Vibration-Minimizing Motion Retargeting for Robotic Characters

SHAYAN HOSHYARI, Disney Research and University of British Columbia
HONGYI XU, Disney Research
ESPEN KNOOP, Disney Research
STELIAN COROS, ETH Zurich
MORITZ BÄCHER, Disney Research

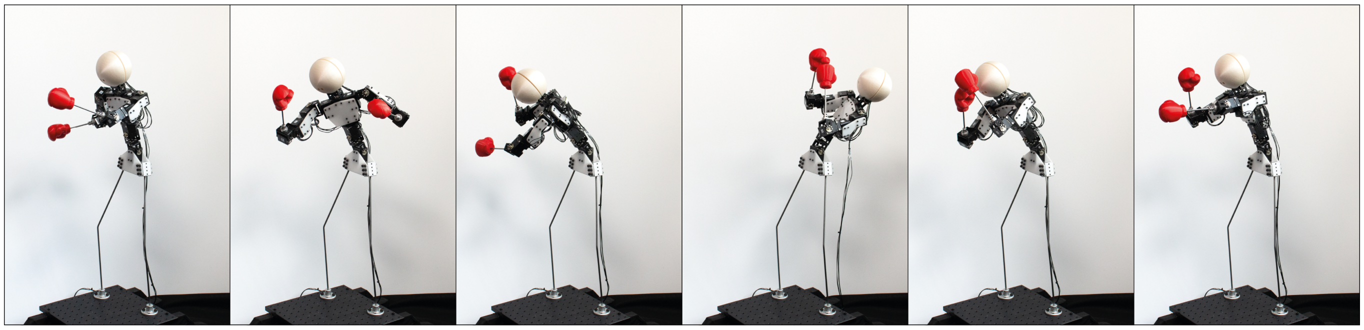


Fig. 1. We present a method for retargeting fast and dynamic animations onto physical robot characters, where the motor trajectories are optimized in order to suppress unwanted structural vibrations and match the artistic intent as closely as possible. We demonstrate our approach on a range of examples, including, as seen here, a boxing sequence with fast punches, blocks, and dodges.

Creating animations for robotic characters is very challenging due to the constraints imposed by their physical nature. In particular, the combination of fast motions and unavoidable structural deformations leads to mechanical oscillations that negatively affect their performances. Our goal is to automatically transfer motions created using traditional animation software to robotic characters while avoiding such artifacts. To this end, we develop an optimization-based, dynamics-aware motion retargeting system that adjusts an input motion such that visually salient low-frequency, large amplitude vibrations are suppressed. The technical core of our animation system consists of a differentiable dynamics simulator that provides constraint-based two-way coupling between rigid and flexible components. We demonstrate the efficacy of our method through experiments performed on a total of five robotic characters including a child-sized animatronic figure that features highly dynamic drumming and boxing motions.

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

Additional Key Words and Phrases: animation retargeting, robotic characters, dynamics, model reduction, vibration minimization, adjoint method

ACM Reference Format:
Shayan Hoshyari, Hongyi Xu, Espen Knoop, Stelian Coros, and Moritz Bächer. 2019. Vibration-Minimizing Motion Retargeting for Robotic Characters. *ACM Trans. Graph.* 38, 4, Article 102 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3323034

Authors' addresses: Shayan Hoshyari, Disney Research and University of British Columbia; Hongyi Xu, Disney Research; Espen Knoop, Disney Research; Stelian Coros, ETH Zurich; Moritz Bächer, Disney Research.

1 INTRODUCTION

Ever since Leonardo da Vinci's times, children and adults alike have been fascinated by mechanical systems that are designed to generate natural movements. Over the centuries, da Vinci's first automatons—the Mechanical Lion and Knight—have evolved into lifelike animatronic figures that are routinely deployed in museums, theme parks and movie studios across the world. And today, thanks to the advent of affordable, easy-to-use digital fabrication technologies and electromechanical components, the process of creating compelling robotic characters is easily accessible to anyone.

Keyframing techniques are commonly used to breathe life into animatronic characters. While these techniques are conceptually identical to those employed in *Computer Animation*, creating vibrant motions for *real-world* characters introduces unique challenges. These challenges stem from the physical characteristics of an animatronic figure's design. It is easy, for example, to design virtual characters that have as many degrees of freedom as needed. The design of their robotic counterparts, on the other hand, must balance motion versatility against the constraints imposed by the size, weight and placement of its mechanical components. Furthermore, the motions of real-world characters are strictly bound to the laws of physics. While the idealized limbs of a virtual character are perfectly rigid, for example, structural deformations are unavoidable in physical systems. Unfortunately, the combination of dynamic motions, weighty components and structural deformations lead to large-scale vibrations. These mechanical oscillations are due to the cyclic transfer between kinetic and potential deformation energy, and they negatively impact the character's performance.

To combat vibrations, mechanical structures are typically engineered to be as stiff as possible. While such designs work well in industrial settings, they are heavy and bulky, and therefore ill-suited for many types of robotic characters. An alternative is to slow down the motions that are performed until they approach the quasi-static regime. This, of course, is also undesirable. Animators are therefore left with only one option: endlessly tweaking motions in a painstaking, trial-and-error workflow. The goal of our paper is to fundamentally rethink this process through a physics-based motion retargeting method that is tailored for physical characters.

We present a novel approach to creating compelling motions for real-world characters. The input to our method consists of motions that are authored using standard animation software such as Autodesk's Maya. Leveraging a simulation model that balances speed and accuracy, the computational method we present generates an optimized motion that deviates from the input as little as possible while minimizing displeasing artifacts that arise from structural vibrations. We demonstrate the efficacy of our method through a variety of lightweight physical structures that generate complex motions. Succinctly, our technical contributions are as follows:

- Dynamics-aware motion retargeting for physical characters.
- Continuous space-time optimization for computationally suppressing structural vibrations.
- A general formulation of constrained multibody dynamics with two-way coupling between rigid and elastic components.
- Evaluations of our system through lively animations that are retargeted to complex robotic characters.

2 RELATED WORK

Computational Design and Fabrication. Fueled by advances in digital fabrication technologies, the past five years have seen a surge of research projects aimed at bringing animated characters to the real world. For example, a large body of work has targeted kinematic mechanisms that are specifically designed to create compelling motions [Bächer et al. 2015; Ceylan et al. 2013; Coros et al. 2013; Song et al. 2017; Thomaszewski et al. 2014; Zhang et al. 2017], structures that create motions by virtue of quasi-static deformations [Bern et al. 2017; Gauge et al. 2014; Megaro et al. 2017; Skouras et al. 2013; Xu et al. 2018], increasingly complex robotic creatures that locomote using legs and wheels [Geilinger et al. 2018; Ha et al. 2017; Megaro et al. 2015], and even devices that are designed to fly [Du et al. 2016; Umetani et al. 2014]. The technique we introduce here complements this body of work. In particular, we focus our attention on physical systems that are composed of both rigid and deformable elements, and are expected to perform very dynamic motions. This setting introduces a challenge that has not yet been addressed in the computer graphics community, namely that of reasoning about visually salient low-frequency large amplitude structural vibrations. Closest to our work is the recent method of [Chen et al. 2017], which accurately models the dynamics of elastic bodies in contact with the environment. However, while they target systems which are entirely passive, our method addresses deformation- and dynamics-aware active control of actuated systems such as animatronic characters.

Trajectory Optimization. Trajectory optimization, or spacetime constraints, is a general methodology often employed for control problems. Using this methodology, motion synthesis is formulated as an optimization problem where the equations of motion are treated as soft or hard constraints, and control forces that exploit the natural dynamics of the system are generated as output. In computer animation, many research works have focused on developing trajectory optimization techniques that target specific systems such as rigid-body simulations [Popović et al. 2003], fluids [McNamara et al. 2004; Treuille et al. 2003], deformable bodies [Barbič et al. 2009; Li et al. 2014; Schulz et al. 2014], human and animal motions [Lee et al. 2018; Si et al. 2014; Tan et al. 2011], etc. Different from these prior works, we seek an optimal control formulation to create motions for *physical* robotic characters that are composed of both rigid and compliant mechanical components. The lightweight, compliant nature of such physical designs demands techniques which can reason about deformations and mechanical vibrations. This problem setting therefore differentiates our work from research efforts aimed at robotic creatures that are mobile, but assumed to be free of structural deformations [Geilinger et al. 2018; Megaro et al. 2015]. It is worth noting that the engineering community is actively studying the problem of vibration suppression [Economou et al. 2000; Huertas and Rohal'-Ilkiv 2012; Pappalardo and Guida 2018; Zhou et al. 2002], but the focus there is on specific structures and mechanisms in isolation. Our goal, in contrast, is to develop a flexible framework that can be applied to a variety of animatronic figure designs.

Dynamic Simulation and Adjoint Method. The area of physics-based modeling has a rich history within computer graphics. For our problem domain, techniques that couple different types of simulation models are highly relevant, since the physical characters we consider are multi-body systems composed of both rigid and flexible components. In particular, our simulation model, which accurately captures two-way coupling effects, is derived by combining concepts developed for constraint-based simulation [Baraff 1996; Tournier et al. 2015], volumetric simulation of elastic bodies undergoing large deformation [Smith et al. 2018] and subspace formulations [An et al. 2008; Barbič and James 2005; Hauser et al. 2003; James and Pai 2002]. The latter is particularly important: because we use forward simulations in the inner loop of a motion optimization scheme, computational efficiency and predictive power are both of utmost importance. Unlike prior works either on reduced character simulation driven with predetermined rigged motion [Galoppo et al. 2009; Kim and James 2012; Xu and Barbič 2016] or full space simulation with coupling between skeleton and deformable skin [Kim and Pollard 2011; Liu et al. 2013; Shinar et al. 2008], we accurately model the two-way coupling between rigid body dynamics and reduced flexible body motion. Our simulation-based motion optimization technique uses the adjoint method to compute gradients [Lions 1971]. In this sense, our work is related to others that employe the *discrete* adjoint method for time-dependent problems, such as the control of fluids [McNamara et al. 2004], particle systems [Wojtan et al. 2006], or soft deformable bodies [Barbič et al. 2009] in computer animation. Our method, however, is tailored to physical characters. We therefore formulate *constrained* time-dependent Differential Algebraic Equations (DAEs) to model mechanical joints and the two-way coupling between rigid and elastic bodies. To decouple our adjoint formulation from the underlying time-stepping scheme, we use the *continuous-in-time* adjoint formulation for the governing DAEs, the theory of which is thoroughly reviewed in [Cao et al. 2003]. While related to the recent approach of Pappalardo and Guida [2018], we note that our simulation and motion optimization formulations scale to physical systems that are significantly more complex.

3 OVERVIEW

Given an animation sequence created by an artist, in the form of time-varying motor angles for the rig joints, the goal of our work is to retarget the motion onto a physical robotic character while avoiding undesirable vibrations due to system dynamics. We can extract the motor angles from the input animation sequence and replay them on the physical robot. However, this may often lead to substantial unwanted vibrations e.g. due to unavoidable compliance in the components of the physical robot. This can cause the physical robot to deviate significantly from the target animation sequence (cf. supporting video). In this work, we provide a computational tool that optimizes the motor trajectories in order to suppress vibrations, while matching the intended animation as closely as possible.

Our robotic characters are multi-body systems consisting of both rigid components, such as mechanical joints and motors, and flexible bodies that will deform under dynamic motions (see Fig. 2). Rigid components are connected to each other using mechanical joints, and flexible bodies are coupled to adjacent rigid components. To physically simulate the dynamics of the system, we timestep all subsystems together in a unified integration scheme, while accurately modeling the two-way coupling effects between subsystems. The deformations of flexible bodies are moderate when observed from a coupled rigid body, and by exploiting this, we achieve fast and accurate simulation of the robotic characters via modal reduction techniques.

In the following sections, we first describe the constituent elements of the simulation, including elastodynamics, rigid body dynamics, and constraints. We then present a unified integration scheme for dynamics simulation (Sec. 4). Fast subspace simulation of the robotic characters is presented in Sec. 5, where we detail how the rigid-body dynamics are integrated in global coordinates while the reduced deformable simulation are solved in the local frame of a coupled rigid body. Using our dynamics simulator, we formulate a space-time optimization on motor controls in order to suppress vibrations using a continuous adjoint method (Sec. 6). In Sec. 7, we validate our simulation model, present two simple and illustrative examples, and demonstrate the efficacy of our optimization method by retargeting five rich motions to complex characters.

4 CONSTRAINED DYNAMICS

To simulate our robotic characters (compare with Fig. 2), we use an elastodynamics model to represent flexible parts of our components, and rigid bodies to represent parts with a high enough stiffness. The rigid bodies, in turn, are connected to one another with either mechanical joints such as hinges or ball-and-sockets, or standard rotational motors.

4.1 Elastodynamics

The motion of a 3D point $\mathbf{x}(\mathbf{X}, t)$ on a deformable body, located at the reference location $\mathbf{X}$ at time t_0, is governed by the equations of

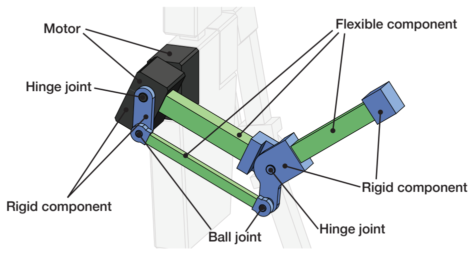


Fig. 2. **Simulating Robotic Characters** To simulate the dynamic behavior of our robotic characters, we represent flexible parts of components with deformable bodies (in green), and sufficiently stiff parts with rigid bodies (in blue). To enforce two-way coupling constraints between deformable and rigid bodies, and mechanical constraints between pairs of rigid bodies, we formulate a unified constrained dynamics model. We support assemblies with loops, and components with a wide range of geometries, masses, and stiffnesses.

motion

$$\rho\,\ddot{\mathbf{x}}(\mathbf{X}, t) - \nabla_{\mathbf{X}} \cdot \mathbf{P}^T(\mathbf{X}, t) - \rho\,\mathbf{g} = 0 \tag{1}$$

where the density ρ may vary within the reference domain Ω. Integrated over Ω, the divergence of the transposed first Piola-Kirchhoff stress tensor $\mathbf{P}$ results in internal elastic forces that counteract inertia and gravity $\mathbf{g}$.

Spatially discretizing the components with tetrahedral elements and interpolating their nodal displacements with Lagrange shape functions, we form the standard first-order ODE

$$\dot{\mathbf{u}} = \mathbf{v} \quad \text{and} \quad \underline{\mathbf{M}}\dot{\mathbf{v}} = \underline{\mathbf{f}}(\mathbf{u}, \mathbf{v}), \tag{2}$$

where $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{3n}$ collect the displacements and velocities of the n nodal degrees of freedom. The forces $\underline{\mathbf{f}} = -\mathbf{f}_{\text{damp}} - \mathbf{f}_{\text{int}} + \mathbf{f}_{\text{ext}}$ combine internal forces $\mathbf{f}_{\text{int}}$, external forces $\mathbf{f}_{\text{ext}}$ that we set to gravity, and Rayleigh damping $\mathbf{f}_{\text{damp}}(\mathbf{u}, \dot{\mathbf{u}}) = \left(\alpha\underline{\mathbf{M}} + \beta\mathbf{K}(\mathbf{u})\right)\dot{\mathbf{u}}$ with mass matrix $\underline{\mathbf{M}}$ and tangent stiffness $\mathbf{K}$. Our components undergo large rotations, and deformations are moderate even with our relative coordinate formulation (see Sec. 5). Experimenting with elements and hyperelastic models of increasing order and complexity, we observed quadratic elements and the Neo-Hookean model to provide the best tradeoff between model complexity and accuracy. Linear elements failed to capture bending deformations of our components. Above, we underline quantities for which the same standard letters are common in rigid body dynamics (where we overline them).

4.2 Rigid Body Dynamics

To transform points in body coordinates into global coordinates, we characterize states of bodies with their orientation $\mathbf{R}(t)$ and position $\mathbf{c}(t)$. Columns of rotations $\mathbf{R}$ represent a body's frame axes $[\mathbf{r}_x, \mathbf{r}_y, \mathbf{r}_z]$, and the position of its center of mass $\mathbf{c}$ the frame center at time t. The state of a body is governed by the first-order ODE form

$$\begin{bmatrix} \dot{\mathbf{c}} \\ \dot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{w} \\ \mathbf{Q}\boldsymbol{\omega} \end{bmatrix} \text{ and } \begin{bmatrix} \overline{\mathbf{M}} & \\ & \mathbf{I_c} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{w}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{f}} \\ \boldsymbol{\tau}_\mathbf{c} - [\boldsymbol{\omega}]_\times \mathbf{I_c}\boldsymbol{\omega} \end{bmatrix} \tag{3}$$

of the Newton-Euler equations where $\mathbf{w}$ is the linear and $\boldsymbol{\omega}$ the angular velocity, and $\overline{\mathbf{M}}$ the mass and $\mathbf{I_c} = \mathbf{R}\mathbf{I}_{\text{rb}}\mathbf{R}^T$ the moment of the inertia of the body, with $\mathbf{I}_{\text{rb}}$ referring to the *constant* angular mass in body coordinates. To reduce required normalization, we represent rotations with quaternions $\mathbf{q}$, and use the 4×3-transformation matrix $\mathbf{Q}(\mathbf{q})$ [Witkin and Baraff 1997]. The net force $\bar{\mathbf{f}}$ acting on the body includes gravity, and for the sake of brevity, we absorb the term $[\boldsymbol{\omega}]_\times \mathbf{I}_\mathbf{c}\boldsymbol{\omega}$ into the net torque $\boldsymbol{\tau}_\mathbf{c}$ in what follows. $[\boldsymbol{\omega}]_\times$ is the matrix form of the cross-product with $\boldsymbol{\omega}$.

