PHYSICS-BASED AND DATA-DRIVEN MODELING OF FLEXIBLE BODIES IN MULTIBODY DYNAMICS SIMULATIONS

by

Zhenhao Zhou

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Mechanical Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2026

Date of final oral examination: 04/28/2026

The dissertation is approved by the following members of the Final Oral Committee:

Dan Negrut, Professor, Mechanical Engineering; Affiliate Faculty, Computer Sciences
Xiangru Xu, Assistant Professor, Mechanical Engineering; Electrical & Computer Engineering by courtesy
Jinlong Wu, Assistant Professor, Mechanical Engineering
Wei Wang, Assistant Professor, Mechanical Engineering
Mike Zinn, Professor, Mechanical Engineering

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Dan Negrut, for his guidance, encouragement, and support throughout this work. I would also like to thank Dr. Radu Serban for his help and support during the course of this Ph.D. I am grateful to both of them for providing mentorship throughout my Ph.D.

I thank the members of my dissertation committee, Professor Dan Negrut, Professor Xiangru Xu, Professor Jinlong Wu, Professor Wei Wang, and Professor Mike Zinn, for their time, feedback, and suggestions that helped improve the quality of this dissertation.

I thank the National Science Foundation, Intuitive Machines, the National Aeronautics and Space Administration (NASA), Artfit 3D, and my advisor, Professor Dan Negrut, for providing Research Assistantships that supported me financially during the course of this Ph.D.

I thank Professor Jennifer Detlor, Professor Gregory Nellis, and Professor Antonio Hernandez, at the Mechanical Engineering Department at UW-Madison, for the Teaching Assistantship that funded the final year of this Ph.D.

I thank NVIDIA Corporation for covering the cost of VLM evaluation in the last part of this Ph.D. thesis, for recognizing and endorsing the research presented here, and for the employment opportunity upon graduation.

I am also grateful to my collaborators, colleagues, and the members of the Simulation-Based Engineering Laboratory for the many discussions, suggestions, and technical insights that helped shape this dissertation.

Finally, I would like to thank my family and friends for their patience, encouragement, and support during the course of this Ph.D.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

PHYSICS-BASED AND DATA-DRIVEN MODELING OF FLEXIBLE BODIES IN MULTIBODY DYNAMICS SIMULATIONS

Zhenhao Zhou

Under the supervision of Professor Dan Negrut

At the University of Wisconsin-Madison

The accurate simulation of flexible and deformable bodies is central to modern robotics, surgical tool design, tire–terrain interaction, and soft-robot locomotion. Classical rigid-body or small-strain models cannot capture the finite deformations, nonlinear constitutive response, and contact behavior that govern dynamics in these settings. The finite element method, and in particular the Total Lagrangian (TL) formulation and the Absolute Nodal Coordinate Formulation (ANCF), provides a principled continuum-mechanical framework for such problems. However, the computational cost of high-fidelity finite element analysis has historically restricted its use to offline design studies, preventing its adoption in real-time robotics training loops and interactive workflows.

The first part of this dissertation develops a GPU-accelerated Total Lagrangian finite element framework for finite-deformation multibody dynamics. A compact factored position field, $\mathbf{r} = \mathbf{N}(t)\mathbf{s}(\mathbf{u})$, separates time-dependent nodal unknowns from precomputed reference-configuration geometry, yielding a deformation gradient $\mathbf{F} = \mathbf{N}(t)\mathbf{H}(\mathbf{u})$ that unifies ten-node tetrahedral (T10), ANCF 3243 beam, and ANCF 3443 shell elements under a single notational and computational template. Constitutive models—St. Venant–Kirchhoff and Mooney–Rivlin, together with a finite-strain Kelvin–Voigt damping law—are expressed through the first Piola–Kirchhoff stress interface, so that adding a new material model requires only specifying the stress function and its derivative while the surrounding assembly machinery remains unchanged. Kinematic joints are constructed from four scalar constraint primitives (DP1, DP2, DIST, CD) and enforced via an Augmented Lagrangian Method, and a BVH-free GPU collision pipeline with two asynchronous threads handles frictional contact. Two complementary nonlinear solvers are provided: a first-order AdamW optimizer, whose second-

moment accumulation functions as an inexpensive diagonal preconditioner without Hessian assembly, and a second-order Newton solver that assembles the sparse augmented-Lagrangian Hessian through a two-stage GPU kernel—a write-independent stress evaluation stage followed by atomic-scatter force assembly—and factorizes it via cuDSS under a fixed-sparsity strategy that eliminates repeated symbolic analysis. Systematic benchmarks across three element types and six mesh resolutions demonstrate approximately one order of magnitude reduction in real-time factor relative to CPU baselines at the largest mesh sizes tested.

The second part addresses the reality gap through a data-driven approach to contact-rich world modeling. We curated DreamerBench, a 12-hour human-supervised dataset of contact-rich robot interactions generated in simulation across multiple friction regimes and object geometries, covering the three contact modes of no-contact, sticking, and sliding. A contact-encoding method projects frictional contact forces onto the camera image plane as depth-weighted Gaussian splats, encoding contact location, magnitude, and direction as an RGB image that can be processed by the same tokenization pipeline as ordinary video frames. ChronoDreamer, a spatial-temporal transformer with 24 layers trained via masked token prediction on this dataset, jointly predicts future video and contact splat frames conditioned on commanded actions and joint angle histories, without any physics supervision at training time. To evaluate contact-physics fidelity in generated rollouts—a capability not captured by standard pixel-level reconstruction losses—we introduce VLM-AUC, in which a vision-language model scores each predicted rollout against a structured rubric anchored to collision evidence, and the scores are aggregated via AUC against binary human contact labels.

Dan Negrut

1 INTRODUCTION

Physics-based simulation occupies a central role in modern robotics. Simulators such as MuJoCo [2], Bullet [3], Chrono [4], Isaac Gym [5], Warp [6], and Brax [7] are now routinely used for reinforcement learning, motion planning, system identification, and hardware-in-the-loop testing. Liu and Negrut survey the expanding role of physics engines in robotics and note that simulation fidelity—particularly in contact, friction, and deformable-body modeling—is a primary factor in determining whether policies and designs developed in simulation transfer reliably to physical hardware [8]. At the same time, the growing demand for higher-fidelity simulations involving flexible and deformable bodies has exposed computational and methodological limitations in existing tools. International assessments of simulation-based engineering stress both the centrality of numerical simulation and the substantial computational and process overheads it introduces at industry scale [9, 10].

Many robotics applications of practical interest—soft robot locomotion, surgical tool manipulation, cable routing, tire–terrain interaction, compliant grasping—require the accurate representation of flexible and deformable bodies undergoing large strains and displacements. Classical rigid-body or linearized small-strain models cannot capture the finite deformations, nonlinear constitutive response, and deformation-dependent contact behavior that govern the dynamics in these settings. The finite element method (FEM) provides a principled continuum-mechanical framework for such problems by discretizing each body into a mesh of elements and approximating the displacement field with polynomial shape functions [11, 12, 13]. Within the FEM literature, the *Total Lagrangian* (TL) formulation is well suited to large-deformation multibody dynamics: all kinematics and numerical integration are referenced to a fixed, undeformed configuration, which fixes shape-function gradients and geometric metrics in time, reduces assembly cost, and improves numerical stability [14]. The Absolute Nodal Coordinate Formulation (ANCF), introduced by Shabana [15], is a closely related TL approach that employs position vectors and their spatial gradients as nodal degrees of freedom, enabling a unified treatment of large rigid-body motion and elastic deformation

without separate rotational parameterizations. Nachbagauer provides a comprehensive review of ANCF element formulations and their locking behavior [16]. Together, TL-FEA and ANCF provide a versatile foundation for beam, shell, and solid element types relevant to flexible multibody dynamics.

This thesis develops a TL-FEA framework for finite-deformation multibody dynamics. The formulation establishes a compact deformation map $\mathbf{r} = \mathbf{N}(t)\,\mathbf{s}(\mathbf{u})$ and its associated deformation gradient $\mathbf{F} = \mathbf{N}(t)\,\mathbf{H}(\mathbf{u})$, which unify the treatment of isoparametric and non-isoparametric elements—ten-node tetrahedral (T10), ANCF 3243 beam, and ANCF 3443 shell—under a single notational and computational template. Hyperelastic constitutive models (St. Venant–Kirchhoff and Mooney–Rivlin) and a Kelvin–Voigt viscoelastic damping law are formulated in terms of the right Cauchy–Green tensor and the second Piola–Kirchhoff stress, covering material behaviors from rubber-like hyperelasticity to rate-dependent damping. Kinematic constraints for joints and prescribed motions are expressed through modular primitives (DP1, DP2, DIST, CD) and enforced via an Augmented Lagrangian Method (ALM) that regularizes constraint enforcement without collapsing to a pure-penalty formulation.

Contact detection for deformable finite element meshes on GPU hardware presents challenges that differ markedly from the rigid-body case. In large-scale multibody dynamics, GPU-based collision pipelines have traditionally relied on spherical decomposition—representing complex geometry as large collections of spheres whose pairwise overlap tests are trivially parallel [17, 4]. This strategy works well for convex and mildly non-convex rigid shapes, but it is poorly suited to the evolving surface geometry of deformable FEA meshes, where maintaining a consistent and accurate sphere packing across large deformations is impractical. Classical narrow-phase algorithms such as the Gilbert–Johnson–Keerthi (GJK) distance algorithm and the Minkowski Portal Refinement (MPR) method are the standard choice for convex–convex queries in CPU-based engines. However, both algorithms are inherently iterative and involve data-dependent branching—the simplex case selection in GJK, for instance, forces threads within the same GPU warp to take divergent code paths, degrading throughput. Montanari

et al. show that the Johnson sub-algorithm used inside GJK imposes significant register pressure and numerical fragility, particularly in single-precision GPU arithmetic [18], and Govender *et al.* adopt a separating-plane method in place of GJK for GPU-based DEM of convex polyhedra precisely because the separating-plane test is more amenable to data-parallel execution [19]. Parallel bounding-volume hierarchy construction [20], spatial hashing, and AABB-based broad-phase methods offer GPU-friendly alternatives at the broad-phase level [21], but the narrow-phase query for deformable tetrahedral elements remains an open design problem. The approach taken in this thesis sidesteps iterative geometric queries entirely: the broad-phase employs sweep-and-prune on per-element axis-aligned bounding boxes with an adjacency filter that suppresses topologically neighboring element pairs, and the narrow-phase constructs iso-pressure contact patches directly from the nodal pressure field interpolated within each tetrahedron. The resulting patch descriptors—area, centroid, normal, and directional pressure gradients—feed a Hertz–Mindlin force model that assembles nodal contact forces with dissipative and frictional contributions. A two-thread asynchronous design decouples collision detection from the dynamics integration loop, improving throughput without sacrificing contact responsiveness.

On the numerical side, the framework provides both first-order and second-order solvers for the nonlinear systems arising from the ALM formulation. The first-order path uses AdamW—an adaptive, per-coordinate gradient method with decoupled weight decay—as the inner solver for ALM iterations; its second-moment accumulation acts as an inexpensive diagonal preconditioner that helps tame mesh anisotropy and multi-scale stiffness variations. The second-order path assembles the sparse Hessian of the augmented Lagrangian and factors it via a GPU-accelerated direct solver, recovering quadratic local convergence for well-conditioned problems. Both solvers are implemented in CUDA, with internal force, gradient, and Hessian evaluations parallelized at the element and quadrature-point levels and sparse matrix assembly performed through atomic scatter-add operations. The resulting GPU-accelerated implementation achieves real-time or faster-than-real-time performance on

meshes with tens of thousands of elements, making high-fidelity deformable-body simulation practical within robotics training loops and interactive design workflows.

Despite these advances, fundamental limitations of physics-based simulation persist. High-fidelity finite element calculations remain computationally expensive, and the fidelity of any simulator is bounded by the accuracy of its input parameters—geometry, material constants, boundary conditions, and contact models—which are rarely known exactly for real-world objects. This *reality gap* can systematically bias policies trained in simulation. Muratore *et al.* review how incorrect or overly optimistic simulators can lead to policies that exploit simulator artifacts instead of learning transferable strategies [22]. In parallel, domain randomization methods deliberately inject parametric and visual perturbations into approximate simulators. Tobin *et al.* show that randomized, low-fidelity renderings can suffice to train object detectors that transfer robustly to real images [23], and Peng *et al.* and Chen *et al.* demonstrate that policies trained with aggressive dynamics randomization—using families of models that are individually less accurate than a carefully hand-tuned simulator—can exhibit better sim-to-real transfer because they learn strategies robust across a range of plausible dynamics [24, 25]. These results indicate that pushing fidelity alone is not sufficient; the statistical structure and diversity of simulated training data matter at least as much as detailed continuum mechanics. Furthermore, online scene reconstruction and meshing remain open problems in dynamic, cluttered environments [26], adding a practical barrier to purely physics-based workflows.

A growing body of work attacks these barriers through *real-to-sim* pipelines that reconstruct simulation-ready scenes directly from sensor data. Ehsani *et al.* show that physical forces applied during human–object interaction can be inferred from video by using a physics simulator as supervision: the estimated forces must reproduce the observed effects, closing the loop between perception and simulation without ground-truth force labels [27]. Jatavallabhula *et al.* introduce gradSim, which couples differentiable multiphysics simulation with differentiable rendering so that physical attributes—mass, friction, stiffness—can be identified by backpropagating through the full render-simulate pipeline from pixel-level losses [28].

More recently, neural scene representations have been integrated with physics engines to build interactive digital twins on the fly. Byravan *et al.* reconstruct a scene's contact geometry and appearance via a Neural Radiance Field (NeRF) from a short phone video, then train vision-guided locomotion and manipulation policies entirely in the resulting simulation, achieving zero-shot sim-to-real transfer on a bipedal robot [29]. Abou-Chakra *et al.* propose a dual Gaussian-particle representation in which 3D Gaussian splats handle rendering while a coupled particle system handles physics, producing a real-time correctable world model that can predict future states conditioned on robot actions and be corrected online from visual observations [30]. Torne *et al.* present RialTo, a full real-to-sim-to-real system that scans a workspace, constructs a digital twin, and fine-tunes manipulation policies via reinforcement learning in the reconstructed environment, reporting substantial improvements in policy robustness over direct sim-to-real transfer [31]. Taken together, these results suggest that on-the-fly scene reconstruction coupled with fast predictive models can allow a robot to build a simulation of its current environment, anticipate the consequences of candidate actions, and act accordingly—precisely the capability that a contact-aware world model, as developed in the second part of this thesis, is designed to provide.

Recent work on learned simulators provides a complementary path forward. Interaction Networks [32] and Graph Network-based Simulators [33] demonstrate that message-passing neural networks can learn to approximate the dynamics of fluids, rigid bodies, and deformable materials directly from rollout data. MeshGraphNets extend this idea to mesh-based discretizations, emulating finite element and finite volume solvers for structural mechanics and fluid flow while achieving one to two orders of magnitude speedup at test time [34]. World models take the idea a step further by learning compact latent state spaces in which both dynamics and observations are modeled jointly. Early work by Ha and Schmidhuber showed that a generative recurrent model trained on pixels and actions can support policy learning entirely in imagination, with the resulting policies deployed back to the original environment [35, 36]. The Dreamer family of algorithms learns stochastic latent world models

from image streams and optimizes policies by planning in the latent space, achieving strong sample efficiency on a wide range of visual control benchmarks [37, 38]. Wu *et al.* demonstrate with DayDreamer that similar ideas scale to physical robots: a single world-model-based agent can learn locomotion, manipulation, and navigation skills directly from cameras and sparse rewards with modest real-world interaction budgets [39]. World models are thus emerging as a practical tool for robotics, not only as a conceptual framework.

In this thesis, we adopt the view that next-generation robotics simulators will combine high-fidelity continuum mechanics with learned world models built from data. The first part develops the TL-FEA framework described above: Chapter 2 reviews background and motivation; Chapter 3 presents the finite element formulation and constitutive modeling; Chapter 4 details the frictional contact pipeline; Chapter 5 describes the augmented Lagrangian solvers and GPU-accelerated implementation; and Chapter 6 reports benchmark and validation experiments. The second part turns to data-driven world models: Chapter 7 introduces a contact-aware world model architecture that combines depth-weighted Gaussian contact splats with a spatial-temporal transformer trained via masked token prediction, together with a large-language-model-based collision judge for online action screening. Chapter 8 summarizes contributions and outlines directions for future work. The overall aim is not to replace physics engines with neural networks, but to use world models as a statistical layer that compensates for the limits of pure simulation while preserving the structure and insight that continuum mechanics provides.

2 BACKGROUND AND MOTIVATION

2.1 Background

2.1.1 Accurate Simulation for Flexible and Deformable Bodies

The Total Lagrangian (TL) formulation is the standard setting for geometrically nonlinear solid mechanics [13, 40]: all kinematic and stress quantities are referred to the fixed reference configuration, shape function gradients are precomputed once, and the deformation gradient $\mathbf{F}$ serves as the natural interface between kinematics and constitutive response. Peng et al. benchmark TL against the Updated Lagrangian and corotational alternatives and find that TL consistently provides the cleanest separation between reference geometry and deformation-dependent response in large-deformation regimes [41]. Within flexible multibody dynamics (FMBD), the Absolute Nodal Coordinate Formulation (ANCF) [42, 43] is a particular instance of this TL perspective: it uses absolute positions and their spatial gradients as nodal degrees of freedom, yields a constant mass matrix, and employs the same $\mathbf{F}$ as standard continuum finite elements, making it compatible with any TL constitutive model. A variant by Zhang [44] eliminates $\mathbf{F}$ entirely in an explicit single-body setting; retaining $\mathbf{F}$ as the central kinematic quantity is necessary for implicit constrained FMBD, where both the tangent stiffness and the constraint Jacobians derive from it.

A recurring practical issue in TL-FEA concerns the representation of the interpolation. The majority of the FEM and ANCF literature writes the position field as $\mathbf{r} = \mathbf{S}(\mathbf{u})\,\mathbf{e}(t)$, where $\mathbf{S}$ collects shape functions and $\mathbf{e}$ the nodal unknowns [13, 45]. An equivalent transposed form, $\mathbf{r} = \mathbf{N}(t)\,\mathbf{s}(\mathbf{u})$, stores the nodal unknowns in $\mathbf{N}$ and the scalar shape functions in $\mathbf{s}$, yielding the deformation gradient $\mathbf{F} = \mathbf{N}(t)\,\mathbf{H}(\mathbf{u})$. Earlier contributions in this direction include the edge-matrix form for linear tetrahedra [46] and the component-free Lagrangian formulation of Stickle and Pastor [47]. Because $\mathbf{F}$, the internal force gradient, and the position first-variation all factor through the same $\mathbf{N}$–$\mathbf{H}$ split, constraint derivatives across all

element types follow from a single chain-rule pattern applied to the point-evaluation operator, eliminating element-specific case analysis when deriving Jacobians.

On the constitutive side, a central challenge for implicit ANCF solvers is the availability of *closed-form* consistent tangents. García-Vallejo et al. derived analytic elastic-force Jacobians for the St. Venant–Kirchhoff model in the block shape-function notation [48]. Subsequent ANCF work on nonlinear models—Mooney–Rivlin rubber and Poisson-locking-resistant materials—fell back on numerical tangent approximations [49, 50], which are acceptable for moderate Newton iterations but degrade convergence rate and introduce step-size sensitivity. Providing closed-form tangents through a unified first Piola–Kirchhoff interface $\mathbf{P}(\mathbf{F})$ and its material derivative makes any constitutive model immediately available to any element topology without element-specific derivations.

The treatment of kinematic constraints is a central challenge when extending FMBD to deformable bodies. In rigid-body dynamics, primitive constraints such as DP1, DP2, CD, and DIST [51] provide a standard building block for engineering joints. Betsch and Steinmann [52] developed rotationless constrained dynamics for geometrically exact beams, providing an early deformable-body treatment, but their formulation is specific to that element class and does not address Jacobian accumulation or the conditioning issues that arise when dot-product and coordinate-difference constraints are mixed. Sugiyama et al. [53] formulated joint constraints for ANCF using position-gradient degrees of freedom as orientation surrogates, giving a foundation for deformable-body joints; however, element-level Jacobian accumulation rules, the curvature contribution to the Newton system, and the row-scaling analysis required for a well-conditioned mixed constraint set were not addressed.

Efficient GPU implementation of implicit FMBD requires solutions to two infrastructure problems: fast element-level parallelism and a GPU-resident sparse direct solver. Early GPU-accelerated FEM work focused on explicit dynamics and linear problems [54], where per-element independence makes data-parallel execution straightforward. Position-based dynamics solvers such as NVIDIA FleX [55] achieve real-time rates on large cloth and fluid

systems by relaxing constitutive accuracy; physics engines built on spatially sparse data structures such as Taichi [56] extend GPU utilization broadly but are generally limited to explicit or penalty-based contact. General-purpose frameworks such as SOFA [57] have demonstrated GPU acceleration of specific kernels while retaining CPU-side sparse solvers for the global system. Assembling and factorizing a sparse global Newton Hessian at each iteration inside an implicit second-order solver requires a GPU-resident direct sparse solver; the release of NVIDIA cuDSS [58] makes this pathway viable at scale.

For implicit integrators, Gear and Rix [59] established backward-Euler and generalized-α methods as the standard for stiff flexible body systems, and Brüls, Cardona, and Arnold extended these integrators to the ANCF setting [60]. Extending implicit integration to constrained systems introduces a coupled saddle-point system at each step. Baumgarte stabilization [61] is the most widely used remedy for constraint drift but requires careful tuning of stabilization parameters. Augmented Lagrangian methods (ALM) [62, 63] avoid this sensitivity: the penalty parameter is an outer-loop tuning knob decoupled from the inner velocity solve, and the two-level structure is particularly attractive when the inner solve is itself iterative or GPU-resident.

Two established open-source frameworks serve as CPU baselines against which GPU-accelerated TL-FEA performance is evaluated. The FEniCS Project [64] is a widely used automated finite element platform that solves variational problems expressed in the Unified Form Language (UFL). It has been cross-validated against Abaqus on hyperelastic large-deformation problems with sub-percent agreement [65], and has been adopted as a CPU reference by other GPU-accelerated solvers [66]. FEniCS does not support ANCF elements, bilateral kinematic constraints, or deformable-body contact, making it appropriate as a reference only for solid-element (T10) benchmarks. Project Chrono [4] is a multi-physics multibody dynamics engine whose FEA module provides mature ANCF beam and shell elements [67, 68] and CPU-side direct sparse solvers. Taylor showed that careful implementation can accelerate ANCF internal-force and Jacobian evaluations by one to two orders

of magnitude on the CPU [69, 70], establishing a strong CPU-side upper bound for ANCF beam and shell performance comparisons.

2.1.2 Towards Next Generation Robotics Simulators: World Models

Despite decades of progress, high-fidelity physics simulation remains difficult to deploy as a routine engineering tool in robotics. Large surveys on simulation-based engineering stress both the centrality of numerical simulation and the substantial computational and process overheads it introduces for industry-scale workflows [9]. Recent industrial surveys by NAFEMS and McKinsey report that organizations struggle not only with solver performance and software integration, but also with data management and verification processes when simulation is adopted at scale [10]. At the same time, NAFEMS-led projects such as EASIT2 and related industry-needs surveys identify recruitment of staff with strong analysis and simulation skills—and the time needed for their training—as major bottlenecks for effective use of advanced finite-element and multi-physics tools [71, 72, 73]. These findings suggest a structural skills and capacity limitation around classical simulation technology, even before one considers the additional complexity of modeling non-smooth contact, friction, and flexible bodies for robotics.

From the sim-to-real perspective, higher-fidelity physics is also not a universal remedy. A large body of work documents the *reality gap*: small errors in contact modeling, friction, sensing, or actuation can systematically bias policies that are trained purely in simulation. Muratore *et al.* review how incorrect or overly optimistic simulators can lead to policies that exploit simulator artifacts instead of learning strategies that transfer [22]. In parallel, domain randomization methods deliberately use approximate, sometimes visually crude, simulators and inject large parametric and visual perturbations. Tobin *et al.* show that such randomized, low-fidelity renderings can be enough to train object detectors that transfer robustly to real images [23]. Peng *et al.* and Chen *et al.* demonstrate that policies trained with aggressive

dynamics randomization—using families of models that are individually less accurate than a carefully hand-tuned simulator—can exhibit *better* sim-to-real performance because they learn strategies that are robust across a range of plausible dynamics [24, 25]. These results indicate that pushing fidelity alone is not sufficient; the statistical structure of the training data and the diversity of simulated worlds matter at least as much as detailed continuum mechanics.

In parallel, online scene reconstruction and meshing remain challenging even when powerful SLAM and depth-sensing systems are available. The SLAM survey by Cadena *et al.* emphasises that long-term, robust mapping in dynamic, cluttered environments is still an open problem [26], and multiple systematic reviews of real-time 3D reconstruction for telepresence or dynamic scenes report persistent issues with drift, latency, and reconstruction quality in realistic setups [74, 75]. On the geometric processing side, surface reconstruction from point clouds is known to be technically ill-posed and sensitive to sampling density, noise, and sensor artifacts. Comprehensive surveys by Berger *et al.* and Huang *et al.* show that even state-of-the-art methods can produce holes, spurious surfaces, and oversmoothed regions in the presence of misaligned scans, missing data, or outliers [76, 77]. This means that generating clean meshes for contact-rich simulation in real time is itself a nontrivial estimation problem, not a solved preprocessing step.

Recent work on learned simulators provides one way to bridge these gaps. Interaction Networks [32] and Graph Network-based Simulators [33] show that message-passing neural networks can learn to approximate complex dynamics of fluids, rigid bodies, and deformable materials directly from rollout data. MeshGraphNets extend this idea to mesh-based discretisations, learning to emulate finite volume and finite element solvers for structural mechanics and fluid flow while achieving one to two orders of magnitude speedup at test time [34]. These models effectively learn surrogate dynamics on top of classical discretisations: the mesh and boundary conditions still encode geometry and topology, but local update rules are represented by a neural network trained from data rather than a hand-coded constitutive law

plus a fixed time integrator.

World models take this idea a step further by learning compact latent state spaces in which both dynamics and observations are modeled jointly. Early work by Ha and Schmidhuber showed that a generative recurrent model trained on pixels and actions can support policy learning entirely in imagination, with the resulting policies deployed back to the original environment [35, 36]. The Dreamer family of algorithms learns stochastic latent world models from image streams and optimizes policies by planning in the latent space, achieving strong sample efficiency on a wide range of visual control benchmarks [37, 38]. DayDreamer demonstrates that similar ideas scale to physical robots: a single world-model-based agent can learn locomotion, manipulation, and navigation skills directly from cameras and sparse rewards with modest real-world interaction budgets [39]. World models are thus emerging as a practical tool for robotics, not only as a conceptual framework.

In this thesis, we adopt the view that next-generation robotics simulators will combine high-fidelity continuum mechanics and contact models with learned world models built from data. GPU-accelerated finite element solvers can generate rich offline datasets, probe rare contact events, and provide physically consistent supervision for stresses, strains, and deformations under controlled conditions. Learned world models can then absorb this offline computation, together with logs from real robots operating in imperfectly modeled environments, to learn latent dynamics that are fast to query online and robust to model mismatch, unmodeled contacts, and reconstruction artifacts. The overall aim is not to replace physics engines with neural networks, but to use world models as a statistical layer that compensates for the limits of pure simulation while preserving the structure and insight that continuum mechanics provides.

3 FINITE ELEMENT FORMULATION AND CONSTITUTIVE MODELING ASPECTS

3.1 The TL Deformation Map

Large-deformation multibody dynamics can be posed within a continuum mechanics setting by describing each body's deformation, which in turn is defined by the state of deformation associated with each of the body's elements. Focusing on one element, there is a mapping $\mathbf{r} = \boldsymbol{\phi}(\mathbf{u}; t) \in \mathbb{R}^3$ and its associated deformation gradient $\mathbf{F}(\mathbf{u}; t) = \partial \mathbf{r}/\partial \mathbf{u}$ that indicates where each point $\mathbf{u} = [u, v, w]^T \in \mathbb{R}^3$ in an element's reference configuration is mapped to in the current configuration at time t.

The reference configuration of an element is that in which the element experiences no strain and no stress. Typically, this is produced by a mesher, for example `Gmsh` [78] or `TetGen` [79]. Associated with this reference configuration of the element there is a reference frame $Ouvw$, whose position and orientation are fixed in space, as is the element in the reference configuration. A point P in the reference configuration of coordinates (u, v, w) is mapped at time t into a different point of coordinates $\mathbf{r}(u, v, w; t) = [x(u, v, w; t), y(u, v, w; t), z(u, v, w; t)]^T$ in a global and fixed reference frame $Oxyz$. This captures the essence of TL-FEA: one always reports the current configuration in relation to a fixed and inertial reference frame.

Moving beyond the reference and current configurations, the initial configuration is the configuration of the element at time $t = 0$. In many cases, at time $t = 0$ the element is in a state of stress, which can be solved for. Finally, we consider a fourth configuration, the parent (or canonical) configuration, which is used to carry out the numerical integration required to compute the nodal forces induced by internal strain, surface tractions, and other volume-distributed forces. The elements can be isoparametric, or, for simple cases, the mapping between the parent and the reference configuration can be an affine transformation.

The most common approach in use in the TL-FEA field is to consider as unknowns the

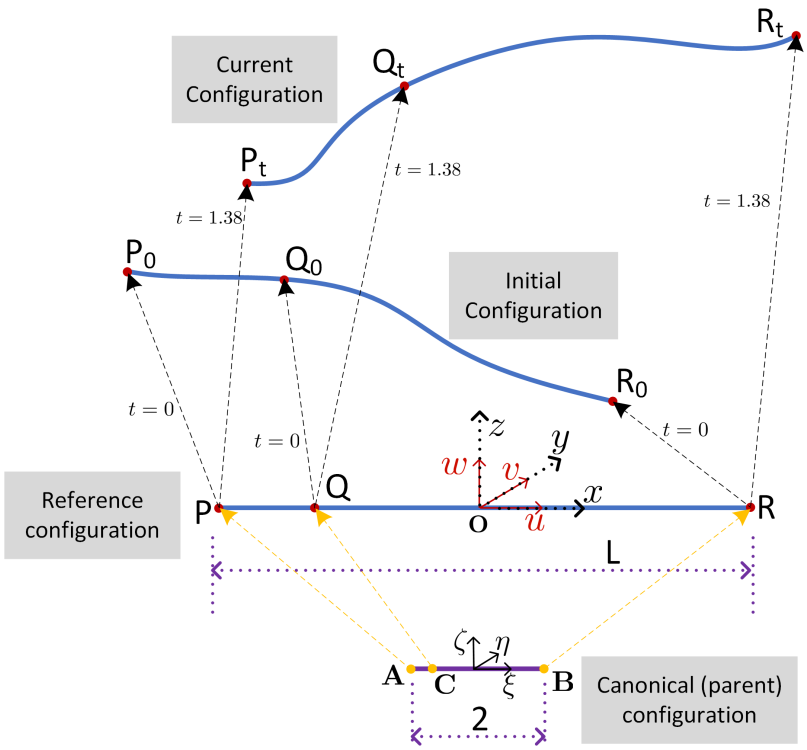


Figure 3.1: Four key element configurations in TL-FEA. When discussing the parent, reference, initial, and current configurations, the discussion pertains to *one element*. The union of these elements produces a body, and the body's state is simply captured by its elements' states. The parent and reference configurations are abstractions; the initial and current configurations are physical configurations of the element at time $t = 0$ and, as shown in this figure, at time $t = 1.38$, respectively.

locations of the nodes of the element, that is, the positions $\mathbf{e}_i(t) \in \mathbb{R}^3$ of the nodes in the current configuration at time t relative to the global reference frame $Oxyz$. Thus, for a ten-node nonlinear tetrahedral element T10 [11], the nodal unknowns are $\mathbf{e}_1(t), \ldots, \mathbf{e}_{10}(t)$. Within the flexible multibody dynamics community, the Absolute Nodal Coordinate Formulation (ANCF) [15] is another example of a TL-FEA formulation. The defining feature of ANCF is the choice of nodal coordinates: it employs position vectors and spatial gradients as nodal degrees of freedom. A large body of work has since expanded ANCF element technology, from early beam and cable elements capturing geometrically nonlinear coupling without rotation interpolation [80], to shear-deformable and higher-order elements together with formulation-

level remedies for locking pathologies [81]. Comprehensive surveys [82, 83, 84] highlight a recurring theme: although ANCF elements come in many variants (beams, plates/shells, solid-like continua), their kinematics can be consistently viewed as a continuum mapping whose primary computational artifact is the deformation gradient and its derived strain measures. This is the hallmark of the Total Lagrangian formulation, in which all strain measures, stress measures, and constitutive responses are expressed with respect to the reference configuration of the element.

In the standard TL-FEA literature, which includes the ANCF literature that followed, the unknown map $\boldsymbol{\phi}$ has been approximated through a polynomial interpolation of the nodal unknowns $\mathbf{e}_i(t)$, which with a slight abuse of notation can be written as

$$\mathbf{r}(\mathbf{u}; t) = \mathbf{S}(\mathbf{u})\, \mathbf{e}(t), \tag{3.1}$$

where $\mathbf{S}(\mathbf{u}) \in \mathbb{R}^{3\times 3n}$ is a block-structured shape-function matrix and $\mathbf{e}(t) \in \mathbb{R}^{3n}$ stacks nodal positions and slopes into a generalized coordinate vector of dimension $3n$ [43, 84]. For instance, for the T10 nonlinear tetrahedral element, $n = 10$, and $\mathbf{S}$ and $\mathbf{e}(t)$ are of size 3×30 and 30, respectively, with $\mathbf{S}$ being a sparse matrix. Likewise, for a two-node, fully parameterized beam element, $n = 8$, and $\mathbf{S}$ and $\mathbf{e}(t)$ are of size 3×24 and 24, respectively.

The present overview concerns the TL-FEA formulation in multibody dynamics, of which the ANCF elements are a special case. Rather than embracing the classical TL-FEA notation, the reference-to-current map at time t for point $\mathbf{u}$ is expressed as

$$\mathbf{r}(\mathbf{u}; t) = \mathbf{N}(t)\, \mathbf{s}(\mathbf{u}), \tag{3.2}$$

where $\mathbf{N}(t) = [\mathbf{e}_1(t), \ldots, \mathbf{e}_n(t)] \in \mathbb{R}^{3\times n}$ is the *matrix of nodal unknowns* and $\mathbf{s}(\mathbf{u}) \in \mathbb{R}^n$ is the *vector of shape functions*. The shape functions are functions, most often polynomials, of the reference coordinates $\mathbf{u}$, and the nodal unknowns in classical TL-FEA are the nodal positions in the current configuration, while in ANCF they are a combination of position and position

gradients, potentially of a higher order. Compared to the reference-to-current map described in Eq. (3.1), which uses a sparse matrix of size $3 \times 3n$ and a vector of size $3n$, the one in Eq. (3.2) uses a dense matrix of size $3 \times n$ and a vector of size n. The immediate benefits of using Eq. (3.2) are as follows: the storage of the data is simpler, it leads to simpler linear algebra operations, and it has an intuitive interpretation in terms of nodal unknowns and shape functions. Indeed, the current location of a point in the reference configuration is a linear combination of the nodal positions with the shape functions acting as the weights in this linear combination of nodal positions, or positions and gradients for ANCF elements. It turns out that capturing the deformation mapping as in Eq. (3.2), rather than in the classical TL-FEA notation of Eq. (3.1), leads to a tighter notation involving dense matrix-vector operations and is amenable to expressing the Jacobians that come into play in constraint reaction forces, the formulation of the first and second Piola–Kirchhoff stress tensors, and the virtual work associated with a deformable multibody system.

In light of Eq. (3.2), the velocity, acceleration, and virtual displacement of the point to which $\mathbf{u}$ in the reference configuration is mapped in the current configuration at time t are given by

$$\dot{\mathbf{r}}(u, v, w; t) = \dot{\mathbf{N}}(t)\, \mathbf{s}(u, v, w), \qquad \ddot{\mathbf{r}}(u, v, w; t) = \ddot{\mathbf{N}}(t)\, \mathbf{s}(u, v, w), \tag{3.3a}$$

and

$$\delta \mathbf{r}^T = \mathbf{s}^T(u, v, w)\, (\delta \mathbf{N})^T = \mathbf{s}^T(u, v, w)\ \mathrm{col}\Big(\delta \mathbf{e}_1^T, \delta \mathbf{e}_2^T, \ldots, \delta \mathbf{e}_n^T\Big)\,. \tag{3.3b}$$

3.2 Local Deformation Metrics

Let $s_i(u, v, w)$ be the ith entry of the vector $\mathbf{s}(u, v, w)$, and define

$$\mathbf{h}_i(u, v, w) \equiv \nabla_{\mathbf{u}}\, s_i(u, v, w) \in \mathbb{R}^3. \tag{3.4a}$$

By the same token, define

$$\mathbf{H}(u,v,w) = \mathrm{col}\Big(\mathbf{h}_1^T(u,v,w), \mathbf{h}_2^T(u,v,w), \ldots, \mathbf{h}_n^T(u,v,w)\Big) = \frac{\partial \mathbf{s}(u,v,w)}{\partial \mathbf{u}} \in \mathbb{R}^{n\times 3}. \tag{3.4b}$$

Then, the deformation gradient $\mathbf{F} \in \mathbb{R}^{3\times 3}$ assumes the form

$$\mathbf{F} = \frac{\partial \mathbf{r}}{\partial \mathbf{u}} = \mathbf{N}(t)\,\mathbf{H}(u,v,w) = \sum_{i=1}^{n} \mathbf{e}_i(t)\,\mathbf{h}_i^T(u,v,w), \tag{3.5}$$

where the last sum combines the outer products of the basis vectors $\mathbf{e}_i$ with the gradients of the shape functions $\mathbf{h}_i^T$.

The right Cauchy–Green tensor is defined as

$$\mathbf{C} \coloneqq \mathbf{F}^T\mathbf{F}. \tag{3.6a}$$

It is a symmetric, positive-definite tensor that provides a measure of how squared material lengths change under the deformation:

$$\|d\mathbf{r}\|^2 = d\mathbf{u}^T\,\mathbf{C}\,d\mathbf{u}. \tag{3.6b}$$

$\mathbf{C}$ measures stretch and is invariant to rigid rotations. Strictly speaking, it is not a strain measure. Rather, $\mathbf{C}$ encodes the total effect of the deformation on material distances, independent of the current spatial frame. It is often referred to simply as the deformation tensor, more precisely the right Cauchy–Green deformation tensor. This is distinct from the deformation gradient $\mathbf{F}$, which serves a different role.

The Green–Lagrange strain tensor is defined as the symmetric tensor

$$\mathbf{E} \coloneqq \frac{1}{2}(\mathbf{C} - \mathbf{I}) = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I}). \tag{3.7a}$$

It quantifies the relative change in squared length of a material fiber:

$$\|d\mathbf{r}\|^2 - \|d\mathbf{u}\|^2 = 2\, d\mathbf{u}^T\, \mathbf{E}\, d\mathbf{u}. \tag{3.7b}$$

Unlike $\mathbf{F}$ and $\mathbf{C}$, $\mathbf{E}$ is a true strain measure: it vanishes when the body is undeformed, is symmetric, and eliminates the effects of rigid body rotation.

In summary,

- $\mathbf{F}$ tells us how a differential material fiber moves, including both how it stretches or compresses and how it rotates.
- $\mathbf{C}$ tells us how the squared length of a fiber changes under the motion.
- $\mathbf{E}$ tells us the relative change in squared length of the fiber, that is, the strain.

Thus,

$$\begin{aligned} &\mathbf{F}: \quad \text{maps } d\mathbf{u} \mapsto d\mathbf{r}, \text{ capturing both stretching and rotation through } d\mathbf{r} = \mathbf{F}\, d\mathbf{u}, \\ &\mathbf{C}: \quad \|d\mathbf{r}\|^2 = d\mathbf{u}^T\, \mathbf{C}\, d\mathbf{u}, \\ &\mathbf{E}: \quad \|d\mathbf{r}\|^2 - \|d\mathbf{u}\|^2 = 2\, d\mathbf{u}^T\, \mathbf{E}\, d\mathbf{u}. \end{aligned} \tag{3.8}$$

One important aspect that depends crucially on the deformation gradient is tied to how an infinitesimal surface element in the reference configuration maps to a surface element in the current configuration. This is given by Nanson's formula.

Let $\mathbf{a}_1^0, \mathbf{a}_2^0 \in \mathbb{R}^3$ be two infinitesimal fibers lying on a surface in the reference configuration. In this discussion, a superscript or subscript "0" refers to the reference configuration, and not to the initial time $t = 0$. These fibers span a planar patch, an area element, with area and unit normal

$$dA_0 = \left\|\mathbf{a}_1^0 \times \mathbf{a}_2^0\right\|, \qquad \mathbf{n}_0 = \frac{\mathbf{a}_1^0 \times \mathbf{a}_2^0}{\left\|\mathbf{a}_1^0 \times \mathbf{a}_2^0\right\|}.$$

Under a deformation with gradient $\mathbf{F}$, the fibers map to

$$\mathbf{a}_1 = \mathbf{F}\,\mathbf{a}_1^0, \qquad \mathbf{a}_2 = \mathbf{F}\,\mathbf{a}_2^0,$$

spanning the corresponding area element in the current configuration, with

$$dA = \left\|\mathbf{a}_1 \times \mathbf{a}_2\right\|, \qquad \mathbf{n} = \frac{\mathbf{a}_1 \times \mathbf{a}_2}{\left\|\mathbf{a}_1 \times \mathbf{a}_2\right\|}.$$

The deformed unit normal $\mathbf{n}$ is generally not aligned with $\mathbf{n}_0$, except in special cases, one of which is a pure rigid motion without stretch or shear.

Nanson's formula captures the transformation of a normal-area element from the reference to the current configuration:

$$dA\,\mathbf{n} = J\,\mathbf{F}^{-T}\,\mathbf{n}_0\,dA_0, \tag{3.9}$$

which captures both the change of area and the rotation or stretching of the normal. Finally, we note the following:

- The deformed area element remains locally planar since it is spanned by two deformed fibers $\mathbf{a}_1$ and $\mathbf{a}_2$. Typically, this plane is not the same as the one in the reference configuration.

- The shape of the area element in the current configuration need not remain rectangular even if it was initially rectangular in the reference configuration.

- The normal $\mathbf{n}$ is the unit normal in the current configuration and it is not a stretched or compressed version of $\mathbf{n}_0$. What transforms is the normal-area $\mathbf{n}_0\,dA_0$ through Nanson's formula in Eq. (3.9), which describes how the normal-area is reoriented and rescaled.

- In light of Eq. (3.9), the current unit normal $\mathbf{n}$ is obtained by normalizing $\mathbf{F}^{-T}\mathbf{n}_0$:

$$\mathbf{n} = \frac{\mathbf{F}^{-T}\mathbf{n}_0}{\|\mathbf{F}^{-T}\mathbf{n}_0\|},$$

and the area element in the current configuration is given by

$$dA = J \left\| \mathbf{F}^{-T} \mathbf{n}_0 \right\| dA_0.$$

3.3 Deformable Element Formulations

3.3.1 ANCF 3243 Beam Element

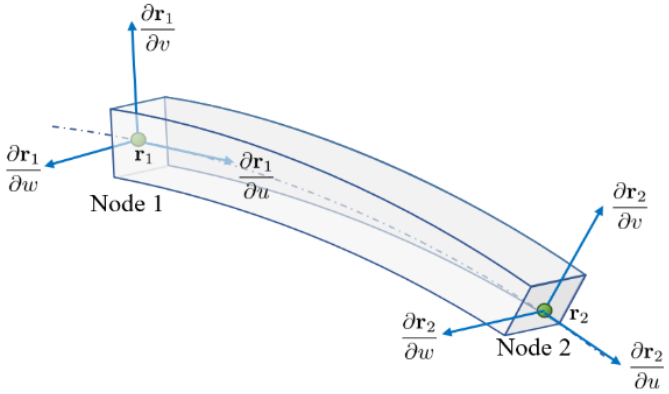


Figure 3.2: Visualization of a fully-parameterized 2-node ANCF beam element 3243.

The most basic element considered in the comparison is the original 2-node ANCF beam element 3243, which is fully parameterized, as referenced in [85]. As illustrated in Figure 2.1, each node is described by 12 coordinates, including a position vector $\mathbf{r}_i$ and its gradients with respect to the local coordinates u, v, and w, denoted as $\frac{\partial \mathbf{r}_i}{\partial u}$, $\frac{\partial \mathbf{r}_i}{\partial v}$, and $\frac{\partial \mathbf{r}_i}{\partial w}$. This full set of coordinates per node defines the "fully parameterized" terminology. With two such nodes, the element possesses 24 degrees of freedom when unconstrained. The visualization of this element can be seen in Figure 3.2.

The vector of basis functions for the 3243 beam element is:

$$\mathbf{b}^T(u, v, w) = [1, u, v, w, uv, uw, u^2, u^3]$$

The constant matrix $\mathbf{B}_{12} \in \mathbb{R}^{8\times 8}$ can be defined as:

$$\mathbf{B}_{12} \equiv [\mathbf{b}(P_1),\ \mathbf{b}_{,u}(P_1),\ \mathbf{b}_{,v}(P_1),\ \mathbf{b}_{,w}(P_1),\ \mathbf{b}(P_2),\ \mathbf{b}_{,u}(P_2),\ \mathbf{b}_{,v}(P_2),\ \mathbf{b}_{,w}(P_2)]\ ,$$

Therefore, the time-dependent global nodal coordinates, as well as the gradients components, x_{12}, y_{12}, and z_{12},

$$\begin{aligned}
\mathbf{x}_{12}^{\top} &\equiv [\, x_1,\ x_{1,u},\ x_{1,v},\ x_{1,w},\ x_2,\ x_{2,u},\ x_{2,v},\ x_{2,w}\,], \\
\mathbf{y}_{12}^{\top} &\equiv [\, y_1,\ y_{1,u},\ y_{1,v},\ y_{1,w},\ y_2,\ y_{2,u},\ y_{2,v},\ y_{2,w}\,], \\
\mathbf{z}_{12}^{\top} &\equiv [\, z_1,\ z_{1,u},\ z_{1,v},\ z_{1,w},\ z_2,\ z_{2,u},\ z_{2,v},\ z_{2,w}\,]\ .
\end{aligned}$$

as well as the unknown coefficients α, β, and γ:

$$\begin{aligned}
\mathbf{B}_{12}^{\top}\cdot\boldsymbol{\alpha} = \mathbf{x}_{12} \qquad &\Rightarrow \qquad \boldsymbol{\alpha} = \mathbf{B}_{12}^{-\top}\cdot\mathbf{x}_{12}\ , \\
\mathbf{B}_{12}^{\top}\cdot\boldsymbol{\beta} = \mathbf{y}_{12} \qquad &\Rightarrow \qquad \boldsymbol{\beta} = \mathbf{B}_{12}^{-\top}\cdot\mathbf{y}_{12}\ , \\
\mathbf{B}_{12}^{\top}\cdot\boldsymbol{\gamma} = \mathbf{z}_{12} \qquad &\Rightarrow \qquad \boldsymbol{\gamma} = \mathbf{B}_{12}^{-\top}\cdot\mathbf{z}_{12}\ .
\end{aligned}$$

which leads to:

$$[\boldsymbol{\alpha}\ \ \boldsymbol{\beta}\ \ \boldsymbol{\gamma}] = \mathbf{B}_{12}^{-\top}[\mathbf{x}_{12}\ \ \mathbf{y}_{12}\ \ \mathbf{z}_{12}]\ .$$

,

$$\mathrm{col}\Big(\boldsymbol{\alpha}^{\top},\, \boldsymbol{\beta}^{\top},\, \boldsymbol{\gamma}^{\top}\Big) = \mathrm{col}\Big(\mathbf{x}_{12}^{\top},\, \mathbf{y}_{12}^{\top},\, \mathbf{z}_{12}^{\top}\Big)\, \mathbf{B}_{12}^{-1}\ ,$$

which means that:

$$\mathbf{r}(u,v,w,t) = \mathrm{col}\Big(\mathbf{x}_{12}^{\top},\, \mathbf{y}_{12}^{\top},\, \mathbf{z}_{12}^{\top}\Big)\, \mathbf{B}_{12}^{-1}\cdot\mathbf{b}(u,v,w) = \mathbf{N}(t)\cdot\mathbf{s}(u,v,w)\ ,$$

where $\mathbf{N}(t) \in \mathbb{R}^{3\times 8}$ and $\mathbf{s}(u,v,w) \in \mathbb{R}^{8\times 1}$ are defined as:

$$\mathbf{N}(t) \equiv \begin{bmatrix} x_1 & x_{1,u} & x_{1,v} & x_{1,w} & x_2 & x_{2,u} & x_{2,v} & x_{2,w} \\ y_1 & y_{1,u} & y_{1,v} & y_{1,w} & y_2 & y_{2,u} & y_{2,v} & y_{2,w} \\ z_1 & z_{1,u} & z_{1,v} & z_{1,w} & z_2 & z_{2,u} & z_{2,v} & z_{2,w} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_{1,u} & \mathbf{r}_{1,v} & \mathbf{r}_{1,w} & \mathbf{r}_2 & \mathbf{r}_{2,u} & \mathbf{r}_{2,v} & \mathbf{r}_{2,w} \end{bmatrix} \equiv \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 & \mathbf{e}_4 & \mathbf{e}_5 & \mathbf{e}_6 & \mathbf{e}_7 & \mathbf{e}_8 \end{bmatrix}, \tag{3.10a}$$

with $\mathbf{e}_i \in \mathbb{R}^3$, for $1 \leq i \leq 8$, and the shape function is defined as

$$\mathbf{s}(u,v,w) \equiv \mathbf{B}_{12}^{-1} \cdot \mathbf{b}(u,v,w) \in \mathbb{R}^8 \, . \tag{3.10b}$$

3.3.2 ANCF 3443 Shell Element

The 3443 ANCF shell element is a fully parameterized 4-node element, as shown in Figure 3.3. Each of the four nodes in this quadrilateral element is defined by a position vector and three corresponding position vector gradients. Altogether, this configuration yields 48 degrees of freedom for a single unconstrained element, which is double that of the fully parameterized 2-node ANCF beam element 3243.

