

DiffGIPC: Differentiable Mixed-Coordinate Contact Dynamics on GPU

KEMENG HUANG, The University of Hong Kong, Hong Kong SAR, China

BOWEN REN, The University of Hong Kong, Hong Kong SAR, China

XINYU LU, TransGP, Hong Kong SAR, China

MINCHEN LI, Carnegie Mellon University, USA

TAKU KOMURA, The University of Hong Kong, Hong Kong SAR, China



Fig. 1. Optimizing the placement of seven affine bodies. Keeping their shapes fixed, DiffGIPC adjusts only the horizontal initial positions to form a stable stack. The panels show the unoptimized configuration and the simulated results after 10, 50, and 78 iterations.

Contact-rich inverse problems require efficient gradients and reliable enforcement of non-penetration. Existing differentiable methods provide fast simulation and rigid-deformable coupling, but contact treatments based on penalties or velocity-level constraints can permit penetration or positional drift. Incremental Potential Contact (IPC) maintains non-penetration through geometric barriers and continuous collision detection (CCD). However, repeated forward and backward evaluations remain costly in full-space finite-element (FEM) implementations, particularly when nearly rigid components retain many vertex degrees of freedom (DoFs). We present DiffGIPC, a GPU-accelerated differentiable IPC framework for optimizing interacting deformable and nearly rigid bodies. Our affine-body derivatives enable rest-shape design, initial-state optimization, and articulated control while retaining only twelve motion DoFs per affine body. These gradients account for how changes in geometry, placement, and actuation affect other objects through collision and friction. We preserve these interactions in one coupled FEM-affine adjoint system per frame, reducing the backward-system size without discarding the dependencies that drive the inverse problem. GPU-optimized backward computation further reduces the cost of repeated gradient evaluations. A reusable PyTorch interface connects supported physical parameters to user-defined parameter maps and losses across design, identification, motion, placement, and articulated control. Experiments demonstrate peak speedups over DiffIPC of approximately $150.0\times$

Authors' Contact Information: Kemeng Huang, kmhuang@connect.hku.hk, kmhuang819@gmail.com, The University of Hong Kong, Hong Kong SAR, China; Bowen Ren, boweyna@gmail.com, The University of Hong Kong, Hong Kong SAR, China; Xinyu Lu, lxy819469559@gmail.com, TransGP, Hong Kong SAR, China; Minchen Li, minchernl@gmail.com, Carnegie Mellon University, Pittsburgh, USA; Taku Komura, taku@cs.hku.hk, The University of Hong Kong, Hong Kong SAR, China.

with full-space FEM and $4133.3\times$ with affine bodies for combined forward simulation and backward differentiation.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; *Parallel algorithms*.

Additional Key Words and Phrases: differentiable simulation, contact, affine body dynamics, finite elements, GPU simulation, articulated control, inverse design

1 Introduction

Gradient-based design, system identification, and control often optimize one object's parameters for their effect on another object. A hanger's shape determines whether a garment remains supported. An initial velocity determines where another body moves after collision, and a robot command changes the motion of a grasped deformable object. In each case, the gradient must follow forces transmitted through contact and the subsequent motion. Computing these derivatives efficiently is particularly challenging when a scene combines deformable objects with detailed, nearly rigid components: the representation must resolve the interactions without making every stiff component expensive to differentiate.

Existing methods address different parts of this problem. Differentiable projective-dynamics methods, including DiffPD and DiffCloth [Du et al. 2021; Li et al. 2022], accelerate repeated solves through local-global iterations and matrix prefactorization. Representing nearly rigid components as stiff deformable meshes nevertheless retains vertex-level motion DoFs, increasing simulation

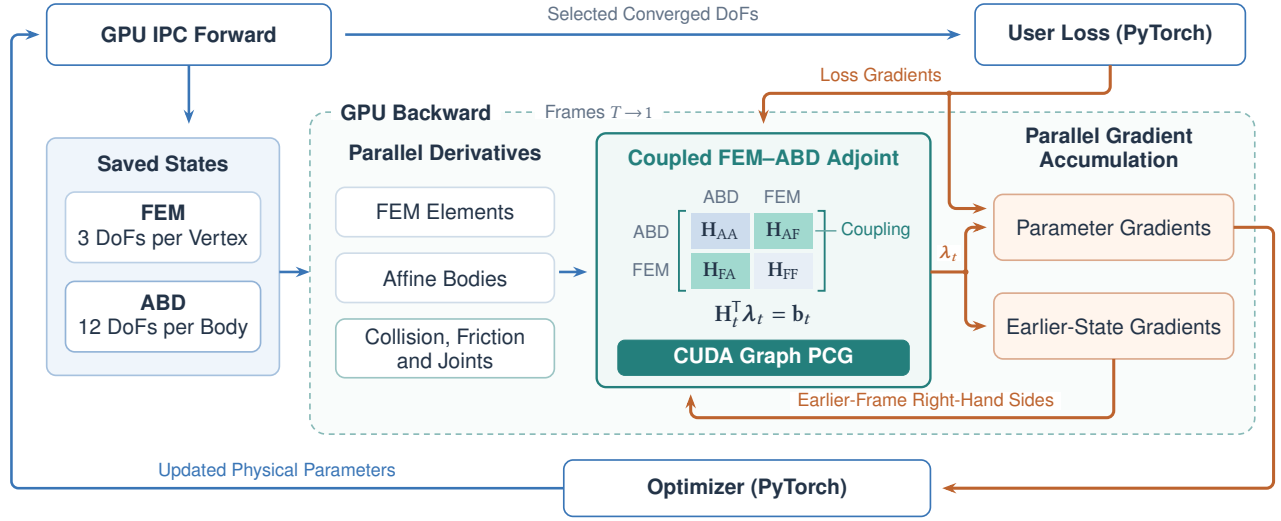


Fig. 2. DiffGIPC’s backward pipeline. Each reverse frame assembles one coupled FEM-ABD adjoint system, retaining collision, friction, and joint interactions with twelve DoFs per affine body. The GPU solve and local accumulation propagate gradients to earlier states and physical parameters for optimization through PyTorch.

and differentiation costs even when only near-rigid motion is required [Lai et al. 2026]. ADD [Geilinger et al. 2020] provides fully implicit, two-way coupled rigid and deformable dynamics with analytic adjoints, but uses penalty-based normal contact, which does not strictly prevent interpenetration. DiffIPC [Huang et al. 2024c] differentiates fully implicit FEM dynamics with friction, using IPC barriers and continuous collision detection to maintain non-penetration. Its full-space representation, however, carries the motion DoFs of every stiff component into the backward solve, making repeated evaluations costly in its CPU implementation. The resulting opportunity is to combine contact-coupled differentiation with compact motion coordinates and efficient GPU execution.

We present DiffGIPC, a GPU-accelerated framework for differentiating coupled FEM-affine IPC simulations (Figure 2). Building on the forward formulation of StiffGIPC [Huang et al. 2025a], we derive frame-wise adjoints directly in the mixed motion coordinates. Full-space FEM retains the freedom needed for local deformation, while each affine body contributes only twelve motion DoFs regardless of its surface resolution. Contact and joint derivatives couple interacting bodies within each frame. Inertia and friction make the time integration equations depend on earlier states. The backward systems therefore remain compact while preserving cross-object coupling and the temporal dependencies required for coupled inverse problems.

Compact motion coordinates alone do not provide shape or control gradients. Changing an affine rest surface changes its mass, the positions at which contact is evaluated, and the maps that convert point-space forces to affine coordinates. Under friction, the same rest change also moves the previous-frame points used to define sliding and the lagged contact quantities. We differentiate these dependencies explicitly, together with initial states and revolute or prismatic joint controls, and accumulate them in parallel without enlarging the adjoint system. To retain the advantage of compact systems,

conditional CUDA Graphs are employed to keep preconditioned conjugate gradient (PCG) iterations and convergence checks on the GPU, thereby reducing the overhead of repeated kernel launches and CPU-GPU synchronization. A PyTorch interface connects these physical gradients to user-defined parameter maps and losses.

In summary, our contributions are:

- **Coupled FEM-affine adjoints for differentiable IPC.** We derive trajectory adjoints in mixed FEM and affine coordinates, preserving contact, friction, and joint coupling while using only twelve motion DoFs per affine body.
- **Affine-body derivatives for shape design and articulated control.** We derive gradients for affine rest shapes, initial states, and joint controls, including shape-dependent inertia and frictional contact.
- **A reusable GPU optimization pipeline.** We develop a GPU-parallel adjoint and gradient-accumulation pipeline with conditional CUDA Graphs and a PyTorch interface for design, system identification, and control.
- **Fully GPU-parallel differentiable IPC.** We present the first fully GPU-parallel differentiable IPC framework for coupled nearly rigid and deformable bodies, with substantial speedups demonstrated across contact-rich inverse problems.

Our evaluation examines the benefits of GPU execution and compact motion representations across contact-rich inverse problems. For comparison with DiffIPC, we retain full-space FEM in both systems and use relatively low-resolution benchmarks to keep evaluation times manageable for the CPU-based baseline. Even at these resolutions, DiffGIPC achieves 12.3–34.1× speedups in complete backward evaluation and 10.6–60.9× speedups in combined forward and backward evaluation, covering initial motion, rest shape, material, and friction optimization. A separate resolution study examines how performance scales with geometric detail and the additional

benefit of affine coordinates. Further experiments demonstrate cloth design, affine shape optimization, stable placement, and articulated control. The source code of our framework will be released.

2 Related Work

Efficient differentiable simulation and contact. Inverse problems such as parameter identification from measured motion [Hahn et al. 2019; Jatavallabhula et al. 2021] require repeated simulation and gradient evaluations, making the efficiency of both computations important. DiffTaichi automates the differentiation of parallel simulation programs through source-code transformations [Hu et al. 2020]. Brax integrates differentiable rigid-body simulation and learning on accelerators, reducing data transfer between simulation and optimization [Freeman et al. 2021]. ChainQueen [Hu et al. 2019] uses explicit MLS-MPM with parallel local updates, but stiff materials require small time steps. Implicit deformable simulation methods [Li et al. 2026; Liang et al. 2019; Rojas et al. 2021] offer greater time-step stability, at the cost of solving linear systems during both simulation and differentiation. Projective dynamics (PD) reduces this cost through local projections and a global system whose matrix can be prefactorized for fixed discretization and coefficients [Bouaziz et al. 2014]. DiffPD reuses this factorization in its backward iterations [Du et al. 2021]. Quasi-Newton methods support more general hyperelastic energies through efficient approximate systems [Liu et al. 2017], and DiffQN further reduces differentiation costs by reusing factorizations from the forward simulation [Cai et al. 2025]. More recently, DiffPhD combines GPU factor reuse with techniques for heterogeneous and hyperelastic materials [Lai et al. 2026].

For contact-rich inverse problems, these computational benefits must also be considered alongside the contact response being differentiated. DiffPD’s penalty formulation supports frictional sliding but permits interpenetration, whereas its complementarity formulation enforces non-penetration under a sticky-contact assumption [Du et al. 2021]. DiffCloth differentiates the dry-friction PD formulation of Ly et al. [2020], supporting sliding cloth contact [Li et al. 2022]. However, its self-collision treatment considers only vertex-vertex pairs and does not handle vertex-face or edge-edge self-collisions. DiffPhD supports Signorini-Coulomb contact on the GPU, but its velocity-level enforcement can produce position drift under quasi-static support [Lai et al. 2026]. These limitations are particularly relevant when the optimized behavior depends on sustained support, sliding, or contact between thin surfaces. They motivate efficient differentiation of contact models that preserve geometric non-penetration throughout the simulated motion.

Differentiable contact and coupled dynamics. Qiao et al. [2020] enable differentiable two-way coupling between rigid bodies and cloth, using local impact zones to reduce the cost of collision handling and differentiation. ADD provides fully implicit rigid-deformable coupling with analytic adjoints [Geilinger et al. 2020]. Its normal penalty makes contact differentiable while allowing finite overlap. Increasing the penalty stiffness reduces overlap at the cost of poorer conditioning. IPC instead combines a geometric barrier with continuous collision detection to maintain intersection-free motion from feasible initial configurations [Li et al. 2020]. DiffIPC brings this contact model to differentiable FEM simulation and supports

shape, material, friction, and initial-state optimization [Huang et al. 2024c]. It therefore provides a strong basis for inverse problems in which penetration would change the intended interaction. However, representing nearly rigid components with highly stiff, full-space FEM retains vertex-level motion DoFs even when their local deformation is negligible. In its CPU-based implementation, repeated simulation and differentiation of these large systems remain costly. DiffGIPC addresses this cost with coupled FEM-ABD adjoints and GPU backward computation while retaining IPC’s geometric contact safeguards.

Compact coordinates for coupled design and control. Reducing motion DoFs addresses a cost that faster linear algebra alone does not remove. Modal and adaptive subspaces reduce deformable motion to a smaller set of directions [Barbič and James 2005; Trusty et al. 2024]. Their accuracy depends on whether those directions capture the deformation induced by loads and contact. Hyper-reduced PD also approximates local constraint projections [Brandt et al. 2018], while subspace mixed FEM combines reduced positions with stretch variables to handle heterogeneous materials [Trusty et al. 2023]. These approaches retain selected deformation modes and reduce their evaluation cost. For components whose internal deformation is negligible, rigid-body coordinates avoid resolving these modes altogether. Rigid IPC preserves non-penetration in this representation by accounting for curved rotational trajectories in collision detection [Ferguson et al. 2021]. ABD relaxes exact rigidity through a stiff orthogonality energy and uses twelve affine DoFs per body [Lan et al. 2022]. Its surface positions are linear in these DoFs, allowing piecewise-linear collision trajectories and a pose-independent mass matrix at fixed rest geometry. This offers a compact representation for stiff parts, while full-space FEM retains local deformation where it is needed. DiffGIPC carries both representations into the same backward system, retaining contact, friction, and joint coupling between bodies.

Compact rigid-body states have supported differentiable learning and control [Belbute-Peres et al. 2018; Degraeve et al. 2019], including articulated dynamics in Nimble [Werling et al. 2021]. Shape optimization requires an additional distinction between the motion representation and the geometry being designed. Contact-based assemblies and shock-protecting microstructures [Huang et al. 2024b; Tozoni et al. 2021] use geometric changes to alter load transfer and contact-dependent mechanical response. Pneumatic actuator design and garment reconstruction [Gjoka et al. 2024; Li et al. 2025] likewise optimize rest geometry through its effect on deformation. These applications motivate retaining a rich design space even when the motion state can be compact.

SDRS uses convex polyhedra and auxiliary separating planes to maintain differentiability under large shape changes [Ye et al. 2025]. Our focus is mesh-based design and control of affine bodies interacting with deformable objects under IPC. Affine rest-shape changes affect both the mass matrix and the maps from affine DoFs to surface points. Differentiating these dependencies allows a body’s shape to be optimized for its frictional interactions with other objects. Derivatives of joint targets and forces likewise account for the coupled response of articulated and deformable bodies. For a fixed motion representation, additional design parameters do not enlarge the

per-frame backward system. Each affine body therefore contributes only twelve motion DoFs, even when its rest shape is described by many design parameters. Contact processing still depends on surface resolution, while parameter-gradient evaluation requires additional work as the number of design parameters grows.

GPU computation for trajectory adjoints. Compact coordinates reduce adjoint system size, while efficient linear solvers further reduce the cost of trajectory differentiation. PD-based local-global methods exploit a fixed global matrix to reuse precomputed factors. Zeng et al. [2025] use a sparse representation of its inverse to accelerate GPU forward simulation. DiffPhD adopts this strategy for differentiable simulation, reusing an inverse factor across forward, backward, and frictional-contact computations [Lai et al. 2026]. In IPC, however, the frame Hessian varies with deformation and contact, so a fixed factorization cannot be reused across all adjoint systems along a trajectory. Preconditioned iterative solvers provide an alternative, with effective preconditioning reducing the number of iterations [Chen et al. 2021; Wu et al. 2022]. GIPC parallelizes IPC computation [Huang et al. 2024a], while StiffGIPC provides efficient FEM-ABD coupling and GPU solvers across a wide range of stiffnesses [Huang et al. 2025a], and AGIPC further reduces GPU linear-solve costs through adaptive in-solve algebraic coarsening [Wang et al. 2026]. These methods advance GPU IPC efficiency, with StiffGIPC supplying the forward simulation foundation for our work.

DiffGIPC combines coupled frame-wise adjoint solves with parallel gradient accumulation on the GPU. Compact affine coordinates reduce PCG’s numerical workload, making repeated CPU kernel launches and convergence synchronization more prominent. Conditional CUDA Graphs reduce this overhead by keeping iteration control and convergence checks on the GPU, with compatible graphs reused across frames as the Hessian changes. After each solve, local derivative products are accumulated in parallel to compute physical-parameter gradients and update earlier-frame adjoint right-hand sides.

3 A Unified Differentiable IPC Framework

The framework separates the coordinates needed to describe contact geometry from those needed to solve motion. Collision and friction derivatives are evaluated at world-space surface points and mapped into a coupled system of FEM vertex positions and affine body coordinates. This preserves cross-object derivatives while assigning twelve motion DoFs to each affine body.

The forward formulation builds on StiffGIPC [Huang et al. 2025a]. Simulation DoFs describe solved motion, whereas physical parameters are inputs changed by the optimizer. One frame is an accepted time step of length h , and mesh connectivity is fixed within each differentiated trajectory. The state and incremental potential below define the single-step adjoint. Section 4 extends it through time, Section 5 supplies affine design and control derivatives, and Section 6 develops their GPU computation and task interface.

3.1 Simulation degrees of freedom and contact points

A free FEM vertex i is represented by its world-space position $\mathbf{x}_{t,i} \in \mathbb{R}^3$ at frame t . An affine body b is represented by its translation $\mathbf{p}_{t,b} \in$

$\mathbb{R}^3$ and affine transformation $\mathbf{A}_{t,b} \in \mathbb{R}^{3 \times 3}$. Let $\boldsymbol{\alpha}_{t,b,1}^\top$, $\boldsymbol{\alpha}_{t,b,2}^\top$, and $\boldsymbol{\alpha}_{t,b,3}^\top$ be the rows of $\mathbf{A}_{t,b}$. Stacking these vectors with the translation defines the body’s twelve DoFs:

$$\mathbf{A}_{t,b} = \begin{bmatrix} \boldsymbol{\alpha}_{t,b,1}^\top \\ \boldsymbol{\alpha}_{t,b,2}^\top \\ \boldsymbol{\alpha}_{t,b,3}^\top \end{bmatrix}, \quad (1)$$

$$\mathbf{q}_{t,b} = [\mathbf{p}_{t,b}^\top, \boldsymbol{\alpha}_{t,b,1}^\top, \boldsymbol{\alpha}_{t,b,2}^\top, \boldsymbol{\alpha}_{t,b,3}^\top]^\top \in \mathbb{R}^{12}.$$

For a surface vertex with rest position $\bar{\mathbf{x}}_i$, the world-space position is

$$\mathbf{x}_{t,i} = \mathbf{p}_{t,b} + \mathbf{A}_{t,b} \bar{\mathbf{x}}_i. \quad (2)$$

Let $\mathbf{I}_3$ denote the 3×3 identity matrix and $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ the Cartesian unit vectors. With rest position fixed, the Jacobian of this position with respect to $\mathbf{q}_{t,b}$ is the 3×12 matrix

$$\mathbf{J}_i = \mathbf{J}(\bar{\mathbf{x}}_i) = [\mathbf{I}_3 \ \mathbf{e}_1 \bar{\mathbf{x}}_i^\top \ \mathbf{e}_2 \bar{\mathbf{x}}_i^\top \ \mathbf{e}_3 \bar{\mathbf{x}}_i^\top], \quad (3)$$

$$\mathbf{x}_{t,i} = \mathbf{J}_i \mathbf{q}_{t,b}.$$

All surface vertices on body b share the same $\mathbf{q}_{t,b}$, but each has its own rest-dependent map $\mathbf{J}_i$.

For N_A free affine bodies and N_F free FEM vertices, we collect the simulation DoFs in

$$\mathbf{y}_t = [\mathbf{q}_{t,1}^\top, \dots, \mathbf{q}_{t,N_A}^\top, \mathbf{x}_{t,1}^\top, \dots, \mathbf{x}_{t,N_F}^\top]^\top. \quad (4)$$

The total number of simulation DoFs is $n = 12N_A + 3N_F$. The set and ordering of free DoFs do not change throughout each trajectory. Adding surface vertices to an affine body can increase contact work, but does not enlarge its block in the linear system. Rest coordinates are design parameters, not additional simulation DoFs.

For contact c , we collect the world-space positions of its n_c surface points into $\mathbf{x}_t^c$. Let $\mathbf{y}_t \in \mathbb{R}^n$ contain the free simulation DoFs. The Jacobian $\mathbf{K}_{t,c}$ describes how changes in these DoFs affect the contact-point positions:

$$\mathbf{x}_t^c = [\mathbf{x}_{t,i_1}^\top, \dots, \mathbf{x}_{t,i_{n_c}}^\top]^\top, \quad (5)$$

$$\mathbf{K}_{t,c} = \frac{\partial \mathbf{x}_t^c}{\partial \mathbf{y}_t} \in \mathbb{R}^{3n_c \times n}.$$

Each group of three rows corresponds to one contact point. For a free FEM vertex, its three positional DoFs appear directly in $\mathbf{y}_t$, so these rows contain an identity block $\mathbf{I}_3$ in the corresponding columns. For a point on a free affine body, its position depends on the body’s twelve affine DoFs through $\mathbf{J}_i$, so these rows contain $\mathbf{J}_i$ in that body’s columns. For each free point, columns outside its associated vertex or body block are zero. A fixed point contributes three zero rows. When differentiating with respect to the free motion DoFs, we hold the rest geometry and prescribed motion fixed. When differentiating with respect to the rest shape, we also differentiate the dependence of the affine point maps $\mathbf{J}_i$, and hence $\mathbf{K}_{t,c}$, on the rest positions (Section 5.2).

3.2 Incremental potential and its energy gradient

Let $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ collect the physical parameters selected for optimization, such as material and friction coefficients, rest coordinates, initial states, and joint controls. Here n_θ is the number of selected scalar parameters. These parameters are held fixed during one forward simulation. A joint-control schedule can still contain different prescribed values for different frames. For a given $\boldsymbol{\theta}$, each time step

of length h computes the simulation DoFs $\mathbf{y}_t$ by minimizing the incremental potential

$$\Phi_t(\mathbf{y}_t) = \frac{1}{2} \|\mathbf{y}_t - \hat{\mathbf{y}}_t\|_{\mathbf{M}}^2 + \sum_{e \in \mathcal{E}} w_e(h, \boldsymbol{\theta}) \Psi_{t,e}(\mathbf{y}_t, \mathbf{y}_{t-1}; \boldsymbol{\theta}). \quad (6)$$

The first term accounts for inertia, with $\|\mathbf{v}\|_{\mathbf{M}}^2 = \mathbf{v}^\top \mathbf{M} \mathbf{v}$. The mass matrix $\mathbf{M}$ is expressed in the simulation coordinates: it contains vertex-mass blocks for FEM and a 12×12 block for each affine body. Section 5 gives the affine mass matrix and its rest-shape derivatives. The predictor $\hat{\mathbf{y}}_t$ is computed from earlier states and prescribed accelerations $\mathbf{a}_t$. For dynamic bodies, backward Euler (BDF1) gives

$$\hat{\mathbf{y}}_t = 2\mathbf{y}_{t-1} - \mathbf{y}_{t-2} + h^2 \mathbf{a}_t, \quad t \geq 2. \quad (7)$$

At the first step, the predictor is $\hat{\mathbf{y}}_1 = \mathbf{y}_0 + h\dot{\mathbf{y}}_0 + h^2 \mathbf{a}_1$, where $\mathbf{y}_0$ specifies the initial configuration and $\dot{\mathbf{y}}_0$ contains the initial FEM vertex velocities and the initial velocities of the affine coordinates.

The second term in Equation (6) sums the contributions from FEM elastic energy, affine shape energy, collision, friction, and joints. Each contribution is indexed by $e \in \mathcal{E}$ and written as $w_e(h, \boldsymbol{\theta}) \Psi_{t,e}$, with energy function $\Psi_{t,e}$ and scalar weight $w_e(h, \boldsymbol{\theta})$. The collision energy is the non-penetration barrier, distinct from the friction energy for tangential resistance. IPC's friction energy depends on the relative tangential displacement of the two configurations, determined by $\mathbf{y}_t$ and $\mathbf{y}_{t-1}$. Its tangent bases and normal forces are held fixed from the previous-frame positions when solving for $\mathbf{y}_t$, and their dependence on those positions is included in the derivative with respect to $\mathbf{y}_{t-1}$.

At a minimum of the incremental potential, its gradient $\mathbf{g}_t$ with respect to the free simulation DoFs satisfies the stationarity equation

$$\begin{aligned} \mathbf{g}_t(\mathbf{y}_t, \mathbf{y}_{t-1}, \mathbf{y}_{t-2}; \boldsymbol{\theta}) &:= \nabla_{\mathbf{y}_t} \Phi_t \\ &= \mathbf{M}(\mathbf{y}_t - \hat{\mathbf{y}}_t) \\ &\quad + \sum_{e \in \mathcal{E}} w_e(h, \boldsymbol{\theta}) \nabla_{\mathbf{y}_t} \Psi_{t,e} \\ &= \mathbf{0}. \end{aligned} \quad (8)$$

The energy gradient $\mathbf{g}_t$ is also the nonlinear residual of this equation. For each parameter choice, the forward solver finds DoFs that satisfy this stationarity condition. Differentiating the condition determines how a parameter change affects the solved DoFs and the loss evaluated on them.

3.3 Parameter gradients from the step equation

Consider first a single time step from a prescribed initial state, omitting the frame index. The solved DoFs satisfy $\mathbf{g}(\mathbf{y}, \boldsymbol{\theta}) = \mathbf{0}$. A parameter change affects this equation both directly and through the resulting change in $\mathbf{y}$. Differentiating the stationarity equation gives

$$\underbrace{\frac{\partial \mathbf{g}}{\partial \mathbf{y}} \frac{d\mathbf{y}}{d\boldsymbol{\theta}}}_{\mathbf{H}} + \underbrace{\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}}_{\mathbf{G}} = \mathbf{0}. \quad (9)$$

Here, $\mathbf{G}$ is the partial derivative of the energy gradient with respect to the physical parameters. The derivative $\mathbf{H}$ with respect to the

DoFs is the Hessian of the incremental potential:

$$\mathbf{H} = \frac{\partial \mathbf{g}}{\partial \mathbf{y}} = \nabla_{\mathbf{y}}^2 \Phi. \quad (10)$$

Both matrices are evaluated at the converged DoFs satisfying $\mathbf{g}(\mathbf{y}, \boldsymbol{\theta}) = \mathbf{0}$. Rather than forming $d\mathbf{y}/d\boldsymbol{\theta}$ explicitly, we use the adjoint below. Restoring the frame index, we denote these matrices by $\mathbf{H}_t \in \mathbb{R}^{n \times n}$ and $\mathbf{G}_t \in \mathbb{R}^{n \times n_\theta}$.

For a scalar loss $L(\mathbf{y}, \boldsymbol{\theta})$, let $\boldsymbol{\ell}_y = \nabla_{\mathbf{y}} L$ and $\boldsymbol{\ell}_\theta = \nabla_{\boldsymbol{\theta}} L$ denote its partial gradients with respect to the DoFs and physical parameters, respectively. The chain rule gives

$$\nabla_{\boldsymbol{\theta}} L = \boldsymbol{\ell}_\theta + \left(\frac{d\mathbf{y}}{d\boldsymbol{\theta}} \right)^\top \boldsymbol{\ell}_y. \quad (11)$$

Explicitly computing $d\mathbf{y}/d\boldsymbol{\theta}$ becomes costly as the number of optimized parameters grows. We avoid forming this matrix and compute the loss gradient by solving the adjoint system:

$$\mathbf{H}^\top \boldsymbol{\lambda} = \boldsymbol{\ell}_y. \quad (12)$$

The solution $\boldsymbol{\lambda}$ is the adjoint vector. Combining this solve with Equation (9) gives the parameter gradient

$$\nabla_{\boldsymbol{\theta}} L = \boldsymbol{\ell}_\theta - \mathbf{G}^\top \boldsymbol{\lambda}. \quad (13)$$

For a single time step, all parameter gradients share the same coupled FEM-affine adjoint. Section 4 extends this formulation through time.

