problem in Eq. (15) is solved for the new state, i.e., positions $\mathbf{x}$ and the auxiliary variable α. All variables are stacked to make $\mathbf{q}$, i.e., $\mathbf{q} = [\mathbf{x};\ \alpha] (\in \mathbb{R}^{3m+1})$. Then, a vector-valued function can be defined, $\mathbf{c}(\mathbf{q}) : \mathbb{R}^{3m+1} \to \mathbb{R}^7$. It aggregates $\mathbf{c}_P(\mathbf{q}) = P(\mathbf{q}) - P_{n+1}$ (for the linear momentum), $\mathbf{c}_L(\mathbf{q}) = L(\mathbf{q}) - L_{n+1}$ (for the angular momentum), and $\mathbf{c}_H(\mathbf{q}) = H(\mathbf{q}) - (1-\alpha)H_{n+1} - \alpha K$ (for the total energy). Then, our constrained optimization problem is rephrased as follows:

$$\underset{\mathbf{q}}{\arg\min}\ \mathcal{G}(\mathbf{q}) \quad \text{subject to} \quad \mathbf{c}(\mathbf{q}) = \mathbf{0} \tag{16}$$

where $\mathcal{G}(\mathbf{q}) = g(\mathbf{x}) + \frac{1}{2}\epsilon\alpha^2$.

We solve the problem using sequential quadratic programming (SQP) [Nocedal and Wright 2006], which computes the search direction $\Delta\mathbf{q}^\kappa$ at the κ-th iteration via the quadratic programming subproblem as follows:

$$\begin{aligned} &\underset{\Delta\mathbf{q}^\kappa}{\arg\min}\ \nabla\mathcal{G}(\mathbf{q}^\kappa)^T\Delta\mathbf{q}^\kappa + \frac{1}{2}(\Delta\mathbf{q}^\kappa)^T\nabla^2_{\mathbf{qq}}\mathcal{L}(\mathbf{q}^\kappa,\lambda^\kappa)\Delta\mathbf{q}^\kappa \\ &\text{subject to} \quad \mathbf{c}(\mathbf{q}^\kappa) + \nabla\mathbf{c}(\mathbf{q}^\kappa)^T\Delta\mathbf{q}^\kappa = \mathbf{0} \end{aligned} \tag{17}$$

where $\mathcal{L}(\mathbf{q},\lambda) = \mathcal{G}(\mathbf{q}) + \mathbf{c}(\mathbf{q})^T\lambda$, which is the Lagrangian function for Eq. (16). Then, the first-order necessary conditions (also known as KKT conditions) for the solution of Eq. (17) state the following system of equations:

$$\begin{bmatrix} \nabla^2_{\mathbf{qq}}\mathcal{L}(\mathbf{q}^\kappa,\lambda^\kappa) & \nabla\mathbf{c}(\mathbf{q}^\kappa) \\ \nabla\mathbf{c}(\mathbf{q}^\kappa)^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{q}^\kappa \\ \lambda^{\kappa+1} \end{bmatrix} = -\begin{bmatrix} \nabla\mathcal{G}(\mathbf{q}^\kappa) \\ \mathbf{c}(\mathbf{q}^\kappa) \end{bmatrix} \tag{18}$$

Although this SQP method applying the Newton's method converges fast, evaluating $\nabla^2_{\mathbf{qq}}\mathcal{L}(\mathbf{q}^\kappa,\lambda^\kappa)$ for solving the linear system at each iteration is costly. Reviving the quasi-Newton interpretation of PD and the effective approximation $\mathbf{A}_{\text{PD}}$ ($\approx \nabla^2 g(\mathbf{x}^k)$) discussed in Section 3, we can accelerate the linear system solve using a quasi-Newton approximation $\mathbf{A}$ ($\approx \nabla^2_{\mathbf{qq}}\mathcal{L}(\mathbf{q}^\kappa,\lambda^\kappa)$):

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{\text{PD}} & 0 \\ 0 & \epsilon \end{bmatrix} \tag{19}$$

Then, the linear system of Eq. (18) is simplified as follows:

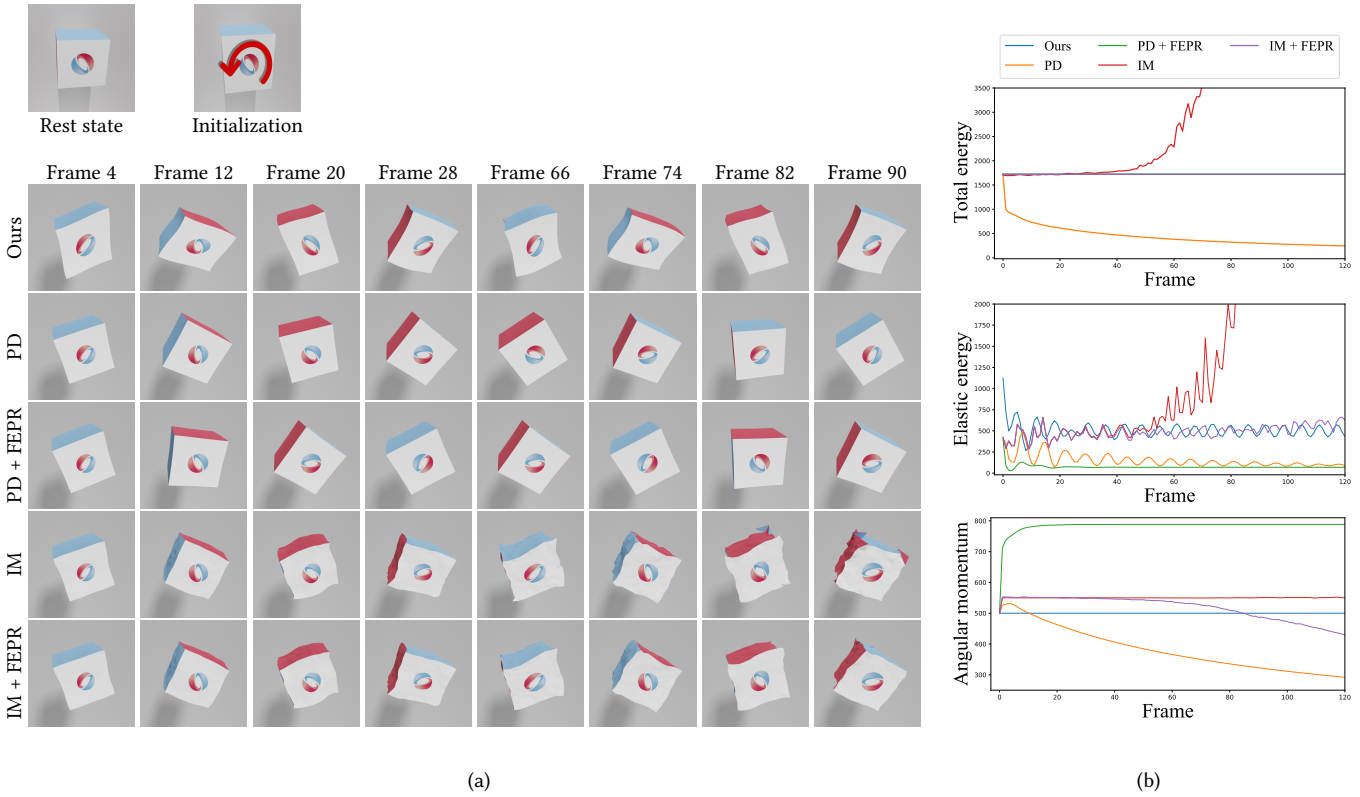
$$\begin{bmatrix} \mathbf{A} & \nabla\mathbf{c}(\mathbf{q}^\kappa) \\ \nabla\mathbf{c}(\mathbf{q}^\kappa)^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{q}^\kappa \\ \lambda^{\kappa+1} \end{bmatrix} = -\begin{bmatrix} \nabla\mathcal{G}(\mathbf{q}^\kappa) \\ \mathbf{c}(\mathbf{q}^\kappa) \end{bmatrix} \tag{20}$$