4.3 Constraints

We have two types of constraints in our dynamic system: mechanical joints that couple rigid bodies pairwise, and interfaces between deformable and rigid bodies where degrees of freedom of the tetrahedral simulation mesh are moving as-rigidly-as-possible with the attached bodies.

Mechanical Joints. To formulate constraints between pairs of rigid bodies, we attach rigid frames to each body, then transform them with their respective orientations and positions. Asking their centers or axes to coincide or remain orthogonal to one another (see, e.g., [Coros et al. 2013]), we formulate constraints

$$C(t, \mathbf{c}(t), \mathbf{q}(t)) = 0 \tag{4}$$

that depend on the orientations and positions of the bodies due to these transformations. For motors, we actively change the relative positions of the two frames. Hence, we have a direct dependence on t.

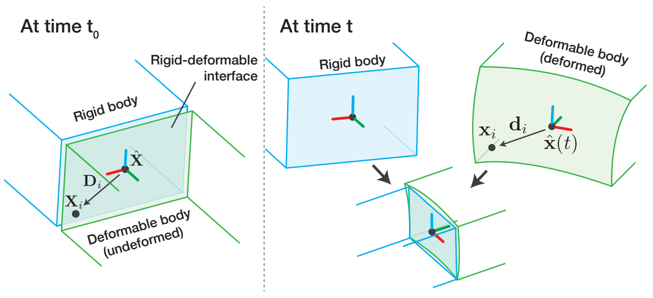


Fig. 3. **Coupling Deformable to Rigid Bodies** To couple deformable to rigid bodies, we define a frame on each body, asking their centers and axes to be equal at time t. To define a center on the rigid body, we extract the "centroid" $\hat{\mathbf{X}}$ of the area-weighted interface nodes $\mathbf{X}_i$ at t_0, transforming its location to rigid body coordinates. At time t, we then ask the transformed center to equal the "centroid" of the deformed interface nodes $\hat{\mathbf{x}}(t)$. To extract the rotation between the undeformed and deformed interface, we compute the transformation that maps vectors $\mathbf{D}_i = \mathbf{X}_i - \hat{\mathbf{X}}$ to their deformed configuration $\mathbf{d}_i = \mathbf{x}_i - \hat{\mathbf{x}}$. We then ask the orientation of the frame on the rigid body to equal the rotational part of this transformation.

Coupling Deformable to Rigid Bodies. To couple deformable to rigid bodies, we use a similar mechanism, asking frames on each body to coincide in global coordinates (compare with Fig. 3). However, because interface nodes on the deformable body do not remain rigid, the formulation of coupling constraints is more involved. We rely on a similar approach as described by Barbič and Zhao [2011]. However, unlike them, we do not assume our input assemblies to be hierarchical. Due to loops in our robotic character assemblies, deformable bodies can be coupled to more than one rigid body (compare with Fig. 2). Moreover, rather than using a penalty method [Barbič and Zhao 2011; Kim and James 2012], we enforce our coupling with *constraints* which constitutes a more physically accurate model besides enabling the stable integration without the necessity of a tedious parameter tuning.

In a first step, we extract frame centers on each body that we seek to coincide throughout the motion. To this end, we compute the area-weighted average $\hat{\mathbf{X}}$ and $\hat{\mathbf{x}}(t)$ of nodes on the interface at rest and time t, respectively. We then ask the two "centroids" to coincide in the local frame of the rigid body

$$\mathbf{R}(t_0)^T(\hat{\mathbf{X}} - \mathbf{c}(t_0)) - \mathbf{R}(t)^T(\hat{\mathbf{x}}(t) - \mathbf{c}(t)) = 0. \tag{5}$$

To constrain the relative orientations of the two bodies, we first find the optimal *linear* transformation $\mathbf{A}(t)$ that transforms difference vectors $\mathbf{D}_i = \mathbf{X}_i - \hat{\mathbf{X}}$ to their deformed configuration $\mathbf{d}_i(t) = \mathbf{x}_i(t) - \hat{\mathbf{x}}(t)$ by minimizing the sum of squared differences $\sum_i \hat{w}_i(\mathbf{A}\mathbf{D}_i - \mathbf{d}_i)^2$, weighted with the normalized interface area incident to node i [Müller et al. 2005]. We then ask the orientations of the interface on the deformable and rigid body to coincide

$$\mathbf{R}(t)\mathbf{R}(t_0)^T - \text{PD}(\mathbf{A}(t)) = 0, \tag{6}$$

where we use the polar decomposition (operator PD) to extract the rotational part of the transformation $\mathbf{A}$. To minimize the number of constraints, we enforce the orthogonality between columns of the combined rotation $\mathbf{R}(t)\mathbf{R}(t_0)^T$ and the axes of the rotational part of $\mathbf{A}$ with three dot product constraints (compare with Fig. 3).

Because the deformed nodes on the interface depend on the displacement, our combined constraints

$$C(t, \mathbf{c}(t), \mathbf{q}(t), \mathbf{u}(t)) = 0 \tag{7}$$

depend on $\mathbf{u}$ as well as the rigid body locations and orientations.

While an automated extraction of interfaces would be possible, we consider them user input. This has the advantage that only the rigid bodies' mass properties but not their exact geometry have to be known to ensure simulation accuracy. Note that per-vertex constraints are not an alternative, as they can cause locking artifacts in reduced simulations.

Enforcing Constraints. To enforce constraints, we form constraint Jacobians $C_\mathbf{c}$, $C_\mathbf{q}$, and $C_\mathbf{u}$ w.r.t. locations and orientations of rigid bodies, and displacements of our deformable bodies. We then add constraint forces $C_\mathbf{c}^T\boldsymbol{\Lambda}$ and $C_\mathbf{u}^T\boldsymbol{\Lambda}$ to $\bar{\mathbf{f}}$ and $\underline{\mathbf{f}}$, respectively. $\boldsymbol{\Lambda}$ is a vector of Lagrange multipliers (one per constraint). Note that constraint torques $(C_\mathbf{q}\mathbf{Q})^T\boldsymbol{\Lambda}$ require a transformation with matrix $\mathbf{Q}$ before we add them to $\boldsymbol{\tau}_\mathbf{c}$. By construction, these generalized forces do not do any work on the system. Hence, they do not add nor remove energy [Witkin and Baraff 1997].

To form Jacobians of our coupling constraints, we need to take derivatives of the polar decomposition. We refer the reader to the appendices of Barbič and Zhao [2011] and Pérez et al. [2015] for derivations of the first and second derivative of operator PD. While we only need first derivatives for simulation, second derivatives are required for gradient computations during optimization (see Sec. 6).

Unified DAE. To form the system of DAEs that describes the dynamics of our robotic characters, we combine the deformable and rigid body ODEs (Eqs. 2 and 3)

$$\dot{\mathbf{U}} - \mathbf{T}\mathbf{V} = 0 \quad \text{with} \quad \mathbf{T}(\mathbf{U}) = \operatorname{diag}(\mathbf{E}, \mathbf{Q}(\mathbf{q}), \mathbf{E}) \tag{8}$$

$$\mathbf{M}\dot{\mathbf{V}} - \mathbf{F} - (C_{\mathbf{U}}\mathbf{T})^T\boldsymbol{\Lambda} = 0 \quad \text{with} \quad \mathbf{M}(\mathbf{U}) = \operatorname{diag}\left(\overline{\mathbf{M}}, \mathbf{I}_\mathbf{c}(\mathbf{q}), \underline{\mathbf{M}}\right) \tag{9}$$

with the algebraic equations $C = 0$ into one unified system. In above equations, we collect the positions and orientations of the rigid bodies, and the displacements of the deformable bodies, in a generalized position vector $\mathbf{U} = [\mathbf{c}\ \mathbf{q}\ \mathbf{u}]^T$, and corresponding velocities in a generalized velocity vector $\mathbf{V} = [\mathbf{w}\ \boldsymbol{\omega}\ \mathbf{v}]^T$. Transformation matrix $\mathbf{T}$ relates velocities to the time derivatives of positions, and mass matrix $\mathbf{M}$ relates accelerations to generalized $\mathbf{F}(\mathbf{U},\mathbf{V}) = [\overline{\mathbf{f}}\ \boldsymbol{\tau}_\mathbf{c}(\mathbf{c},\mathbf{q},\boldsymbol{\omega})\ \underline{\mathbf{f}}(\mathbf{u},\mathbf{v})]^T$ and constraint forces $(C_\mathbf{U}\mathbf{T})^T\boldsymbol{\Lambda}$. The torque does not only depend on the locations, but also on the the orientations and angular velocities, as we consider torques $[\boldsymbol{\omega}]_\times \mathbf{I}_\mathbf{c}\boldsymbol{\omega}$ to be part of $\boldsymbol{\tau}_\mathbf{c}$. Matrices $\mathbf{E}$ are identities of appropriate but different sizes.

4.4 Time Discretization

A direct time integration of above DAEs is not possible. This is because our constraints $C = 0$ do not directly depend on velocities, hence neither an explicit nor an implicit discretization scheme would lead to a solvable system. To enforce our constraints, we have two options: we can either use the first or second time-derivative, $\dot{C} = 0$ or $\ddot{C} = 0$, respectively. The use of the second-time derivative results in a *semi-explicit, index*-1 DAE [Ascher and Petzold 1998; Brenan et al. 1996] that we can discretize and solve with *either* an explicit *or* implicit scheme. However, because our components are flexible but very stiff, high-order Runge-Kutta (RK) is unstable even if we only time-stepped the nonlinear deformable body ODE (see supplemental material; Sec. 3). Hence, an explicit scheme is not an option. We rely on velocity constraints $\dot{C} + \alpha C = 0, \alpha > 0$, where we add Baumgarte stabilization [Baumgarte 1972] to avoid numerical drift in positions, resulting in the *pure, index*-2 DAE [Ascher and Petzold 1998; Brenan et al. 1996]

$$\mathbf{G} = \begin{bmatrix} \mathbf{E} & & \\ & \mathbf{M} & -(C_\mathbf{U}\mathbf{T})^T \\ & & \end{bmatrix} \begin{bmatrix} \dot{\mathbf{U}} \\ \dot{\mathbf{V}} \\ \boldsymbol{\Lambda} \end{bmatrix} - \begin{bmatrix} \mathbf{T}\mathbf{V} \\ \mathbf{F} \\ -C_t - C_\mathbf{U}\mathbf{T}\mathbf{V} - \alpha C \end{bmatrix} = 0 \tag{10}$$

that we can only (directly) discretize and solve with an *implicit* scheme. This is because the constraints are *independent* of the algebraic variables $\boldsymbol{\Lambda}$. Due to the dependence of the motor angle on time t, the partial time-derivative of our constraints, C_t, is non-zero for constraints involving motors.

While a Backward Difference Formula (BDF) discretization of either the semi-implicit, index-1 or the pure, index-2 DAE is stable and would fulfill our requirements, the index-2 is preferable as it does not require second derivatives of our constraints for simulation, or *third* derivatives for gradient computations for our optimization (see Sec. 6). The latter would be tedious to derive and implement due to the polar decomposition in our coupling constraints.

To avoid numerical damping, we discretize our nonlinear system of DAEs, $\mathbf{G}(t, \mathbf{S}(t), \dot{\mathbf{S}}(t)) = 0$, with *state* vector $\mathbf{S} = [\mathbf{U}\ \mathbf{V}\ \boldsymbol{\Lambda}]^T$ and its time-derivative $\dot{\mathbf{S}} = [\dot{\mathbf{U}}\ \dot{\mathbf{V}}\ \dot{\boldsymbol{\Lambda}}]^T$, with a BDF-2 scheme [Ascher and Petzold 1998]

$$\begin{bmatrix} \left(\frac{\mathbf{U}_n - \hat{\mathbf{U}}_p}{\Delta\hat{t}}\right) - \mathbf{T}(\mathbf{U}_n)\mathbf{V}_n \\ \mathbf{M}(\mathbf{U}_n)\left(\frac{\mathbf{V}_n - \hat{\mathbf{V}}_p}{\Delta\hat{t}}\right) - \mathbf{F}(\mathbf{U}_n, \mathbf{V}_n) - [C_\mathbf{U}(t_n, \mathbf{U}_n)\mathbf{T}(\mathbf{U}_n)]^T\,\boldsymbol{\Lambda}_n \\ C_t(t_n, \mathbf{U}_n) + C_\mathbf{U}(t_n, \mathbf{U}_n)\mathbf{T}(\mathbf{U}_n)\mathbf{V}_n + \alpha C(t_n, \mathbf{U}_n) \end{bmatrix} = 0.$$

In the above equations, $\mathbf{S}_n = [\mathbf{U}_n\ \mathbf{V}_n\ \boldsymbol{\Lambda}_n]^T$ is the *unknown* next state, $\hat{\mathbf{S}}_p$ is set to the combination of the two previous states $-\frac{4}{3}\mathbf{S}_p + \frac{1}{3}\mathbf{S}_{p-1}$ that are *known*, and the time step $\Delta\hat{t}$ to $\frac{2}{3}$ of the chosen step size Δt (set to 0.5 ms for all our demonstrations). We assume the system to be at rest at time t_0, and set the initial conditions $\mathbf{S}_0$ accordingly (we use one BDF1 step at the start). To solve the resulting nonlinear equations for the next state $\mathbf{S}_n$, we use Newton's method where we linearize the system at the current iteration.

5 FAST SIMULATION OF ROBOTIC CHARACTERS

Our dynamic simulation scheme, as described in the previous section, enables us to accurately simulate our robotic characters when actuating the motors according to the artist-specified motion profiles. Nevertheless, simulations are slow for our complex characters due to the thousands of deformable Degrees Of Freedom (DOFs). It becomes even prohibitively expensive as we seek to *optimize* the motor trajectories. This is because every objective evaluation during line search, and every objective gradient evaluation, requires a simulation of the *entire* animation.

To enable fast *and* accurate simulation of our characters, we rely on established subspace methods [An et al. 2008; Barbič and James 2005]. However, our flexible components undergo large rotations, for which modal models are poorly suited. Taking a closer look at our flexible bodies, we observe that their motion is passive, driven by the inertial forces that are due to transformations of coupled rigid bodies. Moreover, their deformations are moderate w.r.t. the body coordinates of the latter. Inspired by multi-domain reduced simulation [Barbič and Zhao 2011], we therefore seek to integrate the reduced deformable dynamics in *relative coordinates* of one of the coupled rigid bodies, while time-stepping the rigid body motion in *global coordinates* (compare with Sec. 4). Before we discuss our reduced formulation, we describe the full simulation in relative coordinates.

Full Simulation in Relative Coordinates. For every deformable body, we choose the adjacent rigid body closest to the assembly "root" as a reference. While simulations are largely insensitive to this choice, bodies closer to the "root" are often the primary driver of the motion of a flexible component. As discretized in Sec. 4, a point $\mathbf{X} \in \mathbb{R}^3$ in the reference domain Ω is mapped to its deformed position $\mathbf{x}(\mathbf{X}, t) \in \mathbb{R}^3$ via interpolation of the nodal displacements $\mathbf{u} \in \mathbb{R}^{3n}$

$$\mathbf{x}(\mathbf{X}, t) = \mathbf{X} + \mathbf{u}(\mathbf{X}, t) \qquad \mathbf{u}(\mathbf{X}, t) = \boldsymbol{\Phi}(\mathbf{X})\mathbf{u}(t), \tag{11}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{3\times 3n}$ is the concatenation of 3×3 diagonal matrices set to the identity times the quadratic basis function of the corresponding node.

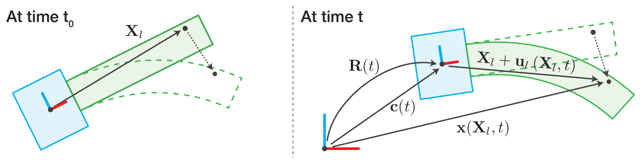

At time t_0
$\mathbf{X}_l$
At time t
$\mathbf{R}(t)$
$\mathbf{X}_l + \mathbf{u}_l(\mathbf{X}_l, t)$
$\mathbf{c}(t)$
$\mathbf{x}(\mathbf{X}_l, t)$

Fig. 4. **Relative Coordinate Formulation.** We discretize deformable parts of components *relative* to one of the coupled rigid bodies.