3.4 Collision and friction derivatives across representations

Collision and friction energies are evaluated at world-space surface points, whereas the adjoint in Equation (12) acts on simulation DoFs. The map $\mathbf{K}_{t,c}$ from Equation (5) converts the point-space derivatives into FEM and affine coordinates. The collision energy uses GIPC's squared-log barrier on squared distance [Huang et al. 2024a]

$$b_{\text{bar}}(d) = \begin{cases} \kappa(d - \widehat{d})^2 \log^2(d/\widehat{d}), & 0 < d < \widehat{d}, \\ 0, & d \geq \widehat{d}, \end{cases} \quad (14)$$

where d is the squared distance, $\widehat{d}$ is its activation threshold, and κ is the barrier coefficient. This squared-log form is the GIPC barrier, rather than the original IPC single-log form. At each Newton iteration, we truncate the step along the search direction using continuous collision detection to prevent intersections, followed by line search to ensure a decrease in the incremental potential. Backward differentiation uses the converged frame state and its associated collision and friction data.

For a contact c at frame t , let $\mathbf{g}_{t,c}^x$ and $\mathbf{H}_{t,c}^x$ denote the gradient and Hessian of the collision energy with respect to the stacked world-space point positions $\mathbf{x}_t^c$. The superscript x denotes point coordinates, whereas y denotes simulation DoFs. With rest geometry fixed, the point map $\mathbf{K}_{t,c}$ from Equation (5) is constant with respect to $\mathbf{y}_t$. The chain rule therefore gives the contributions to the frame energy gradient and Hessian:

$$\mathbf{g}_{t,c}^y = \mathbf{K}_{t,c}^\top \mathbf{g}_{t,c}^x, \quad \mathbf{H}_{t,c}^y = \mathbf{K}_{t,c}^\top \mathbf{H}_{t,c}^x \mathbf{K}_{t,c}. \quad (15)$$

The same transformation applies to friction derivatives with respect to the current-frame DoFs. Friction derivatives with respect to earlier states are included in the trajectory calculation in Section 4.

Each point contributes through the map associated with its object. For point i on affine body b and point j on affine body b' , let $\mathbf{H}_{ij}^x$ be their 3×3 point-space Hessian block. Mapping this block gives a 12×12 contribution to $\mathbf{H}_{bb'}^{AA}$. If the second point is instead a FEM vertex j , its identity map gives a 12×3 contribution to $\mathbf{H}_{bj}^{AF}$. For the point pair (i, j) , the block contributions are

$$\begin{aligned}\Delta \mathbf{H}_{bb'}^{AA,(i,j)} &= \mathbf{J}_i^\top \mathbf{H}_{ij}^x \mathbf{J}_j, \\ \Delta \mathbf{H}_{bj}^{AF,(i,j)} &= \mathbf{J}_i^\top \mathbf{H}_{ij}^x.\end{aligned}\quad (16)$$

All surface vertices of an affine body share its twelve DoFs. The contributions of all relevant point pairs are summed into the corresponding body blocks, without introducing independent vertex DoFs. For distinct points on the same body, both ordered point-block contributions must be included when they map to the same diagonal body block.

After assembly, grouping affine (A) and finite-element (F) DoFs gives the frame Hessian

$$\mathbf{H}_t = \begin{bmatrix} \mathbf{H}_{AA} & \mathbf{H}_{AF} \\ \mathbf{H}_{FA} & \mathbf{H}_{FF} \end{bmatrix}, \quad \mathbf{H}_{FA} = \mathbf{H}_{AF}^\top. \quad (17)$$

The diagonal blocks $\mathbf{H}_{AA}$ and $\mathbf{H}_{FF}$ collect derivatives within the affine and FEM coordinates. Collision and friction contribute $\mathbf{H}_{AF}$ and $\mathbf{H}_{FA}$, while joints couple affine bodies within $\mathbf{H}_{AA}$. These cross blocks let a loss on one object influence parameters of another. Compact affine coordinates reduce system size while the cross terms preserve the interacting response.

The same point maps form the loss-gradient vector ℓ_y on the right-hand side of Equation (12). For a loss evaluated on world-space surface points, its gradient with respect to the affine coordinates of body b , whose surface-vertex index set is $\mathcal{V}_b$, is

$$\nabla_{q_b} L = \sum_{i \in \mathcal{V}_b} \mathbf{J}_i^\top \nabla_{x_i} L. \quad (18)$$

For a FEM vertex, the loss gradient with respect to its DoFs equals that with respect to its world-space position. These mappings hold with rest geometry fixed. Section 5.2 includes the additional dependence of the point maps and surface positions on rest shape. Section 4 extends this coupled adjoint to trajectories, where earlier states affect both later motion and the loss.

4 Frame-Wise Adjoint

The single-step adjoint captures interactions at one frame. A trajectory also depends on how earlier motion changes later inertia and friction. We account for these dependencies by backward substitution through the causal sequence of frame equations. Each reverse frame solves one coupled FEM-affine system, then contributes to the parameters and earlier states that influenced it. Algorithm 1 summarizes this recurrence. We derive its temporal structure and then specify the inertia and friction terms, using the converged states and recorded data that define the forward equations.

4.1 Equations for the whole trajectory

Let $\mathbf{y}_t \in \mathbb{R}^n$ contain the free simulation DoFs in the order defined in Section 3.1. The equation at frame t depends on its current state,

Algorithm 1 Frame-wise adjoint for a converged trajectory

Input: Converged states $\mathbf{y}_1, \dots, \mathbf{y}_T$ and their frame data, and loss partial gradients $\ell_1, \dots, \ell_T$ and ℓ_θ (§4.2)

Output: Parameter gradient $\bar{\theta} = \nabla_{\theta} L$

1: $\mathbf{b}_t \leftarrow \ell_t$ for $t = 1, \dots, T$; $\bar{\theta} \leftarrow \ell_\theta$ (§4.3)

2: **for** $t = T, T-1, \dots, 1$ **do** (§4.2)

3: Restore frame t 's converged state and earlier-frame data

4: Assemble the coupled Hessian $\mathbf{H}_t$

5: Solve $\mathbf{H}_t^\top \lambda_t = \mathbf{b}_t$ (§4.3)

6: $\bar{\theta} \leftarrow \bar{\theta} - \mathbf{G}_t^\top \lambda_t$ (§4.3)

 Include optimized initial-state derivatives in $\mathbf{G}_t$ (§4.4)

7: **for** each earlier solved frame s affecting $\mathbf{g}_t$ **do** (§4.1)

8: $\mathbf{b}_s \leftarrow \mathbf{b}_s - (\partial \mathbf{g}_t / \partial \mathbf{y}_s)^\top \lambda_t$

 Inertia: §4.4, and friction: §4.5

9: **end for**

10: **end for**

11: **return** $\bar{\theta}$

a subset of earlier solved states, and the physical parameters:

$$\mathbf{g}_t(\mathbf{y}_t, \{\mathbf{y}_s : s \in \mathcal{D}(t)\}, \boldsymbol{\theta}) = \mathbf{0}. \quad (19)$$

The set $\mathcal{D}(t) \subseteq \{1, \dots, t-1\}$ identifies those earlier frames. The initial state is an input rather than a solved frame: its optimized components contribute to $\mathbf{G}_t$, while unselected components remain fixed. Prescribed joint commands likewise enter $\mathbf{G}_t$ through the applied controls. The matrices $\mathbf{H}_t$ and $\mathbf{G}_t$ retain their state and parameter definitions from Section 3.3. Temporal dependence introduces the additional derivative

$$\mathbf{C}_{t,s} = \frac{\partial \mathbf{g}_t}{\partial \mathbf{y}_s} \in \mathbb{R}^{n \times n}, \quad s \in \mathcal{D}(t). \quad (20)$$

Thus $\mathbf{H}_t$ describes the coupled response within frame t , whereas $\mathbf{C}_{t,s}$ describes its dependence on an earlier state and vanishes for $s \notin \mathcal{D}(t)$. Both current and earlier solved states vary with $\boldsymbol{\theta}$. Differentiating Equation (19) includes these changes together with the direct parameter derivative:

$$\mathbf{H}_t \frac{d\mathbf{y}_t}{d\boldsymbol{\theta}} + \sum_{s \in \mathcal{D}(t)} \mathbf{C}_{t,s} \frac{d\mathbf{y}_s}{d\boldsymbol{\theta}} + \mathbf{G}_t = \mathbf{0}. \quad (21)$$

With $\mathbf{S}_t = d\mathbf{y}_t/d\boldsymbol{\theta} \in \mathbb{R}^{n \times n_\theta}$, the equations for all frames form the block system

$$\underbrace{\begin{bmatrix} \mathbf{H}_1 & 0 & \cdots & 0 \\ \mathbf{C}_{2,1} & \mathbf{H}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \mathbf{C}_{T,1} & \cdots & \mathbf{C}_{T,T-1} & \mathbf{H}_T \end{bmatrix}}_{\mathcal{J}} \underbrace{\begin{bmatrix} \mathbf{S}_1 \\ \mathbf{S}_2 \\ \vdots \\ \mathbf{S}_T \end{bmatrix}}_{\mathcal{S}} = - \underbrace{\begin{bmatrix} \mathbf{G}_1 \\ \mathbf{G}_2 \\ \vdots \\ \mathbf{G}_T \end{bmatrix}}_{\mathcal{G}}. \quad (22)$$

Causality makes $\mathcal{J}$ block lower triangular, with the frame Hessians $\mathbf{H}_t$ on the diagonal and nonzero off-diagonal blocks only for dependencies in $\mathcal{D}(t)$.

The predictor in Equation (7) depends on $\mathbf{y}_{t-1}$ and $\mathbf{y}_{t-2}$. Friction adds to the first of these dependencies, through $\mathbf{C}_{t,t-1}$. For the first three solved frames, the system is

$$\begin{bmatrix} \mathbf{H}_1 & 0 & 0 \\ \mathbf{C}_{2,1} & \mathbf{H}_2 & 0 \\ \mathbf{C}_{3,1} & \mathbf{C}_{3,2} & \mathbf{H}_3 \end{bmatrix} \begin{bmatrix} \mathbf{S}_1 \\ \mathbf{S}_2 \\ \mathbf{S}_3 \end{bmatrix} = - \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{G}_2 \\ \mathbf{G}_3 \end{bmatrix}. \quad (23)$$

This three-frame system illustrates the reverse calculation below. The sensitivities $\mathbf{S}_t$ and the assembled trajectory matrix are used for the derivation, but are not formed by the backward algorithm.

4.2 Reverse-time adjoint equations

We extend the single-step adjoint in Equation (12) to the trajectory loss $L(\mathbf{y}_1, \dots, \mathbf{y}_T; \boldsymbol{\theta})$. Let $\boldsymbol{\ell}_t = \nabla_{\mathbf{y}_t} L$ denote its partial gradient with respect to the simulation DoFs at frame t , and let $\boldsymbol{\ell}_\theta$ denote its partial parameter gradient with all simulated states fixed. For a loss evaluated only on the final state, $\boldsymbol{\ell}_t = \mathbf{0}$ for $t < T$, although earlier states can still affect the loss through the subsequent motion. Using $\mathbf{S}_t = d\mathbf{y}_t/d\boldsymbol{\theta}$ from Section 4.1, the chain rule extends Equation (11) to

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} L &= \boldsymbol{\ell}_\theta + \sum_{t=1}^T \left(\frac{d\mathbf{y}_t}{d\boldsymbol{\theta}} \right)^\top \boldsymbol{\ell}_t \\ &= \boldsymbol{\ell}_\theta + \sum_{t=1}^T \mathbf{S}_t^\top \boldsymbol{\ell}_t \\ &= \boldsymbol{\ell}_\theta + \mathbf{S}^\top \boldsymbol{\ell}_{\text{all}}, \end{aligned} \quad (24)$$

where $\boldsymbol{\ell}_{\text{all}}$ stacks the per-frame loss gradients in the same order as the blocks of $\mathbf{S}$. The first term accounts for the explicit parameter dependence of the loss. The second accounts for changes in the simulated trajectory.

To evaluate the trajectory contribution without computing $\mathbf{S}$, we introduce an adjoint vector $\boldsymbol{\lambda}_t$ for each frame and solve

$$\underbrace{\mathcal{J}^\top}_{\Lambda} \underbrace{\begin{bmatrix} \boldsymbol{\lambda}_1 \\ \vdots \\ \boldsymbol{\lambda}_T \end{bmatrix}}_{\boldsymbol{\ell}_{\text{all}}} = \begin{bmatrix} \boldsymbol{\ell}_1 \\ \vdots \\ \boldsymbol{\ell}_T \end{bmatrix}. \quad (25)$$

Combining this equation with $\mathcal{J}\mathbf{S} = -\mathcal{G}$ from Equation (22) gives

$$\mathbf{S}^\top \boldsymbol{\ell}_{\text{all}} = \mathbf{S}^\top \mathcal{J}^\top \Lambda = -\mathcal{G}^\top \Lambda. \quad (26)$$

The transpose reverses the causal order: $\mathcal{J}^\top$ is block upper triangular, so the trajectory adjoint is solved from the last frame backward. For the three-frame example,

$$\begin{bmatrix} \mathbf{H}_1^\top & \mathbf{C}_{2,1}^\top & \mathbf{C}_{3,1}^\top \\ 0 & \mathbf{H}_2^\top & \mathbf{C}_{3,2}^\top \\ 0 & 0 & \mathbf{H}_3^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \\ \boldsymbol{\lambda}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\ell}_1 \\ \boldsymbol{\ell}_2 \\ \boldsymbol{\ell}_3 \end{bmatrix}. \quad (27)$$

Backward substitution first determines $\boldsymbol{\lambda}_3$, then includes its effect when solving for the earlier adjoints:

$$\begin{aligned} \mathbf{H}_3^\top \boldsymbol{\lambda}_3 &= \boldsymbol{\ell}_3, \\ \mathbf{H}_2^\top \boldsymbol{\lambda}_2 &= \boldsymbol{\ell}_2 - \mathbf{C}_{3,2}^\top \boldsymbol{\lambda}_3, \\ \mathbf{H}_1^\top \boldsymbol{\lambda}_1 &= \boldsymbol{\ell}_1 - \mathbf{C}_{2,1}^\top \boldsymbol{\lambda}_2 - \mathbf{C}_{3,1}^\top \boldsymbol{\lambda}_3. \end{aligned} \quad (28)$$

Applying the same backward substitution to all T frames gives

$$\mathbf{H}_t^\top \boldsymbol{\lambda}_t = \boldsymbol{\ell}_t - \sum_{\substack{\tau > t \\ t \in \mathcal{D}(\tau)}} \mathbf{C}_{\tau,t}^\top \boldsymbol{\lambda}_\tau, \quad t = T, \dots, 1. \quad (29)$$

Only later equations that depend on $\mathbf{y}_t$ contribute to its right-hand side. The resulting adjoints account for these temporal effects in the

parameter gradient:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} L &= \boldsymbol{\ell}_\theta - \mathcal{G}^\top \Lambda \\ &= \boldsymbol{\ell}_\theta - \sum_{t=1}^T \mathbf{G}_t^\top \boldsymbol{\lambda}_t. \end{aligned} \quad (30)$$

4.3 Computing one coupled system per frame

For each frame t , $\mathbf{b}_t$ denotes the right-hand side of its adjoint equation. The vector $\bar{\boldsymbol{\theta}}$ accumulates contributions to the loss gradient with respect to the physical parameters $\boldsymbol{\theta}$. The reverse pass starts from the direct loss derivatives, $\mathbf{b}_t = \boldsymbol{\ell}_t$ for $t = 1, \dots, T$ and $\bar{\boldsymbol{\theta}} = \boldsymbol{\ell}_\theta$. Before frame t is solved, $\mathbf{b}_t$ combines the direct loss gradient with respect to its motion DoFs $\mathbf{y}_t$ with all contributions propagated from later frames that depend on $\mathbf{y}_t$. The backward pass processes frames in reverse order. For each frame, $\mathbf{H}_t$ is first assembled from that frame's saved converged state. The adjoint is then obtained by solving

$$\mathbf{H}_t^\top \boldsymbol{\lambda}_t = \mathbf{b}_t. \quad (31)$$

Let $\Delta \mathbf{b}_s^{(t)}$ and $\Delta \bar{\boldsymbol{\theta}}^{(t)}$ denote frame t 's contributions to an earlier-frame right-hand side and the parameter gradient, respectively. They are given by

$$\Delta \mathbf{b}_s^{(t)} = -\mathbf{C}_{t,s}^\top \boldsymbol{\lambda}_t, \quad s \in \mathcal{D}(t), \quad (32)$$

$$\Delta \bar{\boldsymbol{\theta}}^{(t)} = -\mathbf{G}_t^\top \boldsymbol{\lambda}_t. \quad (33)$$

The history contributions are summed into the corresponding $\mathbf{b}_s$, giving the right-hand side in Equation (29). The parameter contributions are summed into $\bar{\boldsymbol{\theta}}$. After all frames have been processed, $\bar{\boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}} L$, as given by Equation (30).

Each history contribution reaches frame $s < t$ before that frame is solved (Figure 3). The products in Equations (32) and (33) are evaluated locally without assembling full $\mathbf{C}_{t,s}$ or $\mathbf{G}_t$. Only the current frame's Hessian is retained during this reverse traversal, together with the saved trajectory and accumulated right-hand sides.

Each retained $\mathbf{H}_t$ preserves cross-object coupling, while backward substitution carries that response through time. The trajectory length determines the number of frame solves. Additional parameters contribute local derivative products and share these solves, making the same recurrence suitable for low-dimensional controls and detailed shape design.

4.4 Inertia and initial-state dependencies

The history contribution in Equation (32) requires the derivative of $\mathbf{g}_t$ with respect to each earlier state. For inertia, $\mathbf{g}_t^{\text{in}} = \mathbf{M}(\mathbf{y}_t - \hat{\mathbf{y}}_t)$. Substituting the BDF1 predictor from Equation (7) and differentiating with respect to $\mathbf{y}_{t-1}$ and $\mathbf{y}_{t-2}$ gives

$$\mathbf{C}_{t,t-1}^{\text{in}} = -2\mathbf{M}, \quad \mathbf{C}_{t,t-2}^{\text{in}} = \mathbf{M}. \quad (34)$$

Substituting these blocks into Equation (32) gives the inertial parts of $\Delta \mathbf{b}_s^{(t)}$, denoted by the superscript in:

$$\begin{aligned} \Delta \mathbf{b}_{t-1}^{(t),\text{in}} &= 2\mathbf{M}\boldsymbol{\lambda}_t, \quad t \geq 2, \\ \Delta \mathbf{b}_{t-2}^{(t),\text{in}} &= -\mathbf{M}\boldsymbol{\lambda}_t, \quad t \geq 3. \end{aligned} \quad (35)$$

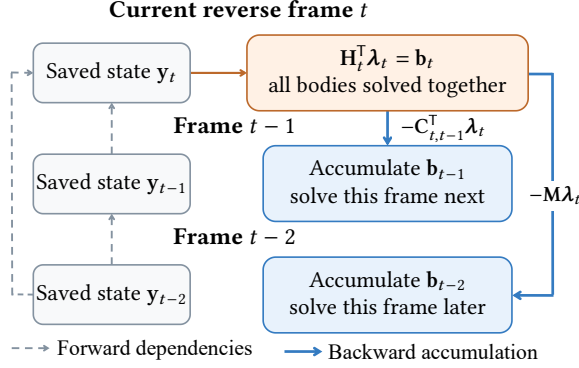


Fig. 3. Reverse-time accumulation in the frame-wise adjoint. Dashed arrows indicate forward state dependencies. Blue arrows carry contributions from the solved λ_t to the earlier right-hand sides $\mathbf{b}_{t-1}$ and $\mathbf{b}_{t-2}$. For the dynamic predictor shown, inertia and friction contribute to $\mathbf{b}_{t-1}$, while inertia contributes $-\mathbf{M}\lambda_t$ to $\mathbf{b}_{t-2}$. Each right-hand side also includes its direct loss gradient.

The same mass products apply to both representations: $\mathbf{M}$ contains the FEM vertex-mass blocks and the 12×12 mass block of each affine body.

The initial configuration and velocity are inputs to the simulation, not additional solved frames. Their contributions therefore enter the parameter-gradient contribution in Equation (33), rather than a right-hand-side vector $\mathbf{b}_0$. The first-step predictor is $\hat{\mathbf{y}}_1 = \mathbf{y}_0 + h\dot{\mathbf{y}}_0 + h^2\mathbf{a}_1$, so the inertial derivatives with respect to $\mathbf{y}_0$ and $\dot{\mathbf{y}}_0$ are $-\mathbf{M}$ and $-h\mathbf{M}$, respectively. For each frame t , the partial derivatives of $\mathbf{g}_t$ with respect to the optimized initial configuration and velocity form the corresponding $\mathbf{G}_t$. We evaluate these derivatives at the converged state of frame t and accumulate their contributions to the initial-state gradients using Equation (33).

For the velocity-reset approximation of quasi-static behavior, we set the predictor velocity to zero at every step, giving $\hat{\mathbf{y}}_t = \mathbf{y}_{t-1} + h^2\mathbf{a}_t$. The mass-weighted quadratic term in Equation (6) remains, so each update still depends on the previous configuration. Differentiating its energy gradient with respect to $\mathbf{y}_{t-1}$ contributes $-\mathbf{M}$ to $\mathbf{C}_{t,t-1}$. Substituting this contribution into Equation (32) gives the inertial contribution

$$\Delta \mathbf{b}_{t-1}^{(t),\text{in}} = \mathbf{M}\lambda_t, \quad t \geq 2. \quad (36)$$

4.5 Friction between frames

Friction adds a second source of temporal dependence: its energy uses both the current displacement and contact quantities computed from the previous configuration. DiffIPC differentiates this dependence for finite elements [Huang et al. 2024c]. Here the point maps carry these history derivatives into the coupled FEM-affine recurrence, including contacts whose points use different motion representations. For contact c , let $\mathbf{g}_{t,c}^{\text{fr},x}$ be the friction energy gradient with respect to the current world-space point positions $\mathbf{x}_t^c$. The maps $\mathbf{K}_{t,c}$ and $\mathbf{K}_{t-1,c}$ relate the same contact points to the simulation DoFs at the two frames. With rest geometry fixed, these maps do not depend on the solved DoFs. Differentiating $\mathbf{g}_{t,c}^{\text{fr},y} = \mathbf{K}_{t,c}^T \mathbf{g}_{t,c}^{\text{fr},x}$ with

respect to $\mathbf{y}_{t-1}$ therefore gives

$$\frac{\partial \mathbf{g}_{t,c}^{\text{fr},y}}{\partial \mathbf{y}_{t-1}} = \mathbf{K}_{t,c}^T \frac{\partial \mathbf{g}_{t,c}^{\text{fr},x}}{\partial \mathbf{x}_{t-1}^c} \mathbf{K}_{t-1,c}. \quad (37)$$

This is the contact's contribution to $\mathbf{C}_{t,t-1}$. For $t \geq 2$, contact c contributes $\Delta \mathbf{b}_{t-1}^{(t,c),\text{fr}}$ to the earlier-frame right-hand side. Equation (32) gives

$$\Delta \mathbf{b}_{t-1}^{(t,c),\text{fr}} = -\mathbf{K}_{t-1,c}^T \left(\frac{\partial \mathbf{g}_{t,c}^{\text{fr},x}}{\partial \mathbf{x}_{t-1}^c} \right)^T \mathbf{K}_{t,c} \lambda_t. \quad (38)$$

Summing these per-contact vectors gives the friction part of $\Delta \mathbf{b}_{t-1}^{(t)}$. Each contact point retains its FEM or affine map, so the same expression covers contact between the two representations.

The relative tangential displacement is computed from the corresponding vertex displacements using closest-point interpolation weights and a tangent basis. These weights, the tangent basis, and the normal force are computed from the converged configuration of the previous frame. They remain fixed throughout the current frame's solve. When differentiating with respect to the previous-frame positions, we also include the derivatives of these quantities. Appendix A.1 gives the friction law and its derivatives.

5 Affine-Body Derivatives for Shape, Initial States, and Control

The coupled adjoint describes how a loss responds to motion. To optimize an affine body's shape or controls, we must also differentiate how those parameters enter the frame equations. In particular, an affine body can expose many rest-shape coordinates for design, while adding only twelve motion DoFs per solve. A rest-shape change affects mass and inertia, world-space contact geometry, and the maps from point-space forces to affine coordinates. Friction adds the dependence on that geometry at the previous frame.

We derive these parameter derivatives, followed by initial-state and joint-control derivatives. For a scalar parameter θ_k , the partial derivative $\partial \mathbf{g}_t / \partial \theta_k$ forms column k of $\mathbf{G}_t$, evaluated at fixed solved DoFs. The adjoint accounts for the resulting motion response, and Equation (30) combines $-\mathbf{G}_t^T \lambda_t$ with the explicit loss derivative ℓ_{θ} . The following derivatives preserve this separation, allowing surface design, placement, and control to share the twelve-DoF affine representation.

5.1 Rest-dependent mass and inertia

For affine body b , let Ω_b be its rest domain and ρ_b its constant density. For a material point with rest position $\bar{\mathbf{x}}$, the map $\mathbf{J}(\bar{\mathbf{x}})$ in Equation (3) gives its world-space position as $\mathbf{x}_t = \mathbf{J}(\bar{\mathbf{x}})\mathbf{q}_{t,b}$. With rest geometry fixed, its velocity is $\dot{\mathbf{x}}_t = \mathbf{J}(\bar{\mathbf{x}})\dot{\mathbf{q}}_{t,b}$. The corresponding mass block is

$$\mathbf{M}_b = \int_{\Omega_b} \rho_b \mathbf{J}(\bar{\mathbf{x}})^T \mathbf{J}(\bar{\mathbf{x}}) dV. \quad (39)$$

Both $\mathbf{M}_b$ and its shape derivative can be computed from the closed surface mesh. We express the volume integral as signed triangle contributions, each depending explicitly on the triangle's rest vertices. Differentiating these expressions gives the mass-matrix derivatives directly from the design surface, without an interior tetrahedral discretization.

Expanding $\mathbf{J}(\bar{\mathbf{x}})^\top \mathbf{J}(\bar{\mathbf{x}})$ in Equation (39) gives terms that are constant, linear, or quadratic in the rest position $\bar{\mathbf{x}}$. The mass matrix $\mathbf{M}_b$ is determined by three integrals over the body's rest volume:

$$\begin{aligned} m_b &= \int_{\Omega_b} \rho_b dV, \\ \mathbf{m}_b^{(1)} &= \int_{\Omega_b} \rho_b \bar{\mathbf{x}} dV, \\ \mathbf{Q}_b^{(2)} &= \int_{\Omega_b} \rho_b \bar{\mathbf{x}} \bar{\mathbf{x}}^\top dV. \end{aligned} \quad (40)$$

Here m_b is the total mass. The vector $\mathbf{m}_b^{(1)}$ is the first mass moment and the matrix $\mathbf{Q}_b^{(2)}$ is the second mass moment. With density fixed, m_b , $\mathbf{m}_b^{(1)}$, and $\mathbf{Q}_b^{(2)}$ depend only on the body's rest shape.

The closed surface mesh provides these volume integrals. For an outward-oriented triangle f with rest vertices $\bar{\mathbf{x}}_{f,1}$, $\bar{\mathbf{x}}_{f,2}$, and $\bar{\mathbf{x}}_{f,3}$, the three vertices and the origin define a signed tetrahedron. Its volume and mass-weighted integrals can be calculated directly from the three vertex positions. With $\mathbf{s}_f = \bar{\mathbf{x}}_{f,1} + \bar{\mathbf{x}}_{f,2} + \bar{\mathbf{x}}_{f,3}$, the signed contributions are

$$\begin{aligned} V_f &= \frac{1}{6} \bar{\mathbf{x}}_{f,1}^\top (\bar{\mathbf{x}}_{f,2} \times \bar{\mathbf{x}}_{f,3}), \\ m_f &= \rho_b V_f, \quad \mathbf{m}_f^{(1)} = \frac{m_f}{4} \mathbf{s}_f, \\ \mathbf{Q}_f^{(2)} &= \frac{m_f}{20} \left(\sum_{j=1}^3 \bar{\mathbf{x}}_{f,j} \bar{\mathbf{x}}_{f,j}^\top + \mathbf{s}_f \mathbf{s}_f^\top \right). \end{aligned} \quad (41)$$

Summing m_f , $\mathbf{m}_f^{(1)}$, and $\mathbf{Q}_f^{(2)}$ over all surface triangles gives m_b , $\mathbf{m}_b^{(1)}$, and $\mathbf{Q}_b^{(2)}$, respectively. Retaining the signs in these sums recovers the volume moments from the oriented surface triangles.