Thanks to the structure of the system matrix of Eq. (20), we can further accelerate its solve using the Schur complement. Finally, $\mathbf{q}^{\kappa+1}$ is solved efficiently via the following equations:

$$\begin{aligned} \mathbf{S}^\kappa\lambda^{\kappa+1} &= \mathbf{c}(\mathbf{q}^\kappa) - \nabla\mathbf{c}(\mathbf{q}^\kappa)^T\mathbf{A}^{-1}\nabla\mathcal{G}(\mathbf{q}^\kappa) \\ \mathbf{q}^{\kappa+1} &= \mathbf{q}^\kappa - \mathbf{A}^{-1}\left(\nabla\mathcal{G}(\mathbf{q}^\kappa) + \nabla\mathbf{c}(\mathbf{q}^\kappa)\lambda^{\kappa+1}\right) \end{aligned} \tag{21}$$

where $\mathbf{S}^\kappa = \nabla\mathbf{c}(\mathbf{q}^\kappa)^T\mathbf{A}^{-1}\nabla\mathbf{c}(\mathbf{q}^\kappa)$ ($\in \mathbb{R}^{7\times 7}$).

Eq. (21) includes PD's quasi-Newton update of $\Delta\mathbf{x}^k$ (presented in Eq. (9)). Recall that only a scalar variable α is appended to $\mathbf{x}$ to make $\mathbf{q}$ and only the regularization term $\frac{1}{2}\epsilon\alpha^2$ is added to g to make $\mathcal{G}$. Therefore, the cost of our solve added to that of the PD solve is kept small. Let us elaborate the components in Eq. (21). Evaluating $\nabla\mathcal{G}(\mathbf{q})$



(a) (b)

Fig. 2. Cube is stretched vertically and assigned an angular momentum. (a) Five different solvers simulate the same cube. (b) The graphs for the total energy, elastic energy and angular momentum.

requires little additional cost because $\nabla\mathcal{G}(\mathbf{q}) = [\nabla g(\mathbf{x});\ \epsilon\alpha]$ and $\nabla g(\mathbf{x})$ is computed in the PD solve. A similar argument can be made for $\mathbf{A}^{-1}\nabla\mathcal{G}(\mathbf{q})$. $\nabla\mathbf{c}$ consists of $\nabla\mathbf{c}_P$, $\nabla\mathbf{c}_L$ and $\nabla\mathbf{c}_H$: (1) $\nabla\mathbf{c}_P$ remains constant "over the simulation" and therefore $\nabla\mathbf{c}_P$ and $\mathbf{A}^{-1}\nabla\mathbf{c}_P$ are precomputed. (2) $\nabla\mathbf{c}_L$ remains constant "over the solver iterations" and therefore $\nabla\mathbf{c}_L$ and $\mathbf{A}^{-1}\nabla\mathbf{c}_L$ are updated at each solve for the new state. (3) $\nabla\mathbf{c}_H$ and $\mathbf{A}^{-1}\nabla\mathbf{c}_H$ are computed "per iteration" but the cost added to that of the PD solve is negligible because $\nabla\mathbf{c}_H = [\nabla g + h\mathbf{M}\mathbf{v}_n + h^2\mathbf{f}_{\text{ext}};\ H_{n+1} - K]$ and $\mathbf{A}^{-1}\nabla\mathbf{c}_H = [\mathbf{A}_{\text{PD}}^{-1}(\nabla g + h\mathbf{M}\mathbf{v}_n + h^2\mathbf{f}_{\text{ext}});\ (H_{n+1} - K)/\epsilon]$. Note that ∇g and $\mathbf{A}_{\text{PD}}^{-1}\nabla g$ are computed during the PD solve whereas $h\mathbf{M}\mathbf{v}_n + h^2\mathbf{f}_{\text{ext}}$, $\mathbf{A}_{\text{PD}}^{-1}(h\mathbf{M}\mathbf{v}_n + h^2\mathbf{f}_{\text{ext}})$ and $(H_{n+1} - K)$ remain constant "over the solver iterations." Note that $\mathbf{S}$ is positive definite and therefore is invertible. Inverting $\mathbf{S}$ is fast as it is a 7×7 matrix.

Starting from $\mathbf{q}^0 = [\mathbf{x}^0;\ 0]$, we iteratively update $\mathbf{q}$ until either $\mathbf{q}^\kappa$ converges in terms of the constraints (e.g., $|\mathbf{c}(\mathbf{q})| < 10^{-4}$) or the number of iterations reaches a predefined maximum. The overall algorithm of our method is presented in **ALGORITHM** 1.