Discretized *relative* to one of the coupled rigid bodies (compare with Fig. 4), the reference point $\mathbf{X}_l \in \mathbb{R}^3$, in *local coordinates* of the coupled body, is transformed to its *global* deformed configuration according to

$$\mathbf{x}(\mathbf{X}_l, t) = \mathbf{R}(t)\,[\mathbf{X}_l + \mathbf{\Phi}_l(\mathbf{X}_l)\mathbf{u}_l(t)] + \mathbf{c}(t), \tag{12}$$

with the displacements $\mathbf{u}_l$ and the basis $\mathbf{\Phi}_l$ defined w.r.t. the *local* frame. To increase readability, we drop the subscript l in the following.

Starting from Eq. 1 and omitting arguments in the interest of brevity, we formulate the weak form of the equations of motion

$$\int_\Omega \mathbf{\Phi}^T\,\mathbf{R}^T(\rho\,\ddot{\mathbf{x}} - \nabla_\mathbf{X}\cdot\mathbf{P}^T - \rho\,\mathbf{g})\,d\mathbf{X} = 0 \implies$$

$$\int_\Omega \rho\,\mathbf{\Phi}^T\mathbf{R}^T\ddot{\mathbf{x}}\,d\mathbf{X} = \int_\Omega \mathbf{\Phi}^T\mathbf{R}^T\,\nabla_\mathbf{X}\cdot\mathbf{P}^T\,d\mathbf{X} + \int_\Omega \rho\,\mathbf{\Phi}^T\mathbf{R}^T\,\mathbf{g}\,d\mathbf{X}, \tag{13}$$

where we transform the equations to local coordinates of the coupled body by multiplying with $\mathbf{R}^T$. The integral on the left represents inertial forces (generalized mass times acceleration), and the two integrals on the right represent internal elastic and external gravitational forces, respectively. Plugging the second time derivative of the deformed configuration in relative coordinates (Eq. 12) into the integral on the left (see App. A for generalized mass matrices $\underline{\mathbf{M1}}$-$\underline{\mathbf{M6}}$; Sec. 2 in our supplemental material for a detailed derivation), we form the inertial forces that correspond to the linear and angular acceleration of our rigid bodies, and the accelerations of our deformable bodies

$$\underline{\mathbf{M}}_\omega\dot{\boldsymbol{\omega}} \qquad \underline{\mathbf{M}}_\mathrm{w}\dot{\mathbf{w}} \qquad \underline{\mathbf{M}}\dot{\mathbf{v}} \tag{14}$$

$$\underline{\mathbf{M}}_\omega(\mathbf{q},\mathbf{u}) = -\left(\underline{\mathbf{M1}} + \underline{\mathbf{M2}}(\mathbf{u})\right)\mathbf{R}^T \qquad \underline{\mathbf{M}}_\mathrm{w}(\mathbf{q}) = \underline{\mathbf{M3}}\mathbf{R}^T, \tag{15}$$

together with the fictitious centrifugal and Coriolis forces

$$\mathbf{f}_\mathrm{cen}(\mathbf{q},\mathbf{u},\boldsymbol{\omega}) = \sum_j\left(\boldsymbol{\omega}^T\mathbf{R}\left(\underline{\mathbf{M4}}_j + \underline{\mathbf{M5}}_j(\mathbf{u})\right)\mathbf{R}^T\boldsymbol{\omega}\,\mathbf{e}_j\right) \tag{16}$$
$$-\left(\underline{\mathbf{M6}} + \underline{\mathbf{M}}\mathbf{u}\right)(\boldsymbol{\omega}\cdot\boldsymbol{\omega})$$

$$\mathbf{f}_\mathrm{cor}(\mathbf{q},\boldsymbol{\omega},\mathbf{v}) = -\,2\underline{\mathbf{M2}}(\mathbf{v})\,\mathbf{R}^T\boldsymbol{\omega}, \tag{17}$$

that are due to our relative coordinate formulation. $\mathbf{e}_j$ is the j-th column of the identity matrix.

It remains to discuss the first and second integrals on the left. Because we rely on the Green strain (invariant under rigid body motion) and time-step the deformable bodies in their respective *relative* coordinates, we can discretize the internal and damping forces for each individual deformable body as usual (Sec. 4). Our constant gravity vector $\mathbf{g}$ is defined in *absolute* coordinates. Hence, we rotate it to relative coordinates before we evaluate the third integral, setting the gravitational forces of the deformable body to $\mathbf{f}_\mathrm{grav}(\mathbf{q}) = \underline{\mathbf{M3}}\,\mathbf{R}^T\mathbf{g}$.

In summary, to time-step deformable bodies in relative coordinates, we replace the deformable body ODE, $\mathbf{M}\dot{\mathbf{V}} - \mathbf{F} = 0$, in our simulation DAE (Eq. 8) with the relative formulation

$$\begin{bmatrix} \overline{\mathbf{M}} & & \\ & \mathbf{I}_\mathrm{c} & \\ \underline{\mathbf{M}}_\mathrm{w} & \underline{\mathbf{M}}_\omega & \underline{\mathbf{M}} \end{bmatrix}\begin{bmatrix}\dot{\mathbf{w}}\\ \dot{\boldsymbol{\omega}}\\ \dot{\mathbf{v}}\end{bmatrix} = \begin{bmatrix}\overline{\mathbf{f}}\\ \boldsymbol{\tau}_\mathrm{c}\\ \underline{\mathbf{f}}\end{bmatrix} \tag{18}$$

where $\underline{\mathbf{f}}$ is set to the forces $-\mathbf{f}_\mathrm{cen} - \mathbf{f}_\mathrm{cor} - \mathbf{f}_\mathrm{damp} - \mathbf{f}_\mathrm{int} + \mathbf{f}_\mathrm{grav}$.

Our generalized mass matrices are either constant ($\underline{\mathbf{M}}$, $\underline{\mathbf{M1}}$, $\underline{\mathbf{M3}}$, $\underline{\mathbf{M6}}$), consist of constant blocks ($\underline{\mathbf{M4}}$), or are or consist of blocks that are *linearly* dependent on the displacements or velocities of our deformable bodies (see App. A). Hence, they are well-suited for precomputation.

Note that our mass matrices are similar to the ones derived by Barbič and Zhao [2011] (they define 7 separate matrices while we reduce them to 6). However, our relative formulation differs in two fundamental ways: (1) their multi-domain simulation works on deformable bodies only, time-stepping each body in *relative* coordinates of the deformable subdomain of its parent. Hence, their method only works on hierarchical input and does not support loops. Because we represent deformable bodies *relative* to a rigid body, and time-step *all* of our rigid bodies in *global* coordinates, we can couple deformable bodies to *multiple* rigid bodies, and thus support loops in our robotic character assemblies. (2) In their system, they extract frame rotations and accelerations from the *previous* state when integrating their inertial forces. Hence, they integrate the latter *explicitly*. In contrast, our inertial forces depend on the *next* state of both our rigid and deformable bodies, enabling stable and accurate integration with an *implicit* scheme.

Reduced Simulation in Relative Coordinates. With our relative formulation, we can express local deformations of individual deformable bodies in a subspace while keeping the accuracy required for our retargeting. Precomputing modes $\mathbf{U}_\mathrm{r} \in \mathbb{R}^{3n\times r}$, $r \ll 3n$, we time-step with the same system (Eq. 18) as for a full simulation, projecting it onto and solving in the reduced space instead. To compute *constant* (blocks of) mass matrices for our reduced system, we set the basis $\mathbf{\Phi}$ to the reduced basis $\mathbf{\Phi}\mathbf{U}_\mathrm{r}$ (see App. A; Sec. 1 in our supplemental material for derivations). Precomputing the reduced mass matrices, we can then integrate all of our inertial and gravitational forces in $O(r^2)$ flops, without the need for computations in full space. We rely on cubature [An et al. 2008; von Tycowicz et al. 2013] for efficient evaluations of reduced internal forces and tangent stiffnesses.

To construct a subspace for our *constrained* deformations, we opt for a PCA basis (see Sec. 7 for validations and comparisons to modal derivatives): we first run full simulations of our robotic characters, then perform mass-PCA [Barbič and James 2005] on the local displacements of each deformable component. It is worth pointing out that this defines subspaces for individual components instead of a single subspace for the entire character. To ensure proper transfer of generalized forces between bodies, we add the first 6 linear modes, responsible for rigid body motion, when forming $\mathbf{U}_r$.

6 OPTIMIZATION

To retarget artist-specified input onto our physical robots, we seek to minimize differences between simulated and target states, putting a priority on the suppression of visible vibrations of large amplitude. To do so, we parameterize and optimize the time-varying motor angles, and solve for their optimal control.

Due to the flexibility in our components, our robots would deform under gravity even if we slowed down the target animation to the degree where inertial forces are negligible. Because we cannot hope to remove deformations that are due to gravity, we first perform quasi-static solves with motor angles set according to our input animation. Performing these quasi-static solves at the same time intervals as used for dynamic simulations, we define the target states $\tilde{\mathrm{S}}(t)$. In the remainder, we will discuss how we minimize the distance of simulated $\mathrm{S}(t)$ to target states $\tilde{\mathrm{S}}(t)$.

Parameterization. We represent the time-varying angle $\theta_i(t, \mathbf{p}_i)$ of motor i (hereafter referred to as i's *motor profile*) with a spline interpolation, parameterized with control points $\mathbf{p}_i$. Because we require C^2-continuity to prevent infinite motor torques in simulations, we rely on B-Splines. We initialize the control points by fitting the parameterized profiles to the input animation, collecting the spline parameters of all motors in a global parameter vector $\mathbf{p}$.

Retargeting Objective. The current state of a dynamic system depends on the *entire* history of previous states. To effectively suppress vibrations, and enable the formation of calibrated counterbalancing motion (cf. Single Motor Single Rod example, Fig. 7), it is essential to penalize *integrated* differences. Hence, a first objective that comes to mind minimizes deviations of the generalized position vector $\mathbf{U}$ from its target, integrated over the interval $[0, T]$.

Observing that deformable bodies do *not* deform if the relative positions and orientations of coupled rigid bodies remain constant over time, it follows that it is sufficient to track the target matching performance of the *rigid* bodies only. However, because vibrations tend to "accumulate" (cf. Single Motor Two Rods example, Fig. 8), it is imperative to track their performance in absolute instead of relative coordinates. In summary, we ask the simulated positions of our rigid bodies to remain as-close-as-possible to their target state in *global* coordinates

$$g_{\text{pos}}(t, \mathbf{U}(\mathbf{p})) = \frac{1}{2}\,\|\mathbf{c}(t, \mathbf{U}(\mathbf{p})) - \tilde{\mathbf{c}}(t)\|^2\,. \tag{19}$$

To penalize differences between simulated and target orientations, we introduce a second objective

$$g_{\text{ori}}(t, \mathbf{U}(\mathbf{p})) = \frac{1}{2}\left((\mathbf{r}_x \cdot \tilde{\mathbf{r}}_z)^2 + (\mathbf{r}_y \cdot \tilde{\mathbf{r}}_z)^2 + (\mathbf{r}_z \cdot \tilde{\mathbf{r}}_x)^2\right) \tag{20}$$

where the $\mathbf{r}$-axes are the columns of the simulated and target rotations, $\mathbf{R}(t, \mathbf{U}(\mathbf{p}))$ and $\tilde{\mathbf{R}}(t)$, of a rigid body. By integrating these differences over time, we formulate our *retargeting objective*

$$G(\mathbf{U}(\mathbf{p})) = \int_0^T g(t, \mathbf{U}(t, \mathbf{p}))\,\mathrm{d}t \tag{21}$$

$$g(t, \mathbf{U}(t, \mathbf{p})) = \sum_k w_{\text{pos}}^k(t) g_{\text{pos}}^k + w_{\text{ori}}^k(t) g_{\text{ori}}^k \tag{22}$$

where the weights, $w_{\text{loc}}^k(t)$ and $w_{\text{ori}}^k(t)$, for body k provide the user with a means to *emphasize* particular fractions of an animation (see our Dancer for an example, Figs. 9 and 10). Note that these weights are *constant* in the sense that no time-derivatives are required for numerical optimization.

Discussion. To avoid the "accumulation" of global and visible vibrations, and trade them for less visible vibrations, we prioritize the penalization of differences in *global positions*, setting weights w_{ori}^k for most bodies to zero during retargeting. We only use non-zero orientation weights when orientation affects function (e.g., for the end-effector attached to our robotic Bartender, Fig. 11).

In terms of frequencies and amplitudes, we can interpret our retargeting as compensating for *low*-frequency vibrations of *large* amplitude.

Regularization. Our objective measures performance w.r.t. *absolute coordinates.* To provide a means to penalize *relative differences*, i.e., closeness to the artistic input, we formulate a regularizer that compares the current profile to the input profiles. In addition, we ask motor profiles to be smooth by penalizing high accelerations

$$r_{\text{pro}}^i(\mathbf{p}) = \frac{1}{2}(\theta_i(t, \mathbf{p}) - \tilde{\theta}_i(t))^2 \quad \text{and} \quad r_{\text{acc}}^i(\mathbf{p}) = \frac{1}{2}\left(\ddot{\theta}_i(t, \mathbf{p})\right)^2. \tag{23}$$

Analogously to our objectives, we weigh these regularization terms with weights that vary with time but are constant from an optimization perspective

$$R(\mathbf{p}) = \int_0^T \left(\sum_i w_{\text{pro}}^i(t) r_{\text{pro}}^i + w_{\text{acc}}^i(t) r_{\text{acc}}^i\right) \mathrm{d}t. \tag{24}$$

Note that our regularizer only depends on the spline parameters but not on the state of the robot.

DAE-Constrained Retargeting. To retarget motor profiles to our physical robots, we minimize our objective under the *dynamic equilibrium constraint* ($\mathbf{G} = 0$, Eq. 10), satisfied at every $t \in [0, T]$

$$\min_{\mathbf{p}} G(\mathbf{p}, \mathbf{U}(\mathbf{p})) + R(\mathbf{p}) \tag{25}$$

$$\text{subject to} \quad \mathbf{G}(t, \mathbf{p}, \mathrm{S}(t, \mathbf{p}), \dot{\mathrm{S}}(t, \mathbf{p})) = 0 \quad \text{and} \quad \mathrm{S}(t_0) = \mathrm{S}_0.$$

As we assume our system to be at rest at the start of an animation sequence, the initial conditions S_0 do not depend on the spline parameters $\mathbf{p}$.

6.1 Adjoint System and Objective Gradient

While sensitivity analysis and the adjoint method have become standard tools for implicitly enforcing quasi-static equilibrium constraints in minimizations of design objectives (see, e.g., [Geilinger et al. 2018; Ha et al. 2017; Megaro et al. 2015] for recent examples), the computation of analytical gradients of our retargeting objective requires additional machinery [Cao et al. 2003]. As for quasi-static problems, we solve the equilibrium constraint whenever we make adjustments to spline parameters and seek to evaluate our objective or objective gradient. There are two options for computing analytical gradients for DAE models: *forward* and *backward* sensitivity analysis. If the number of parameters is small, and does not exceed the number of simulation DOFs, forward analysis is preferable. In all other circumstances, backward analysis is the method of choice [Cao et al. 2003]. Because the number of state variables is in the order of

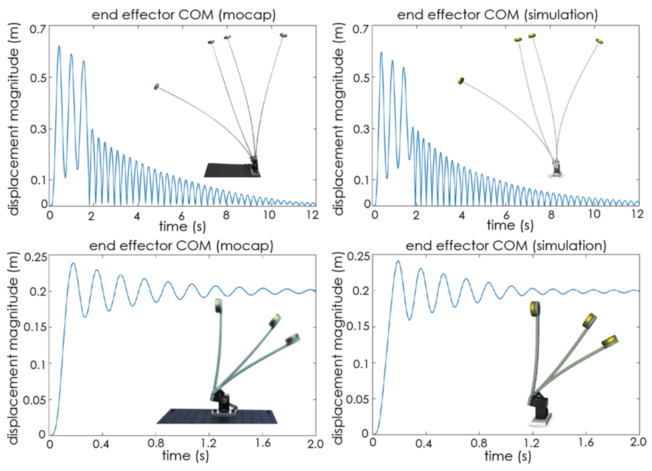


Fig. 5. **Material Fitting.** Using motion capture data, we fit elastic and damping parameters for our deformable rod and 3D-printed materials. Parameters are fitted with bisection, and we obtain the final parameters within 5 iterations. Our simulations are in close agreement with mocap data, and the obtained parameters are in agreement with tabulated values.

hundreds (for reduced simulation), and the number of parameters in the order of thousands, we rely on backward analysis. Note that forward analysis would be impractical for our problem.