Using these quantities and the coordinate order in Equation (1), the affine-body mass matrix is

$$\mathbf{M}_b = \begin{bmatrix} m_b \mathbf{I}_3 & \mathbf{e}_1 (\mathbf{m}_b^{(1)})^\top & \mathbf{e}_2 (\mathbf{m}_b^{(1)})^\top & \mathbf{e}_3 (\mathbf{m}_b^{(1)})^\top \\ \mathbf{m}_b^{(1)} \mathbf{e}_1^\top & \mathbf{Q}_b^{(2)} & 0 & 0 \\ \mathbf{m}_b^{(1)} \mathbf{e}_2^\top & 0 & \mathbf{Q}_b^{(2)} & 0 \\ \mathbf{m}_b^{(1)} \mathbf{e}_3^\top & 0 & 0 & \mathbf{Q}_b^{(2)} \end{bmatrix}. \quad (42)$$

The total mass determines the translation block. The first mass moment determines the blocks between translation and the affine transformation, while the second mass moment determines the blocks for the affine transformation.

Each contribution in Equation (41) is an explicit function of the triangle's rest vertex positions. With density and mesh connectivity fixed, summing their derivatives over the surface triangles gives dm_b , $d\mathbf{m}_b^{(1)}$, and $d\mathbf{Q}_b^{(2)}$. Substituting them into Equation (42) gives $d\mathbf{M}_b$. When the inverse mass matrix is needed, its derivative is $d(\mathbf{M}_b^{-1}) = -\mathbf{M}_b^{-1} (d\mathbf{M}_b) \mathbf{M}_b^{-1}$. Shape updates preserve the closed, consistently oriented, nondegenerate surface and the validity of its mass distribution. Invalid candidates are rejected before updating the simulation geometry.

5.2 Rest-shape gradients

Rest-shape derivatives express how the frame equations change when the affine design surface changes. Solved FEM positions and affine states are held fixed when forming $\mathbf{G}_t$, while an affine rest

variation still moves its surface point by $d\mathbf{x}_{t,i} = \mathbf{A}_{t,b} d\bar{\mathbf{x}}_i$. The resulting mass, contact-map, and energy-gradient variations are denoted by $d\mathbf{M}_b$, $d\mathbf{K}_{t,c}$, and $d\mathbf{g}_t$. They supply the corresponding parameter derivatives, with direct surface-loss dependence added separately to ℓ_θ .

Mass, inertia, and shape energy. For body b at frame t , the inertial energy gradient is $\mathbf{g}_{t,b}^{\text{in}} = \mathbf{M}_b (\mathbf{q}_{t,b} - \hat{\mathbf{q}}_{t,b})$. The product rule gives the corresponding first-order change:

$$d\mathbf{g}_{t,b}^{\text{in}} = (d\mathbf{M}_b) (\mathbf{q}_{t,b} - \hat{\mathbf{q}}_{t,b}) - \mathbf{M}_b d\hat{\mathbf{q}}_{t,b}. \quad (43)$$

The first term uses the mass derivative derived above. The second includes any rest-shape dependence in the inputs to the predictor. The affine shape incremental potential is $h^2 \kappa_A V_b \psi_A(\mathbf{q}_{t,b})$, where κ_A is the affine stiffness coefficient, V_b is the rest volume, and ψ_A is the near-rigidity penalty. Let $\mathbf{g}_{t,b}^{\text{shape}}$ be its gradient with respect to $\mathbf{q}_{t,b}$. With κ_A and $\mathbf{q}_{t,b}$ fixed, the rest-shape derivative of the combined inertial and shape contributions is

$$\begin{aligned} d(\mathbf{g}_{t,b}^{\text{in}} + \mathbf{g}_{t,b}^{\text{shape}}) &= d\mathbf{g}_{t,b}^{\text{in}} \\ &\quad + h^2 \kappa_A (dV_b) \nabla_{\mathbf{q}_{t,b}} \psi_A. \end{aligned} \quad (44)$$

The two contributions describe different physical effects of redesign: redistributing material changes the inertial response through $\mathbf{M}_b$, while changing the rest volume changes the scale of the near-rigidity energy. Both enter the parameter derivative at fixed affine coordinates. The adjoint then accounts for the resulting change in motion.

Collision-barrier derivatives and the point map. For a surface vertex i on body b , partial differentiation with respect to $\bar{\mathbf{x}}_i$ holds the translation $\mathbf{p}_{t,b}$ and affine transformation $\mathbf{A}_{t,b}$ fixed. Equation (2) gives the resulting world-space position change as $d\mathbf{x}_{t,i} = \mathbf{A}_{t,b} d\bar{\mathbf{x}}_i$. This variation also changes $\mathbf{J}_i$ and the corresponding entries of the contact map $\mathbf{K}_{t,c}$. For contact c , let dx_t^c collect the changes in its world-space point positions. Applying the product rule to the collision energy gradient $\mathbf{g}_{t,c}^{\text{ct},y} = \mathbf{K}_{t,c}^\top \mathbf{g}_{t,c}^{\text{ct},x}$ gives its first-order change:

$$\begin{aligned} d\mathbf{g}_{t,c}^{\text{ct},y} &= (d\mathbf{K}_{t,c})^\top \mathbf{g}_{t,c}^{\text{ct},x} \\ &\quad + \mathbf{K}_{t,c}^\top \mathbf{H}_{t,c}^{\text{ct},x} dx_t^c. \end{aligned} \quad (45)$$

Here ct denotes the collision-barrier term, and $\mathbf{H}_{t,c}^{\text{ct},x}$ is its Hessian with respect to the world-space point positions $\mathbf{x}_t^c$. The first term accounts for the changed mapping from a point force to the body's affine equations: even an unchanged point-space force has a different contribution when its material point moves. The second term accounts for the force change caused by the altered world-space collision geometry. Rest-shape changes affect both the world-space collision gradient and the mapping $\mathbf{K}_{t,c}$, so both terms must be included. Varying only a selected scalar rest-coordinate θ_k , while holding all other parameters and solved DoFs fixed, gives the corresponding partial derivative:

$$\begin{aligned} \frac{\partial \mathbf{g}_{t,c}^{\text{ct},y}}{\partial \theta_k} &= \left(\frac{\partial \mathbf{K}_{t,c}}{\partial \theta_k} \right)^\top \mathbf{g}_{t,c}^{\text{ct},x} \\ &\quad + \mathbf{K}_{t,c}^\top \mathbf{H}_{t,c}^{\text{ct},x} \frac{\partial \mathbf{x}_t^c}{\partial \theta_k}. \end{aligned} \quad (46)$$

$\partial \mathbf{g}_{t,c}^{\text{ct},y} / \partial \theta_k$ is contact c 's collision contribution to column k of $\mathbf{G}_t$.

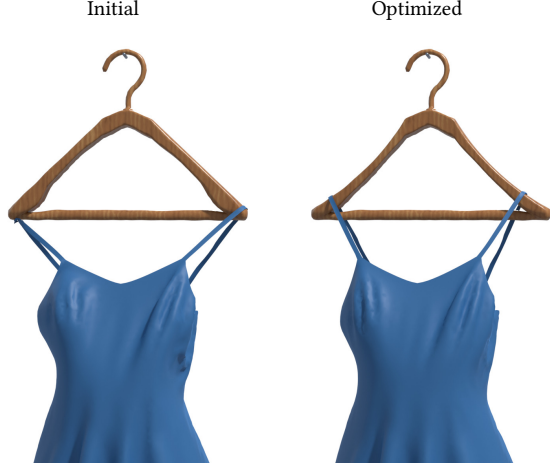


Fig. 4. Garment Hanger shape design. Two shared Bézier control scales modify both shoulders of the affine rest surface under frictional cloth contact. The initial and optimized configurations are shown at frame 33. The optimized hanger retains both garment straps.

Rest shape under frictional contact. The friction energy at frame t depends on the surface-point positions at both frames t and $t - 1$. For vertex i on affine body b , these two positions are computed from the same rest position $\bar{\mathbf{x}}_i$ using Equation (2). We vary $\bar{\mathbf{x}}_i$ while keeping each frame's translation and affine transformation at their converged values. The resulting world-space position changes are

$$d\mathbf{x}_{t,i} = \mathbf{A}_{t,b} d\bar{\mathbf{x}}_i, \quad d\mathbf{x}_{t-1,i} = \mathbf{A}_{t-1,b} d\bar{\mathbf{x}}_i.$$

A rest vertex identifies the same material point at both frames. Moving it can therefore change the relative sliding displacement even when the two affine transforms are held fixed. Its previous-frame position also determines the interpolation weights, contact-plane basis, and normal force used by the friction law. The shape derivative must include these changes together with the current point displacement.

Let $\mathbf{H}_{t,c}^{\text{fr},x} = \partial \mathbf{g}_{t,c}^{\text{fr},x} / \partial \mathbf{x}_t^c$ and $\mathbf{C}_{t,c}^{\text{fr},x} = \partial \mathbf{g}_{t,c}^{\text{fr},x} / \partial \mathbf{x}_{t-1}^c$ be the derivatives with respect to the current and previous point positions. The block $\mathbf{C}_{t,c}^{\text{fr},x}$ includes the changes in the interpolation weights, tangent basis, and normal force. The rest-shape variation also changes $\mathbf{K}_{t,c}$. Applying the product rule to $\mathbf{g}_{t,c}^{\text{fr},y} = \mathbf{K}_{t,c}^\top \mathbf{g}_{t,c}^{\text{fr},x}$ therefore gives all three contributions:

$$\begin{aligned} d\mathbf{g}_{t,c}^{\text{fr},y} &= (d\mathbf{K}_{t,c})^\top \mathbf{g}_{t,c}^{\text{fr},x} \\ &\quad + \mathbf{K}_{t,c}^\top \mathbf{H}_{t,c}^{\text{fr},x} d\mathbf{x}_t^c \\ &\quad + \mathbf{K}_{t,c}^\top \mathbf{C}_{t,c}^{\text{fr},x} d\mathbf{x}_{t-1}^c. \end{aligned} \quad (47)$$

These terms capture the changed force map and the current- and previous-frame contact geometry at fixed simulated DoFs. Their contribution to $\mathbf{G}_t$ complements the earlier-state dependence propagated by the history update in Section 4.5. Appendix A.1 provides the friction law in terms of contact-point positions and its derivatives.

Direct dependence of a surface loss. A loss evaluated on an affine surface also depends on rest geometry through $\mathbf{x}_{t,i} = \mathbf{J}_i \mathbf{q}_{t,b}$. For a loss on surface positions at one frame t , holding the simulation

DoFs fixed gives the direct contribution

$$\nabla_{\bar{\mathbf{x}}_i} L|_{\mathbf{y}_t \text{ fixed}} = \mathbf{A}_{t,b}^\top \nabla_{\mathbf{x}_{t,i}} L. \quad (48)$$

For losses on several frames, these direct contributions are summed over the corresponding frames and included in ℓ_θ . They are added to the contributions through the frame equations in Equation (30). Garment Hanger (Figure 4) illustrates both contributions. Its support objective depends on the relative positions of the cloth straps and hanger shoulders. Changing the rest surface therefore affects the loss directly through the surface map and indirectly through the coupled motion, including frictional contact. Both contributions enter the rest-shape gradient.

5.3 Initial-state gradients

Following Section 4.4, we include the selected components of an affine body's initial configuration $\mathbf{q}_{0,b}$ and initial velocity $\dot{\mathbf{q}}_{0,b}$ in θ . Their parameter derivatives enter $\mathbf{G}_t$ and determine the initial-state components of $\Delta \bar{\theta}^{(t)}$ in Equation (33). For these partial derivatives, the rest geometry, prescribed accelerations, and computed states $\mathbf{y}_1, \dots, \mathbf{y}_T$ remain unchanged. Only the selected initial values vary.

The first-frame inertial derivatives in Section 4.4 use the mass block $\mathbf{M}_b$ for affine body b :

$$\frac{\partial \mathbf{g}_{1,b}^{\text{in}}}{\partial \mathbf{q}_{0,b}} = -\mathbf{M}_b, \quad \frac{\partial \mathbf{g}_{1,b}^{\text{in}}}{\partial \dot{\mathbf{q}}_{0,b}} = -h\mathbf{M}_b. \quad (49)$$

The initial configuration also appears in the second-frame BDF1 predictor, $\hat{\mathbf{q}}_{2,b} = 2\mathbf{q}_{1,b} - \mathbf{q}_{0,b} + h^2 \mathbf{a}_{2,b}$. Differentiating $\mathbf{g}_{2,b}^{\text{in}} = \mathbf{M}_b(\mathbf{q}_{2,b} - \hat{\mathbf{q}}_{2,b})$ with respect to $\mathbf{q}_{0,b}$ gives

$$\frac{\partial \mathbf{g}_{2,b}^{\text{in}}}{\partial \mathbf{q}_{0,b}} = \mathbf{M}_b. \quad (50)$$

The first-frame derivatives contribute to $\mathbf{G}_1$, and the second-frame derivative contributes to $\mathbf{G}_2$. Let $\lambda_{t,b}$ contain the twelve entries of the coupled adjoint λ_t corresponding to body b . Substituting these derivatives into Equation (33) gives the inertial contributions to the initial-state gradients:

$$\begin{aligned} \Delta \bar{\mathbf{q}}_{0,b} &= \mathbf{M}_b \lambda_{1,b} - \mathbf{M}_b \lambda_{2,b}, \\ \Delta \bar{\dot{\mathbf{q}}}_{0,b} &= h\mathbf{M}_b \lambda_{1,b}. \end{aligned} \quad (51)$$

For a one-frame trajectory, the term containing $\lambda_{2,b}$ is omitted.

Friction provides an additional initial-configuration derivative. At the first frame, its previous-frame positions are the initial surface positions $\mathbf{x}_{0,i} = \mathbf{J}_i \mathbf{q}_{0,b}$. Changing $\mathbf{q}_{0,b}$ changes these positions and the interpolation weights, tangent basis, and normal force computed from them. We compute this derivative using Equation (37) with $t = 1$. Since $\mathbf{q}_{0,b}$ is an input parameter, this derivative contributes to $\mathbf{G}_1$ and the parameter-gradient update, rather than an earlier-frame vector $\mathbf{b}_0$.

For the velocity-reset updates used to approximate quasi-static behavior in Section 4.4, the first-frame predictor is $\hat{\mathbf{q}}_{1,b} = \mathbf{q}_{0,b} + h^2 \mathbf{a}_{1,b}$. Its inertial derivatives are $-\mathbf{M}_b$ with respect to $\mathbf{q}_{0,b}$ and zero with respect to $\dot{\mathbf{q}}_{0,b}$. Subsequent predictors use the previous solved configuration without an explicit $\mathbf{q}_{0,b}$ term. In this mode, the inertial contributions are $\Delta \bar{\mathbf{q}}_{0,b} = \mathbf{M}_b \lambda_{1,b}$ and $\Delta \bar{\dot{\mathbf{q}}}_{0,b} = \mathbf{0}$. The first-frame friction contribution is included as described above.

5.4 Joint targets and forces

Revolute and prismatic joint targets and forces provide actuation parameters alongside rest shapes and initial states. Their reference geometry and driving stiffness remain fixed during control optimization. Let $u_{j,t}$ denote the scalar target, force, or torque applied to joint j at frame t , and let $\mathbf{g}_{t,j}^{\text{joint}}$ denote its contribution to the frame energy gradient. Equation (33) gives this joint’s contribution $\Delta \bar{u}_{j,t}$ to the control gradient:

$$\Delta \bar{u}_{j,t} = - \left(\frac{\partial \mathbf{g}_{t,j}^{\text{joint}}}{\partial u_{j,t}} \right)^T \lambda_t. \quad (52)$$

Although the local control derivative involves only the two joint-connected bodies, its product with the coupled adjoint includes their response through contact with the rest of the scene. Joint targets and forces can therefore be optimized for objectives on interacting objects, using the same backward system as shape and initial-state parameters.

Actuation energies and parameter derivatives. For the local formulas, we omit the joint and frame indices. Let a be the applied target, f the joint force (a linear force for a prismatic joint and a torque for a revolute joint), and k the effective driving stiffness. The affine DoFs of the two bodies connected by the joint are denoted by $\mathbf{q}_p$ and $\mathbf{q}_c$. The vector $\mathbf{q} = [\mathbf{q}_p^T, \mathbf{q}_c^T]^T$ collects both bodies’ DoFs, with $\nabla = \nabla_{\mathbf{q}}$. For a prismatic joint, let ξ denote the relative displacement between the two joint attachment points, projected onto the joint axis of body c in world space. Its energy and control derivatives are

$$\begin{aligned} \Phi^{\text{pri}} &= \frac{1}{2} k (\xi - a)^2 - f \xi, \\ \mathbf{g}^{\text{pri}} &= \nabla \Phi^{\text{pri}} = [k(\xi - a) - f] \nabla \xi, \\ \frac{\partial \mathbf{g}^{\text{pri}}}{\partial a} &= -k \nabla \xi, \quad \frac{\partial \mathbf{g}^{\text{pri}}}{\partial f} = -\nabla \xi. \end{aligned} \quad (53)$$

The gradient $\nabla \xi$ also accounts for how the world-space joint axis of body c changes with $\mathbf{q}_c$. The quadratic term penalizes displacement away from the target a , whereas $-f\xi$ specifies the applied joint force along the joint coordinate. Targets and joint forces therefore enter the same frame equation through different parameter derivatives.

For a revolute joint, two scalar quantities s_r and c_r computed from the bodies’ transformed reference directions define the relative angle $\alpha = \text{atan2}(s_r, c_r)$. For target a , define

$$S_r = s_r \cos a - c_r \sin a, \quad C_r = c_r \cos a + s_r \sin a. \quad (54)$$

These quantities compare the relative orientation with the target. For rigid rotations with orthonormal reference directions and aligned joint axes, they reduce to $S_r = \sin(\alpha - a)$ and $C_r = \cos(\alpha - a)$. The squared sine vanishes both at the target and a half-turn away. The cosine term below gives the half-turn a positive penalty. The periodic target energy is

$$\begin{aligned} \Phi^{\text{rev}} &= \frac{1}{2} k [S_r^2 + w_c (1 - C_r)^2] - f \alpha, \\ \mathbf{g}^{\text{rev}} &= \nabla \Phi^{\text{rev}} \\ &= k [S_r \nabla S_r - w_c (1 - C_r) \nabla C_r] - f \nabla \alpha. \end{aligned} \quad (55)$$

Here $w_c = 0.02$ is the fixed relative weight of the cosine penalty, not an optimized control. The joint energy uses s_r and c_r computed directly from the affine-transformed joint reference directions. Their

derivatives account for changes in both the lengths and orientations of these directions. The angle derivatives used by the torque term $-f\alpha$ are given in Appendix A.3.

The partial derivatives with respect to the applied target and joint force are

$$\begin{aligned} \frac{\partial \mathbf{g}^{\text{rev}}}{\partial a} &= k [(-C_r - w_c (1 - C_r)) \nabla S_r \\ &\quad + (w_c - 1) S_r \nabla C_r], \\ \frac{\partial \mathbf{g}^{\text{rev}}}{\partial f} &= -\nabla \alpha. \end{aligned} \quad (56)$$

All control derivatives are evaluated at the converged frame states. At fixed control values, differentiating the state gradients gives the full joint Hessians $\nabla^2 \Phi^{\text{pri}}$ and $\nabla^2 \Phi^{\text{rev}}$. These include the second derivatives of the joint coordinates and of the potential terms linear in the joint force. Appendix A.3 gives the geometric definitions of ξ , s_r , and c_r , their first and second coordinate derivatives, and the full Hessians in Equation (91).

6 GPU Backward and Reusable Interface

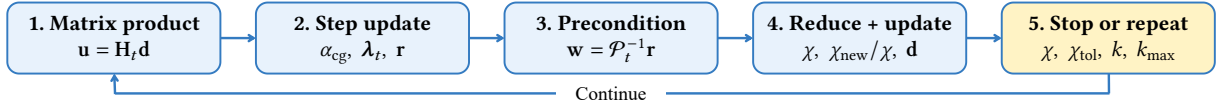
Compact coordinates and GPU parallelism reduce different parts of the backward cost. The coupled adjoint solve captures interactions across the scene, while local derivative products expose parallel work over elements, surface triangles, contacts, and joints. Conditional CUDA Graphs reduce repeated CPU coordination during the solve, and parallel accumulation returns gradients to earlier states and physical parameters (Figure 5). A PyTorch interface makes this computation reusable across task-specific designs and objectives.

The adjoint system is assembled from each frame’s converged FEM and affine states. Equation (2) recovers the affine surface positions. The backward pass evaluates contact derivatives using the same contact pairs and distance formulas as the forward solve at this converged state. Previous-frame friction weights, contact-plane bases, and normal forces retain the history dependence derived in Section 4.5. StiffGIPC’s parallel global Hessian assembly [Huang et al. 2025a] combines element, body, collision, friction, and joint contributions, including FEM-affine cross terms. The resulting Hessian is assembled once per reverse frame and remains fixed throughout its PCG solve, avoiding a repeat of the nonlinear forward iterations.

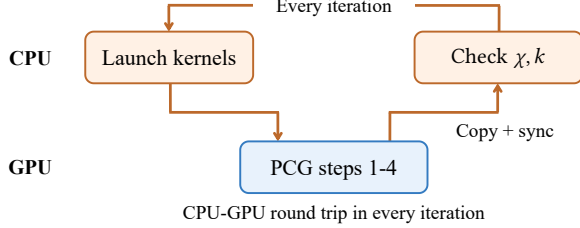
6.1 PCG with conditional CUDA Graphs

The adjoint in Equation (31) uses the same frame Hessian as the converged forward equations, allowing StiffGIPC’s highly parallel GPU PCG solver [Huang et al. 2025a] to serve both forward and backward computations. Compact affine coordinates reduce PCG’s numerical workload, making repeated CPU kernel launches and convergence synchronization a larger fraction of solve time. The stricter backward residual tolerance can amplify this cost over more iterations. To address this, we build a PCG pipeline managed by a conditional CUDA Graph, based on the PCG recurrence kernels [Baraff and Witkin 1998], their execution dependencies, and the convergence checks (Figure 5). Once launched from the CPU, the graph initializes PCG and then repeats its iterations and convergence checks entirely on the GPU until convergence or the iteration limit is reached, eliminating per-iteration kernel launches and CPU-GPU synchronization.

(a) One PCG iteration after initialization



(b) CPU-controlled iterations



(c) Conditional CUDA Graph

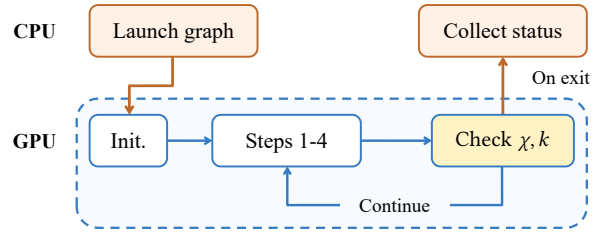


Fig. 5. PCG computation and execution control. (a) Five stages of one iteration (Algorithm 2). (b) A host-controlled loop retrieves a convergence value after each GPU iteration. (c) A conditional CUDA Graph keeps iteration and stopping on the GPU. Frame restoration, Hessian assembly, and gradient accumulation remain outside the graph. Both execution modes use the same PCG recurrence and stopping criterion.

Algorithm 2 PCG with Conditional CUDA Graphs

Input: Current $\mathbf{H}_t$, $\mathbf{b}_t$, $\mathcal{P}_t$, relative target η_{cg} , iteration limit k_{max}

Output: Candidate λ_t and convergence status

CPU: construct or reuse CUDA Graph

- 1: Use $\mathbf{H}_t$, $\mathbf{b}_t$, and $\mathcal{P}_t$ as inputs to the CUDA Graph
- 2: **if** no compatible CUDA Graph exists **then**
- 3: Construct a CUDA Graph from the GPU kernels of PCG
- 4: Connect dependent kernels
- 5: Add conditional loop node that reads GPU-updated continue flag
- 6: **end if**

7: Launch the CUDA Graph once from the CPU

GPU: PCG initialization within CUDA Graph

- 8: $\lambda_t \leftarrow \mathbf{0}$; $\mathbf{r} \leftarrow \mathbf{b}_t - \mathbf{H}_t \lambda_t$; $\mathbf{w} \leftarrow \mathcal{P}_t^{-1} \mathbf{r}$; $\mathbf{d} \leftarrow \mathbf{w}$
- 9: $\chi \leftarrow \mathbf{r}^T \mathbf{w}$; $\chi_{tol} \leftarrow \eta_{cg} |\chi|$; $k \leftarrow 0$
- 10: Set the continue flag from $|\chi| > \chi_{tol} \wedge k < k_{max}$ on the GPU

GPU: PCG iterations within CUDA Graph

- 11: **while** the graph's continue flag is true **do**
- 12: $\mathbf{u} \leftarrow \mathbf{H}_t \mathbf{d}$; $\alpha_{cg} \leftarrow \chi / (\mathbf{d}^T \mathbf{u})$
- 13: $\lambda_t \leftarrow \lambda_t + \alpha_{cg} \mathbf{d}$; $\mathbf{r} \leftarrow \mathbf{r} - \alpha_{cg} \mathbf{u}$
- 14: $\mathbf{w} \leftarrow \mathcal{P}_t^{-1} \mathbf{r}$; $\chi_{new} \leftarrow \mathbf{r}^T \mathbf{w}$
- 15: $\mathbf{d} \leftarrow \mathbf{w} + (\chi_{new} / \chi) \mathbf{d}$; $\chi \leftarrow \chi_{new}$; $k \leftarrow k + 1$
- 16: A GPU kernel clears the continue flag if $|\chi| \leq \chi_{tol}$ or $k \geq k_{max}$
- 17: **end while**

CPU: read final convergence status

- 18: Read χ and k
- 19: **return** λ_t

For each reverse frame, we solve the assembled adjoint system using a reusable CUDA Graph (Algorithm 2). Its inputs $\mathbf{H}_t$, $\mathbf{b}_t$, and preconditioner $\mathcal{P}_t$ are updated between frames and stay fixed during each solve. The graph preserves the dependencies between PCG kernels, while GPU convergence checks govern its conditional loop, so the same graph can run a different number of iterations. It can still be reused when contact sets change, as long as the kernel configuration and storage stay compatible. This reuse spreads graph construction across frames and removes per-iteration CPU coordination, so compact FEM-affine adjoint systems can benefit from their

reduced numerical workload. The GPU convergence check compares the preconditioned residual measure $|\chi^{(k)}| = |(\mathbf{r}^{(k)})^T \mathcal{P}_t^{-1} \mathbf{r}^{(k)}|$ with the target $\chi_{tol} = \eta_{cg} |\chi^{(0)}|$, set at the start of each solve. The conditional loop stops when this target is reached or the iteration count reaches k_{max} . The default backward settings are $\eta_{cg} = 10^{-14}$ and $k_{max} = 3 \times 10^4$. Only an adjoint whose final residual measure meets the target is accepted for the gradient accumulation described next.

6.2 GPU-parallel gradient accumulation

The parameter-gradient and earlier-frame contributions in Equations (33) and (32) share the coupled adjoint λ_t . Its entries contain the influence of interactions across the scene, but each derivative product accesses the DoFs of a local element, contact and joint. Surface triangles provide the local mass and volume derivatives for affine shape design. This locality exposes parallel work even when a body has a compact motion state.

The parameter-gradient update is organized as parallel GPU tasks over FEM elements, contact pairs, joints, and affine surface triangles. Each task computes one local contribution to the gradient of a selected parameter. For affine rest-shape gradients, tasks assigned to individual surface triangles compute the mass-matrix and volume derivative contributions. Each task reads only the adjoint entries for the DoFs involved and evaluates their inner product with the corresponding parameter derivative. With the frame adjoint λ_t solved, the local derivative products are independent and evaluated in parallel. Atomic additions combine contributions to the same parameter-gradient entry. Let $\mathbf{g}_t^{[i]}$ denote a local contribution to $\mathbf{g}_t$. The total loss gradient for parameter θ_k is

$$\bar{\theta}_k = (\ell_\theta)_k - \sum_{t=1}^T \sum_i \left(\frac{\partial \mathbf{g}_t^{[i]}}{\partial \theta_k} \right)^T \lambda_t^{[i]}. \quad (57)$$

Here $(\ell_\theta)_k$ is the direct loss derivative with all solved states fixed. This decomposition sums the frame contributions in Equation (33) without assembling the global parameter-derivative matrix $\mathbf{G}_t$. The

required derivatives come from Section 5, including inertia in Equation (43), initial states in Equations (49)–(50), and joint controls in Equation (52). Material derivatives are given in Appendix A.2.