The vector of basis functions for the 3443 shell element is:

$$\mathbf{b}^T = [1, x, y, z, xz, yz, yx, x^2, z^2, x^3, z^3, x^2z, z^2x, xyz, x^3z, xz^3]$$

The constant interpolation matrix $\mathbf{B} \in \mathbb{R}^{16\times 16}$ can be constructed by evaluating the shape functions and their partial derivatives with respect to x, y, and z at each of the four nodes:

$$\mathbf{B_{1234}} = [\mathbf{b}(P_1),\ \mathbf{b}_{,x}(P_1),\ \mathbf{b}_{,y}(P_1),\ \mathbf{b}_{,z}(P_1),\ \ldots,\ \mathbf{b}(P_4),\ \mathbf{b}_{,x}(P_4),\ \mathbf{b}_{,y}(P_4),\ \mathbf{b}_{,z}(P_4)]$$

The time-dependent global nodal coordinates can be grouped similarly to the beam case,

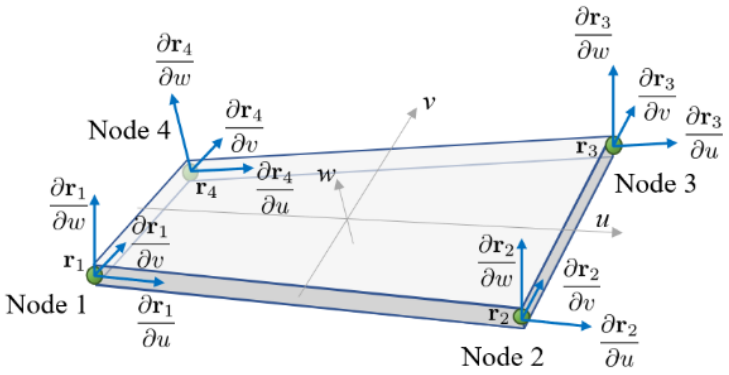


Figure 3.3: Visualization of a fully-parameterized 4-node ANCF shell element 3443.

into three vectors containing the x, y, and z components across all nodes and their gradients:

$$\mathbf{x}_{1234}^{\top} \equiv [\, x_1,\ x_{1,x},\ x_{1,y},\ x_{1,z},\ x_2,\ x_{2,x},\ x_{2,y},\ x_{2,z},\ x_3,\ x_{3,x},\ x_{3,y},\ x_{3,z},\ x_4,\ x_{4,x},\ x_{4,y},\ x_{4,z}\,],$$
$$\mathbf{y}_{1234}^{\top} \equiv [\, y_1,\ y_{1,x},\ y_{1,y},\ y_{1,z},\ y_2,\ y_{2,x},\ y_{2,y},\ y_{2,z},\ y_3,\ y_{3,x},\ y_{3,y},\ y_{3,z},\ y_4,\ y_{4,x},\ y_{4,y},\ y_{4,z}\,],$$
$$\mathbf{z}_{1234}^{\top} \equiv [\, z_1,\ z_{1,x},\ z_{1,y},\ z_{1,z},\ z_2,\ z_{2,x},\ z_{2,y},\ z_{2,z},\ z_3,\ z_{3,x},\ z_{3,y},\ z_{3,z},\ z_4,\ z_{4,x},\ z_{4,y},\ z_{4,z}\,]\ .$$

As with the 3243 element, the unknown coefficient vectors $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ can be determined using the inverse of the matrix $\mathbf{B}$:

$$\boldsymbol{\alpha} = \mathbf{B}^{-T} \cdot \mathbf{x}_{1234},$$
$$\boldsymbol{\beta} = \mathbf{B}^{-T} \cdot \mathbf{y}_{1234},$$
$$\boldsymbol{\gamma} = \mathbf{B}^{-T} \cdot \mathbf{z}_{1234}\ .$$

Therefore, the position vector field can be expressed as:

$$\mathbf{r}(x,y,z,t) = \mathrm{col}\left(\mathbf{x}_{1234}^{\top},\ \mathbf{y}_{1234}^{\top},\ \mathbf{z}_{1234}^{\top}\right) \mathbf{B_{1234}}^{-1} \cdot \mathbf{b}(u,v,w) = \mathbf{N}(t) \cdot \mathbf{s}(u,v,w)\ ,$$

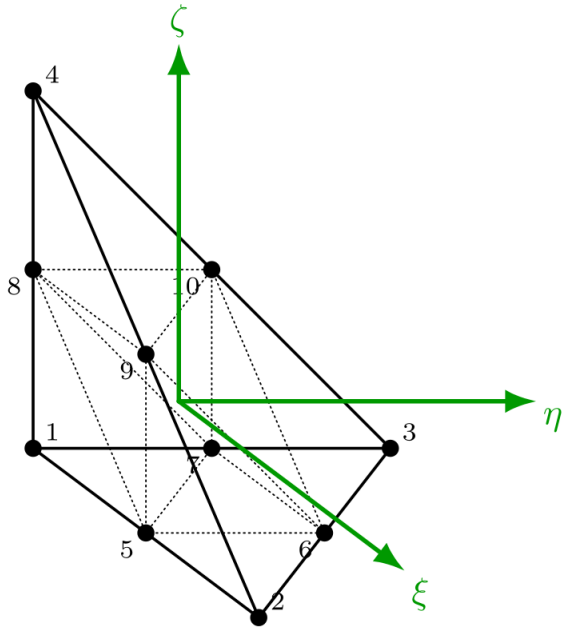


Figure 3.4: Visualization of a 10-node tetrahedron (T10) element.

where $\mathbf{N}(t) \in \mathbb{R}^{3\times 16}$ and $\mathbf{s}(x,y,z) \in \mathbb{R}^{16\times 1}$ are defined as:

$$\mathbf{N}(t) \equiv \begin{bmatrix} x_1 & x_{1,x} & x_{1,y} & x_{1,z} & x_2 & x_{2,x} & x_{2,y} & x_{2,z} & x_3 & x_{3,x} & x_{3,y} & x_{3,z} & x_4 & x_{4,x} & x_{4,y} & x_{4,z} \\ y_1 & y_{1,x} & y_{1,y} & y_{1,z} & y_2 & y_{2,x} & y_{2,y} & y_{2,z} & y_3 & y_{3,x} & y_{3,y} & y_{3,z} & y_4 & y_{4,x} & y_{4,y} & y_{4,z} \\ z_1 & z_{1,x} & z_{1,y} & z_{1,z} & z_2 & z_{2,x} & z_{2,y} & z_{2,z} & z_3 & z_{3,x} & z_{3,y} & z_{3,z} & z_4 & z_{4,x} & z_{4,y} & z_{4,z} \end{bmatrix}$$

$$= [\, \mathbf{e}_1 \;\; \mathbf{e}_2 \;\; \ldots \;\; \mathbf{e}_{16} \,], \quad \text{with } \mathbf{e}_i \in \mathbb{R}^3 \tag{3.11a}$$

$$\mathbf{s}(u,v,w) \equiv {\mathbf{B}_{\mathbf{1234}}}^{-1} \cdot \mathbf{b}(u,v,w) \in \mathbb{R}^{16} \,. \tag{3.11b}$$

3.3.3 10-Node Tetrahedron Element

The 10-node tetrahedron (T10), as shown in Figure 3.4, is a quadratic isoparametric solid element whose shape functions are defined on a parent tetrahedron with parametric coordinates (ξ, η, ζ); the isoparametric mapping carries these parent coordinates to the physical reference

configuration $\mathbf{X}$, as described below. The parent domain is the unit simplex

$$\xi \geq 0, \quad \eta \geq 0, \quad \zeta \geq 0, \quad \xi + \eta + \zeta \leq 1. \tag{3.12}$$

It contains four corner nodes at the vertices of the simplex and six nodes located at the midpoints of the edges. The nodal positions in the parent domain can be expressed in barycentric coordinates L_i $(i = 1, \ldots, 4)$,

$$L_1 = 1 - \xi - \eta - \zeta, \qquad L_2 = \xi, \qquad L_3 = \eta, \qquad L_4 = \zeta, \tag{3.13}$$

which satisfy $L_i \geq 0$ and $L_1 + L_2 + L_3 + L_4 = 1$. The four vertex nodes correspond to $L_i = 1$ at each corner, and the six mid-edge nodes correspond to $L_i = L_j = 1/2$ on each edge (i, j).

The T10 element employs the quadratic Lagrange basis on the tetrahedral simplex. In terms of barycentric coordinates, the shape functions for the four corner nodes are

$$N_1 = L_1(2L_1 - 1), \qquad N_2 = L_2(2L_2 - 1), \qquad N_3 = L_3(2L_3 - 1), \qquad N_4 = L_4(2L_4 - 1), \tag{3.14}$$

and the six edge-midpoint nodes are

$$\begin{aligned} N_5 &= 4L_1L_2, \qquad N_6 = 4L_2L_3, \qquad N_7 = 4L_3L_1, \\ N_8 &= 4L_1L_4, \qquad N_9 = 4L_2L_4, \qquad N_{10} = 4L_3L_4. \end{aligned} \tag{3.15}$$

These functions form a complete basis for all polynomials of total degree ≤ 2 on the tetrahedron (dimension 10) and satisfy the usual isoparametric properties:

$$\sum_{a=1}^{10} N_a(\xi, \eta, \zeta) = 1, \qquad N_a(\xi_b, \eta_b, \zeta_b) = \delta_{ab}, \tag{3.16}$$

which implies C^0 continuity across conforming meshes and Kronecker delta interpolation of nodal degrees of freedom.

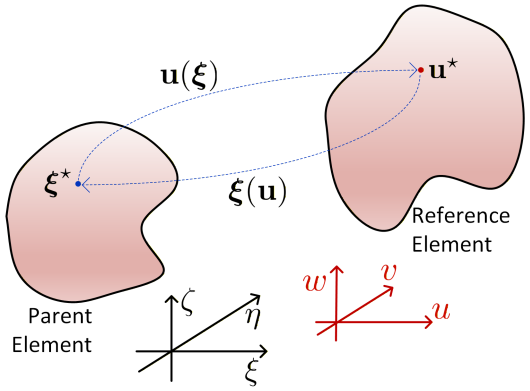


Figure 3.5: The one-to-one mapping from the parent element to the reference configuration. A necessary condition for this change of variables to be well-defined is that the parent-to-reference Jacobian $J_p(\xi,\eta,\zeta) := \det\Big(\partial\mathbf{X}/\partial(\xi,\eta,\zeta)\Big) > 0$ throughout the parent domain.

Let $\mathbf{X}_a$ and $\mathbf{x}_a$ denote the nodal positions in the reference and current configurations, respectively. The isoparametric mappings from the parent coordinates (ξ,η,ζ) to the reference and current configurations are

$$\mathbf{X}(\xi,\eta,\zeta) = \sum_{a=1}^{10} N_a(\xi,\eta,\zeta)\,\mathbf{X}_a, \qquad \mathbf{x}(\xi,\eta,\zeta) = \sum_{a=1}^{10} N_a(\xi,\eta,\zeta)\,\mathbf{x}_a. \tag{3.17}$$

The deformation gradient in the Total Lagrangian setting is obtained from

$$\mathbf{F}(\xi,\eta,\zeta) = \frac{\partial\mathbf{x}}{\partial\mathbf{X}} = \frac{\partial\mathbf{x}}{\partial(\xi,\eta,\zeta)}\left(\frac{\partial\mathbf{X}}{\partial(\xi,\eta,\zeta)}\right)^{-1}. \tag{3.18}$$

The formula above assumes that the parent-to-reference map is one-to-one and sufficiently regular: for every evaluation point $(\xi^\star,\eta^\star,\zeta^\star)$ in the parent element the Jacobian $\partial\mathbf{X}/\partial(\xi,\eta,\zeta)$ must be invertible, and a valid element mesh requires

$$J_p(\xi,\eta,\zeta) := \det\left(\frac{\partial\mathbf{X}}{\partial(\xi,\eta,\zeta)}\right) > 0 \tag{3.19}$$

throughout the parent domain (see Figure 3.5). Both Jacobians in the deformation gradient

are computed from the shape function derivatives $N_{a,\xi}$, $N_{a,\eta}$, $N_{a,\zeta}$, and the nodal reference coordinates $\mathbf{X}_a$, so that the standard Total Lagrangian strain measures follow directly.

Element integrals in the reference configuration are evaluated over the parent tetrahedron using Gaussian quadrature. In this work we employ a five-point rule that is exact for polynomials of total degree up to three, with quadrature points (ξ_q, η_q, ζ_q) and weights W_q chosen such that

$$\int_{\Omega_0^e} (\cdot)\, \mathrm{d}V = \int_{\hat{\Omega}} (\cdot)\, J_p(\xi, \eta, \zeta)\, \mathrm{d}\xi\, \mathrm{d}\eta\, \mathrm{d}\zeta \approx \sum_{q=1}^{5} (\cdot)_q\, J_{p,q}\, W_q, \tag{3.20}$$

where $\hat{\Omega}$ is the parent tetrahedron, $J_p = \det\left(\partial \mathbf{X}/\partial(\xi, \eta, \zeta)\right)$ is the parent-to-reference Jacobian, and $(\cdot)_q$ denotes the integrand evaluated at quadrature point q. The same quadrature rule is used for both mass and internal-force contributions, providing a consistent and robust discretization for the T10 element.

3.4 Hyperelastic Material Models and Internal Force Contribution to the EOM

3.4.1 Generic TL internal force expression

The contribution of the internal force to the EOM for the ANCF beam element and ANCF 3443 shell element is discussed and derived as follows:

Following the notation used in Equation 3.10a, for ANCF 3243, the expression for deformation gradient:

$$\begin{aligned}
\mathbf{F} &= \frac{\partial \mathbf{r}}{\partial \mathbf{u}} = \mathbf{N}(t) \cdot \frac{\partial \mathbf{s}}{\partial \mathbf{u}} \\
&= \mathbf{N}(t) \cdot \mathrm{col}(s_{1,\mathbf{u}}, s_{2,\mathbf{u}}, s_{3,\mathbf{u}}, s_{4,\mathbf{u}}, s_{5,\mathbf{u}}, s_{6,\mathbf{u}}, s_{7,\mathbf{u}}, s_{8,\mathbf{u}}) \\
&= \mathbf{e}_1 s_{1,\mathbf{u}} + \mathbf{e}_2 s_{2,\mathbf{u}} + \ldots + \mathbf{e}_8 s_{8,\mathbf{u}} = \sum_{i=1}^{8} \mathbf{e}_i s_{i,\mathbf{u}} \in \mathbb{R}^{3\times 3}\ .
\end{aligned}$$

Following the notation used earlier, the deformation gradient $\mathbf{F}$ for the 3443 shell element is:

$$\mathbf{F} = \frac{\partial \mathbf{r}}{\partial \mathbf{u}} = \sum_{i=1}^{16} \mathbf{e}_i \, s_{i,\mathbf{u}} \in \mathbb{R}^{3\times 3}$$

where each $s_{i,\mathbf{u}} \in \mathbb{R}^{1\times 3}$.

3.4.2 St Venant–Kirchhoff (SVK) material

The Saint-Venant–Kirchhoff (SVK) model is the simplest hyperelastic extension of linear elasticity to the finite-strain regime [12]. It retains the same Lamé-constant parameterization as the linear theory but replaces the infinitesimal strain tensor with the Green–Lagrange strain $\mathbf{E}$, leading to a quadratic strain energy density $\Psi(\mathbf{E})$ (Eq. (3.21)) and a linear stress–strain relationship in terms of the second Piola–Kirchhoff stress $\mathbf{S}$. This structure makes SVK well suited to problems involving large rotations with moderate stretches, and its closed-form stress and tangent expressions integrate directly into the TL assembly pipeline described in preceding sections. A known limitation is unphysical softening under large compressive strains [86], which restricts its applicability to regimes where stretches remain moderate.

The strain energy density function is quadratic in $\mathbf{E}$:

$$\Psi(\mathbf{E}) = \frac{\lambda}{2}\Big(\mathrm{tr}(\mathbf{E})\Big)^2 + \mu \; \mathrm{tr}\Big(\mathbf{E}^2\Big), \tag{3.21}$$

where λ and μ are the Lamé constants.

The second Piola–Kirchhoff stress follows from differentiation:

$$\mathbf{S} = \lambda \; \mathrm{tr}(\mathbf{E}) \, \mathbf{I} + 2\mu \, \mathbf{E}. \tag{3.22}$$

From this, the first Piola–Kirchhoff stress is:

$$\mathbf{P} = \mathbf{F}\mathbf{S} = \lambda \; \mathrm{tr}(\mathbf{E}) \, \mathbf{F} + 2\mu \, \mathbf{F}\mathbf{E}. \tag{3.23}$$

Using $\mathbf{E} = \frac{1}{2}\Big(\mathbf{F}^T\mathbf{F} - \mathbf{I}\Big)$, this can also be written as:

$$\mathbf{P} = \lambda \left(\tfrac{1}{2}\,\mathrm{tr}(\mathbf{F}^T\mathbf{F}) - \tfrac{3}{2}\right)\mathbf{F} + \mu\,\mathbf{F}\mathbf{F}^T\mathbf{F} - \mu\,\mathbf{F}. \tag{3.24}$$

The SVK model is only accurate for small to moderate strains, although it can accommodate arbitrarily large rotations. For large strains, it can give non-physical predictions, such as negative stiffness in compression.

For hyperelastic materials, the first Piola–Kirchhoff stress satisfies $\mathbf{P} = \partial\Psi/\partial\mathbf{F}$; for the SVK model this is consistent with Eqs. (3.23) and (3.24).

The internal force associated with nodal unknown $\mathbf{e}_i$ is defined as the gradient of the strain energy with respect to $\mathbf{e}_i$:

$$\mathbf{f}_i^T := \frac{\partial}{\partial \mathbf{e}_i}\int_{V_r} \Psi(\mathbf{E})\,\mathrm{d}V_r = \int_{V_r} \frac{\partial \Psi(\mathbf{E})}{\partial \mathbf{e}_i}\,\mathrm{d}V_r = \int_{V_r} \frac{\partial \Psi(\mathbf{E})}{\partial \mathbf{F}} : \frac{\partial \mathbf{F}}{\partial \mathbf{e}_i}\,\mathrm{d}V_r \tag{3.25}$$

$$= \int_{V_r} \mathbf{P} : \frac{\partial \mathbf{F}}{\partial \mathbf{e}_i}\,\mathrm{d}V_r = \int_{V_r} (\mathbf{P}\mathbf{h}_i)^T\,\mathrm{d}V_r. \tag{3.26}$$

which leads to the following result:

$$\mathbf{f}_i = \int_{V_r} \mathbf{P}\,\mathbf{h}_i\,\mathrm{d}V_r, \tag{3.27}$$

where $\mathbf{P}$ is the first Piola–Kirchhoff stress tensor and $\mathbf{h}_i := s_{i,\mathbf{u}}^T \in \mathbb{R}^{3\times 1}$ is the reference-gradient of the shape function $s_i(u, v, w)$. Note that this important relation holds for *any* hyperelastic material model, not only for the SVK material model. Plugging in the expression of $\mathbf{P}$ for the SVK material model, see Eq. (3.24), we get:

$$\mathbf{f}_i = \int_{V_r} \left[\lambda\left(\tfrac{1}{2}\,\mathrm{tr}(\mathbf{F}^T\mathbf{F}) - \tfrac{3}{2}\right)\mathbf{F} + \mu\mathbf{F}\mathbf{F}^T\mathbf{F} - \mu\mathbf{F}\right]\mathbf{h}_i\,\mathrm{d}V_r, \tag{3.28}$$

and therefore, the expression of the virtual work of the internal force over the element is

given by:

$$\delta W_{\text{int}} = \sum_{i=1}^{n_u} \delta \mathbf{e}_i^T \cdot \mathbf{f}_i. \tag{3.29}$$

Second-order (Newton-type) methods (Chapter 5) require the consistent linearization of the internal force vector, i.e., the Jacobian (tangent stiffness) $\partial \mathbf{f}_i / \partial \mathbf{e}_j$. For the SVK material model, this Jacobian reads

$$\begin{aligned} \frac{\partial \mathbf{f}_i}{\partial \mathbf{e}_j} = \int_{V_r} \Big[& \lambda (\mathbf{F}\mathbf{h}_i)(\mathbf{F}\mathbf{h}_j)^T + \lambda \operatorname{tr}(\mathbf{E}) \, (\mathbf{h}_j^T \mathbf{h}_i) \, \mathbf{I} \\ & + \mu (\mathbf{F}\mathbf{h}_j)^T (\mathbf{F}\mathbf{h}_i) \, \mathbf{I} + \mu (\mathbf{F}\mathbf{h}_j)(\mathbf{F}\mathbf{h}_i)^T + \mu (\mathbf{h}_j^T \mathbf{h}_i) \, \mathbf{F}\mathbf{F}^T - \mu (\mathbf{h}_j^T \mathbf{h}_i) \, \mathbf{I} \Big] \, \mathrm{d}V_r. \end{aligned} \tag{3.30}$$

3.4.3 Mooney–Rivlin (MR) material

The Mooney–Rivlin material model is a widely adopted hyperelastic constitutive framework for finite element analysis (FEA) of rubber-like materials, formulated in terms of strain invariants I_1 and I_2 of the right Cauchy–Green tensor to accommodate the near-incompressible behavior of elastomers [87, 88]. In this work, we employ the compressible two-parameter form in Eq. (3.31), which augments the isochoric Mooney–Rivlin response with a volumetric penalty term [12]. The model is effective for moderate to large strains and is widely used for elastomers and soft tissues due to its computational efficiency and compatibility with total Lagrangian formulations [89].

It is expressed directly in terms of the deformation gradient $\mathbf{F}$ and the invariants of the right Cauchy–Green tensor $\mathbf{C} = \mathbf{F}^T\mathbf{F}$. It depends on the following parameters:

- μ_{10}, μ_{01} are material parameters related to the shear response,
- k is a bulk modulus controlling compressibility.

The Mooney–Rivlin model reduces to the neo-Hookean model when $\mu_{01} = 0$, and to a purely incompressible form when $k \to \infty$ with $J = 1$.

The compressible Mooney–Rivlin strain energy density function is given by

$$\Psi(\mathbf{F}) = \mu_{10}\,(\bar{I}_1 - 3) + \mu_{01}\,(\bar{I}_2 - 3) + \frac{k}{2}\,(J-1)^2, \tag{3.31}$$

where

$$\begin{aligned}
&\mathbf{C} = \mathbf{F}^T\mathbf{F} && \text{(right Cauchy–Green tensor)},\\
&I_1 = \operatorname{tr}(\mathbf{C}),\\
&I_2 = \tfrac{1}{2}\Big[(\operatorname{tr}\mathbf{C})^2 - \operatorname{tr}(\mathbf{C}^2)\Big],\\
&J = \det\mathbf{F},\\
&\bar{I}_1 = J^{-2/3}I_1,\\
&\bar{I}_2 = J^{-4/3}I_2.
\end{aligned}$$

We will compute the first Piola–Kirchhoff stress

$$\mathbf{P} = \frac{\partial\Psi}{\partial\mathbf{F}} \tag{3.32}$$

using the chain rule. First, the required derivative identities are:

$$\begin{aligned}
\frac{\partial I_1}{\partial\mathbf{F}} &= 2\mathbf{F},\\
\frac{\partial I_2}{\partial\mathbf{F}} &= 2\Big(I_1\mathbf{F} - \mathbf{F}\mathbf{C}\Big),\\
\frac{\partial J}{\partial\mathbf{F}} &= J\mathbf{F}^{-T},\\
\frac{\partial J^{-m}}{\partial\mathbf{F}} &= -mJ^{-m}\mathbf{F}^{-T}.
\end{aligned}$$

We now compute the derivatives of the isochoric invariants:

$$\frac{\partial\bar{I}_1}{\partial\mathbf{F}} = \frac{\partial(J^{-2/3}I_1)}{\partial\mathbf{F}} = J^{-2/3}\frac{\partial I_1}{\partial\mathbf{F}} + I_1\frac{\partial J^{-2/3}}{\partial\mathbf{F}}$$

$$
\begin{aligned}
&= J^{-2/3}(2\mathbf{F}) - \frac{2}{3} I_1 J^{-2/3}\mathbf{F}^{-T} \\
&= 2J^{-2/3}\mathbf{F} - \frac{2}{3} J^{-2/3} I_1 \mathbf{F}^{-T}.
\end{aligned} \tag{3.33}
$$

Similarly,

$$
\begin{aligned}
\frac{\partial \bar{I}_2}{\partial \mathbf{F}} &= \frac{\partial (J^{-4/3} I_2)}{\partial \mathbf{F}} \\
&= J^{-4/3}\frac{\partial I_2}{\partial \mathbf{F}} + I_2 \frac{\partial J^{-4/3}}{\partial \mathbf{F}} \\
&= J^{-4/3}\Big[2(I_1\mathbf{F} - \mathbf{F}\mathbf{C})\Big] - \frac{4}{3} I_2 J^{-4/3}\mathbf{F}^{-T} \\
&= 2J^{-4/3}(I_1\mathbf{F} - \mathbf{F}\mathbf{C}) - \frac{4}{3} J^{-4/3} I_2 \mathbf{F}^{-T}.
\end{aligned} \tag{3.34}
$$

The derivative of the volumetric penalty term is:

$$
\begin{aligned}
\frac{\partial}{\partial \mathbf{F}}\left[\frac{k}{2}(J-1)^2\right] &= k(J-1)\frac{\partial J}{\partial \mathbf{F}} \\
&= k(J-1)J\mathbf{F}^{-T}.
\end{aligned} \tag{3.35}
$$

Putting everything together:

$$
\begin{aligned}
\mathbf{P} &= \mu_{10}\frac{\partial \bar{I}_1}{\partial \mathbf{F}} + \mu_{01}\frac{\partial \bar{I}_2}{\partial \mathbf{F}} + k(J-1)J\mathbf{F}^{-T} \\
&= 2\mu_{10}J^{-2/3}\mathbf{F} - \frac{2}{3}\mu_{10}J^{-2/3}I_1\mathbf{F}^{-T} \\
&\quad + 2\mu_{01}J^{-4/3}(I_1\mathbf{F} - \mathbf{F}\mathbf{C}) - \frac{4}{3}\mu_{01}J^{-4/3}I_2\mathbf{F}^{-T} \\
&\quad + k(J-1)J\mathbf{F}^{-T}.
\end{aligned} \tag{3.36}
$$

Thus, the first Piola–Kirchhoff stress for the compressible Mooney–Rivlin model is:

$$
\boxed{\mathbf{P} = 2\mu_{10}J^{-2/3}\Big(\mathbf{F} - \tfrac{1}{3}I_1\mathbf{F}^{-T}\Big) + 2\mu_{01}J^{-4/3}\Big(I_1\mathbf{F} - \mathbf{F}\mathbf{C} - \tfrac{2}{3}I_2\mathbf{F}^{-T}\Big) + k(J-1)J\mathbf{F}^{-T}} \tag{3.37}
$$

Using Eq. (3.27) together with Eq. (3.37), the internal force associated with the nodal

unknown $\mathbf{e}_i$ admits the decomposition

$$
\begin{aligned}
\mathbf{f}_i &= \mathbf{t}_1 - \mathbf{t}_2 + \mathbf{t}_3 - \mathbf{t}_4 - \mathbf{t}_5 + \mathbf{t}_6, \\
\mathbf{t}_1 &= 2\mu_{10} \int_{V_r} J^{-2/3}\, \mathbf{F}\mathbf{h}_i \,\mathrm{d}V_r, \\
\mathbf{t}_2 &= \frac{2\mu_{10}}{3} \int_{V_r} J^{-2/3} I_1\, \mathbf{F}^{-T}\mathbf{h}_i \,\mathrm{d}V_r, \\
\mathbf{t}_3 &= 2\mu_{01} \int_{V_r} J^{-4/3} I_1\, \mathbf{F}\mathbf{h}_i \,\mathrm{d}V_r, \\
\mathbf{t}_4 &= 2\mu_{01} \int_{V_r} J^{-4/3}\, \mathbf{F}\mathbf{F}^T\mathbf{F}\mathbf{h}_i \,\mathrm{d}V_r, \\
\mathbf{t}_5 &= \frac{4\mu_{01}}{3} \int_{V_r} J^{-4/3} I_2\, \mathbf{F}^{-T}\mathbf{h}_i \,\mathrm{d}V_r, \\
\mathbf{t}_6 &= k \int_{V_r} J\,(J-1)\, \mathbf{F}^{-T}\mathbf{h}_i \,\mathrm{d}V_r.
\end{aligned}
\tag{3.38}
$$

Second-order (Newton-type) methods require the Jacobian of the internal force, $\mathbf{K}_{ij} := \partial\mathbf{f}_i/\partial\mathbf{e}_j \in \mathbb{R}^{3\times 3}$. Differentiating Eq. (3.38) yields

$$
\begin{aligned}
\frac{\partial \mathbf{f}_i}{\partial \mathbf{e}_j} &= \frac{\partial \mathbf{t}_1}{\partial \mathbf{e}_j} - \frac{\partial \mathbf{t}_2}{\partial \mathbf{e}_j} + \frac{\partial \mathbf{t}_3}{\partial \mathbf{e}_j} - \frac{\partial \mathbf{t}_4}{\partial \mathbf{e}_j} - \frac{\partial \mathbf{t}_5}{\partial \mathbf{e}_j} + \frac{\partial \mathbf{t}_6}{\partial \mathbf{e}_j}, \\
\frac{\partial \mathbf{t}_1}{\partial \mathbf{e}_j} &= 2\mu_{10} \int_{V_r} J^{-2/3}\Big[(\mathbf{h}_j^T\mathbf{h}_i)\,\mathbf{I} - \frac{2}{3}(\mathbf{F}\mathbf{h}_i)(\mathbf{F}^{-T}\mathbf{h}_j)^T\Big]\,\mathrm{d}V_r, \\
\frac{\partial \mathbf{t}_2}{\partial \mathbf{e}_j} &= \frac{2\mu_{10}}{3} \int_{V_r} J^{-2/3}\Big[(\mathbf{F}^{-T}\mathbf{h}_i)(2\mathbf{F}\mathbf{h}_j)^T - \frac{2}{3}I_1(\mathbf{F}^{-T}\mathbf{h}_i)(\mathbf{F}^{-T}\mathbf{h}_j)^T - I_1(\mathbf{F}^{-T}\mathbf{h}_j)(\mathbf{F}^{-T}\mathbf{h}_i)^T\Big]\,\mathrm{d}V_r, \\
\frac{\partial \mathbf{t}_3}{\partial \mathbf{e}_j} &= 2\mu_{01} \int_{V_r} J^{-4/3}\Big[I_1(\mathbf{h}_j^T\mathbf{h}_i)\,\mathbf{I} + (\mathbf{F}\mathbf{h}_i)(2\mathbf{F}\mathbf{h}_j)^T - \frac{4}{3}I_1(\mathbf{F}\mathbf{h}_i)(\mathbf{F}^{-T}\mathbf{h}_j)^T\Big]\,\mathrm{d}V_r, \\
\frac{\partial \mathbf{t}_4}{\partial \mathbf{e}_j} &= 2\mu_{01} \int_{V_r} J^{-4/3}\Big[(\mathbf{F}\mathbf{h}_j)^T(\mathbf{F}\mathbf{h}_i)\,\mathbf{I} + (\mathbf{F}\mathbf{h}_j)(\mathbf{F}\mathbf{h}_i)^T + (\mathbf{h}_j^T\mathbf{h}_i)\,\mathbf{F}\mathbf{F}^T - \frac{4}{3}\,\mathbf{F}\mathbf{F}^T(\mathbf{F}\mathbf{h}_i)(\mathbf{F}^{-T}\mathbf{h}_j)^T\Big]\,\mathrm{d}V_r, \\
\frac{\partial \mathbf{t}_5}{\partial \mathbf{e}_j} &= \frac{4\mu_{01}}{3} \int_{V_r} J^{-4/3}\Big[(\mathbf{F}^{-T}\mathbf{h}_i)\Big(2(\mathbf{F}\mathbf{h}_j)^T(I_1\mathbf{I} - \mathbf{F}\mathbf{F}^T) - \frac{4}{3}I_2(\mathbf{F}^{-T}\mathbf{h}_j)^T\Big) - I_2(\mathbf{F}^{-T}\mathbf{h}_j)(\mathbf{F}^{-T}\mathbf{h}_i)^T\Big]\,\mathrm{d}V_r, \\
\frac{\partial \mathbf{t}_6}{\partial \mathbf{e}_j} &= k \int_{V_r} J\Big[(2J-1)(\mathbf{F}^{-T}\mathbf{h}_i)(\mathbf{F}^{-T}\mathbf{h}_j)^T - (J-1)(\mathbf{F}^{-T}\mathbf{h}_j)(\mathbf{F}^{-T}\mathbf{h}_i)^T\Big]\,\mathrm{d}V_r.
\end{aligned}
\tag{3.39}
$$

3.5 Viscous Damping Model in TL-FEA Formulation

3.5.1 Kinematics and Strain-Rate Measure in the Reference Configuration

In the Total Lagrangian setting, all kinematic quantities are expressed with respect to the reference configuration. Let $\mathbf{X}$ denote a material point in the reference configuration and $\mathbf{x}(\mathbf{X}, t)$ its current position at time t. The deformation gradient is

$$\mathbf{F}(\mathbf{X}, t) = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}, \qquad J = \det \mathbf{F}. \tag{3.40}$$

The right Cauchy–Green tensor and Green–Lagrange strain tensor are given by

$$\mathbf{C} = \mathbf{F}^T \mathbf{F}, \qquad \mathbf{E} = \frac{1}{2}(\mathbf{C} - \mathbf{I}). \tag{3.41}$$

The material time derivative of the deformation gradient is

$$\dot{\mathbf{F}} = \frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{X}}, \tag{3.42}$$

and the corresponding Green–Lagrange strain rate follows from

$$\dot{\mathbf{E}} = \frac{1}{2}\big(\dot{\mathbf{C}}\big) = \frac{1}{2}\big(\dot{\mathbf{F}}^T \mathbf{F} + \mathbf{F}^T \dot{\mathbf{F}}\big). \tag{3.43}$$

The tensor $\dot{\mathbf{E}}$ constitutes the basic strain-rate measure used to define viscous damping in the reference configuration.

For the ANCF 3243 beam and 3443 shell elements, the deformation gradient at a given quadrature point can be expressed, following the notation in the previous subsections, as

$$\mathbf{F} = \sum_i \mathbf{e}_i \, \mathbf{s}_{i,\mathbf{u}}, \tag{3.44}$$

where $\mathbf{e}_i \in \mathbb{R}^3$ denotes the vector of nodal unknowns (position and gradients) and $\mathbf{s}_{i,\mathbf{u}}$ is the row vector of shape-function derivatives with respect to the local coordinates. The time derivative $\dot{\mathbf{F}}$ is then

$$\dot{\mathbf{F}} = \sum_i \dot{\mathbf{e}}_i \, \mathbf{s}_{i,\mathbf{u}}, \tag{3.45}$$

and $\dot{\mathbf{E}}$ is obtained from (3.43). For the T10 element, analogous expressions are obtained by replacing the ANCF basis with the isoparametric T10 shape functions and their reference gradients.

3.5.2 Finite-Strain Kelvin–Voigt Law in the Reference Configuration

The viscous damping model adopted in this work is a finite-strain Kelvin–Voigt law formulated in the reference configuration, where the viscous contribution to the stress depends linearly on the Green–Lagrange strain rate $\dot{\mathbf{E}}$ defined in (3.43). This type of reference-configuration viscoelastic extension of a hyperelastic backbone is widely used in finite-strain constitutive modeling and finite element formulations, see for example Simo [90], Reese and Govindjee [91], Holzapfel [92], Holzapfel and Simo [93], and Treutenaere et al. [94]. The total second Piola–Kirchhoff stress $\mathbf{S}$ is decomposed into an elastic and a viscous contribution,

$$\mathbf{S} = \mathbf{S}^{\mathrm{el}} + \mathbf{S}^{\mathrm{vis}}, \tag{3.46}$$

where $\mathbf{S}^{\mathrm{el}}$ is provided by the chosen hyperelastic backbone (SVK or Mooney–Rivlin), and $\mathbf{S}^{\mathrm{vis}}$ is a linear function of $\dot{\mathbf{E}}$ of Kelvin–Voigt type.

In analogy with the small-strain isotropic Kelvin–Voigt model, the viscous second Piola–Kirchhoff stress is chosen as

$$\mathbf{S}^{\mathrm{vis}} = 2\eta \, \dot{\mathbf{E}} + \lambda_v \, \mathrm{tr}(\dot{\mathbf{E}}) \, \mathbf{I}, \tag{3.47}$$

where $\eta \geq 0$ and $\lambda_v \geq 0$ are viscosity-like parameters that play a role analogous to the

shear and bulk Lamé constants in the elastic law. The total first Piola–Kirchhoff stress then becomes

$$\mathbf{P} = \mathbf{F}\,\mathbf{S} = \mathbf{F}\,\mathbf{S}^{\mathrm{el}} + \mathbf{F}\,\mathbf{S}^{\mathrm{vis}} = \mathbf{P}^{\mathrm{el}} + \mathbf{P}^{\mathrm{vis}}, \tag{3.48}$$

with

$$\mathbf{P}^{\mathrm{el}} = \mathbf{F}\,\mathbf{S}^{\mathrm{el}}, \qquad \mathbf{P}^{\mathrm{vis}} = \mathbf{F}\,\mathbf{S}^{\mathrm{vis}}. \tag{3.49}$$

This formulation is compatible with both SVK and Mooney–Rivlin hyperelastic models, since only the elastic part $\mathbf{S}^{\mathrm{el}}$ changes when switching the constitutive backbone, while the viscous Kelvin–Voigt contribution (3.47) retains the same structure [90, 91, 92].

3.5.3 Dissipation and Thermodynamic Consistency

The viscous part of the stress should dissipate, rather than create, mechanical energy. In the reference configuration, the power of internal forces per unit reference volume associated with the viscous stress is

$$\dot{W}_{\mathrm{diss}} = \mathbf{S}^{\mathrm{vis}} : \dot{\mathbf{E}}. \tag{3.50}$$

Using the Kelvin–Voigt law (3.47),

$$\begin{aligned} \dot{W}_{\mathrm{diss}} &= \Big(2\eta\,\dot{\mathbf{E}} + \lambda_v\ \mathrm{tr}(\dot{\mathbf{E}})\,\mathbf{I}\Big) : \dot{\mathbf{E}} \\ &= 2\eta\,\dot{\mathbf{E}} : \dot{\mathbf{E}} + \lambda_v\ \mathrm{tr}(\dot{\mathbf{E}})\,\Big(\mathbf{I} : \dot{\mathbf{E}}\Big) \\ &= 2\eta\,\dot{\mathbf{E}} : \dot{\mathbf{E}} + \lambda_v\,\Big[\mathrm{tr}(\dot{\mathbf{E}})\Big]^2. \end{aligned} \tag{3.51}$$

Both terms on the right-hand side are quadratic forms in $\dot{\mathbf{E}}$. For $\eta \geq 0$ and $\lambda_v \geq 0$ one obtains

$$\dot{W}_{\mathrm{diss}} \geq 0 \tag{3.52}$$

for all admissible strain rates, so that the viscous model is thermodynamically consistent in the sense that it cannot generate mechanical energy. This type of quadratic dissipation potential and non-negativity requirement is standard in finite-strain viscoelastic theories [90, 91, 92,

93, 94]. In particular, $\dot{W}_{\text{diss}}$ vanishes if and only if $\dot{\mathbf{E}} = \mathbf{0}$, corresponding to a configuration with no strain evolution.

3.5.4 Element-Level Formulation with Viscous Damping

The derivation of the internal nodal forces for hyperelastic materials in the Total Lagrangian formulation leads to

$$\mathbf{f}_i = \int_{V_r} \mathbf{P}\,\mathbf{h}_i \,\mathrm{d}V_r, \tag{3.53}$$

where V_r is the element volume in the reference configuration, $\mathbf{P}$ is the first Piola–Kirchhoff stress, and $\mathbf{h}_i$ denotes the Jacobian (with respect to $\mathbf{X}$) of the shape function corresponding to nodal unknown $\mathbf{e}_i$. This expression holds for any hyperelastic constitutive model, as discussed earlier.

When the viscous contribution (3.48) is included, the internal force naturally splits into an elastic part and a viscous part,

$$\mathbf{f}_i = \mathbf{f}_i^{\text{el}} + \mathbf{f}_i^{\text{vis}}, \tag{3.54}$$

with

$$\mathbf{f}_i^{\text{el}} = \int_{V_r} \mathbf{P}^{\text{el}}\,\mathbf{h}_i \,\mathrm{d}V_r, \qquad \mathbf{f}_i^{\text{vis}} = \int_{V_r} \mathbf{P}^{\text{vis}}\,\mathbf{h}_i \,\mathrm{d}V_r. \tag{3.55}$$

Using $\mathbf{P}^{\text{vis}} = \mathbf{F}\mathbf{S}^{\text{vis}}$ and (3.47), the viscous force contribution at a quadrature point is computed as follows:

1. Assemble $\mathbf{F}$ from the current nodal unknowns (ANCF gradients or T10 displacements) and shape-function gradients.
2. Assemble $\dot{\mathbf{F}}$ from the current nodal velocities and the same shape-function gradients.
3. Compute $\dot{\mathbf{E}}$ using (3.43).
4. Evaluate $\mathbf{S}^{\text{vis}}$ by (3.47).
5. Form $\mathbf{P}^{\text{vis}} = \mathbf{F}\mathbf{S}^{\text{vis}}$.

6. Accumulate the contribution $\mathbf{P}^{\text{vis}}\mathbf{h}_i$ into $\mathbf{f}_i^{\text{vis}}$ at the quadrature point, multiply by the geometric Jacobian and quadrature weight, and sum over all quadrature points.

The total internal force vector for the element is the sum of the elastic and viscous contributions. This procedure is identical for the ANCF 3243 beam, ANCF 3443 shell, and T10 solid elements, differing only in the specific definitions of the shape functions and their reference gradients.

The corresponding contribution to the element tangent matrix (if an implicit time integrator is employed) can be obtained by linearizing $\mathbf{f}_i^{\text{vis}}$ with respect to the nodal velocities and displacements. In the present work, the viscous term is primarily used to regularize the explicit dynamics and to introduce physically motivated damping; therefore, an explicit evaluation of $\mathbf{f}_i^{\text{vis}}$ is sufficient.

3.5.5 Remarks on Applicability and Limitations

The finite-strain Kelvin–Voigt damping model adopted here is simple, computationally inexpensive, and easy to incorporate into the Total Lagrangian finite element framework. It provides a physically motivated regularization that dissipates mechanical energy in proportion to the Green–Lagrange strain rate and can be combined with both the SVK and Mooney–Rivlin hyperelastic backbones without modifying the underlying elastic formulation.

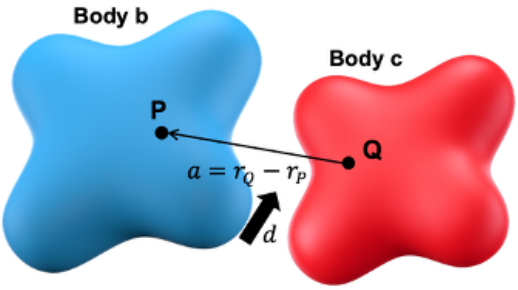
It should be noted, however, that a Kelvin–Voigt model based on $\dot{\mathbf{E}}$ does not provide an exact description of material viscoelasticity over broad frequency ranges and is not strictly objective under large, time-dependent rigid-body rotations. In many flexible multibody applications and large-deformation beam and shell problems, these limitations are acceptable, and the model offers a reasonable compromise between physical fidelity and numerical simplicity. For materials with strongly frequency-dependent damping or complex relaxation behavior, more sophisticated viscoelastic models (e.g., generalized Maxwell or standard linear solid formulations) could be employed within the same Total Lagrangian framework at the expense of additional internal variables and implementation complexity.

3.6 Kinematic Constraints

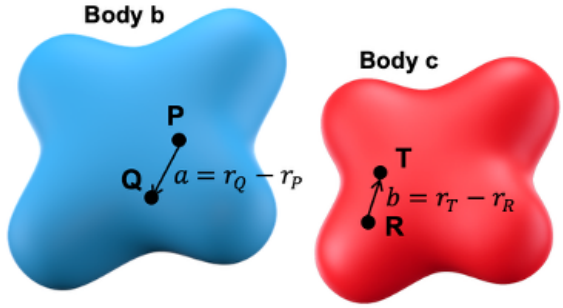
We summarize the representation and enforcement of bilateral kinematic constraints in the proposed TL-FEA multibody setting. We employ a compact set of scalar geometric primitives—dot-product 1 (DP1), dot-product 2 (DP2), distance (DIST), and coordinate-difference (CD)—as building blocks for intermediate constraints and engineering joints obtained through constraint composition. Practical applications may involve additional primitives and joint-specific constraints beyond those listed here; we therefore restrict attention to these four canonical forms to present the essential theory. The same point-evaluation and linearization operators introduced earlier extend directly to richer constraint libraries. Each primitive is posed as a scalar holonomic condition at the position level, and its first variation provides the Jacobian contributions required by the velocity-level augmented Lagrangian residual in Eq. (5.6). This section is organized around three components. We first define a unified constraint interface consistent with the step map $\mathbf{q}_{n+1} = \mathbf{q}_n + h\,\mathbf{v}$ and detail the four primitive constraints in TL-FEA notation. We then show how these primitives are composed into representative engineering joints. Numerical tests for these constraints are deferred to Chapter 6.

3.6.1 Primitive Constraints

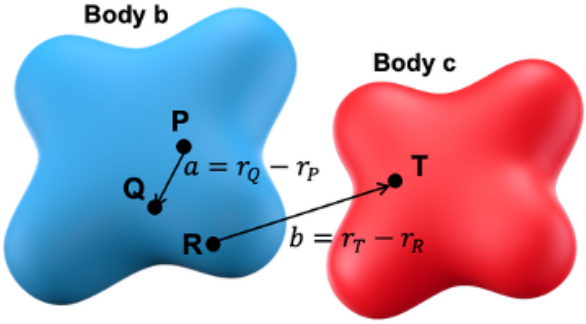
The primitives DP1, DP2, DIST, and CD are scalar compositions of point evaluations and point differences, so their Jacobians follow the same chain-rule pattern. For compactness, we state each primitive as a scalar constraint $c(\mathbf{q}, t) = 0$, provide its first variation δc, and summarize the corresponding nonzero element-level Jacobian blocks implied by the first-order variation of the TL-FEA point-evaluation operator. Figure 3.6 gives a geometric overview of the four primitives; the subsections below derive each one in detail.



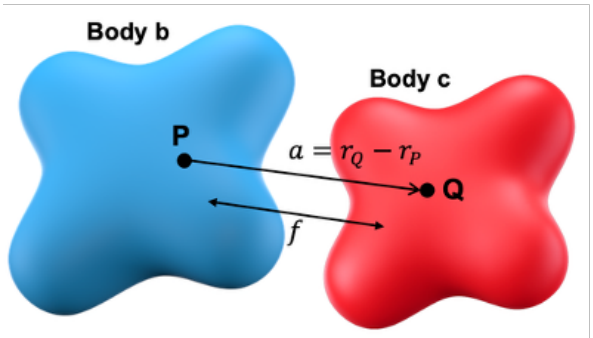
(a) CD: Coordinate difference. Point $\mathcal{P}$ on body b, point $\mathcal{Q}$ on body c; the vector $\mathbf{a} = \mathbf{r}_\mathcal{Q} - \mathbf{r}_\mathcal{P}$ is projected onto a fixed unit direction $\mathbf{d}$.



(b) DP1: Direction–direction. Directions $\mathbf{a}$ and $\mathbf{b}$ are each defined by a pair of material points on their respective bodies; the constraint prescribes their dot product.



(c) DP2: Direction–connector. Direction $\mathbf{a}$ is body-attached; connector $\mathbf{b} = \mathbf{r}_\mathcal{T} - \mathbf{r}_\mathcal{R}$ spans from body b to body c; the constraint prescribes $\mathbf{a}^\mathsf{T}\mathbf{b}$.



(d) DIST: Point distance. Points $\mathcal{P}$ and $\mathcal{Q}$ lie on different bodies; the scalar f prescribes their separation $\|\mathbf{r}_\mathcal{Q} - \mathbf{r}_\mathcal{P}\|$.

Figure 3.6: Geometric interpretation of the four scalar primitive constraints. $\mathcal{P}, \mathcal{Q}, \mathcal{R}, \mathcal{T}$ are material points on deformable bodies (body b: blue; body c: red). Each primitive contributes one scalar row to the constraint vector $\mathbf{c}(\mathbf{q})$; engineering joints are assembled by stacking multiple rows drawn from this set.

