Physical Cyclic Animations

SHIYANG JIA, University of California, San Diego
STEPHANIE WANG, University of California, San Diego
TZU-MAO LI, University of California, San Diego
ALBERT CHERN, University of California, San Diego

We address the problem of synthesizing physical animations that can loop seamlessly. We formulate a variational approach by deriving a physical law in a periodic time domain. The trajectory of the animation is represented as a parametric closed curve, and the physical law corresponds to minimizing the bending energy of the curve. Compared to traditional keyframe animation approaches, our formulation is constraint-free, which allows us to apply a standard Gauss–Newton solver. We further propose a fast projection method to efficiently generate an initial guess close to the desired animation. Our method can handle a variety of physical cyclic animations, including clothes, soft bodies with collisions, and N-body systems.

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

Additional Key Words and Phrases: keyframe animation, periodic motion, physical simulation

ACM Reference Format:
Shiyang Jia, Stephanie Wang, Tzu-Mao Li, and Albert Chern. 2023. Physical Cyclic Animations. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 2 (August 2023), 18 pages. https://doi.org/10.1145/3606938

1 INTRODUCTION

Cyclic animations are short animations that can be played in loops. They have existed since the early history of animations in the form of phenakistoscopes (Fig. 1). Looping animations continue to draw great interest in modern days for video games and movies.[1] They also play a central role in keyframe-controlled animations [Barbič et al. 2009], and designing characters' and animatronics' cyclic motion such as walking [Nunes et al. 2012; Mordatch et al. 2013; Starke et al. 2022].

Fig. 1. A phenakistoscope by Eadweard Muybridge [1892].

We aim to produce cyclic animations of physical systems. The animation should locally approximate the physical laws of motion, and globally loop back seamlessly (Fig. 2). In particular, we aim to reduce the number of user inputs, such as keyframes for guiding the animation [Barbič et al. 2009]. Our problem setting poses new challenges: while fewer keyframe fidelity constraints leave more room for chaotic physical systems to develop a larger variety of interesting motions, it produces worse control gradients evaluated by the common adjoint methods.

[1] There are several video tutorials on Youtube for generating looping animation (link, link, and link).

Authors' addresses: Shiyang Jia, University of California, San Diego; Stephanie Wang, University of California, San Diego; Tzu-Mao Li, University of California, San Diego; Albert Chern, University of California, San Diego.

2577-6193/2023/8-ART $15.00
https://doi.org/10.1145/3606938

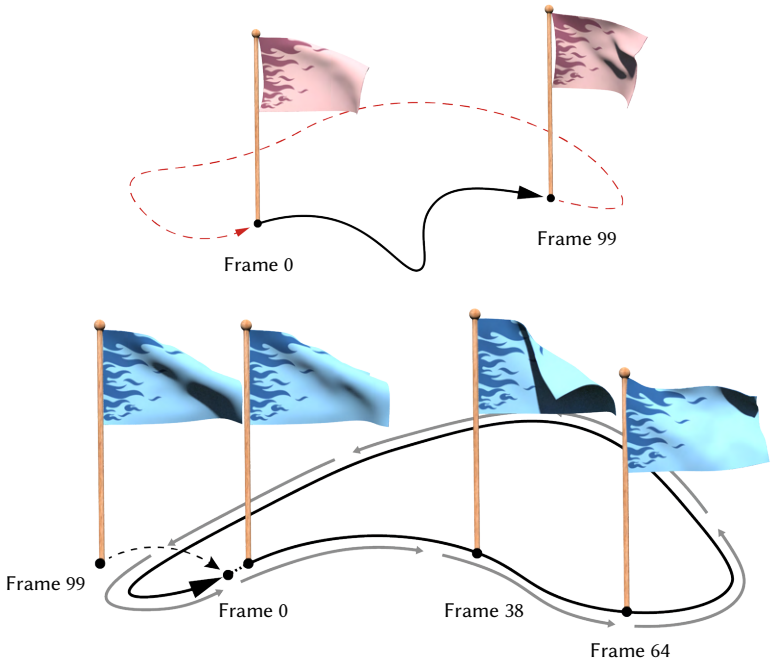

Fig. 2. Flag post. Given a physical system with an initial condition (top), our method synthesizes a physical animation (bottom) that can loop back seamlessly, such that the end frame would be the same as the initial frame. Here we show a cyclic animation of a flag under the wind.

We describe a variational approach to generate physical cyclic animations. The cyclic simulation is represented as a parameterized closed curve (loop) in the configuration space of the physical system (Fig. 3). The physical law assigns to each configuration point an associated acceleration of the trajectory. A trajectory passing through a point with a different acceleration is similar to a bent curve which gives rise to a cost of bending energy. Using this interpretation of a residual force minimization problem, we model our cyclic physical simulation as a closed curve of minimal total bending energy. Users can specify initial conditions (positions/velocities at the starting frame) either as soft or hard constraints of specific points of the loop.

Our closed curve parameterization guarantees a cyclic animation. Moreover, the resulting minimization problem is essentially unconstrained, in which case many optimization methods directly apply. We present a *fast projection method* for an initial guess followed by a *Gauss–Newton method* to obtain the optimal solution. Our optimization can often overcome the challenges caused by the non-linear constraints when applying the classical shooting method to a chaotic system (Fig. 3).

We show results on a variety of physical systems with a moderately large degrees of freedom , including thin shell models, hyperelastic finite element models with contact, and N-body simulation (see Section 5).

Comparing with the conventional key-frame animation control methods, we highlight:

- We build the cyclic constraint in the topology of trajectory, leaving an essentially *unconstrained* optimization problem.

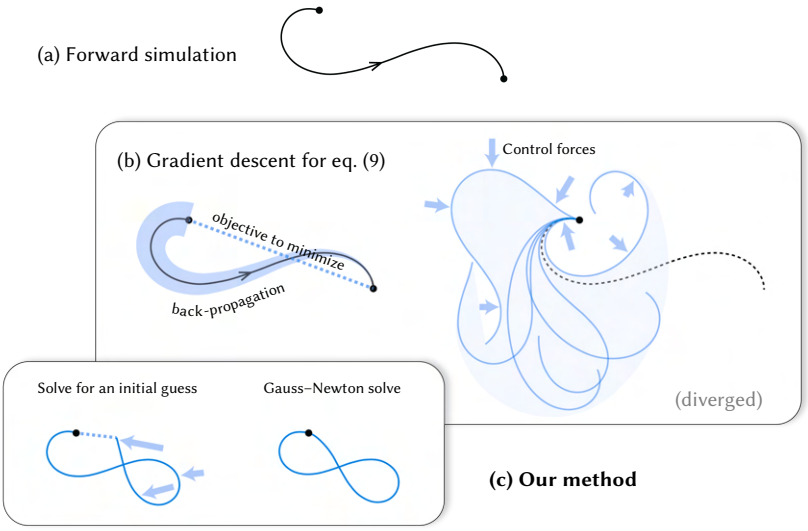

Fig. 3. For a chaotic dynamical system, a direct gradient descent method for the optimal control problem (9) often diverges due to the amplification in the sensitivity propagation (b). Our method (c) overcomes the problem by treating the time domain as a closed loop and avoiding using gradient descent for the initial guess.

- In the traditional optimal control approach, the gradient descent can be unstable or inefficient since the trajectory can be sensitive to control force in a general dynamical system. We replace the gradient descent procedure with a more efficient and robust fast projection method.
- The heuristic fast projection is backed by a Gauss–Newton solve that seeks a local minimum. The method works even for chaotic dynamical systems where traditional gradient-based optimal control fails.

2 RELATED WORK

Some current practice for generating physical looping animations is to interpolate between states of the end frame and the initial frame, or to resort to non-physical periodic motions such as a sine wave (Footnote 1). These approaches guarantee cyclic animations, at the cost of producing less physical motion. Some approaches solve partial differential equations with periodic boundary conditions using spectral methods [Castro et al. 2010].

Spacetime constraints. Our method belongs to the methods of spacetime constraints [Witkin and Kass 1988]. These methods specify the initial and end conditions (the keyframes) and find the most physically accurate trajectories that satisfy the conditions by applying control forces. In its general form, spacetime constraints involve non-convex optimization with non-linear constraints. Prior work has demonstrated the use of spacetime constraints to generate periodic motions [Barbič et al. 2009; Wampler and Popović 2009; Nunes et al. 2012; Schulz et al. 2014].

While spacetime constraints can be solved using techniques such as gradient descent and sequential quadratic programming, meeting the non-linear constraints is hard. The optimization becomes especially challenging when the keyframes are distant apart, leaving long time intervals that can support chaotic dynamics.

To address the challenges in optimization and provide user control, previous work employed several strategies. First, multiple keyframes are often used to further constrain the animation. Next, the non-linear optimization is often projected to a lower-dimensional subspace [Barbič et al. 2009] or converted into a linear system [Huang et al. 2011; Barbič et al. 2012; Hildebrandt et al. 2012; Li et al. 2014; Schulz et al. 2014]. Alternatively, some work applies derivative-free optimization [Wampler and Popović 2009; Nunes et al. 2012].