Friction follows the same decomposition. The mapped adjoint $\mathbf{K}_{t,i}\boldsymbol{\lambda}_t$ is computed once per vertex and reused by all contacts involving that vertex. For each contact, a separate GPU task computes one entry of

$$\mathbf{y}_{t,c} = \left(\frac{\partial \mathbf{g}_{t,c}^{\text{fr},x}}{\partial \mathbf{x}_{t-1}^c} \right)^\top \mathbf{K}_{t,c}\boldsymbol{\lambda}_t,$$

without assembling the contact derivative matrix. Entries are summed at mesh vertices, and the resulting sum is mapped to previous-frame DoFs (Equation (38)). Contacts sharing a vertex therefore apply $\mathbf{K}_{t-1,i}^\top$ only once. For a free FEM vertex i , $-\mathbf{y}_t^i$ is accumulated directly into the corresponding entries of $\mathbf{b}_{t-1}$. For affine bodies, the contributions $-\mathbf{K}_{t-1,i}^\top \mathbf{y}_t^i$ are summed over their vertices into the body’s twelve entries in $\mathbf{b}_{t-1}$. The per-vertex vectors $\mathbf{y}_t^i$ are also reused for the rest-shape gradient contribution arising from previous-frame contact positions (Equation (47)). Motion changes propagate through the earlier-frame right-hand sides, and the remaining rest-shape terms reuse the point gradients and Hessians assembled before affine conversion, without adding motion DoFs.

The inertial history terms in Equations (35)–(36) are likewise evaluated independently across vertices and bodies. All contributions to an earlier right-hand side must be accumulated before its adjoint solve. Parallelism therefore lies within each reverse frame, while frames follow the dependency order in Section 4.3. For a fixed trajectory and scalar loss, all selected parameters share the same frame adjoint, so adding parameters requires local derivative computation rather than additional coupled solves.

6.3 A reusable interface for inverse problems

The parameter map converts user-defined design variables into physical parameters such as rest geometry, material properties, or joint controls. The output map converts simulated DoFs into quantities used to evaluate the loss, such as surface vertex positions. PyTorch differentiates both maps and the loss, while DiffGIPC computes derivatives through the coupled simulation. A loss evaluated on one object can therefore guide the optimization of another object’s parameters through their physical interactions.

Let $\mathbf{z} \in \mathbb{R}^{n_z}$ contain the optimizer variables, ϕ map them to physical parameters $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$, and $\mathcal{O}$ form the quantities used by the scalar loss ℓ from the converged states:

$$\begin{aligned} \boldsymbol{\theta} &= \phi(\mathbf{z}), \\ \mathbf{o} &= \mathcal{O}(\mathbf{y}_1, \dots, \mathbf{y}_T, \boldsymbol{\theta}), \\ L &= \ell(\mathbf{o}, \boldsymbol{\theta}). \end{aligned} \quad (58)$$

The parameter map imposes task-specific constraints or shares design variables across physical parameters, so n_z need not equal n_θ . The output map combines quantities from different objects and frames without changing the coordinates of the coupled solve.

PyTorch differentiates the loss and output map to obtain

$$\boldsymbol{\ell}_o = \nabla_{\mathbf{o}} \ell, \quad \boldsymbol{\ell}_t = \left(\frac{\partial \mathcal{O}}{\partial \mathbf{y}_t} \right)^\top \boldsymbol{\ell}_o. \quad (59)$$

For affine surface points, $\mathbf{J}_i^\top$ maps point gradients to body DoFs as in Equation (18), whereas FEM positions use the identity map. Selecting outputs sets the direct loss gradients, not the dependencies retained by the adjoint. A frame absent from $\mathcal{O}$ has $\boldsymbol{\ell}_t = \mathbf{0}$ but can still receive contributions from later frames. The output map and loss may also depend explicitly on physical parameters. With the simulated DoFs fixed, their contribution is

$$\boldsymbol{\ell}_\theta = \nabla_{\boldsymbol{\theta}} \ell|_{\mathbf{o} \text{ fixed}} + \left(\frac{\partial \mathcal{O}}{\partial \boldsymbol{\theta}} \right)^\top \boldsymbol{\ell}_o. \quad (60)$$

For example, changing an affine rest surface moves its world-space points even at fixed motion DoFs. This explicit geometric effect is distinct from the change in simulated motion accounted for by the adjoint. The vectors $\boldsymbol{\ell}_t$ and $\boldsymbol{\ell}_\theta$ initialize the right-hand sides and parameter gradient in Section 4.3. The GPU backward pass then supplies the physical gradient, which PyTorch maps to the optimizer variables:

$$\begin{aligned} \bar{\boldsymbol{\theta}} &= \boldsymbol{\ell}_\theta - \sum_{t=1}^T \mathbf{G}_t^\top \boldsymbol{\lambda}_t, \\ \nabla_{\mathbf{z}} L &= \left(\frac{\partial \phi}{\partial \mathbf{z}} \right)^\top \bar{\boldsymbol{\theta}}. \end{aligned} \quad (61)$$

Task parameterizations and objectives can therefore change without changing the coupled backward equations. In Garment Hanger (Figure 4), two shared shoulder-control scales specify the rest surface, while the loss measures support using both hanger and cloth positions. PyTorch differentiates the shape parameterization and geometric loss. DiffGIPC combines the direct surface dependence with the gradient through their coupled motion, retaining twelve motion DoFs for the hanger. The same composition applies to the physical parameter families listed in Table 7 (Appendix B). Additional physical models enter through their local derivatives, preserving the distinction between task design and coupled simulation.

7 Evaluation

Our evaluation examines the parameter-to-output dependencies supported by DiffGIPC and the cost of forward and backward evaluations across full-space and affine motion representations. The shared full-space FEM benchmarks from DiffIPC [Huang et al. 2024c] cover initial motion, rest design, and material or friction fitting at their original low mesh resolutions. In these benchmarks, both methods use full-space FEM. Further experiments exercise cloth design, affine shape and placement, and articulated control, including mixed-coordinate inverse problems. We first compare optimization outcomes and evaluation times on the shared FEM benchmarks, then demonstrate the coupled design and control tasks. Gradient validation assesses agreement between analytic gradients and finite differences, while studies of motion representation, resolution, and GPU execution explain the optimization cost.

For the shared benchmarks, DiffGIPC runs on an NVIDIA GeForce RTX 4090 and DiffIPC on an Intel Core i9-13900K. Both use double-precision arithmetic and the same convergence tolerances. Appendix B summarizes the simulation settings in Table 6 and the differentiated parameter families in Table 7. A forward evaluation simulates a trajectory to the task’s observation horizon. A backward evaluation computes its loss gradient over that trajectory. N_{fwd} counts

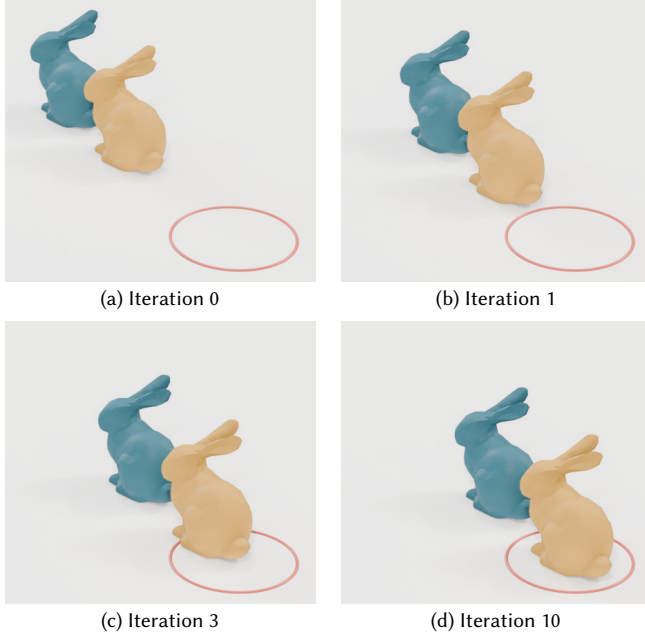


Fig. 6. Collision-mediated initial-velocity optimization. The first bunny’s initial velocity is adjusted so that the second bunny’s horizontal centroid reaches its target. The panels compare frame 40 before optimization and after 1, 3, and 10 iterations.

forward evaluations, including line-search trials and verification runs, while N_{bwd} counts complete gradient evaluations used for optimization. Let $\bar{t}_f$ and $\bar{t}_b$ denote their mean times. The combined mean time is $\bar{t}_{f+b} = \bar{t}_f + \bar{t}_b$. The accumulated evaluation time is $T_{\text{eval}} = N_{\text{fwd}}\bar{t}_f + N_{\text{bwd}}\bar{t}_b$. In the loss definitions below, $\mathbf{x}_{t,i}$ is the world-space position of vertex i at observation frame t , and a star denotes a prescribed target held fixed during optimization. Each task specifies its observation endpoint T or selected frame set. Unsubscripted norms are Euclidean norms, and $\|\cdot\|_F$ denotes the Frobenius norm for matrices.

7.1 Comparison with DiffIPC

The shared FEM benchmarks establish the framework’s coverage in full-space FEM before introducing affine design and control. They progress from contact-driven motion to rest-shape design and parameter fitting, changing both the optimized input and the observed response. Figure 11 compares their optimization histories, and Table 1 reports the corresponding evaluation times. We relate each task’s objective to its physical outcome before examining the computational cost.

Bunny collision. We optimize a shared initial velocity for the vertices of one bunny so that collision moves the other bunny to a target. The objective uses only the second bunny’s volume-weighted horizontal center after 40 frames:

$$L_{\text{bunny}} = \|\mathbf{c}_{xz}(\mathbf{x}_T^{(2)}) - \mathbf{c}_{xz}^*\|^2. \quad (62)$$

Here $\mathbf{c}_{xz}$ is the volume-weighted horizontal (x, z) center of the second bunny and $T = 40$. Because the loss observes only that body, the optimized velocity affects it through collision. Figure 6 shows

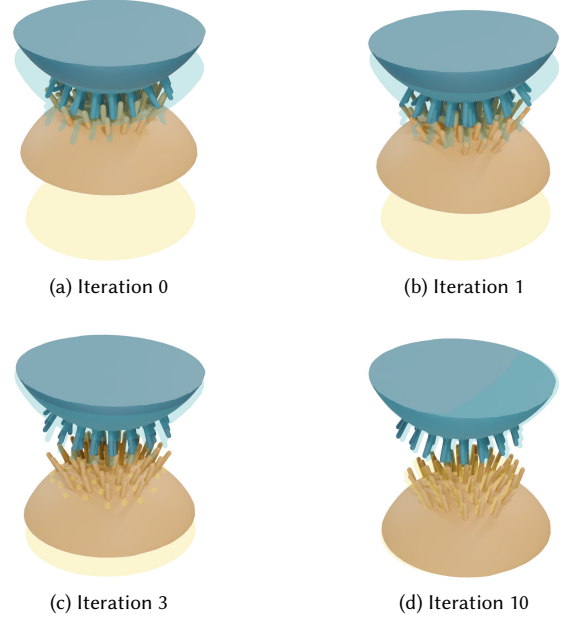


Fig. 7. Trajectory matching through contact between two deformable bodies. The first body’s initial velocity is optimized to reproduce the target motion. Opaque surfaces show the simulation and translucent surfaces the target at frame 100, before optimization and after 1, 3, and 10 iterations.

the final position approaching the target. Section 7.4 revisits this task across motion representations.

Tentacle trajectory matching. The first deformable body’s shared initial velocity is optimized while the second body’s initial velocity remains fixed. Let $\mathbf{c}_t^{\text{tent}}$ be the first body’s volume-weighted position, using fixed rest-tetrahedron weights. Its trajectory-matching objective is

$$L_{\text{tentacle}} = 200h \sum_{t=1}^T \|\mathbf{c}_t^{\text{tent}} - \mathbf{c}_t^{\text{tent},*}\|^2, \quad (63)$$

where $T = 100$ and $\mathbf{c}_t^{\text{tent},*}$ is the prescribed target. Thus observations throughout the trajectory contribute to the initial-velocity gradient through motion and friction history; Figure 7 shows the endpoint.

Loaded Support design. Selected rest vertices of a FEM support are optimized under downward loading, with its supports fixed. At the end of each 20-frame simulation, the squared-stress term is $L_{\text{stress}} = 10^{-8} \sum_{e \in \mathcal{T}_d} V_e \|\mathbf{P}_e\|_F^2$. Here $\mathcal{T}_d$ is the evaluated tetrahedron set, V_e is the rest volume, and $\mathbf{P}_e$ is the first Piola stress evaluated at $T = 20$. Rest-volume weighting makes larger stressed regions contribute proportionally more. A surface-smoothing penalty discourages local irregularities in the designed rest geometry. The optimized loss is $L_{\text{support}} = L_{\text{stress}} + L_{\text{sm}}$. The smoothing set $\mathcal{V}_{\text{sm}}$ contains surface vertices selected in rest coordinates. For each $i \in \mathcal{V}_{\text{sm}}$, let $\mathcal{N}_{\text{sm}}(i)$ be its surface-edge neighbors that also belong to $\mathcal{V}_{\text{sm}}$. Using the current rest positions $\bar{\mathbf{x}}_i$, the penalty is

$$L_{\text{sm}} = \sum_{i \in \mathcal{V}_{\text{sm}}} \left\| \frac{\sum_{j \in \mathcal{N}_{\text{sm}}(i)} (\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j)}{\sum_{j \in \mathcal{N}_{\text{sm}}(i)} \|\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j\|} \right\|^2. \quad (64)$$

Table 1. Mean time per forward and backward evaluation (seconds) on the shared FEM benchmarks. Forward + Backward gives the sum of the two per-evaluation means.

Example	System	Forward						Backward				Forward + Backward
		Linear	CCD	Hessian	Search	Other	Total	Hessian	Linear	Other	Total	
Bunny collision (Fig. 6)	DiffIPC	4.5	10	4.6	1.1	1.9	22	1.4	1.1	0.44	2.9	25
	DiffGIPC	0.65	0.044	0.037	0.017	0.002	0.75	0.0069	0.22	0.0057	0.24	0.98
	Speedup	6.9×	232.0×	123.9×	67.7×	963.8×	30.0×	204.7×	4.9×	76.3×	12.5×	25.8×
Tentacle matching (Fig. 7)	DiffIPC	140	14	120	66	11	350	7.4	11	2.8	21	370
	DiffGIPC	8.1	2.4	2.2	1.4	0.011	14	0.04	1	0.024	1.1	15
	Speedup	16.9×	5.9×	55.3×	46.7×	1037.5×	24.6×	184.8×	10.9×	120.7×	19.7×	24.3×
Loaded Support (Fig. 8)	DiffIPC	22	0.60	3.2	1.1	11	38	2.9	12	0.49	16	54
	DiffGIPC	0.93	0.02	0.014	0.0017	0.00051	0.97	0.011	0.85	0.0066	0.87	1.8
	Speedup	23.8×	29.5×	232.1×	653.7×	21172.2×	39.3×	266.4×	14.7×	74.2×	18.3×	29.4×
Material fitting (Fig. 9)	DiffIPC	0.89	0.029	0.69	0.028	0.61	2.2	0.35	0.53	0.057	0.94	3.2
	DiffGIPC	0.015	0.0021	0.0042	0.00064	0.0032	0.025	0.0026	0.023	0.0016	0.028	0.053
	Speedup	60.3×	14.3×	161.8×	43.5×	190.7×	90.1×	135.0×	22.8×	36.0×	34.2×	60.7×
Friction optimization (Fig. 10)	DiffIPC	17	22	21	8.1	6.8	76	2.5	1.9	1.2	5.6	81
	DiffGIPC	3.8	1.4	1.3	0.7	0.0093	7.2	0.037	0.36	0.039	0.44	7.7
	Speedup	4.6×	16.3×	15.7×	11.7×	734.8×	10.5×	66.3×	5.3×	31.8×	12.8×	10.6×

Vertices without neighbors in this set contribute zero. This term measures each selected vertex’s deviation from its neighbors’ mean position, normalized by the mean incident edge length. Figure 8 shows how the redesigned geometry redistributes stress under the same loading and color scale. The loss histories in Figure 11 reach similar final objective levels for both methods. Their intermediate jumps occur when remeshing changes the stress discretization. Gradients are evaluated on each current fixed mesh, with feasible rest-shape updates selected by backtracking. The task demonstrates rest-shape design through the combined changes in geometry, element volume, and elastic response.

Material identification. A homogeneous FEM cube is compressed by prescribed displacements on its two opposing faces. Its material

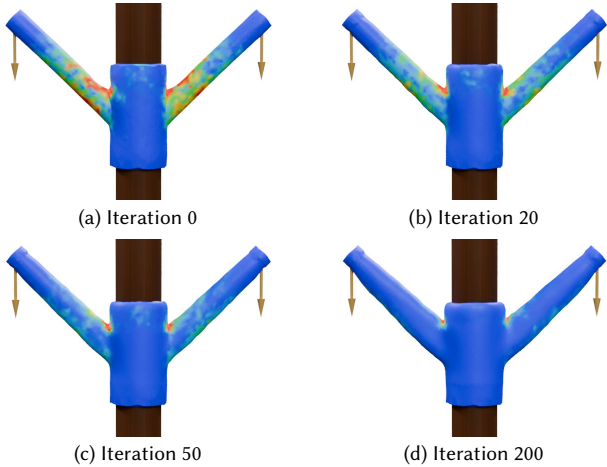


Fig. 8. Rest-shape optimization of a loaded FEM support. Under fixed constraints and downward loading, the objective minimizes squared first-Piola stress. The panels show the deformed support at frame 20 before optimization and after 20, 50, and 200 iterations.

response is fitted to a fixed reference deformation using surface-vertex positions. Two shared logarithmic variables, $z_{\text{vol}} = \log \lambda_{\text{Lamé}}$ and $z_{\text{stretch}} = \log \mu_{\text{Lamé}}$, parameterize the positive Lamé inputs $\lambda_{\text{Lamé}}$ and $\mu_{\text{Lamé}}$. They determine the deformation coefficients of the stable Neo-Hookean energy as

$$\lambda = e^{z_{\text{vol}}} + e^{z_{\text{stretch}}}, \quad \mu = e^{z_{\text{stretch}}}. \quad (65)$$

The same coefficient pair is assigned to every tetrahedron. The loss is $L_{\text{mat}} = 5 \times 10^4 \sum_{i \in \mathcal{M}} \|\mathbf{x}_{T,i} - \mathbf{x}_i^*\|^2$. Here $\mathcal{M}$ contains prescribed vertex markers on the cube’s face in the rest configuration. The marker indices remain unchanged during optimization, so $\mathbf{x}_{T,i}$ and $\mathbf{x}_i^*$ refer to the simulated and reference positions of the same vertex, not to a nearest-point match. The factor 5×10^4 sets the numerical scale of the squared-position loss, which is summed rather than averaged over the markers. The timing configuration uses $T = 10$ simulation steps with $h = 0.1$ s, and the loss is evaluated only at the final step. Both Lamé input parameters are initialized to 4×10^5 . Both methods reach similar marker-position errors at the plotted scale. Figure 9 shows the separately rendered frame-19 configurations approaching the reference deformation. This task demonstrates fitting a shared material response from surface-position errors through the positive coefficient map.

Friction optimization. The bunny-slide friction coefficient is fitted to a target position at frame 100, with $h = 0.02$ s. The loss is $L_{\text{fr}} = 100(c_z(\mathbf{x}_T) - z_{\text{end}})^2$, where c_z is the bunny’s volume-weighted center coordinate, z_{end} is the slide endpoint, and $T = 100$. The coefficient changes from 0.5 to 0.1 (Table 6 in Appendix B). Figure 10 shows the corresponding change in sliding distance at the fixed observation time. This task isolates the coefficient-to-motion dependence: geometry and initial motion remain fixed, and the optimized friction changes the cumulative sliding response.

Together, the shared benchmarks test gradients through collision-transmitted motion, observations over time, loaded shape changes, material fitting and friction response. Table 1 measures the cost of

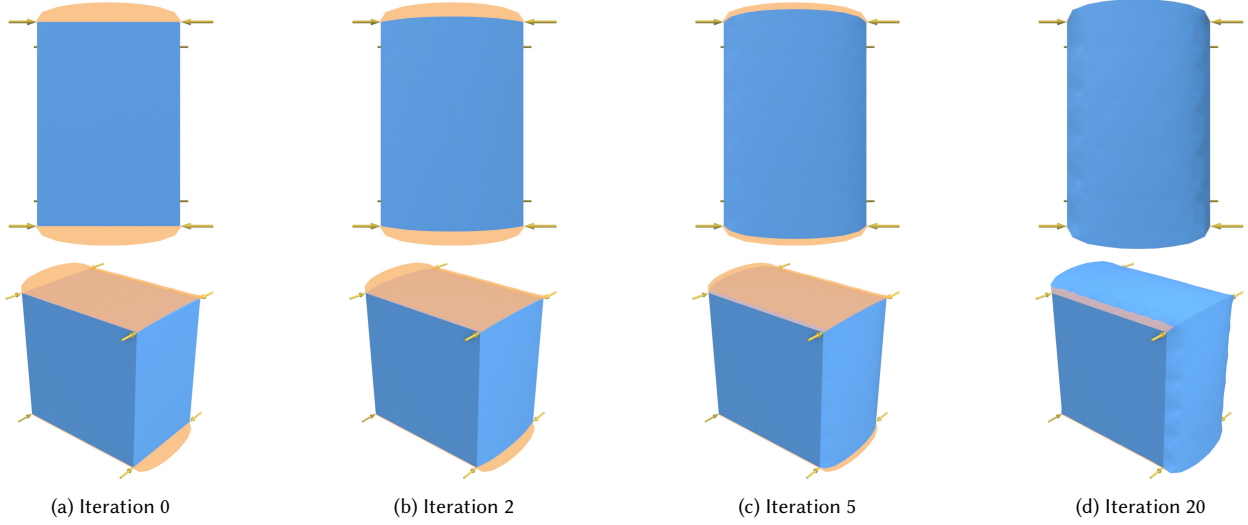


Fig. 9. Fitting the material response of a constrained cube to marker positions. Each column pairs side (top) and oblique (bottom) views at frame 19, before optimization and after 2, 5, and 20 iterations. Opaque surfaces denote the simulation, translucent surfaces the target, and arrows the Dirichlet constraints.

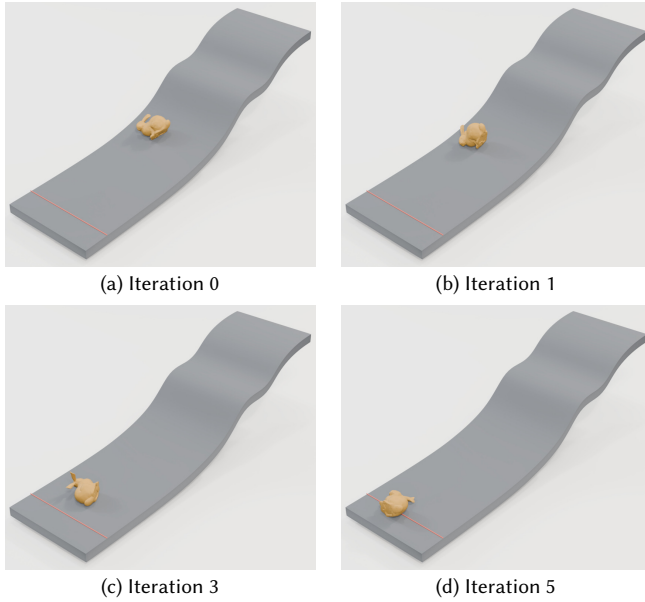


Fig. 10. Friction optimization on a curved slide. A bunny-slide friction coefficient is adjusted so that the bunny reaches its target position at frame 100. The panels show the unoptimized simulation and results after 1, 3, and 5 iterations.

that optimization capability with full-space FEM in both systems. The complete backward pass is 12.3–34.1 $\times$ faster, including Hessian assembly, the adjoint solve, and gradient accumulation. Combined forward and backward speedups range from 10.6 $\times$ for friction fitting to 60.9 $\times$ for material fitting, while forward speedups are 10.5–88.1 $\times$. These gains demonstrate efficient trajectory-gradient evaluation before introducing affine motion reduction. Section 7.4 examines that additional benefit, and Section 7.5 relates evaluation cost to complete optimization workloads.

7.2 Rest-shape design and stable placement

The design tasks separate the object being changed from the quantities used to judge the result. Cloth fitting maps a size parameter to a deforming garment, hanger design optimizes an affine surface through its effect on supported cloth, and stone placement adjusts initial positions while preserving shape. Together they demonstrate how rest geometry and initial-state parameters act through the same coupled trajectory gradient.

Cloth rest-scale fitting. A single optimizer variable scales the garment’s rest shape, with a fixed mannequin. This map changes many rest coordinates while preserving full-space cloth deformation. At the observation endpoint $T = 48$, the target drape defines $L_{\text{cloth}} = \sum_{i \in \mathcal{V}_{\text{cloth}}} \|\mathbf{x}_{T,i} - \mathbf{x}_i^*\|^2$, where $\mathcal{V}_{\text{cloth}}$ contains all garment vertices and $\mathbf{x}_i^*$ is the fixed target position of the corresponding material vertex. After 5 updates, the optimized drape approaches the target in Figure 12. The result demonstrates how a low-dimensional design parameter controls detailed simulated geometry through rest-shape derivatives.

Garment Hanger shoulder design. Each shoulder profile is controlled by a quadratic Bézier curve with two fixed endpoints and one movable middle control point. The design variables s_x and s_y adjust the middle control point’s horizontal and vertical positions. Starting from $(s_x, s_y) = (1, 1)$, decreasing s_x moves the point toward the hanger’s centerline, and decreasing s_y lowers it. The reflected curve uses the same pair, so both shoulders change symmetrically without moving their endpoints. The affine hanger retains twelve motion DoFs and supports a garment with friction coefficient 0.05.

At the terminal frame, R_s^{tip} and R_s^{strap} denote the horizontal positions of the hanger’s shoulder tip and the corresponding strap’s outer edge, respectively, on the left or right side $s \in \{L, R\}$. On each side, horizontal coordinates increase from the hanger’s centerline toward the shoulder tip. Their difference therefore measures how

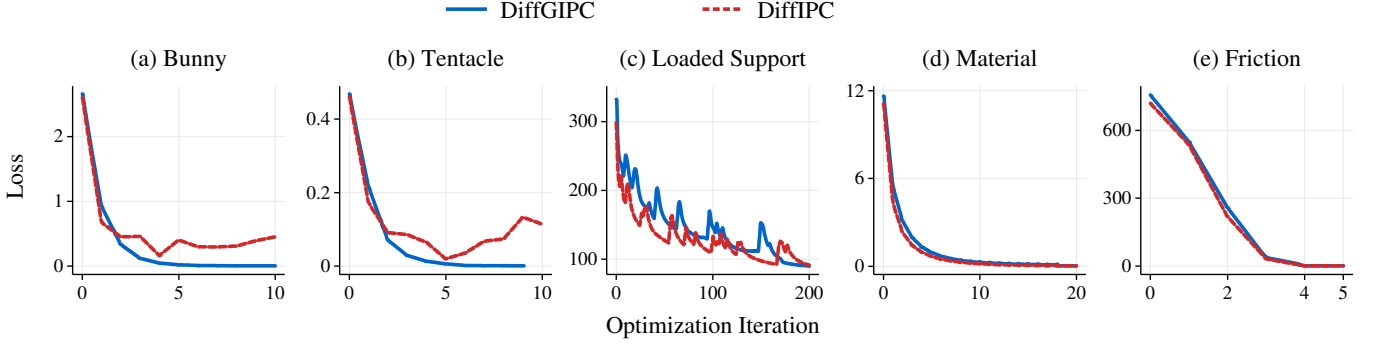


Fig. 11. Optimization histories for the shared FEM benchmarks. Blue and red denote DiffGIPC and DiffIPC, respectively. The horizontal axes count accepted parameter updates from the initial parameters (iteration 0). Loss scales are task-specific, and Loaded Support is remeshed between updates.