ALGORITHM 1: Constrained Projective Dynamics

```
Input: Precomputed ∇c_P and A^{-1}∇c_P
y_n = x_n + h v_n + h^2 M^{-1} f_ext
Update P, L and H (Sections 4.2 and 4.4)
Update ∇c_L, A^{-1}∇c_L and A_PD^{-1}(hMv_n + h^2 f_ext)
while (not converged) and (κ < maxSolverIteration) do
    Compute ∇G(q^κ), A^{-1}∇G(q^κ) and A^{-1}∇c_H
    Solve the linear system of Eq. (21) for q^{κ+1} (= [x^{κ+1}; α^{κ+1}])
    κ = κ + 1
end
x_{n+1} = x^κ
v_{n+1} = (x_{n+1} − x_n)/h
```

4.4 Attachment and Collision

We can handle *attachments* and *collisions* as typical PD solvers do, i.e., they can be implemented with zero rest-length springs and temporarily inserted repulsion springs, respectively. We define the energy of a repulsion spring as in [Dinev et al. 2018b]:

$$E_{\text{col}}(\mathbf{x}) = \begin{cases} \frac{1}{3}k_{col}((\mathbf{S}\mathbf{x} - \mathbf{x}_{\text{surf}})^T\mathbf{n})^3 & (\mathbf{S}\mathbf{x} - \mathbf{x}_{\text{surf}})^T\mathbf{n} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

where k_{col} is the stiffness of the repulsion spring, $\mathbf{S}$ is a matrix that selects the colliding vertex, $\mathbf{x}_{\text{surf}}$ is the surface point closest to the colliding vertex, and $\mathbf{n}$ is the collision normal. Because collision may change the energy and momenta, we calculate the force generated by collision by taking the gradient of $E_{\text{col}}(\mathbf{x})$. Then, it is added to $\mathbf{f}_{\text{ext}}$

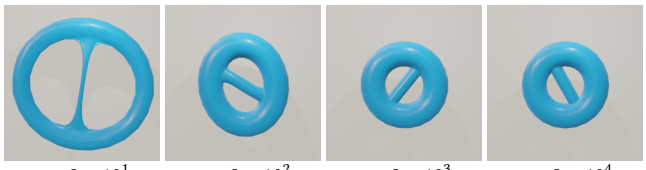


Fig. 3. A torus rotating with a dynamically changing material property: The softer the torus is, the more stretched it is.

along with other external forces, such as gravity, at the beginning of every frame.

4.5 Simulation Control

Our constrained optimization formulation allows users to directly control the energy and momenta using the user-defined values (denoted as P_{user}, L_{user} and H_{user}):

$$\begin{aligned} P_{n+1} &= P_{\text{user}} \\ L_{n+1} &= L_{\text{user}} \\ H_{n+1} &= H_{\text{user}} \end{aligned} \tag{23}$$

On the other hand, for example, suppose that the user wants to apply a *momentum-conserving damping* as in [Kharevych et al. 2006; Li et al. 2018; Müller et al. 2007]. In our formulation, it is implemented as follows:

$$\begin{aligned} P_{n+1} &= P_n \\ L_{n+1} &= L_n \\ H_{n+1} &= H_n - \gamma h(H_n - K) \end{aligned} \tag{24}$$

where γ is a damping coefficient and K is defined in Eq. (13). Then, our solver naturally damps out only the elastic motion while conserving the momenta. It is worth noting that the user need not worry about the incompatibility between energy and momenta since it is resolved automatically by Eq. (15), as discussed in Section 4.2.

5 RESULTS

This section reports the results of our experiments made on an AMD Ryzen 9 3900X 3.8 GHz processor. Our method is implemented using OpenMP for parallel computing. (Evaluations of $\nabla\mathcal{G}(\mathbf{q})$ and $\nabla\mathbf{c}_L(\mathbf{q})$ (in Eq. (21)) are parallelized.) With additional optimization, the efficiency of our implementation can be further enhanced.

Each experiment is made not only with a unique deformable object but also under a distinct initial setup. In the experiments, our method is compared with PD, PD + FEPR (FEPR after PD), implicit midpoint (IM), or IM + FEPR (FEPR after IM). The time step size h is fixed to $1/30$ seconds for all solvers. By default, both PD and IM use 10 iterations. The same termination condition is used for our method and FEPR: conservation error $< 10^{-4}$.

Table 1 summarizes the statistics of the experiments made with our method. Observe that our method adds an insignificant amount of computation to PD; on average, it takes about 6.3% more time than PD "per solver iteration."

Cube: In Fig. 2, the cube is vertically stretched and is given an angular momentum (as depicted with an arrow). Fig. 2-(a) shows that

Fig. 4. Dzhanibekov effect. A T-pipe is initialized with two different angular-momentum setups in terms of the axis of rotation. Our method produces the distinctive motions conserving the angular momentum: (top) unstable rotation around the intermediate principal axis and (bottom) stable rotation around the third principal axis.

our method conserves the elastic motion as well as the rotational motion. In contrast, PD suffers from artificial damping; the elastic motion immediately vanishes and the rotational motion fades away. (See the accompanying video.) The graphs in Fig. 2-(b) show that our method (in blue) conserves the total energy and angular momentum and successfully presents the elastic energy that oscillates over time. In contrast, the artificial damping problem of PD (in orange) is clearly depicted.

The third row in Fig. 2-(a) shows the results of PD + FEPR. The video reveals more evidently that the cube appears still rigid whereas its rotational motion revives. It is explained by the graphs of PD + FEPR (in green) in Fig. 2-(b). The total energy is conserved but most of it comes from the angular momentum. The elastic energy is underestimated, resulting in the almost rigid motion. The angular momentum graph of PD (in orange) shows an impulsive increase at the early frames, and we speculate that it makes FEPR keep overestimating the energy attributed to the angular momentum.

The fourth row in Fig. 2-(a) shows the results of IM, which is symplectic and known to have good energy-momentum-conserving behavior [Dinev et al. 2018a]. However, IM produces the energy-momentum conserving motion only at the beginning of simulation but does not continue producing a stable motion. See the red graphs in Fig. 2-(b). As reported by Dinev et al. [2018a], IM remains stable only with a very small time step size, e.g., $h = 10^{-4}$, which is impractical for real-time applications.

The fifth row in Fig. 2-(a) shows that FEPR alleviates the instability problem of IM to some extent but cannot eliminate the surface bumpiness. In Fig. 2-(b), observe the graphs of IM + FEPR (in purple): Even though the total energy is conserved, the elastic energy is overestimated whereas the angular momentum is underestimated. The video clearly reveals the problems.

In the cube example, PD + FEPR takes 169% more time than our method; while the average iteration count of our method is 5.1, PD and FEPR consume 10 and 2 iterations, respectively. On the other hand, IM + FEPR takes 209% more time than ours; IM and FEPR consume 10 and 2.5 iterations, respectively. The cube example demonstrates that our method not only produces more desirable motions (as shown in Fig. 2-(a)) but also is more efficient than FEPR.