Our method targets degrees of freedom higher than usual optimal control approaches (e.g., we optimize for control forces on each vertex of a cloth), while focusing on the case where the only constraints are that the animation should loop back and be as physical as possible. Our method does not require additional keyframes (other than the initial condition), subspace projection, or linearization, that could sacrifice the physics. This is enabled by our constraint-free reformulation of the spacetime constraints problem and our Gauss–Newton optimization, whereas previous work still needs to meet the constraints even with the aforementioned remedies. On the other hand, we leave having more user controls as future work.

Shooting methods and optimal control. A common way of solving spacetime constraints in graphics is the shooting method [Betts 1998]: given a forward simulation, the constrained optimization is solved using the derivatives with respect to the control forces [Popović et al. 2003; Treuille et al. 2003; Wojtan et al. 2006]. The derivatives are often computed by the *adjoint method* [Johnson 2012], which shares similar computational efficiency to backpropagation or reverse-mode automatic differentiation. Some recent *differentiable physics* work also belong to this category [de Avila Belbute-Peres et al. 2018; Hu et al. 2020; Du et al. 2021], as well as work on trajectory optimization [Zimmermann et al. 2019], physical design [Zehnder et al. 2021], and fluid control [McNamara et al. 2004; Pan et al. 2013; Tang et al. 2021].

Cyclic animation can also be solved using the shooting method, by placing a constraint on the ending state. Prior work that demonstrates periodic motions using spacetime constraints often solves the optimization using the shooting method [Barbič et al. 2009; Schulz et al. 2014]. However, meeting the non-linear constraints through optimizing control forces is challenging. Another class of optimal control methods, direct collocation [Hargraves and Paris 1987; Posa et al. 2014], optimizes for both the control forces and trajectory, but they still need to handle non-linear constraints. We show that shooting methods, when used for generating cyclic animations, can violate the constraints and produce non-smooth "jumps" in the animation. Our method optimizes only the trajectory, making it trivial to satisfy the constraints, and can be used in conjunction with the shooting methods to improve their results.

Image/video-based methods. Some methods directly process videos to generate infinite loops [Schödl et al. 2000; Schödl and Essa 2002; Liao et al. 2013; He et al. 2017; Härkönen et al. 2022]. Among this class of methods, some use physically-inspired periodic motion to produce cyclic videos [Chuang et al. 2005; Davis et al. 2015]. We instead directly operate on the 3D physical states.

Data-driven motion synthesis. Some recent work applies data-driven techniques to synthesize a variety of (potentially periodic) motions from examples [Starke et al. 2022; Li et al. 2022]. We focus on physics-based approaches, and leave the combination between our work and data-driven methods as future work.

3 FORMULATION

Our goal is to formulate a physical law on a periodic time domain. We consider physical systems whose equations of motion take the form:

$$\mathbf{M}\ddot{\mathbf{q}}(t) = \mathbf{F}(\mathbf{q}(t)), \quad t \in \mathbb{R}, \quad \mathbf{q}(t) \in \mathbb{R}^m, \tag{1}$$

where $\mathbb{R}^m$ represents the physical configuration space,[2] $\mathbf{M} = (M_{\alpha\beta}) \in \mathbb{R}^{m\times m}$ the mass, and $\mathbf{F}$ a known force law. Our problem of producing a cyclic animation takes a set of desired initial values $\mathbf{q}(0) = \mathbf{r}_0, \dot{\mathbf{q}}(0) = \mathbf{v}_0$ and a time period $T > 0$ as inputs.

To model temporal periodicity, we let the domain of time be the periodic interval $[0, T)$, which we denote by the quotient space

$$\mathbb{S}^1_T \coloneqq \mathbb{R}/(T\mathbb{Z}) \quad \textit{i.e. } t \equiv t + Tn \text{ in } \mathbb{S}^1_T \text{ for all } t \in \mathbb{R}, n \in \mathbb{Z}. \tag{2}$$

We consider only those continuous paths $\mathbf{q}\colon \mathbb{S}^1_T \to \mathbb{R}^m$ in the configuration space parameterized by this cyclic time domain. Define the feasible set as the vector space of parameterized closed curves (Fig. 3)

$$\mathcal{V} = \left\{\mathbf{q}\colon \mathbb{S}^1_T \to \mathbb{R}^m\right\}. \tag{3}$$

For each closed curve $\mathbf{q} \in \mathcal{V}$ we define the energy

$$\mathcal{E}\colon \mathcal{V} \to \mathbb{R}, \quad \mathcal{E}[\mathbf{q}] \coloneqq \oint_{\mathbb{S}^1_T} W\left(\mathbf{q}(t), \ddot{\mathbf{q}}(t)\right)\, dt, \tag{4}$$

where the integrand W is a "bending energy" for the curve that measures its deviation from satisfying the physical system (1)

$$W\colon \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}, \quad W(\mathbf{q}, \mathbf{a}) \coloneqq \tfrac{1}{2}\left|\mathbf{M}\mathbf{a} - \mathbf{F}\left(\mathbf{q}\right)\right|^2_{\mathbf{M}^{-1}}, \tag{5}$$

where $|\mathbf{u}|^2_{\mathbf{M}^{-1}} \coloneqq \mathbf{u}^\intercal \mathbf{M}^{-1}\mathbf{u}$. Such a least-squares formulation follows Gauss' principle of least constraint [Gauß 1829; Hildebrandt et al. 2012].

3.1 Optimization problems

Our main optimization problem is to find $\mathbf{q} \in \mathcal{V}$ that solves

$$\min_{\mathbf{q}\in\mathcal{V}} \left(\mathcal{L}[\mathbf{q}] \coloneqq \mathcal{E}[\mathbf{q}] + \tfrac{1}{2\epsilon}|\mathbf{q}(0) - \mathbf{r}_0|^2_{\mathbf{M}} + \tfrac{1}{2\epsilon'}|\dot{\mathbf{q}}(0) - \mathbf{v}_0|^2_{\mathbf{M}}\right). \tag{6}$$

Here, ϵ, ϵ' are parameters that control how much we desire the fidelity for the initial values.

When $\epsilon, \epsilon' \to 0$, these "soft" initial conditions become hard linear constraints. In this limit, the feasible set reduces to the affine subspace

$$\mathcal{V}_{\mathbf{r}_0,\dot{\mathbf{v}}_0} \coloneqq \left\{\mathbf{q} \in \mathcal{V} \,\middle|\, \mathbf{q}(0) = \mathbf{r}_0, \dot{\mathbf{q}}(0) = \mathbf{v}_0\right\}, \tag{7}$$

over which we solve

$$\underset{\mathbf{q}\in\mathcal{V}_{\mathbf{r}_0,\mathbf{v}_0}}{\text{minimize}}\ \mathcal{E}[\mathbf{q}]. \tag{8}$$

[2]This could be a discretization of a continuum.

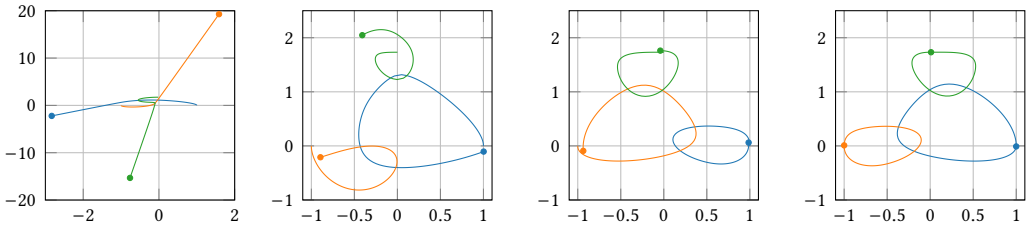

Fig. 4. Starting with a forward simulation of a 3-body system (a), we compare solving the optimal control problem (9) with non-linear constraints, using soft constraints (b), and projected gradient descent (c). The figure visualizes the trajectories. Projected gradient descent only converged for 9% of the initial conditions in our randomized trials: here we show the rare case where it converges. Our constraint-free formulation allows us to generate better seamless and physical animations (d) compared to the traditional methods based on solving non-linear constraints.