3.6.1.1 Dot-product 1 (DP1)

DP1 prescribes the dot product between two directions, each defined by a pair of points (Figure 3.6b). Let $\mathcal{P}, \mathcal{Q}$ lie on body b (elements E, F) and $\mathcal{R}, \mathcal{T}$ lie on body c (elements G, H). With the directions

$$\mathbf{a} = \mathbf{r}_{\mathcal{Q}} - \mathbf{r}_{\mathcal{P}}, \qquad \mathbf{b} = \mathbf{r}_{\mathcal{T}} - \mathbf{r}_{\mathcal{R}}, \tag{3.56}$$

DP1 enforces

$$c_{\mathrm{DP1}}(\mathbf{q}, t) = \mathbf{a}^{\mathsf{T}}\mathbf{b} - f(t) = 0, \tag{3.57}$$

where $f(t)$ is prescribed. In many applications f is constant, and $f \equiv 0$ yields a perpendicularity condition. The first variation is

$$\delta c_{\mathrm{DP1}} = \mathbf{b}^{\mathsf{T}}\delta\mathbf{a} + \mathbf{a}^{\mathsf{T}}\delta\mathbf{b}, \qquad \delta\mathbf{a} = \delta\mathbf{r}_{\mathcal{Q}} - \delta\mathbf{r}_{\mathcal{P}}, \quad \delta\mathbf{b} = \delta\mathbf{r}_{\mathcal{T}} - \delta\mathbf{r}_{\mathcal{R}}. \tag{3.58}$$

Using the point-evaluation linearization at each point, the only nonzero Jacobian blocks are those associated with the four hosting elements. For example, the contribution associated with point $\mathcal{P}$ has the form

$$\frac{\partial c_{\mathrm{DP1}}}{\partial \mathbf{e}_E^b} = -\mathbf{b}^{\mathsf{T}} \left(\mathbf{s}_E^b (\mathbf{u}_{\mathcal{P}})^{\mathsf{T}} \otimes \mathbf{I}_3 \right), \tag{3.59}$$

and the blocks associated with $\mathcal{Q}, \mathcal{R}, \mathcal{T}$ follow by replacing $(E, \mathcal{P}, -\mathbf{b})$ with $(F, \mathcal{Q}, +\mathbf{b})$, $(G, \mathcal{R}, -\mathbf{a})$, and $(H, \mathcal{T}, +\mathbf{a})$, respectively.

3.6.1.2 Dot-product 2 (DP2)

DP2 prescribes the dot product between a body-attached direction and a connector direction (Figure 3.6c). Let $\mathcal{P}, \mathcal{Q}, \mathcal{R}$ lie on body b (elements E, F, G) and $\mathcal{T}$ lie on body c (element H). Define

$$\mathbf{a} = \mathbf{r}_{\mathcal{Q}} - \mathbf{r}_{\mathcal{P}}, \qquad \mathbf{b} = \mathbf{r}_{\mathcal{T}} - \mathbf{r}_{\mathcal{R}}, \tag{3.60}$$

where $\mathbf{a}$ is a direction on body b and $\mathbf{b}$ connects point $\mathcal{R}$ on body b to point $\mathcal{T}$ on body c. DP2 enforces

$$c_{\mathrm{DP2}}(\mathbf{q},t) = \mathbf{a}^{\mathsf{T}}\mathbf{b} - f(t) = 0, \tag{3.61}$$

with first variation

$$\delta c_{\mathrm{DP2}} = \mathbf{b}^{\mathsf{T}}\delta\mathbf{a} + \mathbf{a}^{\mathsf{T}}\delta\mathbf{b}, \qquad \delta\mathbf{a} = \delta\mathbf{r}_{\mathcal{Q}} - \delta\mathbf{r}_{\mathcal{P}}, \quad \delta\mathbf{b} = \delta\mathbf{r}_{\mathcal{T}} - \delta\mathbf{r}_{\mathcal{R}}. \tag{3.62}$$

The only nonzero Jacobian blocks are associated with the hosting elements of $\mathcal{P}, \mathcal{Q}, \mathcal{R}$, and $\mathcal{T}$, and are given by

$$\begin{aligned}
\frac{\partial c_{\mathrm{DP2}}}{\partial \mathbf{e}^b_E} &= -\mathbf{b}^{\mathsf{T}}\left(\mathbf{s}^b_E(\mathbf{u}_{\mathcal{P}})^{\mathsf{T}} \otimes \mathbf{I}_3\right), & \frac{\partial c_{\mathrm{DP2}}}{\partial \mathbf{e}^b_F} &= +\mathbf{b}^{\mathsf{T}}\left(\mathbf{s}^b_F(\mathbf{u}_{\mathcal{Q}})^{\mathsf{T}} \otimes \mathbf{I}_3\right), \\
\frac{\partial c_{\mathrm{DP2}}}{\partial \mathbf{e}^b_G} &= -\mathbf{a}^{\mathsf{T}}\left(\mathbf{s}^b_G(\mathbf{u}_{\mathcal{R}})^{\mathsf{T}} \otimes \mathbf{I}_3\right), & \frac{\partial c_{\mathrm{DP2}}}{\partial \mathbf{e}^c_H} &= +\mathbf{a}^{\mathsf{T}}\left(\mathbf{s}^c_H(\mathbf{u}_{\mathcal{T}})^{\mathsf{T}} \otimes \mathbf{I}_3\right).
\end{aligned} \tag{3.63}$$

3.6.1.3 Distance (DIST)

DIST prescribes the separation between two points (Figure 3.6d). Let $\mathcal{P}$ lie on body b (element E) and $\mathcal{Q}$ lie on body c (element F), and define $\mathbf{a} = \mathbf{r}_{\mathcal{Q}} - \mathbf{r}_{\mathcal{P}}$. We employ the squared-distance form

$$c_{\mathrm{DIST}}(\mathbf{q},t) = \frac{1}{2}\left(\mathbf{a}^{\mathsf{T}}\mathbf{a} - f(t)^2\right) = 0, \tag{3.64}$$

with $f(t) > 0$. The first variation is

$$\delta c_{\mathrm{DIST}} = \mathbf{a}^{\mathsf{T}}\delta\mathbf{a}, \qquad \delta\mathbf{a} = \delta\mathbf{r}_{\mathcal{Q}} - \delta\mathbf{r}_{\mathcal{P}}. \tag{3.65}$$

Therefore, the only nonzero Jacobian blocks are those for the hosting elements of $\mathcal{P}$ and $\mathcal{Q}$:

$$\frac{\partial c_{\mathrm{DIST}}}{\partial \mathbf{e}^b_E} = -\mathbf{a}^{\mathsf{T}}\left(\mathbf{s}^b_E(\mathbf{u}_{\mathcal{P}})^{\mathsf{T}} \otimes \mathbf{I}_3\right), \qquad \frac{\partial c_{\mathrm{DIST}}}{\partial \mathbf{e}^c_F} = +\mathbf{a}^{\mathsf{T}}\left(\mathbf{s}^c_F(\mathbf{u}_{\mathcal{Q}})^{\mathsf{T}} \otimes \mathbf{I}_3\right). \tag{3.66}$$

3.6.1.4 Coordinate-difference (CD)

CD constrains one Cartesian component of a point difference (Figure 3.6a). Let $\mathcal{P}$ lie on body b (element E) and $\mathcal{Q}$ lie on body c (element F), and define $\mathbf{a} = \mathbf{r}_{\mathcal{Q}} - \mathbf{r}_{\mathcal{P}}$. Given a constant unit vector

$$\mathbf{d} \in \left\{ [1,0,0]^{\mathsf{T}},\ [0,1,0]^{\mathsf{T}},\ [0,0,1]^{\mathsf{T}} \right\}$$

selecting the x, y, or z component, CD enforces

$$c_{\mathrm{CD}}(\mathbf{q}, t) = \mathbf{d}^{\mathsf{T}}\mathbf{a} - f(t) = 0. \tag{3.67}$$

The first variation is

$$\delta c_{\mathrm{CD}} = \mathbf{d}^{\mathsf{T}}\delta\mathbf{a}, \qquad \delta\mathbf{a} = \delta\mathbf{r}_{\mathcal{Q}} - \delta\mathbf{r}_{\mathcal{P}}, \tag{3.68}$$

so that the only nonzero Jacobian blocks are

$$\frac{\partial c_{\mathrm{CD}}}{\partial \mathbf{e}_E^b} = -\mathbf{d}^{\mathsf{T}} \left(\mathbf{s}_E^b (\mathbf{u}_{\mathcal{P}})^{\mathsf{T}} \otimes \mathbf{I}_3 \right), \qquad \frac{\partial c_{\mathrm{CD}}}{\partial \mathbf{e}_F^c} = +\mathbf{d}^{\mathsf{T}} \left(\mathbf{s}_F^c (\mathbf{u}_{\mathcal{Q}})^{\mathsf{T}} \otimes \mathbf{I}_3 \right). \tag{3.69}$$

Vector coincidence constraints are obtained by stacking three CD equations with $\mathbf{d} = \mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$.

3.6.2 Engineering Joints

Engineering joints are obtained by composing the scalar primitives above into a constraint set with the desired relative motion restrictions. In what follows we focus on three representative joint types: the revolute joint, the cylindrical joint, and the fixed (weld) joint. Unless otherwise noted, the auxiliary direction-defining points are introduced in the reference configuration by a small offset δ from the attachment location along the joint axis and two independent perpendicular directions. This provides the reference dot products used in the DP1 and DP2 rows while keeping the construction element-agnostic.

For a joint with constraint vector $\mathbf{c}^{\mathrm{joint}}(\mathbf{q}, t) = \mathbf{0}$ and Jacobian $\mathbf{C}_q^{\mathrm{joint}}$, the contribution to

the velocity-level augmented-Lagrangian residual is

$$\mathbf{g}^{\text{joint}} = h\,\mathbf{C}_q^{\text{joint}\,\mathsf{T}}\left(\boldsymbol{\lambda}^{\text{joint}} + \rho\,\mathbf{c}^{\text{joint}}\right), \tag{3.70}$$

where $\boldsymbol{\lambda}^{\text{joint}}$ is the multiplier vector and $\rho > 0$ is the penalty parameter. In implementation, each scalar primitive row is assembled and scattered locally to the hosting elements exactly as in the primitive formulas above.

3.6.2.1 Joint Composition Summary

Table 3.1 summarizes the primitive decomposition for several common engineering joints. The detailed derivations below focus on the revolute, cylindrical, and fixed joints.

Table 3.1: Primitive decomposition for common engineering joints.

Joint type	CD	DP1	DP2	Total m	Remaining DOF
Spherical	3	0	0	3	3 rotational
Universal	3	1	0	4	2 rotational
Revolute	3	2	0	5	1 rotational
Fixed (weld)	3	3	0	6	0
Cylindrical	0	2	2	4	1 rot. + 1 trans.
Prismatic	0	3	2	5	1 translational

3.6.2.2 Revolute Joint

A revolute joint between two deformable bodies permits relative rotation about a single axis while constraining all relative translation and the two off-axis rotational degrees of freedom. In the present formulation it is assembled from three CD constraints and two DP1 constraints, yielding $m = 5$ scalar holonomic conditions.

Joint topology. Consider two deformable bodies b and c. Let P on body b and R on body c denote the attachment points. Let Q on body b define the hinge-axis direction

$$\mathbf{a} = \mathbf{r}_Q - \mathbf{r}_P, \tag{3.71}$$

and let two additional points S and T on body c define

$$\mathbf{d}_1 = \mathbf{r}_S - \mathbf{r}_R, \qquad \mathbf{d}_2 = \mathbf{r}_T - \mathbf{r}_R. \tag{3.72}$$

In the reference configuration, $\mathbf{d}_{1,0}$ and $\mathbf{d}_{2,0}$ must be linearly independent and not both parallel to the hinge axis. The five-point geometry is illustrated in Figure 3.7.

Constraint equations. The hinge location is enforced by three CD rows:

$$c_k^{\mathrm{CD}} = \mathbf{e}_k^{\mathsf{T}} (\mathbf{r}_R - \mathbf{r}_P) = 0, \qquad k = 1, 2, 3, \tag{3.73}$$

where $\mathbf{e}_1 = \mathbf{e}_x$, $\mathbf{e}_2 = \mathbf{e}_y$, and $\mathbf{e}_3 = \mathbf{e}_z$. The two off-axis rotational degrees of freedom are removed by the DP1 rows

$$c_1^{\mathrm{DP1}} = \mathbf{a}^{\mathsf{T}} \mathbf{d}_1 - f_1 = 0, \qquad c_2^{\mathrm{DP1}} = \mathbf{a}^{\mathsf{T}} \mathbf{d}_2 - f_2 = 0, \tag{3.74}$$

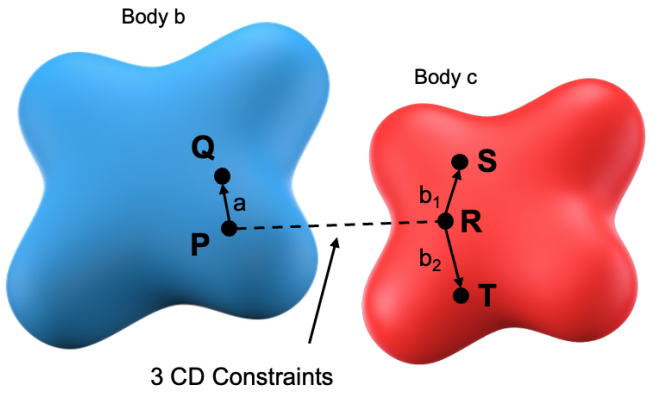


Figure 3.7: Geometry of the revolute joint construction. Points P, Q on body b (blue) define the body-attached axis direction $\mathbf{a} = \mathbf{r}_Q - \mathbf{r}_P$. Points R, S, T on body c (red) define the off-axis directions $\mathbf{d}_1 = \mathbf{r}_S - \mathbf{r}_R$ and $\mathbf{d}_2 = \mathbf{r}_T - \mathbf{r}_R$. Three CD constraints enforce point coincidence at the hinge location ($\mathbf{r}_P = \mathbf{r}_R$); two DP1 constraints prescribe the relative orientation through $\mathbf{a}^\mathsf{T}\mathbf{d}_1 = f_1$ and $\mathbf{a}^\mathsf{T}\mathbf{d}_2 = f_2$, leaving one rotational degree of freedom about the hinge axis.

with reference dot products

$$f_1 = \mathbf{a}_0^\mathsf{T}\mathbf{d}_{1,0}, \qquad f_2 = \mathbf{a}_0^\mathsf{T}\mathbf{d}_{2,0}. \tag{3.75}$$

The revolute-joint constraint vector is therefore

$$\mathbf{c}^{\text{rev}} = \text{col}\left(c_1^{\text{CD}}, c_2^{\text{CD}}, c_3^{\text{CD}}, c_1^{\text{DP1}}, c_2^{\text{DP1}}\right) = \mathbf{0} \in \mathbb{R}^5. \tag{3.76}$$

Joint-specific Jacobian accumulation. The primitive Jacobians follow directly from Eqs. (3.59)–(3.69). At the shared attachment points it is convenient to write the accumulated blocks explicitly. Defining

$$\boldsymbol{\sigma}_P^b := \mathbf{s}_{E_P}^b(\mathbf{u}_P)^\mathsf{T} \otimes \mathbf{I}_3, \qquad \boldsymbol{\sigma}_R^c := \mathbf{s}_{E_R}^c(\mathbf{u}_R)^\mathsf{T} \otimes \mathbf{I}_3,$$

the accumulated rows at P and R are

$$\begin{aligned}\frac{\partial \mathbf{c}^{\mathrm{rev}}}{\partial \mathbf{e}^{b}_{E_P}} &= -\operatorname{col}\Big(\mathbf{e}_x^{\mathsf{T}}, \mathbf{e}_y^{\mathsf{T}}, \mathbf{e}_z^{\mathsf{T}}, \mathbf{d}_1^{\mathsf{T}}, \mathbf{d}_2^{\mathsf{T}}\Big)\, \boldsymbol{\sigma}^{b}_{P},\\ \frac{\partial \mathbf{c}^{\mathrm{rev}}}{\partial \mathbf{e}^{c}_{E_R}} &= \operatorname{col}\Big(\mathbf{e}_x^{\mathsf{T}}, \mathbf{e}_y^{\mathsf{T}}, \mathbf{e}_z^{\mathsf{T}}, -\mathbf{a}^{\mathsf{T}}, -\mathbf{a}^{\mathsf{T}}\Big)\, \boldsymbol{\sigma}^{c}_{R}.\end{aligned} \tag{3.77}$$

The remaining points contribute only through the DP1 rows: Q enters c_1^{DP1} and c_2^{DP1}, S enters c_1^{DP1}, and T enters c_2^{DP1}.

Row normalization and Newton linearization. Mixed CD/DP1 joints exhibit a scale mismatch when the direction-defining fibers are short. For each scalar row c_i, we therefore introduce a constant reference-configuration weight $w_i > 0$ and use

$$\hat{c}_i = w_i\, c_i, \qquad \hat{\mathbf{C}}_{q,i} = w_i\, \mathbf{C}_{q,i}. \tag{3.78}$$

For CD rows we take $w_i = 1$. For any DP1 row formed from two directions $\mathbf{a}$ and $\mathbf{d}$, we use

$$w_i = \frac{1}{\sqrt{|\mathbf{a}_0|^2 + |\mathbf{d}_0|^2}}, \tag{3.79}$$

which balances the corresponding Gauss–Newton contribution. With $\mathbf{q} = \mathbf{q}_n + h\,\mathbf{v}$ and $\eta_i = \lambda_i + \rho\, c_i$, the constraint contribution to the Newton system is

$$\frac{\partial \mathbf{g}_{\mathrm{con}}}{\partial \mathbf{v}} = h^2 \rho\, \mathbf{C}_q^{\mathsf{T}} \mathbf{C}_q + h^2 \sum_{i=1}^{m} \eta_i\, \nabla_q^2 c_i. \tag{3.80}$$

The CD rows are linear in $\mathbf{q}$ and therefore have vanishing Hessian. Each DP1 row is bilinear in the nodal unknowns and has a constant Hessian. For $c_1^{\mathrm{DP1}} = (\mathbf{r}_Q - \mathbf{r}_P)^{\mathsf{T}} (\mathbf{r}_S - \mathbf{r}_R) - f_1$,

ordered as $(\mathbf{e}^b_{E_P}, \mathbf{e}^b_{E_Q}, \mathbf{e}^c_{E_R}, \mathbf{e}^c_{E_S})$, we obtain

$$\nabla^2_q c_1^{\mathrm{DP1}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & +\boldsymbol{\sigma}_P^{\mathsf{T}}\boldsymbol{\sigma}_R & -\boldsymbol{\sigma}_P^{\mathsf{T}}\boldsymbol{\sigma}_S \\ \mathbf{0} & \mathbf{0} & -\boldsymbol{\sigma}_Q^{\mathsf{T}}\boldsymbol{\sigma}_R & +\boldsymbol{\sigma}_Q^{\mathsf{T}}\boldsymbol{\sigma}_S \\ +\boldsymbol{\sigma}_R^{\mathsf{T}}\boldsymbol{\sigma}_P & -\boldsymbol{\sigma}_R^{\mathsf{T}}\boldsymbol{\sigma}_Q & \mathbf{0} & \mathbf{0} \\ -\boldsymbol{\sigma}_S^{\mathsf{T}}\boldsymbol{\sigma}_P & +\boldsymbol{\sigma}_S^{\mathsf{T}}\boldsymbol{\sigma}_Q & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{3.81}$$

and c_2^{DP1} has the same structure with S replaced by T. The Gauss–Newton contribution $h^2\rho\,\mathbf{C}_q^{\mathsf{T}}\mathbf{C}_q$ is positive semidefinite, while the curvature correction arising from the DP1 Hessian is indefinite in general. The full Newton Hessian therefore takes the form

$$\mathbf{H} = \tfrac{1}{h}\mathbf{M} + h\,\mathbf{K}_t + h^2\rho\,\mathbf{C}_q^{\mathsf{T}}\mathbf{C}_q + h^2\textstyle\sum_{i=1}^{m}\eta_i\,\nabla^2_q c_i, \tag{3.82}$$

where $\mathbf{K}_t = \partial\mathbf{f}_{\mathrm{int}}/\partial\mathbf{q}$ is the tangent stiffness. Retaining the curvature term yields the consistent linearization and quadratic Newton convergence, but the resulting matrix is not guaranteed positive definite and therefore requires a symmetric indefinite factorization rather than Cholesky. Omitting $h^2\sum_i \eta_i\,\nabla^2_q c_i$ preserves the symmetric positive definite (SPD) structure at the cost of superlinear rather than quadratic convergence. This SPD/non-SPD trade-off is inherited by every joint that contains dot-product constraints.

3.6.2.3 Cylindrical Joint

A cylindrical joint permits relative rotation about a shared axis and relative translation along that axis, while constraining all lateral translation and the two off-axis rotational degrees of freedom. It is assembled from two DP1 rows enforcing axis parallelism and two DP2 rows enforcing offset collinearity, yielding $m = 4$ scalar holonomic conditions.

Joint topology. Let P on body b and R on body c denote two attachment points on the shared axis; unlike the revolute joint, they need not coincide. Let

$$\mathbf{a}_b = \mathbf{r}_Q - \mathbf{r}_P, \qquad \mathbf{d}_1 = \mathbf{r}_S - \mathbf{r}_R, \qquad \mathbf{d}_2 = \mathbf{r}_U - \mathbf{r}_R, \tag{3.83}$$

where Q lies on body b and S, U lie on body c. Let two additional points V, W on body b define perpendicular directions

$$\mathbf{p}_1 = \mathbf{r}_V - \mathbf{r}_P, \qquad \mathbf{p}_2 = \mathbf{r}_W - \mathbf{r}_P, \tag{3.84}$$

and let

$$\mathbf{b} = \mathbf{r}_R - \mathbf{r}_P \tag{3.85}$$

denote the connector vector between the attachment points.

Constraint equations. The axis-parallelism rows are

$$c_1^{\text{par}} = \mathbf{a}_b^{\mathsf{T}}\mathbf{d}_1 - f_1^{\text{par}} = 0, \qquad c_2^{\text{par}} = \mathbf{a}_b^{\mathsf{T}}\mathbf{d}_2 - f_2^{\text{par}} = 0, \tag{3.86}$$

with $f_j^{\text{par}} = \mathbf{a}_{b,0}^{\mathsf{T}}\mathbf{d}_{j,0}$. The offset-collinearity rows are

$$c_1^{\text{col}} = \mathbf{p}_1^{\mathsf{T}}\mathbf{b} - f_1^{\text{col}} = 0, \qquad c_2^{\text{col}} = \mathbf{p}_2^{\mathsf{T}}\mathbf{b} - f_2^{\text{col}} = 0, \tag{3.87}$$

with $f_j^{\text{col}} = \mathbf{p}_{j,0}^{\mathsf{T}}\mathbf{b}_0$. The constraint vector is

$$\mathbf{c}^{\text{cyl}} = \text{col}\Big(c_1^{\text{par}}, c_2^{\text{par}}, c_1^{\text{col}}, c_2^{\text{col}}\Big) = \mathbf{0} \in \mathbb{R}^4. \tag{3.88}$$

These four rows remove two rotational and two translational relative degrees of freedom, leaving one axial rotation and one axial translation.

Jacobian and Hessian structure. The axis-parallelism rows follow the same DP1 pattern as the revolute joint. The offset-collinearity rows follow the DP2 body-ownership pattern and are distinct because point P appears in both $\mathbf{p}_j$ and $\mathbf{b}$. For $c_1^{\text{col}} = (\mathbf{r}_V - \mathbf{r}_P)^{\mathsf{T}} (\mathbf{r}_R - \mathbf{r}_P) - f_1^{\text{col}}$, the first variation is

$$\delta c_1^{\text{col}} = \mathbf{b}^{\mathsf{T}} \delta \mathbf{r}_V - (\mathbf{b} + \mathbf{p}_1)^{\mathsf{T}} \delta \mathbf{r}_P + \mathbf{p}_1^{\mathsf{T}} \delta \mathbf{r}_R, \tag{3.89}$$

which yields

$$\frac{\partial c_1^{\text{col}}}{\partial \mathbf{e}_{E_P}^b} = -(\mathbf{b} + \mathbf{p}_1)^{\mathsf{T}} \boldsymbol{\sigma}_P^b, \qquad \frac{\partial c_1^{\text{col}}}{\partial \mathbf{e}_{E_V}^b} = \mathbf{b}^{\mathsf{T}} \boldsymbol{\sigma}_V^b, \qquad \frac{\partial c_1^{\text{col}}}{\partial \mathbf{e}_{E_R}^c} = \mathbf{p}_1^{\mathsf{T}} \boldsymbol{\sigma}_R^c. \tag{3.90}$$

Unlike the four-point DP1 Hessian in Eq. (3.81), the corresponding DP2 Hessian has a nonzero diagonal block:

$$\nabla_q^2 c_1^{\text{col}} = \begin{bmatrix} +2\,\boldsymbol{\sigma}_P^{\mathsf{T}} \boldsymbol{\sigma}_P & -\boldsymbol{\sigma}_P^{\mathsf{T}} \boldsymbol{\sigma}_V & -\boldsymbol{\sigma}_P^{\mathsf{T}} \boldsymbol{\sigma}_R \\ -\boldsymbol{\sigma}_V^{\mathsf{T}} \boldsymbol{\sigma}_P & \mathbf{0} & +\boldsymbol{\sigma}_V^{\mathsf{T}} \boldsymbol{\sigma}_R \\ -\boldsymbol{\sigma}_R^{\mathsf{T}} \boldsymbol{\sigma}_P & +\boldsymbol{\sigma}_R^{\mathsf{T}} \boldsymbol{\sigma}_V & \mathbf{0} \end{bmatrix}, \tag{3.91}$$

ordered as $(\mathbf{e}_{E_P}^b, \mathbf{e}_{E_V}^b, \mathbf{e}_{E_R}^c)$. The second collinearity row has the same structure with V replaced by W and $\mathbf{p}_1$ by $\mathbf{p}_2$. This is the main structural difference between the cylindrical and revolute joints at the Newton system level.

3.6.2.4 Fixed (Weld) Joint

A fixed (weld) joint eliminates all relative motion between two deformable bodies, yielding $m = 6$ scalar holonomic conditions. It is assembled from three CD rows enforcing point coincidence and three DP1 rows locking the relative orientation.

Joint topology. Let P and R be the attachment points on bodies b and c. Let two independent directions on body b be

$$\mathbf{a}_1 = \mathbf{r}_Q - \mathbf{r}_P, \qquad \mathbf{a}_2 = \mathbf{r}_W - \mathbf{r}_P, \tag{3.92}$$

and two independent directions on body c be

$$\mathbf{d}_1 = \mathbf{r}_S - \mathbf{r}_R, \qquad \mathbf{d}_2 = \mathbf{r}_T - \mathbf{r}_R. \tag{3.93}$$

The point set therefore consists of six material points: P, Q, W on body b and R, S, T on body c.

Constraint equations. The coincidence block is

$$c_k^{\mathrm{CD}} = \mathbf{e}_k^{\mathsf{T}} \left(\mathbf{r}_R - \mathbf{r}_P\right) = 0, \qquad k = 1, 2, 3, \tag{3.94}$$

and the orientation-lock block is

$$\begin{aligned} c_1^{\mathrm{DP1}} &= \mathbf{a}_1^{\mathsf{T}} \mathbf{d}_1 - f_1 = 0, \\ c_2^{\mathrm{DP1}} &= \mathbf{a}_1^{\mathsf{T}} \mathbf{d}_2 - f_2 = 0, \\ c_3^{\mathrm{DP1}} &= \mathbf{a}_2^{\mathsf{T}} \mathbf{d}_2 - f_3 = 0, \end{aligned} \tag{3.95}$$

with

$$f_1 = \mathbf{a}_{1,0}^{\mathsf{T}} \mathbf{d}_{1,0}, \qquad f_2 = \mathbf{a}_{1,0}^{\mathsf{T}} \mathbf{d}_{2,0}, \qquad f_3 = \mathbf{a}_{2,0}^{\mathsf{T}} \mathbf{d}_{2,0}. \tag{3.96}$$

The corresponding constraint vector is

$$\mathbf{c}^{\mathrm{weld}} = \mathrm{col}\Big(c_1^{\mathrm{CD}}, c_2^{\mathrm{CD}}, c_3^{\mathrm{CD}}, c_1^{\mathrm{DP1}}, c_2^{\mathrm{DP1}}, c_3^{\mathrm{DP1}}\Big) = \mathbf{0} \in \mathbb{R}^6. \tag{3.97}$$

The first two DP1 rows lock the orientation of $\mathbf{a}_1$ relative to body c, while the third row removes the remaining twist degree of freedom by constraining $\mathbf{a}_2$ against $\mathbf{d}_2$.

Jacobian and Hessian remarks. The first two DP1 rows are identical in structure to the revolute-joint rows. The third DP1 row introduces the new direction $\mathbf{a}_2$ and yields

$$\frac{\partial c_3^{\mathrm{DP1}}}{\partial \mathbf{e}_{E_P}^b} = -\mathbf{d}_2^{\mathsf{T}} \boldsymbol{\sigma}_P^b, \qquad \frac{\partial c_3^{\mathrm{DP1}}}{\partial \mathbf{e}_{E_W}^b} = \mathbf{d}_2^{\mathsf{T}} \boldsymbol{\sigma}_W^b, \qquad \frac{\partial c_3^{\mathrm{DP1}}}{\partial \mathbf{e}_{E_R}^c} = -\mathbf{a}_2^{\mathsf{T}} \boldsymbol{\sigma}_R^c, \qquad \frac{\partial c_3^{\mathrm{DP1}}}{\partial \mathbf{e}_{E_T}^c} = \mathbf{a}_2^{\mathsf{T}} \boldsymbol{\sigma}_T^c. \tag{3.98}$$

At the shared attachment points, the accumulated blocks are

$$\begin{aligned} \frac{\partial \mathbf{c}^{\mathrm{weld}}}{\partial \mathbf{e}_{E_P}^b} &= -\operatorname{col}\left(\mathbf{e}_x^{\mathsf{T}}, \mathbf{e}_y^{\mathsf{T}}, \mathbf{e}_z^{\mathsf{T}}, \mathbf{d}_1^{\mathsf{T}}, \mathbf{d}_2^{\mathsf{T}}, \mathbf{d}_2^{\mathsf{T}}\right) \boldsymbol{\sigma}_P^b, \\ \frac{\partial \mathbf{c}^{\mathrm{weld}}}{\partial \mathbf{e}_{E_R}^c} &= \operatorname{col}\left(\mathbf{e}_x^{\mathsf{T}}, \mathbf{e}_y^{\mathsf{T}}, \mathbf{e}_z^{\mathsf{T}}, -\mathbf{a}_1^{\mathsf{T}}, -\mathbf{a}_1^{\mathsf{T}}, -\mathbf{a}_2^{\mathsf{T}}\right) \boldsymbol{\sigma}_R^c. \end{aligned} \tag{3.99}$$

All three DP1 Hessians follow the same four-point pattern as Eq. (3.81): they are constant, sparse, symmetric, and indefinite, while the CD rows remain linear and therefore contribute no Hessian correction. The same normalization strategy from Eq. (3.79) applies row by row to the three DP1 rows of the weld joint.

4 FRICTION AND CONTACT ASPECTS

4.1 GPU-Based Hydroelastic Contact Patch Method

This section describes the collision detection and contact generation pipeline used for deformable tetrahedral discretizations. The broad-phase stage generates candidate tetrahedron pairs using Sweep-and-Prune on per-element axis-aligned bounding boxes (AABBs), together with an adjacency filter that suppresses pairs of topologically neighboring elements (shared-node neighbors). The narrow-phase stage constructs an iso-pressure contact patch polygon for each candidate pair by intersecting an iso-pressure plane with the tetrahedra and clipping the resulting polygon to the overlap region. The resulting patch descriptor includes geometric and field-derived quantities (area, centroid, normal, and directional pressure gradients) that may be used to assemble nodal contact forces with optional dissipative and frictional contributions.

The formulation operates on tetrahedral *elements* rather than rigid bodies: the broad-phase produces candidate element pairs, and the narrow-phase produces (at most) one polygonal contact patch per candidate pair. The narrow-phase assumes a per-node scalar field p is provided and treats it as a pressure (or pressure potential) whose tetrahedral interpolation induces an affine field within each element. This description does not address BVH-based broad-phase methods, continuous collision detection, global complementarity-based contact solvers, or construction of p from signed-distance and material parameters.

We use the following notation. Let $\mathcal{T}$ denote the set of tetrahedral elements with indices $e \in \{0, \ldots, n_{\text{elems}} - 1\}$. For each element $e \in \mathcal{T}$, let $\text{AABB}_e = [\boldsymbol{b}_e^{\min}, \boldsymbol{b}_e^{\max}] \subset \mathbb{R}^3$ be its world-frame AABB with per-axis intervals $I_e^{(a)} = [b_{e,a}^{\min}, b_{e,a}^{\max}]$ for $a \in \{x, y, z\}$. The broad-phase returns a set of candidate element pairs $\mathcal{P} \subset \mathcal{T} \times \mathcal{T}$ with potentially overlapping AABBs (after adjacency filtering). The narrow-phase consumes $\mathcal{P}$ together with the nodal field p and produces one patch descriptor per candidate pair.

4.1.1 Broad-Phase: Sweep and Prune

The broad-phase stage associates each tetrahedral element with an AABB AABB_e computed by componentwise minima/maxima over the element's vertex coordinates. Candidate generation is performed by sorting elements by $b_x^{\min}$ and sweeping in increasing order. For each element i in the sorted order, only subsequent elements $j > i$ satisfying $b_{j,x}^{\min} \leq b_{i,x}^{\max}$ are considered; overlap on y and z is then tested explicitly. This ordering provides an early termination condition in the inner loop when $b_{j,x}^{\min} > b_{i,x}^{\max}$.

Two AABBs overlap on an axis a if and only if

$$I_i^{(a)} \cap I_j^{(a)} \neq \emptyset \quad \Leftrightarrow \quad b_{i,a}^{\min} \leq b_{j,a}^{\max} \ \wedge \ b_{j,a}^{\min} \leq b_{i,a}^{\max}. \tag{4.1}$$

In the sweep, candidates are generated by scanning forward from each element i in the sorted order and testing explicit overlap on y and z for the elements whose x-intervals overlap.

When tetrahedra belong to the same discretized body, many AABB overlaps occur between adjacent elements even in the absence of physical self-contact. To suppress these pairs, the broad-phase applies an adjacency filter using mesh topology: two elements are declared adjacent if they share at least one node, and such adjacent pairs are rejected during candidate generation.

Let $n = |\mathcal{T}|$. The broad-phase cost is dominated by the sort and the sweep. The sweep benefits from the early termination condition on the sweep axis; in the worst case it is $O(n^2)$ but is typically lower in sparse configurations. Adjacency filtering introduces a membership test for each overlapping AABB candidate pair (implemented efficiently by hashing shared-node neighborhoods).

Algorithm 1 Broadphase candidate generation (AABB sweep-and-prune with adjacency filtering)

Require: Element set $\mathcal{T}$ with updated AABBs $\{\text{AABB}_e\}_{e\in\mathcal{T}}$
Require: Optional adjacency set $\mathcal{N} \subset \mathcal{T} \times \mathcal{T}$ (e.g., shared-node adjacency)
Ensure: Candidate element pair list $\mathcal{P}$

```
Sort elements by b_x^min
P ← ∅
for each element i in sorted order do
    for each subsequent element j > i while b_{j,x}^min ≤ b_{i,x}^max do
        if AABB_i and AABB_j overlap on y and z then
            if (i, j) ∉ N then
                P ← P ∪ {(i, j)}
return P
```

4.1.2 Narrow-Phase: Hydroelastic Contact Patch for Deformable Bodies

The narrow-phase stage computes an iso-pressure patch for each tetrahedron pair returned by the broad-phase. The key assumption is that a per-node scalar field p (pressure or pressure potential) is provided. For each tetrahedron, nodal interpolation induces a unique affine field $\pi(\boldsymbol{x})$ over the element; in higher-order tetrahedral elements, one may specialize the procedure to use only the four corner nodes.

For each tetrahedron with vertices $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_3$ and per-vertex values $p_0, \ldots, p_3$, we reconstruct an affine pressure field of the form

$$\pi(\boldsymbol{x}) = \boldsymbol{a}^\top \boldsymbol{x} + b \tag{4.2}$$

by enforcing $\pi(\boldsymbol{v}_i) = p_i$. Writing edge vectors $\boldsymbol{e}_k = \boldsymbol{v}_k - \boldsymbol{v}_0$ for $k = 1, 2, 3$, one solves

$$\begin{bmatrix} \boldsymbol{e}_1^\top \\ \boldsymbol{e}_2^\top \\ \boldsymbol{e}_3^\top \end{bmatrix} \boldsymbol{a} = \begin{bmatrix} p_1 - p_0 \\ p_2 - p_0 \\ p_3 - p_0 \end{bmatrix}, \qquad b = p_0 - \boldsymbol{a}^\top \boldsymbol{v}_0. \tag{4.3}$$

If the associated linear system is (nearly) singular (degenerate tetrahedron or numerically

ill-conditioned configuration), the affine reconstruction is treated as ill-posed and the corresponding candidate pair is discarded.

For a candidate element pair (T_A, T_B) with reconstructed fields π_A and π_B, define $\psi(\boldsymbol{x}) = \pi_A(\boldsymbol{x}) - \pi_B(\boldsymbol{x})$. Since both fields are affine, the condition $\psi(\boldsymbol{x}) = 0$ defines an iso-pressure plane:

$$\psi(\boldsymbol{x}) = \boldsymbol{n}^\top \boldsymbol{x} + c, \qquad \boldsymbol{n} = \boldsymbol{a}_A - \boldsymbol{a}_B, \quad c = b_A - b_B. \tag{4.4}$$

The contact patch polygon is then defined as

$$Q = (T_A \cap T_B) \cap \{\boldsymbol{x} : \psi(\boldsymbol{x}) = 0\}. \tag{4.5}$$

In practice, Q is computed by first intersecting the iso-pressure plane with T_A to obtain a convex polygon, and then clipping this polygon against the four face half-spaces of T_B using Sutherland–Hodgman clipping (Algorithm 3). The patch area and centroid are computed by fan triangulation.

The patch normal is initialized as $\hat{\boldsymbol{n}} = \boldsymbol{n}/\|\boldsymbol{n}\|$ (the iso-pressure plane normal). Directional pressure gradients are recorded using the Drake convention [95]:

$$g_A = -\frac{\partial \pi_A}{\partial n} = -\boldsymbol{a}_A^\top \hat{\boldsymbol{n}}, \qquad g_B = \frac{\partial \pi_B}{\partial n} = \boldsymbol{a}_B^\top \hat{\boldsymbol{n}}. \tag{4.6}$$

If $g_A \leq 0$ or $g_B \leq 0$, the normal is flipped once and the test is repeated. If the condition still fails, the patch is marked as having invalid orientation and excluded from subsequent force assembly. In multi-body or multi-mesh settings, an additional ordering rule can be imposed (e.g., based on body or mesh identifiers) to enforce consistent assignment of the (A, B) labels and, consequently, consistent normal orientation across pairs.

The present formulation assumes both tetrahedra provide a pressure field and uses the equality condition $\pi_A = \pi_B$ to define the interface. A one-sided variant can be obtained by prescribing an interface surface and sampling a single pressure field. From a numerical perspective, the fidelity of the extracted patch depends on the resolution of the volumetric

Algorithm 2 Iso-pressure contact patch construction (tetrahedron–tetrahedron)

Require: Candidate element-pair set $\mathcal{P}$ from the broad-phase
Require: Nodal pressures p and tetrahedral connectivity for each element
Ensure: One patch descriptor per candidate pair (possibly invalid)
for each candidate pair $(T_A, T_B) \in \mathcal{P}$ **in parallel do**
  Build affine fields $\pi_A(\boldsymbol{x}) = \boldsymbol{a}_A^\top \boldsymbol{x} + b_A$ and $\pi_B(\boldsymbol{x}) = \boldsymbol{a}_B^\top \boldsymbol{x} + b_B$
  Form iso-pressure plane $\psi(\boldsymbol{x}) = \pi_A(\boldsymbol{x}) - \pi_B(\boldsymbol{x}) = \boldsymbol{n}^\top \boldsymbol{x} + c$
  Compute polygon $R \leftarrow T_A \cap \{\psi = 0\}$ by intersecting the plane with the six tetrahedron edges
  Clip R against the four half-spaces of T_B using Sutherland–Hodgman to obtain Q
  if Q has fewer than 3 vertices **then**
    mark patch invalid; **continue**
  Compute area A and centroid $\boldsymbol{x}_c$ of Q by fan triangulation
  Set $\hat{\boldsymbol{n}} \leftarrow \boldsymbol{n}/\|\boldsymbol{n}\|$, compute (g_A, g_B), and flip $\hat{\boldsymbol{n}}$ once if needed to satisfy $g_A > 0$ and $g_B > 0$
  Store patch vertices, A, $\boldsymbol{x}_c$, $\hat{\boldsymbol{n}}$, (g_A, g_B), and $p_{\mathrm{eq}} = \pi_A(\boldsymbol{x}_c)$

meshes and the accuracy and physical interpretation of the nodal field p. In the simplest lumped, one-point quadrature model, one resultant force is applied for each valid, well-oriented patch,

$$\boldsymbol{F}_{\mathrm{patch}} = p_{\mathrm{damped}}\, A\, \hat{\boldsymbol{n}}, \qquad p_{\mathrm{damped}} = p_{\mathrm{eq}}\, \max\Big(0,\, 1 - \beta\, v_{\mathrm{rel},n}\Big), \tag{4.7}$$

where $\beta \geq 0$ is a damping coefficient and $v_{\mathrm{rel},n}$ is the relative normal velocity at the patch centroid (estimated by barycentric interpolation of nodal velocities in each tetrahedron). A regularized Coulomb friction term may be added based on relative tangential velocity. The patch force is then distributed to the eight involved nodes using centroid barycentric weights and accumulated into nodal external forces.

In Algorithm 3, the segment–plane intersection may be computed as $\boldsymbol{q} = \boldsymbol{s} + t(\boldsymbol{e} - \boldsymbol{s})$, where $t = \frac{d - \boldsymbol{n}^\top \boldsymbol{s}}{\boldsymbol{n}^\top (\boldsymbol{e} - \boldsymbol{s})}$ when the denominator is nonzero. In floating-point arithmetic, predicates and degeneracy checks are evaluated with a tolerance, and polygons with area below a threshold are discarded before triangulation.

Algorithm 3 Sutherland–Hodgman reentrant polygon clipping algorithm

Require: Convex polygon vertex list $V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ in order
Require: Half-space set $\mathcal{H} = \{\boldsymbol{n}_k^\top \boldsymbol{x} \le d_k\}_{k=1}^{K}$
Ensure: Clipped polygon vertex list V'

$V^{(0)} \leftarrow V$
for $k = 1$ to K **do**
  $(\boldsymbol{n}, d) \leftarrow (\boldsymbol{n}_k, d_k)$
  $V^{(k)} \leftarrow \emptyset$
  if $V^{(k-1)} = \emptyset$ **then**
    return $\emptyset$
  for $i = 1$ to $|V^{(k-1)}|$ **do**
    $\boldsymbol{s} \leftarrow V_i^{(k-1)}$; $\boldsymbol{e} \leftarrow V_{(i \bmod |V^{(k-1)}|)+1}^{(k-1)}$
    $s_{\text{in}} \leftarrow (\boldsymbol{n}^\top \boldsymbol{s} \le d)$; $e_{\text{in}} \leftarrow (\boldsymbol{n}^\top \boldsymbol{e} \le d)$
    if s_{in} and e_{in} **then**
      Append $\boldsymbol{e}$ to $V^{(k)}$
    else if s_{in} and not e_{in} **then**
      Compute intersection $\boldsymbol{q}$ of segment $\overline{\boldsymbol{se}}$ with $\boldsymbol{n}^\top \boldsymbol{x} = d$
      Append $\boldsymbol{q}$ to $V^{(k)}$
    else if not s_{in} and e_{in} **then**
      Compute intersection $\boldsymbol{q}$ of segment $\overline{\boldsymbol{se}}$ with $\boldsymbol{n}^\top \boldsymbol{x} = d$
      Append $\boldsymbol{q}$ and then $\boldsymbol{e}$ to $V^{(k)}$
return $V^{(K)}$

Implementation note and limitation The hydroelastic contact patch formulation described above has been implemented in the research codebase; Figure 4.1 shows an example applied to a Stanford Bunny mesh clamped between two flat rigid paddles. This approach was not adopted as the primary contact algorithm in the framework reported here, however. The key obstacle is the construction of a physically valid pressure field for each contacting body. In the hydroelastic model, each body must carry a per-node scalar pressure value—typically $p(\boldsymbol{x}) = E_H \max(0, -\phi(\boldsymbol{x}))$, where ϕ is the signed distance field (SDF) to the body's surface and E_H is the hydroelastic modulus [96, 95]. For convex bodies this computation is straightforward: the closest surface point is unique and the interior/exterior sign is unambiguous. For non-convex geometries, the sign of ϕ—which must distinguish interior from exterior—cannot be recovered from distance alone; it requires additional processing such as

winding-number computation or flood-fill propagation, steps that are susceptible to failure on meshes with concavities, near-self-intersections, or non-manifold features [95, 97]. For a body with the complex topology of the Stanford Bunny, such preprocessing is non-trivial and does not transfer automatically across mesh resolutions or deformed configurations, making reliable pressure-field construction difficult to guarantee in a general-purpose deformable-body simulation.

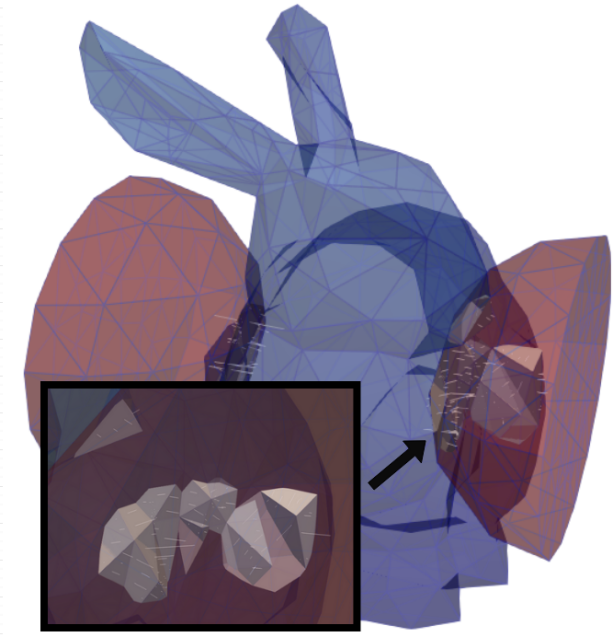


Figure 4.1: Stanford Bunny clamped between two bubble grippers in a hydroelastic contact patch test using the implemented algorithm in the companion codebase [1]. The highlighted polygons represent the iso-pressure contact patches generated by the narrow-phase algorithm. The white line segments indicate the contact normal direction.

4.2 GPU-Based Surface Mesh Collision Algorithm Design

Triangle-mesh (*trimesh–trimesh*) collision detection on GPUs is most often implemented using hierarchy-based culling, where bounding-volume hierarchies (BVHs) reduce the number

of expensive primitive-level tests and expose substantial data parallelism in both hierarchy construction and query processing [98, 20, 99, 100]. A recurring practical limitation is that proximity queries over BVHs remain highly data-dependent: traversal work and memory access patterns vary significantly across contact configurations, which can induce workload imbalance and control-flow divergence on SIMT hardware [101].

For narrow-phase contact between *convex* shapes, GJK and MPR-style methods are widely used due to their efficiency and generality in support-mapping form [102, 103]. However, extending these approaches to large triangle meshes typically requires convex decomposition or an equivalent strategy that converts a nonconvex mesh query into many convex queries, increasing the number of pair tests and amplifying the impact of data-dependent iteration counts. In a GPU setting, such iterative, branch-heavy control flow can exacerbate divergence and reduce throughput, particularly when the underlying geometry contains many triangles and the contact set evolves rapidly [101, 104].

A common GPU-friendly approximation is spherical decomposition (e.g., multi-sphere models or sphere-trees), which trades geometric fidelity for very inexpensive and easily batched distance checks [105, 106]. While this representation can be effective for real-time collision queries, achieving high accuracy near sharp features or thin structures may require a large number of spheres, increasing memory and broad-phase cost; conversely, using fewer spheres improves performance but introduces approximation error in contact location and normal estimation [105, 106].

In this section, we present a two-thread asynchronous, GPU-oriented collision detection algorithm for the Total Lagrangian finite element analysis framework for multibody dynamics. The two key features of this algorithm are: (i) using a triangle soup decomposed from meshes, rather than the original meshes, to derive the contact information; and (ii) using a two-thread asynchronous collision detection algorithm to fill the GPU workload as much as possible. These features are designed to increase data granularity and enforce execution overlap, better utilizing the GPU's hardware architecture.

4.2.1 A Two-Thread Asynchronous Collision Detection Algorithm

The implementation builds on the existing infrastructure of Chrono DEM-Engine [107], which uses two distinct and parallel computational threads to update the active contacts set (done by the "kinematics thread", abbreviated as kT), and the integration of the equations of motion (done by the "dynamics thread", abbreviated as dT), respectively. These threads are distinct from GPU threads; they are CPU-managed threads that each control a GPU stream and launch kernels. The dynamics thread processes each contact in the Active-Contact Set (ACS) at each time step to reassess the contact penetration δ_n and the ancillary information. The dynamics thread receives an ACS update when the kinematics thread finishes producing it, or if so desired, it can wait for the ACS update when the dynamics thread advances the system state too far ahead of the time stamp of the last ACS update from the kinematics thread. Through this collaboration pattern, the two threads work concurrently and the cost of contact detection is nearly hidden in the background of computation done by the dynamics thread, which continuously advances the state of the system.

To avoid missing mutual contacts that might crop up between the moments the ACS is updated, we artificially enlarge all contact geometries in the system, as detailed in Sec. 4.2.2. This extra margin allows for the preemptive detection of potential contact pairs that might emerge in the near future.