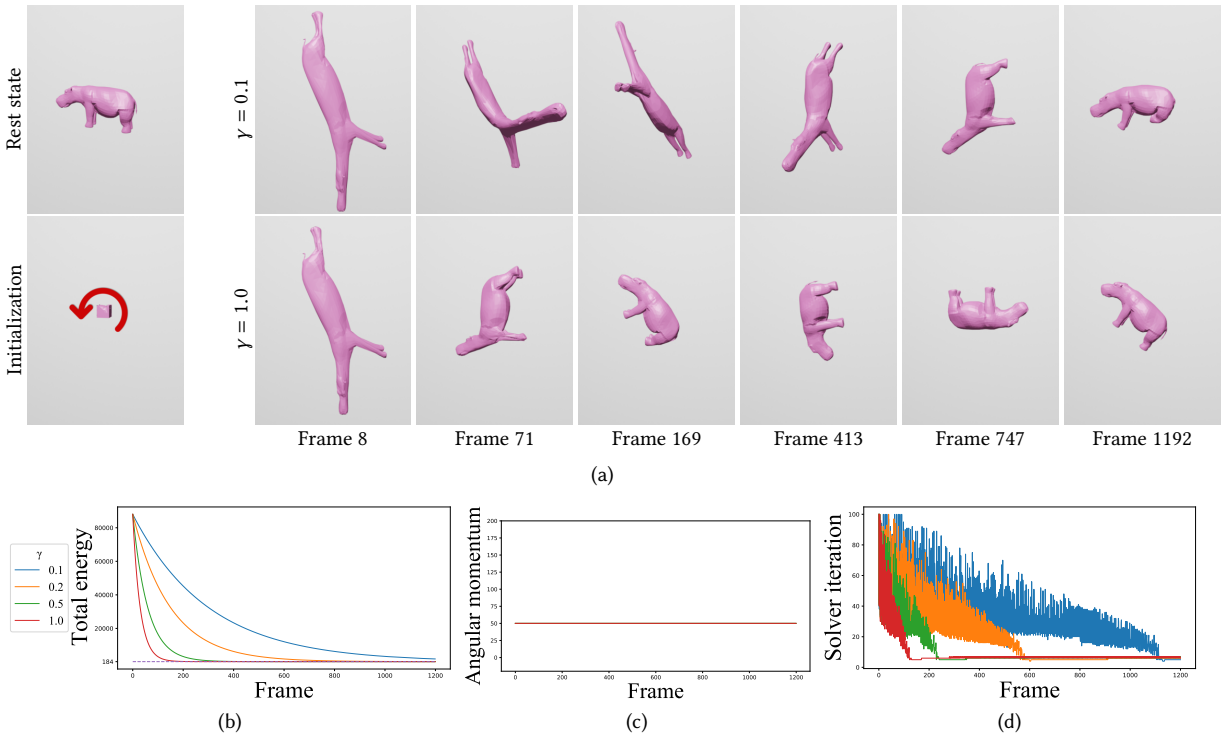


Fig. 5. Momentum-conserving damping. (a) The smaller the damping coefficient is ($\gamma = 0.1$), the longer the deforming motions take. The larger damping coefficient ($\gamma = 1.0$) quickly leads to rigid motions. (b) Total energy for four different damping coefficients, $\gamma \in \{0.1, 0.2, 0.5, 1.0\}$. (c) Angular momentum. (d) Solver iterations.

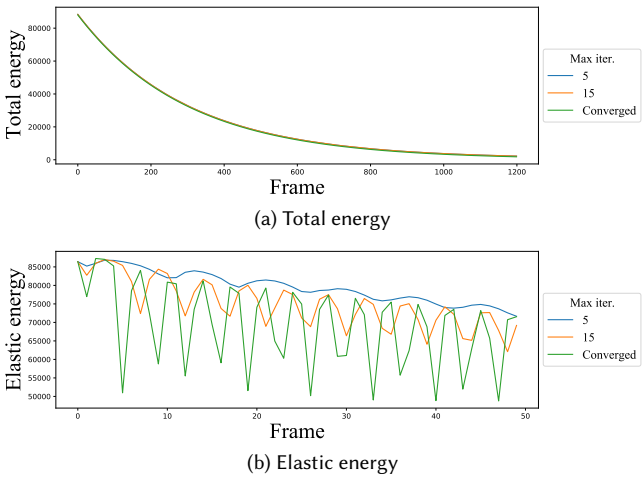


Fig. 6. Energy changes with different solver iteration counts in the hippo example ($\gamma = 0.1$).

Torus: We set an angular momentum to a torus such that it rotates about the axis of revolution (used to create the torus) and is stretched due to the centrifugal force. While the torus is rotating, its material property changes from softer to harder. Fig. 3 shows the change over time. In the figure, μ represents the first Lamé parameter [Sifakis and Barbic 2012]; the smaller μ is, the softer the material is, which results in a more stretched torus. Fig. 3 demonstrates that our energy-momentum conserving scheme nicely incorporates different material properties, i.e., realizing different elastic energy constraints. Observe that, when the material is hard enough, e.g., $\mu \geq 2 \times 10^3$, the torus keeps the similar rigid rotational motions. Moreover, the torus volume is also conserved by our solver; the more stretched, the thinner.

T-pipe: Fig. 4 demonstrates that our method is able to reproduce an interesting physical phenomenon, *Dzhanibekov effect*, also known as the *intermediate axis theorem* [Levi 2014]. A T-pipe is given two angular momenta. Whereas T-pipe rotating around its intermediate axis is unstable (the first row), rotation around the third principal axis is stable (the second row).

Hippo: Fig. 5 demonstrates the *momentum-conserving damping* presented in Eq. (24). In order to initiate compelling elastic motions, the hippo is squeezed by projecting its vertices into a small box. Being assigned an angular momentum, it is released. Despite the challenging configuration caused by excessive deformation, our method stably simulates the elastic motions.

Fig. 5-(a) compares the results of simulation with different damping coefficients: $\gamma = 0.1$ vs. $\gamma = 1.0$. Observe that smaller γ leads to

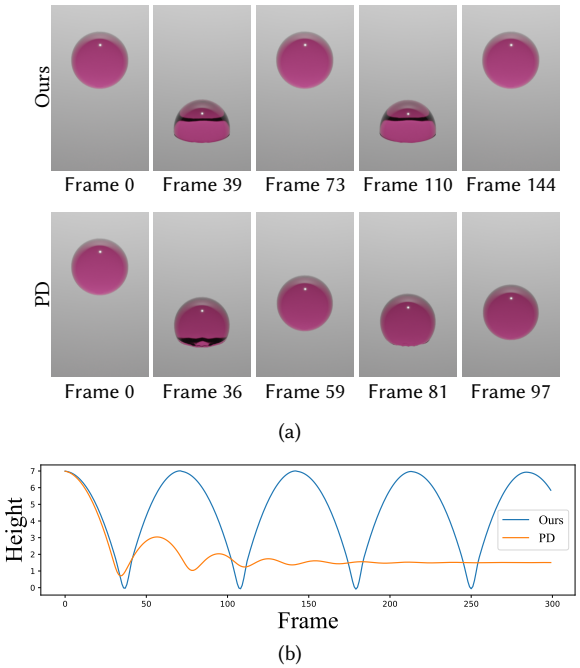


Fig. 7. Free falling ball. (a) Our method successfully conserves the energy and momenta thus recovers the initial height whereas PD suffers from artificial damping. (b) The ball's heights are traced.