3.2 Relation to optimal control

With a substitution $\mathbf{u} = \mathbf{M}\ddot{\mathbf{q}} - \mathbf{F}(\mathbf{q})$, our optimization is equivalent to an optimal control problem:

$$\begin{aligned} &\underset{\mathbf{q},\mathbf{u}:\ [0,T]\to\mathbb{R}^m}{\text{minimize}} \int_0^T \tfrac{1}{2}|\mathbf{u}(t)|^2_{\mathbf{M}^{-1}}\, dt \\ &\text{subject to } \begin{cases} \mathbf{M}\ddot{\mathbf{q}}(t) = \mathbf{F}(\mathbf{q}(t)) + \mathbf{u}(t) \text{ for all } t \in (0,T) \\ \mathbf{q}(0) = \mathbf{q}(T) = \mathbf{r}_0 \\ \dot{\mathbf{q}}(0) = \dot{\mathbf{q}}(T) = \mathbf{v}_0. \end{cases} \end{aligned} \tag{9}$$

The optimizer $\mathbf{u}$ is the minimal *control force* added to the system (1) to guide $\mathbf{q}$ to meet the constraints at $t = T$.

Remark 1. *Although the optimal control problem* (9) *is equivalent to the unconstrained problem* (8) *by variable substitution, the non-linear physical constraint* $\mathbf{M}\ddot{\mathbf{q}} = \mathbf{F}(\mathbf{q}) + \mathbf{u}$ *makes satisfying the cyclic constraints much more difficult. Constraint violation is not tolerable for seamless cyclic animation. Constrained optimization, such as Lagrange multipliers or soft constraints with gradient descent, works well for small time intervals with monotonic motion. Such control becomes much more difficult for animation over a long time interval that supports complex physical motions. In practice, for the optimal control approach to work robustly, one adds multiple keyframes throughout the animation interval (rather than just an initial condition) [Barbič et al. 2009]; however, each additional keyframe constraint can sacrifice the physics. Our method builds cyclicity into the problem and removes the cyclic constraint from optimization.*

Our experiments (Fig. 4c) show that solving (9) with projected gradient descent or Lagrange multipliers usually leads to divergent iterations with increased loss and violated constraints [Platt and Barr 1988]. It is possible to relax the constraints $\mathbf{q}(0) = \mathbf{q}(T)$ and $\dot{\mathbf{q}}(0) = \dot{\mathbf{q}}(T)$ into soft constraints [Zehnder et al. 2021; Du et al. 2021], but this can lead to a sudden jump between the end and initial states (Fig. 4b). Optimizing soft constraints also requires extensive parameter tuning.

4 METHOD

We develop an algorithm to solve the variational problem (6). We discretize the periodic time domain and formulate the discrete counterpart of the optimization (6) (Section 4.1), which we solve iteratively using the Gauss–Newton method (Section 4.2) with an efficient initial guess (Section 4.3).

We use index notation to specify the components of the variable. An element $q_{j\alpha}^{(n)}$ is the scalar value of the α-th component of the physical configuration at time step j at optimization iteration n. We use boldface $\mathbf{q}_j = (q_{j0}, q_{j1}, \cdots, q_{j,m-1}) \in \mathbb{R}^m$ to denote the physical state at time step j.

4.1 Discretization

We discretize the temporal domain (2) uniformly into N nodes

$$t_j \coloneqq \tfrac{jT}{N}, \quad j \in \mathbb{Z}_N = \{0, 1, \cdots, N-1\}. \tag{10}$$

We use arithmetic *modulo* N on the index $j \in \mathbb{Z}_N$, and denote time step size by $h = T/N$. Under this time-discretization the collection of closed curves (3) becomes a finite-dimensional vector space

$$V = \left\{ \mathbf{q} = (q_{j\alpha}) \,\middle|\, j = 0, \cdots, N-1, \alpha = 0, \cdots, m-1 \right\} = \mathbb{R}^{mN}. \tag{11}$$

We define the operator that takes the second partial derivative with respect to time

$$\boldsymbol{\Delta} : \mathbb{R}^{mN} \to \mathbb{R}^{mN}, \quad (\boldsymbol{\Delta}\mathbf{q})_{j\alpha} \coloneqq \tfrac{1}{h^2}(q_{j+1,\alpha} - 2q_{j,\alpha} + q_{j-1,\alpha}). \tag{12}$$

Using this operator we define the discrete version of the total loss function (4) into a functional $E\colon V \to \mathbb{R}$. We have two options of the discrete loss function depending on the time integration scheme:

$$E_{\text{back}}[\mathbf{q}] \coloneqq \textstyle\sum_{j\in\mathbb{Z}_N} hW\left(\mathbf{q}_{j+1}, (\boldsymbol{\Delta}\mathbf{q})_j\right) \tag{13a}$$

$$= \textstyle\sum_{j\in\mathbb{Z}_N} \frac{h}{2} \left|\mathbf{M}(\boldsymbol{\Delta}\mathbf{q})_j - \mathbf{F}(\mathbf{q}_{j+1})\right|^2_{\mathbf{M}^{-1}} \tag{13b}$$

and

$$E_{\text{symp}}[\mathbf{q}] \coloneqq \textstyle\sum_{j\in\mathbb{Z}_N} hW\left(\mathbf{q}_j, (\boldsymbol{\Delta}\mathbf{q})_j\right) \tag{14a}$$

$$= \textstyle\sum_{j\in\mathbb{Z}_N} \frac{h}{2} \left|\mathbf{M}(\boldsymbol{\Delta}\mathbf{q})_j - \mathbf{F}(\mathbf{q}_j)\right|^2_{\mathbf{M}^{-1}} \tag{14b}$$

The difference between (13) and (14) corresponds to the two time integration schemes: backward Euler uses the force $\mathbf{F}(\mathbf{q}_{j+1})$ for the acceleration $(\boldsymbol{\Delta}\mathbf{q})_j$, where as symplectic Euler uses $\mathbf{F}(\mathbf{q}_j)$.[3] With the discrete energy ((13) or (14)), our main optimization problem (6) becomes

$$\min_{\mathbf{q}\in\mathcal{V}} \left(L[\mathbf{q}] \coloneqq E[\mathbf{q}] + \tfrac{1}{2h^3\epsilon_0}|\mathbf{q}_0 - \mathbf{r}_0|^2_{\mathbf{M}} + \tfrac{1}{2h^3\epsilon_1}|\mathbf{q}_1 - \mathbf{r}_1|^2_{\mathbf{M}} \right). \tag{15}$$

We add h^3 factor in the denominator to balance the units. The desired initial positions $\mathbf{r}_0$, $\mathbf{r}_1$ and the penalty parameters ϵ_0, ϵ_1 are prescribed by the user. In our discretization, the velocity $\dot{\mathbf{q}}(0)$ is discretized using finite difference $\frac{\mathbf{q}_1 - \mathbf{q}_0}{h}$. In the following, we stick with the symplectic energy (14). Converting to backward Euler is straightforward.

4.2 Optimization with Gauss–Newton method

We seek the minimum of (15) iteratively by the *Gauss–Newton method*, which is suitable for our loss function as a sum of squares

$$\textstyle\sum_{j\in\mathbb{Z}_N} \frac{h}{2} \left|\mathbf{M}(\boldsymbol{\Delta}\mathbf{q})_j - \mathbf{F}(\mathbf{q}_j)\right|^2_{\mathbf{M}^{-1}} + \tfrac{1}{2h^3\epsilon_0}|\mathbf{q}_0 - \mathbf{r}_0|^2_{\mathbf{M}} + \tfrac{1}{2h^3\epsilon_1}|\mathbf{q}_1 - \mathbf{r}_1|^2_{\mathbf{M}}. \tag{16}$$

[3]The choices for the arguments of W ensure that the absolute zero energy state $W(\mathbf{q}_{j+1}, (\boldsymbol{\Delta}\mathbf{q})_j) = 0$ (resp. $W(\mathbf{q}_j, (\boldsymbol{\Delta}\mathbf{q})_j) = 0$) is equivalent to the commonly employed backward (resp. symplectic) Euler time integration [Baraff and Witkin 1998]

$$\text{backward} \begin{cases} \mathbf{q}_{j+1} = \mathbf{q}_j + h\mathbf{v}_{j+1} \\ \mathbf{v}_{j+1} = \mathbf{v}_j + h\mathbf{M}^{-1}\mathbf{F}(\mathbf{q}_{j+1}); \end{cases} \quad \text{symplectic} \begin{cases} \mathbf{q}_{j+1} = \mathbf{q}_j + h\mathbf{v}_{j+1} \\ \mathbf{v}_{j+1} = \mathbf{v}_j + h\mathbf{M}^{-1}\mathbf{F}(\mathbf{q}_j). \end{cases}$$