Pointing the reader to our supplemental material (Sec. 2) for a detailed derivation, we here provide a recipe of how we compute analytical gradients. Upon parameter changes, we solve our simulation DAE, $\mathbf{G} = 0$ for $t \in [0, T]$, and store the states $\mathbf{S}(t)$ for later use. By time-stepping *backwards*, we then solve the *linear adjoint DAE*

$$\begin{bmatrix} \mathbf{E} & & \\ & \mathbf{M}^T & \\ & & \end{bmatrix} \dot{\boldsymbol{\lambda}} = \left(- \begin{bmatrix} & & \\ & \dot{\mathbf{U}}^T \mathbf{M}_{\mathbf{U}}^T & \\ & & \end{bmatrix} + \begin{bmatrix} \mathbf{G}_{\mathbf{U}}^T \\ \mathbf{G}_{\mathbf{V}}^T \\ \mathbf{G}_{\boldsymbol{\Lambda}}^T \end{bmatrix} \right) \boldsymbol{\lambda} + \begin{bmatrix} g_{\mathbf{U}}^T \\ g_{\mathbf{V}}^T \\ \\ \end{bmatrix} \tag{26}$$

with *initial conditions*, $\boldsymbol{\lambda}(T) = 0$, for the *adjoint variables* $\boldsymbol{\lambda}(t)$. In evaluations of gradients $g_{\mathbf{U}}$ and $g_{\mathbf{V}}$ of our objective, Jacobians $\mathbf{G}_{\mathbf{U}}$, $\mathbf{G}_{\mathbf{V}}$, and $\mathbf{G}_{\boldsymbol{\Lambda}}$ of our constrained dynamics equations (Eq. 10), and our generalized mass matrix $\mathbf{M}$ and the Jacobian $\mathbf{M}_{\mathbf{U}}$, we make use of the states $\mathbf{S}(t)$ from the solve of our simulation DAE. We then evaluate the objective gradient

$$\frac{\mathrm{d}G}{\mathrm{d}\mathbf{p}} = -\int_0^T \boldsymbol{\lambda}^T \mathbf{G}_{\mathbf{p}} \, \mathrm{d}t, \tag{27}$$

where we use the states $\mathbf{S}(t)$ from forward-stepping Eq. 10 in evaluations of the Jacobian $\mathbf{G}_{\mathbf{p}}$, and $\boldsymbol{\lambda}(t)$ from backward-stepping the corresponding adjoint system.

Discretizing both DAEs (Eqs. 10 and 26) with a BDF2 scheme with the *same* time interval (dividing the interval $[0, T]$ by the number of desired time steps), we ensure correspondence along the time axis. For numerical integration of our objective and objective gradient, we rely on a cubic Simpson's rule, and for minimization of our objective we use standard BFGS.

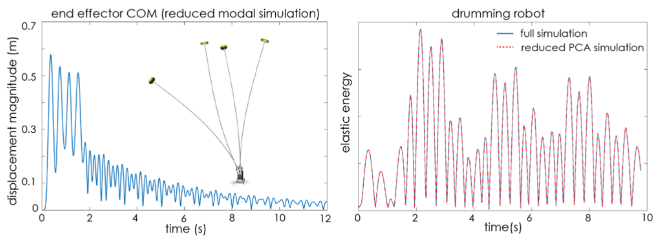


Fig. 6. **Reduced Basis.** The use of linear modes and modal derivatives (left) activates circular motion, and observed frequencies do not match the mocap data (compare with Fig. 5, left). Reduced simulations are in close agreement with full simulations of the *drumming* motion, even though we use the mass-PCA basis estimated from a full simulation of the *boxing* motion of our 13-DOF character (right).

7 RESULTS

We have used our computational framework to optimize a total of 6 examples with 7 different motion sequences, ranging from didactic mechanical systems to full-body robotic characters. By progressively increasing the complexity of the assemblies, their physical size, and the target motion sequences, we demonstrate the suitability of our method for complex and large-scale systems. In our accompanying video, we show the target animation sequence, and the playback of the unoptimized and optimized motor profiles on the physical characters, for all our examples.

Fabrication. All our demonstrators are driven with Dynamixel XM-430-W210 servomotors, controlled from a PC through a Dynamixel U2D2 interface. The servos provide sufficiently high torque that we can assume them to follow the specified motor angle trajectory with no deviation. Demonstrators are assembled using a combination of off-the-shelf Dynamixel mounting brackets, a small number of custom aluminum machined connectors, spring steel rods (diameters 4 and 5 mm), and 3D-printed parts. Structural parts are printed with Digital ABS material on an Objet Connex 350, and parts for visual appearance are printed with PLA on an Ultimaker 2+. Flat parts for the 13-DOF robot are laser cut from acetal sheets. The 100 g and 200 g weights are machined from brass.

Material Fitting. In order to accurately simulate the dynamics of our deformable bodies, we fit physical parameters (Young's modulus and Rayleigh damping coefficients) for the materials we use for our rods and 3D-printed components. Fig. 5 shows our physical setup where we mount either a rod or a 3D-printed piece on a single-DOF motor, comparing simulations to motion captured data. To measure the physical properties in the frequency range we are interested in, we attach a 100 g rigid mass to the end of the deformable body. Applying a 30 or 45 degree rotation, and returning to the rest angle, the deformable body starts to vibrate, and we use an OptiTrack system to capture the end effector motion. We observe that our reduced simulations (Fig. 5, left) closely match the captured motion (right). We note that the use of quadratic elements and implicit BDF2 (negligible numerical damping) is pivotal to accurately simulate these oscillatory bending deformations.

Validation. In reduced simulation, we reply on mass-PCA instead of standard linear modes for multiple reasons: vibrations in our systems are moderate in amplitude. Hence, we would need a large number of linear modes to express them. To approximate large deformations, linear modes can be augmented with modal derivatives [Barbič and James 2005], which are known to provide visually-pleasing dynamics. However, we found them to not perform well for our application domain as we illustrate in Fig. 6 (left) where we use 80 linear modes and modal derivatives (40 FPS) and still witness significant mismatch. In contrast, the PCA modes (Fig. 5, top right) match the captured data well with only 11 modes (275 FPS). Moreover, with a sufficient number of PCA modes, one can closely approximate full simulations, whereas with linear and modal derivatives one has to tune material parameters to non-physical values due to the numerical stiffening. For our retargeting, PCA modes suffice to express the observed vibrations and even extrapolate well for other motions with vibrations of similar amplitude (see Fig. 6, right). Contrary to standard linear modes, PCA modes do not introduce any locking artifacts due to constraints, as the full simulation enforces the constraints across the motion.

Single Motor Single Rod. Our first example consists of a single vertical deformable rod (spring steel, 70 cm) mounted on a servo-motor, with a 100 g mass attached at the end (see Fig. 7). Starting from the vertical rest configuration, we rotate the motor by 30°, pause for 1.5 s, and then return to the rest configuration. With the non-optimized piecewise linear motor control, the compliant rod vibrates substantially around the target poses, and significantly deviates from the target motion. Our optimization uses the Center of Mass (CoM) trajectory of the end effector as the objective. We observe that the optimized motor control suppresses the vibration, while the timing of the intended motion is kept as closely as possible. It can be seen that this is done in part by smoothing out the transitions to poses, and in part by preempting the motions and adding deformations ahead that cancel oscillations.

Single Motor Two Rods. Extending our first example, we move the motor from the base to the middle, connecting it to the base and the end effector with two 30 cm compliant rods (Fig. 8). Similarly to the first example, we attach a 200 g rigid mass as the end effector. Starting from the straight vertical pose, we rotate the upper rod by 90° to a horizontal pose. Due to the inertia of the mass, the lower rod deforms, and under the non-optimized motor control, the system vibrates significantly. In our retargeting objective, we track the positions of both the end effector and the motor. Fig. 8 shows that our optimization successfully removes the unwanted vibrations.

Dancer. In our third example, we increase the complexity to a 4-DOF character assembly where a 25 cm rod represents the body, and two arms are connected at the clavicle (Fig. 9). Each arm is comprised of two motors at the shoulder, one deformable rod as the arm, and a 200 g mass as the hand. The target motion is a 8.5 s dance sequence featuring expressive 3D arm movements (as shown in the supplementary video). A naïve transfer of the motion sequence to the servo-motors causes very significant vibrations, in particular due to the waving motion of the left hand (Fig. 9, first row), causing

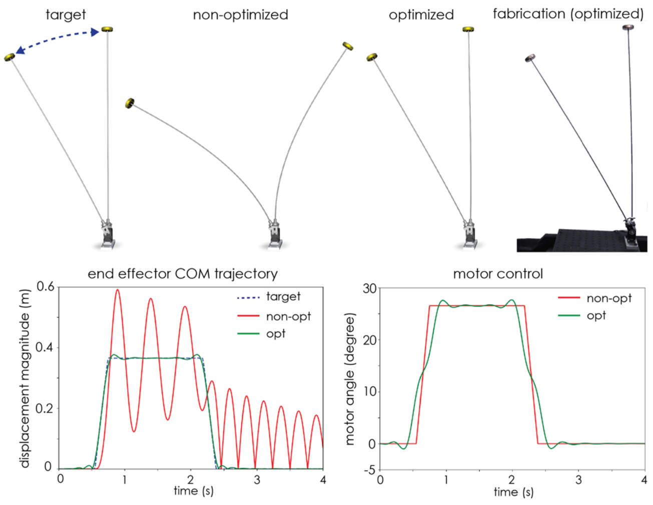


Fig. 7. **Single Motor Single Rod.** By preempting the movement of the input animation (bottom right), our optimized control produces a motion sequence that closely matches the target, as indicated in the animation frames (first row) and trajectory plot (botton left).

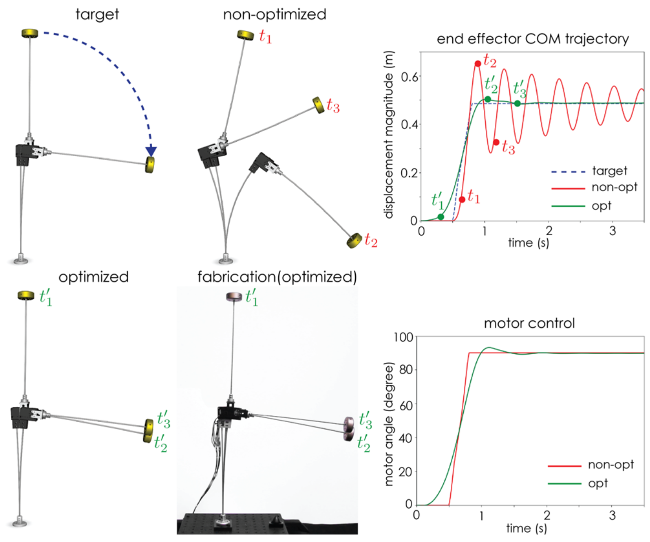


Fig. 8. **Single Motor Two Rods.** The two left columns show animation frames for our target, and non-optimized and optimized motion. The end effector trajectory and motor control is shown in the right column. Compared to the piecewise linear motor control, our optimized control signal successfully suppresses the vibration, smoothing out the two discontinuities.

the resulting motion of the physical character to be very different from the target.

In order to suppress the undesirable vibrations, we optimize the control signals for the 4 motors such that the center of mass trajectories of both the hands and the clavicle match the target as closely as possible. To demonstrate the user control over the optimization results, we experiment with 3 different sets of weights on the target matching terms. In the first experiment, we assign a weight of unity

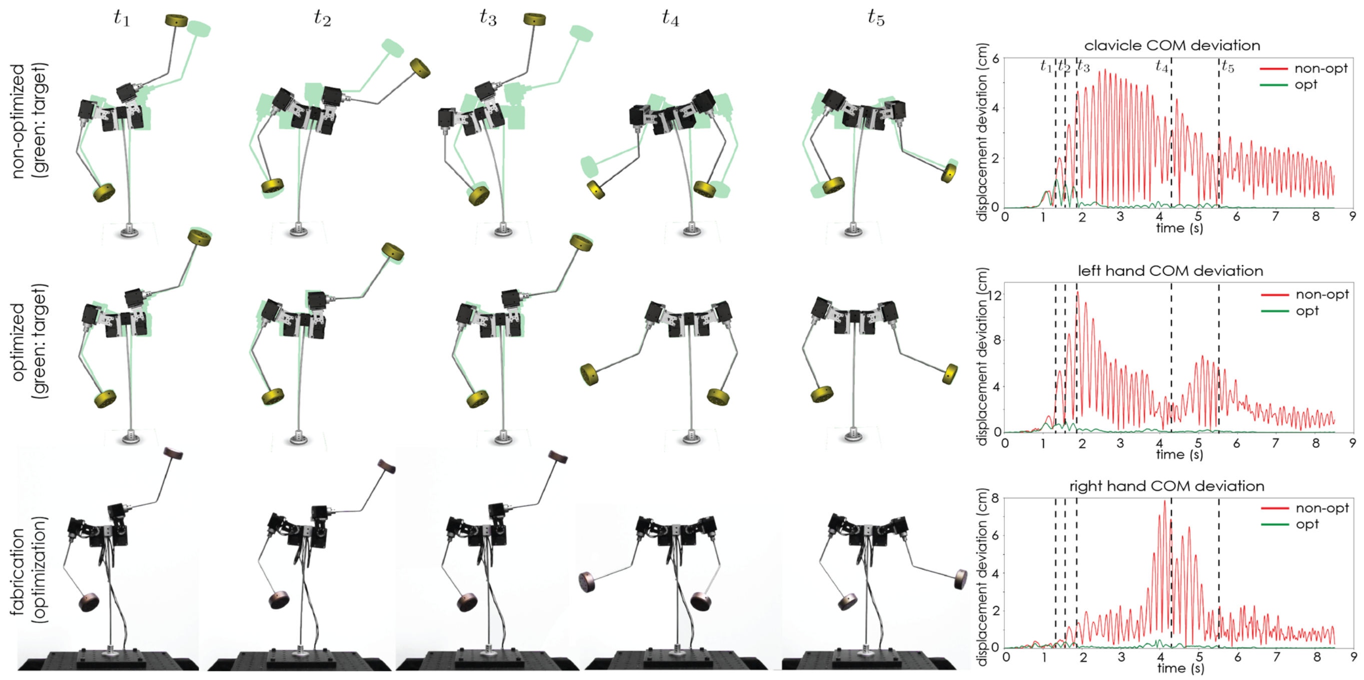


Fig. 9. 4-**DOF Dancer.** We show 5 representative simulation frames of the non-optimized (left columns, first row) and optimized animation (left columns, middle row), overlaid the target (in green). The bottom row shows the optimized motion on the physical character. Visualizations of the displacement error for the positions of the clavicle, left hand, and right hand, are shown in the column on the right. The first 3 frames show the waving motion while the last 2 are dancing poses. The non-optimized animation shows substantial vibrations on the body rod which contribute to even larger deviations on the arms. Our optimized control successfully removed these excessive vibrations. Deviations for the waving subsequence are largely removed, and counteracted by small deformations of the body rod. For optimization, weights w_{pos} were set to unity.

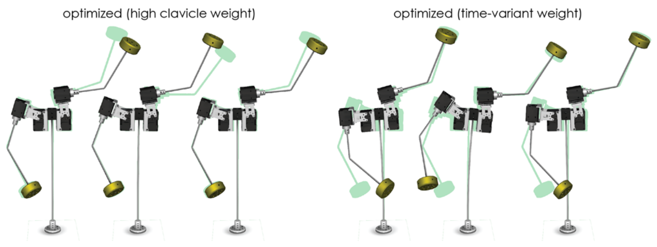

optimized (high clavicle weight)
optimized (time-variant weight)

Fig. 10. **User-Directable Control.** If we set the weight for the clavicle position to a high value (left), the body rod deviates less from the target than for the result with all weights set to unity (compare with Fig. 9, second row, first three frames). However, deviations for the left hand increase significantly for the waving subsequence. If we use a time-varying weight for the left hand, setting it to a high value for the waving segment (right), the motion of the right hand counterbalances the waving motion of the left hand. Visible vibrations are suppressed for all combinations of weights.

to all the target positions (Fig. 9, second row). The resulting motion is close to the target animation sequence, without inducing noticeable vibrations. However, we note that for the waving sequence, the displacement of the left hand end effector is mostly achieved by the deformation of the body rod, as opposed to the rotation of the corresponding shoulder motors. Therefore, in the second experiment (see Fig. 10, left), we increase the objective weight for trajectory matching at the clavicle by 100×. As expected, visible vibrations are suppressed, and the deformation of the torso is reduced. However, the latter comes at the cost of weakening the target-matching quality for the waving subsequence. We note that our system also supports the specification of time-varying weights, enabling the user to *emphasize* the importance of particular segments of animation. In the third experiment (Fig. 10, right), in order to restore the waving motion, we assign a time-varying weight to the left hand by increasing its weight by 100× for the waving segment. Interestingly, in this scenario the right arm deviates more from the input target, performing a counterbalancing motion to the wave while preventing vibrations. These experiments demonstrate that our system is able to suppress vibrations in multiple different ways, and can thus be tuned by the user in order to match a particular artistic intent.