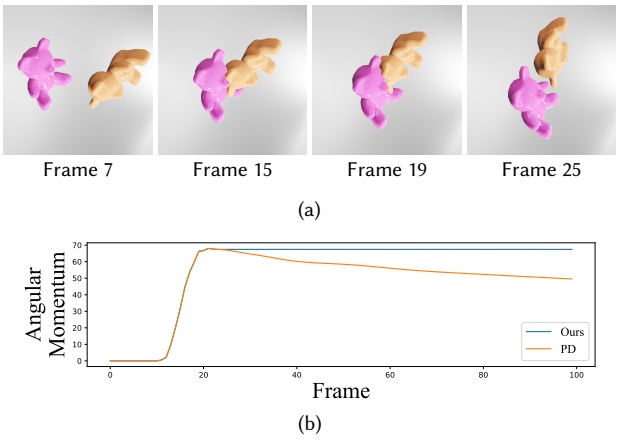


Fig. 8. Colliding bears. (a) The repulsive forces bring about the angular momenta. (b) The total angular-momentum magnitudes traced over time.

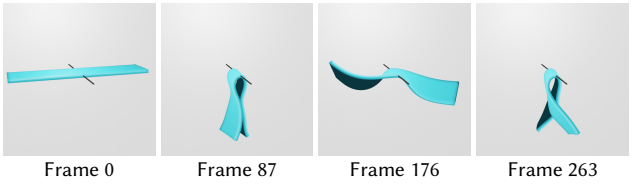


Fig. 9. A self-collision example with a hanging strip under gravity.

slower damping. For example, compare the images at Frame 169. (See the accompanying video.)

Fig. 5-(b) depicts the decreasing total energy with different γ values, 0.1, 0.2, 0.5 and 1.0, whereas Fig. 5-(c) shows the conserved angular momentum. In Fig. 5-(b), the energies converge to a non-zero value that represents K derived from the conserved angular momentum.

Fig. 5-(d) plots the solver's iteration counts over the simulation. Similar patterns are observed for all γ values: As the energy is getting damped out, fewer number of iterations are required, converging to around six.

Table 1 shows that 18.5 iterations are made on average. The hippo consumes considerably more iterations than the other objects (due to the challenging initial configuration.) This led us to test two more setups, where the iteration counts are fixed to 5 and 15. Fig. 6 plots the total and elastic energies traced in three setups when $\gamma = 0.1$. Whereas the total energy decreases at the same rate in all solutions (Fig. 6-(a)), the elastic energy in the new solutions decreases more smoothly but resembles the global behavior of the converged solution's (Fig. 6-(b)). The solution with 5 iterations does not produce visually pleasing animation; the hippo does not sufficiently oscillate. In contrast, the behavior observed with 15 iterations is closer to that of the converged solution. (See the accompanying video.) Under a limited computing budget, it would be a reasonable compromise.

Ball: In Fig. 7-(a), the ball falls down under the sole influence of gravity with no initial momenta, hits the floor, and then bounces back. The frames are captured when the ball reaches the maximum and minimum heights during bounces. As can be observed from the first row of Fig. 7-(a), our method recovers the initial height even after several bounces whereas PD suffers from excessive damping as shown at the second row. The graphs in Fig. 7-(b) depict the heights over time and demonstrate the ability of our method to conserve energy and momenta.

Colliding bears: In our method, collision is handled by computing the repulsive forces (Eq. (22)) and then updating the energy and momenta (Eq. (12)). In Fig. 7, collision between the ball and floor is resolved that way. Collisions between dynamic objects are handled in the same manner. In Fig. 8, two bears are initialized with linear momenta such that they approach each other. They collide at frame 15 and start to separate at frame 20. (A spatial hashing algorithm is used to detect collisions [Teschner et al. 2003].) In this example, each bear's motion is governed by its own energy-momentum constraints. In Fig. 8-(b), each graph shows the total magnitude of two bears' angular momenta (generated by the repulsive forces) over time. Note that our method conserves the angular momenta even when multiple dynamic objects collide. In contrast, PD suffers from artificial damping. Fig. 9 shows another example, where a deformable strip hanging on a bar falls under gravity, resulting in *self-collision*. Using the energy-momentum constraints, our method produces vivid motions of the self-colliding object.

Trampoline: Fig. 10 shows the simulation results of a trampoline, which is attached to four fixed points and then given a linear

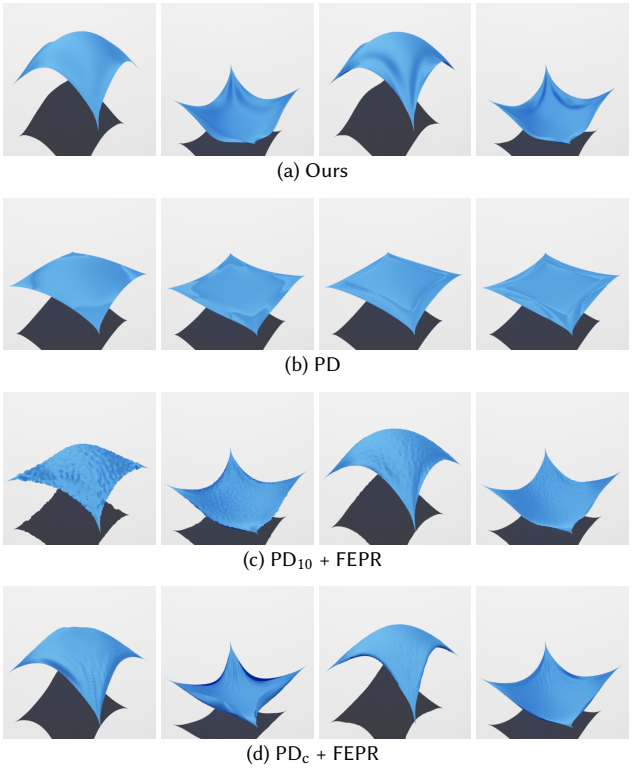


Fig. 10. Initially, the trampoline is stretched horizontally and attached to four fixed points. Then, a linear momentum is given along the vertical direction, producing oscillating motions. (Unlike the other objects used in our experiments, which are represented in tetrahedral volumetric meshes, the trampoline is represented in a square surface mesh and thus uses the mass-spring constraint in the solver.)

momentum along the vertical direction. In Fig. 10, the frames are captured when the center of mass reaches the peak or the bottom. As shown in Fig. 10-(a), our method successfully produces the oscillating motion thanks to its ability of conserving the energy and momenta. Table 1 shows that our method takes only approximately 3 iterations, proving its efficiency. Fig. 10-(b) shows that PD suffers from artificial damping; the oscillating motion of the trampoline quickly vanishes.