Equation (16) can be written as a quadratic form:

$$L[\mathbf{q}] = \tfrac{1}{2}\mathbf{u}^\intercal \mathbf{B}\mathbf{u} + \tfrac{1}{2}\mathbf{g}^\intercal \mathbf{C}\mathbf{g}, \tag{17}$$

where $\mathbf{B} = h\begin{bmatrix} \mathbf{M}^{-1} & & \\ & \ddots & \\ & & \mathbf{M}^{-1} \end{bmatrix}$, $\mathbf{C} = \frac{1}{h^3}\begin{bmatrix} \mathbf{M}/\epsilon_0 & \\ & \mathbf{M}/\epsilon_1 \end{bmatrix}$, $\mathbf{u}\colon V \to V$ defined by

$$\begin{aligned} \mathbf{u}_j[\mathbf{q}] &\coloneqq \mathbf{M}(\Delta\mathbf{q})_j - \mathbf{F}(\mathbf{q}_j) \\ &= \tfrac{1}{h^2}(\mathbf{M}\mathbf{q}_{j-1} - 2\mathbf{M}\mathbf{q}_j + \mathbf{M}\mathbf{q}_{j+1}) - \mathbf{F}(\mathbf{q}_j), \quad j \in \mathbb{Z}_N, \end{aligned} \tag{18}$$

and $\mathbf{g}\colon V \to \mathbb{R}^{2m}$ by

$$\mathbf{g}[\mathbf{q}] = \begin{bmatrix} \mathbf{q}_0 - \mathbf{r}_0 \\ \mathbf{q}_1 - \mathbf{r}_1 \end{bmatrix}. \tag{19}$$

4.2.1 Gauss–Newton method. A Gauss–Newton method is a quasi-Newton method that uses a specific approximation of the Hessian of the loss function. The gradient of our loss function is given by

$$\tfrac{\partial}{\partial \mathbf{q}} L = \mathbf{J}^\intercal \mathbf{B}\mathbf{u} + \mathbf{K}^\intercal \mathbf{C}\mathbf{g}, \tag{20}$$

where $\mathbf{J} = \frac{\partial \mathbf{u}}{\partial \mathbf{q}}$ and $\mathbf{K} = \frac{\partial \mathbf{g}}{\partial \mathbf{q}}$ are the Jacobians of (18) and (19) respectively (see Section 4.2.2 for the explicit formulas). In a Gauss–Newton method, one takes the semi-positive definite approximation for the Hessian

$$\operatorname{Hess} L \approx \widehat{\operatorname{Hess} L} \coloneqq \mathbf{J}^\intercal \mathbf{B}\mathbf{J} + \mathbf{K}^\intercal \mathbf{C}\mathbf{K}. \tag{21}$$

This yields the following quasi-Newton algorithm.

Algorithm 1: Main algorithm

input: Initial guess $\mathbf{q}^{(0)}$ from Algorithm 2

1 **while** $|\mathbf{q}^{(n+1)} - \mathbf{q}^{(n)}| < \epsilon$ **do**

2 Evaluate $\mathbf{u}$, $\mathbf{g}$, $\mathbf{J} = \frac{\partial \mathbf{u}}{\partial \mathbf{q}}$ and $\mathbf{K} = \frac{\partial \mathbf{g}}{\partial \mathbf{q}}$ at $\mathbf{q}^{(n)}$ ▷ (18), (19)

3 $\frac{\partial L}{\partial \mathbf{q}} \leftarrow \mathbf{J}^\intercal \mathbf{B}\mathbf{u} + \mathbf{K}^\intercal \mathbf{C}\mathbf{g}$ ▷ (20)

4 $\widehat{\operatorname{Hess} L} \leftarrow \mathbf{J}^\intercal \mathbf{B}\mathbf{J} + \mathbf{K}^\intercal \mathbf{C}\mathbf{K}$ ▷ (21)

5 $\delta\mathbf{q} \leftarrow \textsc{Solve}\left(\widehat{\operatorname{Hess} L}\,\delta\mathbf{q} = -\frac{\partial L}{\partial \mathbf{q}}\right)$ ▷ quasi-Newton step

6 $\mathbf{q}^{(n+1)} \leftarrow \mathbf{q}^{(n)} + s\delta\mathbf{q}$ ▷ $s \in (0,1)$ given by *line search.*

7 **end**

output: $\mathbf{q}^{(n)}$ in the last iteration

One can use standard methods to solve the linear system in Step 5. Our current implementation uses a Cholesky decomposition.

4.2.2 Jacobians of the residual forces. The Jacobian $\mathbf{J} = \frac{\partial \mathbf{u}}{\partial \mathbf{q}}$ of the residual force (18) is given by

$$(\mathbf{J}\mathring{\mathbf{q}})_j = \tfrac{1}{h^2}(\mathbf{M}\mathring{\mathbf{q}}_{j-1} - 2\mathbf{M}\mathring{\mathbf{q}}_j + \mathbf{M}\mathring{\mathbf{q}}_{j+1}) - \mathbf{H}_j\mathring{\mathbf{q}}_j, \quad j \in \mathbb{Z}_N \tag{22}$$

for all variation $\mathring{\mathbf{q}} \in V$, where

$$\mathbf{H}_j \coloneqq -\nabla\mathbf{F}(\mathbf{q}_j) \in \mathbb{R}^{m\times m}, \quad j \in \mathbb{Z}_N, \tag{23}$$

is the negative gradient of the force law evaluated at configuration $\mathbf{q}_j \in \mathbb{R}^m$. If $\mathbf{F}$ is a conservative force $\mathbf{F} = -\nabla U$ arising from a potential energy $U\colon \mathbb{R}^m \to \mathbb{R}$, then $\mathbf{H}_j = \operatorname{Hess} U|_{\mathbf{q}_j}$. The Jacobian $\mathbf{K} = \frac{\partial \mathbf{g}}{\partial \mathbf{q}}$ of (19) is given by $\mathbf{K}\mathring{\mathbf{q}} = \begin{bmatrix} \mathring{\mathbf{q}}_0 \\ \mathring{\mathbf{q}}_1 \end{bmatrix}$.

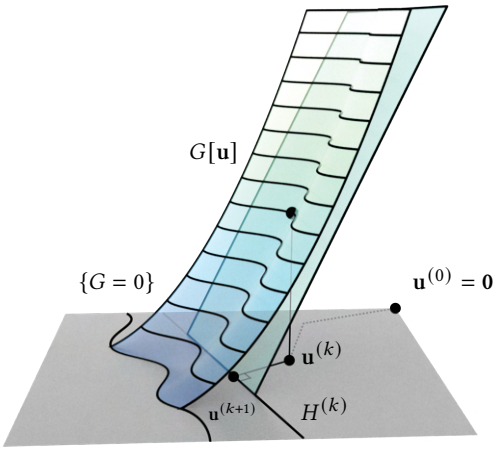

Fig. 5. Fast projection finds a point on the constraint manifold $\{G = 0\}$ that is close to the initial guess $\mathbf{u}^{(0)} = \mathbf{0}$. The method iteratively linearizes the constraint function and project the iterand $\mathbf{u}^{(k)}$ to the zeros $H^{(k)}$.

4.2.3 Hard position constraints. When $\epsilon_0, \epsilon_1 \to 0$ or when there are boundary conditions in the physical system, the values of $q_{j\alpha}$ are prescribed and fixed during optimization for a subset of $(j, \alpha) \in \mathbb{Z}_N \times \mathbb{Z}_m$. In the presence of such hard position constraints, we simply remove the prescribed $q_{j\alpha}$'s from the set of unknown variables. This procedure reduces the size of the matrix $\widehat{\mathrm{Hess}\, L}$, and, in the case of $\epsilon_0, \epsilon_1 = 0$, removes the stiff term $\mathbf{K}^\intercal \mathbf{C}\mathbf{K}$.

4.3 Initial guess with fast projection

For the initial guess for Algorithm 1, we construct a heuristic $\mathbf{q}^{(0)}$ that is close to being cyclic and physical. To achieve this, we solve an approximate solution to the control problem (9). Define a functional $\mathbf{q}$ that maps a control force $\mathbf{u}$ to the solution to the initial value problem (forward simulation)

$$\mathbf{q}[\mathbf{u}] \coloneqq \text{Solution to } \begin{cases} \mathbf{M}(\Delta \mathbf{q})_j = \mathbf{F}(\mathbf{q}_j) + \mathbf{u}_j, \ \ j = 1, \dots, n \\ \mathbf{q}_0 = \mathbf{r}_0, \quad \mathbf{q}_1 = \mathbf{r}_1. \end{cases} \tag{24}$$

With this substitution, we rewrite (9) as

$$\underset{\mathbf{u}}{\text{minimize}} \ \tfrac{1}{2} \sum_{j=1}^{n} |\mathbf{u}_j|^2 \quad \text{subject to } G[\mathbf{u}] = 0, \tag{25}$$

where

$$G[\mathbf{u}] = \tfrac{1}{2} \left| \mathbf{q}[\mathbf{u}]_n - \mathbf{r}_0 \right|^2 + \tfrac{1}{2} \left| \mathbf{q}[\mathbf{u}]_{n+1} - \mathbf{r}_1 \right|^2 . \tag{26}$$

We apply the *fast projection method* [Goldenthal et al. 2007; Dinev et al. 2018] to find a point on the constraint manifold $\{G = 0\}$ close to the origin as an approximate solution to (25). As illustrated in Fig. 5, the method starts with $\mathbf{u}^{(0)} = \mathbf{0}$ and iteratively updates

$$\mathbf{u}^{(k+1)} = \mathscr{P}_{H^{(k)}}(\mathbf{u}^{(k)}) \tag{27}$$

with the orthogonal projection $\mathscr{P}_{H^{(k)}}$ onto the zero-hyperplane $H^{(k)}$ of the linearized constraint functional at $\mathbf{u}^{(k)}$:

$$H^{(k)} = \left\{ \mathbf{u} \,\middle|\, G[\mathbf{u}^{(k)}] + \nabla G|_{\mathbf{u}^{(k)}} (\mathbf{u} - \mathbf{u}^{(k)}) = 0 \right\}. \tag{28}$$

The stepping $\delta\mathbf{u} = \mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}$ is perpendicular to the hyperplane $H^{(k)}$ and therefore

$$\delta\mathbf{u} = -\lambda \nabla G|_{\mathbf{u}^{(k)}}^{\mathsf{T}}, \tag{29}$$

where $\lambda \in \mathbb{R}$ is a scalar multiplier.