Bartender Robot. As well as being visually displeasing, vibrations can also negatively impact the functional performance of robot characters. In this demonstration, we present a 4-DOF bartender robot arm (the end effector has 3 translational and 1 rotational DOFs), moving a glass of water between 3 locations. The height of the robot (without the base) is 45 cm. For this demonstrator, we not only ask the positions of the elbow and end effector to remain close to their target trajectories but also the *orientation* of the end effector, holding the glass, to remain in an upright position. The input trajectory, when played back on the robot, causes the robot to spill the drink

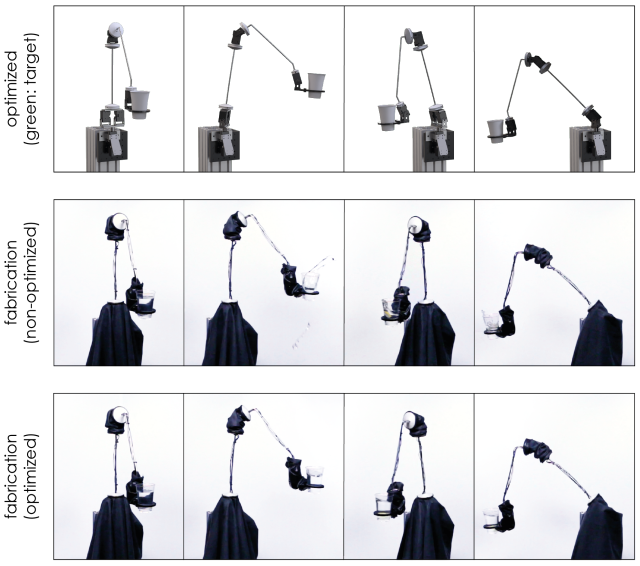


Fig. 11. **Bartender Robot.** The top row shows 4 frames of the simulated optimization result while the middle row shows the physical robot executing the non-optimized trajectory and spilling the drink (frames 2 and 4). Bottom row shows corresponding frames from the optimized trajectory, with no spillage.

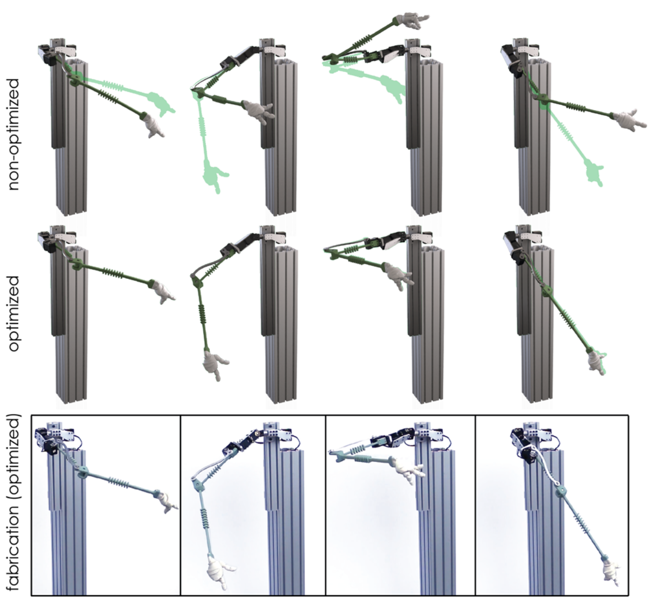


Fig. 12. **Rapper Arm.** The top row shows that under non-optimized control, the arm deviates significantly from the input sequence (in green). Our optimized control leads to a dynamic motion that is visually indistinguishable from the target, as shown in the second row (simulation) and third row (physical character).

due to the vibrations. By running our optimization, the robot is able to perform the same motion sequence without spilling the drink. We assume the water in the cup to behave like a rigid body of fixed mass—although this is an oversimplification, our modeling is sufficiently accurate to achieve the desired function. While the results are best seen in the supporting video, Fig. 11 shows the optimized result in simulation, the non-optimized motion of the physical robot which causes the drink to spill, and the retargeted animation which does not spill the drink. This demonstrator showcases that our method has applications outside of entertainment.

Rapper Arm. In this example, we demonstrate that our method supports mechanical systems with kinematics loops and flexible components of non-rod geometry, made of materials other than steel (Fig. 12). The system consists of two 3D-printed compliant arm components, connected with a 4-bar linkage, and 4 motors controlling the 3-DOF shoulder and the 1-DOF elbow motion. Note that unlike conventional rigid 4-bar linkages, here we have two compliant links and two rigid links that are connected with two hinge joints, one universal joint and one spherical joint in order to obtain the correct number of constraints while allowing for the deformable components to move out-of-plane. The input motion is rapping sequence where the arm is moving in a large 3D space. Our optimized motor controls successfully remove the substantial vibrations that are present in the non-optimized control input.

Drummer. We further examine the scalability of our framework, retargeting a drumming sequence for an 80 cm tall, 13-DOF full-body articulated robot (Fig. 13). Each arm has 3 DOFs at the shoulder and 1 DOF at the elbow. In addition, the upper body is actuated with 5 motors, controlling the motion of the head, neck, torso, spine, and pelvis. The character's legs are two 45 cm rods, fixed at the base, and the lower arms are 10 cm rods. The duration of the drumming motion is 10 s. In the accompanying video, we can see that the main deformation mode, which is excited by the drumming motion, is a backward-forward swaying motion. Interestingly, for the non-optimized case, the vibration amplitude exhibits significant periodic variation (being close to zero at some points in time)—this is seen in both the simulated and physical system. This periodic variation would make the manual suppression of the vibration, by either smoothening or offsetting controls, a very challenging, if not impossible, task. Our optimization targets the center of mass trajectories of the head, pelvis, and both hands. Without any noticeable degeneracy of motion quality, we successfully suppress the visible vibration of the overall system (peak amplitude is reduced from 7 cm to 1 cm)

Boxer. In a final demonstration, we retarget a boxing animation to the same 13-DOF full-body character, replacing the two hands with boxing gloves on both our simulation model and our physical system. Unlike the drumming sequence, the boxing motion contains faster motions with abrupt stops. The naïve retargeting causes excessive vibrations, especially when the character dodges and moves his upper body backwards and forwards. With the same objective and optimization parameters as for our Drummer, our optimized motor controls lead to deviations smaller than 1.5 cm (compared to 9 cm before optimization) while preserving the input animation without noticeable visual differences.

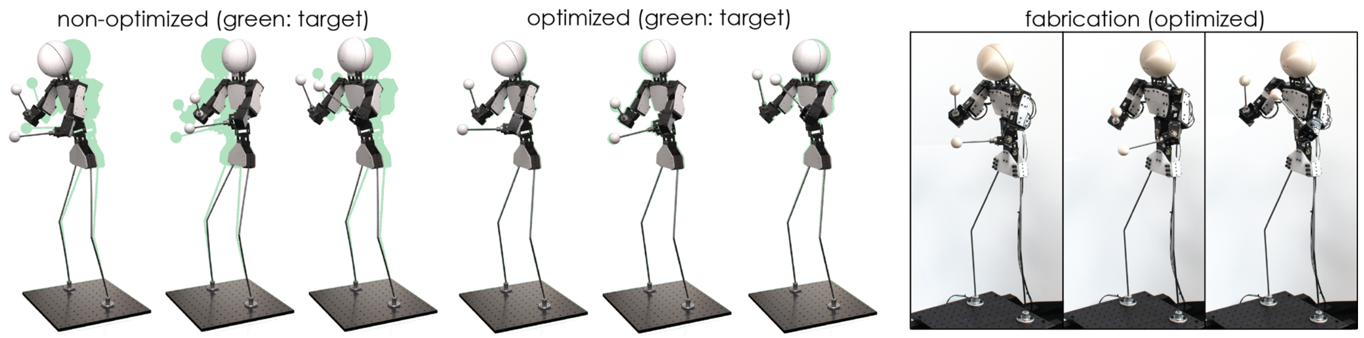


Fig. 13. **Drummer.** From the left to right, we show the non-optimized and optimized simulation motion (overlaid the target in green), and the retargeted motion on the physical character. With the non-optimized control, the vibration is excessive whereas in our result, the vibration is almost invisible.

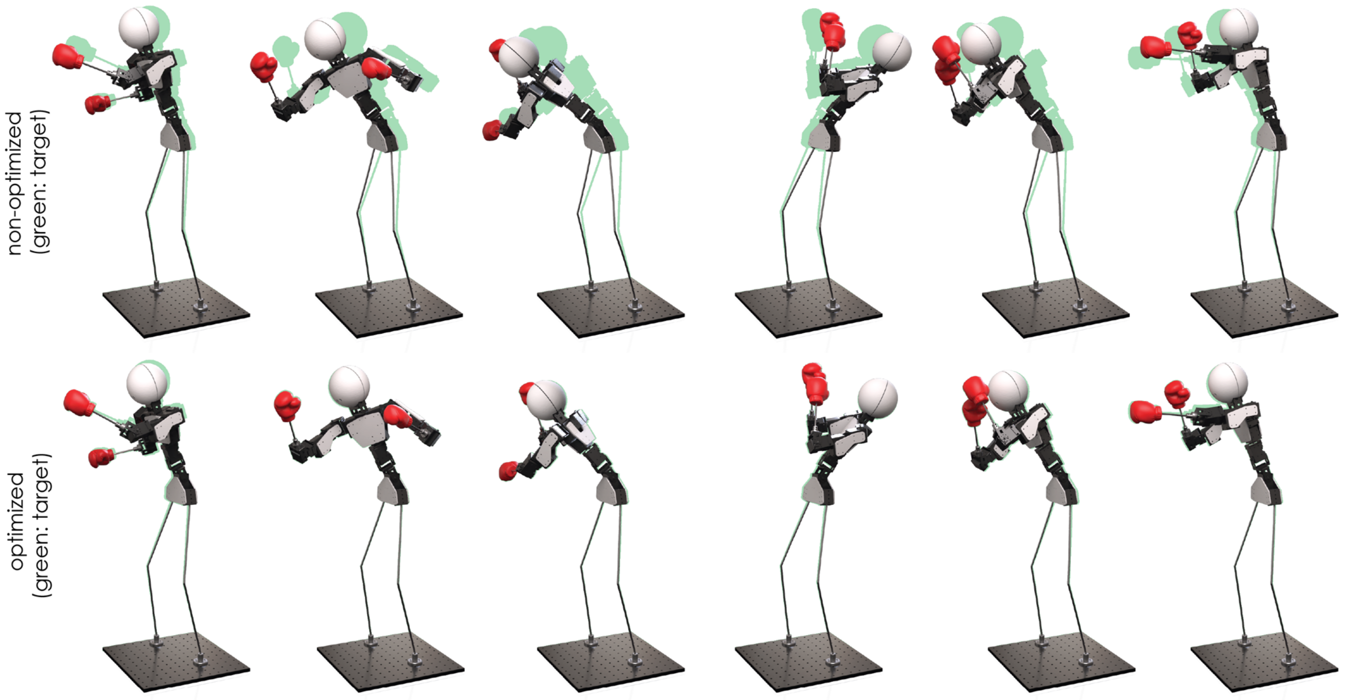
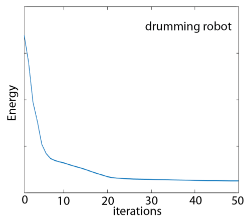


Fig. 14. **Boxer.** We show the boxing motion under non-optimized and optimized motor controls, overlaid the target motion. Compared to the non-optimized result, our result largely removes the vibration and matches the target with high visual quality. See Fig. 1 for the physical robot sequence.

Performance. Our simulations and optimizations are performed on a machine with an Intel Core i7-7700 processor (4 cores, 4.2 GHz) with 32 GB of RAM. Evaluations of internal forces and tangent stiffnesses of our deformable bodies are parallelized (multithreading). For minimization, we use standard quasi-Newton with BFGS [Nocedal and Wright 2006]. In our experiments, we set the relative residual tolerance to 10^{-4} and the maximum number of minimization iterations to 100. As we observe in the inset energy plot, our minimizations converge well and substantially decrease the vibrations from the input sequence, without sacrificing the quality of motion. See Table 1 for key statistics. A small timestep (0.5 ms) is required to achieve the accuracy we need, and reduced simulations are 550×-1100× faster than full simulations.



8 CONCLUSION

We have presented a computational tool that retargets artist-created animation onto physical robotic characters while minimizing unwanted vibrations due to system dynamics. Using model reduction to speed up simulation, we have accurately modeled the two-way coupling between rigid bodies and flexible bodies. Leveraging this

Table 1. **Performance Statistics.** From left to right, we report the number of deformable DOFs (defo) for full/reduced simulations, number of rigid bodies (rigid), optimization DOFs (opt), overall duration of the animation (dur), FPS for reduced simulations (sim), and the number of minutes it takes for retargeting (opt).

Example	defo #DOF	rigid #DOF	opt #DOF	dur (secs)	sim FPS	opt (mins)
motor+1 rod	30390/11	12	150	4	275	24.5
motor+2 rods	46242/19	18	100	3.5	260	70
dancer	81552/76	42	800	8.5	115	160
bartender	82377/67	36	1000	8.5	120	160
rapper	97788/82	54	1000	10	90	340
drummer	85731/75	96	2600	10	108	182
boxer	85731/75	96	3250	9.75	108	185

simulation model, we have optimized the motor control via a *continuous* adjoint method such that the physical character motion matches the artistic intent as closely as possible.

Our approach provides an automated way to retarget digital animations onto physical characters, and could also be used to evaluate the design of a physical robot before it is built. Moreover, by suppressing vibrations with the tuning of motor trajectories, we enable the design of expressive robotic characters that can be less stiff, and therefore lighter, cheaper, and more accessible to all.

Beyond the application of motion retargeting, our pipeline incorporates elements that will in general be valuable for fast simulation of multi-body systems incorporating coupled rigid-body dynamics and deformable bodies.

Our simulator captures the dynamic response of the physical characters well. However, it can be observed that there is still some deviation between the simulated dynamics and physical system, leading to small residual vibrations. This is likely due to the assumptions of perfectly stiff motor controls and mechanical joints—in reality both of these will have some non-infinite stiffness. Modeling these stiffnesses presents an interesting avenue for future work, but would require more in-depth calibration experiments for model fitting. A promising approach for eliminating these small residual vibrations would be to implement an online closed-loop controller on the physical robot, where the end effector trajectory is measured, and the motor control adjusted accordingly. This would make the approach more robust to modeling errors and to unexpected variations in the external loads.

In the current implementation, the user decides which components to model as rigid, and which to model as deformable. In many applications, this distinction is easy to make. However, for very large and complex assemblies it could be beneficial to automate this selection.

While we keep the design of our robotic characters fixed, and focus on the control problem, our formulation would also support the optimization of parameterized dimensions of components, or the positions and orientation of mechanical joints [Coros et al. 2013]. Inverse contact modeling [Ly et al. 2018] is another exciting avenue for future research.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments; Maurizio Nitti for animations; Alfredo Ayala, Todd Camill, David Christensen, Douglas Fidaleo, Tony Martin, Tom Miller, Günter Niemeyer, Tanner Rinke, Jon Snoddy, Krishna Tamminana (in alphabetic order) from Walt Disney Imagineering and Disney Research for fruitful discussions. This work has been supported by the SOMA project (European Commission, Horizon 2020 Framework Programme, H2020-ICT-645599).

REFERENCES

Steven S An, Theodore Kim, and Doug L James. 2008. Optimizing cubature for efficient integration of subspace deformations. *ACM Trans. Graph.* 27, 5 (2008), 165.

Uri M. Ascher and Linda R. Petzold. 1998. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations* (1st ed.). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Moritz Bächer, Stelian Coros, and Bernhard Thomaszewski. 2015. LinkEdit: Interactive Linkage Editing Using Symbolic Kinematics. *ACM Trans. Graph.* 34, 4, Article 99 (July 2015), 8 pages.

David Baraff. 1996. Linear-time dynamics using Lagrange multipliers. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 137–146.

Jernej Barbič, Marco da Silva, and Jovan Popović. 2009. Deformable object animation using reduced optimal control. *ACM Trans. Graph.* 28, 3 (2009), 53.

Jernej Barbič and Yili Zhao. 2011. Real-time large-deformation substructuring. *ACM Trans. Graph.* 30, 4, 91.

Jernej Barbič and Doug L. James. 2005. Real-Time Subspace Integration for St. Venant-Kirchhoff Deformable Models. *ACM Trans. Graph.* 24, 3 (July 2005), 982–990.

J. Baumgarte. 1972. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering* 1, 1 (1972), 1–16.

James M. Bern, Kai-Hung Chang, and Stelian Coros. 2017. Interactive design of animated plushies. *ACM Trans. Graph.* 36, 4 (2017), 80:1–80:11.

K. E. Brenan, S. L. Campbell, and L. R. Petzold. 1996. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* SIAM.

Yang Cao, Shengtai Li, Linda Petzold, and Radu Serban. 2003. Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution. *SIAM Journal on Scientific Computing* 24, 3 (2003), 1076–1089.

Duygu Ceylan, Wilmot Li, Niloy J. Mitra, Maneesh Agrawala, and Mark Pauly. 2013. Designing and Fabricating Mechanical Automata from Mocap Sequences. *ACM Trans. Graph.* 32, 6, Article 186 (Nov. 2013), 11 pages.

Desai Chen, David I. W. Levin, Wojciech Matusik, and Danny M. Kaufman. 2017. Dynamics-aware Numerical Coarsening for Fabrication Design. *ACM Trans. Graph.* 36, 4, Article 84 (July 2017), 15 pages.

Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W. Sumner, Wojciech Matusik, and Bernd Bickel. 2013. Computational Design of Mechanical Characters. *ACM Trans. Graph.* 32, 4, Article 83 (2013), 12 pages.

Tao Du, Adriana Schulz, Bo Zhu, Bernd Bickel, and Wojciech Matusik. 2016. Computational multicopter design. *ACM Trans. Graph.* 35, 6 (2016), 227:1–227:10.