By adding this artificial margin to all contact geometries, the kinematics thread can report false positives within the provided list of contacts, i.e., a contact between two geometries might be in the ACS, yet the two geometries are not in contact. Such false positives will be identified and ruled out by the dynamics thread when carrying out the force calculation. The thickness of contact geometry i's added margin d_i is determined by the simulation entities' velocity magnitude v_i (which is bounded and known by the solver), the time step size h, which is typically small, and $n_{\max}$, the maximum number of time steps the dynamics thread

is allowed to advance without receiving an ACS update from the kinematics thread. Namely,

$$d_i = (av_i + b)hn_{\max}, \tag{4.8}$$

where a and b are configurable parameters that scale the velocity for contact safety, and take default values of 1 and 0.5, respectively.

The synchronization pattern between the kinematics and dynamics threads is illustrated in Figure 4.2. There, "**S**" represents a time step that the dynamics thread executes, where the contact forces are calculated (see Sec. 4.2.4), and the system state is advanced in time. A contact detection step that the kinematics thread executes is marked with "**CD**". Periodically, the kinematics thread finishes a contact detection step and sends the signal to the dynamics thread, allowing the dynamics thread to receive the contact array, "**ACS**", from the kinematics thread. Then the dynamics thread will send a work order "**WO**" with the current simulation system state, for the kinematics thread to pick up and continue the next contact detection step. Before the next "**ACS**" update is received, the dynamics thread will use this "**ACS**" to execute the time steps.

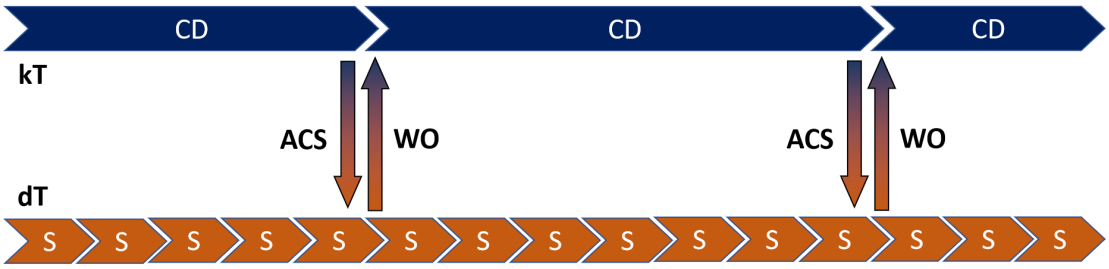


Figure 4.2: Two-thread asynchronous contact detection collaboration pattern, where the dynamics thread advances the physics continuously while the kinematics thread occasionally waits for updated state information to commence an ACS update.

4.2.2 Broad-Phase Contact Detection

The broad-phase contact detection algorithm running on the kinematics thread extends the approach outlined in [17]. This algorithm is optimized for execution on GPU, works with decomposed triangle soups and never requires mesh connectivities, and accommodates simulation entities of vastly differing sizes. In turn, its supplied ACS is also triangle pair-based, which is the expected format of the dynamics thread.

The contact detection process in each step involves a series of tasks, executed sequentially:

1. At the start of each contact detection step, for each triangle facet of the meshes, the solver adds a margin to it, thus creating the *future-proof* contact proxy for it. Recall that the solver determines the size of the margin d_i, then adds it to the triangle along all directions. This effectively creates "prisms" as the contact proxy, as shown in Figure 4.3. Prism contacts warrant potential underlying triangle contacts, so the broad-phase contact detection, at the implementation level, always works with prisms. This is to be differentiated from the force derivation process in Sec. 4.2.3, which still uses the original triangle, because there, the solver works with the present time step only. In this section only, we use "triangle" and "prism" interchangeably because they are the same underlying data structure in the broad-phase contact detection.

2. Initially, all triangle facets are evaluated for potential contact using "bins". These bins are formed by uniformly segmenting the simulation domain into axis-aligned cubic grids. If a triangle intersects with a bin, this bin–triangle pair is recorded for subsequent processing. This is also illustrated in Figure 4.3. It is important to note that due to the allowance for variable sizes of triangles, the maximum number of bins intersecting with a triangle cannot be predetermined. This necessitates two sequential CUDA kernel executions: one to determine the count of intersecting bins per triangle for memory allocation, and another to store the bin–triangle pairs.

3. Next, we sort the bin–triangle pairs based on the bin IDs. This step groups together the

triangles located within the same bin, effectively clustering entities that are adjacent within the simulation.

4. The final step involves checking triangles within the same bin for potential contacts. This is accomplished through launching CUDA blocks, each processing a bin, to derive all potential contact pairs. Similar to the previous step, two CUDA kernel calls are required: the first to ascertain the number of potential contacts per bin for array allocation, and the second to populate the ACS array with these contacts. The contacts are detected by separating-axis-theorem (SAT)-based tests between prisms' base triangles. Given n_{g} geometries in a bin, $(n_{\mathrm{g}} - 1)n_{\mathrm{g}}/2$ checks are necessary to identify all potential contacts. This imposes a limit on n_{g}, influencing the bin size. The bin size is adjusted dynamically based on execution history to maintain optimal performance. Duplicate contacts may be present, so they are then identified and discarded using a CUDA CUB-based device-level search.

5. Note that the triangles' corresponding patch IDs are also consolidated into arrays as part of the contact information. The patch ID is an artificial index assigned to triangles, such that the triangles sharing the same patch ID are considered to be "in the same contact", i.e., in the same connected "island" of mesh surface triangles. We also call the union of the connected triangles that share a patch ID a "patch".

   To generate patch IDs, a GPU-friendly "index flooding" (label propagation) over the triangle adjacency graph is carried out: each active triangle is given an initial label equal to its own index, and over several iterations, each triangle updates its label by taking the minimum label among itself and its neighboring triangles (using the precomputed neighbor lists). This repeatedly propagates the smallest index through connected components, so triangles connected by shared edges converge to the same label while disconnected components keep distinct labels. Finally, that converged "island ID" is combined with higher-level grouping keys (such as contact type identifiers), and the contacts are sorted so a prefix-scan can assign a contiguous patch ID.

The broad-phase contact information is transferred to the dynamics thread at the end of each kinematics thread step. Since the two threads are working asynchronously, the contact information is transferred to a buffer memory. Then the dynamics thread will be notified and copy the contact information to its working memory. The dynamics thread carries out a similar routine when updating the kinematics thread with new element positions. Neither of them directly modifies the working memory of the other to avoid race conditions. This is illustrated in Figure 4.4.

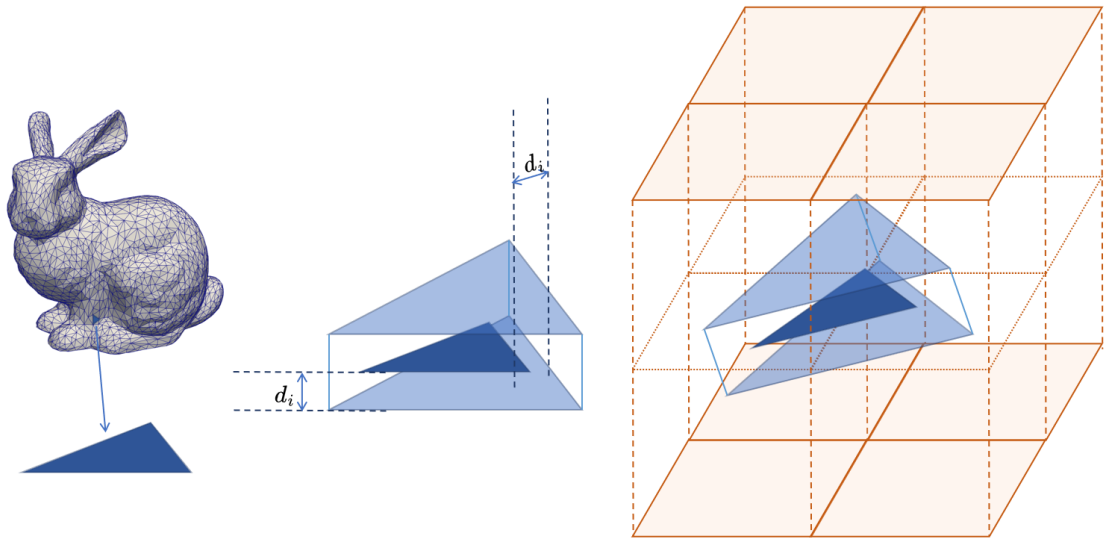


Figure 4.3: Left: The process of adding margin is done for every triangle of a mesh. Middle: The margin with size d_i is added to the triangle, thus creating a prism as the proxy for contact detection. Right: An example of a prism in contact with eight bins.

4.2.3 Narrow-Phase Detection Verification and Force Derivation

From the kinematics-thread-supplied potential ACS array, the dynamics thread detects the physical contacts and derives the contact force. The overall workflow is summarized in Figure 4.5.

Sequentially, the following steps are executed in one dynamics step:

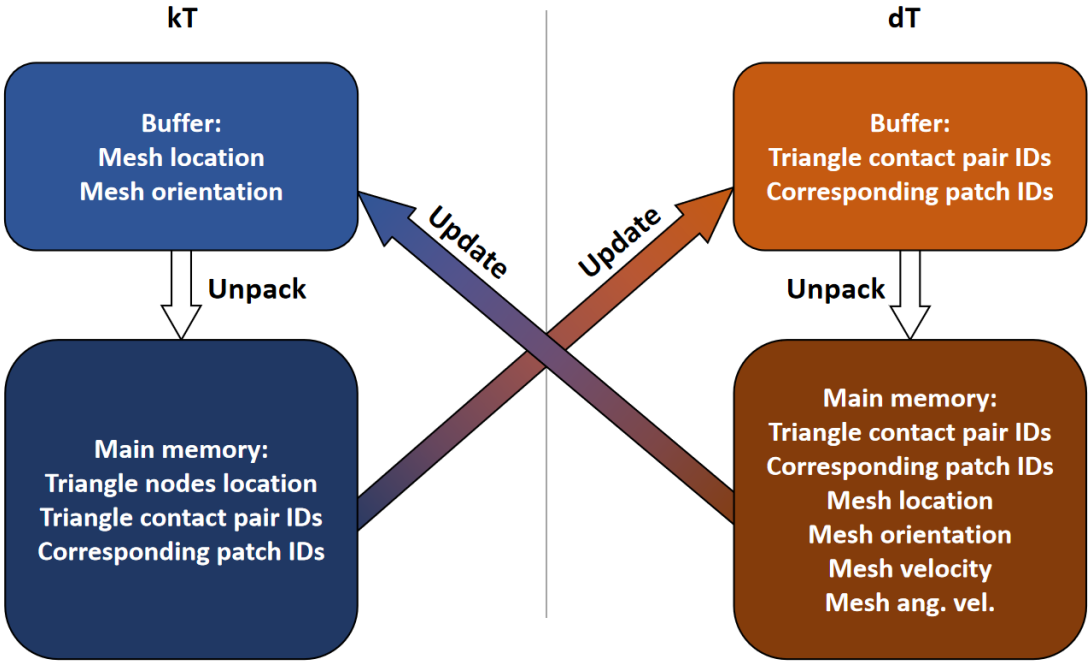


Figure 4.4: The collaboration pattern of the kinematics and dynamics thread, where neither of them directly modifies the working memory of the other.

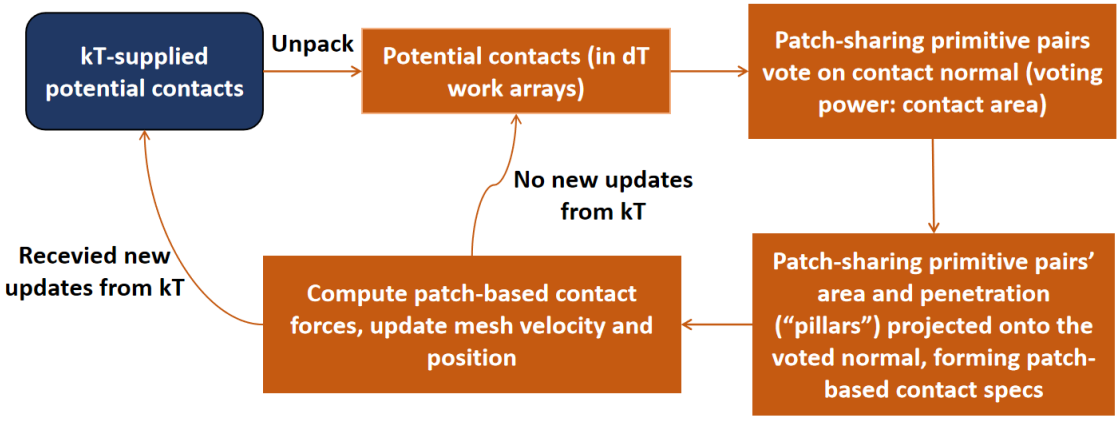


Figure 4.5: The workflow of the dynamics thread in a time step.

1. The solver computes the contribution of each triangle contact pair, specifically the penetration depth, contact point, area, and normal direction. We call these contacts the primitive-based contacts, because triangles are the "primitives" that form patches. This is to be differentiated from the patch-based contacts introduced later. The computation is done via a projection-based method and is illustrated in Figure 4.6. For a potentially in-contact triangle pair, one triangle is projected onto the other's plane, then the projection is clipped against the projectee, forming a polygon shown in purple. The polygon's area is then the contact area, the projection direction becomes the contact normal, and the maximum projection distance becomes the penetration depth. Note that the result is in general different when swapping the projector and the projectee, so we select the one that gives the smaller penetration depth. Finally, the contact point is selected to be the estimated center of the projection "pillar", i.e., the volume swept by the projector triangle until it reaches the projectee's plane. In this way, the primitive contacts are resolved using only the triangles in question, and never require adjacent triangles or mesh topology information, greatly reducing the per-thread data bandwidth.

2. The solver determines the patch-based contact information through a reduction process, i.e., by merging the contributions of the triangles that belong to the same patch. This consolidation step is necessary to ensure stable and traceable contacts for the Mindlin friction model (introduced in Sec. 4.2.4), which can only be enforced consistently at the patch level.

   The first step of this reduction process is to determine the patch-based contact normal using a voting scheme, where each primitive contact contributes with a weight proportional to its contact area. This concept is illustrated in the left part of Figure 4.7. Specifically, the patch-based contact normal $\mathbf{n}_{\text{patch}}$ is computed from its c associated

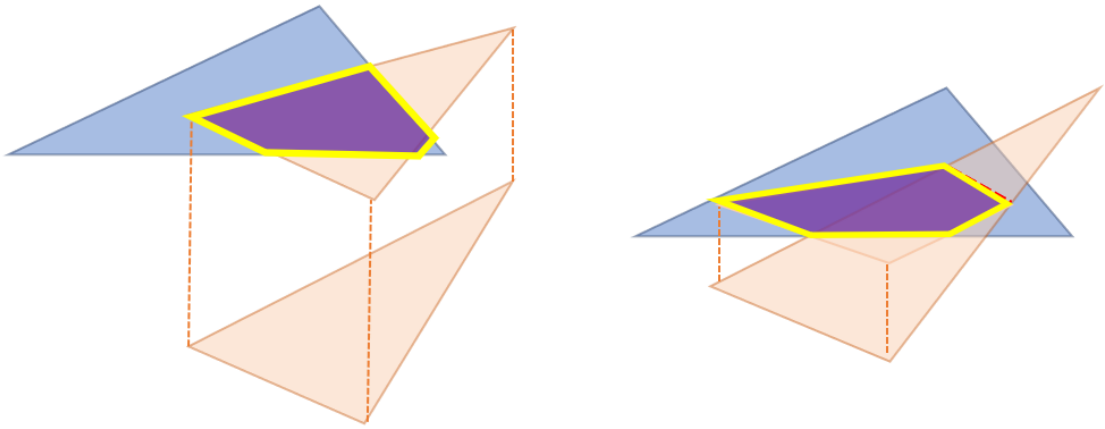


Figure 4.6: The primitive contact's penetration depth, contact point, area, and normal direction are derived by projecting a triangle onto the other's plane, then clipping against the latter. This represents the "contribution" of the primitive pair in the patch-based contact pair to which it belongs. Note this is done when one triangle is completely submerged in the other's owner mesh (shown on the left), as well as when the two triangles are in physical contact (shown on the right).

primitive contacts as

$$\mathbf{n}_{\text{patch}} = \frac{\sum_{k=1}^{c} A_{\text{prim}}^{k} \mathbf{n}_{\text{prim}}^{k}}{\left| \sum_{k=1}^{c} A_{\text{prim}}^{k} \mathbf{n}_{\text{prim}}^{k} \right|}, \tag{4.9}$$

where A_{prim}^{k} denotes the area associated with a primitive contact. In this way, the resulting contact normal reflects the overall orientation of the contacting surface region, providing a physically consistent representation of the patch interaction.

3. The next step of the reduction process is to project the penetration depth, contact point, area, and normal direction of each primitive pair onto the voted normal direction, and then combine these projected quantities to obtain their patch-level counterparts. This procedure is also illustrated in Figure 4.7. Conceptually, each primitive contact contributes to the patch-based contact through a pillar-like projection of its influence. Specifically, the patch-based contact area is obtained as the sum of the projected primitive contact areas, the penetration depth is taken as the maximum projected penetration among the primitives, and the contact point and normal direction are

computed as weighted averages of their projected primitive counterparts, with the weight proportional to the volume of each pillar.

This algorithm has the advantage that all reduction operations can be performed using efficient CUDA device-level primitives, without requiring atomic operations. Together with the absence of neighbor searches or on-the-fly mesh topology queries, this design leverages the highly parallel nature of GPU hardware.

We emphasize that the result of the reduction constitutes the patch-based contact information, i.e., the effective contact used by the force model described in Sec. 4.2.4. At the same time, the primitive-level contact data are retained and remain available for user queries. Although not required in the present study, such fine-grained information can be useful in applications involving tearing, wear, and material fatigue.

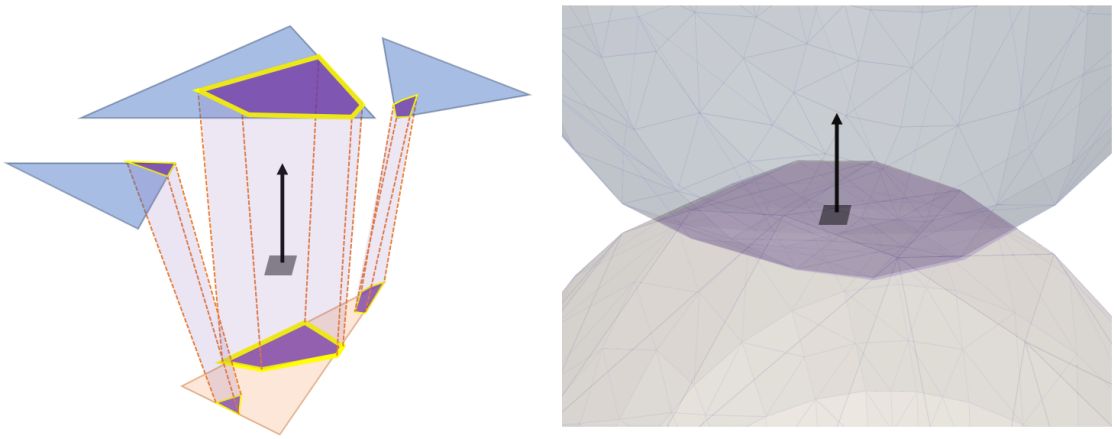


Figure 4.7: Left: For a patch-based contact — where a patch is a connected island of triangles involved in a contact — the solver pools the contact normals of all involved primitive contacts and decides an overall normal direction (shown with black arrow) using a voting process. Then each contact "pillar" (the volume swept by the projection of triangles, shown with purple) is projected onto this overall direction to calculate its contribution to this patch-based contact. Right: The summed contribution of all primitive contact pillars approximately recovers the physical contact volume, which is marked with purple.

This algorithm is shown to produce accurate and stable contact physics in the unit tests documented in Sec. 6.3.

4.2.4 Contact Force Models

Contact interactions are described using a Hertz–Mindlin-type nonlinear spring–dashpot formulation with Coulomb friction and rolling resistance. For two contacting bodies i and j, the normal and tangential forces are expressed as

$$\mathbf{F}_n = \sqrt{A/\pi}\,(k_n \mathbf{d}_n - \gamma_n \bar{m} \mathbf{v}_n)\,, \tag{4.10a}$$

$$\mathbf{F}_t = \sqrt{A/\pi}\,(-k_t \mathbf{d}_t - \gamma_t \bar{m} \mathbf{v}_t)\,, \quad \|\mathbf{F}_t\| \leq \mu \|\mathbf{F}_n\|, \tag{4.10b}$$

$$\bar{m} = m_i m_j / (m_i + m_j), \tag{4.10c}$$

where $\mathbf{d}_n$ and $\mathbf{d}_t$ are the normal and tangential displacement, A is the contact area, $\bar{m}$ is the effective mass, and $\mathbf{v}_n$, $\mathbf{v}_t$ are the normal and tangential components of the relative contact velocity. The stiffness and damping parameters $k_n, k_t, \gamma_n, \gamma_t$ are derived from material properties (Young's modulus, Poisson's ratio, and restitution coefficient) following [108]. Tangential displacement evolves incrementally during contact and is limited to satisfy the Coulomb friction criterion controlled by the coefficient of friction μ.

Rolling resistance is incorporated via an additional torque $\boldsymbol{\tau}$ proportional to the normal force magnitude, accounting for contact-level energy dissipation during rolling [109].

Contact forces are evaluated at the patch-based contact level (as opposed to primitive-level). Recall that a mesh could have multiple disconnected contact "islands" and thus multiple patch-based contacts, so all n_c contacts associated with a specific mesh are accumulated to the mesh's center of mass. The translational and rotational dynamics of body i follow

$$m_i \dot{\mathbf{v}}_i = m_i \mathbf{g} + \sum_{k=1}^{n_c} \mathbf{F}^k, \tag{4.11a}$$

$$I_i \dot{\boldsymbol{\omega}}_i = \sum_{k=1}^{n_c} \left(\mathbf{r}^k \times \mathbf{F}^k + \boldsymbol{\tau}_r^k \right), \tag{4.11b}$$

where $\mathbf{F}^k = \mathbf{F}_n^k + \mathbf{F}_t^k$, and $\mathbf{r}$ is the vector from the mesh's center of mass to the contact point.

5 NUMERICAL SOLVERS AND SOFTWARE IMPLEMENTATION ASPECTS

5.1 Background and Literature Review

Large-deformation solid mechanics is classically posed in the *Total Lagrangian* (TL) frame, where all kinematics and numerical integration are referenced to the undeformed configuration. This choice fixes shape-function gradients and geometric metrics in time, which both reduces assembly cost and improves numerical stability through reusability of reference-space data. Historically, nonlinear TL finite element analyses (FEA) have relied on Newton and quasi-Newton methods, in which a tangent stiffness is assembled and factorized each iteration, yielding quadratic local convergence when the model is well-scaled and contact conditions are benign [12]. As problems grew in size, nonlinearity, and contact complexity—and as GPU and distributed-memory hardware became the default platform—these second-order routes revealed two systemic drawbacks: repeated global factorizations with high synchronization cost, and fragile behavior in the presence of non-smooth constraints. This has motivated a sustained pivot toward *first-order* strategies centered on Augmented Lagrangian (AL) formulations and operator-splitting schemes such as the Alternating Direction Method of Multipliers (ADMM), which deliberately trade per-iteration mathematical sophistication for robustness, parallelism, and predictable memory access patterns well aligned with TL discretizations.

In the TL setting, a typical quasi-static step seeks nodal displacements u minimizing a hyperelastic energy—expressed via the deformation gradient $F(u)$ computed from reference-space derivatives—minus external work, while enforcing a portfolio of constraints: nonpenetration and friction at potential contact interfaces, incompressibility or other material constraints, and sometimes multiphysics couplings (e.g., electro-elasticity). Augmented Lagrangian methods introduce quadratic penalization while keeping explicit Lagrange multipliers, thus regularizing

constraint enforcement without collapsing to a pure-penalty method. ADMM and related primal–dual splittings exploit the problem's natural separability: element-wise elastic terms admit *local* proximal updates; contact and inequality constraints admit *local* projections (onto gap sets, friction cones, box bounds, etc.); these are coupled by a *global* update that solves a linear system with an operator of the form $(M + \rho A^\top A)$, where M is a mass-like or regularization operator, A encodes constraints, and ρ is the augmented penalty. TL helps twice: element Jacobians and quadrature data are time-invariant across inner iterations, so local kernels are cheap and cache-friendly; and the global operator changes slowly across iterations and even across time steps, enabling matrix-free Krylov solves with recycled preconditioning.

A coherent body of FE contact literature underpins this approach. In electro-elastostatics with unilateral and frictional contact, Essoufi, Koko, and Zafrar formulate an ADMM solver that alternates field updates with local contact projections and dual ascent, demonstrating robust convergence in a coupled TL multiphysics setting without repeated stiffness factorizations [110]. A follow-on Jacobi-type variant (J-ADMM) emphasizes parallel scalability by updating contact patches independently, a natural fit for distributed and GPU scenarios [111]. Chorfi and Koko address frictionless contact with an ADMM scheme that places particular emphasis on *automatic* selection and updating of the augmented penalty, balancing primal and dual residuals—crucial for conditioning and iteration counts when stiffnesses vary widely across a TL mesh [112]. At a more foundational level, Burman, Hansbo, and Larson give a rigorous AL finite element treatment of contact that establishes stability and a priori error estimates; viewed algorithmically, this work legitimizes AL as a principled gateway through which first-order inner solvers (prox/projection steps plus a global linear solve) can be plugged into standard FE spaces [113]. More recently, hybridized AL formulations for friction-free contact decouple bulk fields from interface unknowns, shrinking the global coupling to an interface layer and making domain-wise parallelism straightforward; the construction lives naturally in the reference configuration and thus dovetails with TL precomputation [114]. In nonsmooth dynamics and flexible multibody systems—often coupled

to FE bodies—Tasora, Mazhar, and Negrut show that ADMM for variational inequalities and cone complementarity problems benefits from diagonal or block-Jacobi scalings inspired by element/contact stiffness surrogates, simple matrix-free preconditioners for the global Krylov step, and modest over-relaxation; these practical levers transfer directly to TL solids with contact [115]. Complementary evidence from deformable-body simulation in graphics demonstrates the same algorithmic core—local elastic projections, contact projections, and a fixed-form global step—achieving strong performance for frictional soft-body dynamics via ADMM; the implementation heuristics (residual balancing, over-relaxation, memory layout) map closely onto TL FEA codes [116]. Even when the constitutive palette and validation targets differ from engineering practice, the agreement on solver structure is notable.

Momentum-based acceleration has begun to permeate mechanics workflows, with the clearest demonstrations so far in FE-based design and topology optimization rather than direct equilibrium solves. Nesterov-type inertial updates have been used to accelerate projected-gradient schemes on large FE meshes, with careful step-size safeguards and diagonal preconditioning to cope with operator spectra induced by discretization [117]. A complementary line derives Nesterov-accelerated level-set PDEs for structural topology optimization, connecting the discrete method to a damped-hyperbolic continuous-time limit and articulating restart rules that damp oscillations associated with high-frequency FE modes [118]. In multibody dynamics with frictional contact, Mazhar, Heyn, Negrut, and Tasora show that accelerated projected gradient (in the spirit of Nesterov) can dramatically reduce wall-clock time versus Gauss–Seidel-type iterations, while improving parallel scalability on large problems [119]. From the perspective of AL/ADMM for TL solids, the emerging consensus is pragmatic: maintain nonsmooth pieces (contact, bounds) as proximal projections; layer momentum on smooth substeps (e.g., elastic or design variables) with backtracking or residual-based restarts; and couple with diagonal/block preconditioners keyed to element stiffness scales. Recent analyses of *inertial ADMM* for nonconvex settings further support the compatibility of mild momentum with augmented Lagrangian splitting, provided penalty and inertial parameters

obey simple safeguards, which is encouraging for nonlinear TL hyperelasticity with contact [120].

By contrast, explicit reports of *AdamW* as the inner solver for TL ALM in archival mechanics journals remain scarce, but there is credible FE-adjacent evidence that Adam-like adaptive, per-coordinate scaling pairs well with augmented Lagrangians in large optimization loops that embed mechanics. Yuhn, Sato, and collaborators propose a 4D (space–time) topology optimization framework for soft bodies and actuations and state plainly that they "solve this problem using the Adam optimizer with the augmented Lagrangian method," reporting robust performance across dynamic designs with large deformations and contact [121]. Although this is an optimization-over-dynamics setting rather than a pure equilibrium solve, the template is instructive for TL-ALM inner loops: Adam/AdamW's second-moment accumulation acts as an inexpensive diagonal preconditioner, potentially taming mesh anisotropy and multiphysics scale disparities; decoupled weight decay can be interpreted as physics-aware Tikhonov regularization; and warm-starting the accumulator with element-stiffness proxies can reduce the "preconditioner burn-in." In practice, the safest pattern today is to retain AL/ADMM as the backbone for constraint handling and global coupling, keep the global step as a matrix-free PCG with lightweight block-diagonal preconditioning, and experiment with Nesterov or AdamW *inside* smooth subproblems (elastic updates, design variables) with residual monitoring and occasional restarts. This preserves the robustness of projection-based handling of contact while granting some of the adaptivity and acceleration that modern first-order methods can deliver.

5.2 Augmented Total Lagrangian Formulation

Start by assuming the position-level kinematic constraint equations read:

$$c_i(\mathbf{r}_{n+1}, \mathbf{A}_{n+1}, t_{n+1}) = 0 \ , \quad i = 1, 2, \ldots, m \ , \tag{5.1}$$

and they combine to form the vector of constraints $\mathbf{c}(\mathbf{r}_{n+1}, \mathbf{A}_{n+1}, t_{n+1}) = \mathbf{0}$. For compactness, define $c_i^{n+1} \coloneqq c_i(\mathbf{r}_{n+1}, \mathbf{A}_{n+1}, t_{n+1})$. ALM starts with the Lagrangian function:

$$\mathcal{L}_{\boldsymbol{\rho}}(\dot{\mathbf{v}}_{n+1}, \bar{\boldsymbol{\omega}}_{n+1}, \boldsymbol{\lambda}) = \ell(\dot{\mathbf{v}}_{n+1}, \bar{\boldsymbol{\omega}}_{n+1}) + \sum_{i=1}^{m} \left[\lambda_i c_i^{n+1} + \frac{\rho_i}{2} \left(c_i^{n+1} \right)^2 \right] , \tag{5.2}$$

where ρ_i are penalty parameters that control the weight of the quadratic penalty term, and λ_i are the Lagrange multipliers, i.e., the dual variables. The constraint functions $c_i(\mathbf{r}_{n+1}, \mathbf{A}_{n+1}, t_{n+1})$ are evaluated at the position level and depend implicitly on the optimization variables $\dot{\mathbf{v}}_{n+1}$ and $\bar{\boldsymbol{\omega}}_{n+1}$ through the integration scheme used to compute the updated position $\mathbf{r}_{n+1}$ and orientation $\mathbf{A}_{n+1}$.

The optimization approach, either first-order or second-order, can be used to minimize the cost functions in Eqs. 5.2 and 5.3. For the velocity-level approach to solving the constrained multibody dynamics problem, see Eq. 5.3, the constraints are abstracted as

$$\mathbf{c}(\mathbf{x}) = \mathbf{0} , \quad \text{where } \mathbf{x} = \begin{bmatrix} \\ \dot{\mathbf{v}}_{n+1} \\ \\ \\ \bar{\boldsymbol{\omega}}_{n+1} \\ \\ \end{bmatrix} .$$

The augmented Lagrangian for this problem is

$$\mathcal{L}_{\boldsymbol{\rho}}(\mathbf{x}, \boldsymbol{\lambda}) = \ell(\mathbf{x}) + \sum_{i=1}^{m} \left[\lambda_i \tilde{\nu}_i(\mathbf{x}) + \frac{\rho_i}{2} \tilde{\nu}_i(\mathbf{x})^2 \right] , \tag{5.3}$$

where $\ell(\mathbf{x})$ is the cost function, λ_i are the Lagrange multipliers, and $\rho_i > 0$ are penalty weights associated with each constraint.

In the proposed framework for solving the Total Lagrangian finite element analysis for multibody dynamics, we derive the equations of motion from the principle of virtual work.

The total virtual work is written as

$$\delta W = \delta W_{\text{inertia}} + \delta W_{\text{applied}} + \delta W_{\text{force-field}} + \delta W_{\text{internal}} \ , \tag{5.4}$$

where $\delta W_{\text{inertia}}$, $\delta W_{\text{applied}}$, $\delta W_{\text{force-field}}$, and $\delta W_{\text{internal}}$ denote the virtual works of the inertial terms, applied forces, mass-distributed force fields, and internal forces, respectively. Imposing $\delta W = 0$ and factoring out the virtual generalized displacements yields an expression of the form

$$\sum_{i=1}^{n_u} \delta \mathbf{q}_i^T \, \mathbf{g}_i = 0 \ , \tag{5.5}$$

where $\delta \mathbf{q}_i$ is an arbitrary virtual displacement associated with the ith nodal unknown and $\mathbf{g}_i$ is the corresponding residual contribution. Since $\delta \mathbf{q}_i$ are arbitrary, Eq. (5.5) implies $\mathbf{g}_i = \mathbf{0}$ for all i, which yields the discrete equations of motion.

In the following sections, we derive the expressions for all contribution terms that make up Eq. (5.4). The inertial term, mass-distributed force fields, internal forces, and applied forces are treated through the corresponding residual contributions that enter the final discretized system.

With the update $\mathbf{q}_{n+1} = \mathbf{q}_n + h\,\mathbf{v}_{n+1}$, the residual can be written as

$$\begin{aligned} \mathbf{g}(\mathbf{v}, \boldsymbol{\lambda}) = \frac{1}{h}\mathbf{M}(\mathbf{v} - \mathbf{v}_n) + \mathbf{f}_{\text{int}}(\mathbf{q}_n + h\mathbf{v}, \mathbf{v}) - \mathbf{f}_{\text{ext}} - \mathbf{f}_{\text{ff}} \\ + h\,\mathbf{C}_q(\mathbf{q}_n + h\mathbf{v})^T \Big(\boldsymbol{\lambda} + \rho\,\mathbf{c}(\mathbf{q}_n + h\mathbf{v}) \Big) \ , \end{aligned} \tag{5.6}$$

where $\mathbf{M}$ is a preassembled and constant mass matrix of the dynamic system, ρ is the ALM penalty parameter, and $\mathbf{c}(\mathbf{q}) = 0$ denotes bilateral constraints with constraint Jacobian $\mathbf{C}_q = \partial \mathbf{c}/\partial \mathbf{q}$. Note that $\mathbf{c}(\mathbf{q}) \in \mathbb{R}^m$ is a constraint-space quantity and is mapped to the generalized space through $\mathbf{C}_q(\cdot)^T$ in Eq. (5.6). We evaluate $\mathbf{C}_q^T(\cdot)$ via a sparse representation of $\mathbf{C}_q^T$, and enforce constraints with an augmented-Lagrangian outer loop that alternates an inner solve for $\mathbf{v}$ with multiplier updates until $\|\mathbf{c}\|_2$ meets a prescribed tolerance.

In Eq. (5.6), $\mathbf{f}_{\text{int}}(\mathbf{q}, \mathbf{v})$ denotes the assembled internal force vector induced by material

deformation, including elastic and viscous contributions, while $\mathbf{f}_{\text{ext}}$ collects applied loads specified directly as external nodal forces, including point loads and contact/friction forces treated as external. The term $\mathbf{f}_{\text{ff}}$ accounts for mass-distributed force fields, or body forces, specified per unit mass, such as gravity; although it can be absorbed into $\mathbf{f}_{\text{ext}}$, we keep it separate to emphasize its origin in the force-field virtual work component in Eq. (5.4).

The residual form in Eq. (5.6) can be interpreted as the first-order optimality condition of a scalar augmented objective in the velocity unknown. Introduce the step map $\mathbf{q}(\mathbf{v}) \coloneqq \mathbf{q}_n + h\,\mathbf{v}$, and assume that the internal force operator admits an incremental potential $\Pi_{\text{int}}(\mathbf{q}, \mathbf{v})$ such that

$$\nabla_{\mathbf{v}}\Big(\frac{1}{h}\Pi_{\text{int}}(\mathbf{q}(\mathbf{v}), \mathbf{v})\Big) = \mathbf{f}_{\text{int}}(\mathbf{q}(\mathbf{v}), \mathbf{v}).$$

Then the residual $\mathbf{g}(\mathbf{v}, \boldsymbol{\lambda})$ is obtained by differentiating the discrete augmented Lagrangian cost

$$\begin{aligned}\Phi_\rho(\mathbf{v}, \boldsymbol{\lambda}) = \frac{1}{2h}\,(\mathbf{v} - \mathbf{v}_n)^T\mathbf{M}(\mathbf{v} - \mathbf{v}_n) \\ + \frac{1}{h}\,\Pi_{\text{int}}(\mathbf{q}_n + h\mathbf{v}, \mathbf{v}) - \mathbf{f}_{\text{ext}}^T\mathbf{v} - \mathbf{f}_{\text{ff}}^T\mathbf{v} \\ + \boldsymbol{\lambda}^T\mathbf{c}(\mathbf{q}_n + h\mathbf{v}) \\ + \frac{\rho}{2}\,\big\|\mathbf{c}(\mathbf{q}_n + h\mathbf{v})\big\|_2^2.\end{aligned} \tag{5.7}$$

Using $\nabla_{\mathbf{v}}\mathbf{c}(\mathbf{q}_n + h\mathbf{v}) = h\,\mathbf{C}_q(\mathbf{q}_n + h\mathbf{v})$ and the chain rule, one verifies that $\nabla_{\mathbf{v}}\Phi_\rho(\mathbf{v}, \boldsymbol{\lambda}) = \mathbf{g}(\mathbf{v}, \boldsymbol{\lambda})$. Consequently, for fixed multipliers $\boldsymbol{\lambda}$, the inner solve $\mathbf{g}(\mathbf{v}, \boldsymbol{\lambda}) = \mathbf{0}$ is equivalent to enforcing stationarity of the augmented cost (5.7) with respect to $\mathbf{v}$, while the outer ALM loop updates $\boldsymbol{\lambda}$ to reduce the constraint violation $\|\mathbf{c}\|_2$.

5.3 Precomputation of Constant Terms

Before time integration, we precompute and cache in device memory all quantities that depend only on the reference configuration, the element interpolation, and the chosen quadrature rule. The following terms are precomputed once and reused throughout the simulation:

- $s_i(\mathbf{u}_q)$ **(shape-function values at quadrature points).** Scalar shape functions evaluated at quadrature nodes $\mathbf{u}_q$ in the parent domain. These values enter inertia terms through products $s_i(\mathbf{u}_q)s_j(\mathbf{u}_q)$ and force-field terms through $s_i(\mathbf{u}_q)$, avoiding repeated evaluation of basis polynomials (or barycentric forms) at every time step.

- $\mathbf{H}(\mathbf{u}_q) = \partial s/\partial \mathbf{u}$ and $\mathbf{h}_i^\top(\mathbf{u}_q) = \partial s_i/\partial \mathbf{u}$ **(reference shape-function gradients).** Reference gradients of the interpolation evaluated at $\mathbf{u}_q$. In the Total Lagrangian setting, these operators are purely geometric and constant. They are reused to assemble both the deformation gradient and its rate via $F(\mathbf{u}_q, t) = N(t)H(\mathbf{u}_q)$ and $\dot{F}(\mathbf{u}_q, t) = \dot{N}(t)H(\mathbf{u}_q)$; consequently, the same precomputed gradients are shared by hyperelasticity and the finite-strain Kelvin–Voigt damping model.

- $\{(\mathbf{u}_q, W_q)\}_{q=1}^{n_{qp}}$ (or $\{((\xi_q, \eta_q, \zeta_q), W_q)\}_{q=1}^{n_{qp}}$ for $T10$) **(quadrature nodes and weights).** Quadrature rules are tabulated once per element type and integration order. The cached nodes and weights are reused across mass, internal-force, and damping integrations, eliminating repeated construction of quadrature tables.

- $J_q = \det(\partial X/\partial \boldsymbol{\xi})|_q$ (and, when needed, $(\partial X/\partial \boldsymbol{\xi})^{-1}|_q$) **(geometric Jacobian factors for isoparametric solids).** For isoparametric solid elements integrated on the parent domain $\hat{\Omega}$ (e.g., $T10$), the reference mapping $X(\boldsymbol{\xi}) = \sum_a N_a(\boldsymbol{\xi})X_a$ depends only on reference nodal positions X_a. Therefore the Jacobian determinant J_q (and optionally the inverse Jacobian) is constant and can be precomputed at each quadrature point.

- $m_{ij} = \int_{V_r} \rho_r(\mathbf{u})\, s_i(\mathbf{u})\, s_j(\mathbf{u})\, dV_r$ and $M_e = [m_{ij} I_3]$ **(constant inertia coefficients and element mass matrix).** In the Total Lagrangian formulation, ρ_r and s_i live in the reference configuration, hence m_{ij} is constant and is computed once. The resulting consistent element mass matrix M_e (and the assembled global mass matrix) is constant and reused throughout the simulation.

5.4 The Sparse Assembly

In our implementation, the *structure* (row offsets and column indices) of all global sparse operators is treated as time-invariant as long as the mesh topology and the set of active constraints remain unchanged. For the consistent mass matrix, we construct and cache a coefficient-level sparsity pattern from element connectivity; subsequent assemblies update only the numerical values while preserving the same `offsets` and `columns` arrays. The constraint Jacobian $\mathbf{C}_q$ is likewise stored in CSR form with a fixed pattern: for clamped Dirichlet constraints, each constraint row couples to a single generalized degree of freedom (identity-like rows), whereas for general linear constraints the nonzero structure is fully determined by the user-provided CSR description of $\mathbf{C}_q$ and therefore remains constant throughout the simulation. Finally, the Newton system matrix $\mathbf{H}$ is stored in a solver-owned CSR structure whose pattern is constructed once by combining element-induced couplings (obtained by lifting the coefficient-level adjacency implied by element connectivity into generalized-coordinate space) with any additional couplings implied by the nonzero structure of $\mathbf{C}_q$ (to support the $\mathbf{C}_q^\top \mathbf{C}_q$ penalty contribution without structural fill-in). With this fixed-pattern design, each Newton iteration updates only CSR *values* (mass, tangent, and constraint contributions), while the symbolic analysis in the sparse direct solver can be performed once and reused across iterations; numerical factorization then proceeds as repeated refactorization on the same CSR pattern.

All sparse operators are stored using the compressed sparse row (CSR) format, which represents a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ by three arrays: (i) `offsets` (row pointers) of length $m{+}1$, (ii) `columns` (column indices) of length nnz, and (iii) `values` of length nnz, where the nonzeros in row r occupy the contiguous range `offsets[`r`] : offsets[`$r{+}1$`]` in `columns` and `values` [122, 123]. This layout is standard in large-scale scientific simulation software because it provides compact storage and supports efficient sparse matrix–vector products and sparse factorizations [124, 125, 126].

To form the coefficient-level sparsity, we enumerate all element-local index pairs (i, j) that

can contribute to the same global row/column, map them to global indices, and encode each pair as a 64-bit key. These keys are sorted on the device and duplicates are removed, yielding the unique set of column indices per row. A row-count pass followed by an exclusive scan produces the CSR row offsets. This procedure is used both to allocate the time-invariant mass-matrix structure and to provide a compact adjacency graph that is subsequently lifted to the generalized-coordinate space when constructing the Newton-matrix sparsity pattern.

5.5 The Parallelization of Internal Force Evaluation

In the Total Lagrangian formulation the evaluation of the global internal force vector $\mathbf{f}^{\text{int}}$ consists of two conceptually distinct phases: the pointwise evaluation of the first Piola–Kirchhoff stress tensor $\mathbf{P}$ at every quadrature point, and the subsequent assembly of element-level nodal contributions into the global force vector. Because these two phases exhibit fundamentally different data-access patterns, the former being entirely write-independent while the latter requires updates to shared global degrees of freedom, they are parallelized on the GPU using distinct strategies. We describe the two stages next.

Stage 1: Parallelization of Stress Evaluation. In Stage 1 the first Piola–Kirchhoff stress tensor $\mathbf{P}^{(e,q)}$ is evaluated independently at every pair (e, q). Because each result is stored in a dedicated buffer location indexed uniquely by (e, q), no write conflicts can occur and the computation is embarrassingly parallel. A GPU thread is assigned to each (element, quadrature point) pair, with a total of $n_{\text{el}} \times n_{\text{qp}}$ threads launched. The number of quadrature points per element n_{qp} equals 5 for the quadratic tetrahedral (T10) elements (5-point Keast rule), 12 for the ANCF beam elements ($3 \times 2 \times 2$ Gauss–Legendre product rule), and 48 for the ANCF shell elements ($4 \times 4 \times 3$ Gauss–Legendre product rule).

Each thread gathers the current generalized nodal coordinates $\{\mathbf{x}_a\}$ for element e and

assembles the deformation gradient at quadrature point q:

$$\mathbf{F}^{(e,q)} = \sum_{a=1}^{n_{\text{en}}} \mathbf{x}_a \otimes \nabla_{\mathbf{X}} N_a^{(e,q)}. \tag{5.8}$$

The constitutive model is then evaluated to obtain the elastic stress:

$$\mathbf{P}_{\text{el}}^{(e,q)} = \mathcal{P}\Big(\mathbf{F}^{(e,q)}\Big), \tag{5.9}$$

where $\mathcal{P}(\cdot)$ represents the chosen hyperelastic relation (Saint Venant–Kirchhoff or compressible Mooney–Rivlin in the present framework). When Kelvin–Voigt viscoelastic damping is active, a viscous stress contribution $\mathbf{P}_{\text{vis}}^{(e,q)}$, computed from the rate of deformation, is added to yield the total stress

$$\mathbf{P}^{(e,q)} = \mathbf{P}_{\text{el}}^{(e,q)} + \mathbf{P}_{\text{vis}}^{(e,q)}. \tag{5.10}$$

The result is stored in the dedicated buffer entry for (e, q) without requiring any inter-thread communication. The cached stress tensors are further reused in the Hessian evaluation described in Section 5.6.2, avoiding redundant constitutive calls.

Stage 2: Parallelization of Internal Force Assembly. Given the stress buffer $\{\mathbf{P}^{(e,q)}\}$ produced by Stage 1, the contribution of element e to the internal force at its local shape-function index a is

$$\mathbf{f}_a^{\text{int},(e)} = \sum_{q=1}^{n_{\text{qp}}} \mathbf{P}^{(e,q)}\, \nabla_{\mathbf{X}} N_a^{(e,q)}\, J_0^{(e,q)}\, w_q, \tag{5.11}$$

where $J_0^{(e,q)} = \det(\partial \mathbf{X}/\partial \boldsymbol{\xi})|_q$ is the determinant of the reference-configuration Jacobian at quadrature point q of element e (precomputed once and fixed in the TL frame), and w_q is the quadrature weight associated with point q. The global internal force is assembled by summing contributions from all elements sharing a given global degree of freedom I:

$$\mathbf{f}_I^{\text{int}} = \sum_{e \in \mathcal{S}(I)} \mathbf{f}_{a(e,I)}^{\text{int},(e)}, \tag{5.12}$$

where $\mathcal{S}(I)$ is the set of elements that share degree of freedom I and $a(e, I)$ is the local shape-function index of I within element e.

This assembly introduces write conflicts: threads associated with different elements may concurrently attempt to update the same entry in $\mathbf{f}^{\text{int}}$, due to multiple elements sharing the same node in the mesh topology. The thread decomposition in Stage 2 therefore differs from Stage 1. One thread is assigned to each (element, local shape-function) pair, with $n_{\text{el}} \times n_{\text{en}}$ total threads. Each thread independently evaluates the local accumulation in Eq. (5.11) by looping over all n_{qp} quadrature points of its element, and then scatters the result to the corresponding global DOF entries via GPU atomic addition:

$$\mathbf{f}^{\text{int}}[\mathcal{G}(e, a)] \mathrel{+}= \mathbf{f}_a^{\text{int},(e)}, \tag{5.13}$$

where $\mathcal{G}(e, a)$ denotes the global DOF index triplet corresponding to local shape-function a of element e. Atomic addition guarantees correctness in the presence of concurrent writes from different elements, and incurs negligible overhead because each global node is shared by only a small number of elements.

The number of shape functions per element n_{en} equals 10 for the T10 tetrahedral elements, 8 for the ANCF beam elements (four generalized coordinates per physical node times two nodes), and 16 for the ANCF shell elements (four generalized coordinates per physical node times four nodes).

Figure 5.1 provides a visualization of the two-stage GPU parallelization strategy for computing $\mathbf{P}^{(e,q)}$ and assembling $\mathbf{f}^{\text{int}}$. The visualization uses a 3-element T10 mesh as an example; for ANCF beam and ANCF shell elements, the processes are similar but with different numbers of shape functions and different numbers of quadrature points.

Admittedly, the parallelization strategy adopted for the internal force assembly is not unique. At least two alternative strategies offer potential for further acceleration. In the first, atomic operations are avoided entirely by allocating a larger global GPU memory buffer to store the nodal contributions from each element independently; a subsequent reduce-by-key

kernel then accumulates contributions across elements. For this approach to be effective, the reduction must be implemented efficiently, for instance by leveraging a GPU library such as CUB [127]. The trade-off, however, is an increased global memory footprint and the overhead associated with the reduce-by-key operation. In the second strategy, a greater degree of parallelism is achieved by launching threads on a per-node-per-quadrature-point basis, rather than looping over all quadrature points for a given nodal unknown within a single thread. Atomic additions are then used to scatter the per-node-per-quadrature-point contributions into the global internal force vector. While this approach increases the available parallelism, it also increases the number of atomic operations, which may offset the performance gains. Furthermore, hybrid strategies combining elements of the above approaches are also conceivable. The optimal choice among these strategies is likely problem- and hardware-dependent. In the present work, the proposed method was selected to balance global memory footprint, the number of atomic operations, and the overall degree of parallelism, targeting NVIDIA RTX-class GPUs.