The projection step (27) thus amounts to solving

$$\begin{bmatrix} I & \nabla G|_{\mathbf{u}^{(k)}}^{\mathsf{T}} \\ \nabla G|_{\mathbf{u}^{(k)}} & 0 \end{bmatrix} \begin{bmatrix} \delta\mathbf{u} \\ \lambda \end{bmatrix} = - \begin{bmatrix} \mathbf{0} \\ G[\mathbf{u}^{(k)}] \end{bmatrix}. \tag{30}$$

We can solve the linear system efficiently: observe (by Gauss elimination) that $\lambda = {}^{G}/_{|\nabla G|^2}$, and therefore we can evaluate $\delta\mathbf{u} = -\lambda \nabla G = -\frac{G}{|\nabla G|^2} \nabla G$ explicitly. We evaluate ∇G by the standard adjoint method or reverse-mode automatic differentiation [Johnson 2012]. The overall process of finding an initial guess only adds little computational overhead. We summarize our algorithm for finding an initial guess below.

Algorithm 2: Fast projection for an initial guess for Algorithm 1

$\mathbf{u}^{(0)} \leftarrow \mathbf{0}$
for $k = 0, 1, 2, \ldots$ **do**
  Evaluate G, ∇G at $\mathbf{u}^{(k)}$ ▷ forward sim. and back propagation
  $\delta\mathbf{u} \leftarrow ({}^{G}/_{|\nabla G|^2}) \nabla G$ ▷ solves (30)
  $\mathbf{u}^{(k+1)} \leftarrow \mathbf{u}^{(k)} + \delta\mathbf{u}$
end
output: $\mathbf{q}^{(0)} \leftarrow$ Solve (24). ▷ forward simulation

Remark 2. *The fast projection* (30) *is an approximation of the projected gradient descent method by assuming both the control force and the Lagrange multiplier are zero. This relaxation helps stabilizing the optimization when the system is less stiff. As a cautionary note, the iteration could diverge when the required control force is large and the approximation is no longer adequate.*

5 RESULTS

We demonstrate that our algorithm generates physical cyclic animations for various physical systems, including N-body systems, cloth simulation under externally driven forces or periodic boundary conditions, and soft body simulation with contact. We then evaluate the effectiveness of fast projection as the initial guess. We also show one example where fast projection fails, but Gauss–Newton solver can still generate physical cyclic animation by taking forward simulation as an initial guess.

For cloth and soft body simulation, we adopt the finite element implementation from the C-IPC codebase [Li et al. 2021]. We also implement penalty-based collision with external objects. We use CHOLMOD [Chen et al. 2008] for its linear solvers and OpenMP for parallel computing. All related experiments are run on a 2.3 GHz 8-Core Intel Core i7-11800H CPU with 32 GB memory. Apart from that, we implement the N-body example in Python and execute it on an 8-Core Apple M1 and 8 GB memory. We use symplectic integration in N-Body and implicit integration in cloth and soft body simulation for stability. We report the experiment details and optimization run time in Table 1. A time step of 10 ms is used in all of the experiments.

(a) Physical (b) Cyclic (ours)

Fig. 6. N-body. A 5-body system of equal mass interacting under mutual gravity. Physically, the system evolves in a chaotic manner (left). Our proposed method allows for synthesizing a realistic cyclic animation (right). The trajectories of forward simulation experience a jump (visualized in dashed lines), whereas our method loops back to the initial orbit.

Table 1. Performance and statistics.

Name	Vertices	Elements	Frames	Iteration		Avg. time (s)	
				FP	GN	FP	GN
Flag post (Fig. 2)	174	300	100	24	246	5.6	11.2
N-body (Fig. 6)	5	-	200	-	80	-	0.16
Ribbons (Fig. 7)	243	243	100	96	156	2.1	2.7
Bouncing doughnut (Fig. 8)	281	990	60	19	86	2.3	33.8
Wobbly bar (Fig. 10)	133	390	250	19	82	6.5	11.1
Swinging cloth (Fig. 11)	121	200	200	97	17	3.7	8.6
Twisting bar (Fig. 12)	133	390	100	-	59	-	5.4

5.1 N-body system

A number (N) of celestial bodies interacting under Newton's gravity often results in chaotic dynamics (for $N \geq 3$) [Barrow-Green 1997]. Seeking time-periodic solutions in an N-body system has been a computationally challenging problem in astronomy [Li and Liao 2019]. Despite the chaotic nature of the physical system, our algorithm can generate close-to-physical time-periodic solutions with relatively small computational cost.

As shown in Fig. 6, a physical simulation of an N-body problem by solving an initial value problem (left) is unlikely to be close to cyclic. Our method simulates the system while guaranteeing time periodicity. The trajectories of the 5 bodies form 5 interlinking rings.

5.2 Cloth

We demonstrate cyclic cloth simulations using a thin shell [Grinspun et al. 2003] with bending energy and Neo-Hookean membrane energy using the physical material parameters of 100% cotton.

5.2.1 Flag Post. Fig. 2 shows an animation of a rectangular flag with two fixed corners. In addition to the elastic forces, we add a wind force proportional to the dot product between the surface normal and a constant wind vector.[4] To get natural starting positions, we run forward simulation and use the configurations from the 60^{th} and 61^{st} frame to set $\mathbf{r}_0$ and $\mathbf{r}_1$ respectively. Forward simulations with this setup yield a physics-based animation of a waving flag (Fig. 2 left), but the

[4]A constant wind vector is a simplified model compared to more accurate models involving turbulent flow surrounding the flag.

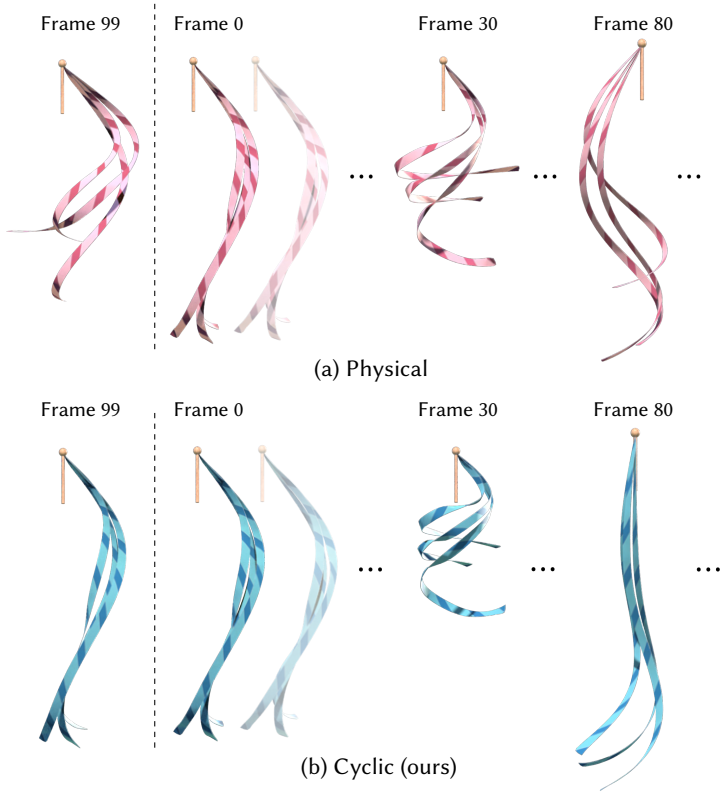

Fig. 7. Ribbons. A rotating point handle is driving three ribbons. The handle orbits in a circular motion and circles 3 times in 100 frames. The forward simulation produces an end state (Frame 99) that significantly differs from the initial state (Frame 0).

animation generally does not repeat. Our initial guess (Section 4.3) finds a plausible solution, but still exhibits an unnatural jump at the end. Optimizing using our method from the initial guess finds a physically realistic cyclic animation that shows no visual artifact.