Fig. 12. Cloth rest-scale fitting on a fixed mannequin. The blue panels show the initial drape and results after 1, 3, and 5 updates. The orange panel shows the target.

far the hanger extends beyond the strap. The quantity d_s is the minimum three-dimensional distance between the selected strap and corresponding upper-shoulder vertices. For the fixed target margin $\delta = 0.205$ m, the support margin and loss are

$$m_s = R_s^{\text{tip}} - R_s^{\text{strap}} - \max(d_s - 0.01, 0),$$

$$L_{\text{hanger}} = 10^4 \sum_{s \in \{L, R\}} \max(\delta - m_s, 0)^2.$$

The gap penalty reduces the margin when the strap separates from the shoulder by more than 0.01 m. Both surfaces contribute to the loss, whose shape gradient includes direct geometric and coupled-motion contributions (Section 5.2).

Each optimization rollout ends at the first margin violation $m_s < \delta$ after the corresponding strap approaches within 0.01 m of its shoulder, or at frame 600. The optimized control-point scales $(s_x, s_y) = (0.8285, 0.8000)$ move the middle control points inward and downward. The resulting hanger retains both straps in two independent 600-frame checks (Figure 4). After each strap contacts the hanger, its support margin must remain nonnegative ($m_s \geq 0$) until frame 600. The larger target margin δ is used to guide optimization. Figure 16 shows optimization progress, and Section 7.4 compares a surface-matched FEM hanger.

Seven-stone placement. We optimize the horizontal initial positions of seven affine bodies with fixed shapes. Their detailed surfaces determine the contacts, while the motion system contains only 84 DoFs. Let b_{base} identify the bottom reference stone. At frame t , $d_{t,b}$

is the Euclidean distance between the horizontal center positions of stone b and that reference stone. With allowed distance δ , the placement objective is $L_{\text{stone}} = \sum_{t \in \mathcal{F}} \sum_{b \neq b_{\text{base}}} \max(d_{t,b} - \delta, 0)^2$, where $\mathcal{F}$ is the set of observation frames. The base is excluded from the inner sum, and only distances exceeding δ contribute. Optimization starts from horizontally offset stone placements. The first 50 updates use $\delta = 0.01$ to encourage the simulated centers of mass to align horizontally with that of the base stone. The remaining updates use $\delta = 0.15$ to allow a broader set of contact-supported configurations. Both stages take $\mathcal{F} = \{t^*\}$, where t^* is the first frame where $d_{t,b} > \delta$ for any non-base stone. If $d_{t,b} \leq \delta$ for every non-base stone at every simulated frame, the loss is zero. The optimizer adjusts the stones’ initial horizontal positions. The loss is computed from the horizontal distances between their simulated centers of mass and that of the base stone. Figure 1 shows updates 10, 50, and 78, with separate 1,000-frame forward simulations testing whether the resulting stacks remain stable. The task demonstrates contact-aware placement through affine initial-state gradients: detailed surface interactions guide the horizontal positions without changing the stones’ shapes.

7.3 Articulated control

Articulated tasks extend the coupled gradients from geometry and placement to actuation. Grasp-angle control tests joint-target tracking under contact, and cup catching optimizes joint commands for the motion of an interacting object. Dual-arm grasping optimizes

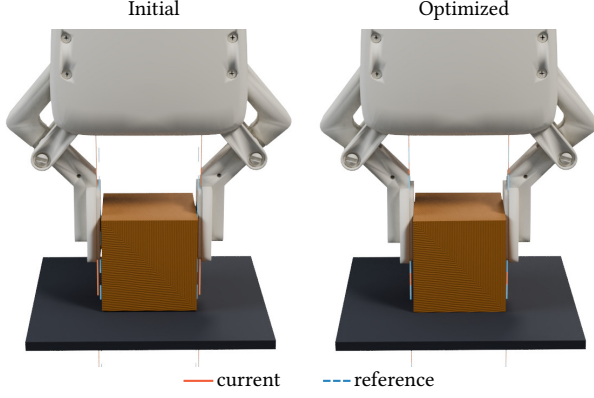


Fig. 13. Grasp-angle control. Initial (left) and optimized (right) grasps at frame 60. Joint-target optimization moves the grasp toward the 0.45-rad target. Coral guides follow the fingers. Dashed blue guides indicate visual reference directions.

one vertical offset added to both grippers' target positions to control their normal contact force on the cloth. These objectives span motion and force while sharing the coupled FEM-affine backward computation.

Grasp-angle control. An articulated affine robot approaches and grasps a FEM object above a fixed support. The closure objective measures two revolute joints in each finger linkage. The 120-frame horizon includes approach, closure, and return motion. Let $\mathcal{J}_f$ be this set of four gripper joints and $\alpha_{t,j}$ their simulated revolute angles. At the closure observation $t = 60$, the objective is $L_{\text{grasp}} = \frac{1}{2} \sum_{j \in \mathcal{J}_f} (\alpha_{60,j} - a^*)^2$, with fixed desired angle $a^* = 0.45$ rad. The optimized input is the common target angle. The four angles $\alpha_{60,j}$ are measured from the simulated response to that target and need not be identical under contact. The optimized joint-target angle is 0.447843 rad. The objective acts on the four simulated angles, so the gradient accounts for the robot's response to the command within the coupled scene. Figure 13 compares the grasps. Its dashed guides are visual references and do not enter the loss.

Cup catching. Joint targets move a plate held firmly by an xArm gripper. The task requires the plate motion to account for the incoming cup's response through contact. A uniformly decelerating free-plate trajectory, constructed from the cup's initial velocity, supplies an initial joint-target motion through sequential URDF inverse kinematics (IK). This initializes the commands geometrically. Differentiable simulation then optimizes their effect on the coupled cup-and-plate dynamics, as compared with the stationary and initial-trajectory cases in Figure 14.

The simulation uses 240 frames at $h = 0.005$ s with friction coefficient 1.0. The optimized command matrix is $\mathbf{U}_{\text{cmd}} = (\mathbf{u}_1, \dots, \mathbf{u}_{100})^T \in \mathbb{R}^{100 \times 7}$, where $\mathbf{u}_k \in \mathbb{R}^7$ contains the relative target angles of the seven arm joints at control sample k . The command at simulation frame t is

$$\mathbf{a}_t^{\text{cmd}} = \begin{cases} \mathbf{0}, & 1 \leq t \leq 98, \\ \mathbf{u}_{t-98}, & 99 \leq t \leq 198, \\ \mathbf{u}_{100}, & 199 \leq t \leq 240. \end{cases}$$

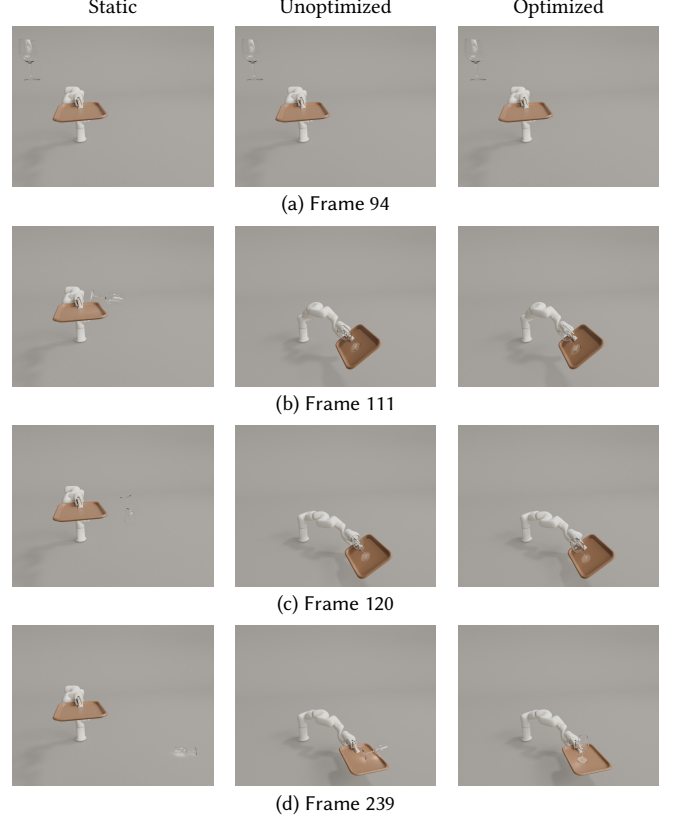


Fig. 14. Articulated cup catching through plate-trajectory optimization. With the arm held at rest (left), the cup falls from the plate. After optimization (right), the plate catches and retains it. The middle column shows the initial IK trajectory, which also fails to retain the cup.

The final 42 frames hold $\mathbf{u}_{100}$ fixed, and their gradients accumulate into this target.

Five state losses over frames 99–240, starting when the plate commands begin. For body $b \in \{\text{plate}, \text{cup}\}$, $\mathbf{p}_t^b$ and $\mathbf{A}_t^b$ denote its simulated center position and affine orientation matrix, and $\mathbf{v}_t^b = (\mathbf{p}_t^b - \mathbf{p}_{t-1}^b)/h$ is its center velocity. Starred plate quantities denote the prescribed reference. Position, velocity, and orientation tracking keep the plate near this path with the intended motion and orientation:

$$\begin{aligned} L_p &= \frac{1}{142s_p^2} \sum_{t=99}^{240} \|\mathbf{p}_t^{\text{plate}} - \mathbf{p}_t^{\text{plate},*}\|^2, \\ L_v &= \frac{1}{142s_v^2} \sum_{t=99}^{240} \|\mathbf{v}_t^{\text{plate}} - \mathbf{v}_t^{\text{plate},*}\|^2, \\ L_A &= \frac{1}{142s_A^2} \sum_{t=99}^{240} \|\mathbf{A}_t^{\text{plate}} - \mathbf{A}_t^{\text{plate},*}\|_F^2. \end{aligned}$$

The normalization scales are $s_p = 0.02$, $s_v = \sqrt{8}$, and $s_A = \sqrt{2}/20$.

Cup uprightness discourages tipping without constraining yaw, and a relative-velocity loss discourages sliding between the cup and

plate. With world up $\mathbf{e}_{\text{up}} = (-1, 0, 0)^\top$, the cup up vector is

$$\mathbf{z}_t^{\text{cup}} = \frac{\mathbf{A}_t^{\text{cup}} (\mathbf{A}_0^{\text{cup}})^{-1} \mathbf{e}_{\text{up}}}{\|\mathbf{A}_t^{\text{cup}} (\mathbf{A}_0^{\text{cup}})^{-1} \mathbf{e}_{\text{up}}\|}.$$

The two losses are

$$L_{\text{up}} = \frac{1}{142} \sum_{t=99}^{240} \|\mathbf{z}_t^{\text{cup}} - \mathbf{e}_{\text{up}}\|^2,$$

$$L_{\text{vel}} = \frac{1}{142s_v^2} \sum_{t=99}^{240} \|\mathbf{v}_t^{\text{cup}} - \mathbf{v}_t^{\text{plate}}\|^2.$$

Command smoothness is encouraged by $L_\Delta = \frac{1}{99} \sum_{k=2}^{100} \|\mathbf{u}_k - \mathbf{u}_{k-1}\|^2$, which penalizes changes between consecutive independent targets. The total objective is

$$L_{\text{cup}} = w_p L_p + w_v L_v + w_A L_A \\ + w_{\text{up}} L_{\text{up}} + w_{\text{vel}} L_{\text{vel}} + w_\Delta L_\Delta,$$

with fixed weights $w_p = 100$, $w_v = 20$, $w_A = 20$, $w_{\text{up}} = 20$, $w_{\text{vel}} = 50$, and $w_\Delta = 0.001$.

The cup tilt is $\alpha_t = \arccos((\mathbf{z}_t^{\text{cup}})^\top \mathbf{e}_{\text{up}})$. A catch is successful when, in each of the final ten frames, $\alpha_t \leq \pi/18$ rad, the minimum cup-plate vertex distance is at most 2 mm, and both center speeds $\|\mathbf{v}_t^{\text{cup}}\|$ and $\|\mathbf{v}_t^{\text{plate}}\|$ are below 0.05 m/s. Optimization stops once all of these conditions are met. The optimized plate catches and retains the cup (Figure 14). All 700 joint-target variables are optimized jointly through the arm dynamics and cup-plate contact. Their gradients share one coupled adjoint solve per frame (Sections 4.3 and 5.4). At a fixed simulation horizon, additional control variables therefore require local derivative accumulation rather than additional coupled solves.

Dual-arm towel grasp. The same vertical offset is added to both grippers' target positions during descent and the subsequent hold. Optimizing this one value changes how firmly the fingers press the cloth. The horizontal trajectories and finger-closing commands remain fixed. Let F_{120} be the total normal contact force between the cloth and the fingers of both arms at frame 120 during the hold.

The target normal force is estimated from the friction needed to accelerate the cloth with the closing fingers, using the cloth mass, prescribed closing acceleration, and cloth-finger friction coefficient. A fixed factor of three provides additional margin, giving $F^* = 1.4573$ N. The optimized loss is $L_{\text{force}} = \frac{1}{2} (F_{120} - F^*)^2$, with F_{120} measured from the simulated contacts. Starting from a finger-surface height of 1.9 mm and contact force 0.0226 N, the optimizer produces a grasp that lifts and folds the towel after seven updates (Figure 15). This task uses a force objective during the press-down hold to set the contact needed for the subsequent cloth motion, linking an affine position command to a deformable-object outcome.

These outcomes connect parameter choice to physical effect: rest geometry establishes support, initial placement forms a stable assembly, and actuation controls contacting objects. All use the same coupled backward equations, with task-specific parameter maps and losses. Figure 16 complements the physical sequences with the corresponding optimization histories.

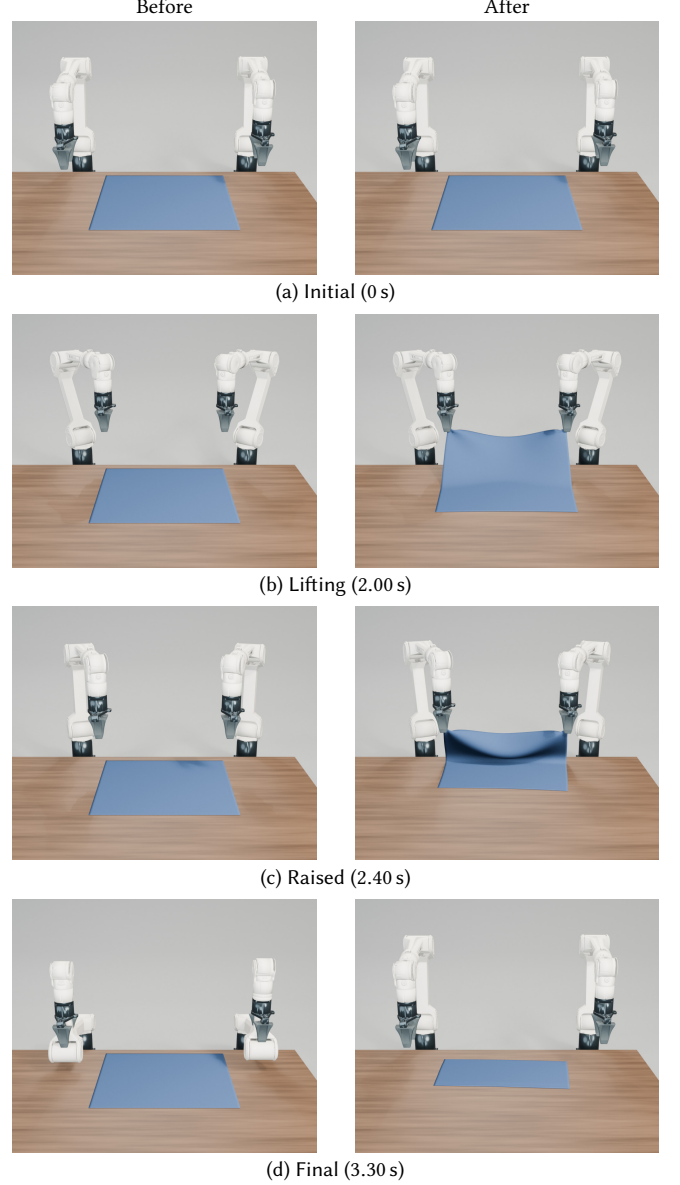


Fig. 15. Dual-arm towel grasp by optimizing one vertical offset applied to both grippers' target positions. Before optimization (left), the arms fail to grasp the towel. After optimization (right), they grasp the towel and then lift and fold it. Rows show the motion at 0, 2.00, 2.40, and 3.30 s.

7.4 Gradient validation and representation studies

Gradient validation. Table 2 reports finite-difference (FD) checks at the initial optimization parameters for all DiffGIPC examples and the shared DiffIPC benchmarks. For the checked parameter vector $\mathbf{z}$, let $\mathbf{g}_{\text{ana}}$ be its analytic loss gradient. The direction $\mathbf{d}$ is the normalized gradient, or its sign for a scalar parameter. With perturbation size ϵ_{FD} , the centered difference is $D_{\text{FD}}L = [L(\mathbf{z} + \epsilon_{\text{FD}}\mathbf{d}) - L(\mathbf{z} - \epsilon_{\text{FD}}\mathbf{d})]/(2\epsilon_{\text{FD}})$. The reported relative error is $|\mathbf{g}_{\text{ana}}^\top \mathbf{d} - D_{\text{FD}}L|/\max(|\mathbf{g}_{\text{ana}}^\top \mathbf{d}|, |D_{\text{FD}}L|)$. The shared-case checks use the first 10 frames. The other checks use the stated task-specific windows.

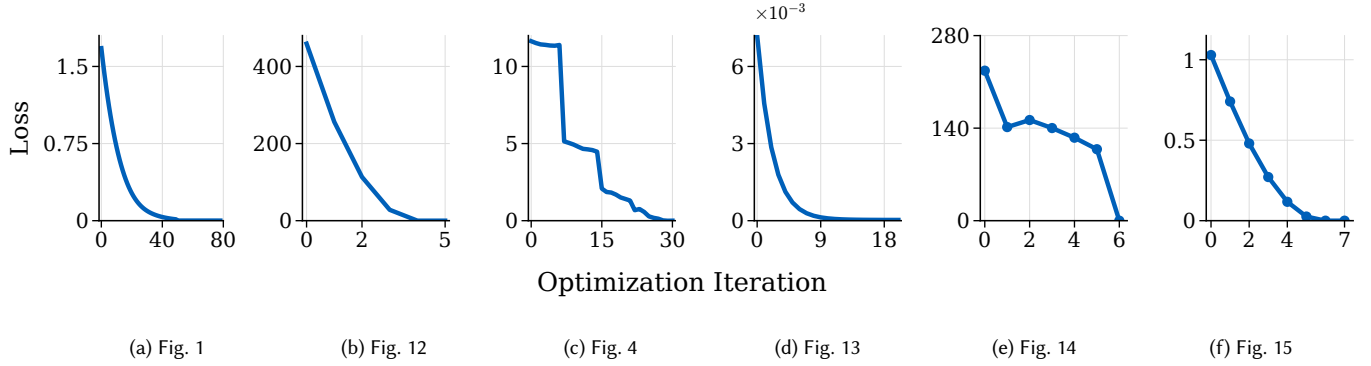


Fig. 16. Optimization histories for the additional tasks. Stone placement changes its centroid tolerance after 50 updates. Garment Hanger, cup catching, and Dual-arm grasp use constant offsets that set their final plotted values to zero.

Table 2. Finite-difference checks at the initial optimization parameters.

Example	System	Dim	Relative FD error
Stone placement (Fig. 1)	DiffGIPC	14	7.53×10^{-11}
Garment Hanger (Fig. 4)	DiffGIPC	2	5.45×10^{-3}
Bunny collision (Fig. 6)	DiffGIPC	3	1.21×10^{-6}
	DiffIPC	3	1.01×10^{-4}
Tentacle matching (Fig. 7)	DiffGIPC	3	3.33×10^{-5}
	DiffIPC	3	3.83×10^{-11}
Loaded Support (Fig. 8)	DiffGIPC	9939	2.56×10^{-6}
	DiffIPC	9939	4.02×10^{-4}
Material fitting (Fig. 9)	DiffGIPC	2	8.37×10^{-7}
	DiffIPC	2	6.71×10^{-7}
Friction optimization (Fig. 10)	DiffGIPC	1	4.38×10^{-5}
	DiffIPC	1	6.27×10^{-3}
Cloth rest scale (Fig. 12)	DiffGIPC	1	9.45×10^{-3}
Grasp angle (Fig. 13)	DiffGIPC	1	9.04×10^{-7}
Cup catching (Fig. 14)	DiffGIPC	700	4.48×10^{-4}
Dual-arm grasp (Fig. 15)	DiffGIPC	1	1.70×10^{-3}

Bunny resolution study. Table 3 compares three body representations as bunny resolution increases. FEM+FEM uses two deformable bodies, ABD+FEM uses one affine and one FEM body, and ABD+ABD uses two affine bodies. As resolution grows from 933 to 28,435 vertices, the combined forward and backward time for DiffGIPC FEM+FEM increases from 0.98 to 10 s, whereas the ABD+ABD time increases from 0.14 to 0.37 s. The increases are about 10.0 $\times$ and 2.6 $\times$, respectively.

At the highest resolution, ABD+ABD has a forward-plus-backward speedup of about 27 $\times$ over DiffGIPC FEM+FEM. The time spent in backward linear solves falls from 2.3 to 0.0044 s (about 520 $\times$), while the time for a complete backward evaluation falls from 2.5 to 0.048 s (about 52 $\times$). With twelve motion DoFs per body, ABD reduces the adjoint-system size without simplifying the surface mesh used for contact and shape derivatives. For FEM+FEM, the speedup of DiffGIPC over DiffIPC grows from 25.4 $\times$ to 150.0 $\times$ across resolutions. Combining GPU execution with an affine motion representation gives a 4133.3 $\times$ speedup for DiffGIPC ABD+ABD over DiffIPC FEM+FEM at the highest resolution.

Garment Hanger: ABD versus FEM. This comparison keeps the surface discretization fixed at 4,476 vertices and 8,952 triangles while changing the motion representation. The FEM replacement adds 902 interior vertices and 18,256 tetrahedra, with a harmonic extension of the two Bézier shape fields to the interior. The mean forward-plus-backward cost is 7.1 s for ABD and 16 s for FEM, giving a speedup of 2.2 $\times$. The backward costs are 0.27 and 1.0 s. The corresponding optimized control-point scales (s_x, s_y) are (0.8285, 0.8000) and (0.8301, 0.8000). At fixed surface discretization, the two models allocate computation differently. FEM resolves non-affine deformation through an interior tetrahedral mesh, whereas ABD retains a compact motion state and differentiates mass directly from the surface. The timings compare forward and backward evaluations of the ABD and FEM models for the same cloth-support design task.

7.5 GPU execution and optimization cost

This section measures the speedup from CUDA Graphs and the time spent on forward simulation and backward differentiation during optimization. Conditional CUDA Graphs reduce coordination during PCG, while parallel assembly and derivative accumulation process the local work surrounding each solve. Table 4 gives a detailed breakdown of the time spent on forward simulation and backward differentiation, together with the evaluation counts and accumulated evaluation time for each optimization. Table 5 compares execution with CUDA Graphs enabled and disabled, reporting PCG and total evaluation times for forward simulation and backward differentiation, together with the resulting speedups.

CUDA Graph execution provides 1.4–1.8 $\times$ speedups in forward PCG and 1.3–1.8 $\times$ in backward PCG over conventional kernel launches (Table 5). The PCG recurrence and convergence criterion are unchanged. These gains demonstrate the benefit of reducing kernel-launch overhead and CPU–GPU synchronization within the iterative solve (Section 6.1).

PCG is the largest contributor to backward time in most tasks (Table 4), making the adjoint solver a major source of savings in gradient evaluation. The acceleration remains visible in the complete computation: combined forward and backward evaluations achieve 1.2–1.5 $\times$ speedups across all reported tasks. These timings include collision detection, Hessian assembly, line search, and derivative

Table 3. Mean forward and backward evaluation times (seconds) for the bunny-collision resolution study and Garment Hanger rest-shape design. Forward + Backward gives the sum of the two per-evaluation means.

Resolution / case	System	Forward						Backward				Forward + Backward
		Linear	CCD	Hessian	Search	Other	Total	Hessian	Linear	Other	Total	
Bunny collision (Fig. 6): resolution study												
933 vertices	DiffGIPC ABD+ABD	0.025	0.039	0.041	0.021	0.0023	0.13	0.0081	0.0042	0.0032	0.016	0.14
	DiffGIPC ABD+FEM	0.58	0.049	0.058	0.03	0.0025	0.72	0.011	0.22	0.0085	0.24	0.96
	DiffGIPC FEM+FEM	0.65	0.044	0.037	0.017	0.002	0.75	0.0069	0.22	0.0057	0.24	0.98
	DiffIPC FEM+FEM	4.5	10	4.6	1.1	1.9	22	1.4	1.1	0.44	2.9	25
7,991 vertices	DiffGIPC ABD+ABD	0.032	0.054	0.11	0.029	0.0023	0.22	0.018	0.0043	0.0036	0.026	0.25
	DiffGIPC ABD+FEM	1.4	0.075	0.18	0.045	0.0033	1.7	0.028	0.32	0.025	0.37	2.1
	DiffGIPC FEM+FEM	2.1	0.069	0.18	0.032	0.0031	2.4	0.029	0.49	0.02	0.54	3
	DiffIPC FEM+FEM	80	24	54	9.7	7.6	170	15	17	0.98	33	210
28,435 vertices	DiffGIPC ABD+ABD	0.027	0.059	0.21	0.026	0.0026	0.32	0.039	0.0044	0.0043	0.048	0.37
	DiffGIPC ABD+FEM	3.5	0.091	0.53	0.059	0.0049	4.2	0.081	0.99	0.074	1.1	5.4
	DiffGIPC FEM+FEM	6.9	0.088	0.73	0.052	0.0043	7.8	0.11	2.3	0.057	2.5	10
	DiffIPC FEM+FEM	740	58	440	55	32	1300	100	110	4.6	220	1500
Garment Hanger (Fig. 4): rest-shape optimization												
Bezier pair	DiffGIPC ABD	3.7	0.88	1.5	0.8	0.0088	6.9	0.033	0.21	0.022	0.27	7.1
	DiffGIPC FEM	11	1.1	1.7	0.73	0.0076	15	0.029	0.95	0.02	1	16

Table 4. Complete optimization workloads and mean phase times (seconds) of the eleven examples. N_{fwd} counts forward evaluations, including line-search trials and verification runs. N_{bwd} counts complete gradient evaluations used for optimization. Linear comprises PCG and Others, and Hessian combines local and global assembly.

Example	N_{fwd}	N_{bwd}	Forward mean							Backward mean					T_{eval}
			Linear		CCD	Hessian	Search	Other	Total	Hessian	Linear		Other	Total	
			PCG	Others							PCG	Others			
Stone placement (Fig. 1)	80	78	0.097	0.0093	0.079	0.12	0.024	0.0088	0.34	0.051	0.067	0.003	0.023	0.14	39
Bunny collision (Fig. 6)	15	12	0.62	0.027	0.044	0.037	0.017	0.002	0.75	0.0069	0.22	0.005	0.0057	0.24	14
Tentacle matching (Fig. 7)	11	10	6.8	1.3	2.4	2.2	1.4	0.011	14	0.04	0.99	0.024	0.024	1.1	170
Garment Hanger (Fig. 4)	32	30	2.8	0.87	0.88	1.5	0.8	0.0088	6.9	0.033	0.2	0.01	0.022	0.27	230
Loaded Support (Fig. 8)	231	200	0.92	0.012	0.02	0.014	0.0017	0.00051	0.97	0.011	0.84	0.0065	0.0066	0.87	400
Material fitting (Fig. 9)	21	20	0.011	0.0035	0.0021	0.0042	0.00064	0.0032	0.025	0.0026	0.021	0.002	0.0016	0.028	1.1
Friction optimization (Fig. 10)	6	5	3.2	0.6	1.4	1.3	0.7	0.0093	7.2	0.037	0.35	0.014	0.039	0.44	45
Cloth rest scale (Fig. 12)	7	6	13	0.38	0.54	0.68	0.31	0.0046	15	0.0029	0.48	0.00087	0.001	0.48	110
Grasp angle (Fig. 13)	44	20	0.34	0.044	0.074	0.27	0.12	0.0035	0.85	0.048	0.058	0.0073	0.064	0.18	41
Cup catching (Fig. 14)	7	6	2.3	0.17	2.1	2.2	1.1	0.017	7.9	0.2	0.68	0.009	0.13	1	62
Dual-arm grasp (Fig. 15)	9	7	3.4	0.24	0.84	1.6	0.4	0.012	6.5	0.2	0.9	0.024	0.58	1.7	70

accumulation, demonstrating a reduction in the cost of simulation and differentiation beyond the PCG loop itself.