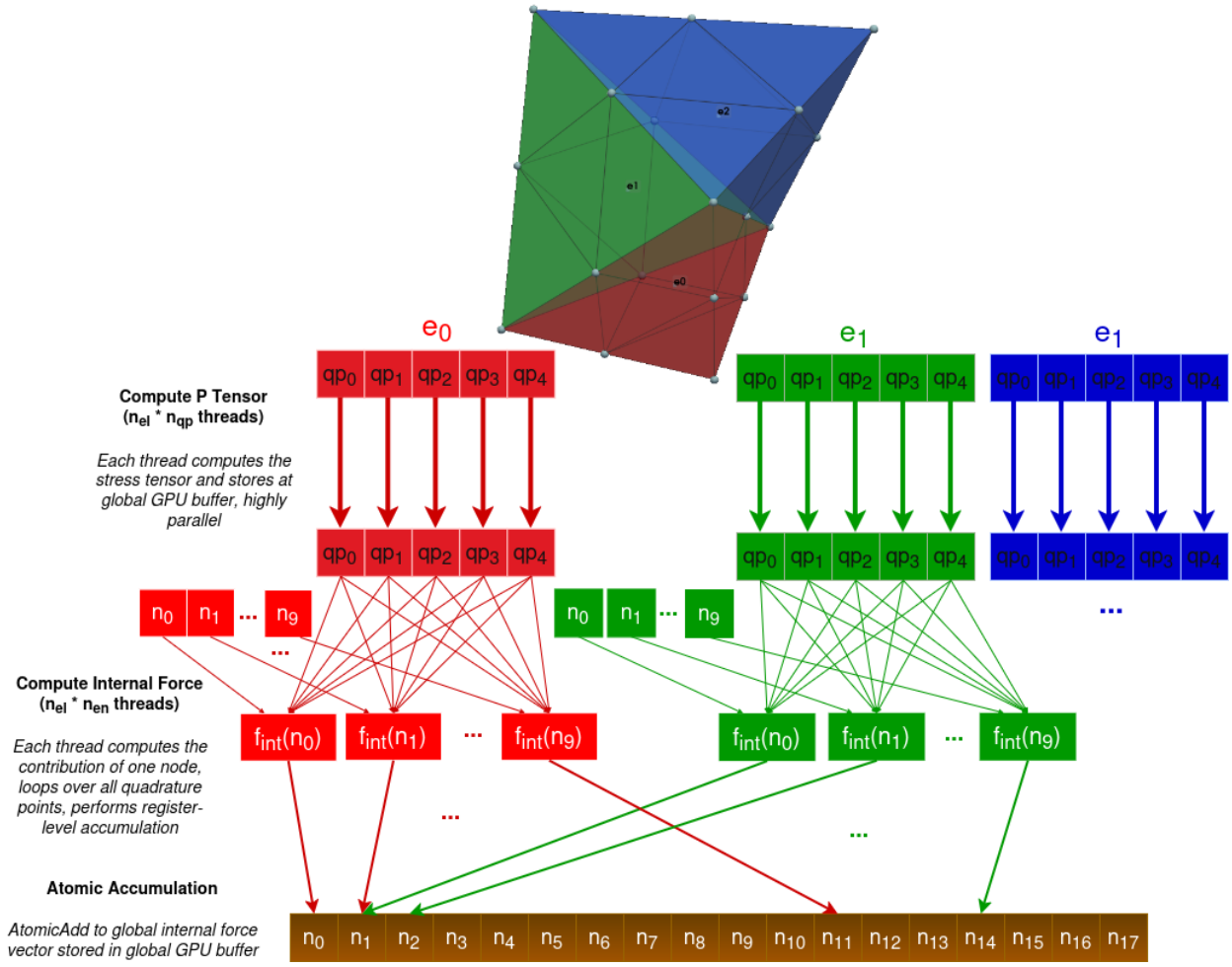


Figure 5.1: A visualization of the parallelization strategy used for the evaluation of the First Piola–Kirchhoff stress and the internal force vector. For the computation of the First Piola–Kirchhoff stress tensor $\mathbf{P}$, each thread is responsible for computing $\mathbf{P}^{(e,q)}$ for a specific element e and quadrature point q; this stage is highly parallel without write conflicts. The resulting stress tensors are stored in a dedicated continuous GPU buffer indexed by (e, q), which is then reused in the assembly of the internal force vector and the Hessian matrix. In the internal force assembly stage, each thread is responsible for computing the contribution of a specific element e and local shape-function index a to the internal force. The thread loops over all quadrature points of its element to compute the local contribution, and then scatters the result to the corresponding global DOF entries using atomic addition to ensure correctness in the presence of concurrent writes from different elements.

5.6 The Parallelization of Gradient and Hessian Evaluation

Every iteration of the time integrator requires two quantities derived from the discrete augmented Lagrangian: the gradient $\nabla_{\mathbf{v}}\mathcal{L}$, and the system Hessian $\mathbf{H} = \nabla^2_{\mathbf{v}}\mathcal{L}$, which is additionally required by second-order Newton-type methods. As discussed in Section 5.7, first-order methods such as the AdamW and Nesterov accelerated gradient solvers advance the velocity state using the gradient alone, and the substantial cost of Hessian assembly is avoided entirely. Second-order methods, most notably the Newton solver presented in Section 5.8, construct the full Hessian at every Newton iteration and solve the resulting sparse linear system for the velocity correction. The two computations are therefore described separately below.

5.6.1 Gradient Evaluation

The gradient of the discrete augmented Lagrangian with respect to the generalized velocity iterate $\mathbf{v}^{(k)}$ is a vector of length $n_{\text{dof}} = 3n_{\text{coef}}$, where n_{coef} is the total number of coefficient indices (nodes for T10 elements; physical nodes times the number of gradient DOFs per node for ANCF elements). Its entry at global DOF index (I, d), node I, spatial component d, takes the form

$$g^{(k)}_{I,d} = \frac{1}{h}\sum_{J} M_{IJ}\Big(v^{(k)}_{J,d} - v^{n}_{J,d}\Big) + f^{\text{int}}_{I,d} - f^{\text{ext}}_{I,d} + h\sum_{c}[\mathbf{C}_q^{\top}]_{(I,d),\,c}\Big(\lambda_c + \rho\, c_c\Big), \tag{5.14}$$

where h is the time-step size, M_{IJ} is the consistent mass matrix entry between coefficient indices I and J, v^n is the converged velocity from the previous time step, $\mathbf{f}^{\text{int}}$ is the internal force vector assembled in Section 5.5, $\mathbf{f}^{\text{ext}}$ is the applied external force, $\mathbf{C}_q$ is the constraint Jacobian, and λ_c and ρ are the dual variable and penalty parameter of the augmented Lagrangian, respectively.

A GPU thread is assigned to each scalar DOF entry of the gradient vector, with n_{dof} threads launched in total. Because each thread writes to a single, privately owned scalar location in the gradient vector, no write conflicts arise and no atomic operations are required.

The first term in Eq. (5.14) is a sparse matrix-vector product involving the consistent mass matrix $\mathbf{M}$, which is pre-assembled once and stored in CSR format (Section 5.4). Each thread traverses the CSR row corresponding to coefficient index I and accumulates

$$\sum_{J \in \mathcal{N}(I)} M_{IJ} \, \frac{v_{J,d}^{(k)} - v_{J,d}^{n}}{h}, \tag{5.15}$$

where $\mathcal{N}(I)$ is the set of coefficient neighbors of I in the mesh adjacency graph. No element-level loop is needed; the mass matrix is accessed directly in its globally assembled CSR form.

The contributions $f_{I,d}^{\text{int}}$ and $f_{I,d}^{\text{ext}}$ are read directly from device-resident global arrays. The internal force has already been assembled in full by the two-stage process of Section 5.5, so no additional computation is required here.

When equality constraints are present, each thread additionally reads a row of the transposed constraint Jacobian $\mathbf{C}_q^{\top}$, stored as a separate CSR array on the device. Traversing the entries of that row, the thread accumulates

$$h \sum_{c \in \mathcal{C}(I,d)} J_{c,(I,d)}^{\top} \Big(\lambda_c + \rho \, c_c \Big), \tag{5.16}$$

where $\mathcal{C}(I, d)$ is the set of constraint indices whose Jacobian row has a non-zero entry at DOF (I, d), and c_c is the constraint residual.

5.6.2 Hessian Assembly

The system Hessian required by the Newton solver takes the form

$$\mathbf{H} = \frac{1}{h}\,\mathbf{M} + h\,\mathbf{K}_t + h^2\rho\,\mathbf{C}_q^\top \mathbf{C}_q, \tag{5.17}$$

where $\mathbf{K}_t$ is the tangent stiffness matrix. The matrix $\mathbf{H}$ is symmetric positive definite and is stored in CSR format at the DOF level.

Remark 5.1 (Positive definiteness of $\mathbf{H}$)**.** *Equation* (5.17) *is a Gauss–Newton approximation of the full augmented-Lagrangian Hessian: the constraint-curvature term* $h^2 \sum_c \eta_c \nabla^2_{\mathbf{q}} c_c$ *is omitted (see companion paper [128], Appendix B). This term is indefinite for DP1-containing constraints, so its omission is what preserves the symmetric positive definite structure required by the Cholesky factorization in cuDSS. When nonlinear constraints are present, the solver attains superlinear rather than quadratic convergence near the solution.*

With the curvature term absent, $\mathbf{H}$ *is positive definite whenever* $\mathbf{M} \succ 0$ *(which holds by construction),* $h > 0$*,* $\rho > 0$*, and* $\mathbf{M} + h^2\mathbf{K}_t \succ 0$*. When* $\mathbf{K}_t \succeq 0$ *— as in the elastic regime of the SVK model — the last condition is satisfied for any* h*. When* $\mathbf{K}_t$ *is indefinite (e.g., under large compressive strains), a sufficient scalar bound is* $h < \sqrt{\lambda_{\min}(\mathbf{M}) \,/\, |\lambda_{\min}(\mathbf{K}_t)|}$*; the same step-size reductions used to improve robustness and accuracy also strengthen definiteness.*

Before the first Newton iteration, a one-time symbolic analysis determines the non-zero structure of $\mathbf{H}$. The analysis starts from the coefficient-level adjacency graph, which is derived directly from element connectivity via the mass matrix sparsity pattern (see Section 5.4). Each adjacent pair of coefficient indices (I, J) contributes a 3×3 block of non-zeros to $\mathbf{H}$. When linear CSR constraints are present, the outer product $\mathbf{C}_q^\top \mathbf{C}_q$ may couple coefficient indices that are otherwise non-adjacent in the mesh, and these additional fill-in entries are incorporated into the pattern. The DOF-level CSR arrays (row offsets and column indices) are assembled on the host using a prefix-sum scan and uploaded to device memory, where they remain fixed for the entire simulation since the mesh topology does not change.

The mass contribution $\mathbf{M}/h$ is assembled by a kernel that launches one thread per coefficient index. Each thread traverses its CSR mass row and, for every neighbor coefficient J, adds M_{IJ}/h to the diagonal elements $H(3I+d,\, 3J+d)$ for $d = 0, 1, 2$; only the diagonal components of each 3×3 block are non-zero due to the isotropic mass scaling. The CSR column position is located via binary search, and the value is deposited with `atomicAdd`.

The tangent stiffness contribution $h\mathbf{K}_t$ is assembled by a kernel that applies the same thread decomposition used in Stage 1 of Section 5.5, one thread per (element, quadrature point) pair, with $n_{\text{el}} \times n_{\text{qp}}$ threads in total.

Each thread independently performs the following computation. It gathers the current nodal positions $\{\mathbf{x}_a\}$ and the precomputed reference gradients $\nabla_{\mathbf{X}} N_a^{(e,q)}$, then re-assembles the deformation gradient $\mathbf{F}^{(e,q)}$ as in Eq. (5.8). From $\mathbf{F}^{(e,q)}$ it evaluates the fourth-order material tangent tensor $\mathcal{A}^{(e,q)}$, whose components depend on the chosen constitutive model (Saint Venant–Kirchhoff or compressible Mooney–Rivlin). For every pair of local shape-function indices (a, b), the 3×3 tangent block is then

$$[\mathbf{K}_{ab}]_{de}^{(e,q)} = \mathcal{A}_{dJeL}^{(e,q)} \, (\nabla_{\mathbf{X}} N_a)_J \, (\nabla_{\mathbf{X}} N_b)_L \, J_0^{(e,q)} \, w_q, \tag{5.18}$$

where repeated indices J and L are summed over $\{1, 2, 3\}$. The blocks are accumulated into a thread-local element stiffness matrix $\mathbf{K}_{\text{elem}}$, whose size is 30×30 for T10 elements, 24×24 for ANCF beam elements, and 48×48 for ANCF shell elements.

Once $\mathbf{K}_{\text{elem}}$ is complete, the thread scatters each entry $h[\mathbf{K}_{\text{elem}}]_{r,s}$ to its corresponding position in the global CSR array. The global row and column indices are obtained from the element-to-global node mapping; the CSR column position is located by binary search over the pre-built column index array; and the contribution is accumulated with `atomicAdd`. Concurrent writes from threads processing different elements that share a global node are correctly serialized by the atomic operation, with negligible overhead given the small number of elements per node in a well-conditioned mesh.

The constraint contribution $h^2 \rho \, \mathbf{C}_q^\top \mathbf{C}_q$ is assembled by a kernel that launches one thread

per constraint. Thread c accesses the CSR row of the constraint Jacobian $\mathbf{C}_q$ corresponding to constraint index c and loops over all pairs of DOFs $(\mathrm{dof}_i, \mathrm{dof}_j)$ whose Jacobian entries $J_{c,i}$ and $J_{c,j}$ are both non-zero. The contribution $h^2 \rho \, J_{c,i} \, J_{c,j}$ is added to $\mathbf{H}[\mathrm{dof}_i, \mathrm{dof}_j]$ via binary search and `atomicAdd`. When different constraints share common DOFs, the resulting write conflicts are resolved by the atomic operation.

5.7 First-Order Optimization Method

The velocity-level augmented Lagrangian subproblem can be solved using a first-order optimizer. Since the objective and its gradient evaluation decompose across element- and constraint-level contributions, the resulting computations admit a data-parallel GPU implementation. We employ AdamW, an adaptive gradient method that combines exponential moving averages of first and second moments with decoupled weight decay [129]. In each inner iteration (Algorithm 4), we compute the gradient of the velocity-level augmented Lagrangian with respect to the optimization variable, update the moment estimates, and apply an AdamW step with decoupled decay. Iterations terminate when the gradient norm falls below a prescribed tolerance or when the relative decrease of the objective drops below a threshold.

Algorithm 4 One Time Step: AdamW Solver

Require: Precomputed on device: $\nabla_{\mathbf{X}}\mathbf{N}^{(e,q)}$, $J_0^{(e,q)}$; $\mathbf{M}$ and $\mathbf{C}_q$ in CSR format; device arrays for $\mathbf{f}_{\text{ext}}$

$\mathbf{q} \leftarrow \mathbf{q}_n$
for $k = 1, \ldots, K_{\max}$ **do** ▷ ALM outer loop; drives constraint satisfaction
  $\mathbf{g} \leftarrow \mathbf{0},\ \mathbf{m} \leftarrow \mathbf{0},\ \mathbf{s} \leftarrow \mathbf{0}$ ▷ Reset gradient and moment accumulators
  for $l = 1, \ldots, L_{\max}$ **do** ▷ AdamW inner loop; minimizes $\mathcal{L}_\rho$ w.r.t. $\boldsymbol{v}$
    [GPU, n_v**]** $\mathbf{m} \leftarrow \beta_1\mathbf{m} + (1-\beta_1)\mathbf{g}$ ▷ First moment
    [GPU, n_v**]** $\mathbf{s} \leftarrow \beta_2\mathbf{s} + (1-\beta_2)\mathbf{g}\odot\mathbf{g}$ ▷ Second moment
    [GPU, n_v**]** $\hat{\mathbf{m}} \leftarrow \mathbf{m}/(1-\beta_1^l),\ \hat{\mathbf{s}} \leftarrow \mathbf{s}/(1-\beta_2^l)$ ▷ Bias correction
    [GPU, n_v**]** $\boldsymbol{v} \leftarrow (1-\alpha_l\lambda_{\text{wd}})\boldsymbol{v} - \alpha_l\,\hat{\mathbf{m}}/(\sqrt{\hat{\mathbf{s}}}+\varepsilon)$ ▷ Velocity update; weight decay decoupled
    [GPU, n_v**]** $\mathbf{q} \leftarrow \mathbf{q}_n + h\boldsymbol{v}$ ▷ Backward-Euler step map
    [GPU, $n_{el}\times n_{qp}$**]** Evaluate $\mathbf{F}^{(e,q)}$ and $\mathbf{P}^{(e,q)}$ ▷ PK1 stress per quadrature point
    [GPU, $n_{el}\times n_{en}$**]** Accumulate $\mathbf{f}_{\text{int}}$ atomically ▷ Internal forces computation
    [GPU, n_c**]** Evaluate $\mathbf{c}(\mathbf{q})$ ▷ Bilateral constraint residual
    [GPU, n_v**]** $\mathbf{g} \leftarrow \frac{1}{h}\mathbf{M}(\boldsymbol{v}-\boldsymbol{v}_n) + \mathbf{f}_{\text{int}} - \mathbf{f}_{\text{ext}} + h\mathbf{C}_q^\top(\boldsymbol{\lambda} + \rho\,\mathbf{c})$ ▷ Compute gradient
    Transfer $\|\mathbf{g}\|$ to host; **if** $\|\mathbf{g}\| \le \varepsilon_{\text{in}}(1+\|\boldsymbol{v}\|)$ **or** $\|\mathbf{g}\| \le \varepsilon_{\text{rel}}\|\mathbf{g}_0\|$: **break** ▷ Inner convergence
  `inner_converged` $\leftarrow$ **true** if the inner stopping criterion was met
  $\boldsymbol{v}_n \leftarrow \boldsymbol{v};\ \mathbf{q} \leftarrow \mathbf{q}_n + h\boldsymbol{v}$ ▷ Commit velocity; update position
  [GPU, n_c**]** Re-evaluate $\mathbf{c}(\mathbf{q})$
  [GPU, n_c**]** $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \rho\,\mathbf{c}(\mathbf{q})$ ▷ Dual ascent step
  Transfer $\|\mathbf{c}\|$ to host; **if** `inner_converged` **and** $\|\mathbf{c}\| \le \varepsilon_{\text{out}}$: **break** ▷ Outer convergence
return $\mathbf{q}_{n+1} \leftarrow \mathbf{q},\ \boldsymbol{v}_{n+1} \leftarrow \boldsymbol{v}_n$

5.8 Second-Order Optimization Method

The velocity-level augmented Lagrangian subproblem can also be solved using a second-order Newton method. In each inner iteration (Algorithm 5), the gradient and Hessian of the augmented Lagrangian are computed with respect to the generalized velocity, the resulting sparse linear system is solved for the Newton increment, and the iterate is updated. Unlike the first-order AdamW solver, the Newton method requires assembling and factorizing the full sparse Hessian at every iteration.

The dominant contribution to the Hessian is the tangent stiffness matrix $\mathbf{K}_t$, whose derivation for the St. Venant–Kirchhoff and Mooney–Rivlin material models is presented in Chapter 3; the Hessian assembly procedure is described in Section 5.6.2.

Once the assembled Hessian $\mathbf{H}$ is available in CSR format, each Newton iteration requires the solution of the linear system

$$\mathbf{H}^{(k)}\mathbf{p}^{(k)} = -\mathbf{g}^{(k)}, \tag{5.19}$$

for the velocity correction $\mathbf{p}^{(k)}$. The solution is obtained using cuDSS [58], NVIDIA's GPU-native sparse direct solver library, which operates entirely in device memory and is designed to exploit the data parallelism available on modern GPU architectures. The Hessian is registered with cuDSS as a symmetric positive definite matrix, with only the upper triangular part stored, allowing cuDSS to employ a Cholesky factorization $\mathbf{H} = \mathbf{L}\mathbf{L}^\top$ rather than a general LU decomposition, thereby reducing both storage and arithmetic cost. Iterative refinement is disabled in the solver configuration, as the outer Newton iterations themselves provide sufficient convergence control without the additional overhead of post-solve refinement passes.

The solution procedure is organized into three distinct phases that differ substantially in their computational cost and in the frequency with which they must be executed.

Analysis phase The analysis phase is performed once, prior to the first Newton iteration of the simulation. Given the symbolic sparsity pattern of $\mathbf{H}$, which, as established in Section 5.4, remains fixed throughout the simulation, cuDSS computes a fill-reducing column reordering, performs the symbolic factorization, and allocates the internal data structures required for numerical factorization. Because the CSR row offsets and column indices of $\mathbf{H}$ do not change across Newton iterations or time steps, the cost of this analysis is incurred exactly once and is amortized over the entire simulation run. After the analysis completes, an internal flag is reset to indicate that no numerical factorization has yet been performed, ensuring that the very first factorization call unconditionally executes the full factorization phase rather than refactorization.

Factorization phase At each Newton iteration, once the CSR values of $\mathbf{H}$ have been updated by the assembly kernels described in Section 5.6.2, cuDSS performs numerical factorization using the symbolic structure established during the analysis phase. On the first Newton iteration of the simulation, cuDSS executes a full numerical Cholesky factorization that initializes all internal factor buffers. On every subsequent Newton iteration, cuDSS instead executes a *refactorization*: this mode reuses the symbolic structure, including the fill-reducing permutation and pre-allocated factor storage, and only updates the numerical values of the triangular factor $\mathbf{L}$. Because the symbolic work, which involves graph traversal, ordering algorithms, and structural memory allocation, constitutes the dominant setup cost of a sparse direct factorization, refactorization is substantially faster than a full factorization for the same sparsity pattern. This efficiency gain is precisely what makes the fixed-sparsity strategy described in Section 5.4 valuable: the sparsity pattern is constructed once and held constant so that every Newton iteration after the first can take advantage of the cheaper refactorization path. Both factorization modes operate entirely on device memory, avoiding host–device data transfers and reusing the precomputed permutation arrays from the analysis step.

Solve phase Given the factored representation $\mathbf{H} = \mathbf{L}\mathbf{L}^\top$, cuDSS performs forward and backward triangular solves to compute the Newton step $\mathbf{p}^{(k)}$. The right-hand side vector $\mathbf{g}^{(k)}$ resides in device memory as produced by the gradient evaluation (Section 5.6.1), and the computed step is written directly to device memory for use in the velocity update of Algorithm 5.

This three-phase design, with analysis executed once per simulation, a single full factorization on the first call, and numerical refactorization repeated at every subsequent Newton iteration, aligns naturally with the fixed-pattern, repeated-refactorization strategy described in Section 5.4. The dominant per-iteration cost of the Newton method is consequently the Hessian assembly and numerical refactorization, both of which execute entirely on the GPU without intermediate host transfers.

Line search (optional) In highly nonlinear regimes such as near contact events or at large deformation increments, the full Newton step $\delta \boldsymbol{v}$ can overshoot a local minimum of the augmented Lagrangian, causing the gradient norm to increase rather than decrease. An optional Armijo backtracking line search guards against this. After computing $\delta \boldsymbol{v}$, a step size $\alpha \in (0, 1]$ is sought such that the trial gradient satisfies the sufficient decrease condition

$$\phi(\alpha) \;\leq\; (1 - 2c_1\alpha)\,\phi_0, \qquad \phi(\alpha) := \tfrac{1}{2}\|\mathbf{g}^{\mathrm{tr}}(\alpha)\|^2, \quad \phi_0 := \tfrac{1}{2}\|\mathbf{g}\|^2, \tag{5.20}$$

where $\mathbf{g}^{\mathrm{tr}}(\alpha)$ is the gradient re-evaluated at the trial point $(\boldsymbol{v} + \alpha\,\delta\boldsymbol{v},\ \mathbf{q}_n + h(\boldsymbol{v} + \alpha\,\delta\boldsymbol{v}))$. Starting from $\alpha = 1$, the step is reduced by a factor $\beta \in (0, 1)$ at each backtracking iteration until (5.20) is satisfied or a maximum of $j_{\max}$ reductions is reached; in the latter case the last trial point is accepted regardless. Typical values are $c_1 = 10^{-4}$, $\beta = 0.5$, and $j_{\max} = 10$. Each re-evaluation requires one pass through the internal force, constraint, and gradient assembly kernels, so the per-backtrack cost is comparable to a single gradient evaluation step. When the problem is well-conditioned and step sizes remain near unity, the line search terminates immediately at $j = 0$ and incurs no additional cost beyond the single trial evaluation. The complete per-step procedure, including both the optional line search and the ALM outer loop, is given in Algorithm 5.

Algorithm 5 One Time Step: Newton Solver

Require: Precomputed on device: $\nabla_{\mathbf{X}}\mathbf{N}^{(e,q)}$, $J_0^{(e,q)}$; $\mathbf{M}$, $\mathbf{C}_q$, $\mathbf{H}$ in CSR format; $\mathbf{f}_{\text{ext}}$; cuDSS handle with pre-analyzed sparsity pattern

$\mathbf{q} \leftarrow \mathbf{q}_n,\ \boldsymbol{v} \leftarrow \boldsymbol{v}_n$
for $k = 1, \ldots, K_{\max}$ **do** ▷ ALM outer loop; drives constraint satisfaction
  for $l = 1, \ldots, L_{\max}$ **do** ▷ Newton inner loop; second-order solve of $\mathcal{L}_\rho$
    [GPU, $n_{el} \times n_{qp}$**]** Evaluate $\mathbf{F}^{(e,q)}$ and $\mathbf{P}^{(e,q)}$ ▷ PK1 stress per quadrature point
    [GPU, $n_{el} \times n_{en}$**]** Accumulate $\mathbf{f}_{\text{int}}$ atomically ▷ Internal forces
    [GPU, n_c**]** Evaluate $\mathbf{c}(\mathbf{q})$ ▷ Bilateral constraint residual
    [GPU, n_v**]** $\mathbf{g} \leftarrow \frac{1}{h}\mathbf{M}(\boldsymbol{v} - \boldsymbol{v}_n) + \mathbf{f}_{\text{int}} - \mathbf{f}_{\text{ext}} + h\mathbf{C}_q^\top(\boldsymbol{\lambda} + \rho\,\mathbf{c})$ ▷ Gradient
    Transfer $\|\mathbf{g}\|$ to host; **if** $l = 1$: $\|\mathbf{g}_0\| \leftarrow \|\mathbf{g}\|$
    if $\|\mathbf{g}\| \le \max(\varepsilon_{\text{atol}}, \varepsilon_{\text{rtol}}\|\mathbf{g}_0\|)$: **break** ▷ Inner convergence (abs. or rel.)
    [GPU] $\mathbf{H} \leftarrow \frac{1}{h}\mathbf{M} + h\mathbf{K}_t + h^2\rho\,\mathbf{C}_q^\top\mathbf{C}_q$ ▷ Hessian assembly
    [cuDSS] Numeric (re)factorize $\mathbf{H}$ ▷ Reuses sparsity pattern
    [cuDSS] Solve $\mathbf{H}\,\delta\boldsymbol{v} = -\mathbf{g}$ ▷ Newton direction
    if Armijo line search enabled **then**
      $\phi_0 \leftarrow \frac{1}{2}\|\mathbf{g}\|^2;\ \alpha \leftarrow 1$
      for $j = 1, \ldots, N_{\text{bt}}$ **do** ▷ Backtracking; $\beta = 0.5,\ c_1 = 10^{-4}$
        [GPU, n_v**]** $\boldsymbol{v}^{\text{tr}} \leftarrow \boldsymbol{v} + \alpha\,\delta\boldsymbol{v};\ \mathbf{q}^{\text{tr}} \leftarrow \mathbf{q}_n + h\boldsymbol{v}^{\text{tr}}$ ▷ Trial step
        [GPU] Re-eval $\mathbf{F}, \mathbf{P}, \mathbf{f}_{\text{int}}, \mathbf{c}, \mathbf{g}^{\text{tr}};\ \phi \leftarrow \frac{1}{2}\|\mathbf{g}^{\text{tr}}\|^2$
        if $\phi \le (1 - 2c_1\alpha)\,\phi_0$ **then**
          $\boldsymbol{v} \leftarrow \boldsymbol{v}^{\text{tr}};\ \mathbf{q} \leftarrow \mathbf{q}^{\text{tr}};$ **break** ▷ Accept; exit backtracking
        $\alpha \leftarrow \beta\alpha$ ▷ Shrink step
      if no trial accepted: restore $\boldsymbol{v}$, $\mathbf{q}$; **break** ▷ Line-search failure; exit inner loop
    else
      [GPU, n_v**]** $\boldsymbol{v} \leftarrow \boldsymbol{v} + \delta\boldsymbol{v};\ \mathbf{q} \leftarrow \mathbf{q}_n + h\boldsymbol{v}$ ▷ Full Newton step
  `inner_converged` $\leftarrow$ **true** if inner stopping criterion was met
  [GPU, n_c**]** Re-evaluate $\mathbf{c}(\mathbf{q})$
  if `inner_converged` **and** $n_c > 0$ **then**
    [GPU, n_c**]** $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \rho\,\mathbf{c}(\mathbf{q})$ ▷ Dual ascent
  Transfer $\|\mathbf{c}\|$ to host; **if** `inner_converged` **and** $\|\mathbf{c}\| \le \varepsilon_{\text{out}}$: **break** ▷ Outer convergence
return $\mathbf{q}_{n+1} \leftarrow \mathbf{q},\ \boldsymbol{v}_{n+1} \leftarrow \boldsymbol{v}$

5.9 Performance Optimizations and Memory Management

Several optimization strategies are employed throughout the implementation to maximize GPU utilization and minimize memory traffic. First, we maintain persistent library handles for both cuBLAS, used for norm computations during convergence checking, and cuDSS. Creating and destroying handles incurs significant overhead that accumulates over many iterations; persistent handles eliminate this cost entirely.

Second, we preallocate all GPU memory during solver construction rather than performing dynamic allocation during the solve phase. This includes velocity vectors, gradient storage, Hessian values, and temporary workspaces. Dynamic allocation on the GPU via `cudaMalloc` involves synchronization and kernel scheduling overhead that can stall the GPU pipeline.

Third, the element traits abstraction uses `constexpr` values and compile-time dispatch via `if constexpr`, enabling the compiler to eliminate dead code paths and optimize loop bounds. For example, when compiling for 10-node tetrahedral elements, loops over nodes are unrolled with a fixed iteration count of 10, enabling register allocation optimizations that would be impossible with runtime-determined bounds.

Fourth, convergence monitoring employs cuBLAS for norm computations rather than custom reduction kernels. The `cublasDnrm2` function is highly optimized for various vector lengths and provides the gradient norm needed for convergence testing. We configure cuBLAS to use device pointers for the result, minimizing synchronization requirements.

Fifth, the kernel launch configurations use a fixed block size of 256 threads, which provides good occupancy across NVIDIA architectures while leaving sufficient registers and shared memory per thread. The number of blocks is computed to cover all work items, whether DOFs, elements, or quadrature points, depending on the kernel, with minimal excess threads.

The memory footprint of the solver scales with the number of DOFs and the Hessian sparsity. For a problem with n DOFs and nnz non-zero entries in the Hessian, the primary

allocations are: the Hessian CSR structure requiring $(n + 1 + \text{nnz})$ integers and nnz doubles; velocity and gradient vectors each requiring n doubles; and cuDSS working memory for factorization, which is automatically managed by the library. The sparsity analysis phase temporarily requires $n \times \lceil n/32 \rceil$ unsigned integers for the bitset, but this memory is freed after the pattern is extracted.

Algorithm 6 Augmented Lagrangian Method for Velocity-Level Constraints

Require: $\mathcal{L}_\rho(\boldsymbol{v}; \boldsymbol{\lambda}) = \Phi(\boldsymbol{v}) + \boldsymbol{\lambda}^\top \mathbf{c}(\mathbf{q}) + \frac{\rho}{2}\|\mathbf{c}(\mathbf{q})\|^2, \quad \mathbf{q} = \mathbf{q}_n + h\boldsymbol{v}$

Initialize $\boldsymbol{\lambda}^0 \leftarrow \mathbf{0}$, $\rho > 0$
for $k = 0, 1, \ldots, K_{\max}$ **do**
  $\boldsymbol{v}^{k+1} \leftarrow \arg\min_{\boldsymbol{v}} \mathcal{L}_\rho(\boldsymbol{v}; \boldsymbol{\lambda}^k)$ ▷ Primal subproblem; solved via AdamW or Newton
  $\mathbf{q}^{k+1} \leftarrow \mathbf{q}_n + h\boldsymbol{v}^{k+1}$ ▷ Backward-Euler step map
  $\boldsymbol{\lambda}^{k+1} \leftarrow \boldsymbol{\lambda}^k + \rho\, \mathbf{c}(\mathbf{q}^{k+1})$ ▷ Dual ascent; drives constraint satisfaction
  if $\|\mathbf{c}(\mathbf{q}^{k+1})\| \le \varepsilon_{\text{out}}$ **then break** ▷ Constraints satisfied
return $\boldsymbol{v}_{n+1} \leftarrow \boldsymbol{v}^{k+1}$, $\mathbf{q}_{n+1} \leftarrow \mathbf{q}^{k+1}$

The AdamW solver implements the primal minimization in each outer augmented-Lagrangian iteration of Algorithm 6; Algorithm 4 lists the resulting fused time-step routine. At the start of each outer iteration k, the gradient vector $\mathbf{g}$ and the first and second moment accumulators $\mathbf{m}$ and $\mathbf{s}$ are reset to zero, and the reference nodal positions $\mathbf{q}_n$ are kept resident on-device to support repeated evaluations of the backward-Euler step map. Each inner iteration l then proceeds as a sequence of GPU phases: (i) the velocity update applies the bias-corrected AdamW rule with decoupled weight decay; (ii) the updated configuration $\mathbf{q} = \mathbf{q}_n + h\boldsymbol{v}$ is computed; (iii) the first Piola–Kirchhoff stress $\mathbf{P}^{(e,q)}$ is evaluated at every quadrature point (Stage 1 of Section 5.5); (iv) the internal force $\mathbf{f}^{\text{int}}$ is assembled (Stage 2 of Section 5.5); (v) the bilateral constraint residual $\mathbf{c}(\mathbf{q})$ is evaluated; and (vi) the gradient $\mathbf{g}$ is computed. Convergence of the inner loop is assessed at a prescribed check interval by computing $\|\mathbf{g}\|_2$ on-device and transferring the scalar to the host to evaluate a combined absolute–relative stopping criterion; the inner loop terminates when $\|\mathbf{g}\| \le \varepsilon_{\text{in}}(1 + \|\boldsymbol{v}\|)$ or $\|\mathbf{g}\| \le \varepsilon_{\text{rel}}\|\mathbf{g}_0\|$, where $\mathbf{g}_0$ denotes the first gradient evaluation within the current outer iteration. After the inner loop, the committed velocity $\boldsymbol{v}^n \leftarrow \boldsymbol{v}$ and position $\mathbf{q}$ are updated,

followed by the dual ascent step $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \rho\, \mathbf{c}(\mathbf{q})$, consistent with Algorithm 6. Outer convergence requires both that the inner loop has converged and that $\|\mathbf{c}\| \leq \varepsilon_{\text{out}}$.

A key property of this implementation is that the gradient evaluation in Algorithm 4 is fully data-parallel. One GPU thread is assigned to each scalar DOF entry (I, d), with $n_{\text{dof}} = 3n_{\text{coef}}$ threads launched in total. Each thread independently evaluates $g_{I,d}^{(k)}$ from Eq. (5.14): it traverses the CSR row of the preassembled mass matrix $\mathbf{M}$ to accumulate the inertia contribution, reads $f_{I,d}^{\text{int}}$ and $f_{I,d}^{\text{ext}}$ directly from device-resident global arrays, and traverses the corresponding row of the transposed constraint Jacobian $\mathbf{C}_q^\top$ stored in CSR format to accumulate the constraint contribution. Because each thread writes exclusively to the single device-memory entry $g_{I,d}$, no write conflicts arise and no atomic operations are required. This per-thread independence extends to the AdamW velocity update in Algorithm 4: the moment estimates $\mathbf{m}$ and $\mathbf{s}$ are maintained in persistent device-side buffers of length n_{dof}, and each thread reads and overwrites only the entries at its own index, computing the bias-corrected update and writing the new velocity $v_{I,d}$ without any inter-thread communication. The entire gradient evaluation and velocity update are therefore fully parallel across all DOFs.

6 BENCHMARK AND TESTING EXPERIMENT RESULTS

6.1 Solver Performance Benchmarks

We evaluate the performance of our GPU-accelerated framework through two complementary sets of experiments. The first consists of scaling benchmarks for each of the three supported element types—the 10-node tetrahedral (T10) element, the ANCF3243 beam element, and the ANCF3443 shell element—using a canonical beam-sagging problem across a range of mesh resolutions. Each element benchmark compares our AdamW and Newton GPU solvers against a CPU reference implementation. The second set validates both performance and physical correctness on three geometrically complex meshes that are representative of shapes arising in practical engineering and computer graphics simulation scenarios. All GPU experiments were conducted on an NVIDIA GeForce RTX 5090 (32 GB VRAM) under CUDA 12.8 (driver 570.195.03) with cuDSS 0.7.1.4. CPU baselines were obtained on an Intel i7-13700KF; FEniCS [64] results use OpenMP parallelization swept over $\{1, 2, 4, 8, 16\}$ threads with the best timing reported for each resolution, and Project Chrono 8.1.0 [130] was used for ANCF element comparisons. Throughout this section, the *real-time factor* (RTF) denotes the ratio of wall-clock runtime to total simulated time; lower RTF indicates faster execution relative to real time.

Comparative evaluation requires reference implementations whose licensing permits publication of performance data. Widely used commercial finite element packages such as ANSYS explicitly prohibit the use of academic-licensed software for competitive benchmarking [131]. External comparisons are therefore restricted to two open-source frameworks selected according to element type. For the T10 tetrahedral element benchmarks, FEniCS [64] serves as the reference; as a classical finite element platform, it does not implement ANCF-based element formulations [84, 16], which are specific to the flexible multibody dynamics domain. For the ANCF beam and shell element benchmarks, Project Chrono [4, 132] serves as the reference, as it provides native ANCF element support.

For all GPU solvers, the inner loop terminates when the gradient norm $\|\mathbf{g}\|_2$ of the augmented Lagrangian falls below ε_{in}; this criterion is identical for the Newton and AdamW solvers, so ε_{in} is directly comparable across both. The outer ALM loop terminates when the constraint residual satisfies $\|\mathbf{c}\| \leq \varepsilon_{\text{out}}$. Because $\|\mathbf{g}\|_2$ is an un-normalized ℓ_2 norm summed over all degrees of freedom, it grows naturally with mesh size even at the same per-DOF accuracy; a fixed absolute threshold therefore becomes increasingly demanding at finer resolutions. At the highest resolutions (RES32 for T10 and ANCF3243; RES8 and above for ANCF3443), ε_{in} and ε_{out} are relaxed from 10^{-4} to 10^{-3}. For the Newton solver this relaxation is a deliberate runtime trade-off: the coarser tolerance remains well within the regime where tip-displacement results are indistinguishable from those at 10^{-4}. For the AdamW solver, the first-order convergence rate causes the gradient norm to plateau at large problem sizes within any practical iteration budget, making the tighter tolerance unattainable without a prohibitive increase in iteration count. The ANCF3443 shell system requires the relaxation at a coarser mesh than the solid or beam cases, reflecting the higher condition numbers typical of thin-shell discretizations. FEniCS likewise uses an un-normalized ℓ_2 residual norm via PETSc's `SNESConvergedDefault` [133], which checks $\|\mathbf{F}\|_2 \leq \varepsilon_{\text{atol}}$ with no per-DOF scaling; its tolerance is therefore subject to the same mesh-size dependence, and is not directly comparable to ε_{in} in absolute terms. Per-resolution values of ε_{in} and ε_{out} are reported in Tables 6.1, 6.5, and 6.7.

6.1.1 T10 Tetrahedral Element Scaling

We discretize a $3\,\text{m}\,(\text{L}) \times 2\,\text{m}\,(\text{W}) \times 1\,\text{m}\,(\text{H})$ beam with T10 tetrahedral elements and evaluate six mesh resolutions. One end of the beam is fully fixed via constraints, and a 5,000 N load is applied at the opposite end, distributed uniformly across all nodes on that face. The SVK material uses $E = 7.0 \times 10^8\,\text{Pa}$, $\nu = 0.33$, $\rho_0 = 2700\,\text{kg/m}^3$; each resolution is run for 50 time steps at $\Delta t = 10^{-3}\,\text{s}$ (0.05 s total). Table 6.1 summarizes the mesh statistics and per-resolution solver tolerances; Table 6.2 reports the corresponding RTFs for the SVK

material model; Table 6.4 reports the corresponding RTFs for the Mooney–Rivlin model.

Resolution	DOFs	Nodes	Elements	Constrained DOFs	ε_{in}	ε_{out}
RES0	315	105	36	45	10^{-4}	10^{-4}
RES2	1,398	466	199	135	10^{-4}	10^{-4}
RES4	6,039	2,013	936	459	10^{-4}	10^{-4}
RES8	27,150	9,050	4,567	1,683	10^{-4}	10^{-4}
RES16	118,326	39,442	20,829	6,435	10^{-4}	10^{-4}
RES32	497,310	165,770	83,432	25,155	10^{-3}	10^{-3}

Table 6.1: T10 beam mesh statistics and solver tolerances across six resolution levels. ε_{in}: inner gradient-norm stopping criterion; ε_{out}: outer ALM constraint-residual stopping criterion.

Resolution	Newton (GPU)	AdamW (GPU)	FEniCS (CPU)
RES0	3.63	12.10	5.41
RES2	6.11	11.52	13.73
RES4	12.85	28.13	54.91
RES8	38.78	38.66	293.20
RES16	170.09	173.92	2509.94
RES32	1249.69	512.01	27596.69

Table 6.2: RTF for the T10 beam-sagging benchmark with the SVK material model across six resolution levels. Newton and AdamW are run on the GPU; FEniCS on the CPU. FEniCS enforces Dirichlet conditions by direct matrix modification (strong form) and terminates on force-residual norm; its tolerance is not directly comparable to the ALM criteria $\varepsilon_{\text{in}}/\varepsilon_{\text{out}}$ used by the GPU solvers.

The Mooney–Rivlin benchmark uses material constants chosen to match the small-strain elastic response of the SVK case ($E = 7.0 \times 10^8$ Pa, $\nu = 0.33$). The equivalent Mooney–Rivlin parameters are given in Table 6.3.

Figure 6.1 shows the time-to-solution for each solver across all six resolutions. The AdamW solver achieves the best RTF at coarse-to-mid resolutions, while the Newton solver becomes competitive at RES8 and above, yielding approximately a 22× speedup over FEniCS at the largest resolution tested (RTF 1,250 vs. 27,597 at RES32).

Parameter	Value
C_{10}	$7.89 \times 10^7\,\mathrm{Pa}$
C_{01}	$5.26 \times 10^7\,\mathrm{Pa}$
κ (bulk modulus)	$1.03 \times 10^9\,\mathrm{Pa}$
ρ_0	$2700\,\mathrm{kg/m^3}$

Table 6.3: Mooney–Rivlin material parameters for the T10 beam-sagging benchmark. These constants are chosen to match the small-strain response of the SVK case ($E = 7.0 \times 10^8\,\mathrm{Pa}$, $\nu = 0.33$, $\rho_0 = 2700\,\mathrm{kg/m^3}$), allowing a direct comparison of solver performance between the two constitutive models at identical stiffness.

Resolution	Newton (GPU)	AdamW (GPU)	FEniCS (CPU)
RES0	4.02	14.04	5.64
RES2	6.66	13.28	14.10
RES4	13.23	31.93	57.50
RES8	39.40	51.14	293.91
RES16	171.23	212.66	2425.62
RES32	1255.48	619.19	30547.00

Table 6.4: RTF for the T10 beam-sagging benchmark with the Mooney–Rivlin material model across six resolution levels. Newton and AdamW are run on the GPU; FEniCS on the CPU. FEniCS enforces Dirichlet conditions by direct matrix modification (strong form) and terminates on force-residual norm; its tolerance is not directly comparable to the ALM criteria $\varepsilon_{\text{in}}/\varepsilon_{\text{out}}$ used by the GPU solvers.

6.1.2 ANCF3243 Beam Element Scaling

The beam is discretized as a one-dimensional chain of ANCF3243 elements with uniform element length $L = 0.2\,\mathrm{m}$ and a constant rectangular cross-section of width $W = 0.1\,\mathrm{m}$ and height $H = 0.1\,\mathrm{m}$, with consecutive nodes connected so that element e spans nodes e and $e+1$ along the global x axis. A cantilever constraint clamps the leftmost four nodes throughout the simulation. A concentrated vertical tip force of magnitude $F_z = 5{,}000\,\mathrm{N}$ is applied at the free end and held constant for the first 100 time steps, after which it is released and the beam evolves under inertia and elasticity alone. The SVK material parameters are the same as the T10 benchmark: $E = 7.0 \times 10^8\,\mathrm{Pa}$, $\nu = 0.33$, $\rho_0 = 2700\,\mathrm{kg/m^3}$; the time step is $\Delta t = 10^{-3}\,\mathrm{s}$.

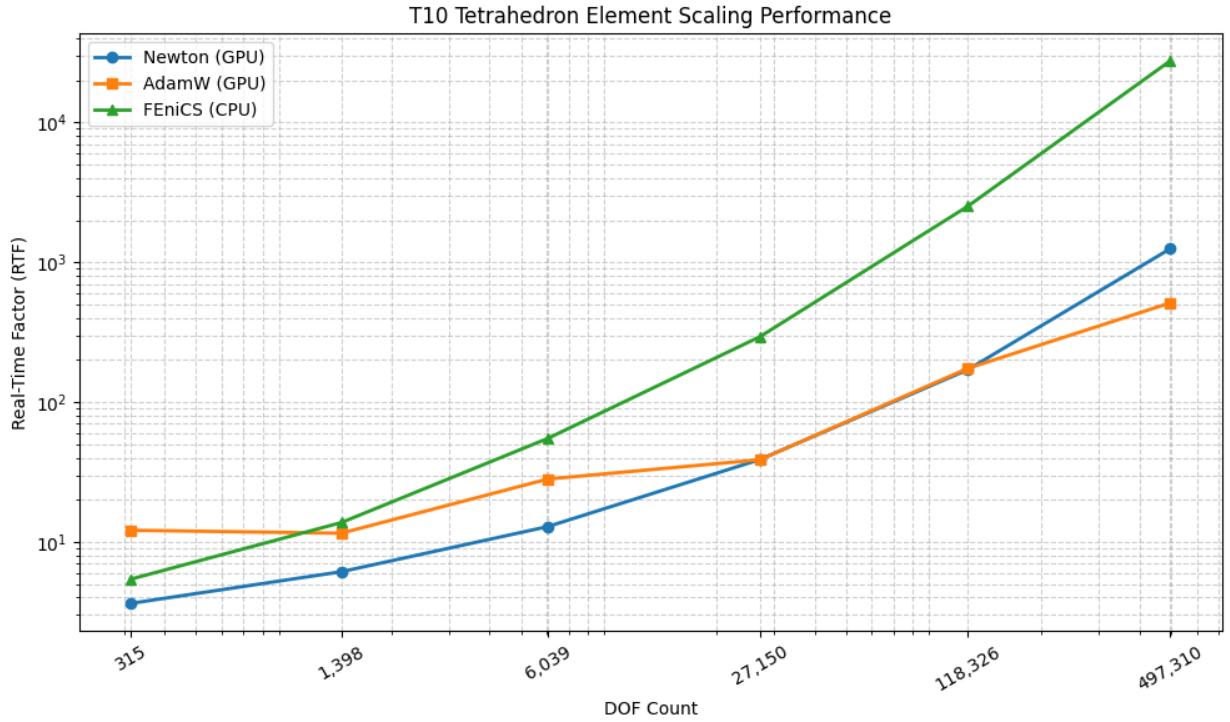


Figure 6.1: Time-to-solution comparison for the T10 beam-sagging benchmark across mesh resolutions. GPU solvers (AdamW and Newton) are compared against the FEniCS CPU baseline.

Resolution	DOFs	Nodes	Elements	Constrained DOFs	ε_{in}	ε_{out}
RES0	12012	1001	1000	12	10^{-4}	10^{-4}
RES2	120012	10001	10000	12	10^{-4}	10^{-4}
RES4	600012	50001	50000	12	10^{-4}	10^{-4}
RES8	1200012	100001	100000	12	10^{-4}	10^{-4}
RES16	2400012	200001	200000	12	10^{-4}	10^{-4}
RES32	6000012	500001	500000	12	10^{-3}	10^{-3}

Table 6.5: ANCF3243 beam mesh statistics across six resolution levels. ε_{in}: inner gradient-norm stopping criterion; ε_{out}: outer ALM constraint-residual stopping criterion.

The CPU baseline for this benchmark is Project Chrono [130], which provides a mature, well-validated implementation of ANCF beam elements. Figure 6.2 shows the timing comparison across resolutions.

Resolution	Newton (GPU)	AdamW (GPU)	Project Chrono (CPU)
RES0	4.85	14.21	93.61
RES2	16.57	44.29	1151.64
RES4	74.38	222.83	6571.27
RES8	146.22	444.42	13018.00
RES16	297.35	891.52	25844.58
RES32	530.05	1769.61	66806.46

Table 6.6: RTF for the ANCF3243 beam-sagging benchmark across six resolution levels. Newton and AdamW are run on the GPU; Project Chrono on the CPU. Project Chrono enforces constraints via a different formulation (penalty/Lagrange multiplier at the rigid-body level) and uses its own internal convergence criterion, which is not directly comparable to the ALM criteria $\varepsilon_{\text{in}}/\varepsilon_{\text{out}}$.