5.2.2 Ribbons. Fig. 7 shows an animation of narrow strips of thin shell approximating anisotropic elastic rods [Bergou et al. 2008]. We attach three ribbons to a point handle which orbits in cyclic motion. Even with such a time-periodic external influence, the forward simulation fails to loop back. Our method is able to produce a natural motion of ribbons that also loops back seamlessly.

5.3 Soft body with collision

We apply our method to soft body simulation with collision. We adopt a penalty-based contact model with the floor and sphere [Macklin et al. 2020].

5.3.1 Bouncing Doughnut. As shown in Fig. 8, we setup a doughnut-shaped soft body with a downward velocity at frame 0. In the forward simulation, the body bounces off to a different course after its collision with the ground and never loops back. With our initial guess, the body manages to loop back to its initial position. However, this approximated solution still suffers from a large virtual acceleration around the last frames, which yields artifacts in video playback. This is also visible as a discontinuity in the velocity curve (Fig. 9). Our optimal solution produces a natural

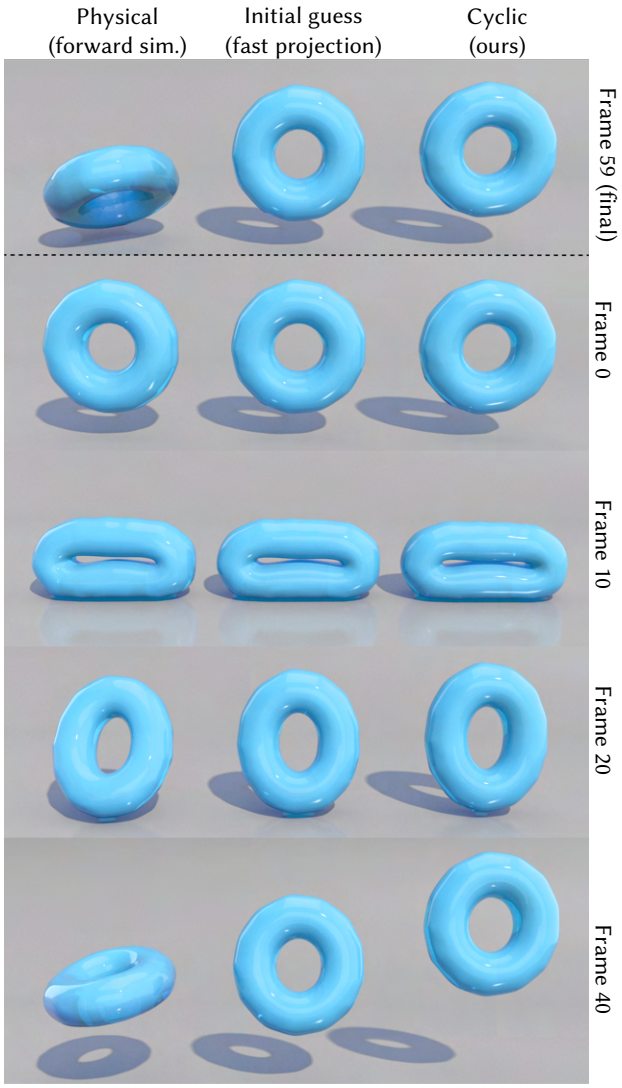

Fig. 8. Bouncing Doughnut. Our method can be applied to soft body simulation with contact. Our initial guess generates an animation that almost loops back, at the cost of losing some momentum due to a large control force, making it not bounce as high. Our optimization produces a seamless loop, while preserving the momentum.

cyclic animation. The soft body bounces up to the correct height (frame 40), from which it falls back to the initial position, matching the velocity of the initial frame. Fig. 9 shows the vertical component of the center of mass plotted against time. The physical forward simulation loses the vertical momentum as it is transferred to an angular one upon collision. The fast projection is not able to restore this loss of vertical momentum. Our method restores the vertical momentum and successfully loops back in both position and velocity.

5.3.2 Wobbly bar. In Fig. 10, we show a scripted rigid ball moving through the scene in constant velocity and colliding with an elastic bar. The elastic bar is initialized from the rest shape. This is a

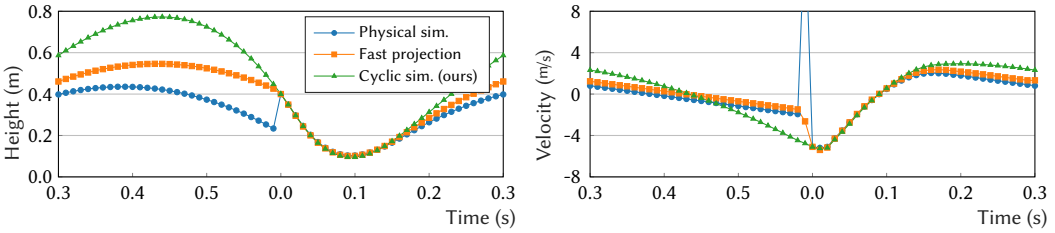

Fig. 9. The vertical position and velocity of the center of mass of the soft body in Fig. 8 plotted against time. The forward simulation exhibits a discontinuous jump at time 0. Our initial guess (fast projection) reduces the jump, but loses the vertical momentum. After the Gauss–Newton optimization, our method restores the momentum, while maintaining a seamless physical cyclic animation.

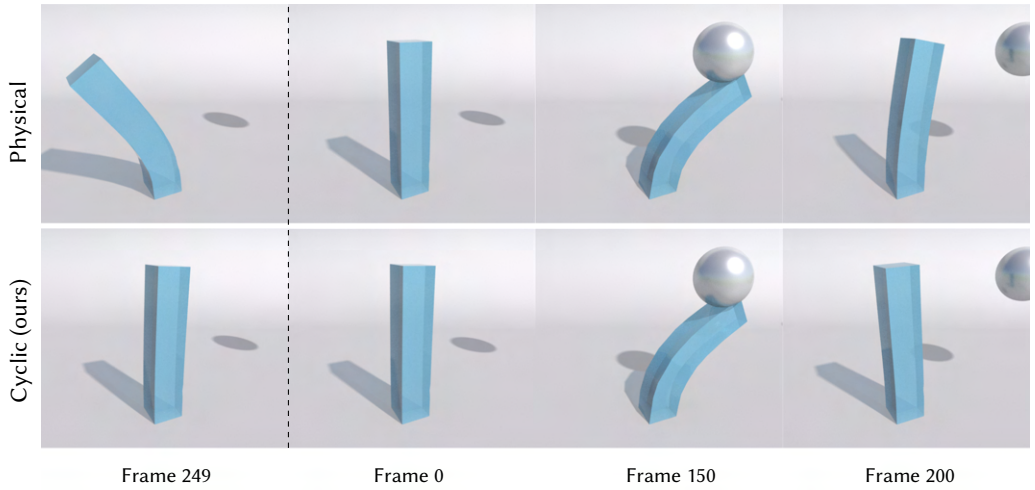

Fig. 10. Wobbly bar. Under the influence of a colliding sphere with an elastic bar, the physical simulation does not loop back to its initial state (top row). Our method generates a natural cyclic animation in this challenging scenario (bottom row).

challenging scenario with large deformation and a long time interval. The forward simulation fails to loop back, while our method generates natural and seamless animation.

5.4 Evaluation

5.4.1 Swinging cloth. Here we compare fast projection with gradient descent in solving keyframe control problems and evaluate their role as initial guesses. We found that directly solving (9) using projected gradient descent usually diverges. Instead, previous spacetime constraints methods [Barbič et al. 2009; Wojtan et al. 2006] often relax the cyclic constraints in (9) as soft constraints and solve it using gradient-based optimization. The experiment setting and convergence plot are shown in Fig. 11. The animation consists of one piece of cloth swinging down from a flat stationary state under gravity. The input forward simulation contains one swing period. The cloth can not return to the initial height because of the damping effect introduced by implicit Euler. Both optimization schemes start from zero control force. We can see from Fig. 11 (b) and Fig. 11 (c) that fast projection achieves lower constraint loss and significantly smaller control force. Visually, the animation looks