In this experiment, FEPR is tested with two different setups of the PD solver in order to see how the base solver's results affect FEPR. First, the PD solver's iteration count is fixed to 10, and then FEPR is applied; this is denoted by PD_{10} + FEPR. Despite the energy-momentum conserving projection of FEPR, Fig. 10-(c) shows that the elastic motions suffer from the bumpiness artifact. Here, FEPR takes approximately 6.5 iterations. PD_{10} + FEPR spends 340% more time (31ms) than ours (9.1ms) per frame but does not produce the desirable result. Second, the PD solver is run until converged using the threshold of 10^{-4}; this is denoted by PD_c + FEPR. As shown in Fig. 10-(d), the 'better' solution of the base PD solver helps FEPR produce the 'smoother' results alleviating the bumpiness artifact. Here, PD and FEPR take approximately 19 and 4 iterations, respectively.

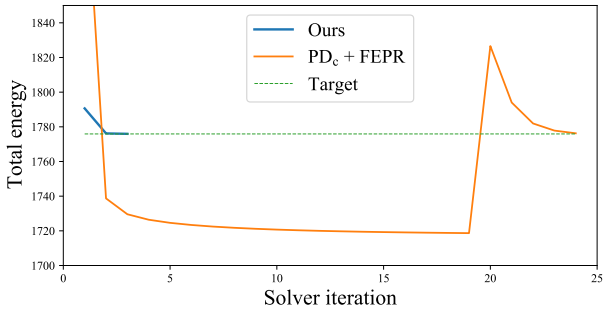


Fig. 11. At the 40^{th} frame of the trampoline test presented in Fig. 10, the total energy is plotted over the solver's iterations. Our method finds a *shortcut* whereas FEPR detours after the base solver's steps.

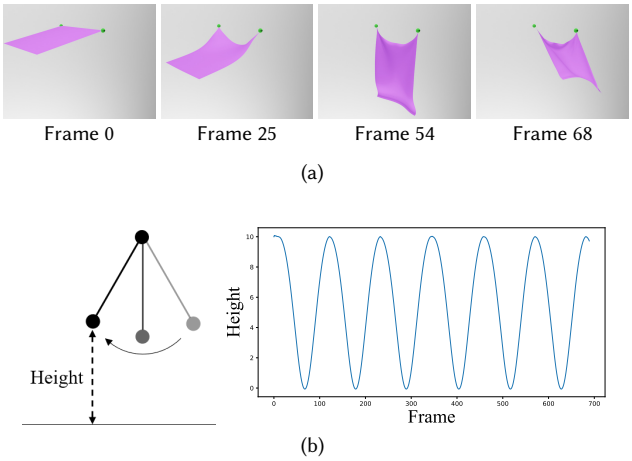


Fig. 12. Swinging motions. (a) A mass-spring cloth falling under gravity. (b) A pendulum simulation and its height change over time.

PD_c + FEPR spends 428% more time (39.5ms) than ours (9.1ms) per frame.

From the trampoline test, we select the 40^{th} frame and plot the total energy over the solver iterations of our method and PD_c + FEPR. Fig. 11 clearly shows the advantage of our "solver-integrated constraint approach" over the "projecting-after-solver approach" of FEPR. By integrating the constraints into the solver, our method can quickly find the solution while avoiding the unnecessary *detouring*.

Bear: Fig. 1 shows our user-controlled simulation result with a deformable bear. Initially, the user pushed the bear to the right. This gave rise to the energy and momenta, and the bear started moving with deforming motions. The user pushed it one more time to the right, making the linear momentum increased. Finally, the user activated the attachment constraints so that the vertices belonging to the bear's hand were connected to the metal ball. This made the bear rotate around the ball. See the accompanying video, which also shows the user interface for simulation control.

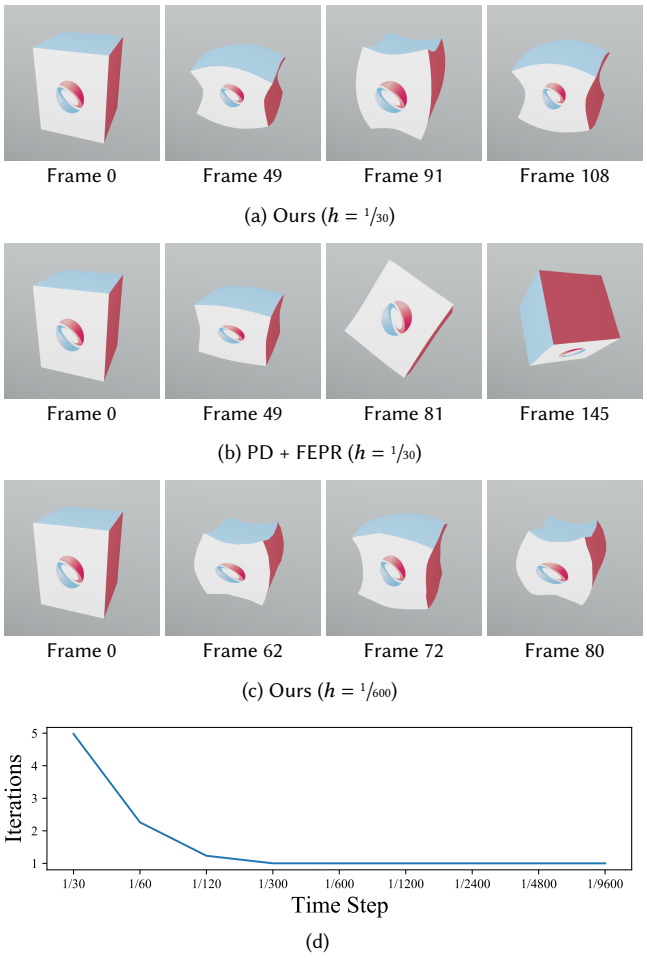


Fig. 13. The cube (used in Fig. 2) is vertically stretched with zero momenta at the initial setup. (a) Our method produces oscillating motions respecting the initial zero momenta. (b) PD + FEPR produces undesired rotational motions. (c) The results of our method with a smaller time step size are similar to those shown in (a). (d) The average solver iteration counts over different time step sizes.

6 DISCUSSION AND LIMITATION

Energy-momentum conservation is an important goal of many numerical methods for simulating deformable objects. To this end, a variety of approaches have been proposed. Early studies that considered only the energy conservation showed the non-physical transformation of energy by which the global rotational motion slows down and eventually vanishes [Dinev et al. 2018a; Hughes et al. 1978]. It is well-known that conserving both energy and angular momentum helps solve the problem [Hairer 2006; Su et al. 2012] and so does our method. Fig. 12 demonstrates that our method does not suffer from such a problem but produces the desired stable swinging motions in both examples.