CUDA Graphs also accelerate the additional forward simulations used to evaluate candidate parameters and verify the optimized motion. Grasp-angle optimization, for example, uses 44 forward evaluations and 20 backward evaluations (Table 4). The same graph-based forward solver serves both the trajectories used for gradient computation and the additional objective checks and verification runs. Together with the representation study, these results establish complementary benefits: compact coordinates reduce the numerical workload, and graph-based execution reduces the recurring overhead of solving these systems throughout optimization.

8 Conclusion and Discussion

DiffGIPC combines mixed-coordinate trajectory adjoints with GPU execution for contact-rich inverse problems. Full-space FEM and twelve-DoF affine bodies remain coupled through contact, friction, and joints, while affine shape and control derivatives expose these interactions to optimization. Parallel accumulation, CUDA Graph-based PCG, and a PyTorch interface provide the GPU backward and task-level pipeline. Across the shared full-space FEM benchmarks, complete backward evaluation is 12.3–34.1× faster than DiffIPC, with 10.6–60.9× combined forward-and-backward speedups. Affine coordinates provide further savings when near-rigid motion is appropriate, although surface resolution still determines contact and assembly work.

Table 5. CUDA Graph enabled (On) versus disabled (Off). Times are mean seconds per evaluation; parentheses in the Off rows give speedups from enabling the graph. Linear comprises PCG and Others; the separate Other is Total minus Linear. The rightmost Total combines forward and backward time.

Example	Graph	Forward (s)				Backward (s)				Total (s)
		Linear		Other	Total	Linear		Other	Total	
		PCG	Others			PCG	Others			
Stone placement (Fig. 1)	On	0.097	0.0093	0.24	0.34	0.067	0.003	0.074	0.14	0.49
	Off	0.18 (1.8×)	0.0094	0.24	0.42 (1.2×)	0.12 (1.8×)	0.003	0.073	0.2 (1.4×)	0.62 (1.3×)
Bunny collision (Fig. 6)	On	0.62	0.027	0.1	0.75	0.22	0.005	0.013	0.24	0.98
	Off	0.97 (1.6×)	0.026	0.1	1.1 (1.5×)	0.35 (1.6×)	0.0051	0.011	0.36 (1.5×)	1.5 (1.5×)
Tentacle matching (Fig. 7)	On	6.8	1.3	6	14	0.99	0.024	0.063	1.1	15
	Off	10 (1.5×)	1.3	6.3	18 (1.3×)	1.5 (1.5×)	0.024	0.067	1.6 (1.4×)	20 (1.3×)
Garment Hanger (Fig. 4)	On	2.8	0.87	3.2	6.9	0.2	0.01	0.055	0.27	7.1
	Off	4.1 (1.5×)	0.8	3.2	8.1 (1.2×)	0.27 (1.4×)	0.01	0.049	0.33 (1.2×)	8.4 (1.2×)
Loaded Support (Fig. 8)	On	0.92	0.012	0.036	0.97	0.84	0.0065	0.017	0.87	1.8
	Off	1.2 (1.4×)	0.012	0.032	1.3 (1.3×)	1.2 (1.4×)	0.0064	0.019	1.2 (1.4×)	2.5 (1.4×)
Material fitting (Fig. 9)	On	0.011	0.0035	0.01	0.025	0.021	0.002	0.0042	0.028	0.053
	Off	0.017 (1.5×)	0.0034	0.011	0.032 (1.3×)	0.033 (1.5×)	0.0018	0.0049	0.039 (1.4×)	0.071 (1.4×)
Friction optimization (Fig. 10)	On	3.2	0.6	3.4	7.2	0.35	0.014	0.076	0.44	7.7
	Off	5 (1.6×)	0.63	3.7	9.3 (1.3×)	0.55 (1.6×)	0.015	0.082	0.64 (1.5×)	9.9 (1.3×)
Cloth rest scale (Fig. 12)	On	13	0.38	1.5	15	0.48	0.00087	0.004	0.48	16
	Off	18 (1.4×)	0.4	1.6	20 (1.3×)	0.64 (1.3×)	0.00093	0.0038	0.64 (1.3×)	21 (1.3×)
Grasp angle (Fig. 13)	On	0.34	0.044	0.47	0.85	0.058	0.0073	0.11	0.18	1
	Off	0.51 (1.5×)	0.045	0.48	1 (1.2×)	0.084 (1.4×)	0.0074	0.11	0.21 (1.2×)	1.2 (1.2×)
Cup catching (Fig. 14)	On	2.3	0.17	5.4	7.9	0.68	0.009	0.33	1	8.9
	Off	3.7 (1.6×)	0.13	5.5	9.3 (1.2×)	1 (1.5×)	0.0099	0.33	1.3 (1.3×)	11 (1.2×)
Dual-arm grasp (Fig. 15)	On	3.4	0.24	2.9	6.5	0.9	0.024	0.78	1.7	8.2
	Off	5 (1.5×)	0.21	2.8	7.9 (1.2×)	1.3 (1.5×)	0.024	0.79	2.1 (1.2×)	10 (1.2×)

Differentiation uses the contact pairs, closest-feature choices, and control operations recorded during forward simulation. A pair of objects that remains outside the barrier’s activation range contributes no direct contact gradient, which can limit the guidance available when the initial trajectory does not establish the desired interaction. Rest-shape derivatives assume fixed connectivity within each differentiated trajectory. Remeshing can occur between optimization steps, but its decisions and vertex transfers are not differentiated. Improving the underlying contact model and the gradients used for optimization offers a direction for future work. Geometric Contact Potential [Huang et al. 2025b] could reduce spurious contact forces and sensitivity to surface discretization. Long-range contact potentials [Ye et al. 2026] could provide gradient information before objects enter the local barrier range, helping optimization establish new contacts. Randomized smoothing with bundled gradients [Suh et al. 2022] could mitigate rapid changes in contact gradients. Adapting these approaches to coupled FEM-affine dynamics and evaluating their effects on simulation accuracy, optimization convergence, and computational cost remain future work.

References

- David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '98)*. Association for Computing Machinery, New York, NY, USA, 43–54. doi:10.1145/280814.280821
- Jernej Barbic and Doug L. James. 2005. Real-Time subspace integration for St. Venant-Kirchhoff deformable models. In *ACM SIGGRAPH 2005 Papers* (Los Angeles, California) (SIGGRAPH '05). Association for Computing Machinery, New York, NY, USA, 982–990. doi:10.1145/1186822.1073300
- Filipe de A. Belbute-Peres, Kevin A. Smith, Kelsey R. Allen, Joshua B. Tenenbaum, and J. Zico Kolter. 2018. End-to-end differentiable physics for learning and control. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) (NIPS'18). Curran Associates Inc., Red Hook, NY, USA, 7178–7189.
- Miklos Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun. 2006. A quadratic bending model for inextensible surfaces. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Cagliari, Sardinia, Italy) (SGP '06). Eurographics Association, Goslar, DEU, 227–230.
- Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics: fusing constraint projections for fast simulation. *ACM Trans. Graph.* 33, 4, Article 154 (July 2014), 11 pages. doi:10.1145/2601097.2601116
- Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-reduced projective dynamics. *ACM Trans. Graph.* 37, 4, Article 80 (July 2018), 13 pages. doi:10.1145/3197517.3201387
- Youshuai Cai, Chen Li, Haichuan Song, Youchen Xie, and ChangBo Wang. 2025. DiffQN: Differentiable Quasi-Newton Method for Elastodynamics. In *Pacific Graphics Conference Papers, Posters, and Demos*, Marc Christie, Ping-Hsuan Han, Shih-Syun Lin, Nico Pietroni, Teseo Schneider, Hsin-Ruey Tsai, Yu-Shuen Wang, and Eugene Zhang (Eds.). The Eurographics Association. doi:10.2312/pg.20251265
- Jiong Chen, Florian Schäfer, Jin Huang, and Mathieu Desbrun. 2021. Multiscale Cholesky preconditioning for ill-conditioned problems. *ACM Trans. Graph.* 40, 4, Article 81 (July 2021), 13 pages. doi:10.1145/3450626.3459851
- Jonas Degraeve, Michiel Hermans, Joni Dambre, and Francis wyffels. 2019. A Differentiable Physics Engine for Deep Learning in Robotics. *Frontiers in Neurorobotics* Volume 13 - 2019 (2019). doi:10.3389/fnbot.2019.00006
- Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. 2021. DiffPD: Differentiable Projective Dynamics. *ACM Trans. Graph.* 41, 2, Article 13 (nov 2021), 21 pages. doi:10.1145/3490168
- Zachary Ferguson, Minchen Li, Teseo Schneider, Francisca Gil-Ureta, Timothy Langlois, Chenfanfu Jiang, Denis Zorin, Danny M. Kaufman, and Daniele Panozzo. 2021. Intersection-free rigid body dynamics. *ACM Trans. Graph.* 40, 4, Article 183 (July 2021), 16 pages. doi:10.1145/3450626.3459802
- C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. 2021. Brax – A Differentiable Physics Engine for Large Scale Rigid Body Simulation. arXiv:2106.13281 [cs.RO] <https://arxiv.org/abs/2106.13281>

- Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. 2020. ADD: analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Trans. Graph.* 39, 6, Article 190 (Nov. 2020), 15 pages. doi:10.1145/3414685.3417766
- Arvi Gjoka, Espen Knoop, Moritz Bächer, Denis Zorin, and Daniele Panozzo. 2024. Soft Pneumatic Actuator Design using Differentiable Simulation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Papers '24* (Denver, CO, USA) (*SIGGRAPH Conference Papers '24*). Association for Computing Machinery, New York, NY, USA, Article 106, 11 pages. doi:10.1145/3641519.3657467
- David Hahn, Pol Banzet, James M. Bern, and Stelian Coros. 2019. Real2Sim: visco-elastic parameter estimation from dynamic motion. *ACM Trans. Graph.* 38, 6, Article 236 (Nov. 2019), 13 pages. doi:10.1145/3355089.3356548
- Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. 2020. DiffTaichi: Differentiable Programming for Physical Simulation. *ICLR* (2020).
- Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. 2019. ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)* (2019).
- Kemeng Huang, Floyd M. Chitalu, Huancheng Lin, and Taku Komura. 2024a. GIPC: Fast and Stable Gauss-Newton Optimization of IPC Barrier Energy. *ACM Trans. Graph.* 43, 2, Article 23 (March 2024), 18 pages. doi:10.1145/3643028
- Kemeng Huang, Xinyu Lu, Huancheng Lin, Taku Komura, and Minchen Li. 2025a. StiffGIPC: Advancing GPU IPC for Stiff Affine-Deformable Simulation. *ACM Trans. Graph.* 44, 3, Article 31 (May 2025), 20 pages. doi:10.1145/3735126
- Zizhou Huang, Maxwell Paik, Zachary Ferguson, Daniele Panozzo, and Denis Zorin. 2025b. Geometric Contact Potential. *ACM Trans. Graph.* 44, 4, Article 158 (July 2025), 24 pages. doi:10.1145/3731142
- Zizhou Huang, Daniele Panozzo, and Denis Zorin. 2024b. Optimized shock-protecting microstructures. *ACM Trans. Graph.* 43, 6, Article 181 (Nov. 2024), 21 pages. doi:10.1145/3687765
- Zizhou Huang, Davi Colli Tozoni, Arvi Gjoka, Zachary Ferguson, Teseo Schneider, Daniele Panozzo, and Denis Zorin. 2024c. Differentiable solver for time-dependent deformation problems with contact. *ACM Trans. Graph.* 43, 3, Article 31 (may 2024), 30 pages. doi:10.1145/3657648
- Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. 2021. gradSim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=c_E8kFWfhp0
- Theodore Kim. 2020. A Finite Element Formulation of Baraff–Witkin Cloth. *Computer Graphics Forum* 39, 8 (2020), 171–179. doi:10.1111/cgf.14111
- Shih-Yu Lai, Sung-Han Tien, Jui-I Huang, Yen-Chen Tseng, Yi-Ting Chiu, Siyuan Luo, Ziqiu Zeng, Fan Shi, Peter Yichen Chen, Tiantian Liu, Yu-Lun Liu, and Bing-Yu Chen. 2026. DiffPhD: A Unified Differentiable Solver for Projective Heterogeneous Materials in Elastodynamics with Contact-Rich GPU-Acceleration. arXiv:2605.14526 [cs.GR] <https://arxiv.org/abs/2605.14526>
- Lei Lan, Danny M. Kaufman, Minchen Li, Chenfanfu Jiang, and Yin Yang. 2022. Affine body dynamics: fast, stable and intersection-free simulation of stiff materials. *ACM Trans. Graph.* 41, 4, Article 67 (July 2022), 14 pages. doi:10.1145/3528223.3530064
- Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. 2020. Incremental Potential Contact: Intersection- and Inversion-free Large Deformation Dynamics. *ACM Trans. Graph. (SIGGRAPH)* 39, 4, Article 49 (2020).
- Minchen Li, Chenfanfu Jiang, Zhao Feng Luo, Wenxin Du, Chang Yu, Žiga Kovačič, and Tianyi Xie. 2026. *Physics-Based Simulation*. doi:10.5281/zenodo.20597655
- Xuan Li, Chang Yu, Wenxin Du, Ying Jiang, Tianyi Xie, Yunuo Chen, Yin Yang, and Chenfanfu Jiang. 2025. Dress-1-to-3: Single Image to Simulation-Ready 3D Outfit with Diffusion Prior and Differentiable Physics. *ACM Trans. Graph.* 44, 4, Article 71 (July 2025), 16 pages. doi:10.1145/3731177
- Yifei Li, Tao Du, Kui Wu, Jie Xu, and Wojciech Matusik. 2022. DiffCloth: Differentiable Cloth Simulation with Dry Frictional Contact. *ACM Trans. Graph.* 42, 1, Article 2 (oct 2022), 20 pages. doi:10.1145/3527660
- Junbang Liang, Ming C. Lin, and Vladlen Koltun. 2019. Differentiable Cloth Simulation for Inverse Problems. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-Newton Methods for Real-Time Simulation of Hyperelastic Materials. *ACM Trans. Graph.* 36, 3, Article 23 (May 2017), 16 pages. doi:10.1145/2990496
- Mickaël Ly, Jean Jouve, Laurence Boissieux, and Florence Bertails-Descoubes. 2020. Projective dynamics with dry frictional contact. *ACM Trans. Graph.* 39, 4, Article 57 (Aug. 2020), 8 pages. doi:10.1145/3386569.3392396
- Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C. Lin. 2020. Scalable Differentiable Physics for Learning and Control. In *ICML*.
- Junior Rojas, Eftychios Sifakis, and Ladislav Kavan. 2021. Differentiable Implicit Soft-Body Physics. arXiv:2102.05791 [cs.LG] <https://arxiv.org/abs/2102.05791>
- Breannan Smith, Fernando de Goes, and Theodore Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Transactions on Graphics* 37, 2, Article 12 (2018), 15 pages. <https://graphics.pixar.com/library/StableElasticity/paper.pdf>
- Hyung Ju Terry Suh, Tao Pang, and Russ Tedrake. 2022. Bundled Gradients Through Contact Via Randomized Smoothing. *IEEE Robotics and Automation Letters* 7, 2 (2022), 4000–4007. doi:10.1109/LRA.2022.3146931
- Davi Colli Tozoni, Yunfan Zhou, and Denis Zorin. 2021. Optimizing contact-based assemblies. *ACM Transactions on Graphics* 40, 6 (2021), 269:1–269:19. doi:10.1145/3478513.3480552
- Ty Trusty, Otman Benchechroun, Eitan Grinspun, Danny M. Kaufman, and David I.W. Levin. 2023. Subspace Mixed Finite Elements for Real-Time Heterogeneous Elastodynamics. In *SIGGRAPH Asia 2023 Conference Papers* (Sydney, NSW, Australia) (SA '23). Association for Computing Machinery, New York, NY, USA, Article 112, 10 pages. doi:10.1145/3610548.3618220
- Ty Trusty, Yun (Raymond) Fei, David Levin, and Danny Kaufman. 2024. Trading Spaces: Adaptive Subspace Time Integration for Contacting Elastodynamics. *ACM Trans. Graph.* 43, 6, Article 227 (Nov. 2024), 16 pages. doi:10.1145/3687946
- Xuan Wang, Zhao Feng Luo, Minchen Li, Taku Komura, and Kemeng Huang. 2026. AGIPC: Adaptive In-Solve Algebraic Coarsening for GPU IPC. In *SIGGRAPH Conference Papers '26*. Association for Computing Machinery, New York, NY, USA, Article 154, 11 pages. doi:10.1145/3799902.3811199
- Keenon Werling, Dalton Omens, Jeongseok Lee, Ioannis Exarchos, and C. Karen Liu. 2021. Fast and Feature-Complete Differentiable Physics Engine for Articulated Rigid Bodies with Contact Constraints. In *Proceedings of Robotics: Science and Systems*. Virtual. doi:10.15607/RSS.2021.XVII.034
- Botao Wu, Zhendong Wang, and Huamin Wang. 2022. A GPU-based multilevel additive schwarz preconditioner for cloth and deformable body simulation. *ACM Trans. Graph.* 41, 4, Article 63 (July 2022), 14 pages. doi:10.1145/3528223.3530085
- Xiaohan Ye, Xifeng Gao, Kui Wu, Zherong Pan, and Taku Komura. 2025. SDRS: Shape-Differentiable Robot Simulator. *IEEE Transactions on Robotics* (2025).
- Xiaohan Ye, Kui Wu, Zherong Pan, and Taku Komura. 2026. Efficient Differentiable Contact Model with Long-Range Influence. In *International Conference on Learning Representations*. <https://arxiv.org/abs/2509.20917>
- Ziqiu Zeng, Siyuan Luo, Fan Shi, and Zhongkai Zhang. 2025. Fast But Accurate: A Real-Time Hyperelastic Simulator with Robust Frictional Contact. *ACM Trans. Graph.* 44, 4, Article 88 (July 2025), 19 pages. doi:10.1145/3730834

A Derivatives for the Coupled FEM–Affine Adjoint

The frame-wise adjoint requires local derivatives for assembling the coupled Hessian and accumulating parameter and earlier-state gradients (Sections 4–6.2). This appendix derives these quantities for friction, elasticity, inertia, and joints. The derivations account for current and previous contact geometry, the dependence of mass and elastic response on rest shape, and the effects of joint geometry and control parameters.

A.1 Friction derivatives

A.1.1 Smoothed friction energy. The friction energy uses current-step sliding and contact quantities computed from previous-frame geometry [Li et al. 2020]. For contact c , the previous positions $\mathbf{x}_{t-1}^c$ determine the closest-point interpolation weights, contact plane, and normal force. Let $\mathbf{x}_j^- = \mathbf{x}_{t-1,i,j}$ and $\Delta \mathbf{x}_j = \mathbf{x}_{t,i,j} - \mathbf{x}_j^-$ denote the previous position and displacement of contact vertex j .

The matrix $\mathbf{W} \in \mathbb{R}^{3 \times 3n_c}$ converts stacked vertex displacements into the relative displacement of the two closest points:

$$\mathbf{W} = [\mathbf{w}_1^{\text{fr}} \mathbf{I}_3 \cdots \mathbf{w}_{n_c}^{\text{fr}} \mathbf{I}_3], \quad \mathbf{W}(\mathbf{x}_t^c - \mathbf{x}_{t-1}^c) = \sum_{j=1}^{n_c} \mathbf{w}_j^{\text{fr}} \Delta \mathbf{x}_j.$$

Adding the same displacement to all contact vertices leaves their relative displacement unchanged because the sum of the weights

is zero. With $\mathbf{w}^{\text{fr}} = (w_1^{\text{fr}}, \dots, w_{n_c}^{\text{fr}})$, the weights for the four mesh-contact types of IPC are

$$\mathbf{w}^{\text{fr}} = \begin{cases} (1, -1), & \text{point-point,} \\ (1, \eta - 1, -\eta), & \text{point-edge,} \\ (1, -\beta_1, -\beta_2, -\beta_3), & \text{point-triangle,} \\ (1 - s_A, s_A, s_B - 1, -s_B), & \text{edge-edge.} \end{cases}$$

The point is numbered first for point-edge and point-triangle contact. For point-edge contact, the other side has endpoints 2, 3, and its closest point is $\mathbf{x}_2^- + \eta \mathbf{e}$, where $\mathbf{e} = \mathbf{x}_3^- - \mathbf{x}_2^-$ and $\eta = \mathbf{e}^T(\mathbf{x}_1^- - \mathbf{x}_2^-)/(\mathbf{e}^T \mathbf{e})$. For point-triangle contact, $\beta_1, \beta_2, \beta_3$ are the barycentric coordinates of the closest point on triangle (2, 3, 4):

$$\mathbf{E}_\Delta = [\mathbf{x}_3^- - \mathbf{x}_2^- \quad \mathbf{x}_4^- - \mathbf{x}_2^-],$$

$$\begin{bmatrix} \beta_2 \\ \beta_3 \end{bmatrix} = (\mathbf{E}_\Delta^T \mathbf{E}_\Delta)^{-1} \mathbf{E}_\Delta^T (\mathbf{x}_1^- - \mathbf{x}_2^-), \quad \beta_1 = 1 - \beta_2 - \beta_3.$$

For edge-edge contact, the two edges have endpoints (1, 2) and (3, 4), with closest points $\mathbf{x}_1^- + s_A \mathbf{e}_A$ and $\mathbf{x}_3^- + s_B \mathbf{e}_B$, where $\mathbf{e}_A = \mathbf{x}_2^- - \mathbf{x}_1^-$ and $\mathbf{e}_B = \mathbf{x}_4^- - \mathbf{x}_3^-$. The coefficients satisfy

$$\begin{bmatrix} \mathbf{e}_A^T \mathbf{e}_A & -\mathbf{e}_A^T \mathbf{e}_B \\ -\mathbf{e}_A^T \mathbf{e}_B & \mathbf{e}_B^T \mathbf{e}_B \end{bmatrix} \begin{bmatrix} s_A \\ s_B \end{bmatrix} = \begin{bmatrix} \mathbf{e}_A^T (\mathbf{x}_3^- - \mathbf{x}_1^-) \\ -\mathbf{e}_B^T (\mathbf{x}_3^- - \mathbf{x}_1^-) \end{bmatrix}.$$

The edge and triangle formulas apply when the closest points lie in the interiors of the respective features. When a closest point lies at an edge endpoint or on a triangle boundary, the contact uses the corresponding point-point or point-edge formula.

The columns of $\mathbf{U} \in \mathbb{R}^{3 \times 2}$ are two orthogonal unit directions in the contact plane. For mesh-mesh contact, the unit normal is $\mathbf{n} = \mathbf{W} \mathbf{x}_{t-1}^c / \|\mathbf{W} \mathbf{x}_{t-1}^c\|$. An explicit construction using a Cartesian unit vector $\mathbf{e}_{\text{ref}}$ not parallel to $\mathbf{n}$ is

$$\mathbf{t}_1 = \frac{(\mathbf{I}_3 - \mathbf{n} \mathbf{n}^T) \mathbf{e}_{\text{ref}}}{\|(\mathbf{I}_3 - \mathbf{n} \mathbf{n}^T) \mathbf{e}_{\text{ref}}\|}, \quad \mathbf{t}_2 = \mathbf{t}_1 \times \mathbf{n},$$

$$\mathbf{U} = [\mathbf{t}_1 \quad \mathbf{t}_2], \quad \mathbf{U}^T \mathbf{U} = \mathbf{I}_2.$$

Any orthonormal basis of this plane gives the same isotropic friction energy and world-space gradient, since $\mathbf{U} \mathbf{U}^T = \mathbf{I}_3 - \mathbf{n} \mathbf{n}^T$. For the fixed horizontal ground, the normal is fixed and only the moving vertex is included in $\mathbf{x}_t^c$. In this case, $\mathbf{W} = \mathbf{I}_3$ and $\mathbf{U} = [\mathbf{e}_x \quad \mathbf{e}_z]^T$, with $\mathbf{e}_x = (1, 0, 0)^T$ and $\mathbf{e}_z = (0, 0, 1)^T$.

For mesh-mesh contact, $\mathbf{W}$ and $\mathbf{U}$ depend on $\mathbf{x}_{t-1}^c$. Both remain fixed when differentiating with respect to current positions. For the stacked displacement $\Delta \mathbf{x}_t^c = \mathbf{x}_t^c - \mathbf{x}_{t-1}^c$, the sliding displacement is

$$\boldsymbol{\xi} = \mathbf{U}^T \mathbf{W} \Delta \mathbf{x}_t^c, \quad \sigma = \|\boldsymbol{\xi}\|^2. \quad (66)$$

Here $\boldsymbol{\xi} \in \mathbb{R}^2$ contains the two in-plane displacement components, and σ is its squared length.

The friction energy is $E_{t,c}^{\text{fr}} = \mu_{\text{fr}} f_n \varphi(\sigma)$, where μ_{fr} is the friction coefficient. The normal-force magnitude f_n is determined by the barrier in Equation (14). For $0 < d < \hat{d}$, let $\Delta_d = d - \hat{d}$ and $\zeta_d = \log(d/\hat{d})$. The first and second barrier derivatives are

$$b'_{\text{bar}}(d) = 2\kappa \Delta_d \zeta_d (\zeta_d + \Delta_d/d),$$

$$b''_{\text{bar}}(d) = 2\kappa [\zeta_d^2 + 4\Delta_d \zeta_d/d + \Delta_d^2(1 - \zeta_d)/d^2]. \quad (67)$$

At the previous squared contact distance d^- , the normal-force magnitude is $f_n = -2\sqrt{d^-} b'_{\text{bar}}(d^-)$. It is held fixed during the current-frame solve. Its dependence on the previous positions is included in the history derivative. The unsmoothed friction energy is proportional to $\|\boldsymbol{\xi}\|$, which is not differentiable at zero sliding displacement. IPC uses smoothed friction to avoid discontinuities. With $\epsilon_{\text{fr}} = h \hat{d}_f$, where $\hat{d}_f$ is the friction-smoothing velocity threshold, the smooth function for mesh-mesh contact is

$$\varphi_{\text{self}}(\sigma) = \begin{cases} \frac{\sigma(-\sqrt{\sigma}/3 + \epsilon_{\text{fr}})}{\epsilon_{\text{fr}}^2} + \frac{\epsilon_{\text{fr}}}{3}, & \sigma \leq \epsilon_{\text{fr}}^2, \\ \sqrt{\sigma}, & \sigma > \epsilon_{\text{fr}}^2. \end{cases} \quad (68)$$

For ground friction, the smoothed function is

$$\varphi_{\text{ground}}(\sigma) = \begin{cases} \sigma/(2\epsilon_{\text{fr}}), & \sigma \leq \epsilon_{\text{fr}}^2, \\ \sqrt{\sigma} - \epsilon_{\text{fr}}/2, & \sigma > \epsilon_{\text{fr}}^2. \end{cases} \quad (69)$$

The required energy gradient is with respect to current vertex positions $\mathbf{x}_t^c$. Since $\mathbf{W}$, $\mathbf{U}$, and f_n are fixed for this derivative, $\partial \boldsymbol{\xi} / \partial \mathbf{x}_t^c = \mathbf{U}^T \mathbf{W}$. The chain rule gives

$$\begin{aligned} \mathbf{g}_{t,c}^{\text{fr},x} &= \nabla_{\mathbf{x}_t^c} E_{t,c}^{\text{fr}} \\ &= \mu_{\text{fr}} f_n \left(\frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}_t^c} \right)^T \nabla_{\boldsymbol{\xi}} \varphi(\|\boldsymbol{\xi}\|^2) \\ &= \mu_{\text{fr}} f_n \mathbf{W}^T \mathbf{U} [2\varphi'(\sigma) \boldsymbol{\xi}]. \end{aligned} \quad (70)$$

The vector $2\varphi'(\sigma) \boldsymbol{\xi}$ is the gradient with respect to the two-dimensional sliding displacement. For mesh-mesh friction below the smoothing threshold,

$$\nabla_{\boldsymbol{\xi}} \varphi_{\text{self}}(\|\boldsymbol{\xi}\|^2) = \frac{2\epsilon_{\text{fr}} - \|\boldsymbol{\xi}\|}{\epsilon_{\text{fr}}^2} \boldsymbol{\xi}. \quad (71)$$

This slip gradient is mapped to vertex positions by the factor $\mu_{\text{fr}} f_n \mathbf{W}^T \mathbf{U}$ in Equation (70). Above the threshold, the slip gradient is $\boldsymbol{\xi}/\|\boldsymbol{\xi}\|$ for both laws. For ground friction below the threshold, it is $\boldsymbol{\xi}/\epsilon_{\text{fr}}$.