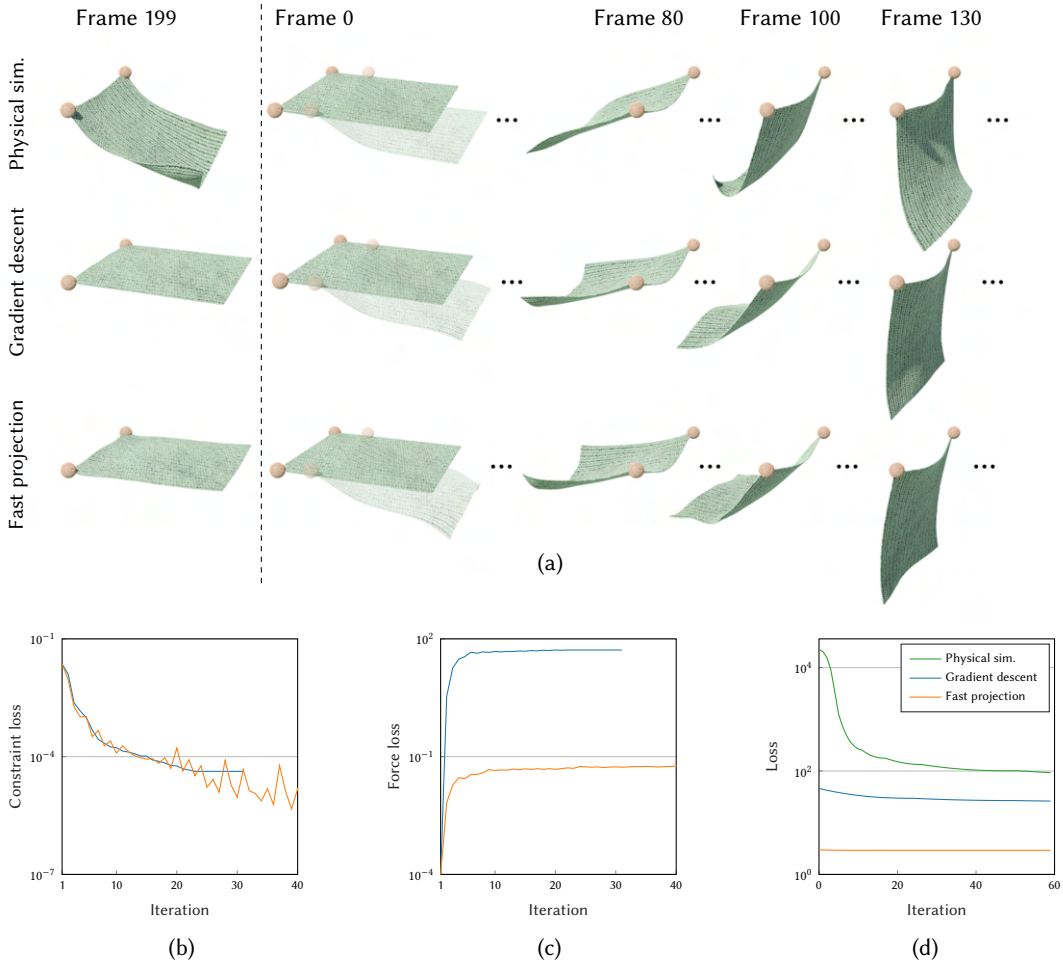

Fig. 11. Swinging cloth. A piece of cloth swings back and forth from a horizontal initial state. The dynamics under physical simulation show that the system does not naturally loop back. By optimizing using gradient descent and fast projection, the dynamics can be forced to loop back, as shown in (a). The fast projection method shows better convergence in both terms of the loss functions, as shown in (b) and (c). In (d) we show our Gauss–Newton optimization with the three initial guesses: fast projection converges to the lowest loss in this case.

smooth and more natural than the gradient descent. From the good initial guess produced by fast projection, our Gauss–Newton solver converges to better local minimum Fig. 11 (d). Interestingly, if we ignore the initial state constraint in our formula by setting $\epsilon = +\infty$, the final solution is a nice energy-preserving animation.

5.4.2 Twisting bar. As mentioned in Remark 2, fast projection tends to fail when the required control force is large. Here we show an elastic bar releasing from a twisted initial state (Fig. 12). In this challenging large deformation scenario, fast projection quickly diverges, and gradient descent also fails to make progress. However, our Gauss–Newton step can still generate seamless cyclic animation using forward simulation as the initial guess.

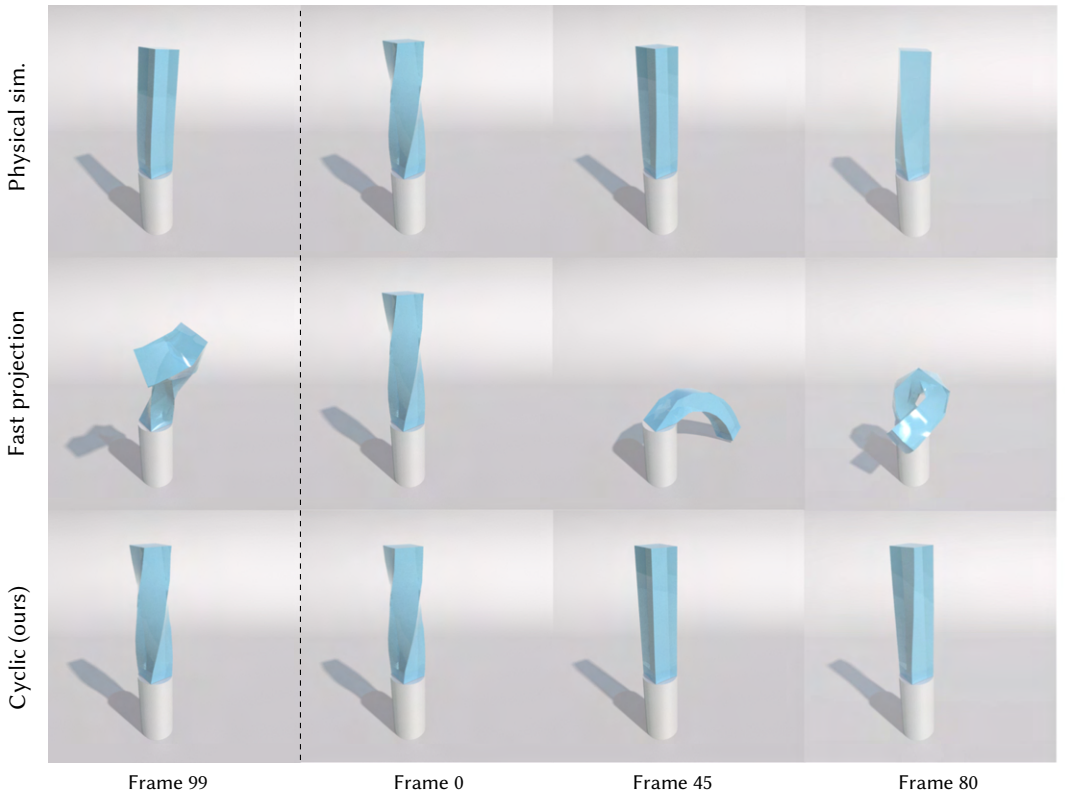

Fig. 12. Twisting bar. An elastic bar is initialized in a twisted state. The forward simulation gradually returns to the rest state (top row). In this case, the fast projection method fails to enforce the cyclic constraint, and as such is not a good choice for the initial guess. By using the physical simulation as the initial guess, the Gauss–Newton solver generates a cyclic animation (bottom row).

6 LIMITATIONS AND FUTURE WORK

Self-collision. Although we integrated our method with a penalty-based contact model to include collision objects in simulations, we have not attempted to address self-collisions in our method.

Damping. Both our continuous and discrete physical systems are always Lyapunov-stable due to their cyclic nature. However, since we do not put constraints on the kinetic energy of the system, it is possible that the system arrives at a more damped solution compared to the initial, non-cyclic trajectory. A different loss function that enables more control can potentially resolve the problem.

Scalability. Our Gauss–Newton solver requires us to invert the Hessian matrix (21) with size $(m \times N)^2$. While our current implementation uses a direct Cholesky decomposition for inversion, it is likely that a more efficient solver can be designed to solve a larger Hessian matrix by exploiting its structure.

7 CONCLUSION

We propose a constraint-free formulation for physical cyclic animations, allowing us to synthesize seamless looping animation without solving non-linear constraints. Our formulation is general and can handle a variety of physical systems.

ACKNOWLEDGMENTS

We thank the reviewers for their comments and suggestions. We thank Mohammad Sina Nabizadeh for the discussions and help on the manuscript. We thank Alec Jacobson and Tiantian Liu for the helpful discussions. We thank Xuanda Yang for providing additional computation resources, he and Yash Belhe also help with proofreading. This research was funded by NSF grant #2238839, an Adobe gift, and additional support from SideFX software.

REFERENCES

David Baraff and Andrew Witkin. 1998. Large steps in cloth simulation. In *SIGGRAPH*. 43–54.

Jernej Barbič, Marco da Silva, and Jovan Popović. 2009. Deformable Object Animation Using Reduced Optimal Control. *ACM Trans. Graph. (Proc. SIGGRAPH)* 28, 3, Article 53 (2009), 9 pages.

Jernej Barbič, Funshing Sin, and Eitan Grinspun. 2012. Interactive Editing of Deformable Simulations. *ACM Trans. Graph.* 31, 4, Article 70 (2012), 8 pages.

June Barrow-Green. 1997. *Poincaré and the three body problem.* Number 11. American Mathematical Society.

Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete elastic rods. *ACM Trans. Graph. (Proc. SIGGRAPH)*, Article 63 (2008), 12 pages.