6.1.3 ANCF3443 Shell Element Scaling

We discretize a rectangular ANCF3443 shell with in plane dimensions $x_{\text{size}} = 4$ and $y_{\text{size}} = 2$ and thickness $H = 0.1$ using a structured mesh of $n_x \times n_y$ elements in the x and y directions, where $n_x = n_y \in \{10, 20, 50, 100, 200, 400\}$ defines six resolution levels and yields uniform element sizes $L = x_{\text{size}}/n_x$ and $W = y_{\text{size}}/n_y$. The material response is modeled with a Saint Venant Kirchoff constitutive law with $E = 7 \times 10^8$, $\nu = 0.33$, and $\rho_0 = 2700$. A cantilever boundary condition is imposed by fully clamping the edge at $x = 0$, fixing all kinematic degrees of freedom of the nodes on that edge throughout the simulation. The external loading is a total vertical force F_z applied on the opposite free edge at $x = x_{\text{size}}$, distributed across all nodes on that edge, with an optional linear bias along the y direction controlled by a load ratio parameter and uniform loading recovered at load ratio 0.5. The load is applied during the initial loading phase and is released by setting $F_z = 0$ starting at step 100, and the system is advanced for 200 steps with time step $\Delta t = 5 \times 10^{-4}$, where the default total force is $F_z = -500$; Table 6.7 summarizes the mesh statistics and Table 6.8 reports the corresponding real time factors.

As with the ANCF3243 benchmark, the CPU reference is Project Chrono [130]. Figure 6.3 presents the timing comparison across resolutions.

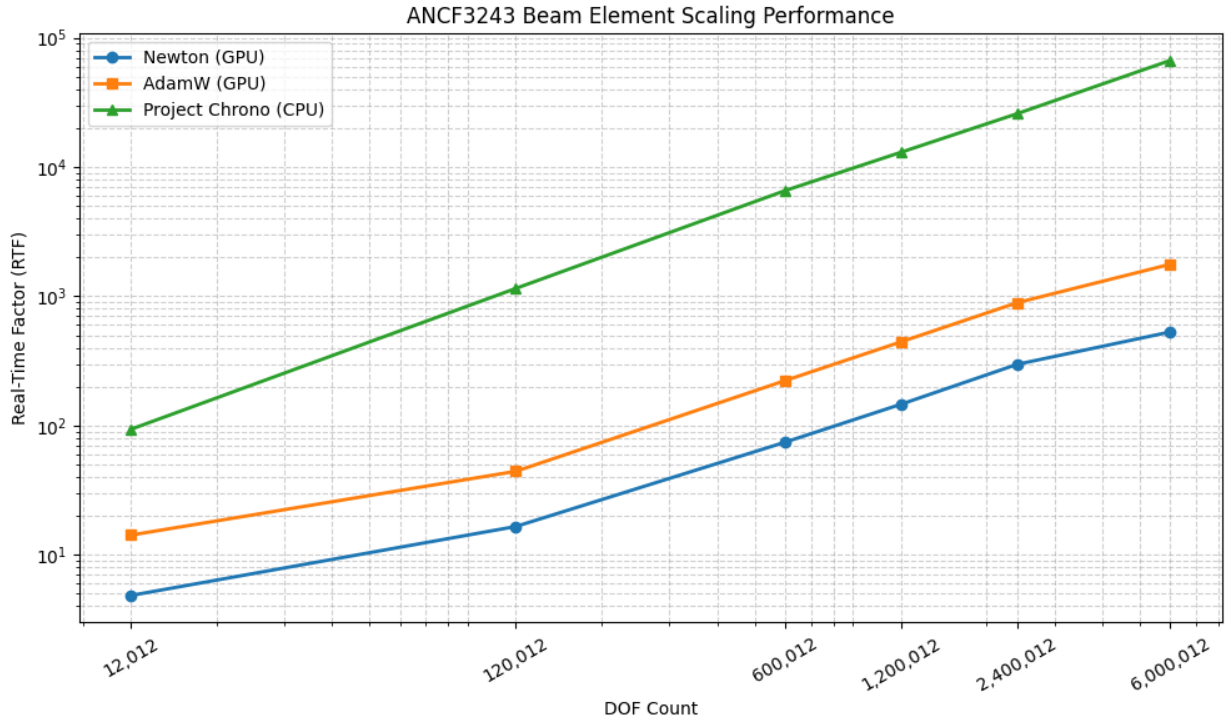


Figure 6.2: Time-to-solution comparison for the ANCF3243 beam-sagging benchmark across mesh resolutions. GPU solvers (AdamW and Newton) are compared against the Project Chrono CPU baseline.

Resolution	DOFs	Nodes	Elements	Constrained DOFs	ε_{in}	ε_{out}
RES0	1452	121	100	132	10^{-4}	10^{-4}
RES2	5292	441	400	252	10^{-4}	10^{-4}
RES4	31212	2601	2500	612	10^{-4}	10^{-4}
RES8	122412	10201	10000	1212	10^{-3}	10^{-3}
RES16	273612	22801	22500	1812	10^{-3}	10^{-3}
RES32	484812	40401	40000	2412	10^{-3}	10^{-3}

Table 6.7: ANCF3443 shell mesh statistics across six resolution levels. ε_{in}: inner gradient-norm stopping criterion; ε_{out}: outer ALM constraint-residual stopping criterion.

6.1.4 Validation on Geometrically Complex Meshes

Beyond the structured beam-sagging case used for scaling analysis, we validate the framework on three geometrically complex meshes that are broadly representative of shapes arising in

Resolution	Newton (GPU)	AdamW (GPU)	Project Chrono (CPU)
RES0	8.11	38.92	25.22
RES2	12.89	63.79	110.66
RES4	46.19	229.57	840.84
RES8	188.26	835.40	4476.33
RES16	432.60	2021.65	11931.25
RES32	967.82	6202.02	23709.70

Table 6.8: RTF for the ANCF3443 shell cantilever benchmark across six resolution levels. Newton and AdamW are run on the GPU; Project Chrono on the CPU. Project Chrono enforces constraints via a different formulation (penalty/Lagrange multiplier at the rigid-body level) and uses its own internal convergence criterion, which is not directly comparable to the ALM criteria $\varepsilon_{\text{in}}/\varepsilon_{\text{out}}$.

practical engineering and computer graphics simulation: a deformable tire, a Stanford Bunny, and a Utah Teapot. These geometries feature irregular mesh topology, varying element valence, and non-trivial curvature distributions, providing a more demanding test of the GPU implementation than the regular beam grid. All three cases use T10 tetrahedral elements, and the base of each mesh is clamped while gravity drives the deformation. Correctness is assessed by visual inspection of the deformed configuration against expected physically plausible behavior.

Utah Teapot. The Utah Teapot mesh comprises 23,006 nodes (69,018 DOFs) and 12,280 T10 tetrahedral elements, with overall dimensions of approximately 1.0053 m (height) $\times$ 1.2753 m (width) $\times$ 2.0529 m (length). The bottom 20% of nodes (5,444 nodes) are clamped, and a total upward force of 1,000 N is applied to the top 20% (1,873 nodes). The applied load is intentionally moderate; this case primarily serves to demonstrate the solver's ability to handle geometrically complex meshes with non-manifold-like junctions (spout, handle, and lid). The simulation is advanced for 1,000 steps at $\Delta t = 5 \times 10^{-4}$ s (0.5 s total). Figure 6.4 visualizes the loading configuration and the deformed teapot mesh geometry.

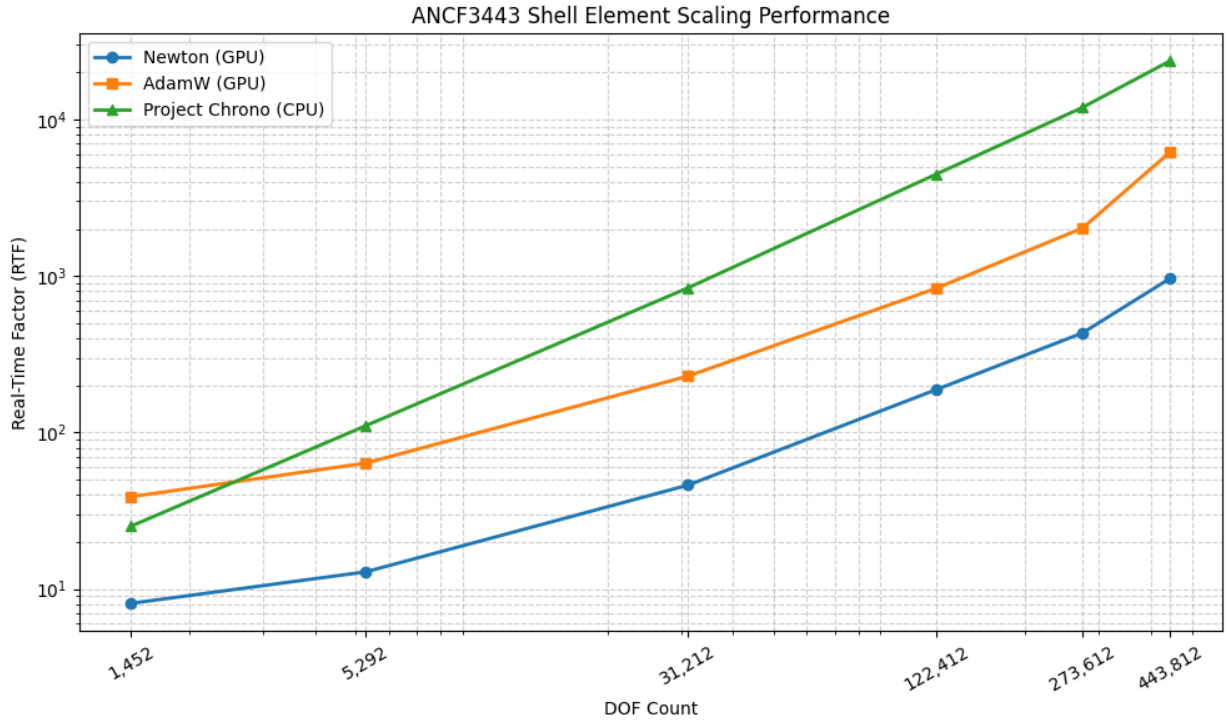
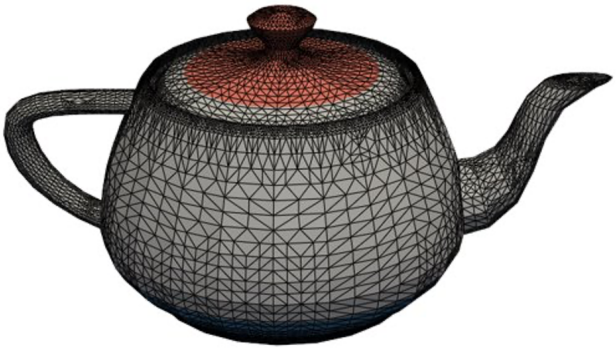


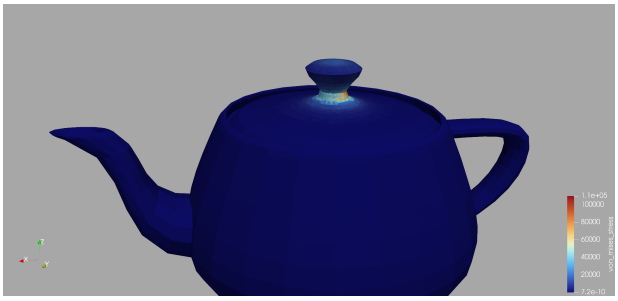
Figure 6.3: Time-to-solution comparison for the ANCF3443 shell beam-sagging benchmark across mesh resolutions. GPU solvers (AdamW and Newton) are compared against the Project Chrono CPU baseline.

Stanford Bunny. The Stanford Bunny mesh comprises 2,095 nodes (6,285 DOFs) and 1,066 T10 tetrahedral elements, spanning approximately 1.5413 m (height) × 1.2083 m (width) × 1.5594 m (length). All nodes with $z < -4$ m are clamped (545 nodes), and a total downward force of 35,000 N is distributed over all nodes with $z > 4$ m (582 nodes). The simulation is advanced for 8,000 steps at $\Delta t = 10^{-3}$ s (8.0 s total). Figure 6.5 visualizes the loading configuration and the deformed bunny mesh geometry.

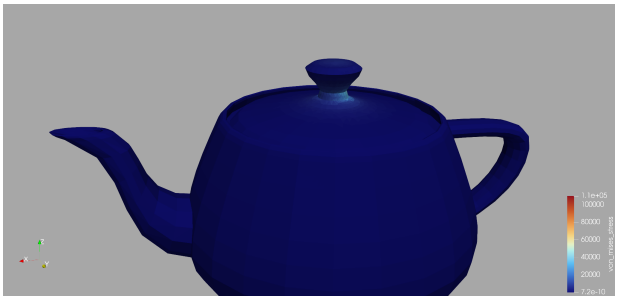
Deformable Tire. The deformable tire mesh comprises 34,414 nodes (103,242 DOFs) and 17,167 T10 tetrahedral elements. The bottom 10% of nodes (3,663 nodes) are clamped, and a total downward force of 3,000 N is applied to the top 20% (3,604 nodes), producing a characteristic stress band along the inner radius with secondary peaks at the contact arc consistent with a rim-clamped toroidal shell under axial compression. The simulation is



(a) Reference mesh, external force is distributed on the red nodes while blue nodes are fixed



(b) von Mises stress distribution at $t = 0.125\,\mathrm{s}$, during load application



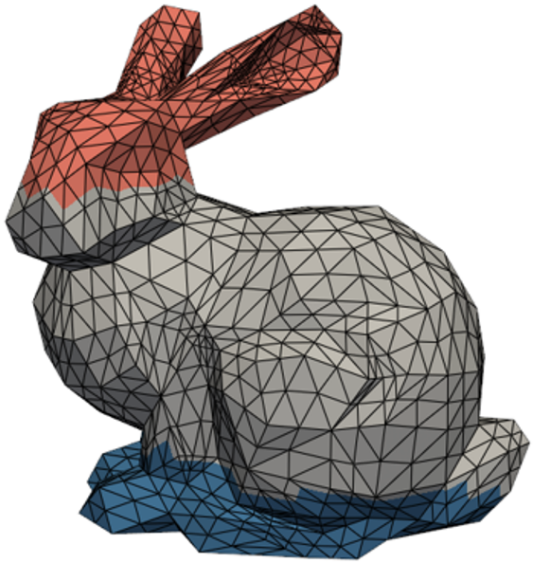
(c) von Mises stress distribution at $t = 0.25\,\mathrm{s}$, immediate after load removed

Figure 6.4: Utah Teapot benchmark snapshots showing the reference mesh, the early loaded configuration, and the subsequent release response under gravitational loading with a clamped base.

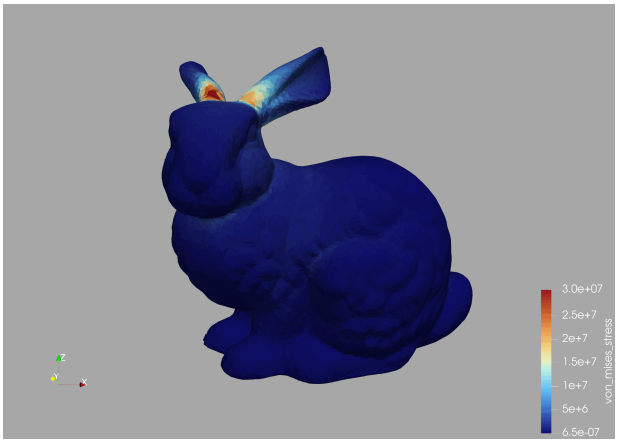
advanced for 1,200 steps at $\Delta t = 5 \times 10^{-4}\,\mathrm{s}$ (0.6 s total). Figure 6.6 visualizes the loading configuration and the deformed tire mesh geometry.

Table 6.9 reports the detailed timing results for the Utah Teapot, Stanford Bunny, and deformable tire benchmarks across four mesh resolutions.

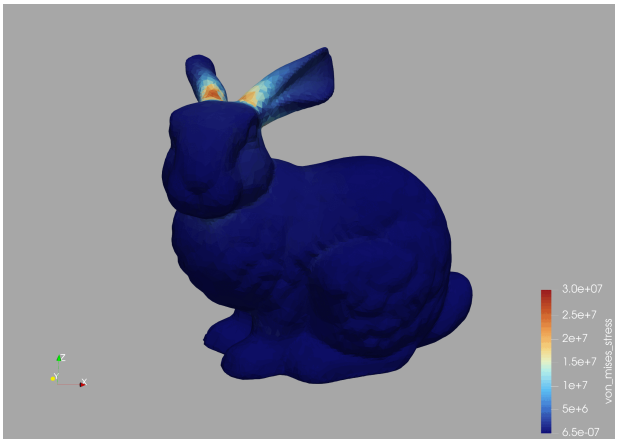
The material parameters used in all three cases are summarized in Table 6.10.



(a) Reference mesh, external force is distributed on the red nodes while blue nodes are fixed

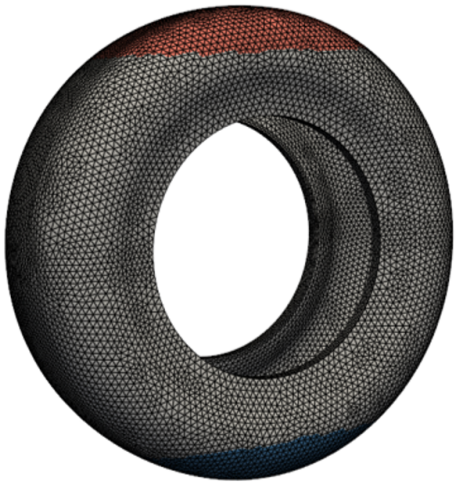


(b) von Mises stress distribution at $t = 0.01\,\mathrm{s}$, during initial load application

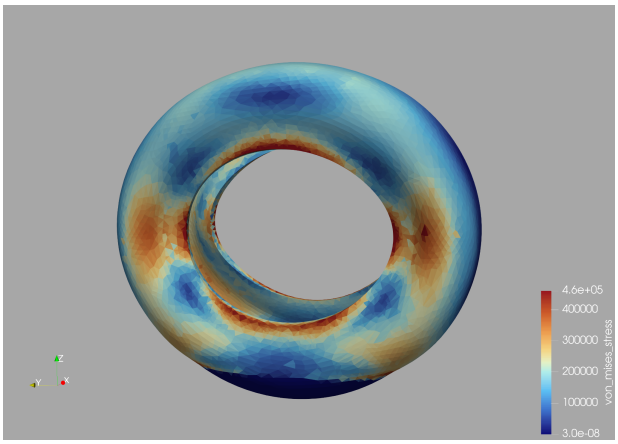


(c) von Mises stress distribution at $t = 0.4\,\mathrm{s}$, while the external force is applied, the steady state of the compressed configuration

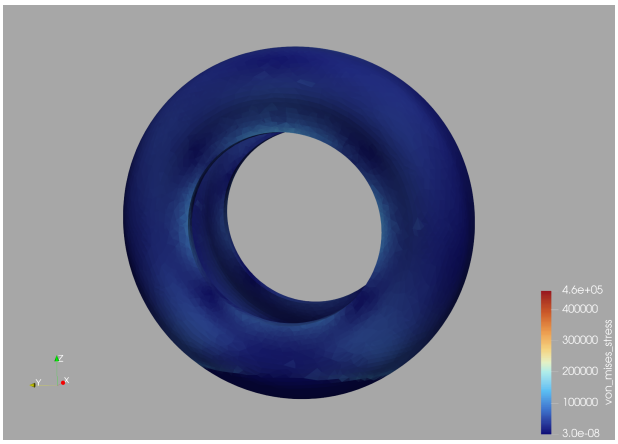
Figure 6.5: Stanford Bunny benchmark snapshots showing the reference mesh, the early loaded configuration, and the later compressed response under gravitational loading with a clamped base.



(a) Reference mesh, external force is distributed on the red nodes while blue nodes are fixed



(b) von Mises stress distribution at $t = 0.01\,\mathrm{s}$, during initial load application, and the mesh has been visually compressed



(c) von Mises stress distribution at $t = 0.21\,\mathrm{s}$, after load removed

Figure 6.6: Deformable tire benchmark snapshots showing the reference mesh, the early loaded configuration, and the subsequent release response under gravitational loading with a clamped rim.

Stanford Bunny		
Resolution	FEniCS (CPU)	Newton (GPU)
RES0	189.48	33.57
RES2	409.57	52.17
RES4	898.61	110.30
RES8	2485.82	270.59

Deformable Tire		
Resolution	FEniCS (CPU)	Newton (GPU)
RES0	4528.23	281.68
RES2	9342.74	450.76
RES4	18912.67	841.60
RES8	35091.84	1505.30

Utah Teapot		
Resolution	FEniCS (CPU)	Newton (GPU)
RES0	4488.44	164.64
RES2	8644.91	227.04
RES4	24063.05	593.83
RES8	85626.86	1808.58

Table 6.9: Detailed timing results for the Utah Teapot, Stanford Bunny, and deformable tire benchmarks across four mesh resolutions. FEniCS timings are reported for the CPU (Intel i7-13700KF); Newton timings are reported for the GPU (NVIDIA RTX 5090).

Utah Teapot	
Model	SVK
E	$2.0 \times 10^6\,\mathrm{Pa}$
ν	0.35
ρ_0	$1000\,\mathrm{kg/m^3}$
η_{damp}	5.0×10^4
λ_{damp}	5.0×10^4

Stanford Bunny	
Model	SVK
E	$3.0 \times 10^8\,\mathrm{Pa}$
ν	0.40
ρ_0	$920\,\mathrm{kg/m^3}$
η_{damp}	0
λ_{damp}	0

Deformable Tire	
Model	SVK
E	$1.0 \times 10^7\,\mathrm{Pa}$
ν	0.35
ρ_0	$1000\,\mathrm{kg/m^3}$
η_{damp}	5.0×10^3
λ_{damp}	5.0×10^3

Table 6.10: Material parameters for the three geometrically complex mesh benchmarks. All cases use the Saint Venant–Kirchhoff (SVK) constitutive model with the Newton solver on the GPU.

6.2 Roofline Analysis and Profiling

The roofline model [134] provides a hardware-aware upper bound on achievable kernel performance by relating a kernel's *arithmetic intensity* — the ratio of floating-point operations performed to bytes of data transferred from memory — to the two primary hardware limits of a GPU: peak compute throughput and peak memory bandwidth. A kernel whose arithmetic intensity falls below the *ridge point* (the intensity at which the diagonal memory-bandwidth

ceiling intersects the horizontal compute-throughput ceiling) is *memory-bandwidth-bound*: its performance is limited by the rate at which data can be delivered from memory, not by the rate at which arithmetic units can process it. The practical implication is that optimization effort is most productively directed toward the binding bottleneck: reducing global memory traffic and improving access coalescing for memory-bound kernels, and improving instruction-level parallelism or SM occupancy for compute-bound kernels. For GPU-accelerated FEA kernels, which involve irregular gather/scatter operations over large per-element data arrays, memory bandwidth is often the dominant bottleneck; the roofline analysis below confirms and quantifies this for most kernels, while some are instead limited by low occupancy or atomic contention.

Experimental setup All kernels are profiled using NVIDIA Nsight Compute (`ncu`) with the `--set full` collection mode on an NVIDIA GeForce RTX 5090. Profiling uses the same compiled binary and benchmark configurations as Section 6.1, specifically the beam-sagging test cases for each element type. Three mesh resolutions are profiled: RES4, RES8, and RES16; their mesh statistics (node counts, element counts, DOF counts) are given in Tables 6.1, 6.5, and 6.7 for the T10, ANCF3243, and ANCF3443 elements, respectively. Arithmetic intensity is computed as the ratio of executed FP64 operations to bytes of DRAM traffic, both extracted from hardware performance counters. The four kernels profiled are: `compute_p` (first Piola–Kirchhoff stress evaluation, Stage 1 of Section 5.5), `compute_internal_force` (internal force assembly, Stage 2 of Section 5.5), `assemble_sparse_hessian_tangent` (tangent stiffness assembly, Section 5.6.2), and `compute_grad_l` (augmented Lagrangian gradient, Section 5.6.1).

The RTX 5090 has a rated FP64 peak of 1,370.90 GFLOP/s. The Nsight-measured sustained DRAM bandwidth is 1,637 GiB/s, yielding a ridge point of approximately 0.78 FLOP/byte. Any kernel whose arithmetic intensity lies below this ridge point is limited by DRAM bandwidth, not compute throughput. The L1 and L2 bandwidth ceilings shown in the figures are included for qualitative context only; they are not strict performance bounds under a

DRAM-traffic-based arithmetic intensity definition.

T10 tetrahedral element — SVK and Mooney–Rivlin Figures 6.7 and 6.8 show the roofline plots for the T10 element with SVK and Mooney–Rivlin material models, respectively. Memory traffic is a dominant performance concern across both material models; however, as the per-kernel discussion below shows, not all kernels are strictly memory-bandwidth-bound. `compute_internal_force` and `compute_grad_l` fall below or near the ridge point and are memory-bound in the classical sense, while `compute_p` (AI ≈1.0–2.3) and the Hessian kernel at RES4 (AI ≈3.25) lie to the right of the ridge and are technically in the compute-bound zone, yet fall far short of the compute peak due to low occupancy and atomic contention discussed below.

The `compute_internal_force` kernel has an arithmetic intensity of approximately 0.73–0.74 FLOP/byte across all T10 resolutions and material models. This near-constant intensity is consistent with the kernel's structure: it loads a fixed volume of precomputed reference-configuration data (shape function gradients, quadrature weights, material state) per element and performs a bounded amount of arithmetic per byte loaded, making the ratio largely independent of mesh size. Achieved throughput scales from 120 GFLOP/s at RES4 to ≈695 GFLOP/s at RES16, as the larger problem size increases SM occupancy and amortizes kernel launch overhead over more concurrent work.

The most notable contrast between the two T10 material models is in `compute_p`. Under SVK, arithmetic intensity is ≈1.0–1.3 FLOP/byte; under Mooney–Rivlin, it rises to ≈2.2–2.3 FLOP/byte. The Mooney–Rivlin strain energy density requires evaluation of both principal invariants I_1 and I_2 of the right Cauchy–Green tensor and their derivatives with respect to $\mathbf{F}$, introducing substantially more arithmetic per stress call relative to the simpler quadratic SVK form. As a result, the MR `compute_p` kernel achieves higher throughput at RES16 (≈606 GFLOP/s) than its SVK counterpart (≈430 GFLOP/s), providing an internal validation that the profiler is capturing physically meaningful differences in kernel compute intensity.

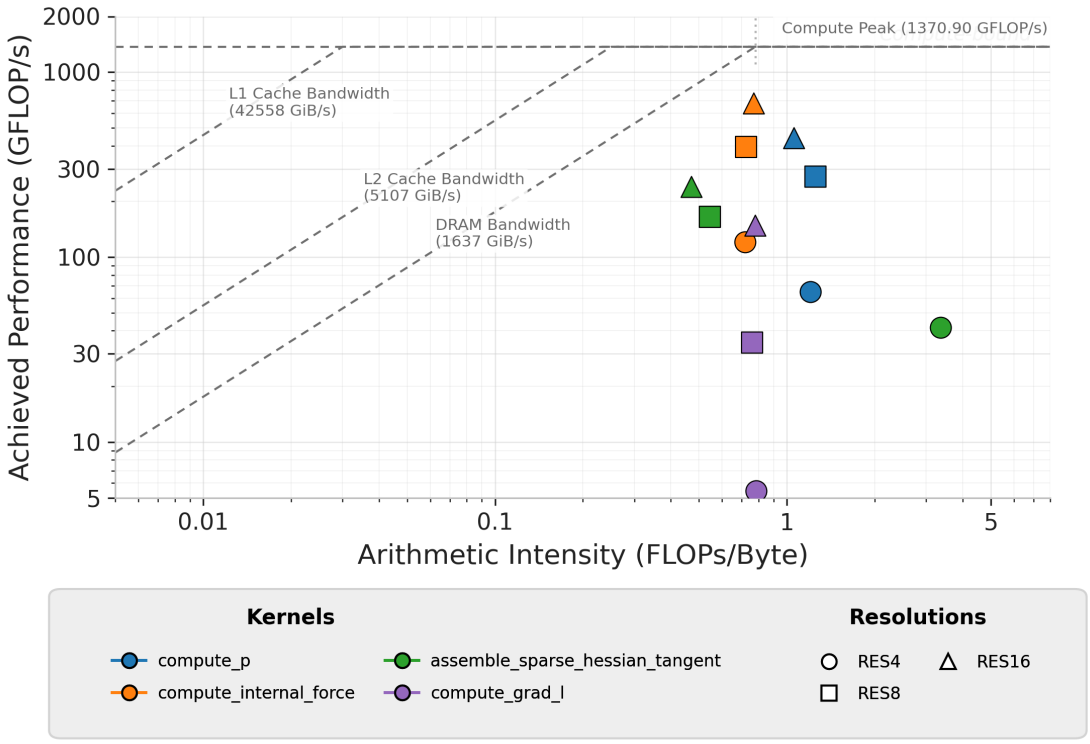


Figure 6.7: Roofline plot for the T10 tetrahedral element with St. Venant–Kirchhoff material on the NVIDIA GeForce RTX 5090. Each point represents one of the four dominant kernels at one of three mesh resolutions (RES4, RES8, RES16; see Table 6.1 for mesh statistics). Most kernels fall below or near the ridge point (≈0.78 FLOP/byte) and are memory-bandwidth-bound in the classical sense. `compute_p` and the Hessian kernel at low resolution lie to the right of the ridge and are nominally in the compute-bound zone, but fall far short of the compute peak due to low occupancy and atomic contention rather than memory bandwidth. The `compute_grad_l` kernel at RES4 is severely underutilized due to insufficient problem size to occupy the GPU.

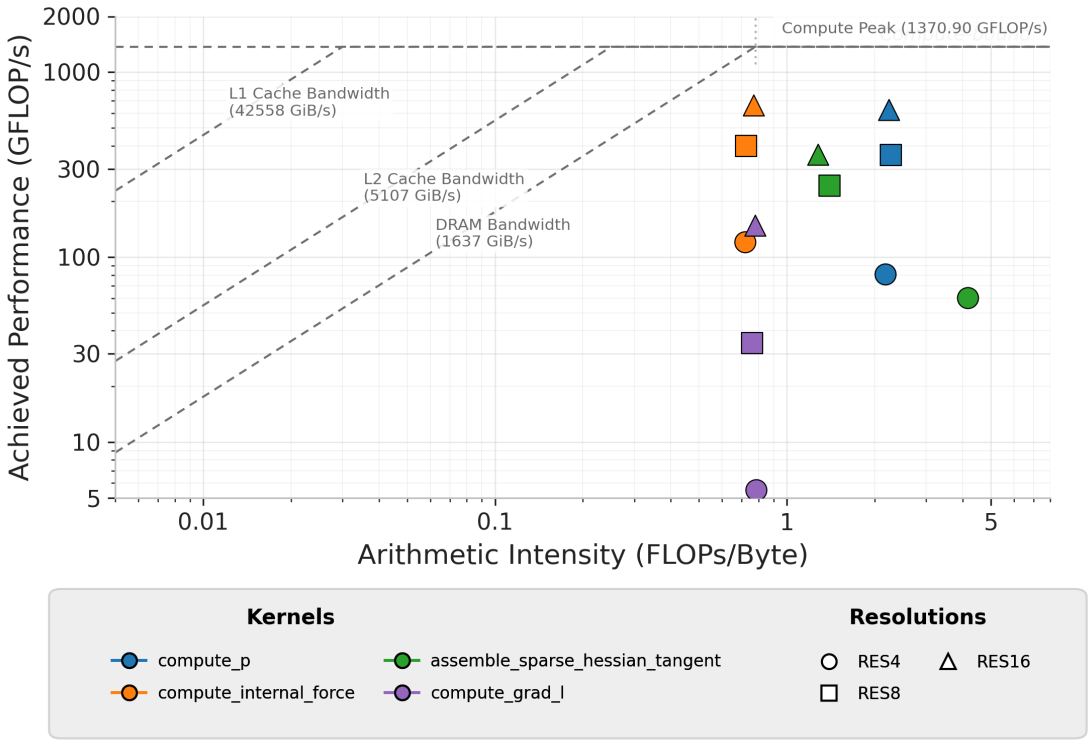


Figure 6.8: Roofline plot for the T10 tetrahedral element with Mooney–Rivlin material on the NVIDIA GeForce RTX 5090 (same resolution levels as Figure 6.7; see Table 6.1). The `compute_p` kernel shifts to higher arithmetic intensity relative to the SVK case (≈2.2–2.3 versus ≈1.0–1.3 FLOP/byte), reflecting the additional invariant computations required by the Mooney–Rivlin strain energy, and achieves correspondingly higher throughput at RES16.

The `compute_grad_l` kernel exhibits strongly resolution-dependent behavior: at T10 RES4, achieved throughput is only ≈5.5 GFLOP/s, growing to ≈150 GFLOP/s by RES16. At RES4 there are only 18,117 DOFs, meaning that a single kernel launch issues roughly 18,000 threads. The RTX 5090 contains 170 streaming multiprocessors, each capable of scheduling multiple resident warps; at 18,000 threads, the vast majority of SMs have little or no work to execute and remain idle. This is a GPU underutilization effect rather than a memory-bandwidth limitation, and it is consistent with the results in Section 6.1: the GPU Newton solver's real-time factor (RTF) at T10 RES4 is 12.85, not far ahead of the CPU baseline RTF of 54.91 relative to the gains seen at larger resolutions, where the GPU reaches

full occupancy.

ANCF3243 beam and ANCF3443 shell elements Figures 6.9 and 6.10 show the roofline plots for the ANCF3243 beam and ANCF3443 shell elements, respectively. In marked contrast to the T10 plots, both ANCF element types achieve substantially higher throughput on the dominant kernels. For ANCF3243, `compute_internal_force` reaches 788–856 GFLOP/s at an arithmetic intensity of approximately 0.82 FLOP/byte, placing it just to the right of the ridge point and at roughly 57–62% of the FP64 peak. ANCF3443 reaches 725–802 GFLOP/s near 1.0 FLOP/byte, or roughly 53–59% of peak. The gradient kernel `compute_grad_l` reaches 673–786 GFLOP/s for ANCF3243 and 250–610 GFLOP/s for ANCF3443.

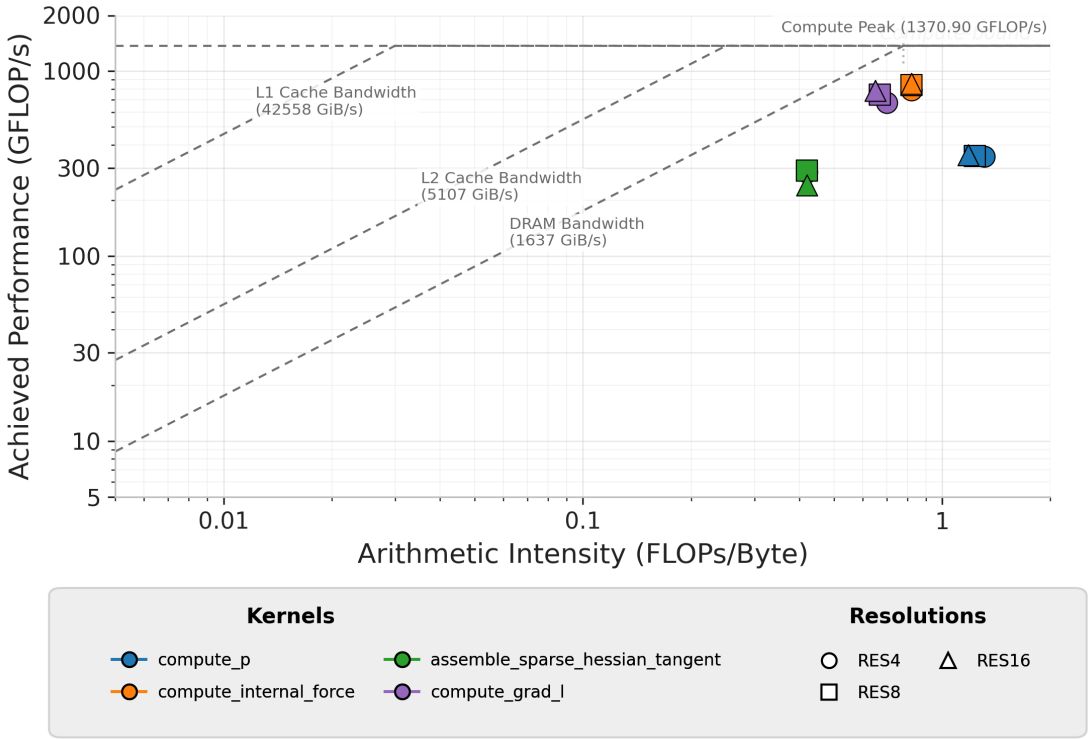


Figure 6.9: Roofline plot for the ANCF3243 beam element on the NVIDIA GeForce RTX 5090 (RES4, RES8, RES16; see Table 6.5 for mesh statistics). The `compute_internal_force` kernel operates just to the right of the ridge point at roughly 57–62% of the FP64 peak. The flat resolution-to-resolution trend reflects GPU saturation already at RES4.

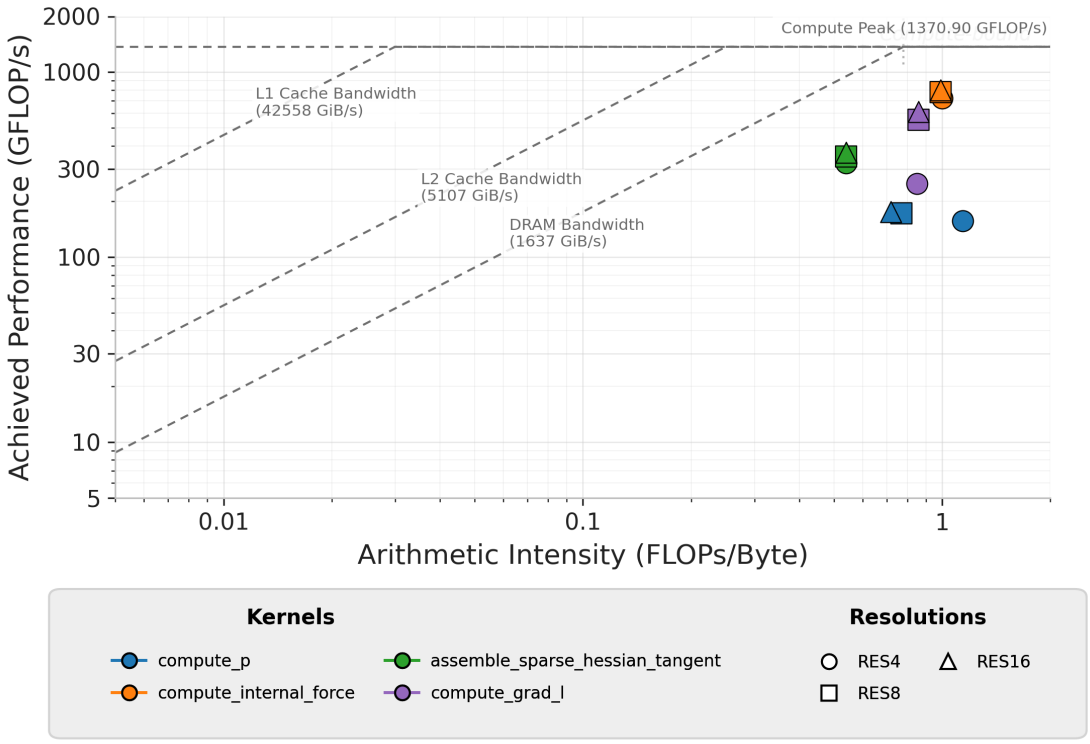


Figure 6.10: Roofline plot for the ANCF3443 shell element on the NVIDIA GeForce RTX 5090 (RES4, RES8, RES16; see Table 6.7 for mesh statistics). The `compute_internal_force` kernel approaches the DRAM bandwidth roof; the `compute_p` kernel achieves markedly lower throughput, consistent with the higher register pressure introduced by the full three-dimensional shell kinematics.

For ANCF3243, `compute_internal_force` operates at a constant 0.82 FLOP/byte across all three resolutions, placing it just to the right of the ridge point; achieved throughput of 788–856 GFLOP/s corresponds to roughly 57–62% of the FP64 peak. The gradient kernel `compute_grad_l` reaches 673–786 GFLOP/s at arithmetic intensities of approximately 0.65–0.70 FLOP/byte, placing it near the DRAM bandwidth ceiling at those intensities. Both kernels achieve substantially higher throughput than the T10 kernels at comparable intensities. This efficiency is attributable primarily to problem size: even at the smallest resolution profiled, ANCF3243 RES4 presents 600,012 DOFs and 50,000 elements to the GPU, fully occupying all SMs and enabling the memory system to sustain near-peak throughput.

This also explains the flat resolution scaling visible in Figure 6.9: because the GPU is already saturated at RES4 in the beam-sagging benchmark of Section 6.1, increasing the problem size to RES8 or RES16 yields only marginal additional throughput per kernel invocation, even though the total work — and wall-clock time — grows proportionally.

For ANCF3443, `compute_internal_force` achieves 725–802 GFLOP/s at arithmetic intensities near 1.0 FLOP/byte, or roughly 53–59% of the FP64 peak. By contrast, `compute_p` achieves only 157–176 GFLOP/s despite arithmetic intensities of 0.72–1.14 FLOP/byte — substantially below what the bandwidth ceiling would permit at those intensities. This behavior is consistent with higher per-thread register consumption from the full three-dimensional shell kinematics of ANCF3443, which includes thickness-stretch and higher-order gradient terms; elevated register pressure would reduce the number of warps resident per SM, lowering occupancy and suppressing throughput. Direct confirmation would require occupancy and register usage metrics from Nsight Compute.

Hessian assembly behavior The `assemble_sparse_hessian_tangent` kernel exhibits the most irregular behavior across element types and resolutions and warrants specific discussion. For T10 SVK at RES4, Nsight Compute reports an arithmetic intensity of ≈3.25 FLOP/byte — the highest value measured across all profiled configurations — yet the achieved throughput is only ≈41 GFLOP/s, far below what even DRAM bandwidth would permit at that intensity. At RES4 there are only 2,013 T10 elements; many pairs of adjacent elements share global DOF indices, so the atomic additions used to scatter element-level tangent contributions into the CSR matrix target a small set of memory locations repeatedly. This atomic contention serializes warp execution and dominates the runtime regardless of arithmetic intensity. As mesh resolution grows to RES16 (20,829 elements), contention per global entry decreases, the pattern of write targets diversifies, and throughput rises to ≈247 GFLOP/s, though atomic serialization remains a limiting factor.

For ANCF elements, the Hessian assembly kernel shows lower arithmetic intensities of 0.42–0.54 FLOP/byte and throughputs of 241–367 GFLOP/s, which are consistent across all

three resolutions. The larger problem sizes for ANCF mean that the ratio of atomic write targets to total operations is more favorable even at RES4, explaining the absence of the sharp low-resolution throughput collapse observed for T10.

Summary and optimization implications The roofline analysis establishes three principal findings. First, memory traffic is the dominant performance concern for most kernels and element types: no kernel approaches the FP64 compute peak of 1,370.90 GFLOP/s on the RTX 5090. Most kernels fall below or near the ridge point and are memory-bandwidth-bound in the classical sense; those that lie to the right of the ridge (`compute_p` and the Hessian kernel at low T10 resolution) fail to approach the compute peak due to occupancy limitations or atomic serialization, so data movement remains a central concern throughout. Second, the primary opportunity for performance improvement lies in `assemble_sparse_hessian_tangent`: it is the weakest kernel in absolute throughput across all element types, and the atomic contention at small T10 problem sizes points to a structural inefficiency that is not an inherent consequence of the memory-bandwidth bound. Element-coloring strategies that eliminate atomic writes, or two-pass assembly schemes (compute all contributions, then apply conflict-free scatter), are the most promising directions. Third, the throughput gap between T10 and ANCF elements on `compute_internal_force` and `compute_grad_l` is primarily a GPU occupancy effect: T10 at coarse resolutions does not generate enough concurrent thread work to fill the device, while ANCF problems — even at RES4 — present sufficient parallelism to sustain near-peak bandwidth utilization. This finding also explains why the shared-memory staging strategies discussed in Section 5.9 yielded negligible gains: staging intermediate data in on-chip shared memory does not reduce the initial DRAM load that constitutes the actual bottleneck, and the additional shared memory allocation per thread block further suppresses SM occupancy, negating any potential latency benefit.

Comparison with CPU roofline results A qualitative comparison with the CPU roofline study reported in Taylor's thesis [69] is instructive, although it should not be interpreted

as a strict apples-to-apples efficiency comparison. The CPU and GPU studies were carried out on different architectures, with different peak floating-point rates, cache hierarchies, sustained memory bandwidths, and somewhat different formulations and kernel organizations. In Taylor's case, the spread among the MR-A, MR-B, MR-C, and MR-D variants also reflects differences in OpenMP parallelization strategy rather than differences in the underlying finite-element operation alone. This point is important: arithmetic intensity and achieved throughput are not immutable properties of a mathematical kernel, but depend materially on implementation choices that affect locality, synchronization overhead, reduction structure, and effective memory traffic. In that sense, both studies lead to a similar methodological conclusion: roofline position is shaped not only by the governing equations, but also by how the computation is mapped onto the hardware. For reference, Taylor reports that the generalized internal-force evaluation for the ANCF3243 beam lies in a relatively low arithmetic-intensity regime of 0.106–0.136 FLOP/byte across the tested methods, while the corresponding range for the ANCF3443 shell is 0.113–0.146 FLOP/byte.

With that caveat in place, the overall trend is nonetheless favorable for the present GPU implementation. Relative to the CPU results in Taylor's thesis [69], the dominant GPU kernels in this work tend to operate at higher arithmetic intensity, indicating that the GPU formulation and kernel structure perform more floating-point work per byte transferred. This indicates a more compute-intensive realization of the same finite-element workloads per byte of DRAM traffic. The comparison is most meaningful at the level of trends rather than exact values. In the CPU study, some OpenMP variants move upward and rightward together, showing that improved parallelization strategy can simultaneously increase arithmetic intensity and delivered performance. Similarly, the GPU implementation not only benefits from the higher raw capabilities of the hardware, but also appears to organize the work so that a larger fraction of the available data movement is converted into useful floating-point evaluation.

At the same time, higher arithmetic intensity alone should not be taken as a sufficient indicator of superior performance. The present roofline results themselves show why: some

kernels that lie to the right of the ridge point remain far from the compute peak because they are limited by occupancy loss, atomic contention, or other implementation-level effects rather than by global memory bandwidth. Conversely, some lower-intensity kernels achieve strong performance precisely because they stream data efficiently and run close to the DRAM roof. Accordingly, the most defensible interpretation is not simply that the GPU implementation is superior because its arithmetic intensity is higher, but rather that it is more compute-dense and uses data movement more effectively, while also delivering much higher absolute throughput on the target architecture.

6.3 Small-Scale Unit Test

We performed three different unit tests to validate the friction and contact modeling. The first test is an oblique impact test, which validates the tangential coefficient of restitution (COR) and post-collision angular velocity under varying impact angles and material parameters, and verifies the simulation results against their corresponding theoretical values in the sliding regime. The second test is a sphere stacking test, which evaluates whether the contact formulation remains stable for multiple resting contacts under gravity. The third test is a brick sliding on a slope test, which checks the transition between sticking and sliding as the incline angle crosses the friction limit.

Due to the lack of analytical solution for deformable body dynamics, the validation calculate the approximated rigid-equivalent Center of Mass (CoM) translation, linear and angular velocity for a given mesh instance, and then compare the post-collision CoM velocity and angular velocity with the theoretical solution of a rigid body under the same initial conditions. The rigid-equivalent state is computed by treating the mesh as a collection of discrete mass points (the nodes) and applying classical mechanics principles to derive an effective CoM and velocity that would correspond to a rigid body with the same mass distribution and motion. For a deformable body discretized by nodal shape functions ($N_i(\mathbf{X})$) over the reference domain (Ω_0), the consistent (scalar) mass matrix is obtained from the

kinetic energy,

$$T = \frac{1}{2}\int_{\Omega_0} \rho_0(\mathbf{X}), \dot{\mathbf{u}}(\mathbf{X}) \cdot \dot{\mathbf{u}}(\mathbf{X}), \mathrm{d}V, \qquad \dot{\mathbf{u}}(\mathbf{X}) = \sum_i N_i(\mathbf{X}), \dot{\mathbf{u}}_i,$$

which yields

$$M_{ij} = \int_{\Omega_0} \rho_0(\mathbf{X}), N_i(\mathbf{X}), N_j(\mathbf{X}), \mathrm{d}V, \qquad \mathbf{M}^{(3D)} = \mathbf{M} \otimes \mathbf{I}_3.$$

In practice, the element contributions are evaluated by numerical quadrature; for each element (e) and quadrature point (q), one uses the reference density (ρ_0), the shape functions ($N_i(\xi_q)$), and the Jacobian determinant ($\det \mathbf{J}e(\xi_q)$) to approximate

$$M_{ij}^{(e)} \approx \sum_q \rho_0^{(e)}, N_i(\xi_q), N_j(\xi_q), \det \mathbf{J}e(\xi_q), w_q,$$

followed by global assembly. The lumped nodal mass associated with node (i) is then derived from the consistent mass matrix via row-sum lumping,

$$m_i = \sum_j M_{ij}, \qquad \mathbf{M}_L = \mathrm{diag}(m_1, \ldots, m_n),$$

which preserves the total mass ($\sum_i m_i = \sum_{i,j} M_{ij}$) and defines the nodal weights used in the rigid-equivalent reduction.