Aiming for the energy-momentum conservation, our work was inspired by FEPR, but our approach is different from FEPR's. Similar to FEPR, our method solves a constrained optimization problem with the energy-momentum constraints. Unlike FEPR, however, we devise our own position-based constraints and handle them within PD, such that the constraints are taken into account together with the elasticity constraints. As demonstrated, this not only makes our method stable, efficient and simple to implement but also allows the conserved quantities to be controlled easily.

Our strategy for handling the potential incompatibility in the energy-momentum constraints is different from FEPR's. As presented in Section 4.2, our method respects the conservation of momenta and the induced kinetic energy. In contrast, FEPR respects the conservation of energy and uses regularization variables on the momentum constraints. In Fig. 13, the cube is initially stretched vertically with zero momenta. It should then oscillate without rigid motions. Fig. 13-(a) shows that our method produces the desired elastic motions without linear and angular motions, but Fig. 13-(b) shows that PD + FEPR produces undesired rigid motions.

By the nature of the constrained optimization formulation, our method conserves the energy and momenta at each solve, thus producing consistent simulation results even though different time step sizes are used. Fig. 13-(c) shows the results with a smaller time step size, $1/600$. Observe that the global motions in Fig. 13-(a) and -(c) are similar although the results in Fig. 13-(c) show higher-frequency motions as can be found in the accompanying video. Additionally, Fig. 13-(d) plots the average counts of solver iterations with different time step sizes: Our solver converges faster with a smaller time step size. It is because the initial guess, $\mathbf{y}$ (introduced in Eq. (2)), made with the smaller time step size would be closer to the energy-momentum manifold.

Despite its stability and efficiency, however, a drawback of our method is that it is tightly coupled with the underlying time integration scheme. This is contrasted with the flexibility of FEPR in choosing a base solver. On the other hand, we do not claim that our method is physically accurate although conservation of energy and momenta is our key contribution and one of the important aspects of accuracy in physics-based simulations. For example, our method does not generate periodic elastic motions and more elaborate motions such as elastic waves.

7 CONCLUSION AND FUTURE WORK

In this paper, we proposed the constrained projective dynamics method that solves the new constrained minimization problem of numerical time integration for deformable object simulations. Our method effectively and efficiently conserves the important physical quantities (i.e., the total energy and momenta) thus avoids undesired damping artifact. Through a variety of animation scenarios, we demonstrated that our method yields vivid rotational and deformable motions in real-time. Furthermore, our method enables users to readily create the desired animations by directly controlling the physical quantities to be conserved on the fly.

In this work, we realized our method solely with the typical time integration scheme for PD (i.e., implicit Euler) and experimented its simulation efficiency and stability. Despite the success, implementing our approach with other numerical integration schemes (such as the implicit midpoint method) will be an interesting future work.

Our solver is based on the quasi-Newton interpretation of PD. However, all the presented experiments focused on the elastic potential constraints of the original PD and thus did not perform elaborate Hessian approximations and line search algorithms, which are required to handle more general (e.g., hyperelastic) materials [Liu et al. 2017]. Extending our energy-momentum conserving technique to support such materials will make up another future work.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments to improve this work. We also thank JaeKwang Lee and Jaehyung Doh for developing the renderer used in all examples and JaeHyun Lee for implementing the collision detection algorithm used in the colliding bears and hanging strip examples.

This research was supported by the Ministry of Science and ICT, Korea, under the ICT Creative Consilience program(IITP-2021-2020-0-01819), the Information Technology Research Center support program(IITP-2021-2020-0-01460) and the grant No.2020-0-00861 that are all supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation). This work was also supported by Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT1902-07.

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*. 43–54.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics: fusing constraint projections for fast simulation. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.

Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-Reduced Projective Dynamics. *ACM Trans. Graph.* 37, 4, Article 80 (July 2018), 13 pages. https://doi.org/10.1145/3197517.3201387

George E. Brown, Matthew Overby, Zahra Forootaninia, and Rahul Narain. 2018. Accurate Dissipative Forces in Optimization Integrators. *ACM Trans. Graph.* 37, 6, Article 282 (Dec. 2018), 14 pages. https://doi.org/10.1145/3272127.3275011

Yu Ju Chen, Seung Heon Sheen, Uri M Ascher, and Dinesh K Pai. 2020. SIERE: A Hybrid Semi-Implicit Exponential Integrator for Efficiently Simulating Stiff Deformable Objects. *ACM Transactions on Graphics (TOG)* 40, 1 (2020), 1–12.

Kwang-Jin Choi and Hyeong-Seok Ko. 2005. Stable but responsive cloth. In *ACM SIGGRAPH 2005 Courses*. 1–es.

Alex Dahl and Adam Bargteil. 2019. Global Momentum Preservation for Position-Based Dynamics. In *Motion, Interaction and Games (MIG '19)*. Association for Computing Machinery, New York, NY, USA, 1–5. https://doi.org/10.1145/3359566.3360078

Dimitar Dinev, Tiantian Liu, and Ladislav Kavan. 2018a. Stabilizing Integrators for Real-Time Physics. *ACM Transactions on Graphics (TOG)* 37, 1 (2018), 9.

Dimitar Dinev, Tiantian Liu, Jing Li, Bernhard Thomaszewski, and Ladislav Kavan. 2018b. FEPR: fast energy projection for real-time simulation of deformable objects. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Theodore F Gast, Craig Schroeder, Alexey Stomakhin, Chenfanfu Jiang, and Joseph M Teran. 2015. Optimization integrator for large time steps. *IEEE transactions on visualization and computer graphics* 21, 10 (2015), 1103–1115.

Ernst Hairer. 2006. Long-time energy conservation of numerical integrators. *Foundations of computational mathematics, Santander 2005* (2006), 162–180.

Thomas J Hughes, TK Caughey, and WK Liu. 1978. Finite-element methods for nonlinear elastodynamics which conserve energy. *Journal of Applied Mechanics, Transactions ASME* 45, 2 (1978), 366–370.

Couro Kane, Jerrold E Marsden, Michael Ortiz, and Matthew West. 2000. Variational integrators and the Newmark algorithm for conservative and dissipative mechanical systems. *International Journal for numerical methods in engineering* 49, 10 (2000), 1295–1325.

L. Kharevych, Weiwei Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun. 2006. Geometric, Variational Integrators for Computer Animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Vienna, Austria) *(SCA '06)*. Eurographics Association, Goslar, DEU, 43–51.

Martin Komaritzan and Mario Botsch. 2019. Fast Projective Skinning. In *Motion, Interaction and Games* (Newcastle upon Tyne, United Kingdom) *(MIG '19)*. Association for Computing Machinery, New York, NY, USA, Article 22, 10 pages. https://doi.org/10.1145/3359566.3360073

Tassilo Kugelstadt, Dan Koschier, and Jan Bender. 2018. Fast Corotated FEM using Operator Splitting. *Computer Graphics Forum (SCA)* 37, 8 (2018).