John T Betts. 1998. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics* 21, 2 (1998), 193–207.

G González Castro, Michael Athanasopoulos, and Hassan Ugail. 2010. Cyclic animation using partial differential equations. *The Visual Computer* 26, 5 (2010), 325–338.

Yanqing Chen, Tim Davis, William Hager, and Siva Rajamanickam. 2008. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Trans. Math. Softw.* 35 (01 2008).

Yung-Yu Chuang, Dan B Goldman, Ke Colin Zheng, Brian Curless, David H Salesin, and Richard Szeliski. 2005. Animating pictures with stochastic motion textures. *ACM Trans. Graph. (Proc. SIGGRAPH)* 24, 3 (2005), 853–860.

Abe Davis, Justin G. Chen, and Frédo Durand. 2015. Image-Space Modal Bases for Plausible Manipulation of Objects in Video. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 34, 6, Article 239 (2015), 7 pages.

Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J. Zico Kolter. 2018. End-to-End Differentiable Physics for Learning and Control. In *Advances in Neural Information Processing Systems*, Vol. 31. 7178–7189.

Dimitar Dinev, Tiantian Liu, Jing Li, Bernhard Thomaszewski, and Ladislav Kavan. 2018. FEPR: Fast Energy Projection for Real-Time Simulation of Deformable Objects. *ACM Trans. Graph. (Proc. SIGGRAPH)* 37, 4, Article 79 (2018), 12 pages.

Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. 2021. DiffPD: Differentiable Projective Dynamics. *ACM Trans. Graph.* 41, 2, Article 13 (2021), 21 pages.

Carl Friedrich Gauß. 1829. Über ein neues allgemeines Grundgesetz der Mechanik. (1829).

Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. 2007. Efficient Simulation of Inextensible Cloth. *ACM Trans. Graph. (Proc. SIGGRAPH)* 26, 3, Article 49 (2007), 8 pages.

Eitan Grinspun, Anil N Hirani, Mathieu Desbrun, and Peter Schröder. 2003. Discrete shells. In *Symposium on Computer Animation*. 62–67.

Charles R Hargraves and Stephen W Paris. 1987. Direct trajectory optimization using nonlinear programming and collocation. *Journal of guidance, control, and dynamics* 10, 4 (1987), 338–342.

Erik Härkönen, Miika Aittala, Tuomas Kynkäänniemi, Samuli Laine, Timo Aila, and Jaakko Lehtinen. 2022. Disentangling random and cyclic effects in time-lapse sequences. *ACM Trans. Graph. (Proc. SIGGRAPH)* 41, 4, Article 72 (2022), 13 pages.

Mingming He, Jing Liao, Pedro V. Sander, and Hugues Hoppe. 2017. Gigapixel Panorama Video Loops. *ACM Trans. Graph.* 37, 1, Article 3 (2017), 15 pages.

Klaus Hildebrandt, Christian Schulz, Christoph von Tycowicz, and Konrad Polthier. 2012. Interactive Spacetime Control of Deformable Objects. *ACM Trans. Graph.* 31, 4, Article 71 (2012), 8 pages.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. 2020. DiffTaichi: Differentiable Programming for Physical Simulation. In *International Conference on Learning Representations*.

Jin Huang, Y. Tong, Kun Zhou, Hujun Bao, and Mathieu Desbrun. 2011. Interactive Shape Interpolation through Controllable Dynamic Deformation. *IEEE Transactions on Visualization and Computer Graphics* 17 (2011), 983–992.

Steven G Johnson. 2012. Notes on adjoint methods for 18.335. *Introduction to Numerical Methods* (2012).

Minchen Li, Danny M. Kaufman, and Chenfanfu Jiang. 2021. Codimensional Incremental Potential Contact. *ACM Trans. Graph. (Proc. SIGGRAPH)* 40, 4, Article 170 (2021), 24 pages.

Peizhuo Li, Kfir Aberman, Zihan Zhang, Rana Hanocka, and Olga Sorkine-Hornung. 2022. GANimator: Neural Motion Synthesis from a Single Sequence. *ACM Trans. Graph. (Proc. SIGGRAPH)* 41, 4, Article 138 (2022), 12 pages.

Siwang Li, Jin Huang, Fernando de Goes, Xiaogang Jin, Hujun Bao, and Mathieu Desbrun. 2014. Space-Time Editing of Elastic Motion through Material Optimization and Reduction. *ACM Trans. Graph. (Proc. SIGGRAPH)* 33, 4, Article 108 (2014), 10 pages.

Xiaoming Li and Shijun Liao. 2019. Collisionless periodic orbits in the free-fall three-body problem. *New Astronomy* 70 (2019), 22–26.

Zicheng Liao, Neel Joshi, and Hugues Hoppe. 2013. Automated video looping with progressive dynamism. *ACM Trans. Graph. (Proc. SIGGRAPH)* 32, 4, Article 77 (2013), 10 pages.

Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Tae-Yong Kim. 2020. Primal/dual descent methods for dynamics. *Comput. Graph. Forum (Proc. SCA)* 39, 8 (2020), 89–100.

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid control using the adjoint method. *ACM Trans. Graph. (Proc. SIGGRAPH)* 23, 3 (2004), 449–456.

Igor Mordatch, Jack M Wang, Emanuel Todorov, and Vladlen Koltun. 2013. Animating human lower limbs using contact-invariant optimization. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 32, 6, Article 203 (2013), 8 pages.

Eadweard Muybridge. 1892. The zoopraxiscope 35: A couple waltzing.

Rubens F Nunes, Joaquim B Cavalcante-Neto, Creto A Vidal, Paul G Kry, and Victor B Zordan. 2012. Using natural vibrations to guide control for locomotion. In *Symposium on Interactive 3D Graphics and Games*. 87–94.

Zherong Pan, Jin Huang, Yiying Tong, Changxi Zheng, and Hujun Bao. 2013. Interactive localized liquid motion editing. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 32, 6, Article 184 (2013), 10 pages.

John C Platt and Alan H Barr. 1988. Constraints methods for flexible models. *Comput. Graph. (Proc. SIGGRAPH)* 22, 4 (1988), 279–288.

Jovan Popović, Steven M Seitz, and Michael Erdmann. 2003. Motion sketching for control of rigid-body simulations. *ACM Trans. Graph.* 22, 4 (2003), 1034–1054.

Michael Posa, Cecilia Cantu, and Russ Tedrake. 2014. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research* 33, 1 (2014), 69–81.

Arno Schödl and Irfan A Essa. 2002. Controlled animation of video sprites. In *Symposium on Computer Animation*. 121–127.

Arno Schödl, Richard Szeliski, David H Salesin, and Irfan Essa. 2000. Video textures. In *SIGGRAPH*. 489–498.

Christian Schulz, Christoph von Tycowicz, Hans-Peter Seidel, and Klaus Hildebrandt. 2014. Animating Deformable Objects Using Sparse Spacetime Constraints. *ACM Trans. Graph. (Proc. SIGGRAPH)* 33, 4, Article 109 (2014), 10 pages.

Sebastian Starke, Ian Mason, and Taku Komura. 2022. DeepPhase: Periodic autoencoders for learning motion phase manifolds. *ACM Trans. Graph. (Proc. SIGGRAPH)* 41, 4, Article 136 (2022), 13 pages.

Jingwei Tang, Vinicius C. Azevedo, Guillaume Cordonnier, and Barbara Solenthaler. 2021. Honey, I Shrunk the Domain: Frequency-aware Force Field Reduction for Efficient Fluids Optimization. *Comput. Graph. Forum (Proc. Eurographics)* 40, 2 (2021), 339–353.

Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. 2003. Keyframe control of smoke simulations. *ACM Trans. Graph. (Proc. SIGGRAPH)* 22, 3 (2003), 716–723.

Kevin Wampler and Zoran Popović. 2009. Optimal Gait and Form for Animal Locomotion. *ACM Trans. Graph. (Proc. SIGGRAPH)* 28, 3, Article 60 (2009), 8 pages.

Andrew Witkin and Michael Kass. 1988. Spacetime constraints. *Comput. Graph. (Proc. SIGGRAPH)* 22, 4 (1988), 159–168.

Chris Wojtan, Peter J. Mucha, and Greg Turk. 2006. Keyframe control of complex particle systems using the adjoint method. In *Symposium on Computer Animation*. 15–23.

Jonas Zehnder, Stelian Coros, and Bernhard Thomaszewski. 2021. SGN: Sparse Gauss-Newton for Accelerated Sensitivity Analysis. *ACM Trans. Graph.* 41, 1, Article 4 (2021), 10 pages.

Simon Zimmermann, Roi Poranne, James M Bern, and Stelian Coros. 2019. PuppetMaster: robotic animation of marionettes. *ACM Trans. Graph. (Proc. SIGGRAPH)* 38, 4, Article 103 (2019), 11 pages.