Given nodal positions ($\mathbf{r}_i(t) \in \mathbb{R}^3$) and nodal velocities ($\mathbf{v}_i(t) \in \mathbb{R}^3$) for the subset of nodes defining a body, the rigid-equivalent center-of-mass (CoM) translational state is computed as the mass-weighted average with respect to the lumped nodal masses. The total mass is

$$M = \sum_{i \in \mathcal{B}} m_i,$$

and, assuming (M>0), the CoM position and CoM velocity are defined by

$$\mathbf{r}_{\mathrm{com}} = \frac{1}{M}\sum_{i \in \mathcal{B}} m_i \mathbf{r}_i, \qquad \mathbf{v}_{\mathrm{com}} = \frac{1}{M}\sum_{i \in \mathcal{B}} m_i \mathbf{v}_i.$$

This choice ensures consistency with the discrete linear momentum of the body, since ($M\mathbf{v}_{\text{com}} = \sum_{i\in\mathcal{B}} m_i\mathbf{v}_i$), and it provides the translational component of the rigid-equivalent motion used subsequently to define relative velocities and angular-momentum quantities about the CoM.

6.3.1 Oblique Impact

To validate the contact force models during a dynamic interaction, a series of oblique impact experiments were performed under microgravity conditions. A spherical body was launched with an initial velocity vector, $\mathbf{v}_i = \mathbf{v}_{i,n} + \mathbf{v}_{i,t}$, striking the contact surface at an impact angle θ. The coordinate system defines n and t as the normal and tangential components relative to the surface, respectively. A 2D/3D visualization of the setup is shown in Figure 6.13.

Upon impact and subsequent separation, the post-collision linear velocity, $\mathbf{v}'_i = \mathbf{v}'_{i,n} + \mathbf{v}'_{i,t}$, and angular velocity, ω'_i, were recorded. The tangential coefficient of restitution (COR), e_t, is defined as the ratio of the tangential velocity magnitudes post- and pre-collision:

$$e_t = \frac{|\mathbf{v}'_{i,t}|}{|\mathbf{v}_{i,t}|} \tag{6.1}$$

In instances where the friction force F_t reaches saturation, governed by the relationship:

$$|F_t| = \mu|F_n| \tag{6.2}$$

the impact remains within the sliding regime. Consequently, the post-collision kinematics may be determined analytically through the application of rigid body dynamics [135]. Specifically, the theoretical tangential COR and post-impact angular velocity are given in Eqs. (6.3) and (6.4).

$$e_t = 1 - \frac{\mu(1+e)}{\tan\theta}, \tag{6.3}$$

$$|\omega_i'| = \frac{5}{2}\frac{\mu(1+e)|v_{i,n}|}{R}. \tag{6.4}$$

The tangential coefficient of restitution, e_t, is functionally dependent upon the normal coefficient of restitution e, the friction coefficient μ, and the impact angle θ. The transition to a sustained sliding regime is governed by a critical impact angle, θ^*, which is analytically derived as [136]:

$$\theta^* = \arctan\left[\frac{7}{2}\mu_s(1+e)\right] \tag{6.5}$$

In the oblique impact test, the sphere mesh is represented by a tetrahedron mesh consisting of 5883 vertices and 3438 tetrahedron elements, with a radius of 0.15 m. The floor is represented by a tetrahedron mesh consisting of 9408 elements and 12645 vertices. The Newton solver parameters used in the test are listed in Table 6.11.

Parameter	Value
Time step, Δt	1×10^{-4} s
Inner/outer abs. tolerance	10^{-6}
ALM ρ	10^{12}
Max outer iterations	3
Max inner iterations	10
Gravity	$0\,\mathrm{m/s^2}$

Table 6.11: Solver parameters for the oblique impact test.

Two parameter sweeps are performed with impact angle θ ranging from 50° to 87° in 1° increments. The material and contact properties for the two experiments are summarized in Table 6.12. The density $\rho = 70.7355\,\mathrm{kg/m^3}$ is chosen such that the discretized sphere mesh has a total mass of 1 kg. Results for experiments 1 and 2 are shown in Figures 6.11 and 6.12, respectively, with the rigid-body analytical predictions of Eqs. (6.3)–(6.4) overlaid for $\theta > \theta^*$. The shaded region marks the sticking regime ($\theta < \theta^*$) where the analytical expressions do not apply.

In Experiment 1 ($\mu = 0.3$, $e = 1.0$, $\theta^* \approx 64.5°$), the tangential COR in panel (a) rises from approximately 0.45 at $\theta = 50°$ toward unity as $\theta \to 87°$, following the analytical prediction

closely for $\theta > \theta^*$. The post-impact angular velocity in panel (b) decreases monotonically with θ as expected from Eq. (6.4), but the simulated values lie systematically above the rigid-body prediction. This offset reflects elastic energy stored in the deformable sphere during contact and partially released as additional spin upon rebound, an effect absent from the rigid-body model.

In Experiment 2 ($\mu = 0.35$, $e = 0.9$, $\theta^* \approx 66.7°$), the higher Kelvin–Voigt damping ($\eta_{\text{damp}} = \lambda_{\text{damp}} = 5 \times 10^2\,\text{Pa·s}$) suppresses elastic oscillations, and both e_t and $|\omega_i'|$ agree with the analytical predictions to within data scatter for $\theta > \theta^*$. This near-exact agreement constitutes the stronger validation result: once material damping is sufficient to remove elastic rebound effects, the frictional contact model recovers the rigid-body impulse response with high fidelity.

Parameter	Experiment 1	Experiment 2
Material model	SVK	SVK
Young's modulus, E	1×10^7	1×10^7
Poisson's ratio, ν	0.3	0.3
Density, ρ	70.7355	70.7355
Damping, η_{damp}	5×10^1	5×10^2
Damping, λ_{damp}	5×10^1	5×10^2
Contact friction, μ	0.3	0.35
Normal COR, e	1.0	0.9

Table 6.12: Material and contact properties for the oblique impact parameter sweeps.

6.3.2 Brick Sliding on a Slope

A rectangular prism (the "brick") resting on a flat surface tilted to a fixed slope angle α is used to isolate the quasi-static stick–slip transition governed by Coulomb friction. For slope angles below the critical angle α_c, the frictional force is sufficient to prevent sliding and the brick remains stationary; above α_c, the brick accelerates down the slope under kinetic friction. The test thereby exercises both branches of the Coulomb friction law and verifies that the contact model correctly captures the stick–slip threshold and subsequent sliding kinematics.

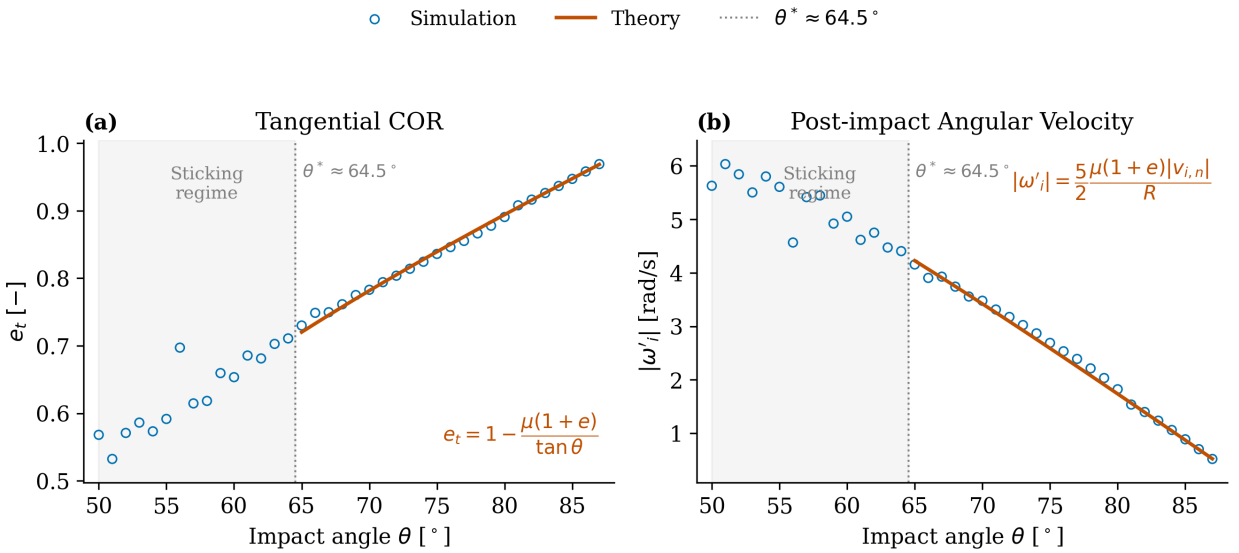


Figure 6.11: Oblique impact results for Experiment 1 ($\mu = 0.3$, $e = 1.0$, $\theta^* \approx 64.5°$). Panel (a): tangential COR e_t vs. impact angle θ. Panel (b): post-impact angular velocity $|\omega_i'|$ vs. θ. Solid line: rigid-body analytical prediction, valid for $\theta > \theta^*$ (sliding regime). Shaded region: sticking regime where the analytical expressions do not apply. The simulated $|\omega_i'|$ in panel (b) lies slightly above the rigid-body prediction, attributable to elastic energy stored in the deformable sphere and released as additional spin upon rebound.

For validation purposes, the simulation results are compared against the rigid-body analytical solution. For a rigid block on an inclined plane, the critical angle at which sliding initiates satisfies:

$$\alpha_c = \arctan(\mu_s), \tag{6.6}$$

where μ_s is the static friction coefficient. Once sliding occurs, the acceleration of the COM along the slope is:

$$a_{\mathrm{COM},\parallel} = g\left(\sin\alpha - \mu_k \cos\alpha\right), \tag{6.7}$$

where g is the gravitational acceleration and μ_k is the kinetic friction coefficient. If sliding initiates from rest at $t = 0$, the COM velocity and displacement along the slope follow $v_\parallel(t) = a_{\mathrm{COM},\parallel}\, t$ and $s_\parallel(t) = \frac{1}{2} a_{\mathrm{COM},\parallel}\, t^2$, respectively. Deformability introduces transient elastic oscillations, but these do not alter the steady-state sliding behavior.

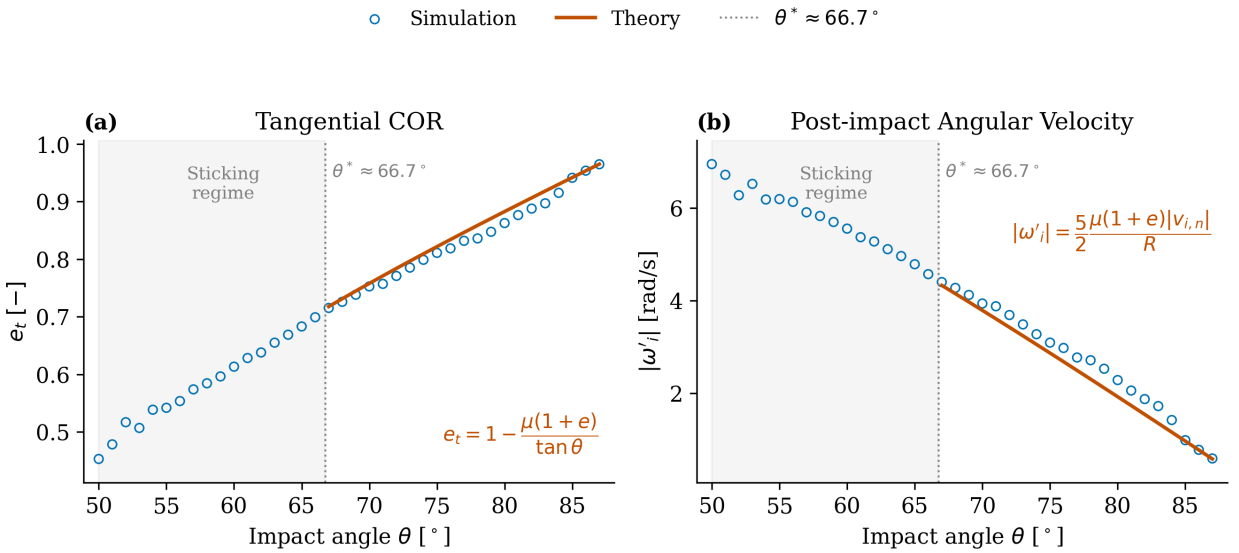


Figure 6.12: Oblique impact results for Experiment 2 ($\mu = 0.35$, $e = 0.9$, $\theta^* \approx 66.7°$). Panel (a): tangential COR e_t vs. impact angle θ. Panel (b): post-impact angular velocity $|\omega'_i|$ vs. θ. Higher material damping suppresses elastic rebound effects; simulated values agree with the rigid-body analytical prediction to within data scatter for $\theta > \theta^*$, demonstrating high-fidelity recovery of rigid-body impulse response.

To bracket the stick–slip transition systematically, four slope configurations are tested using $\mu_k = 0.2$ and $\mu_s = 0.25$, yielding $\alpha_c = \arctan(0.25) \approx 0.2450\,\text{rad}$: Slope 1 at $\alpha = 0.18\,\text{rad}$, well below $\arctan(\mu_k)$ and firmly in the stick regime; Slope 2 at $\alpha = \arctan(\mu_k) \approx 0.1974\,\text{rad}$, at the kinetic-friction threshold but still below α_c and therefore also in stick; Slope 3 at $\alpha = \alpha_c \approx 0.2450\,\text{rad}$, the marginal case at the static threshold; and Slope 4 at $\alpha = 0.25\,\text{rad}$, marginally above α_c and expected to slide. For Slope 4, the analytical prediction gives $a_{\text{COM},\parallel} = 9.81\,(\sin 0.25 - 0.2\cos 0.25) \approx 0.526\,\text{m/s}^2$, with Cartesian projections $a_x = -a_{\text{COM},\parallel}\cos\alpha \approx -0.510\,\text{m/s}^2$ and $a_z = -a_{\text{COM},\parallel}\sin\alpha \approx -0.130\,\text{m/s}^2$. The solver, contact, and material parameters are listed in Table 6.13.

Figure 6.14 shows the COM position and velocity histories for all four cases, with the rigid-body analytical prediction overlaid for Slope 4. Slopes 1 and 2 remain stationary throughout: x_{COM} and z_{COM} are flat in panels (a) and (b), and both velocity components remain at zero in panels (c) and (d), confirming correct enforcement of static friction below α_c. Slope 3, at

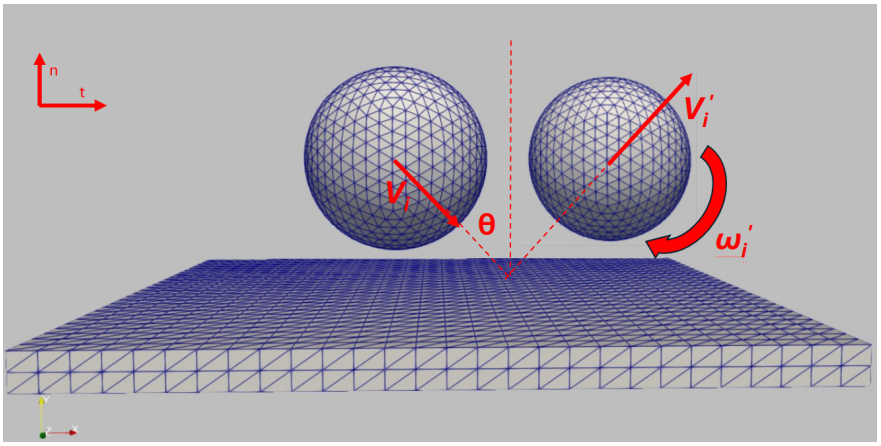


Figure 6.13: Oblique impact configuration. A sphere impacts a flat surface at impact angle θ with initial velocity $\mathbf{v}_i$; following rebound, the post-collision velocity $\mathbf{v}_i'$ and angular velocity ω_i' are recorded. The surface-normal (n) and tangential (t) directions are defined relative to the contact surface.

Parameter	Value
Time step, Δt	5×10^{-4} s
Inner/outer abs. tolerance	10^{-6} (−)
ALM ρ	10^{12} (−)
Max outer iterations	3 (−)
Max inner iterations	10 (−)
Gravity	9.81 m/s^2
Material Young's modulus, E	1×10^7 Pa
Coefficient of restitution, e	0.0 (−)
Static friction coefficient, μ_s	0.25 (−)
Kinetic friction coefficient, μ_k	0.2 (−)
Contact stiffness, k_n^{area}	1×10^8 Pa/m

Table 6.13: Solver, contact, and material parameters for the brick sliding on a slope test.

the marginal static threshold, exhibits a brief bidirectional velocity transient in panels (c) and (d)—including a momentary positive-v_x excursion as the elastic body rebounds against the contact—before arresting. This behavior is a consequence of elastic wave dynamics that locally redistribute contact forces below the sustained kinetic threshold. Slope 4 undergoes continuous sliding: panels (a) and (c) show x_{COM} and v_x tracking the analytical parabola and linear ramp closely after the initial elastic settling, confirming that the inferred along-slope acceleration is consistent with the predicted 0.526 m/s^2. The z-velocity in panel (d) converges to the analytical slope after $t \approx 0.15$ s, with the early oscillations attributable to elastic wave

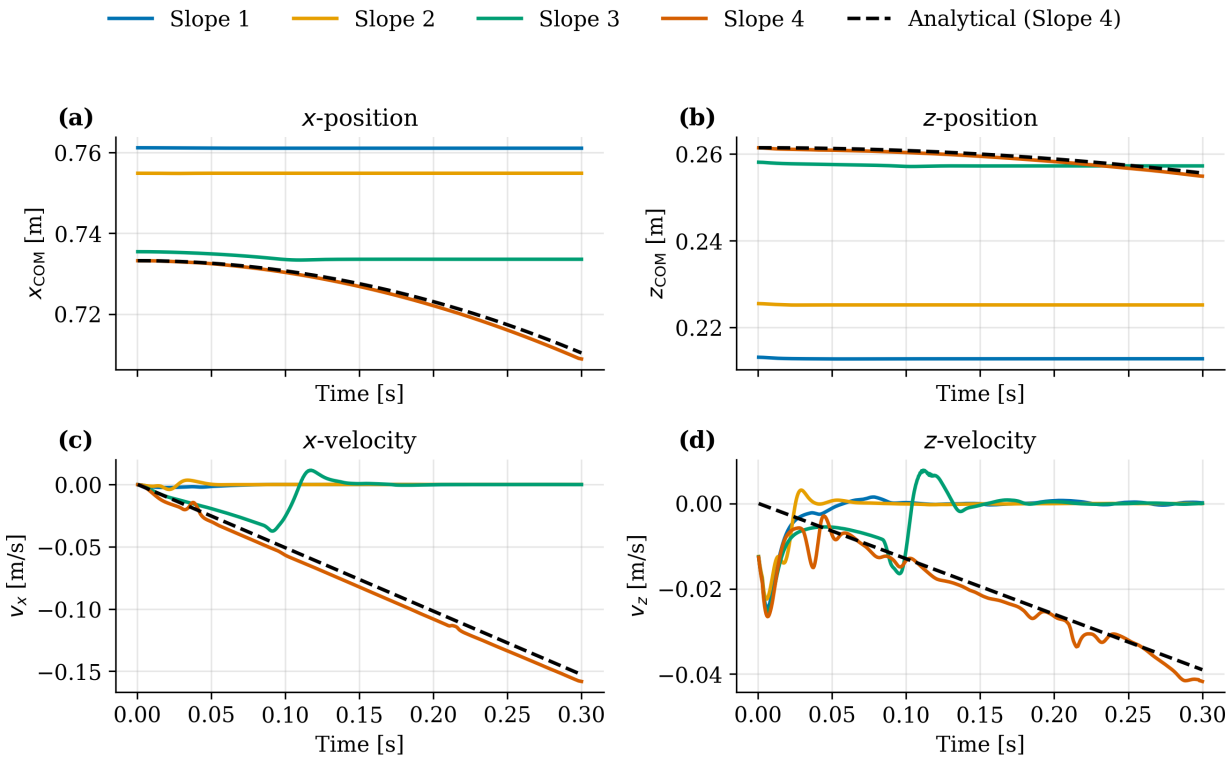


Figure 6.14: COM position and velocity histories for the brick-sliding validation test ($\mu_s = 0.25$, $\mu_k = 0.2$, $\alpha_c \approx 0.2450\,\text{rad}$). Slope 1: $\alpha = 0.18\,\text{rad}$ (stick); Slope 2: $\alpha \approx 0.1974\,\text{rad} = \arctan(\mu_k)$ (stick); Slope 3: $\alpha = \alpha_c \approx 0.2450\,\text{rad}$ (marginal, brief slide then arrest); Slope 4: $\alpha = 0.25\,\text{rad}$ (continuous sliding). The dashed line is the rigid-body analytical prediction for Slope 4, with along-slope acceleration $a_{\text{COM},\parallel} \approx 0.526\,\text{m/s}^2$ projected onto the x- and z-axes. Panels (a) and (b) show COM position in the x- and z-directions; panels (c) and (d) show the corresponding velocity components.

reflections in the deformable brick.

6.4 Engineering Joint Numerical Test

This section is reserved for numerical experiments associated with the primitive constraints and the revolute, cylindrical, and fixed joints introduced in Section 3.6. It is intended to collect joint-level verification examples alongside the solver performance benchmarks, small-scale unit tests, and large-scale application tests presented in this chapter.

6.4.1 Double Pendulum with Revolute and Spherical Joints

6.4.1.1 Double Pendulum with Revolute Joints Motion Test

This test considers a two-link deformable double pendulum constructed from two identical T10 tetrahedral beam meshes, each representing a prismatic beam of length 0.5 m and square cross-section 0.04 m × 0.04 m. The upper beam is attached to the world frame by a revolute joint located at $\mathbf{p}_{\text{top}} = (0,\, 0,\, 0.7)\,\text{m}$, while the lower beam is connected to the distal end of the upper beam by a second revolute joint. Both joints are defined with their hinge axes aligned with the global y-direction, thereby restricting the pendular motion predominantly to the x–z plane. The initial configuration is generated by rigidly rotating the undeformed upper beam by 35° from the downward vertical and the lower beam by −25° from the downward vertical before applying the joint constraints. Gravity acts in the negative z-direction with $g = 9.81\,\text{m/s}^2$, and no additional external forcing is applied; the subsequent motion is therefore driven entirely by gravity and moderated by material damping. The simulation uses a time step of $\Delta t = 5 \times 10^{-4}\,\text{s}$ with a default duration of 5000 steps, corresponding to a total simulated time of 2.5 s.

The beams use SVK material models, and the same material is assigned to both links. The material and solver parameters are summarized in Table 6.14. The revolute-joint formulation enforces point coincidence at the hinge location and rotational alignment about the prescribed axis, allowing the system to undergo pendular motion while maintaining hinge closure and the correct single-axis rotational freedom. This test serves as a dynamic verification case for the revolute-joint formulation under gravity-driven motion, with the primary outputs including joint angles, lower-tip trajectory, energy evolution, constraint residuals, and recovered joint reaction forces.

To investigate the influence of material damping on the dynamic response, four damping configurations are considered by setting $\eta_{\text{damp}} = \lambda_{\text{damp}} \in \{0,\, 10^2,\, 10^3,\, 10^4\}$ while keeping all other parameters fixed. The undamped case ($\eta_{\text{damp}} = \lambda_{\text{damp}} = 0$) isolates the numerical dissipation inherent in the implicit time integrator, whereas the remaining cases introduce

Parameter	Value
Material model	Saint Venant–Kirchhoff (SVK)
Young's modulus, E	$2 \times 10^6\,\mathrm{Pa}$
Poisson's ratio, ν	$0.30\,(-)$
Density, ρ_0	$1200\,\mathrm{kg/m^3}$
Shear damping, η_{damp}	$\{0,\ 10^2,\ 10^3,\ 10^4\}\,(-)$
Volumetric damping, λ_{damp}	$\{0,\ 10^2,\ 10^3,\ 10^4\}\,(-)$
Inner absolute tolerance	$1 \times 10^{-6}\,(-)$
Inner relative tolerance	$1 \times 10^{-6}\,(-)$
Outer tolerance	$1 \times 10^{-8}\,(-)$
ALM ρ	$1 \times 10^{10}\,(-)$
Max outer iterations	$8\,(-)$
Max inner iterations	$10\,(-)$
Time step, Δt	$5 \times 10^{-4}\,\mathrm{s}$
Default total steps	$5000\,(-)$
Line search enabled	false

Table 6.14: Material and solver parameters for the double-pendulum with revolute joints motion test. Four damping configurations are considered by setting $\eta_{\mathrm{damp}} = \lambda_{\mathrm{damp}}$ to each listed value.

progressively stronger material dissipation.

Figure 6.15 shows that the revolute-joint constraints remain well enforced throughout the motion test for all damping cases. The total, position, and orientation residuals remain bounded within approximately 10^{-10} to 10^{-8}, with no evidence of cumulative drift over the 2.5 s simulation. The position residual confirms accurate hinge-point coincidence, while the orientation residual indicates that the prescribed revolute-axis condition is maintained with comparable accuracy. Although the most strongly damped case ($\eta_{\mathrm{damp}} = \lambda_{\mathrm{damp}} = 10^4$) exhibits slightly larger residuals, their magnitudes remain sufficiently small to indicate robust constraint enforcement by the ALM outer loop (Algorithm 6).

To provide an independent dynamics baseline, a rigid-body counterpart of the revolute-joint double pendulum was constructed in Project Chrono [130]. Each flexible link was replaced by a rigid beam with the same overall dimensions ($0.04\,\mathrm{m} \times 0.04\,\mathrm{m} \times 0.5\,\mathrm{m}$), the same density-derived mass (0.96 kg), and the same center-of-mass inertia tensor about the beam frame. The hinge-point locations, initial beam orientations, center-of-mass positions, and gravitational loading were all preserved, and the rigid model uses pin joints at the same hinge

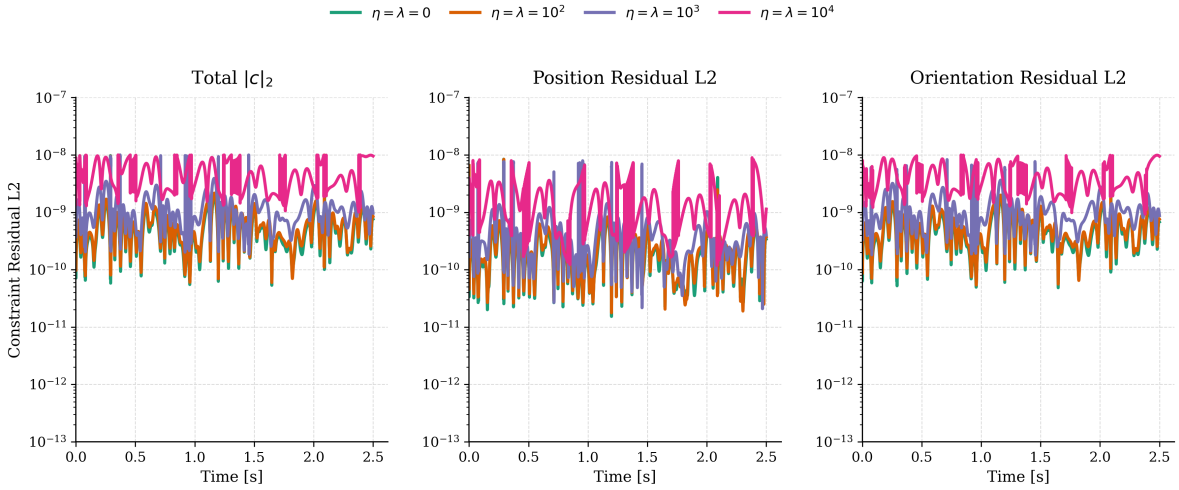


Figure 6.15: Constraint residual histories for the revolute-joint double-pendulum motion test under four damping configurations. Left: total residual $\|\mathbf{c}\|_2$. Center: position residual (CD rows). Right: orientation residual (DP1 rows). All residuals remain bounded within approximately 10^{-10} to 10^{-8} on a logarithmic scale with no cumulative drift.

locations with the same rotation axis. The rigid-body model therefore matches the flexible system in mass, mass distribution, geometry, support conditions, and initial configuration, while deliberately removing elastic deformation so that the comparison isolates the effect of beam flexibility on the pendular dynamics.

Figure 6.16 summarizes the kinematic and dynamic response of the double pendulum, with the Project Chrono rigid-body solution included as a reference. The lower-tip trajectory is confined to the x–z plane, consistent with the expected behavior of a system constrained by revolute joints aligned with the global y-axis. The lightly damped flexible cases follow trajectories broadly consistent with the rigid-body reference, while the differences that develop over time reflect elastic deformation of the beam links; the strongly damped case departs more noticeably as material dissipation reduces the motion envelope. The tip-speed histories display oscillatory variations associated with repeated gravitational acceleration and deceleration, and the rigid-body reference provides a useful baseline for gauging the extent to which beam compliance modifies the peak speeds and oscillation timing. The recovered joint reaction-force magnitudes remain bounded, and the upper-joint reaction is consistently larger than the

lower-joint reaction, as expected because the upper joint transmits the weight and inertial loads of both links.

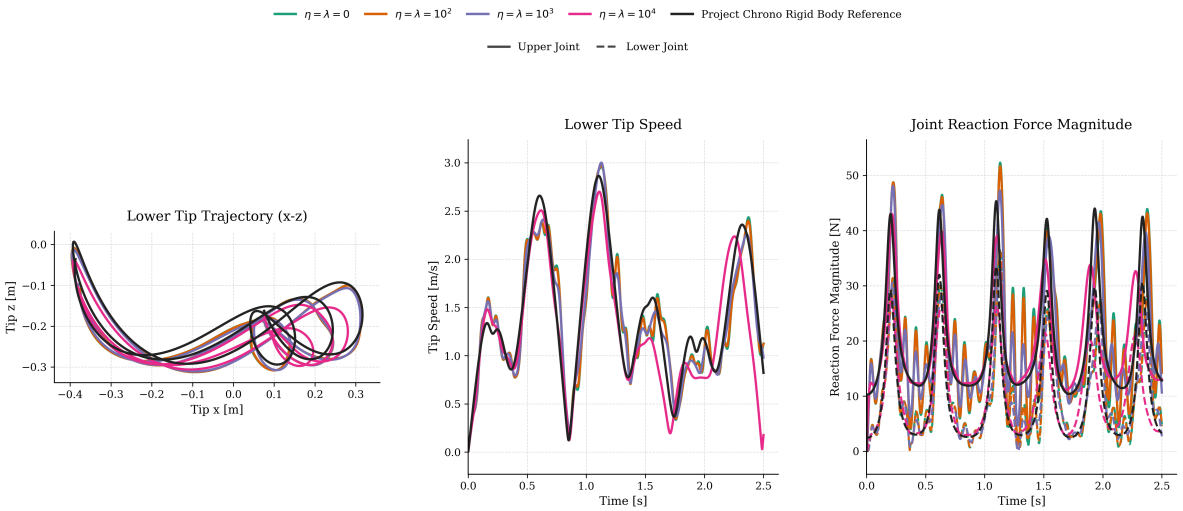


Figure 6.16: Motion and reaction summary for the revolute-joint double-pendulum motion test under four damping configurations, compared against a Project Chrono rigid-body reference. Left: lower-tip trajectory in the x–z plane. Center: lower-tip speed histories. Right: joint reaction-force magnitudes at the upper (solid) and lower (dashed) revolute joints.

Figure 6.17 demonstrates the expected dissipative energy behavior. The normalized total energy decreases monotonically in all cases and decays more rapidly as the damping parameters increase, consistent with the inclusion of material damping and the use of an implicit solver. The kinetic, potential, and elastic strain energy histories display oscillatory variations associated with the periodic exchange of gravitational and kinetic energy during pendular motion, while stronger damping progressively suppresses the amplitudes of these oscillations. Taken together, these trends indicate that the simulated response is physically consistent and that the revolute-joint formulation behaves as intended under dynamic loading.

6.4.1.2 Double Pendulum with Revolute Joints Vertical Pulling Test

Using the same double-pendulum geometry but in a purely vertical (hung) configuration, this test applies a vertical pulling force to the lower pendulum body and evaluates the resulting

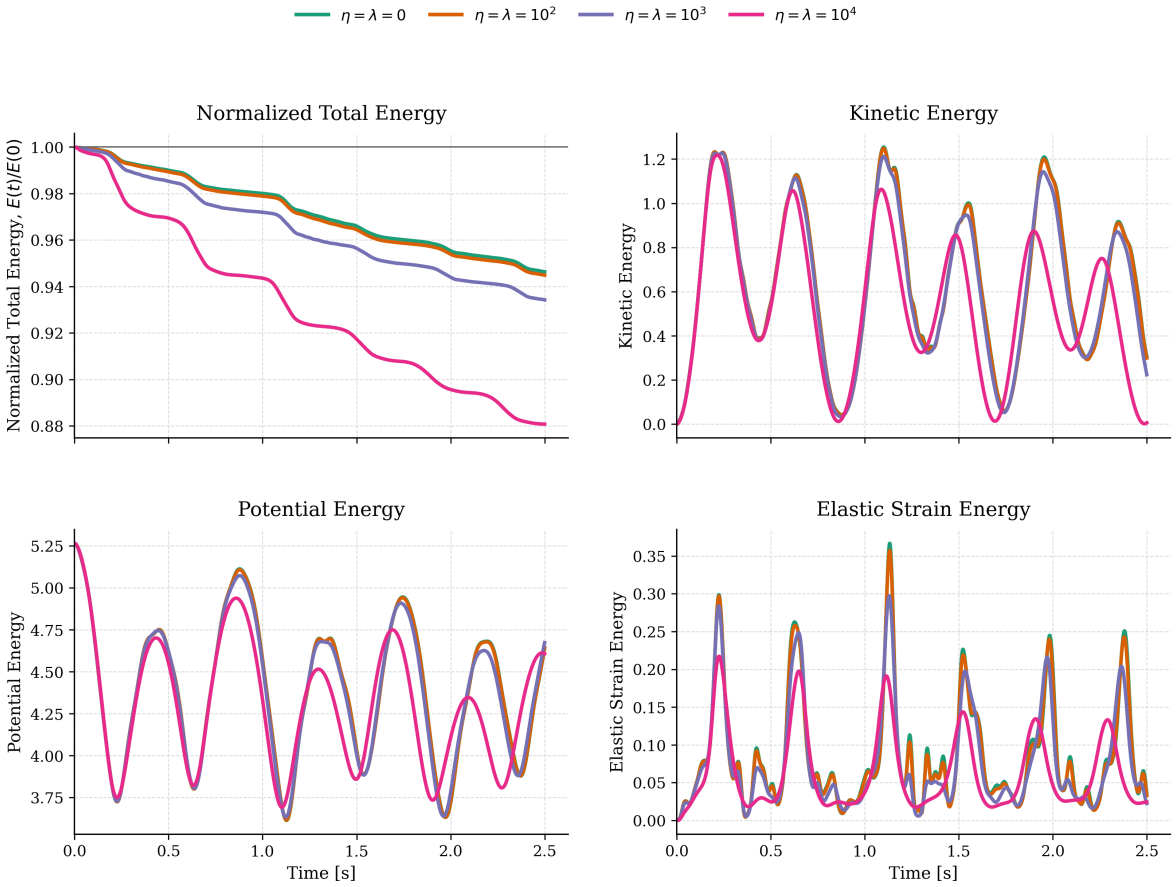


Figure 6.17: Energy response for the revolute-joint double-pendulum motion test under four damping configurations. Top left: normalized total energy $E(t)/E(0)$. Top right: kinetic energy. Bottom left: potential energy. Bottom right: elastic strain energy. Higher damping leads to faster total-energy decay and suppressed oscillation amplitudes.

motion and force transmission through the revolute joints. The purpose of this test is to verify that the ALM formulation recovers joint reaction forces correctly under a quasi-static loading scenario in which the analytical force balance is known exactly.

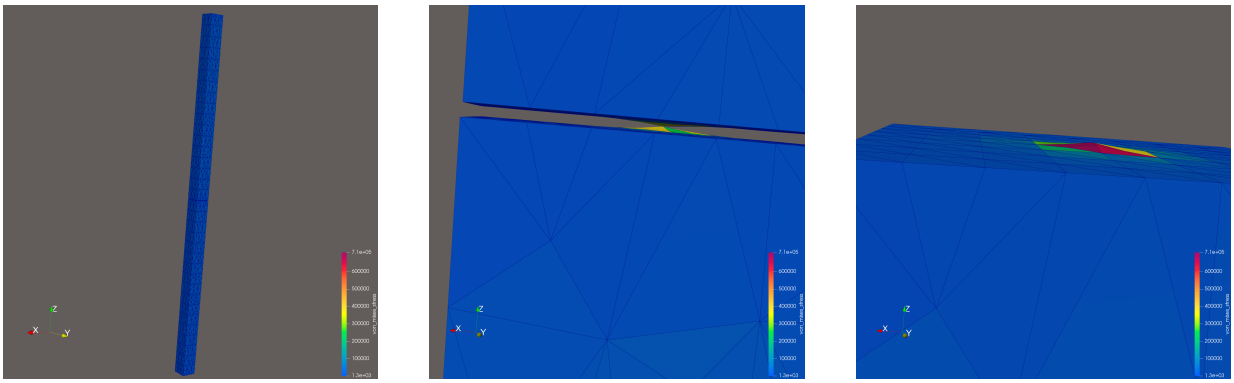
The two-link pendulum was constructed from two identical T10 tetrahedral beam meshes, each modeled with the Saint Venant–Kirchhoff constitutive law whose parameters are listed in Table 6.15. Both beams were initialized in a perfectly vertical hanging configuration: the upper beam was attached to the world frame by a revolute joint, and the lower beam was connected to the upper beam by a second revolute joint. Gravity acted in the negative

z-direction. An additional distributed downward load was applied to the bottom region of the lower beam; this external pull was ramped linearly over the first 200 time steps and then held constant for the remainder of the simulation. Four loading cases were evaluated with total applied pull forces of -20, -40, -60, and $-80\,\mathrm{N}$. Each simulation was run for 1000 steps, and joint reaction forces at both the upper and lower revolute joints were recovered from the ALM constraint forces and recorded as time histories for validation against the known static equilibrium values.

Parameter	Value
Material model	Saint Venant–Kirchhoff (SVK)
Young's modulus, E	$1 \times 10^7\,\mathrm{Pa}$
Poisson's ratio, ν	$0.30\,(-)$
Density, ρ_0	$1200\,\mathrm{kg/m^3}$
Shear damping, η_{damp}	$1 \times 10^4\,(-)$
Volumetric damping, λ_{damp}	$1 \times 10^4\,(-)$
Inner absolute tolerance	$1 \times 10^{-6}\,(-)$
Inner relative tolerance	$1 \times 10^{-6}\,(-)$
Outer tolerance	$1 \times 10^{-8}\,(-)$
ALM ρ	$1 \times 10^{10}\,(-)$
Max outer iterations	$8\,(-)$
Max inner iterations	$10\,(-)$
Time step, Δt	$5 \times 10^{-4}\,\mathrm{s}$
Line search enabled	false

Table 6.15: Material and solver parameters for the double-pendulum with revolute joints vertical pulling test.

Figure 6.19a presents the time histories of the Cartesian reaction-force components F_x, F_y, and F_z at the upper and lower revolute joints for four downward loading cases (-20, -40, -60, and $-80\,\mathrm{N}$). In all cases the response is dominated by the vertical component F_z, whereas the transverse components F_x and F_y remain close to zero throughout the simulation. This behavior is consistent with the loading and boundary conditions of the test, since the two-beam pendulum is initialized in a vertical configuration and subjected to a purely downward external pull applied to the lower tip region. The reaction-force histories also exhibit a clear and nearly linear increase in magnitude with increasing applied load, indicating that the recovered joint reactions are physically consistent and scale appropriately with the imposed external force.



(a) von Mises stress distribution in the double pendulum at steady state.

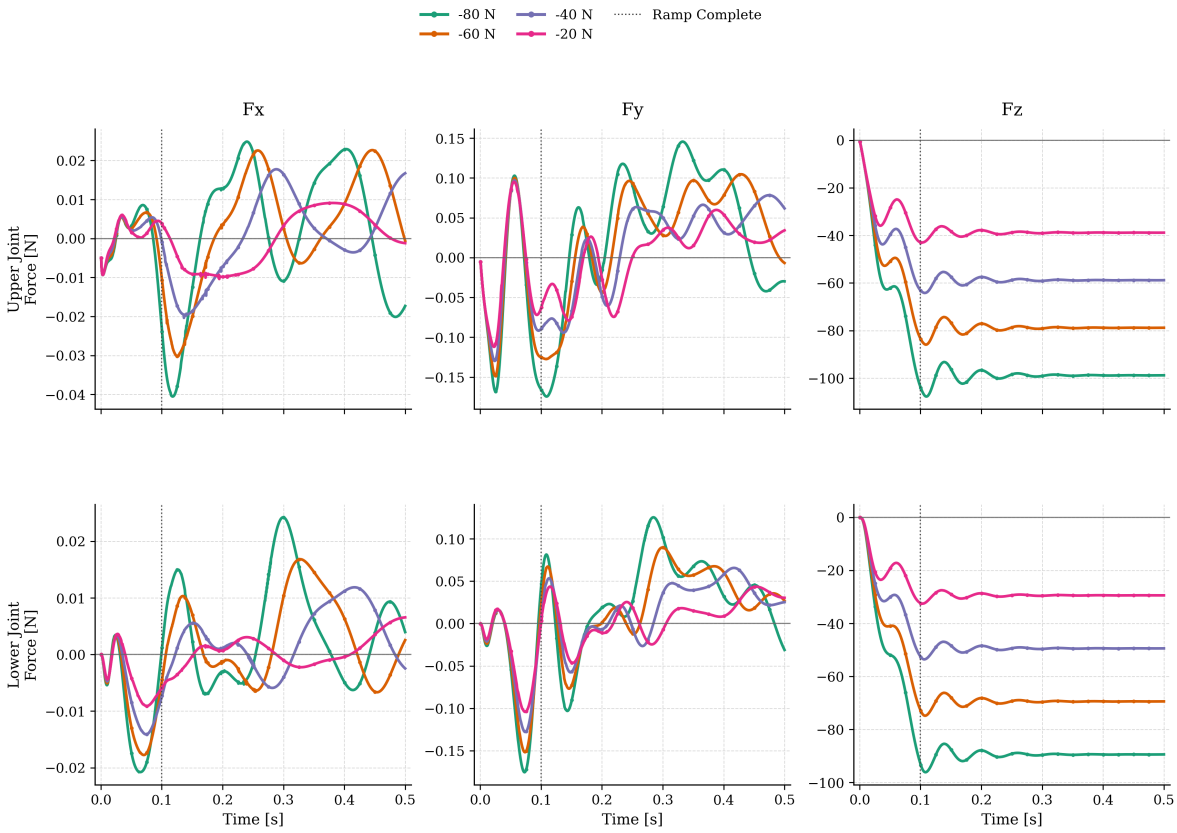
(b) Close-up of the lower revolute joint region.

(c) Close-up of the upper revolute joint region.

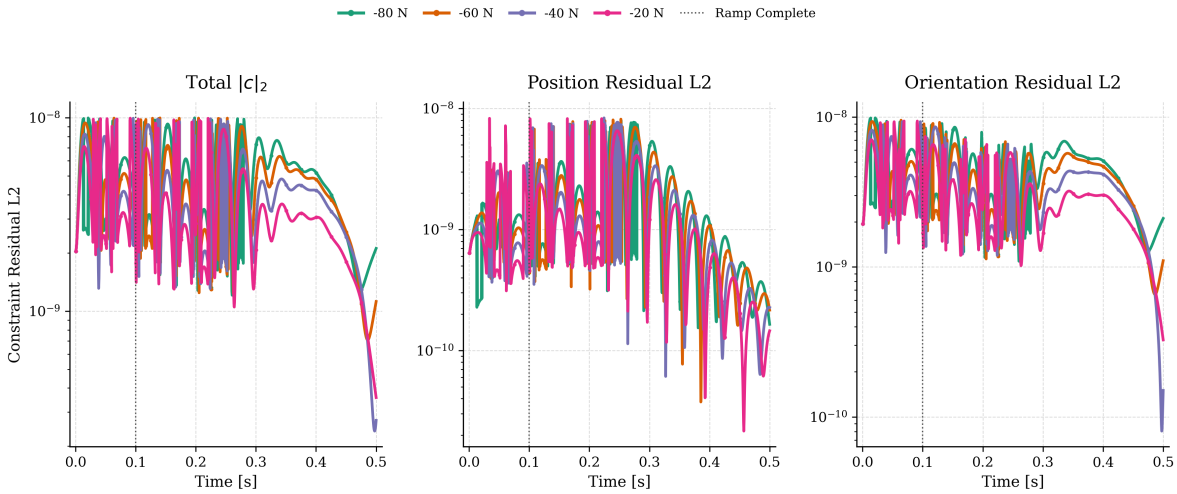
Figure 6.18: Steady-state von Mises stress field for the double-pendulum vertical pulling test under the $-20\,\mathrm{N}$ loading case. (a) Full view of the two-link pendulum under applied load. (b) Detailed screenshot near the lower revolute joint, where stress concentration arises from the transmitted pull force. (c) Detailed screenshot near the upper revolute joint, which carries the combined weight of both links and the applied load.

Table 6.16 compares the simulated vertical joint reactions at the final time step (step 1000) against the corresponding theoretical estimates based on beam self-weight and the applied pull. The expected lower-joint reaction was computed as the weight of one beam plus the applied pull, while the expected upper-joint reaction was computed as the weight of both beams plus the applied pull. Using a beam mass of $m_b = 0.96\,\mathrm{kg}$, these reference values correspond to $m_b g + |F_{\mathrm{pull}}|$ for the lower joint and $2\,m_b g + |F_{\mathrm{pull}}|$ for the upper joint. The small discrepancies observed in Table 6.16 are reasonable for a transient flexible-body simulation and confirm that the ALM-recovered revolute-joint reaction forces remain physically meaningful across the range of applied loads.

Figure 6.19b reports the constraint residuals for the revolute-joint vertical pulling test. The plotted quantities are obtained directly from the assembled joint constraint vector $\mathbf{c}(\mathbf{q})$, evaluated from the current deformed configuration at every time step. As described in Section 3.6.2.2, each revolute joint is assembled from three coordinate-difference (CD) constraints and two dot-product (DP1) constraints, yielding the five-row constraint vector



(a) Time histories of the Cartesian reaction-force components at the upper (top row) and lower (bottom row) revolute joints. The vertical component F_z (right column) dominates and scales with the applied load, while F_x and F_y remain near zero.



(b) Constraint violation histories. Left: total residual $\|\mathbf{c}\|_2$. Center: position residual (CD rows). Right: orientation residual (DP1 rows). All residuals remain below 10^{-8} on a logarithmic scale.

Figure 6.19: Revolute-joint vertical pulling test results for the four loading cases (-20, -40, -60, and $-80\,\mathrm{N}$). (a) Joint reaction-force component histories. (b) Constraint violation histories.

Pull Force	Upper Joint F_z				Lower Joint F_z			
[N]	Simulated [N]	Expected [N]	Error [N]	Rel. Err. [%]	Simulated [N]	Expected [N]	Error [N]	Rel. Err. [%]
−20	−37.78	−38.84	1.06	2.73	−28.66	−29.42	0.76	2.58
−40	−57.47	−58.84	1.37	2.33	−48.44	−49.42	0.98	1.98
−60	−77.08	−78.84	1.76	2.23	−68.16	−69.42	1.26	1.82
−80	−96.66	−98.84	2.18	2.21	−87.84	−89.42	1.58	1.77

Table 6.16: Comparison of simulated and expected vertical reaction forces at the upper and lower revolute joints at the final time step (step 1000). Expected values are computed from static equilibrium: lower joint $F_z = -(m_b g + |F_{\text{pull}}|)$, upper joint $F_z = -(2\, m_b g + |F_{\text{pull}}|)$, with $m_b = 0.96\,\text{kg}$ and $g = 9.81\,\text{m/s}^2$. Relative error is $|\text{Error}|/|\text{Expected}| \times 100\%$.

given by Eq. (3.76). The position residual corresponds to the three CD rows, which enforce coincidence of the two material points defining the hinge location (Eq. (3.73)); it is computed as the L_2 norm of these three scalar residuals and quantifies the hinge-point closure error. The orientation residual corresponds to the two DP1 rows, which enforce the revolute-axis condition by requiring selected body-fixed direction vectors to remain orthogonal to prescribed transverse directions (Eq. (3.74)); it is computed as the L_2 norm of the two DP1 residuals and quantifies violation of the rotational alignment conditions required for pure rotation about the prescribed revolute axis. As shown in Figure 6.19b, both residual components remain at or below 10^{-8} throughout the simulation for all four loading cases, confirming that the ALM outer loop (Algorithm 6) drives the constraint violation to negligible levels.

6.4.1.3 Double Pendulum with Spherical Joints Motion Test

This test considers a two-link deformable double pendulum constructed from two identical T10 tetrahedral beam meshes, each representing a prismatic beam of length 0.5 m and square cross-section 0.04 m × 0.04 m. The upper beam is attached to the world frame through a spherical joint located at $\mathbf{p}_{\text{top}} = (0,\, 0,\, 0.7)\,\text{m}$, while the lower beam is connected to the distal end of the upper beam through a second spherical joint. In contrast to the revolute case, the initial configuration is fully three-dimensional: the upper and lower links are first rotated in the pendulum plane by 35° and −25°, respectively, and are then assigned additional out-of-plane tilts of 18° and −27°. Gravity acts in the negative z-direction, and no external

load is applied beyond self-weight. The resulting motion is therefore governed by gravitational loading, large-deformation beam kinematics, and material damping.