D Economou, C Lee, C Mavroidis, and I Antoniadis. 2000. Robust vibration suppression in flexible payloads carried by robot manipulators using digital filtering of joint trajectories. In *Intl. Symposium on Robotics and Automation.* 244–249.

Nico Galoppo, Miguel A. Otaduy, William Moss, Jason Sewall, Sean Curtis, and Ming C. Lin. 2009. Controlling Deformable Material with Dynamic Morph Targets. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games (I3D '09).* ACM, New York, NY, USA, 39–47.

Damien Gauge, Stelian Coros, Sandro Mani, and Bernhard Thomaszewski. 2014. Interactive Design of Modular Tensegrity Characters. In *The Eurographics / ACM SIGGRAPH Symposium on Computer Animation, SCA 2014, Copenhagen, Denmark, 2014.* 131–138.

Moritz Geilinger, Roi Poranne, Ruta Desai, Bernhard Thomaszewski, and Stelian Coros. 2018. Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels. *ACM Trans. Graph.* 37, 4 (2018), 160.

Sehoon Ha, Stelian Coros, Alexander Alspach, Joohyung Kim, and Katsu Yamane. 2017. Joint optimization of robot design and motion parameters using the implicit function theorem. In *Robotics: Science and Systems.*

Kris K Hauser, Chen Shen, and James F O'Brien. 2003. Interactive Deformation Using Modal Analysis with Constraints.. In *Graphics Interface*, Vol. 3. 16–17.

Vladímir Villaverde Huertas and Boris Rohal'-Ilkiv. 2012. Vibration suppression of a flexible structure. *Procedia Engineering* 48 (2012), 233–241.

Doug L James and Dinesh K Pai. 2002. DyRT: dynamic response textures for real time deformation simulation with graphics hardware. *ACM Trans. Graph.* 21, 3, 582–585.

Junggon Kim and Nancy S Pollard. 2011. Fast simulation of skeleton-driven deformable body characters. *ACM Trans. Graph.* 30, 5 (2011), 121.

Theodore Kim and Doug L James. 2012. Physics-based character skinning using multidomain subspace deformations. *IEEE transactions on visualization and computer graphics* 18, 8 (2012), 1228–1240.

Seunghwan Lee, Ri Yu, Jungnam Park, Mridul Aanjaneya, Eftychios Sifakis, and Jehee Lee. 2018. Dexterous manipulation and control with volumetric muscles. *ACM Trans. Graph.* 37, 4 (2018), 57.

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.* 33, 4 (2014), 108.

Jacques Louis Lions. 1971. *Optimal control of systems governed by partial differential equations.* Vol. 170. Springer Berlin.

Libin Liu, KangKang Yin, Bin Wang, and Baining Guo. 2013. Simulation and control of skeleton-driven soft body characters. *ACM Trans. Graph.* 32, 6 (2013), 215.

Mickaël Ly, Romain Casati, Florence Bertails-Descoubes, Mélina Skouras, and Laurence Boissieux. 2018. Inverse Elastic Shell Design with Contact and Friction. *ACM Trans. Graph.* 37, 6, Article 201 (Dec. 2018), 16 pages. https://doi.org/10.1145/3272127.3275036

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

Vittorio Megaro, Bernhard Thomaszewski, Maurizio Nitti, Otmar Hilliges, Markus Gross, and Stelian Coros. 2015. Interactive design of 3D-printable robotic creatures. *ACM Trans. Graph.* 34, 6 (2015), 216.

Vittorio Megaro, Jonas Zehnder, Moritz Bächer, Stelian Coros, Markus Gross, and Bernhard Thomaszewski. 2017. A Computational Design Tool for Compliant Mechanisms. *ACM Trans. Graph.* 36, 4, Article 82 (July 2017), 12 pages.

Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2005. Meshless Deformations Based on Shape Matching. *ACM Trans. Graph.* 24, 3 (July 2005), 471–478.

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization, second edition.* World Scientific.

Carmine Maria Pappalardo and Domenico Guida. 2018. Use of the Adjoint Method for Controlling the Mechanical Vibrations of Nonlinear Systems. *Machines* 6, 2 (2018), 19.

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

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.

Christian Schulz, Christoph von Tycowicz, Hans-Peter Seidel, and Klaus Hildebrandt. 2014. Animating deformable objects using sparse spacetime constraints. *ACM Trans. Graph.* 33, 4 (2014), 109.

Tamar Shinar, Craig Schroeder, and Ronald Fedkiw. 2008. Two-way coupling of rigid and deformable bodies. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* Eurographics Association, 95–103.

Weiguang Si, Sung-Hee Lee, Eftychios Sifakis, and Demetri Terzopoulos. 2014. Realistic biomechanical simulation and control of human swimming. *ACM Trans. Graph.* 34, 1 (2014), 10.

Mélina Skouras, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, and Markus H. Gross. 2013. Computational design of actuated deformable characters. *ACM Trans. Graph.* 32, 4 (2013), 82:1–82:10.

Breannan Smith, Fernando de Goes, and Theodore Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Trans. Graph.* 37, 2 (2018), 12:1–12:15.

Peng Song, Xiaofei Wang, Xiao Tang, Chi-Wing Fu, Hongfei Xu, Ligang Liu, and Niloy J. Mitra. 2017. Computational design of wind-up toys. *ACM Trans. Graph.* 36, 6 (2017), 238:1–238:13.

Jie Tan, Yuting Gu, Greg Turk, and C. Karen Liu. 2011. Articulated swimming creatures. *ACM Trans. Graph.* 30, 4 (2011), 58:1–58:12.

Bernhard Thomaszewski, Stelian Coros, Damien Gauge, Vittorio Megaro, Eitan Grinspun, and Markus Gross. 2014. Computational Design of Linkage-based Characters. *ACM Trans. Graph.* 33, 4, Article 64 (July 2014), 9 pages.

Maxime Tournier, Matthieu Nesme, Benjamin Gilles, and François Faure. 2015. Stable constrained dynamics. *ACM Trans. Graph.* 34, 4 (2015), 132:1–132:10.

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

Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. 2014. Pteromys: interactive design and optimization of free-formed free-flight model airplanes. *ACM Trans. Graph.* 33, 4 (2014), 65:1–65:10.

Christoph von Tycowicz, Christian Schulz, Hans-Peter Seidel, and Klaus Hildebrandt. 2013. An efficient construction of reduced deformable objects. *ACM Trans. Graph.* 32, 6 (2013), 213.

Andrew Witkin and David Baraff. 1997. Physically Based Modeling: Principles and Practice. In *ACM SIGGRAPH 1997 Courses (SIGGRAPH '97).* New York, NY, USA.

Chris Wojtan, Peter J Mucha, and Greg Turk. 2006. Keyframe control of complex particle systems using the adjoint method. In *ACM SIGGRAPH/Eurographics symposium on Computer animation.* Eurographics Association, 15–23.

Hongyi Xu and Jernej Barbič. 2016. Pose-space subspace dynamics. *ACM Trans. Graph.* 35, 4 (2016), 35.

Hongyi Xu, Espen Knoop, Stelian Coros, and Moritz Bächer. 2018. Bend-it: Design and Fabrication of Kinetic Wire Characters. *ACM Trans. Graph.* 37, 6 (2018), 239:1–239:15.

Ran Zhang, Thomas Auzinger, Duygu Ceylan, Wilmot Li, and Bernd Bickel. 2017. Functionality-aware retargeting of mechanisms to 3D shapes. *ACM Trans. Graph.* 36, 4 (2017), 81:1–81:13.

Tong Zhou, Andrew A Goldenberg, and Jean W Zu. 2002. Modal force based input shaper for vibration suppression of flexible payloads. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, Vol. 3. IEEE, 2430–2435.

A MASS MATRICES AND INERTIAL FORCES

Below, we provide implementation-ready generalized mass matrices for our relative coordinate formulation. For a detailed derivation, we point the reader to our supplemental material (Sec. 1). In derivations of mass matrices, we make use of the observation that our interpolated displacements can be written as sums of columns of our $3 \times 3n$ basis matrix times a *single* entry u_k of our displacements $\mathbf{u} \in \mathbb{R}^{3n}$

$$\mathbf{u}(\mathbf{X}, t) = \mathbf{\Phi}(\mathbf{X})\mathbf{u}(t) = \sum_k \boldsymbol{\phi}_k(\mathbf{X}) u_k(t). \tag{28}$$

Our mass matrices are

$$\underline{\mathbf{M1}} = \int_\Omega \rho\, \mathbf{\Phi}^T [\mathbf{X}]_\times \, d\mathbf{X} \qquad \underline{\mathbf{M2}}(\mathbf{u}) = \sum_k \left(\int_\Omega \rho\, \mathbf{\Phi}^T [\boldsymbol{\phi}_k]_\times \, d\mathbf{X} \right) u_k$$
$$\underline{\mathbf{M3}} = \int_\Omega \rho\, \mathbf{\Phi}^T \, d\mathbf{X} \qquad \underline{\mathbf{M4}}_j = \int_\Omega \rho\, \mathbf{X} \boldsymbol{\phi}_j^T \, d\mathbf{X}$$
$$\underline{\mathbf{M5}}_j(\mathbf{u}) = \sum_k \left(\int_\Omega \rho\, \boldsymbol{\phi}_k \boldsymbol{\phi}_j^T \, d\mathbf{X} \right) u_k \qquad \underline{\mathbf{M6}} = \int_\Omega \rho\, \mathbf{\Phi}^T \mathbf{X} \, d\mathbf{X} \tag{29}$$

where we precompute the constant blocks in brackets if a sum is involved. Mass matrix $\underline{\mathbf{M}} = \int_\Omega \rho\, \mathbf{\Phi}^T \mathbf{\Phi}\, d\mathbf{X}$ is the same in absolute and relative coordinates. To compute matrices $\underline{\mathbf{M1}}_r$-$\underline{\mathbf{M6}}_r$ for our reduced formulation, we set the basis $\mathbf{\Phi}$ to the reduced basis $\mathbf{\Phi}\mathbf{U}_r$. j and k iterate over $\{1, \ldots, 3n\}$ for full, and $\{1, \ldots, r\}$ for reduced simulation. We use numerical integration to evaluate integrals.

Vibration-Minimizing Motion Retargeting for Robotic Characters—Supplementary Material

In our supplemental material, we derive the relative coordinate formulation of our equations of motion (Sec. 1), and derive the adjoint system that we need to compute analytical gradients for our retargeting optimization (Sec. 2). Additional validations and results are discussed in Sec. 3.

1 DERIVATION OF RELATIVE COORDINATE FORMULATION

The deformed configuration is

$$\mathbf{x}(\mathrm{X}, t) = \mathbf{R}(t)\,[\mathrm{X} + \mathbf{\Phi}(\mathrm{X})\mathbf{u}(t)] + \mathbf{c}(t). \tag{1}$$

To derive the corresponding velocity and acceleration, we drop the arguments $\mathbf{x} = \mathbf{R}\,(\mathrm{X} + \mathbf{\Phi u}) + \mathbf{c}$.

The velocity of the deformed configuration is

$$\begin{aligned}\dot{\mathbf{x}} &= \dot{\mathbf{R}}\,(\mathrm{X} + \mathbf{\Phi u}) + \mathbf{R\Phi}\dot{\mathbf{u}} + \dot{\mathbf{c}} \\ &= [\boldsymbol{\omega}]_\times\,\mathbf{R}\,(\mathrm{X} + \mathbf{\Phi u}) + \mathbf{R\Phi v} + \mathbf{w},\end{aligned} \tag{2}$$

and its acceleration

$$\begin{aligned}\ddot{\mathbf{x}} &= \ddot{\mathbf{R}}\,(\mathrm{X} + \mathbf{\Phi u}) + 2\dot{\mathbf{R}}\mathbf{\Phi}\dot{\mathbf{u}} + \mathbf{R\Phi}\ddot{\mathbf{u}} + \ddot{\mathbf{c}} \\ &= \left([\dot{\boldsymbol{\omega}}]_\times\,\mathbf{R} + [\boldsymbol{\omega}]_\times^2\,\mathbf{R}\right)(\mathrm{X} + \mathbf{\Phi u}) + 2\,[\boldsymbol{\omega}]_\times\,\mathbf{R\Phi v} + \mathbf{R\Phi}\dot{\mathbf{v}} + \dot{\mathbf{w}} \\ &= \mathbf{R\Phi}\dot{\mathbf{v}} + [\dot{\boldsymbol{\omega}}]_\times\,\mathbf{R}\,(\mathrm{X} + \mathbf{\Phi u}) + \dot{\mathbf{w}} + [\boldsymbol{\omega}]_\times^2\,\mathbf{R}\,(\mathrm{X} + \mathbf{\Phi u}) + 2\,[\boldsymbol{\omega}]_\times\,\mathbf{R\Phi v} \\ &= \underbrace{\mathbf{R\Phi}\dot{\mathbf{v}}}_{\textit{db acc.}}\ \underbrace{-\mathbf{R}\,([\mathrm{X}]_\times + [\mathbf{\Phi u}]_\times)\,\mathbf{R}^T\dot{\boldsymbol{\omega}} + \dot{\mathbf{w}}}_{\textit{rb acc.}} \\ &\quad + \underbrace{\mathbf{R}[\mathbf{R}^T\boldsymbol{\omega}]_\times^2\,(\mathrm{X} + \mathbf{\Phi u})}_{\textit{centrifugal acc.}} + \underbrace{2\mathbf{R}[\mathbf{R}^T\boldsymbol{\omega}]_\times\mathbf{\Phi v}}_{\textit{Coriolis acc.}}.\end{aligned} \tag{3}$$

In above derivations, we make use of identities $\dot{\mathbf{R}} = [\boldsymbol{\omega}]_\times\,\mathbf{R}$ and $\ddot{\mathbf{R}} = [\dot{\boldsymbol{\omega}}]_\times\,\mathbf{R} + [\boldsymbol{\omega}]_\times^2\,\mathbf{R}$ in the first three lines. Because $[\mathbf{R}^T\mathbf{a}]_\times\mathbf{b} = \mathbf{R}^T\mathbf{a} \times \mathbf{R}^T\mathbf{R}\mathbf{b} = \mathbf{R}^T[\mathbf{a}]_\times\mathbf{R}\mathbf{b}$, the identity $\mathbf{R}^T[\mathbf{a}]_\times\mathbf{R} = [\mathbf{R}^T\mathbf{a}]_\times$ holds. For acceleration terms that have subexpressions $[\mathbf{a}]_\times\mathbf{R}\mathbf{b}$, we make use of the latter identity to move the rotation matrix to the left $\mathbf{R}\mathbf{R}^T[\mathbf{a}]_\times\mathbf{R}\mathbf{b} = \mathbf{R}[\mathbf{R}^T\mathbf{a}]_\times\mathbf{b}$. For the rigid body acceleration term, we further make use of identities $[\mathbf{a}]_\times\mathbf{b} = -[\mathbf{b}]_\times\mathbf{a}$ and $[\mathbf{a} + \mathbf{b}]_\times = [\mathbf{a}]_\times + [\mathbf{b}]_\times$.

To form the inertial forces,

$$\int_\Omega \mathbf{\Phi}^T\mathbf{R}^T\rho\,\ddot{\mathbf{x}}\,\mathrm{dX} = \int_\Omega \rho\,\mathbf{\Phi}^T\mathbf{R}^T\ddot{\mathbf{x}}\,\mathrm{dX},$$

we integrate the individual acceleration terms. When integrating the acceleration of the deformable body (*db acc.*), we see why it is useful to multiply with the transpose of the rigid body rotation

$$\int_\Omega \rho\,\mathbf{\Phi}^T\mathbf{R}^T\mathbf{R\Phi}\dot{\mathbf{v}}\,\mathrm{dX} = \left(\int_\Omega \rho\,\mathbf{\Phi}^T\mathbf{\Phi}\,\mathrm{dX}\right)\dot{\mathbf{v}} = \underline{\mathbf{M}}\dot{\mathbf{v}}. \tag{4}$$

Integration results in the same mass matrix as for the absolute coordinate formulation ($\underline{\mathbf{M}} \in \mathbb{R}^{3n\times 3n}$).

Integration of the rigid body acceleration (*rb acc.*) leads to a force

$$-\left(\underline{\mathbf{M1}} + \underline{\mathbf{M2}}(\mathbf{u})\right)\mathbf{R}^T\dot{\boldsymbol{\omega}} + \underline{\mathbf{M3}}\,\mathbf{R}^T\dot{\mathbf{w}} \tag{5}$$

that depends on a total of three mass matrices

$$\underline{\mathbf{M1}} = \int_\Omega \rho\,\mathbf{\Phi}^T[\mathrm{X}]_\times\,\mathrm{dX} \qquad \underline{\mathbf{M2}}(\mathbf{u}) = \int_\Omega \rho\,\mathbf{\Phi}^T[\mathbf{\Phi u}]_\times\,\mathrm{dX}$$

$$\underline{\mathbf{M3}} = \int_\Omega \rho\,\mathbf{\Phi}^T\,\mathrm{dX}$$

where $\underline{\mathbf{M1}} \in \mathbb{R}^{3n\times 3}$ and $\underline{\mathbf{M3}} \in \mathbb{R}^{3n\times 3}$ are constant and $\underline{\mathbf{M2}} \in \mathbb{R}^{3n\times 3}$ depends on the displacement. To efficiently compute $\underline{\mathbf{M2}}$, we substitute $\sum_{k=1}^{3n}\boldsymbol{\phi}_k u_k$ for the displacement $\mathbf{\Phi u}$

$$\underline{\mathbf{M2}}(\mathbf{u}) = \sum_{k=1}^{3n}\left(\int_\Omega \rho\,\mathbf{\Phi}^T[\boldsymbol{\phi}_k]_\times\,\mathrm{dX}\right)u_k \tag{6}$$

and precompute the $3n \times 3$ blocks in brackets.