Robert A LaBudde and Donald Greenspan. 1975. Energy and momentum conserving methods of arbitrary order for the numerical integration of equations of motion. *Numer. Math.* 25, 4 (1975), 323–346.

M. Levi. 2014. *Classical Mechanics with Calculus of Variations, and Optimal Control: An Intuitive Introduction*. American Mathematical Society. https://books.google.co.kr/books?id=uVSYswEACAAJ

Jing Li, Tiantian Liu, and Ladislav Kavan. 2018. Laplacian Damping for Projective Dynamics. In *VRIPHYS2018: 14th Workshop on Virtual Reality Interaction and Physical Simulation*.

Jing Li, Tiantian Liu, and Ladislav Kavan. 2019. Fast Simulation of Deformable Characters with Articulated Skeletons in Projective Dynamics. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '19)*. Association for Computing Machinery, New York, NY, USA, Article 1, 10 pages. https://doi.org/10.1145/3309486.3340249

Tiantian Liu, Adam W Bargteil, James F O'Brien, and Ladislav Kavan. 2013. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–7.

Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-newton methods for real-time simulation of hyperelastic materials. *ACM Transactions on Graphics (TOG)* 36, 3 (2017), 1–16.

Fabian Löschner, Andreas Longva, Stefan Jeske, Tassilo Kugelstadt, and Jan Bender. 2020. Higher-Order Time Integration for Deformable Solids. *Computer Graphics Forum* 39, 8 (2020), 157–169. https://doi.org/10.1111/cgf.14110 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14110

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*. 49–54.

Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-Based Elastic Materials. *ACM Trans. Graph.* 30, 4, Article 72 (July 2011), 8 pages. https://doi.org/10.1145/2010324.1964967

Dominik L. Michels, Vu Thai Luan, and Mayya Tokman. 2017. A Stiffly Accurate Integrator for Elastodynamic Problems. *ACM Trans. Graph.* 36, 4, Article 116 (July 2017), 14 pages. https://doi.org/10.1145/3072959.3073706

Dominik L. Michels, Gerrit A. Sobottka, and Andreas G. Weber. 2014. Exponential Integrators for Stiff Elastodynamic Problems. *ACM Trans. Graph.* 33, 1, Article 7 (Feb. 2014), 20 pages. https://doi.org/10.1145/2508462

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation* 18, 2 (2007), 109–118.

Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization*. Springer Science & Business Media.

Matthew Overby, George E. Brown, Jie Li, and Rahul Narain. 2017. ADMM ⊇ Projective Dynamics: Fast Simulation of Hyperelastic Models with Dynamic Constraints. *IEEE Transactions on Visualization and Computer Graphics* 23, 10 (Oct 2017), 2222–2234. https://doi.org/10.1109/TVCG.2017.2730875

J. Rojas, T. Liu, and L. Kavan. 2019. Average Vector Field Integration for St. Venant-Kirchhoff Deformable Models. *IEEE Transactions on Visualization and Computer Graphics* 25, 8 (2019), 2529–2539. https://doi.org/10.1109/TVCG.2018.2851233

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*. 1–50.

Juan C Simo, N Tarnow, and KK1187632 Wong. 1992. Exact energy-momentum conserving algorithms and symplectic schemes for nonlinear dynamics. *Computer methods in applied mechanics and engineering* 100, 1 (1992), 63–116.

Ari Stern and Mathieu Desbrun. 2006. Discrete Geometric Mechanics for Variational Time Integrators. In *ACM SIGGRAPH 2006 Courses* (Boston, Massachusetts) *(SIGGRAPH '06)*. Association for Computing Machinery, New York, NY, USA, 75–80. https://doi.org/10.1145/1185657.1185669

A. Stuart and A.R. Humphries. 1998. *Dynamical Systems and Numerical Analysis*. Number vol. 8 in Cambridge Monographs on Applie. Cambridge University Press. https://books.google.fr/books?id=ymoQA8s5pNIC

Jonathan Su, Rahul Sheth, and Ronald Fedkiw. 2012. Energy conservation for the simulation of deformable bodies. *IEEE Transactions on Visualization and Computer Graphics* 19, 2 (2012), 189–200.

Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. 205–214.

Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomerantes, and Markus H Gross. 2003. Optimized spatial hashing for collision detection of deformable objects.. In *Vmv*, Vol. 3. 47–54.

Table 1. Statistics of the experiments with our method: All deformable objects are represented in tetrahedral volumetric meshes except Trampoline and Cloth, which are represented in surface meshes. ([(1)] Initialization corresponds to lines 3 and 4 in **ALGORITHM** 1. [(2)] PD includes evaluating the PD-related terms such as $\mathbf{y}_n, \mathbf{A}_{\text{PD}}$, ∇g and $\mathbf{v}_{n+1}$. [(3)] LS means solving the linear system of Eq. (21).)

Object		# of vertices	# of elements	Material type	Avg. # of iterations	Timings [ms]			
						Per frame	Init.[(1)]	PD[(2)]	LS[(3)]
Bear	(Fig. 1)	3743	14025	Corotated	8.5	47.1	3.3	41.8	2.0
Cube	(Fig. 2)	792	2911	Corotated	5.1	6.3	0.9	5.1	0.3
Torus	(Fig. 3)	2096	7311	Corotated	7.0	20.6	1.7	17.8	1.1
T-pipe	(Fig. 4)	1686	5779	Corotated	5.4	19.2	2.6	15.7	0.9
Hippo	(Fig. 5)	2387	8406	Corotated	18.5	53.0	1.4	48.2	3.4
Ball	(Fig. 7)	889	1772	Corotated	3.4	2.8	0.6	2.0	0.2
Colliding bears	(Fig. 8)	1944	6805	Corotated	2.9	6.6	1.3	4.8	0.5
Strip	(Fig. 9)	738	1920	Corotated	2.7	2.4	0.5	1.8	0.1
Trampoline	(Fig. 10)	3600	10563	Mass spring	3.2	9.1	4.0	5.0	0.1
Cloth	(Fig. 12)	3600	10563	Mass spring	3.1	9.1	4.2	4.8	0.1

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. https://doi.org/10.1145/2816795.2818063

Marcel Weiler, Dan Koschier, and Jan Bender. 2016. Projective fluids. In *Proceedings of the 9th International Conference on Motion in Games*. 79–84.

Matthew West. 2004. *Variational integrators.* Ph.D. Dissertation. California Institute of Technology.

Hongyi Xu and Jernej Barbič. 2017. Example-Based Damping Design. *ACM Trans. Graph.* 36, 4, Article 53 (July 2017), 14 pages. https://doi.org/10.1145/3072959.3073631