Within each smooth branch, differentiating again gives the Hessian:

$$\mathbf{H}_{\boldsymbol{\xi}} = 2\varphi'(\sigma) \mathbf{I}_2 + 4\varphi''(\sigma) \boldsymbol{\xi} \boldsymbol{\xi}^T,$$

$$\mathbf{H}_{t,c}^{\text{fr},x} = \mu_{\text{fr}} f_n \mathbf{W}^T \mathbf{U} \mathbf{H}_{\boldsymbol{\xi}} \mathbf{U}^T \mathbf{W}.$$

At zero slip, the gradient is zero and the limiting $\mathbf{H}_{\boldsymbol{\xi}}$ is $2\mathbf{I}_2/\epsilon_{\text{fr}}$ for mesh-mesh friction or $\mathbf{I}_2/\epsilon_{\text{fr}}$ for ground friction. The dependence of $\mathbf{W}$, $\mathbf{U}$, and f_n on previous-frame positions supplies the temporal coupling in Equation (38).

A.1.2 Friction-coefficient and history derivatives. Point-space friction gradient supports coefficient fitting and differentiation through time. Its coefficient derivative measures how frictional resistance changes with μ_{fr} . Its history derivative also accounts for changes in closest-point weights, contact-plane orientation, normal force, and relative displacement. Let $\mathbf{g}_{\boldsymbol{\xi}} = 2\varphi'(\sigma) \boldsymbol{\xi}$ denote the two-dimensional slip derivative in Equation (70). At fixed frame configurations, $\mathbf{W}$, $\mathbf{U}$, and f_n are independent of μ_{fr} . The coefficient derivative and its

loss-gradient contribution $\Delta \bar{\mu}_{\text{fr}}^{(t,c)}$ from contact c at frame t are

$$\begin{aligned} \frac{\partial \mathbf{g}_{t,c}^{\text{fr},x}}{\partial \mu_{\text{fr}}} &= \mathbf{f}_n \mathbf{W}^\top \mathbf{U} \mathbf{g}_\xi, \\ \Delta \bar{\mu}_{\text{fr}}^{(t,c)} &= - \left(\frac{\partial \mathbf{g}_{t,c}^{\text{fr},x}}{\partial \mu_{\text{fr}}} \right)^\top \mathbf{K}_{t,c} \boldsymbol{\lambda}_t. \end{aligned} \quad (72)$$

The contributions $\Delta \bar{\mu}_{\text{fr}}^{(t,c)}$ are summed over all contacts and frames that share μ_{fr} . Together with the direct loss derivative, this sum gives the total coefficient gradient $\bar{\mu}_{\text{fr}}$ in Equation (30).

A change in $\mathbf{x}_{t-1}^c$ changes both the measured displacement and the geometry defining friction. The corresponding matrix is

$$\mathbf{C}_{t,c}^{\text{fr},x} = \frac{\partial \mathbf{g}_{t,c}^{\text{fr},x}}{\partial \mathbf{x}_{t-1}^c} \in \mathbb{R}^{3n_c \times 3n_c}. \quad (73)$$

With the contact vertices indices and contact type fixed, vary $\mathbf{x}_{t-1}^c$ while holding $\mathbf{x}_t^c$, μ_{fr} , and ϵ_{fr} fixed. Since $d(\Delta \mathbf{x}_t^c) = -d\mathbf{x}_{t-1}^c$, Equation (66) gives

$$\begin{aligned} d\xi &= (d\mathbf{U})^\top \mathbf{W} \Delta \mathbf{x}_t^c + \mathbf{U}^\top (d\mathbf{W}) \Delta \mathbf{x}_t^c \\ &\quad - \mathbf{U}^\top \mathbf{W} d\mathbf{x}_{t-1}^c. \end{aligned}$$

The first two terms account for changes in the contact-plane basis and closest-point weights. The last accounts for the previous positions in the displacement difference. Applying the product rule to Equation (70) yields

$$\begin{aligned} \mathbf{C}_{t,c}^{\text{fr},x} d\mathbf{x}_{t-1}^c &= \mu_{\text{fr}} (df_n) \mathbf{W}^\top \mathbf{U} \mathbf{g}_\xi \\ &\quad + \mu_{\text{fr}} f_n (d\mathbf{W})^\top \mathbf{U} \mathbf{g}_\xi \\ &\quad + \mu_{\text{fr}} f_n \mathbf{W}^\top (d\mathbf{U}) \mathbf{g}_\xi \\ &\quad + \mu_{\text{fr}} f_n \mathbf{W}^\top \mathbf{U} \mathbf{H}_\xi d\xi. \end{aligned}$$

The weight differential is $d\mathbf{W} = [(d\mathbf{w}_1^{\text{fr}}) \mathbf{I}_3 \cdots (d\mathbf{w}_n^{\text{fr}}) \mathbf{I}_3]$. For point-contact, the weights are constant, so $d\mathbf{W} = 0$. For the remaining contact types, the interpolation weights depend on the previous-frame vertex positions through the closest-point parameters. These parameters locate the closest points on the edges or triangle and form the vector

$$\boldsymbol{\omega}_{\text{cp}} = \begin{cases} [\eta], & \text{point-edge,} \\ [\beta_2 \ \beta_3]^\top, & \text{point-triangle,} \\ [s_A \ s_B]^\top, & \text{edge-edge.} \end{cases}$$

The vector $\boldsymbol{\omega}_{\text{cp}}$ is determined by the linear system

$$\mathbf{D}_{\text{cp}} \boldsymbol{\omega}_{\text{cp}} = \mathbf{r}_{\text{cp}}.$$

Here $\mathbf{D}_{\text{cp}}$ is the coefficient matrix and $\mathbf{r}_{\text{cp}}$ is the right-hand side, both determined by the previous-frame vertex positions.

For point-edge contact, the closest point on edge (2, 3) is $\mathbf{x}_2^- + \eta \mathbf{e}$, with $\mathbf{e} = \mathbf{x}_3^- - \mathbf{x}_2^-$. At an interior closest point, the vector $\mathbf{x}_1^- - (\mathbf{x}_2^- + \eta \mathbf{e})$ is perpendicular to the edge direction $\mathbf{e}$, giving

$$\mathbf{D}_{\text{cp}} = \mathbf{e}^\top \mathbf{e}, \quad \mathbf{r}_{\text{cp}} = \mathbf{e}^\top (\mathbf{x}_1^- - \mathbf{x}_2^-).$$

For point-triangle contact, β_2, β_3 are the two independent barycentric coordinates of the closest point on triangle (2, 3, 4), with $\beta_1 = 1 - \beta_2 - \beta_3$. The matrix

$$\mathbf{E}_\Delta = [\mathbf{x}_3^- - \mathbf{x}_2^- \quad \mathbf{x}_4^- - \mathbf{x}_2^-] \in \mathbb{R}^{3 \times 2}$$

has as its columns the two triangle edges starting at vertex 2. The closest point is $\mathbf{x}_2^- + \mathbf{E}_\Delta \boldsymbol{\omega}_{\text{cp}}$. For a closest point inside the triangle, the vector $\mathbf{x}_1^- - (\mathbf{x}_2^- + \mathbf{E}_\Delta \boldsymbol{\omega}_{\text{cp}})$ is perpendicular to both triangle edges, giving

$$\mathbf{D}_{\text{cp}} = \mathbf{E}_\Delta^\top \mathbf{E}_\Delta, \quad \mathbf{r}_{\text{cp}} = \mathbf{E}_\Delta^\top (\mathbf{x}_1^- - \mathbf{x}_2^-).$$

For edge-edge contact, s_A, s_B locate the closest points along edges (1, 2) and (3, 4). With $\mathbf{e}_A = \mathbf{x}_2^- - \mathbf{x}_1^-$ and $\mathbf{e}_B = \mathbf{x}_4^- - \mathbf{x}_3^-$, the closest points are $\mathbf{x}_1^- + s_A \mathbf{e}_A$ and $\mathbf{x}_3^- + s_B \mathbf{e}_B$. For closest points in the interiors of both edges, the vector $\mathbf{x}_1^- - \mathbf{x}_3^- + s_A \mathbf{e}_A - s_B \mathbf{e}_B$ connecting them is perpendicular to both edge directions, yielding

$$\begin{aligned} \mathbf{D}_{\text{cp}} &= \begin{bmatrix} \mathbf{e}_A^\top \mathbf{e}_A & -\mathbf{e}_A^\top \mathbf{e}_B \\ -\mathbf{e}_A^\top \mathbf{e}_B & \mathbf{e}_B^\top \mathbf{e}_B \end{bmatrix}, \\ \mathbf{r}_{\text{cp}} &= \begin{bmatrix} \mathbf{e}_A^\top (\mathbf{x}_3^- - \mathbf{x}_1^-) \\ -\mathbf{e}_B^\top (\mathbf{x}_3^- - \mathbf{x}_1^-) \end{bmatrix}. \end{aligned}$$

Differentiating the linear system $\mathbf{D}_{\text{cp}} \boldsymbol{\omega}_{\text{cp}} = \mathbf{r}_{\text{cp}}$ with respect to the previous-frame vertex positions gives

$$d\boldsymbol{\omega}_{\text{cp}} = \mathbf{D}_{\text{cp}}^{-1} (d\mathbf{r}_{\text{cp}} - (d\mathbf{D}_{\text{cp}}) \boldsymbol{\omega}_{\text{cp}}).$$

The product rule applied to the explicit expressions above gives $d\mathbf{D}_{\text{cp}}$ and $d\mathbf{r}_{\text{cp}}$ from the vertex variations. The edge-edge coordinate variations ds_A and ds_B include the effects of changes in both edge directions and their relative positions. The resulting weight variations are

$$d\mathbf{w}^{\text{fr}} = \begin{cases} (0, d\eta, -d\eta), & \text{point-edge,} \\ (0, d\beta_2 + d\beta_3, -d\beta_2, -d\beta_3), & \text{point-triangle,} \\ (-ds_A, ds_A, ds_B, -ds_B), & \text{edge-edge,} \end{cases}$$

where $d\beta_1 = -d\beta_2 - d\beta_3$ has been substituted in the triangle case. These entries determine $d\mathbf{W}$ through its block definition above.

For mesh-mesh contact, let $\mathbf{r}_{\text{gap}} = \mathbf{W} \mathbf{x}_{t-1}^c$ be the vector between the closest points and $\mathbf{t}_{\text{ref}} = (\mathbf{I}_3 - \mathbf{n} \mathbf{n}^\top) \mathbf{e}_{\text{ref}}$ the reference direction projected into the contact plane. Holding $\mathbf{e}_{\text{ref}}$ fixed, differentiation of the normalized directions gives

$$\begin{aligned} d\mathbf{r}_{\text{gap}} &= (d\mathbf{W}) \mathbf{x}_{t-1}^c + \mathbf{W} d\mathbf{x}_{t-1}^c, \\ d\mathbf{n} &= \frac{\mathbf{I}_3 - \mathbf{n} \mathbf{n}^\top}{\|\mathbf{r}_{\text{gap}}\|} d\mathbf{r}_{\text{gap}}, \\ d\mathbf{t}_{\text{ref}} &= -((d\mathbf{n}) \mathbf{n}^\top + \mathbf{n} (d\mathbf{n})^\top) \mathbf{e}_{\text{ref}}, \\ d\mathbf{t}_1 &= \frac{\mathbf{I}_3 - \mathbf{t}_1 \mathbf{t}_1^\top}{\|\mathbf{t}_{\text{ref}}\|} d\mathbf{t}_{\text{ref}}, \\ d\mathbf{t}_2 &= (d\mathbf{t}_1) \times \mathbf{n} + \mathbf{t}_1 \times d\mathbf{n}, \\ d\mathbf{U} &= [d\mathbf{t}_1 \quad d\mathbf{t}_2]. \end{aligned}$$

These expressions supply the weight and contact-plane derivatives required by the history derivative. Differentiating the normal force using Equation (67) gives

$$df_n = \left[-\frac{b'_{\text{bar}}(d^-)}{\sqrt{d^-}} - 2\sqrt{d^-} b''_{\text{bar}}(d^-) \right] dd^-.$$

For the fixed ground plane, $d\mathbf{W}$ and $d\mathbf{U}$ vanish. Its history derivative includes changes in sliding displacement and in the normal force set by the previous vertex-to-plane distance.

The mapped derivative $\mathbf{K}_{t,c}^\top \mathbf{C}_{t,c}^{\text{fr},x} \mathbf{K}_{t-1,c}$ places FEM and affine contact points in the same temporal coupling. Its transposed product

contributes to $\mathbf{b}_{t-1}$ through Equation (38), carrying the effect of later friction back to earlier motion DoFs. Second-frame friction therefore contributes to initial-state gradients through the first-frame adjoint. At $t = 1$, the previous positions are initial inputs and their parameter dependence contributes directly to the initial-state gradient.

A.1.3 Rest-shape derivatives under friction. With solved motion DoFs fixed, an affine rest-shape change moves contact points at both the current and previous frames. The direct derivative therefore includes both configurations and the map from point forces to affine DoFs. These derivatives hold $\mathbf{y}_1, \dots, \mathbf{y}_T$, the friction coefficient, and the smoothing threshold fixed. The frame adjoints supply the complementary change in simulated motion.

For a surface vertex i on affine body b , let $\theta_k = \bar{x}_{i,a}$ be one component of its rest position $\bar{\mathbf{x}}_i$, where $a = 1, 2, 3$ indexes the x, y , and z components, respectively. The vector $\mathbf{e}_a$ is the unit vector along the selected coordinate axis. Equation (2) gives $\partial \mathbf{x}_{s,i} / \partial \theta_k = \mathbf{A}_{s,b} \mathbf{e}_a$ at both $s = t$ and $s = t - 1$. Thus the same rest-coordinate change moves current and previous contact points through their respective affine transformations. The point map in Equation (3) also changes:

$$\frac{\partial \mathbf{J}_i}{\partial \theta_k} = [\mathbf{0}_{3 \times 3} \quad \mathbf{e}_1 \mathbf{e}_a^\top \quad \mathbf{e}_2 \mathbf{e}_a^\top \quad \mathbf{e}_3 \mathbf{e}_a^\top].$$

For a contact containing vertex i , this block occupies the rows of that vertex and the columns of body b in $\partial \mathbf{K}_{t,c} / \partial \theta_k$. For a shape parameter controlling several rest coordinates, its derivative follows by applying the chain rule through the rest-shape parameterization.

The induced change in the point-space friction gradient is

$$d\mathbf{g}_{t,c}^{\text{fr},x} = \mathbf{H}_{t,c}^{\text{fr},x} d\mathbf{x}_t^c + \mathbf{C}_{t,c}^{\text{fr},x} d\mathbf{x}_{t-1}^c. \quad (74)$$

The current-position term uses the Hessian obtained by differentiating Equation (70). The previous-position term uses Equation (73), including the changes in weights, basis, and normal force derived above. Combining these geometry terms with the point-map derivative gives Equation (47). For contact c at frame t , the friction contribution $\Delta \bar{\theta}_k^{(t,c),\text{fr}}$ to the loss gradient is

$$\begin{aligned} \Delta \bar{\theta}_k^{(t,c),\text{fr}} = & -(\mathbf{g}_{t,c}^{\text{fr},x})^\top \frac{\partial \mathbf{K}_{t,c}}{\partial \theta_k} \boldsymbol{\lambda}_t \\ & - \left(\frac{\partial \mathbf{x}_t^c}{\partial \theta_k} \right)^\top \mathbf{H}_{t,c}^{\text{fr},x} \mathbf{K}_{t,c} \boldsymbol{\lambda}_t \\ & - \left(\frac{\partial \mathbf{x}_{t-1}^c}{\partial \theta_k} \right)^\top (\mathbf{C}_{t,c}^{\text{fr},x})^\top \mathbf{K}_{t,c} \boldsymbol{\lambda}_t. \end{aligned}$$

The three terms account for the force map and the contact geometry at both time levels. Their sum over the relevant contacts and frames is added to the other physical contributions and any direct loss derivative to obtain the total rest-shape gradient $\bar{\theta}_k$.

For free FEM vertices, the solved positions are the motion DoFs and their point maps are identity matrices. The effect of rest shape on simulated motion is propagated through the frame adjoints. Affine surfaces also contribute the direct geometry and point-map derivatives above. If rest shape determines the initial vertex positions, their direct contribution is included in the first-frame friction derivatives with respect to those initial positions.

A.2 Volumetric FEM and cloth parameter derivatives

Element energies supply both the frame Hessian and the parameter derivatives needed for material identification and deformable rest-shape design. Volumetric elements use stable Neo-Hookean [Smith et al. 2018]. Cloth combines the finite-element Baraff–Witkin [Kim 2020] with StiffGIPC’s cubic extension penalty. Differentiating these energies with respect to material and rest-shape parameters supplies local gradient contributions at fixed current positions. The coupled adjoint accounts for the resulting deformation and its interactions with other objects.

A.2.1 Tetrahedral elasticity. For a tetrahedron with local vertex indices 0, 1, 2, 3, define its rest and current edge matrices as

$$\begin{aligned} \mathbf{D}_e^{\text{rest}} &= [\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_0 \quad \bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_0 \quad \bar{\mathbf{x}}_3 - \bar{\mathbf{x}}_0], \\ \mathbf{D}_e^{\text{cur}} &= [\mathbf{x}_1 - \mathbf{x}_0 \quad \mathbf{x}_2 - \mathbf{x}_0 \quad \mathbf{x}_3 - \mathbf{x}_0]. \end{aligned}$$

Its deformation gradient is $\mathbf{F}_e = \mathbf{D}_e^{\text{cur}} (\mathbf{D}_e^{\text{rest}})^{-1}$, and its rest volume is $V_e = |\det \mathbf{D}_e^{\text{rest}}|/6$. Let $J_e = \det \mathbf{F}_e$. The stable Neo-Hookean [Smith et al. 2018] is

$$\psi_e = \frac{\mu}{2} (\|\mathbf{F}_e\|_F^2 - 3) + \frac{\lambda}{2} \left(J_e - 1 - \frac{\mu}{\lambda} \right)^2.$$

The coefficients are per element, with element subscript suppressed. The incremental-potential contribution is $h^2 V_e \psi_e$. The determinant derivative is $\partial J_e / \partial \mathbf{F}_e = \text{cof}(\mathbf{F}_e)$, where $\text{cof}(\mathbf{F}_e)$ denotes the cofactor matrix. Differentiating the energy density with respect to $\mathbf{F}_e$ gives the first Piola stress:

$$\mathbf{P}_e = \frac{\partial \psi_e}{\partial \mathbf{F}_e} = \mu \mathbf{F}_e + [\lambda(J_e - 1) - \mu] \text{cof}(\mathbf{F}_e).$$

Let vec stack matrix columns and $\mathbf{H}_{F,e} = \partial \text{vec}(\mathbf{P}_e) / \partial \text{vec}(\mathbf{F}_e)$. The same geometric Jacobian maps stress, constitutive Hessian, and material derivatives into vertex DoFs:

$$\begin{aligned} \mathbf{R}_e &= \frac{\partial \text{vec}(\mathbf{F}_e)}{\partial \mathbf{x}_e}, \quad \mathbf{g}_e = h^2 V_e \mathbf{R}_e^\top \text{vec}(\mathbf{P}_e), \\ \mathbf{H}_e &= h^2 V_e \mathbf{R}_e^\top \mathbf{H}_{F,e} \mathbf{R}_e. \end{aligned} \quad (75)$$

Here $\mathbf{x}_e$ stacks the four current vertex positions and $\mathbf{R}_e \in \mathbb{R}^{9 \times 12}$ is constant with respect to them. With $\mathbf{d}_{e,a} = (\mathbf{D}_e^{\text{rest}})^{-\top} \mathbf{e}_a$ for $a = 1, 2, 3$ and $\mathbf{d}_{e,0} = -\sum_{a=1}^3 \mathbf{d}_{e,a}$, $\mathbf{R}_e = [\mathbf{d}_{e,0} \otimes \mathbf{I}_3 \quad \dots \quad \mathbf{d}_{e,3} \otimes \mathbf{I}_3]$, where $\otimes$ is the Kronecker product.

The constitutive Hessian follows by differentiating the stress:

$$\begin{aligned} d\mathbf{P}_e &= \mu d\mathbf{F}_e + \lambda(dJ_e) \text{cof}(\mathbf{F}_e) \\ &\quad + [\lambda(J_e - 1) - \mu] d \text{cof}(\mathbf{F}_e), \\ dJ_e &= \text{cof}(\mathbf{F}_e) : d\mathbf{F}_e. \end{aligned}$$

The colon is the Frobenius inner product. For $\mathbf{F}_e = [\mathbf{f}_1 \ \mathbf{f}_2 \ \mathbf{f}_3]$, the cofactor columns are $\mathbf{f}_2 \times \mathbf{f}_3$, $\mathbf{f}_3 \times \mathbf{f}_1$, and $\mathbf{f}_1 \times \mathbf{f}_2$. Their differentials follow the cross-product rule, e.g., $d(\mathbf{f}_2 \times \mathbf{f}_3) = d\mathbf{f}_2 \times \mathbf{f}_3 + \mathbf{f}_2 \times d\mathbf{f}_3$. These expressions determine every entry of $\mathbf{H}_{F,e}$.

At fixed $\mathbf{F}_e$, the material derivatives are

$$\begin{aligned} \frac{\partial \mathbf{P}_e}{\partial \mu} &= \mathbf{F}_e - \text{cof}(\mathbf{F}_e), \\ \frac{\partial \mathbf{P}_e}{\partial \lambda} &= (J_e - 1) \text{cof}(\mathbf{F}_e). \end{aligned} \quad (76)$$

Their contributions to $\partial \mathbf{g}_e / \partial \theta_k$ are $h^2 V_e \mathbf{R}_e^\top \text{vec}(\partial \mathbf{P}_e / \partial \theta_k)$. The material parameterization relates Young’s modulus E and Poisson’s

ratio ν to the energy coefficients through the Lamé input parameters $\mu_{\text{Lamé}} = E/[2(1+\nu)]$ and $\lambda_{\text{Lamé}} = E\nu/[(1+\nu)(1-2\nu)]$, followed by $\mu = \mu_{\text{Lamé}}$ and $\lambda = \lambda_{\text{Lamé}} + \mu_{\text{Lamé}}$. Gradients with respect to E , ν , or the optimizer variables in Equation (65) follow through these maps.

Rest-shape design changes both the reference geometry and the elastic response. Rest-shape differentiation accounts for changes in the inverse rest-edge matrix $(\mathbf{D}_e^{\text{rest}})^{-1}$, the element rest volume V_e , and the deformation-gradient Jacobian $\mathbf{R}_e$. With current positions fixed,

$$\begin{aligned} d(\mathbf{D}_e^{\text{rest}})^{-1} &= -(\mathbf{D}_e^{\text{rest}})^{-1}(d\mathbf{D}_e^{\text{rest}})(\mathbf{D}_e^{\text{rest}})^{-1}, \\ d\mathbf{F}_e &= \mathbf{D}_e^{\text{cur}} d(\mathbf{D}_e^{\text{rest}})^{-1}, \\ dV_e &= V_e \text{tr}[(\mathbf{D}_e^{\text{rest}})^{-1} d\mathbf{D}_e^{\text{rest}}]. \end{aligned} \quad (77)$$

Differentiating the block expression for $\mathbf{R}_e$ gives $d\mathbf{R}_e$. The complete elastic contribution is

$$\begin{aligned} d\mathbf{g}_e &= h^2 \left[(dV_e) \mathbf{R}_e^T \text{vec}(\mathbf{P}_e) \right. \\ &\quad + V_e (d\mathbf{R}_e)^T \text{vec}(\mathbf{P}_e) \\ &\quad \left. + V_e \mathbf{R}_e^T \text{vec}(d\mathbf{P}_e) \right]. \end{aligned}$$

The three terms account for variations in the rest volume V_e , the deformation-gradient Jacobian $\mathbf{R}_e$, and the first Piola stress $\mathbf{P}_e$, respectively. The adjoint converts their effect on the frame equations into loss gradients. For stress-based objectives such as Loaded Support (Figure 8), the loss's explicit dependence on rest geometry contributes separately through ℓ_θ .

A.2.2 Cloth membrane elasticity. For a nondegenerate triangle with local vertices 0, 1, 2, define the rest edges $\bar{\mathbf{e}}_1 = \bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_0$ and $\bar{\mathbf{e}}_2 = \bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_0$, and the current edge matrix $\mathbf{D}_e^{\text{cur}} = [\mathbf{x}_1 - \mathbf{x}_0 \quad \mathbf{x}_2 - \mathbf{x}_0] \in \mathbb{R}^{3 \times 2}$. Aligning the local horizontal axis with the rest edge $\bar{\mathbf{e}}_1$ gives

$$\mathbf{D}_e^{\text{rest}} = \begin{bmatrix} \|\bar{\mathbf{e}}_1\| & \frac{\bar{\mathbf{e}}_1^T \bar{\mathbf{e}}_2}{\|\bar{\mathbf{e}}_1\|} \\ 0 & \frac{\|\bar{\mathbf{e}}_1 \times \bar{\mathbf{e}}_2\|}{\|\bar{\mathbf{e}}_1\|} \end{bmatrix}.$$

The entries depend on edge lengths and their included angle, so a rigid rotation of the rest triangle leaves $\mathbf{D}_e^{\text{rest}}$ unchanged. At a fixed rest shape, this matrix remains constant throughout the forward trajectory. The deformation gradient is $\mathbf{F}_e = \mathbf{D}_e^{\text{cur}} (\mathbf{D}_e^{\text{rest}})^{-1} = [\mathbf{f}_1 \quad \mathbf{f}_2]$. Let $r_a = \|\mathbf{f}_a\|$ and $c_e = \mathbf{f}_1^T \mathbf{f}_2$. The two local coordinate axes specify the orthogonal warp and weft directions in the stretch and shear terms of Kim [2020]. Including StiffGIPC's cubic extension penalty [Huang et al. 2025a] gives

$$\begin{aligned} \psi_e^{\text{mem}} &= \kappa_e^{\text{str}} [s(r_1) + s(r_2)] + \kappa_e^{\text{shr}} c_e^2, \\ s(r) &= (r-1)^2 + \eta_{\text{sl},e} \langle r-1 \rangle_+^3. \end{aligned}$$

Here $\langle z \rangle_+ = \max(z, 0)$ and $\eta_{\text{sl},e}$ is the fixed cubic-penalty coefficient. The cubic term adds resistance in extension, with its derivatives included in the same stress and Hessian. The incremental potential is $h^2 A_e \psi_e^{\text{mem}}$, with A_e the rest area. The constitutive parameterization is $\kappa_e^{\text{str}} = E_{\text{str}} \tau_e / (1 - \nu_{\text{str}}^2)$ and $\kappa_e^{\text{shr}} = E_{\text{shr}} / [2(1 + \nu_{\text{shr}})]$.