To derive the centrifugal forces, we first split the corresponding acceleration (*centrifugal acc.*) into two terms

$$\mathbf{R}\left((\mathbf{R}^T\boldsymbol{\omega})\cdot(\mathrm{X} + \mathbf{\Phi u})\right)\mathbf{R}^T\boldsymbol{\omega} - \mathbf{R}(\mathrm{X} + \mathbf{\Phi u})\,(\boldsymbol{\omega}\cdot\boldsymbol{\omega}) \tag{7}$$

where we apply the identity $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a}\cdot\mathbf{c})\mathbf{b} - \mathbf{c}(\mathbf{a}\cdot\mathbf{b})$ to the subexpression $[\mathbf{R}^T\boldsymbol{\omega}]_\times^2(\mathrm{X} + \mathbf{\Phi u}) = (\mathbf{R}^T\boldsymbol{\omega}) \times (\mathbf{R}^T\boldsymbol{\omega}) \times (\mathrm{X} + \mathbf{\Phi u})$.

To integrate the first term, we split the integral into a sum of integrals and look at the transpose of the j-th column of $\mathbf{\Phi}$

$$\begin{aligned}&\sum_{k=1}^{3n}\int_\Omega \rho\,\boldsymbol{\phi}_j^T\left((\mathbf{R}^T\boldsymbol{\omega})\cdot(\mathrm{X} + \boldsymbol{\phi}_k u_k)\right)\mathbf{R}^T\boldsymbol{\omega}\,\mathrm{dX} \\ &= \boldsymbol{\omega}^T\mathbf{R}\left(\sum_{k=1}^{3n}\int_\Omega \rho\,(\mathrm{X} + \boldsymbol{\phi}_k u_k)\boldsymbol{\phi}_j^T\,\mathrm{dX}\right)\mathbf{R}^T\boldsymbol{\omega} \\ &= \boldsymbol{\omega}^T\mathbf{R}\left(\left(\int_\Omega \rho\,\mathrm{X}\boldsymbol{\phi}_j^T\,\mathrm{dX}\right) + \sum_{k=1}^{3n}\left(\int_\Omega \rho\,\boldsymbol{\phi}_k\boldsymbol{\phi}_j^T\,\mathrm{dX}\right)u_k\right)\mathbf{R}^T\boldsymbol{\omega}.\end{aligned} \tag{8}$$

The derivation of the second term is straightforward.

To form the resulting centrifugal forces

$$\begin{aligned}\mathbf{f}_{\mathrm{cen}}(\mathbf{q}, \mathbf{u}, \boldsymbol{\omega}) = &\sum_{j=1}^{3n}\boldsymbol{\omega}^T\mathbf{R}\left(\underline{\mathbf{M4}}_j + \underline{\mathbf{M5}}_j(\mathbf{u})\right)\mathbf{R}^T\boldsymbol{\omega}\,\mathbf{e}_j \\ &- \left(\underline{\mathbf{M6}} + \underline{\mathbf{M}}\mathbf{u}\right)(\boldsymbol{\omega}\cdot\boldsymbol{\omega})\end{aligned} \tag{9}$$

where $\mathbf{e}_j$ is the j-th column of the $3n \times 3n$ identity matrix, we precompute the additional mass matrices

$$\underline{\mathbf{M4}}_j = \int_\Omega \rho\,\mathrm{X}\boldsymbol{\phi}_j^T\,\mathrm{dX} \qquad \underline{\mathbf{M5}}_j(\mathbf{u}) = \sum_{k=1}^{3n}\left(\int_\Omega \rho\,\boldsymbol{\phi}_k\boldsymbol{\phi}_j^T\,\mathrm{dX}\right)u_k$$

$$\underline{\mathbf{M6}} = \int_\Omega \rho\,\mathbf{\Phi}^T\mathrm{X}\,\mathrm{dX}.$$

Analogously to $\underline{\mathbf{M2}}$, we precompute the blocks in brackets for matrix $\underline{\mathbf{M5}}_j$. Because $\mathbf{\Phi}$ has $3n$ columns, there are $3n$ 3×3 matrices $\underline{\mathbf{M4}}_j$ and $\underline{\mathbf{M5}}_j(\mathbf{u})$ ($j = 1, \ldots, 3n$). $\underline{\mathbf{M6}}$ is a $3n$ *vector.*

To derive the Coriolis force

$$\mathbf{f}_{\mathrm{cor}}(\mathbf{q}, \boldsymbol{\omega}, \mathbf{v}) = -2\underline{\mathbf{M2}}(\mathbf{v})\,\mathbf{R}^T\boldsymbol{\omega}, \tag{10}$$

we substitute $[\mathbf{R}^T\boldsymbol{\omega}]_\times$ for $\mathbf{a}$ and $\boldsymbol{\Phi}\mathbf{v}$ for $\mathbf{b}$ in the identity $[\mathbf{a}]_\times\mathbf{b} = -[\mathbf{b}]_\times\mathbf{a}$ in the Coriolis term (*Coriolis acc.*) prior to integration. For this term, we reuse $\underline{\mathbf{M2}}$, setting its parameter to the velocities $\mathbf{v}$ instead of the displacements $\mathbf{u}$.

Note that the fictitious centrifugal and Coriolis forces depend on $\mathbf{q}$ because we use quaternions instead of rotations. In our implementation, we extract rotations from quaternions $\mathbf{R}(\mathbf{q})$.

1.1 Reduced Basis

In a reduced formulation, the displacements are defined as

$$\mathbf{u}(\mathbf{X}, t) = \boldsymbol{\Phi}(\mathbf{X})\mathbf{U}_\mathrm{r}\mathbf{u}_\mathrm{r}(t). \tag{11}$$

To derive the reduced mass matrices and inertial forces, we can replace the full basis $\boldsymbol{\Phi}(\mathbf{X})$ with the reduced basis $\boldsymbol{\Phi}_\mathrm{r}(\mathbf{X}) = \boldsymbol{\Phi}(\mathbf{X})\mathbf{U}_\mathrm{r}$ in above derivations. We again drop arguments. $\boldsymbol{\Phi}_\mathrm{r}$ is now a $3 \times r$-matrix, $\mathbf{u}_\mathrm{r}$ a r-vector, and $\boldsymbol{\phi}_{r,k} \in \mathbb{R}^3$ the k-th column of $\boldsymbol{\Phi}_\mathrm{r}$.

The mass matrices are

$$\underline{\mathbf{M}}_\mathrm{r} = \mathbf{U}_\mathrm{r}^T \underline{\mathbf{M}}\mathbf{U}_\mathrm{r} = \mathbf{E}_{r\times r} \tag{12}$$

$$\underline{\mathbf{M1}}_\mathrm{r} = \mathbf{U}_\mathrm{r}^T \underline{\mathbf{M1}} \tag{13}$$

$$\underline{\mathbf{M2}}_\mathrm{r}(\mathbf{u}_\mathrm{r}) = \left(\sum_{k=1}^{r}\left(\int_\Omega \rho\, \boldsymbol{\Phi}_\mathrm{r}^T [\boldsymbol{\phi}_{\mathrm{r},k}]_\times \,\mathrm{d}\mathbf{X}\right) u_{\mathrm{r},k}\right) \tag{14}$$

$$\underline{\mathbf{M3}}_\mathrm{r} = \mathbf{U}_\mathrm{r}^T \underline{\mathbf{M3}} \tag{15}$$

$$\underline{\mathbf{M4}}_{\mathrm{r},j} = \int_\Omega \rho\, \mathbf{X}\boldsymbol{\phi}_{\mathrm{r},j}^T \,\mathrm{d}\mathbf{X} \tag{16}$$

$$\underline{\mathbf{M5}}_{\mathrm{r},j}(\mathbf{u}_\mathrm{r}) = \sum_{k=1}^{r}\left(\int_\Omega \rho\, \boldsymbol{\phi}_{\mathrm{r},k}\boldsymbol{\phi}_{\mathrm{r},j}^T \,\mathrm{d}\mathbf{X}\right) u_{\mathrm{r},k} \tag{17}$$

$$\underline{\mathbf{M6}}_\mathrm{r} = \mathbf{U}_\mathrm{r}^T \underline{\mathbf{M6}} \tag{18}$$

where we use numerical integration to precompute the blocks in brackets for $\underline{\mathbf{M2}}_\mathrm{r}(\mathbf{u}_\mathrm{r})$ and $\underline{\mathbf{M5}}_{\mathrm{r},j}(\mathbf{u}_\mathrm{r})$, and the j-th 3×3 matrices $\underline{\mathbf{M4}}_{\mathrm{r},j}$.

The inertial forces corresponding to the deformable body (*db acc.*) reduce to

$$\mathbf{U}_\mathrm{r}^T \underline{\mathbf{M}}\mathbf{U}_\mathrm{r}\dot{\mathbf{v}}_\mathrm{r} = \dot{\mathbf{v}}_\mathrm{r}, \tag{19}$$

and for the rigid body (*rb acc.*) to

$$-\left(\underline{\mathbf{M1}}_\mathrm{r} + \underline{\mathbf{M2}}_\mathrm{r}(\mathbf{u}_\mathrm{r})\right)\mathbf{R}^T\dot{\boldsymbol{\omega}} + \underline{\mathbf{M3}}_\mathrm{r}\mathbf{R}^T\dot{\mathbf{w}}. \tag{20}$$

The centrifugal force becomes

$$\begin{aligned}\mathbf{f}_\mathrm{cen}(\mathbf{q}, \mathbf{u}_\mathrm{r}, \boldsymbol{\omega}) = &\sum_{j=1}^{r} \boldsymbol{\omega}^T\mathbf{R}\left(\underline{\mathbf{M4}}_{\mathrm{r},j} + \underline{\mathbf{M5}}_{\mathrm{r},j}(\mathbf{u})\right)\mathbf{R}^T\boldsymbol{\omega}\,\mathbf{e}_j \\ &- \left(\underline{\mathbf{M6}} + \underline{\mathbf{M}}\mathbf{u}\right)(\boldsymbol{\omega}\cdot\boldsymbol{\omega})\end{aligned} \tag{21}$$

where $\mathbf{e}_j$ is the j-th column of the $r \times r$ identity matrix, and the reduced Coriolis force is

$$\mathbf{f}_\mathrm{cor}(\mathbf{q}, \boldsymbol{\omega}, \mathbf{v}_\mathrm{r}) = -2\underline{\mathbf{M2}}_\mathrm{r}(\mathbf{v}_\mathrm{r})\mathbf{R}^T\boldsymbol{\omega}. \tag{22}$$

2 DERIVATION OF THE ADJOINT SYSTEM

To solve our retargeting problem

$$\min_{\mathbf{p}} G(\mathbf{p}, \mathbf{U}(\mathbf{p})) + R(\mathbf{p}) \tag{23}$$

$$\text{subject to} \quad \mathbf{G}(t, \mathbf{p}, \mathbf{S}(t, \mathbf{p}), \dot{\mathbf{S}}(t, \mathbf{p})) = 0 \quad \text{and} \quad \mathbf{S}(0) = \mathbf{S}_0(\mathbf{p}),$$

we need to be able to compute an analytical gradient $\frac{\mathrm{d}G(\mathbf{p},\mathbf{U}(\mathbf{p}))}{\mathrm{d}\mathbf{p}}$.

We largely follow the derivation in the work of Cao et al. [2003], omitting terms that are not relevant in our context. While our semi-implicit DAE system can easily be brought into standard Hessenberg index-2 form, we prefer to keep the mass matrix $\mathbf{M}$ on the left-hand side because it simplifies the adjoint DAE.

To keep the derivation general, we assume our objective to depend on the state $\mathbf{S}$ for the first part of the derivation

$$G(\mathbf{p}, \mathbf{S}(\mathbf{p})) = \int_0^T g(t, \mathbf{p}, \mathbf{S}(\mathbf{p}))\,\mathrm{d}t. \tag{24}$$

The augmented objective with *continuous, time-dependent* Lagrange multipliers $\boldsymbol{\lambda}(t)$ for above problem is

$$I(\mathbf{p}, \mathbf{S}) = G(\mathbf{p}, \mathbf{S}) - \int_0^T \boldsymbol{\lambda}^T\mathbf{G}(t, \mathbf{p}, \mathbf{S}, \dot{\mathbf{S}})\,\mathrm{d}t. \tag{25}$$

Because our DAE system is satisfied at every $t \in [0, T]$, the total derivative of G and I are equivalent, hence

$$\frac{\mathrm{d}G}{\mathrm{d}\mathbf{p}} = \int_0^T (g_\mathbf{p} + g_\mathbf{S}\mathbf{S}_\mathbf{p})\,\mathrm{d}t - \int_0^T \boldsymbol{\lambda}^T(\mathbf{G}_\mathbf{p} + \mathbf{G}_\mathbf{S}\mathbf{S}_\mathbf{p} + \mathbf{G}_{\dot{\mathbf{S}}}\dot{\mathbf{S}}_\mathbf{p})\,\mathrm{d}t \tag{26}$$

where we use subscripts for partial derivatives. Note that $\mathbf{S}_\mathbf{p}$ and $\dot{\mathbf{S}}_\mathbf{p}$ are total derivatives.

Integration by parts of the term $\boldsymbol{\lambda}^T\mathbf{G}_{\dot{\mathbf{S}}}\dot{\mathbf{S}}_\mathbf{p}$

$$\int_0^T \boldsymbol{\lambda}^T\mathbf{G}_{\dot{\mathbf{S}}}\dot{\mathbf{S}}_\mathbf{p}\,\mathrm{d}t = \left(\boldsymbol{\lambda}^T\mathbf{G}_{\dot{\mathbf{S}}}\mathbf{S}_\mathbf{p}\right)\Big|_0^T - \int_0^T \frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{\lambda}^T\mathbf{G}_{\dot{\mathbf{S}}}\right)\mathbf{S}_\mathbf{p}\,\mathrm{d}t \tag{27}$$

enables us to turn the dependence on $\dot{\mathbf{S}}_\mathbf{p}$ into a dependence on $\mathbf{S}_\mathbf{p}$

$$\begin{aligned}\frac{\mathrm{d}G}{\mathrm{d}\mathbf{p}} = &\int_0^T (g_\mathbf{p} - \boldsymbol{\lambda}^T\mathbf{G}_\mathbf{p})\,\mathrm{d}t - \int_0^T \left(-g_\mathbf{S} + \boldsymbol{\lambda}^T\mathbf{G}_\mathbf{S} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{\lambda}^T\mathbf{G}_{\dot{\mathbf{S}}}\right)\right)\mathbf{S}_\mathbf{p}\,\mathrm{d}t \\ &- \left(\boldsymbol{\lambda}^T\mathbf{G}_{\dot{\mathbf{S}}}\mathbf{S}_\mathbf{p}\right)\Big|_0^T.\end{aligned} \tag{28}$$

By setting the term in brackets of the second integrand to zero, we then form the adjoint system

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{\lambda}^T\mathbf{G}_{\dot{\mathbf{S}}}\right) = \boldsymbol{\lambda}^T\mathbf{G}_\mathbf{S} + g_\mathbf{S}. \tag{29}$$

Applying the chain rule and transposing the system, we form the adjoint DAE

$$\mathbf{G}_{\dot{\mathbf{S}}}^T\dot{\boldsymbol{\lambda}} = \left(-\left(\frac{\mathrm{d}\mathbf{G}_{\dot{\mathbf{S}}}}{\mathrm{d}t}\right)^T + \mathbf{G}_\mathbf{S}^T\right)\boldsymbol{\lambda} + g_\mathbf{S}^T. \tag{30}$$

For our particular DAE system

$$\mathbf{G} = \begin{bmatrix} \dot{\mathbf{U}} - \mathbf{T}(\mathbf{U})\mathbf{V} \\ \mathbf{M}(\mathbf{U})\dot{\mathbf{V}} - \mathbf{F}(\mathbf{U}, \mathbf{V}) - (C_\mathbf{U}(t, \mathbf{U})\mathbf{T}(\mathbf{U}))^T\boldsymbol{\Lambda} \\ C_t(t, \mathbf{U}) + C_\mathbf{U}(t, \mathbf{U})\mathbf{T}(\mathbf{U})\mathbf{V} + \alpha C(t, \mathbf{U}) \end{bmatrix} = 0, \tag{31}$$

the Jacobian w.r.t. the state is

$$\mathbf{G}_\mathbf{S} = \begin{bmatrix} \mathbf{G}_\mathbf{U}(t, \mathbf{U}, \mathbf{V}) & \mathbf{G}_\mathbf{V}(t, \mathbf{U}, \mathbf{V}) & \mathbf{G}_\boldsymbol{\Lambda}(t, \mathbf{U}) \end{bmatrix} \tag{32}$$

with columns

$$\mathbf{G}_\mathbf{U} = \begin{bmatrix} -\mathbf{T}_\mathbf{U}(\mathbf{U})\mathbf{V} \\ \mathbf{M}_\mathbf{U}(\mathbf{U})\dot{\mathbf{V}} - \mathbf{F}_\mathbf{U}(\mathbf{U},\mathbf{V}) - \left(C_{\mathbf{U},\mathbf{U}}(t,\mathbf{U})\mathbf{T}(\mathbf{U}) + C_\mathbf{U}(t,\mathbf{U})\mathbf{T}_\mathbf{U}(\mathbf{U})\right)^T \\ C_{t,\mathbf{U}}(t,\mathbf{U}) + \left(C_{\mathbf{U},\mathbf{U}}(t,\mathbf{U})\mathbf{T}(\mathbf{U}) + C_\mathbf{U}(t,\mathbf{U})\mathbf{T}_\mathbf{U}(\mathbf{U})\right)\mathbf{V} + \alpha C(t,\mathbf{U}) \end{bmatrix},$$

$$\mathbf{G}_\mathbf{V} = \begin{bmatrix} -\mathbf{T}(\mathbf{U}) \\ -\mathbf{F}_\mathbf{V}(\mathbf{U},\mathbf{V}) \\ C_\mathbf{U}(t,\mathbf{U})\mathbf{T}(\mathbf{U}) \end{bmatrix}, \quad \text{and} \quad \mathbf{G}_\Lambda = \begin{bmatrix} \\ -\left(C_\mathbf{U}(t,\mathbf{U})\mathbf{T}(\mathbf{U})\right)^T \\ \\ \end{bmatrix}$$

where $C_{t,\mathbf{U}} = \frac{\partial^2 C}{\partial t \partial \mathbf{U}}$ and $C_{\mathbf{U},\mathbf{U}} = \frac{\partial^2 C}{\partial \mathbf{U}^2}$.