Define $a(r) = s'(r)/r$, where $s'(r) = 2(r-1) + 3\eta_{\text{sl},e} \langle r-1 \rangle_+^2$. Since $\partial r_a / \partial \mathbf{f}_a = \mathbf{f}_a / r_a$ and $\partial c_e / \partial \mathbf{f}_1 = \mathbf{f}_2$, the stress columns are

$$\begin{aligned} \mathbf{P}_e(:, 1) &= \kappa_e^{\text{str}} a(r_1) \mathbf{f}_1 + 2\kappa_e^{\text{shr}} c_e \mathbf{f}_2, \\ \mathbf{P}_e(:, 2) &= \kappa_e^{\text{str}} a(r_2) \mathbf{f}_2 + 2\kappa_e^{\text{shr}} c_e \mathbf{f}_1. \end{aligned}$$

For $a_a = a(r_a)$ and $a'_a = s''(r_a)/r_a - s'(r_a)/r_a^2$, with $s''(r) = 2 + 6\eta_{\text{sl},e} \langle r-1 \rangle_+$, the four 3×3 blocks of $\mathbf{H}_{F,e}$ are

$$\begin{aligned} (\mathbf{H}_{F,e})_{11} &= \kappa_e^{\text{str}} \left(a_1 \mathbf{I}_3 + \frac{a'_1}{r_1} \mathbf{f}_1 \mathbf{f}_1^T \right) + 2\kappa_e^{\text{shr}} \mathbf{f}_2 \mathbf{f}_2^T, \\ (\mathbf{H}_{F,e})_{22} &= \kappa_e^{\text{str}} \left(a_2 \mathbf{I}_3 + \frac{a'_2}{r_2} \mathbf{f}_2 \mathbf{f}_2^T \right) + 2\kappa_e^{\text{shr}} \mathbf{f}_1 \mathbf{f}_1^T, \\ (\mathbf{H}_{F,e})_{12} &= 2\kappa_e^{\text{shr}} (\mathbf{f}_2 \mathbf{f}_1^T + c_e \mathbf{I}_3), \\ (\mathbf{H}_{F,e})_{21} &= (\mathbf{H}_{F,e})_{12}^T. \end{aligned}$$

The parameter derivatives follow from the linear dependence of the stress on the two stiffnesses:

$$\begin{aligned} \frac{\partial \mathbf{P}_e}{\partial \kappa_e^{\text{str}}} &= [\mathbf{a}_1 \mathbf{f}_1 \quad \mathbf{a}_2 \mathbf{f}_2], \\ \frac{\partial \mathbf{P}_e}{\partial \kappa_e^{\text{shr}}} &= 2c_e [\mathbf{f}_2 \quad \mathbf{f}_1]. \end{aligned}$$

The assembly in Equation (75) maps these constitutive derivatives to cloth vertex DoFs, using $\mathbf{R}_e \in \mathbb{R}^{6 \times 9}$ and replacing V_e by A_e .

Rest-shape derivatives follow directly from the edge-based reference coordinates. Let $L_{e,1} = \|\bar{\mathbf{e}}_1\|$ be the first edge length, and let $(u_{e,2}, v_{e,2})$ be vertex 2's local coordinates, given by the second column of $\mathbf{D}_e^{\text{rest}}$. With rest area $A_e = \|\bar{\mathbf{e}}_1 \times \bar{\mathbf{e}}_2\|/2$ and unit normal $\bar{\mathbf{n}} = (\bar{\mathbf{e}}_1 \times \bar{\mathbf{e}}_2)/(2A_e)$, the geometric differentials are

$$\begin{aligned} dL_{e,1} &= \bar{\mathbf{e}}_1^T d\bar{\mathbf{e}}_1 / L_{e,1}, \\ dA_e &= \frac{1}{2} \bar{\mathbf{n}}^T (d\bar{\mathbf{e}}_1 \times \bar{\mathbf{e}}_2 + \bar{\mathbf{e}}_1 \times d\bar{\mathbf{e}}_2), \\ du_{e,2} &= \frac{(\bar{\mathbf{e}}_1^T d\bar{\mathbf{e}}_2 + \bar{\mathbf{e}}_1^T d\bar{\mathbf{e}}_2)}{L_{e,1}} - \frac{u_{e,2}}{L_{e,1}} dL_{e,1}, \\ dv_{e,2} &= \frac{2 dA_e}{L_{e,1}} - \frac{v_{e,2}}{L_{e,1}} dL_{e,1}. \end{aligned}$$

Consequently,

$$d\mathbf{D}_e^{\text{rest}} = \begin{bmatrix} dL_{e,1} & du_{e,2} \\ 0 & dv_{e,2} \end{bmatrix}.$$

The inverse-matrix identity in Equation (77) gives $d(\mathbf{D}_e^{\text{rest}})^{-1}$. At fixed current positions, $d\mathbf{F}_e = \mathbf{D}_e^{\text{cur}} d(\mathbf{D}_e^{\text{rest}})^{-1}$ changes the constitutive stress, and the same inverse-map derivative changes $\mathbf{R}_e$. dA_e supplies the integration-weight variation.

A.2.3 Cloth bending derivatives. The quadratic bending model of Bergou et al. [2006] uses a geometric matrix determined by the adjacent triangle areas and cotangent weights. This matrix supplies the position gradient, Hessian, and bending-stiffness derivative. Its rest-shape derivative includes changes in both the area normalization and the cotangent weights. An interior cloth edge defines a hinge with rest vertices $(\bar{\mathbf{x}}_0, \bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \bar{\mathbf{x}}_3)$, where 0, 1 are endpoints and 2, 3 are opposite vertices. Let $\bar{\mathbf{e}}_0 = \bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_0$, $\bar{\mathbf{e}}_1 = \bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_0$, and $\bar{\mathbf{e}}_2 = \bar{\mathbf{x}}_3 - \bar{\mathbf{x}}_0$. The adjacent rest areas are $A_1 = \|\bar{\mathbf{e}}_0 \times \bar{\mathbf{e}}_1\|/2$ and $A_2 = \|\bar{\mathbf{e}}_0 \times \bar{\mathbf{e}}_2\|/2$. Using $\cot(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T \mathbf{b} / \|\mathbf{a} \times \mathbf{b}\|$, define

$$\begin{aligned} c_{01} &= \cot(\bar{\mathbf{e}}_0, \bar{\mathbf{e}}_1), & c_{02} &= \cot(\bar{\mathbf{e}}_0, \bar{\mathbf{e}}_2), \\ c_{03} &= \cot(-\bar{\mathbf{e}}_0, \bar{\mathbf{e}}_1 - \bar{\mathbf{e}}_0), & c_{04} &= \cot(-\bar{\mathbf{e}}_0, \bar{\mathbf{e}}_2 - \bar{\mathbf{e}}_0), \\ \mathbf{k}_e &= \begin{bmatrix} c_{03} + c_{04} \\ c_{01} + c_{02} \\ -c_{01} - c_{03} \\ -c_{02} - c_{04} \end{bmatrix}, & s_e &= \frac{3}{A_1 + A_2}. \end{aligned}$$

The geometric matrix is $\mathbf{Q}_e^{\text{bend}} = s_e(\mathbf{k}_e \mathbf{k}_e^\top) \otimes \mathbf{I}_3$. With $\mathbf{x}_e \in \mathbb{R}^{12}$ stacking current hinge vertices, the energy is

$$E_e^{\text{bend}} = \frac{1}{2} \kappa_e^{\text{bend}} \mathbf{x}_e^\top \mathbf{Q}_e^{\text{bend}} \mathbf{x}_e. \quad (78)$$

Its contribution to the incremental potential is $h^2 E_e^{\text{bend}}$. The hinge coefficient is the average of its adjacent triangles' bending stiffnesses $E_{\text{bend}} \tau^3 / [12(1 - \nu_{\text{bend}}^2)]$. The matrix $\mathbf{Q}_e^{\text{bend}}$ is symmetric and fixed during differentiation with respect to current positions, giving

$$\begin{aligned} \mathbf{g}_e^{\text{bend}} &= h^2 \kappa_e^{\text{bend}} \mathbf{Q}_e^{\text{bend}} \mathbf{x}_e, & \mathbf{H}_e^{\text{bend}} &= h^2 \kappa_e^{\text{bend}} \mathbf{Q}_e^{\text{bend}}, \\ \frac{\partial \mathbf{g}_e^{\text{bend}}}{\partial \kappa_e^{\text{bend}}} &= h^2 \mathbf{Q}_e^{\text{bend}} \mathbf{x}_e. \end{aligned}$$

At fixed stiffness and current positions, the rest derivative is $d\mathbf{g}_e^{\text{bend}} = h^2 \kappa_e^{\text{bend}} (d\mathbf{Q}_e^{\text{bend}}) \mathbf{x}_e$, where

$$\begin{aligned} d\mathbf{Q}_e^{\text{bend}} &= [(ds_e) \mathbf{k}_e \mathbf{k}_e^\top + s_e (d\mathbf{k}_e) \mathbf{k}_e^\top \\ &\quad + s_e \mathbf{k}_e (d\mathbf{k}_e)^\top] \otimes \mathbf{I}_3, \\ ds_e &= -s_e \frac{dA_1 + dA_2}{A_1 + A_2}. \end{aligned}$$

The area derivatives use the triangle formula above. Each cotangent derivative is explicit:

$$\begin{aligned} d \cot(\mathbf{a}, \mathbf{b}) &= \frac{(d\mathbf{a})^\top \mathbf{b} + \mathbf{a}^\top d\mathbf{b}}{\|\mathbf{a} \times \mathbf{b}\|} \\ &\quad - \frac{(\mathbf{a}^\top \mathbf{b})(\mathbf{a} \times \mathbf{b})^\top (d\mathbf{a} \times \mathbf{b} + \mathbf{a} \times d\mathbf{b})}{\|\mathbf{a} \times \mathbf{b}\|^3}. \end{aligned}$$

These cotangent derivatives determine $d\mathbf{k}_e$ and complete the hinge's rest-shape derivative.

Rest-shape gradients also account for the mass redistribution induced by geometry changes. A tetrahedron contributes $\rho_e V_e / 4$ to each incident vertex, and a cloth triangle contributes $\rho_e A_e \tau_e / 3$. Differentiating these weights gives $d\mathbf{M}$ and the inertial contribution

$$d\mathbf{g}_t^{\text{in}} = (d\mathbf{M})(\mathbf{y}_t - \hat{\mathbf{y}}_t) - \mathbf{M} d\hat{\mathbf{y}}_t$$

at fixed solved states. The second term accounts for parameter-dependent positions or velocities in the predictor. Elastic, bending, and inertial terms provide local contributions to $\mathbf{G}_t^\top \boldsymbol{\lambda}_t$ in Equation (33). Their products share the same coupled adjoint and can be accumulated without forming the global parameter-derivative matrix, as described in Section 6.2.

A.3 Joint geometry and control derivatives

Joint targets and forces influence the coupled scene through the same frame equations as collision and friction. Geometric constraints maintain the connection between the two bodies p and c . The control energy sets a target for the joint's permitted motion or applies a joint force. Differentiating both energies in affine coordinates preserves their cross-body Hessian blocks, allowing the adjoint to connect a joint command to the response of interacting FEM and affine objects. The local expressions use $\mathbf{q} = [\mathbf{q}_p^\top, \mathbf{q}_c^\top]^\top \in \mathbb{R}^{24}$ and $\nabla = \nabla_{\mathbf{q}}$. Joint anchors, reference directions, and stiffnesses are held fixed when differentiating control parameters.

A.3.1 Local maps and geometric constraints. A material-space anchor $\bar{\mathbf{x}}_b$ has world position $\mathbf{J}_b \mathbf{q}_b$, where $\mathbf{J}_b = \mathbf{J}(\bar{\mathbf{x}}_b)$ is the point map from Equation (3). A material-space direction $\bar{\mathbf{a}}$ has no translation component. Its map is

$$\mathbf{D}(\bar{\mathbf{a}}) = [\mathbf{0}_{3 \times 3} \quad \mathbf{e}_1 \bar{\mathbf{a}}^\top \quad \mathbf{e}_2 \bar{\mathbf{a}}^\top \quad \mathbf{e}_3 \bar{\mathbf{a}}^\top], \quad (79)$$

so $\mathbf{D}(\bar{\mathbf{a}}) \mathbf{q}_b = \mathbf{A}_b \bar{\mathbf{a}}$. The joint energy uses these affine-transformed directions directly. For each body, denote the transformed joint axis and two transverse reference directions by $\mathbf{a}_b$, $\mathbf{n}_b$, and $\mathbf{m}_b$. Let $\Delta = \mathbf{J}_c \mathbf{q}_c - \mathbf{J}_p \mathbf{q}_p$.

A revolute joint matches two distinct points on its hinge line. For endpoint $i \in \{0, 1\}$, define $\Delta_i = \mathbf{J}_{c,i} \mathbf{q}_c - \mathbf{J}_{p,i} \mathbf{q}_p$, using the two bodies' material-space copies of that endpoint. A prismatic joint constrains the anchor separation to lie along the joint axes and matches transverse directions. The geometric residuals are

$$\mathbf{c}_g^{\text{rev}} = \begin{bmatrix} \Delta_0 \\ \Delta_1 \end{bmatrix}, \quad \mathbf{c}_g^{\text{pri}} = \begin{bmatrix} \Delta \times \mathbf{a}_p \\ -\Delta \times \mathbf{a}_c \\ \mathbf{n}_p - \mathbf{n}_c \\ \mathbf{m}_p - \mathbf{m}_c \end{bmatrix}.$$

A fixed joint uses anchor coincidence and matching transverse directions: $\mathbf{c}_g^{\text{fix}} = [-\Delta^\top, (\mathbf{n}_p - \mathbf{n}_c)^\top, (\mathbf{m}_p - \mathbf{m}_c)^\top]^\top$. For any of these residuals, the constraint energy is $\Phi_g = \frac{1}{2} k_g \|\mathbf{c}_g\|^2$. With $\mathbf{J}_g = \partial \mathbf{c}_g / \partial \mathbf{q}$, the chain and product rules give

$$\begin{aligned} \nabla \Phi_g &= k_g \mathbf{J}_g^\top \mathbf{c}_g, \\ \nabla^2 \Phi_g &= k_g \left(\mathbf{J}_g^\top \mathbf{J}_g + \sum_m c_{g,m} \nabla^2 c_{g,m} \right). \end{aligned} \quad (80)$$

The affine coordinate maps make point- and direction-matching residuals linear in $\mathbf{q}$, with zero second derivatives. Prismatic cross-product residuals are bilinear and supply additional cross-body Hessian terms. The two cross-product terms in $\mathbf{c}_g^{\text{pri}}$, $\Delta \times \mathbf{a}_p$ and $-\Delta \times \mathbf{a}_c$, contribute second derivatives to the constraint-energy Hessian. To derive these terms, let $\mathbf{c}_g^\times$ denote either cross-product residual:

$$\mathbf{c}_g^\times = (\mathbf{B}_1 \mathbf{q}) \times (\mathbf{B}_2 \mathbf{q}).$$

With the joint reference geometry fixed, $\mathbf{B}_1, \mathbf{B}_2 \in \mathbb{R}^{3 \times 24}$ are constant with respect to $\mathbf{q}$. For the first prismatic block they are $\mathbf{B}_1 = [-\mathbf{J}_p \mathbf{J}_c]$ and $\mathbf{B}_2 = [\mathbf{D}(\bar{\mathbf{a}}_p) \mathbf{0}]$. For the second block, they are $\mathbf{B}_1 = [\mathbf{J}_p \quad -\mathbf{J}_c]$ and $\mathbf{B}_2 = [\mathbf{0} \quad \mathbf{D}(\bar{\mathbf{a}}_c)]$. Writing $[\mathbf{v}]_\times \mathbf{w} = \mathbf{v} \times \mathbf{w}$, the block Jacobian is

$$\mathbf{J}_\times = -[\mathbf{B}_2 \mathbf{q}]_\times \mathbf{B}_1 + [\mathbf{B}_1 \mathbf{q}]_\times \mathbf{B}_2.$$

For component m , let $\mathbf{E}_m = -[\mathbf{e}_m]_\times$, so $\mathbf{u}^\top \mathbf{E}_m \mathbf{v} = (\mathbf{u} \times \mathbf{v})_m$. A second differentiation yields

$$\nabla^2 c_{g,m}^\times = \mathbf{B}_1^\top \mathbf{E}_m \mathbf{B}_2 + \mathbf{B}_2^\top \mathbf{E}_m \mathbf{B}_1. \quad (81)$$

These constant component Hessians supply the residual-weighted term in Equation (80).

A.3.2 Prismatic displacement. The prismatic coordinate depends on the vector joining the two joint anchors and the world-space joint-axis vector of body c . Differentiating both vectors accounts for the motion of both connected bodies in its gradient and Hessian.

Let $\mathbf{D}_c = \mathbf{D}(\bar{\mathbf{a}}_c)$ and $\mathbf{a} = \mathbf{D}_c \mathbf{q}_c$. Differentiating $\xi = \Delta^\top \mathbf{a}$ gives $d\xi = (\mathbf{d}\Delta)^\top \mathbf{a} + \Delta^\top d\mathbf{a}$, hence

$$\xi = \Delta^\top \mathbf{a}, \quad \nabla \xi = \begin{bmatrix} -\mathbf{J}_p^\top \mathbf{a} \\ \mathbf{J}_c^\top \mathbf{a} + \mathbf{D}_c^\top \Delta \end{bmatrix}. \quad (82)$$

The gradient with respect to $\mathbf{q}_c$ includes both anchor motion and axis variation. Differentiating $\nabla \xi$ with respect to $\mathbf{q}$ gives

$$\nabla^2 \xi = \begin{bmatrix} \mathbf{0}_{12 \times 12} & -\mathbf{J}_p^\top \mathbf{D}_c \\ -\mathbf{D}_c^\top \mathbf{J}_p & \mathbf{J}_c^\top \mathbf{D}_c + \mathbf{D}_c^\top \mathbf{J}_c \end{bmatrix}. \quad (83)$$

The bilinear coordinate gives a constant Hessian at fixed joint geometry. Its off-diagonal blocks retain the dependence on both connected bodies.

A.3.3 Revolute orientation. Revolute actuation is expressed directly in the two bodies' affine coordinates. Bilinear orientation quantities give explicit gradients and constant cross-body Hessian blocks, retaining the coupled angular response in compact motion coordinates. Let $\mathbf{n}_b = \mathbf{D}(\bar{\mathbf{n}}_b) \mathbf{q}_b$ and $\mathbf{m}_b = \mathbf{D}(\bar{\mathbf{m}}_b) \mathbf{q}_b$ be the two transformed reference directions on body $b \in \{p, c\}$. The orientation quantities used by the control energy are

$$\begin{aligned} s_r &= \frac{1}{2} (\mathbf{n}_c^\top \mathbf{m}_p - \mathbf{m}_c^\top \mathbf{n}_p), \\ c_r &= \frac{1}{2} (\mathbf{n}_p^\top \mathbf{n}_c + \mathbf{m}_p^\top \mathbf{m}_c). \end{aligned} \quad (84)$$

The two quantities measure relative turning and alignment. For rigid rotations with orthonormal reference frames and aligned joint axes, they equal the sine and cosine of the relative angle. The target energy uses their affine expressions directly, so the derivatives retain changes in both the lengths and orientations of the transformed reference vectors.

To express their bilinear form, let $\mathbf{D}_{n,b} = \mathbf{D}(\bar{\mathbf{n}}_b)$ and $\mathbf{D}_{m,b} = \mathbf{D}(\bar{\mathbf{m}}_b)$, and define

$$\begin{aligned} \mathbf{Q}_s &= \frac{1}{2} (\mathbf{D}_{m,p}^\top \mathbf{D}_{n,c} - \mathbf{D}_{n,p}^\top \mathbf{D}_{m,c}), \\ \mathbf{Q}_c &= \frac{1}{2} (\mathbf{D}_{n,p}^\top \mathbf{D}_{n,c} + \mathbf{D}_{m,p}^\top \mathbf{D}_{m,c}). \end{aligned} \quad (85)$$

Then $s_r = \mathbf{q}_p^\top \mathbf{Q}_s \mathbf{q}_c$ and $c_r = \mathbf{q}_p^\top \mathbf{Q}_c \mathbf{q}_c$, giving

$$\nabla s_r = \begin{bmatrix} \mathbf{Q}_s \mathbf{q}_c \\ \mathbf{Q}_s^\top \mathbf{q}_p \end{bmatrix}, \quad \nabla c_r = \begin{bmatrix} \mathbf{Q}_c \mathbf{q}_c \\ \mathbf{Q}_c^\top \mathbf{q}_p \end{bmatrix}, \quad (86)$$

and

$$\nabla^2 s_r = \begin{bmatrix} \mathbf{0} & \mathbf{Q}_s \\ \mathbf{Q}_s^\top & \mathbf{0} \end{bmatrix}, \quad \nabla^2 c_r = \begin{bmatrix} \mathbf{0} & \mathbf{Q}_c \\ \mathbf{Q}_c^\top & \mathbf{0} \end{bmatrix}. \quad (87)$$

The joint angle is $\alpha = \text{atan2}(s_r, c_r)$. The target quantities S_r, C_r are defined in Equation (54). For $s_r^2 + c_r^2 > 0$ on the selected angle branch, differentiation gives

$$\begin{aligned} \nabla \alpha &= \frac{c_r \nabla s_r - s_r \nabla c_r}{s_r^2 + c_r^2}, \\ \nabla S_r &= \cos \alpha \nabla s_r - \sin \alpha \nabla c_r, \\ \nabla C_r &= \cos \alpha \nabla c_r + \sin \alpha \nabla s_r. \end{aligned} \quad (88)$$

At fixed applied target a , the second derivatives of the target quantities use the same linear combinations:

$$\begin{aligned} \nabla^2 S_r &= \cos \alpha \nabla^2 s_r - \sin \alpha \nabla^2 c_r, \\ \nabla^2 C_r &= \cos \alpha \nabla^2 c_r + \sin \alpha \nabla^2 s_r. \end{aligned} \quad (89)$$

For the angle, set $d_\alpha = s_r^2 + c_r^2$, $\mathbf{u}_s = \nabla s_r$, and $\mathbf{u}_c = \nabla c_r$. Differentiating the numerator and denominator of $\nabla \alpha$ yields

$$\begin{aligned} \nabla^2 \alpha &= \frac{c_r \nabla^2 s_r - s_r \nabla^2 c_r}{d_\alpha} \\ &\quad + \frac{2s_r c_r}{d_\alpha^2} (\mathbf{u}_c \mathbf{u}_c^\top - \mathbf{u}_s \mathbf{u}_s^\top) \\ &\quad + \frac{s_r^2 - c_r^2}{d_\alpha^2} (\mathbf{u}_s \mathbf{u}_c^\top + \mathbf{u}_c \mathbf{u}_s^\top). \end{aligned} \quad (90)$$

A.3.4 Control derivatives and coupled assembly. The coordinate derivatives determine the Hessians of the joint energies in Section 5.4 with respect to the motion DoFs. Differentiating the energy gradients with respect to joint targets and joint forces gives the control derivatives. For a prismatic joint, the scalar multiplying $\nabla \xi$ is $k(\xi - a) - f$. Its derivatives with respect to a and f are $-k$ and -1 , giving Equation (53). For a revolute target, $\partial_a S_r = -C_r$ and $\partial_a C_r = S_r$. Differentiating their state gradients also gives $\partial_a \nabla S_r = -\nabla C_r$ and $\partial_a \nabla C_r = \nabla S_r$. Substitution into the product rule yields Equation (56). The joint-force terms use coordinate gradients $-\nabla \xi$ and $-\nabla \alpha$. At fixed a, f , and k , differentiating the state gradients gives the full control-energy Hessians:

$$\begin{aligned} \mathbf{H}^{\text{pri}} &= k \nabla \xi \nabla \xi^\top + [k(\xi - a) - f] \nabla^2 \xi, \\ \mathbf{H}^{\text{rev}} &= k (\nabla S_r \nabla S_r^\top + S_r \nabla^2 S_r) \\ &\quad + k w_c (\nabla C_r \nabla C_r^\top - (1 - C_r) \nabla^2 C_r) \\ &\quad - f \nabla^2 \alpha. \end{aligned} \quad (91)$$

Here w_c is the fixed cosine weight from Equation (55).

Geometric and control Hessians use the same $[\mathbf{q}_p^\top, \mathbf{q}_c^\top]^\top$ ordering. Their off-diagonal blocks couple the joint-connected bodies within the frame matrix, alongside the contact blocks coupling them to the rest of the scene. The local control derivative is multiplied by the corresponding free-body entries of the global adjoint in Equation (52). A shared control accumulates contributions from every joint and frame to which it is applied. All selected controls therefore use the same frame adjoints rather than separate coupled solves for individual control variables.

B Experimental Settings and Parameters

Tables 6 and 7 summarize the simulation settings and differentiated parameter families, respectively, for the experiments in Section 7. The task losses define the observation frames and optimization stopping events. For tasks with separate optimization and verification phases, the differentiated trajectory provides the gradient at the specified observation or event, and the full listed horizon assesses the resulting motion. Forward and backward evaluation counts follow the definitions in Section 7.

For scene diagonal L_{box} , the collision activation gap is $d_{\text{rel}} L_{\text{box}}$, whose square is $\hat{d}$ in the barrier energy. The friction-smoothing velocity $\hat{d}_f$ gives the displacement scale $\epsilon_{\text{fr}} = h \hat{d}_f$. The coefficients κ and κ_A control the collision barrier and affine shape energy, respectively. With $\delta \mathbf{x}$ denoting the Newton search direction in world coordinates, the stopping criterion is

$$\|\delta \mathbf{x}\|_\infty < \epsilon_N h L_{\text{box}}. \quad (92)$$

Table 6. Simulation settings for the design, identification, and control tasks in Section 7. The columns report time step h , relative collision activation distance d_{rel} , barrier coefficient κ , friction-smoothing velocity threshold $\tilde{d}_f$, motion DoFs, relative Newton tolerance ϵ_N , friction coefficient, Young’s modulus E for volumetric FEM or cloth stretching, and affine stiffness κ_A . For Dual-arm grasp, 0.3 is the arm–cloth friction coefficient and 0.2 is the cloth–table coefficient.

Example	h	Horizon	d_{rel}	κ	$\tilde{d}_f$	#DOF	Newton ϵ_N	Friction	E	κ_A
Stone placement (Fig. 1)	0.03	1000	10^{-3}	10^5	10^{-3}	84	10^{-3}	0.3	–	10^7
Bunny collision (Fig. 6)	0.05	40	10^{-3}	10^{10}	–	5598	3×10^{-4}	0	4×10^7	–
Tentacle matching (Fig. 7)	0.005	100	10^{-3}	10^7	10^{-3}	20688	10^{-2}	1	2×10^4	–
Garment Hanger (Fig. 4)	1/120	600	10^{-3}	10^5	10^{-3}	51357	10^{-2}	0.05	5×10^4	10^8
Loaded Support (Fig. 8)	0.02	20	3×10^{-4}	10^6	–	22545	10^{-3}	0	10^8	–
Material fitting (Fig. 9)	0.1	10	10^{-3}	10^7	–	12567	10^{-2}	0	10^6	–
Friction optimization (Fig. 10)	0.02	100	10^{-3}	10^6	10^{-3}	2799	10^{-5}	0.5→0.1	10^5	–
Cloth rest scale (Fig. 12)	1/24	48	10^{-3}	10^5	–	102732	10^{-3}	0	10^5	10^8
Grasp angle (Fig. 13)	0.02	120	10^{-3}	10^5	10^{-3}	195	5×10^{-3}	0.1	5×10^5	10^7
Cup catching (Fig. 14)	0.005	240	10^{-3}	10^5	10^{-3}	192	10^{-3}	1.0	–	10^7
Dual-arm grasp (Fig. 15)	0.01	330	10^{-3}	10^5	10^{-3}	12504	10^{-3}	0.2/0.3	3×10^4	10^9

Table 7. Physical parameter families exposed to task-specific design variables through the interface in Section 6.3. Each family contributes local derivatives to the shared coupled adjoint.

Physical group	Differentiated quantities	Role
Tetrahedral material	μ, λ	Stable Neo-Hookean material response
Initial FEM motion	Selected initial vertex velocities	Initial dynamic state
Deformable rest shape	Selected rest coordinates	Volumetric and cloth rest geometry
Cloth material	Stretch, shear, and bending coefficients	Cloth constitutive response
Friction	Mesh self and ground friction coefficients	Frictional contact
Initial affine motion	$\mathbf{q}_{0,b}, \dot{\mathbf{q}}_{0,b}$	Initial affine state
Affine rest shape	Selected surface-rest coordinates	Mass, inertia, shape energy, collision, and friction
Joint controls	Joint targets and forces	Revolute or prismatic actuation on selected joints and frames
Affine position targets	Target positions	Position control

The shared FEM benchmarks use the same convergence tolerances to align numerical accuracy in the timing comparisons. The forward PCG relative residual tolerance is 10^{-4} .

The parameter families in Table 7 connect task-specific design variables to material response, geometry, initial motion, and actuation through the interface in Section 6.3. Different parameterizations and objectives share the same coupled backward computation. Each parameter family is differentiated in its associated motion representation. The FEM rest-shape configuration uses free FEM vertex positions. Affine rest-shape derivatives include the effects of coupled contact and active friction.