The Jacobian w.r.t. the time-derivative of the state is

$$\mathbf{G}_{\dot{\mathbf{S}}} = \begin{bmatrix} \mathbf{E} & & \\ & \mathbf{M}(\mathbf{U}) & \\ & & \end{bmatrix}. \tag{33}$$

And its time-derivative is

$$\frac{\mathrm{d}\mathbf{G}_{\dot{\mathbf{S}}}}{\mathrm{d}t} = \begin{bmatrix} & & \\ & \mathbf{M}_\mathbf{U}(\mathbf{U})\dot{\mathbf{U}} & \\ & & \end{bmatrix}, \tag{34}$$

where only the "center" element of the 3-by-3 block matrix is non-zero.

For our particular system, g only depends on generalized positions and velocities

$$g_\mathbf{S} = \begin{bmatrix} g_\mathbf{U} & g_\mathbf{V} & \end{bmatrix}. \tag{35}$$

In summary, our *linear adjoint DAE* is

$$\begin{bmatrix} \mathbf{E} & & \\ & \mathbf{M}^T & \\ & & \end{bmatrix} \dot{\boldsymbol{\lambda}} = \left(- \begin{bmatrix} & & \\ & \dot{\mathbf{U}}^T\mathbf{M}_\mathbf{U}^T & \\ & & \end{bmatrix} + \begin{bmatrix} \mathbf{G}_\mathbf{U}^T \\ \mathbf{G}_\mathbf{V}^T \\ \mathbf{G}_\Lambda^T \end{bmatrix} \right) \boldsymbol{\lambda} + \begin{bmatrix} g_\mathbf{U}^T \\ g_\mathbf{V}^T \\ \end{bmatrix}.$$

It remains to discuss initial conditions. To evaluate the gradient $\frac{\mathrm{d}G}{\mathrm{d}\mathbf{p}}$, we need to know

$$\left(\boldsymbol{\lambda}^T \mathbf{G}_{\dot{\mathbf{S}}} \mathbf{S}_\mathbf{p}\right)\Big|_{t=0} \quad \text{and} \quad \left(\boldsymbol{\lambda}^T \mathbf{G}_{\dot{\mathbf{S}}} \mathbf{S}_\mathbf{p}\right)\Big|_{t=T}. \tag{36}$$

At time $t = 0$, the Jacobian $\mathbf{S}_\mathbf{p}$ is equal to the analytical derivative $\frac{\mathrm{d}\mathbf{S}_0}{\mathrm{d}\mathbf{p}}$ of our initial conditions $\mathbf{S}_0$. If we set the *initial conditions* for our adjoint DAE to be $\boldsymbol{\lambda}(T) = 0$, then

$$\left(\boldsymbol{\lambda}^T \mathbf{G}_{\dot{\mathbf{S}}} \mathbf{S}_\mathbf{p}\right)\Big|_{t=T} = 0. \tag{37}$$

Note that the initial conditions $\mathbf{S}_0$ for our DAE are *not* dependent on $\mathbf{p}$ because we assume the system to be at rest at the start of an animation. Hence, $\mathbf{S}_\mathbf{p}$ is zero at time $t = 0$, and therefore

$$\left(\boldsymbol{\lambda}^T \mathbf{G}_{\dot{\mathbf{S}}} \mathbf{S}_\mathbf{p}\right)\Big|_{t=0} = 0. \tag{38}$$

If objective g depends on the algebraic variables Λ, the initial conditions for our adjoint DAE, $\boldsymbol{\lambda}(T) = 0$, are in conflict with the adjoint DAE, and an additional treatment is necessary [Cao et al. 2003].

In summary, the *analytical gradient* of our objective is

$$\frac{\mathrm{d}G}{\mathrm{d}\mathbf{p}} = \int_0^T \left(g_\mathbf{p} - \boldsymbol{\lambda}^T \mathbf{G}_\mathbf{p}\right) \mathrm{d}t \tag{39}$$

where the *adjoint variables* $\boldsymbol{\lambda}(t)$ are computed by solving the *linear adjoint DAE*

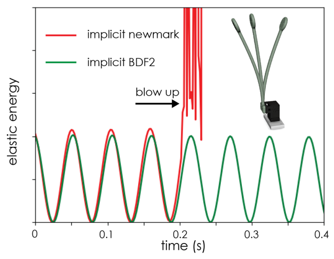
$$\begin{bmatrix} \mathbf{E} & & \\ & \mathbf{M}^T & \\ & & \end{bmatrix} \dot{\boldsymbol{\lambda}} = \left(- \begin{bmatrix} & & \\ & \dot{\mathbf{U}}^T\mathbf{M}_\mathbf{U}^T & \\ & & \end{bmatrix} + \begin{bmatrix} \mathbf{G}_\mathbf{U}^T \\ \mathbf{G}_\mathbf{V}^T \\ \mathbf{G}_\Lambda^T \end{bmatrix} \right) \boldsymbol{\lambda} + \begin{bmatrix} g_\mathbf{U}^T \\ g_\mathbf{V}^T \\ \end{bmatrix}$$

with *initial conditions*

$$\boldsymbol{\lambda}(T) = 0. \tag{40}$$

3 RESULTS

Time Integration. We experimented with explicit 4th-order Runge-Kutta (RK4), implicit Newmark, and implicit second-order backward Euler for simulating our nonlinear and stiff materials, as we illustrate in the inset. Compared to explicit RK4, which blows up immediately even if we set the time step to a 10,000× smaller value, both implicit integrators behave more stably. However, when using the simulation timestep we use to generate our results, implicit Newmark still blows up while implict BDF2 simulates stably and introduces negligible numerical damping.



Motor Profiles and Error Visualization. In Figs. 1, 2, 3, 4 and 5, we show the input and optimized motor profiles for our demonstrations, as well as error plots. We note that with the optimized motor profiles, we can efficaciously suppress visible vibrations in *all* our demonstrations. We also note that there is no clear pattern in the adjustment of motor profiles to minimize vibrations, suggesting that it would be nearly impossible to achieve similar results by manually tuning the motor profiles.

REFERENCES

Yang Cao, Shengtai Li, Linda Petzold, and Radu Serban. 2003. Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution. *SIAM Journal on Scientific Computing* 24, 3 (2003), 1076–1089.

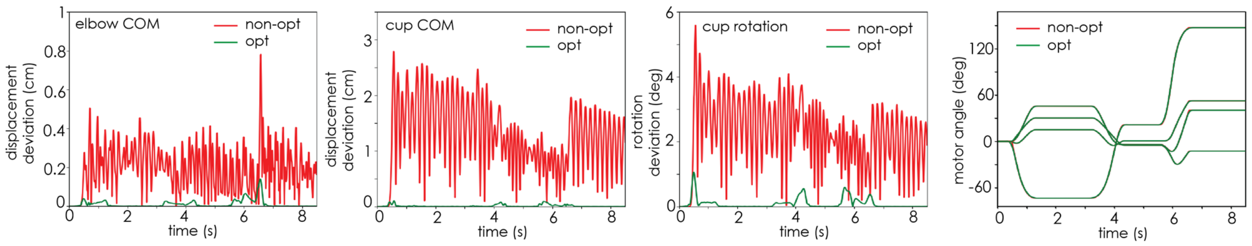


Fig. 1. **Bartender error plots and motor profiles.** We show the position errors at the elbow and the end effector, i.e., cup, as well as the rotation error for the cup. Minor changes to the motor signal suffice to suppress visible vibrations (right).

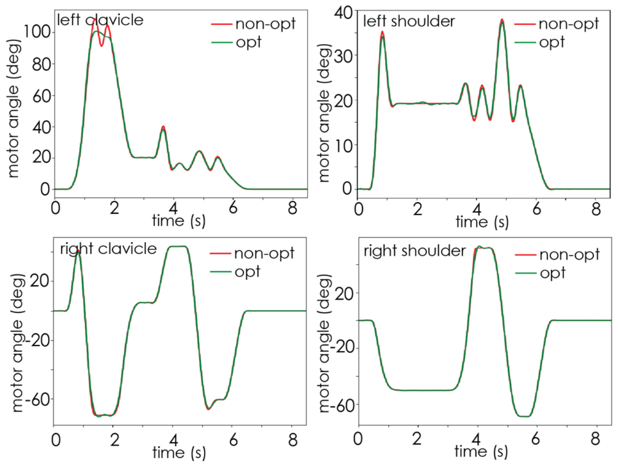


Fig. 2. **Dancing robot motor profiles.** We plot the non-optimized and optimized (with uniform weights) motor profiles for our 4-DOF dancing character. As we can observe from the top left figure, the optimization smoothens the waving motion which adds significant energy to the system.

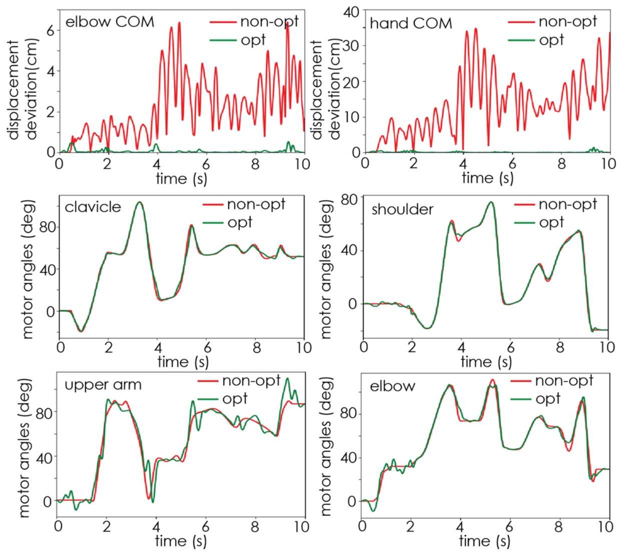


Fig. 3. **Rapper arm error plots and motor profiles.** The first row shows difference in target and simulated trajectories for the elbow and the hand. With our optimized motor profiles (bottom two rows), we suppress the vibration from 35 cm deviation to < 1 cm. Note that the upper arm motor and the elbow motor are coupled in a 4-bar linkage and are actively canceling the vibration via adding higher-frequency motor profiles.

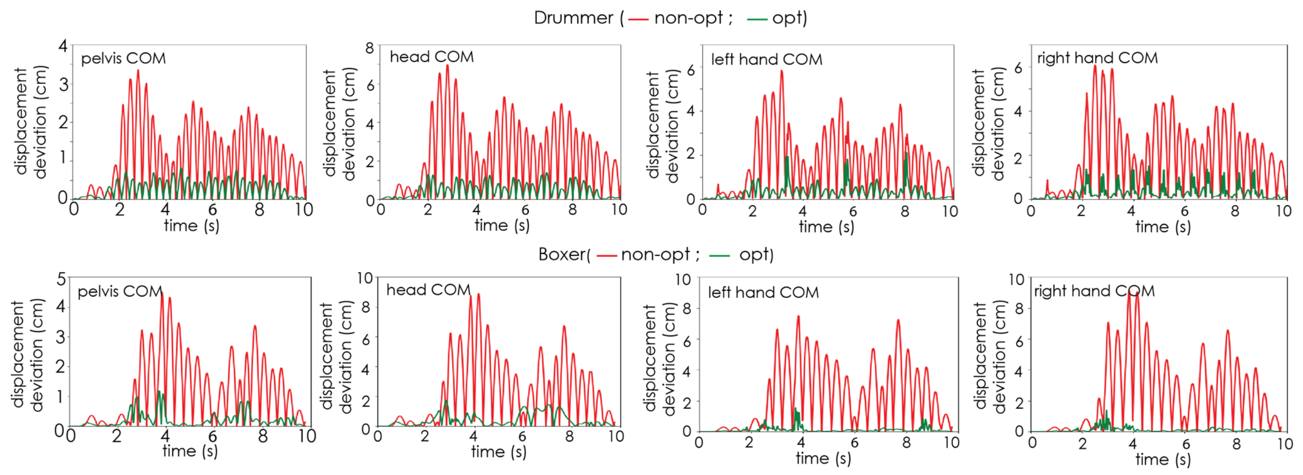


Fig. 4. **Drummer and Boxer error plots.** We show the error plots for our Drummer (top row) and Boxer (bottom row), respectively. Our optimization results significantly reduce deviations from the target.

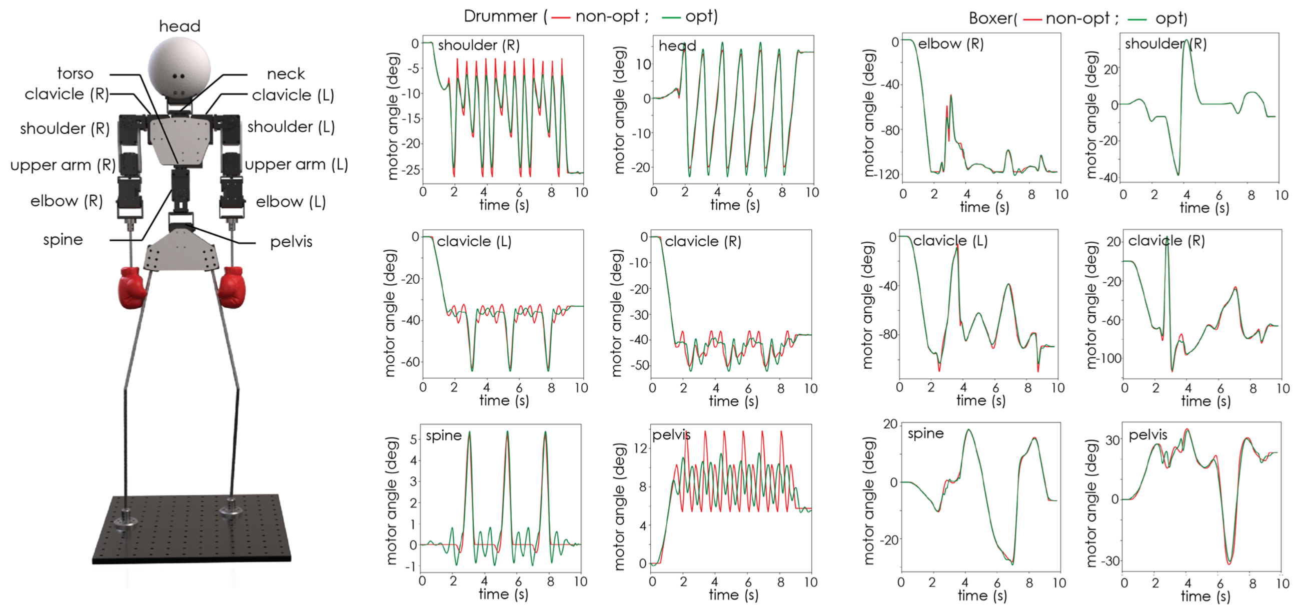


Fig. 5. **Drummer and Boxer motor profiles.** Our 13-DOF full-body character is shown on the left. Motor profiles of 6 representative motors for the drumming (plots in first two columns) and boxing (plots in third and forth columns) animations are shown on the right. Vibrations are suppressed by smoothing some of the motor signals (such as the shoulder (R) and pelvis for drumming) while compensating with motion from other motors (such as head and spine motor for the drumming sequence). Some of them are exactly identical to the input, e.g., the shoulder (R) for boxing.