Each beam is modeled with the SVK material, and the same parameters are assigned to both links. The material and solver parameters are summarized in Table 6.17. As in the revolute motion test, four damping configurations are considered by setting $\eta_{\text{damp}} = \lambda_{\text{damp}} \in \{0,\ 10^2,\ 10^3,\ 10^4\}$ while keeping all other parameters fixed, and each case is advanced for 5000 steps with $\Delta t = 5 \times 10^{-4}\,\text{s}$, corresponding to a total simulated time of 2.5 s.

Parameter	Value
Material model	Saint Venant–Kirchhoff (SVK)
Young's modulus, E	$2 \times 10^6\,\text{Pa}$
Poisson's ratio, ν	$0.30\,(-)$
Density, ρ_0	$1200\,\text{kg/m}^3$
Shear damping, η_{damp}	$\{0,\ 10^2,\ 10^3,\ 10^4\}\,(-)$
Volumetric damping, λ_{damp}	$\{0,\ 10^2,\ 10^3,\ 10^4\}\,(-)$
Inner absolute tolerance	$1 \times 10^{-6}\,(-)$
Inner relative tolerance	$1 \times 10^{-6}\,(-)$
Outer tolerance	$1 \times 10^{-8}\,(-)$
ALM ρ	$1 \times 10^{10}\,(-)$
Max outer iterations	$8\,(-)$
Max inner iterations	$10\,(-)$
Time step, Δt	$5 \times 10^{-4}\,\text{s}$
Total simulation steps	$5000\,(-)$
Line search enabled	false

Table 6.17: Material and solver parameters for the double-pendulum with spherical joints motion test. Four damping configurations are considered by setting $\eta_{\text{damp}} = \lambda_{\text{damp}}$ to each listed value.

Figure 6.20 shows that the spherical-joint constraint residuals remain small and bounded throughout the simulation. Because a spherical joint enforces only point coincidence (Table 3.1), the plotted residuals consist of the total constraint norm together with the upper- and lower-joint position residuals; no orientation residuals are involved. All residuals remain predominantly in the range 10^{-11} to 10^{-8}, with no evidence of cumulative drift over the 2.5 s response. The upper-joint residual is generally slightly larger than the lower-joint residual, which is physically reasonable because the world-attached upper joint transmits the motion of the full two-link system, whereas the lower joint transmits only the response of the distal

link. These residual levels indicate accurate and stable enforcement of the spherical-joint constraints for all damping cases.

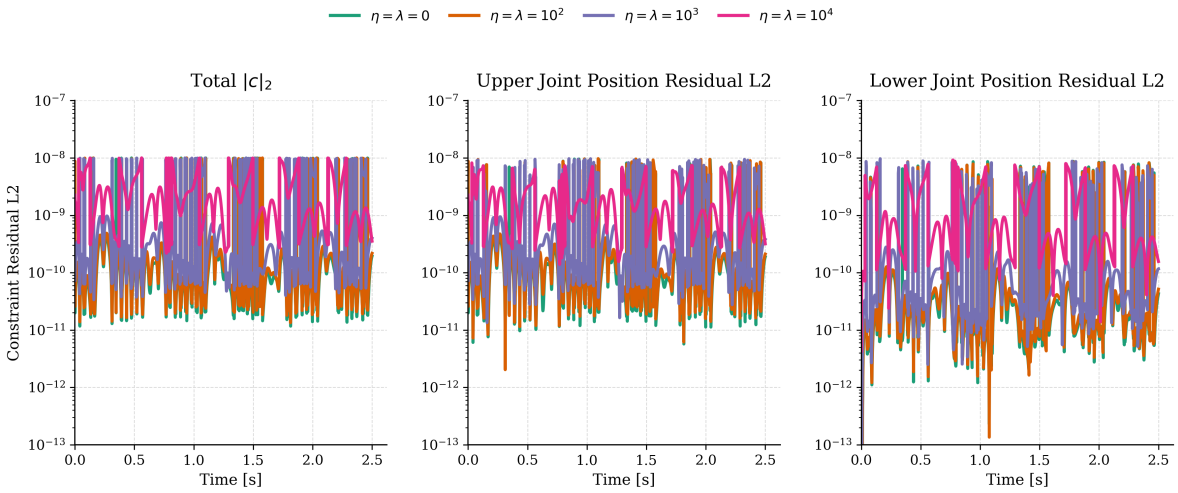


Figure 6.20: Constraint residual histories for the spherical-joint double-pendulum motion test under four damping configurations. Left: total residual $\|\mathbf{c}\|_2$. Center: upper-joint position residual. Right: lower-joint position residual. All residuals remain bounded within approximately 10^{-11} to 10^{-8} with no cumulative drift.

As in the revolute case, a rigid-body counterpart was constructed in Project Chrono [130] to serve as an independent dynamics baseline. The rigid model preserves the same beam dimensions, density-derived mass (0.96 kg), center-of-mass inertia tensor, initial orientations, hinge-point locations, and gravitational loading as the flexible system. For the spherical case, the rigid model uses ball joints at the same connection points so that only point coincidence is enforced while relative rotation remains free, matching the kinematic freedom of the flexible spherical-joint formulation. The comparison therefore isolates the effect of elastic deformation on the three-dimensional pendular dynamics.

Figure 6.21 presents the corresponding motion and recovered joint-force response, with the Project Chrono rigid-body solution included as a reference. The lower-tip trajectories in the x–y and x–z planes exhibit substantial out-of-plane motion, which is the expected behavior for a pendulum connected by spherical joints and clearly distinguishes this case from the planar revolute response. The lightly damped flexible cases trace trajectories broadly

consistent with the rigid-body reference, while the differences that emerge reflect elastic deformation permitted by the flexible formulation; the strongly damped case departs more noticeably as material dissipation reduces the motion envelope. The lower-tip speed histories display repeated oscillatory peaks associated with gravitational acceleration and reversal, and the rigid-body reference again provides a baseline for assessing the influence of beam compliance on the peak speeds and oscillation timing. The recovered joint reaction-force magnitudes remain bounded, and the upper-joint reaction consistently exceeds the lower-joint reaction, as expected because the upper joint supports and redirects the motion of both links. Taken together with the rigid-body comparison, these features support the physical plausibility of the simulated spherical-joint dynamics.

Figure 6.22 shows the energy evolution under the four damping levels. The normalized total energy decreases monotonically in all cases, with progressively faster decay as the damping parameters increase, consistent with material dissipation and implicit time integration. The kinetic, gravitational potential, and elastic strain energies exchange in the oscillatory manner expected of a flexible pendular system. Stronger damping reduces the amplitudes of the kinetic and strain-energy peaks while accelerating total-energy loss. Taken together, Figures 6.20–6.22 indicate that the spherical-joint double pendulum exhibits numerically stable constraint enforcement and physically consistent three-dimensional motion.

6.4.1.4 Double Pendulum with Spherical Joints Vertical Pulling Test

Using the same double-pendulum geometry but replacing both revolute joints with spherical joints, this test examines vertical force transmission in a quasi-static hanging configuration. The purpose of the study is to verify that the ALM formulation recovers physically meaningful spherical-joint reaction forces under distributed loading when the analytical force balance is known. Because an ideal spherical joint constrains relative translation while allowing relative rotation, the primary validation quantity in this case is the recovered joint reaction force rather than an independently transmitted joint moment.

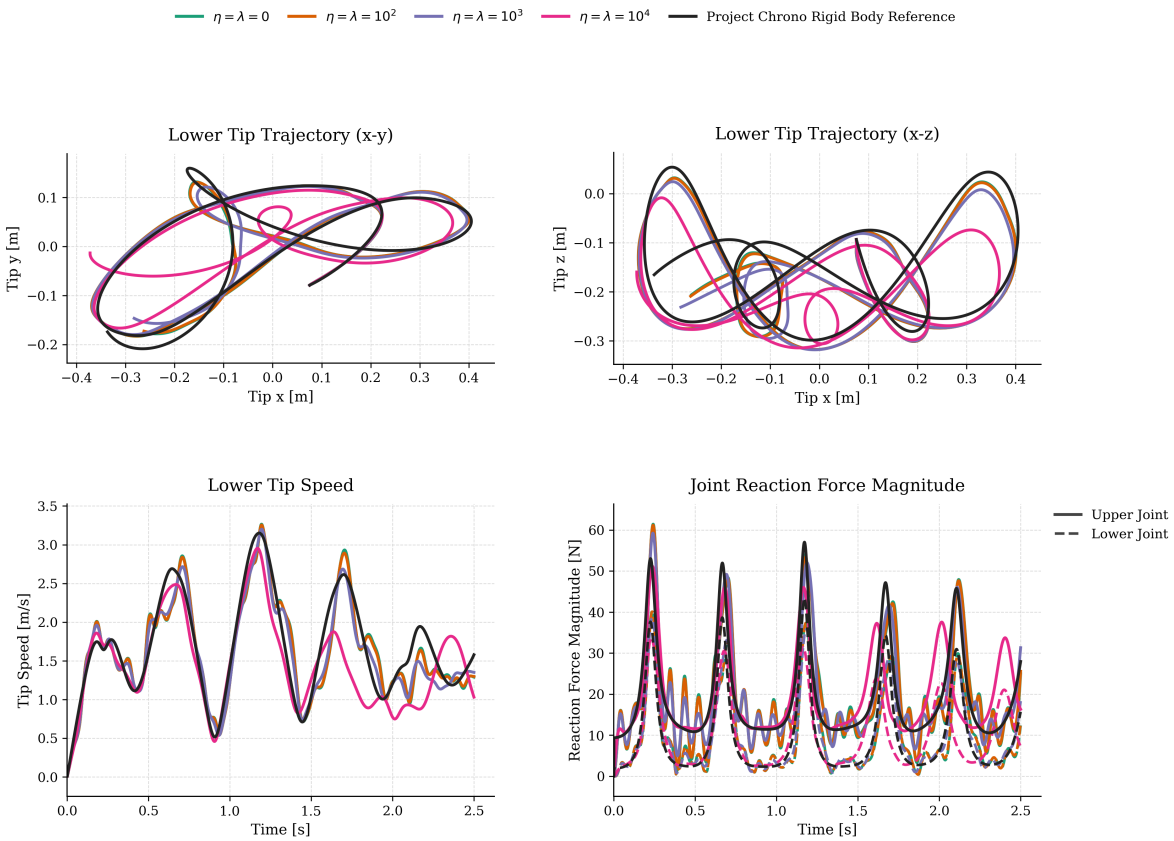


Figure 6.21: Motion and reaction summary for the spherical-joint double-pendulum motion test under four damping configurations, compared against a Project Chrono rigid-body reference. Top left: lower-tip trajectory in the x–y plane. Top right: lower-tip trajectory in the x–z plane. Bottom left: lower-tip speed histories. Bottom right: joint reaction-force magnitudes at the upper (solid) and lower (dashed) spherical joints.

The spherical-joint pendulum is constructed from two identical T10 tetrahedral beam meshes, each modeled with the Saint Venant–Kirchhoff constitutive law and simulation parameters listed in Table 6.18. Both beams are initialized in a perfectly vertical hanging configuration: the upper beam is attached to the world frame by a spherical joint, and the lower beam is connected to the upper beam by a second spherical joint. Gravity acts in the negative z-direction. An additional distributed downward load is applied to the lower-tip region of the second beam; this pull load is ramped linearly over the first 200 time steps and then held constant for the remainder of the simulation. Four loading cases are considered,

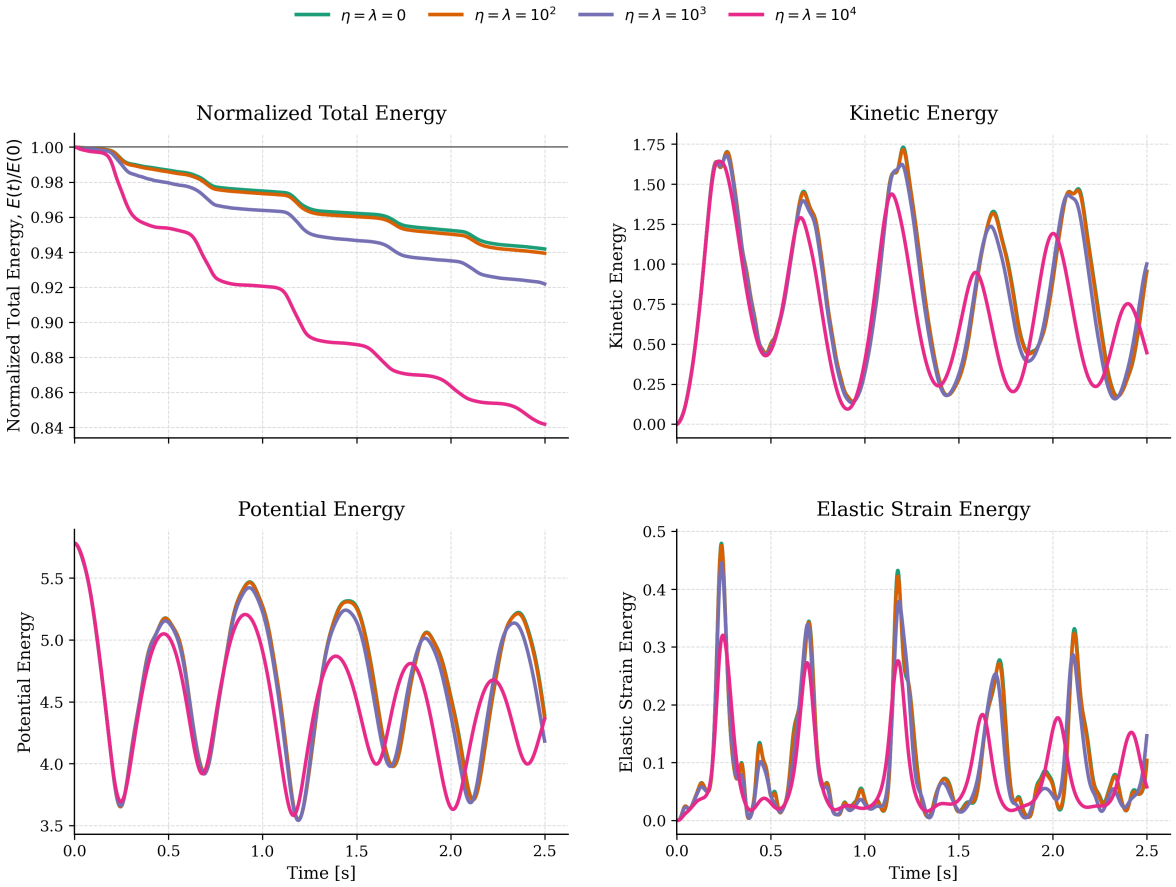


Figure 6.22: Energy response for the spherical-joint double-pendulum motion test under four damping configurations. Top left: normalized total energy $E(t)/E(0)$. Top right: kinetic energy. Bottom left: potential energy. Bottom right: elastic strain energy. Higher damping leads to faster total-energy decay and suppressed oscillation amplitudes.

corresponding to total applied pulls of -20, -40, -60, and $-80\,\mathrm{N}$. Each case is advanced for 1000 time steps using $\Delta t = 5 \times 10^{-4}\,\mathrm{s}$, and the recovered reaction forces at both spherical joints are recorded as time histories for comparison with the known equilibrium values.

For a beam mass of approximately $m_b = 0.96\,\mathrm{kg}$ per link, the expected quasi-static vertical reaction at the lower spherical joint is the weight of one beam plus the applied pull, $F_{z,\mathrm{lower}} = -(m_b g + |F_{\mathrm{pull}}|)$, whereas the expected vertical reaction at the upper spherical joint is the weight of both beams plus the applied pull, $F_{z,\mathrm{upper}} = -(2\,m_b g + |F_{\mathrm{pull}}|)$. With $g = 9.81\,\mathrm{m/s^2}$, these reference values are listed in Table 6.19. The lower-joint reaction

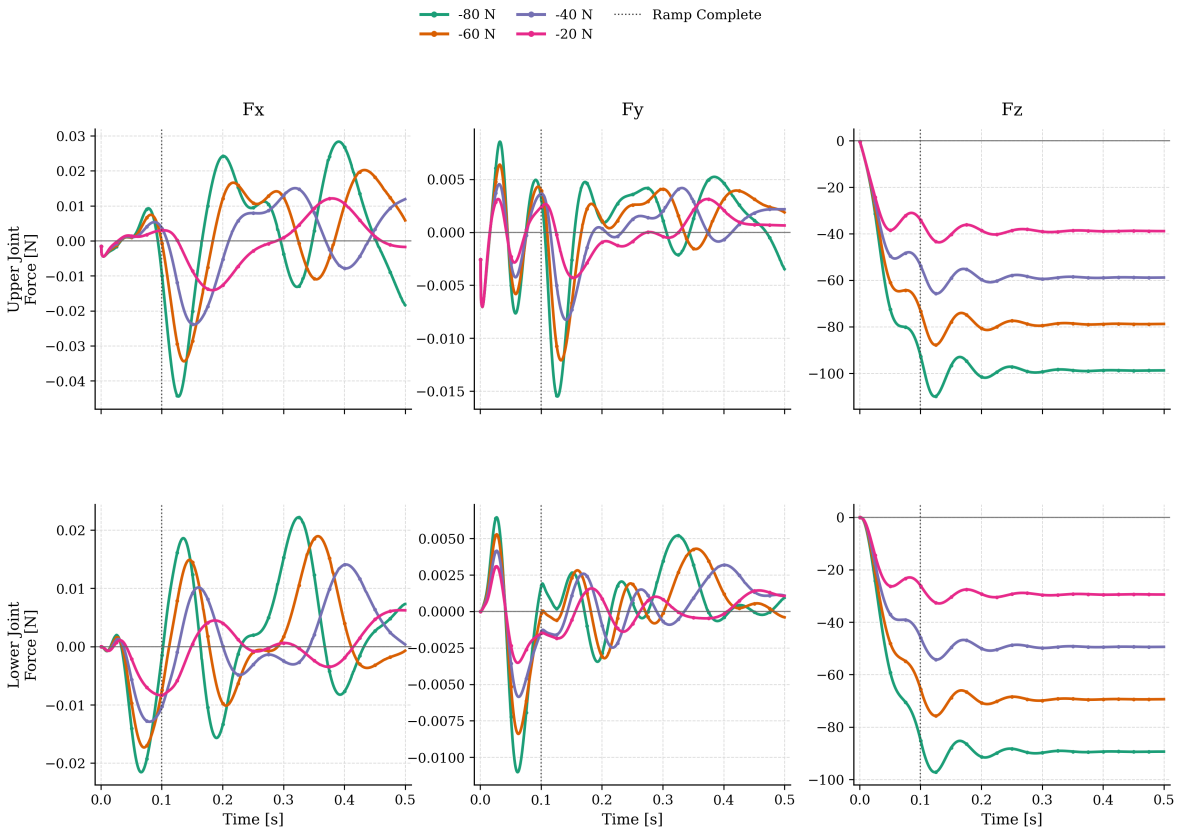
Parameter	Value
Material model	Saint Venant–Kirchhoff (SVK)
Young's modulus, E	$1 \times 10^{7}\,\mathrm{Pa}$
Poisson's ratio, ν	$0.30\,(-)$
Density, ρ_0	$1200\,\mathrm{kg/m^3}$
Shear damping, η_{damp}	$1 \times 10^{4}\,(-)$
Volumetric damping, λ_{damp}	$1 \times 10^{4}\,(-)$
Inner absolute tolerance	$1 \times 10^{-6}\,(-)$
Inner relative tolerance	$1 \times 10^{-6}\,(-)$
Outer tolerance	$1 \times 10^{-8}\,(-)$
ALM ρ	$1 \times 10^{10}\,(-)$
Max outer iterations	$8\,(-)$
Max inner iterations	$10\,(-)$
Time step, Δt	$5 \times 10^{-4}\,\mathrm{s}$
Line search enabled	false

Table 6.18: Material and solver parameters for the spherical-joint double-pendulum vertical pulling test.

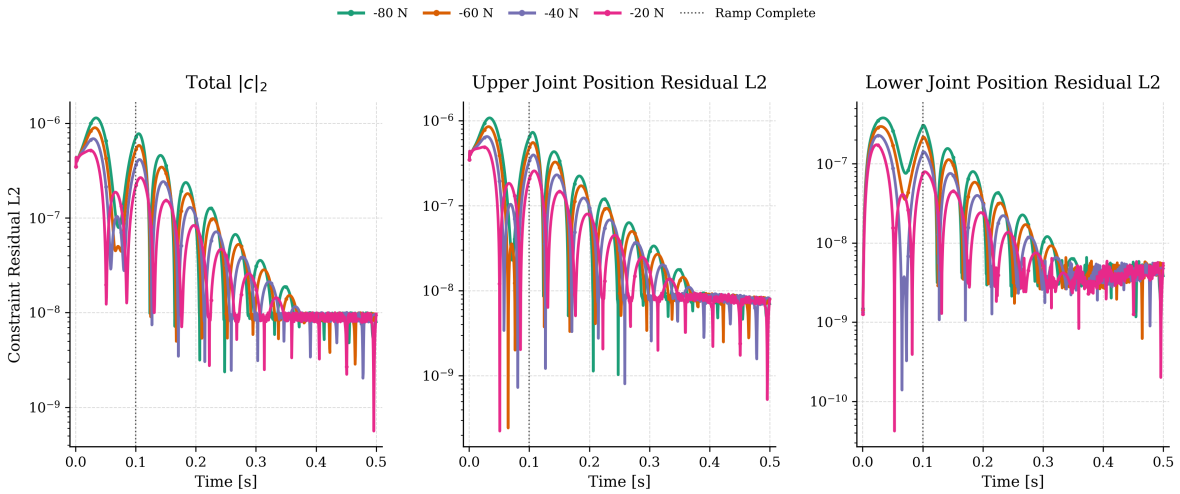
therefore captures the transmitted weight of the lower beam together with the external pull, while the upper-joint reaction additionally includes the weight of the upper beam. Agreement between the recovered ALM reaction forces and these reference values provides a direct check that the spherical-joint constraint formulation is transmitting the correct net support force under vertical loading.

Figure 6.23a presents the time histories of the Cartesian reaction-force components F_x, F_y, and F_z at the upper and lower spherical joints for the four downward loading cases (-20, -40, -60, and $-80\,\mathrm{N}$). As in the revolute-joint benchmark, the response is dominated by the vertical component F_z, whereas the transverse components F_x and F_y remain small throughout the simulation. This is consistent with the vertical hanging configuration and the purely downward distributed pull applied to the lower tip region. The reaction-force histories also show the expected monotonic increase in magnitude with increasing applied load, indicating that the ALM-recovered spherical-joint reaction forces remain physically consistent across the loading range considered here.

Table 6.19 compares the simulated vertical joint reactions at the final time step (step 1000) against the corresponding theoretical estimates based on beam self-weight and the applied pull. The expected lower-joint reaction is computed as the weight of one beam plus the



(a) Time histories of the Cartesian reaction-force components at the upper (top row) and lower (bottom row) spherical joints. The vertical component F_z (right column) dominates and scales with the applied load, while F_x and F_y remain near zero.



(b) Constraint violation histories. Left: total residual $\|\mathbf{c}\|_2$. Center: position residual at the upper joint. Right: position residual at the lower joint. All residuals remain below 10^{-6} on a logarithmic scale.

Figure 6.23: Spherical-joint vertical pulling test results for the four loading cases (-20, -40, -60, and $-80\,\mathrm{N}$). (a) Joint reaction-force component histories. (b) Constraint violation histories.

applied pull, while the expected upper-joint reaction is computed as the weight of both beams plus the applied pull. The small discrepancies observed in Table 6.19 confirm that the ALM-recovered spherical-joint reaction forces remain in very close agreement with the static equilibrium predictions across the full loading range.

Pull Force	Upper Joint F_z				Lower Joint F_z			
[N]	Simulated [N]	Expected [N]	Error [N]	Rel. Err. [%]	Simulated [N]	Expected [N]	Error [N]	Rel. Err. [%]
−20	−38.88	−38.84	0.04	0.10	−29.45	−29.42	0.03	0.10
−40	−58.84	−58.84	0.01	0.02	−49.42	−49.42	0.00	0.00
−60	−78.81	−78.84	0.03	0.04	−69.40	−69.42	0.02	0.03
−80	−98.80	−98.84	0.03	0.03	−89.39	−89.42	0.02	0.02

Table 6.19: Comparison of simulated and expected vertical reaction forces at the upper and lower spherical joints at the final time step (step 1000). Expected values are computed from static equilibrium: lower joint $F_z = -(m_b g + |F_{\text{pull}}|)$, upper joint $F_z = -(2\, m_b g + |F_{\text{pull}}|)$, with $m_b = 0.96\,\text{kg}$ and $g = 9.81\,\text{m/s}^2$. Relative error is $|\text{Error}|/|\text{Expected}| \times 100\%$.

Figure 6.23b reports the constraint residuals for the spherical-joint vertical pulling test. Because each spherical joint is composed entirely of three CD constraints (Table 3.1), only position (coincidence) residuals are present; no orientation constraints are involved. The residuals remain small, bounded, and systematically decaying for all four loading cases, confirming stable and accurate coincidence enforcement throughout the simulation. The total residual $\|\mathbf{c}\|_2$ begins at approximately 3.5×10^{-7} and reaches peak values between roughly 5.2×10^{-7} and 1.14×10^{-6} as the applied load increases from -20 to $-80\,\text{N}$; it subsequently decays to a common late-time level on the order of 10^{-8} after the ramped loading phase, with no evidence of monotonic drift. The upper-joint position residual closely tracks the total residual and constitutes the dominant contribution to the overall constraint error, whereas the lower-joint position residual remains consistently smaller, with peak magnitudes between approximately 1.7×10^{-7} and 3.8×10^{-7} and final values below 6×10^{-9}. A modest load dependence is visible in the transient peaks, particularly for the upper joint, but the residuals remain several orders of magnitude below unity in all cases. These results are consistent with robust numerical enforcement of the spherical-joint coincidence constraints by the augmented Lagrangian outer loop (Algorithm 6) under progressively increasing external loading.

6.4.2 Piston-in-Cup with Cylindrical Joint

This test exercises the cylindrical-joint formulation introduced in Section 3.6 in a practically motivated setting. A deformable piston shaft is inserted into a deformable cylindrical cup; two cylindrical joints placed at the 25% and 75% axial positions of the shaft-bore overlap enforce the constraint that the piston may slide along and rotate about the shared axis, while off-axis translation is prevented. The cup is fixed at its outer bottom annulus; the piston is loaded laterally by a 30 N force applied to its handle bar, ramped linearly over the first ten time steps and then held constant. Both bodies are discretized with T10 quadratic tetrahedral elements. The material and solver parameters are listed in Table 6.20.

Parameter	Cup	Piston
Material model	Saint Venant–Kirchhoff (SVK)	
Young's modulus, E	1×10^7 Pa	5×10^7 Pa
Poisson's ratio, ν	$0.32\,(-)$	$0.32\,(-)$
Density, ρ_0	$1150\,\mathrm{kg/m^3}$	$1750\,\mathrm{kg/m^3}$
Shear damping, η_{damp}	$2 \times 10^4\,(-)$	$5 \times 10^3\,(-)$
Volumetric damping, λ_{damp}	$2 \times 10^4\,(-)$	$5 \times 10^3\,(-)$
Solver (shared)		
Inner abs./rel. tolerance		$1 \times 10^{-5}\,(-)$
Outer tolerance		$1 \times 10^{-10}\,(-)$
ALM ρ		$1 \times 10^{12}\,(-)$
Max outer iterations		$12\,(-)$
Max inner iterations		$8\,(-)$
Time step, Δt		$5 \times 10^{-4}\,\mathrm{s}$
Total steps		$400\,(-)$
Line search enabled		true

Table 6.20: Material and solver parameters for the piston-in-cup cylindrical-joint test. The piston is five times stiffer than the cup, making both bodies meaningfully deformable under the applied lateral load.

Each cylindrical joint is assembled from two dot-product-1 (DP1) parallelism constraints and two dot-product-2 (DP2) collinearity constraints, yielding four scalar constraint rows per joint and eight rows in total for the coupled system. Following Section 3.6.2.3, each joint is defined by a seven-point geometry (p, q, r, s, u, v, w) interpolated from element shape functions. The *cup guide frame* supplies four points: p (axis base), q (axis probe, offset $\delta = 0.006\,\mathrm{m}$ along Z), and two perpendicular probes v and w offset from p by δ in the two

directions normal to the axis. The piston contributes r (axis attachment at the target Z height) and two radial offsets s, u from r on the shaft surface, also separated by δ. The four constraints per joint are:

$$\begin{aligned}
\text{DP1-a:} \quad & (\mathbf{r}_q - \mathbf{r}_p) \cdot (\mathbf{r}_s - \mathbf{r}_r) = 0, \\
\text{DP1-b:} \quad & (\mathbf{r}_q - \mathbf{r}_p) \cdot (\mathbf{r}_u - \mathbf{r}_r) = 0, \\
\text{DP2-a:} \quad & (\mathbf{r}_v - \mathbf{r}_p) \cdot (\mathbf{r}_r - \mathbf{r}_p) = 0, \\
\text{DP2-b:} \quad & (\mathbf{r}_w - \mathbf{r}_p) \cdot (\mathbf{r}_r - \mathbf{r}_p) = 0.
\end{aligned}$$

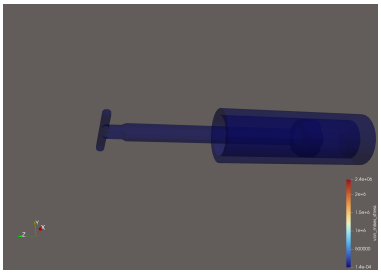
The DP1 rows require the piston radial fibers $(s{-}r)$ and $(u{-}r)$ to be perpendicular to the cup axis direction $(q{-}p)$; the DP2 rows require the connector $(r{-}p)$ to lie in the plane through p perpendicular to each cup off-axis probe, enforcing lateral confinement of the piston axis point r to the cup axis.

Row normalization and DP2 weight compensation. Each scalar row is normalized by $w_i = 1/\sqrt{|\mathbf{a}|^2 + |\mathbf{d}|^2}$ (Eq. (3.79)). For the DP1 rows both vectors are $O(\delta)$, giving $w_{\text{DP1}} \approx 1/(\delta\sqrt{2})$. For the DP2 rows, $|v{-}p| \approx \delta$ but $|r{-}p| \approx \Delta z$ (the axial separation between attachment points), so the raw normalization weight $\approx 1/\sqrt{\delta^2 + \Delta z^2}$ would underweight the DP2 Jacobian sensitivity relative to the DP1 rows. A compensation factor

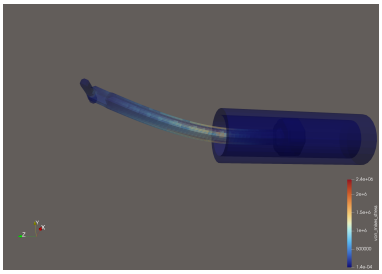
$$w_{\text{dp2}} = \frac{\sqrt{\delta^2 + \Delta z^2}}{\delta} \tag{6.8}$$

is therefore applied to each DP2 row, restoring an effective weight of $1/\delta$ — the same scale as the DP1 rows. Because the lower and upper joints have different axial separations Δz, two independent w_{dp2} values are computed. Figure 6.24 shows three simulation snapshots at representative instants.

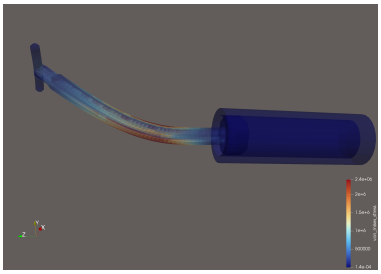
Figure 6.25 shows the cylindrical constraint violation histories over the 400-step simulation. The total constraint norm starts at $O(10^{-12})$ and reaches a peak of $\approx 2.7 \times 10^{-10}$ during the



(a) Initial configuration: piston inserted into cup, no lateral load.



(b) Mid-simulation: piston deflecting laterally under the sustained 30 N load.



(c) Final configuration (step 400): piston has slid axially by $\approx$ 0.175 m, under large deformation; cylindrical constraint maintained throughout.

Figure 6.24: von Mises stress field at three instants of the piston-in-cup cylindrical-joint test. The piston slides freely along the shared axis while the cylindrical constraint suppresses off-axis motion throughout.

simulation. The aggregate $\|\mathbf{c}\|_2$ exceeds the per-row ALM outer tolerance of 10^{-10} simply because it accumulates contributions from all eight constraint rows; the individual per-row residuals remain well within that tolerance throughout. The DP1 parallelism residual is tighter, peaking at $\approx 2\times10^{-11}$, while the DP2 collinearity residual is the dominant contributor to the total norm at $\approx 2.7 \times 10^{-10}$. The DP2 weight compensation (Eq. (6.8)) equalizes the Jacobian sensitivity at the reference configuration, but Δz changes continuously as the piston slides, so the compensation becomes approximate and the DP2 rows carry a somewhat larger effective residual than the DP1 rows throughout the simulation. The radial drift of the piston axis midpoint from the deformed cup axis reaches a maximum of $\approx 1.2\,\mu$m at step 400, with per-joint values of $0.99\,\mu$m and $1.32\,\mu$m at the lower and upper joints, respectively. These sub-micron drift values confirm that the cylindrical joint maintains coaxial alignment throughout the simulation despite sustained lateral loading on both deformable bodies.

Figure 6.26 presents the recovered joint reaction force F_y, reaction moment M_x, and piston axial displacement over the 400-step simulation. The reaction force exhibits a pronounced oscillatory transient with a period of approximately 0.13 s: F_y overshoots to ≈ 69 N near step 149 ($t \approx 0.075$ s), descends to ≈ -11.6 N near step 277 ($t \approx 0.139$ s), and returns to

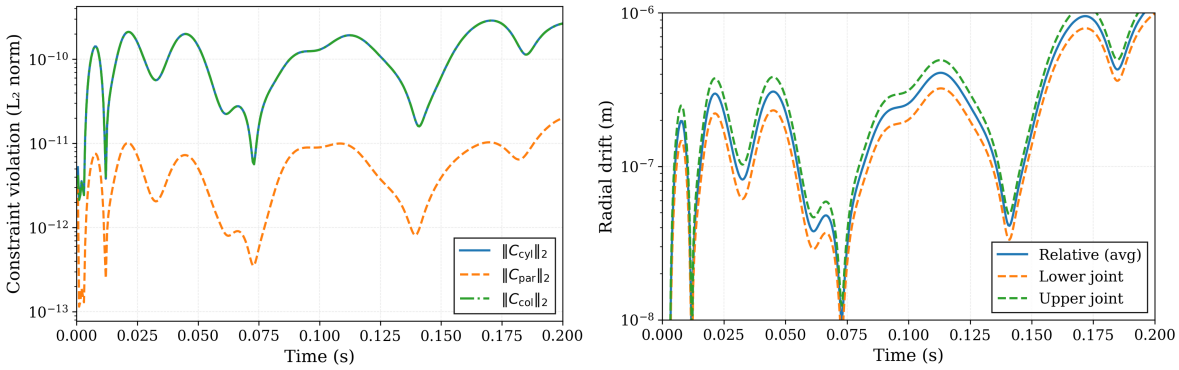


Figure 6.25: Constraint violation and radial drift histories for the piston-in-cup cylindrical-joint test. Left: total constraint norm $\|\mathbf{c}\|_2$ together with the DP1 (parallelism) and DP2 (collinearity) component norms on a logarithmic scale; all individual per-row residuals remain within the outer ALM tolerance (10^{-10}), while the aggregate $\|\mathbf{c}\|_2$ reaches $\approx 2.7 \times 10^{-10}$ owing to accumulation across eight rows. Right: radial drift of the piston axis midpoint from the deformed cup axis (total and per-joint); maximum drift is $\approx 1.2\,\mu$m at step 400.

$\approx 67.2\,$N at step 400. This behavior is physically expected: the piston, with $E = 50\,$MPa, stores and releases elastic energy as it bends under the suddenly applied lateral load, and the applied Kelvin-Voigt damping is insufficient to critically damp the fundamental bending mode within the $0.2\,$s simulation window. The reaction moment $M_x \approx -19.6\,\mathrm{N}\cdot\mathrm{m}$ at step 400 reflects bending about the x-axis consistent with a $+y$ load applied above the joint midpoint. Throughout the oscillation, the piston slides axially from $z \approx 0.235\,$m to $z \approx 0.411\,$m (a net displacement of $\approx 0.175\,$m), confirming that the cylindrical joint correctly permits free axial translation while enforcing off-axis coaxiality.

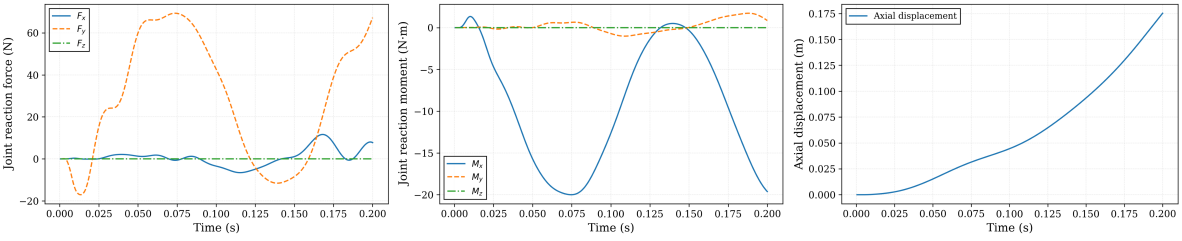


Figure 6.26: Recovered joint wrench and piston kinematics for the piston-in-cup cylindrical-joint test. Left: joint reaction force F_y recovered from ALM multipliers. Center: joint reaction moment M_x. Right: piston axial displacement Δz showing free sliding along the permitted translational degree of freedom.

6.5 Large Scale Test

6.5.1 Vase Drop on Protective Foam

This benchmark examines the transient interaction between a brittle ceramic vase and a compliant protective foam insert containing a matching hollow cavity. The problem is intended to represent a packaging scenario in which a stiff, fragile object is supported by a deformable enclosure and then subjected to motion that produces repeated contact, sliding, and localized impact. The setup is motivated by protective packing of ceramics and glass artifacts. Conservation guidance from the National Park Service notes that polyethylene foam is commonly used for cavity-packing many stable three-dimensional objects, whereas for more vulnerable objects such as ceramics or glass, a softer urethane ester-type foam may be preferred in order to reduce transmitted shock [137]. The present benchmark is therefore intended to capture the essential mechanics of a fragile ceramic object interacting with a compliant protective insert under dynamic disturbance.

The computational model consists of two deformable bodies, namely a bouquet vase and a protective foam insert. Both bodies are discretized with 10-node quadratic tetrahedral elements using Tetgen-format volumetric meshes. The vase mesh is translated by $(0,\ 0.025,\ 0.010)\,\mathrm{m}$ relative to its reference position prior to the start of the simulation so that it is initially positioned above the cavity and can settle into contact under gravity. Surface triangle meshes are extracted automatically from the volumetric discretizations and are used exclusively for contact detection and force transfer. This separation between the volumetric finite element representation and the contact surface representation makes it possible to preserve higher-order volumetric interpolation while employing a dedicated mesh-based collision pipeline at the interface.

The two bodies are assigned distinct constitutive models because they play different mechanical roles in the benchmark. The vase is modeled as a Saint Venant–Kirchhoff material

with parameters

$$E = 50\,\mathrm{GPa}, \qquad \nu = 0.25, \qquad \rho_0 = 2400\,\mathrm{kg/m^3}.$$

This choice is intended as a first-order approximation of a porcelain-like or dense ceramic body whose strains remain small relative to those of the surrounding foam. The adopted elastic constants are consistent with representative ranges reported for fired clay ceramics and ceramic-like materials: sintered illitic clay systems can reach Young's moduli on the order of 30 to 70 GPa, and reviews of ceramic elastic properties commonly place Poisson's ratio in the range 0.25 to 0.35 [138, 139]. The Saint Venant–Kirchhoff model is not intended to represent ceramic fracture. Rather, it provides a computationally efficient elastic description for the purpose of comparing transmitted stress levels under different insert materials. Because the foam undergoes much larger deformation than the vase, this partition of modeling complexity is appropriate for the present benchmark.

The protective insert is modeled using a compressible two-parameter Mooney–Rivlin hyperelastic law. This choice is motivated by the need to represent finite deformation, repeated loading, and large contact-induced shape changes in the insert. Hyperelastic strain-energy formulations of the Mooney–Rivlin type are standard for finite-strain rubber-like response and provide a convenient framework for comparing materials with different compliance levels through a consistent constitutive form [140]. At the same time, the present study is not intended to reproduce all aspects of real packaging foams, such as plateau crushing, damage, or densification. Instead, the Mooney–Rivlin model is used here as a surrogate large-deformation constitutive law that allows the insert stiffness to be varied in a controlled manner while maintaining numerical tractability. This modeling choice is especially useful in the present contact-rich setting, where fully near-incompressible formulations can be difficult to solve robustly, whereas a compressible hyperelastic surrogate permits systematic comparison across candidate insert materials within a common numerical framework [140].

Five insert presets are considered in the study. Two neoprene presets and one polyurethane preset are drawn from published Mooney–Rivlin characterizations of elastomeric materials

Table 6.21: Foam material presets used in the vase–insert benchmark.

Material	C_{10} (MPa)	C_{01} (MPa)	Bulk / compressibility	ρ_0 (kg/m^3)	Ref.
Neoprene 50A	0.302	0.076	$\nu = 0.49$	1350	[141, 144]
Neoprene 60A	0.382	0.096	$\nu = 0.49$	1400	[141, 145]
Polyurethane 50A	0.302	0.076	$\nu = 0.499$	2000	[142]
EVA 80	0.417*	0.104*	$D_1 = 0.736\,\mathrm{MPa}^{-1}$ *	80	[143]
EVA 95 (default)	0.641*	0.160*	$D_1 = 0.478\,\mathrm{MPa}^{-1}$ *	95	[143]

* Study-defined compressible Mooney–Rivlin surrogate calibration based on EVA compression data; not reported directly by Chen et al.

[141, 142]. In addition, two EVA-based presets are included to represent a softer and a stiffer foam-like response associated with different nominal foam densities. Experimental compression data for closed-cell EVA foams show that the effective stiffness increases with density; in particular, the measured quasi-static modulus at 95 kg/m^3 is higher than that at 80 kg/m^3 [143]. In the present benchmark, the EVA presets are therefore interpreted as compressible Mooney–Rivlin surrogate calibrations chosen to preserve this experimentally observed density-dependent compliance trend.

From a packaging standpoint, the selected insert materials span a range from softer elastomeric cushioning to comparatively firmer foam-like support. This is consistent with conservation practice, in which both the cavity geometry and the compliance of the support medium are adjusted according to the fragility of the protected object and the anticipated transport or handling environment [137]. Accordingly, the material set used here is not intended to reproduce one specific commercial packing foam; rather, it provides a controlled compliance ladder for studying how insert stiffness and compressibility influence the stress transmitted to a fragile ceramic body.

The foam presets used in this study are listed in Table 6.21.

In Table 6.21, the neoprene C_{10}, C_{01}, and ν values are retained from Han and Che [141], whereas the densities are replaced by literature-backed solid-neoprene sheet values:

1350 kg/m^3 for 50A [144] and 1400 kg/m^3 for 60A [145]. The polyurethane 50A entry is kept unchanged from Mamashli et al. [142]. For EVA 80 and EVA 95, the densities of 80 and 95 kg/m^3 follow Chen et al. [143], but the listed C_{10}, C_{01}, and D_1 values are not reported directly there; they are study-defined compressible Mooney–Rivlin surrogate calibrations introduced here to represent the experimentally observed density-dependent compression trend.

Rayleigh-type viscous damping is applied to the insert through the coefficients $\eta_{\text{damp}} = \lambda_{\text{damp}} = 5 \times 10^3\,\text{Pa} \cdot \text{s}$. For the elastomer-type presets, the volumetric response is represented through assigned Poisson ratios, whereas for the EVA-type presets it is specified through the compressibility parameter D_1 listed in Table 6.21. This permits the volumetric response to be tuned independently of the deviatoric Mooney–Rivlin coefficients, which is useful for balancing material realism against solver robustness.

Each body is instantiated as an independent GPU-resident FEAT10 finite element block. Volume integration is performed with a 5-point tetrahedral Gauss quadrature rule. The consistent mass matrix is assembled for each block and then row-summed to obtain a lumped mass representation for time integration. This arrangement permits the vase and the foam to retain distinct constitutive laws, densities, and damping parameters while remaining coupled through contact forces in a shared dynamic solve. Dirichlet boundary conditions are applied to all foam nodes satisfying $z < -0.09\,\text{m}$, thereby representing a rigidly supported base at the bottom of the foam block. No essential boundary conditions are imposed directly on the vase, which remains free except for gravity and contact interaction with the insert.

Inter-body contact is handled through a DEME-based mesh-mesh collision system operating on the extracted surface triangle meshes. Contact forces are computed with a penalty stiffness model characterized by contact stiffness $E_c = 8 \times 10^6\,\text{Pa}$, coefficient of restitution 0.1, static friction coefficient $\mu_s = 0.6$, and kinetic friction coefficient $\mu_k = 0.5$. At each time step, the contact algorithm identifies interacting surface regions, evaluates the corresponding normal and tangential contact forces, and accumulates the resulting nodal contributions into

the external force vector before the nonlinear equilibrium solve is performed. Gravity acts simultaneously on both bodies throughout the full simulation.

The coupled system is advanced using an implicit time integration scheme with fixed time step $\Delta t = 1 \times 10^{-4}\,\mathrm{s}$. The total simulated duration is 0.6 s, corresponding to 6000 time steps. The resulting nonlinear algebraic system is solved using a multi-block augmented Lagrangian Newton procedure with inner absolute tolerance 10^{-3}, inner relative tolerance 10^{-4}, outer tolerance 10^{-6}, penalty parameter $\rho = 10^{12}$, maximum of five outer iterations, maximum of ten inner iterations, and line search enabled. This solver configuration is intended to accommodate the large material contrast between the vase and the insert, repeated changes in contact state, and the geometric nonlinearity induced by deformation of the foam.

The loading history is divided into three consecutive phases. During the first phase, corresponding to steps 0 through 999 and time interval $t \in [0, 0.1]\,\mathrm{s}$, only gravity is applied. This pre-settling stage allows the vase to descend into the cavity and establish an initial supported configuration under its own weight. During the second phase, corresponding to steps 1000 through 4999 and time interval $t \in [0.1, 0.5]\,\mathrm{s}$, the fixed base nodes of the foam are subjected to a prescribed displacement in the x-direction following a triangular velocity waveform. The base motion has displacement amplitude $A_x = 5\,\mathrm{mm}$ and peak speed $v_x = 0.2\,\mathrm{m/s}$, which yields a period $T = \frac{4A_x}{v_x} = 0.1\,\mathrm{s}$. This imposed lateral motion excites relative sliding, intermittent separation, and repeated re-contact between the vase and the cavity wall. During the third phase, corresponding to steps 5000 through 5999 and time interval $t \in [0.5, 0.6]\,\mathrm{s}$, the prescribed base motion is removed and the system is allowed to relax again under gravity alone.

Representative von Mises stress snapshots for the EVA80 insert case are shown in Figure 6.27. The two columns provide a three-quarter (45°/45°/45°) view and an underside view looking upward along the positive z-axis. The selected times correspond to three mechanically distinct stages of the motion: $t = 0.08\,\mathrm{s}$ captures the post-impact configuration before the vase has fully settled into the cavity, $t = 0.10\,\mathrm{s}$ captures the supported configuration immediately

before the prescribed lateral shake begins, and $t = 0.25\,\mathrm{s}$ captures a representative instant during the shaking phase when contact redistribution and stress localization are actively evolving.

The primary objective of the benchmark is to compare how the choice of insert material affects the stress state that develops in the vase. Because the vase is the fragile component of interest, post-processing focuses on vase stress histories rather than on foam deformation alone. In particular, several complementary response measures are extracted from the vase mesh over the shaking interval $t \in [0.1, 0.5]\,\mathrm{s}$: the whole-body average von Mises stress, the average von Mises stress over the top 10% highest-stress cells, the average von Mises stress over the top 1% highest-stress cells, and two temporal tail metrics obtained from the smoothed top-1% stress history by averaging the highest 1% and highest 2% of timesteps, respectively. The use of both global and tail-oriented measures makes it possible to distinguish between broadly distributed loading and short-duration hotspot events. This distinction is relevant in protective packaging problems because a more compliant insert can reduce the average transmitted stress while still permitting intermittent localized re-contact events of relatively high magnitude.

The resulting summary metrics are reported in Table 6.22, and the corresponding time histories are shown in Figure 6.28. Figure 6.28 shows the whole-body average vase von Mises stress, the average von Mises stress over the top 10% of cells, and the average von Mises stress over the top 1% of cells for the five material presets. Repeated peaks are observed throughout the shaking interval, reflecting intermittent sliding and re-contact between the vase and the cavity wall.

Taken together, Table 6.22 and Figure 6.28 indicate a consistent ordering in the interval-averaged stress measures. The whole-body average von Mises stress is lowest for EVA80, followed by EVA95, then the two neoprene cases, with polyurethane 50A producing the largest value. The same overall trend is also observed for the top-10% and top-1% spatial hotspot measures. This ordering suggests that, for the present cavity-constrained contact configuration,

Table 6.22: Summary of vase stress measures over the shaking interval $t \in [0.1, 0.5]$ s. All values are reported in Pa. "Top 10%" and "Top 1%" are spatial hotspot measures: at each timestep, vase cells are ranked by von Mises stress, the highest-stress 10% or 1% of cells are selected, and their average stress is computed; these timestep-wise values are then averaged over the interval. "Tail 1%" and "Tail 2%" are temporal upper-tail measures based on the smoothed Top 1% stress history, obtained by averaging the highest 1% and 2% of timesteps, respectively.

Preset	Avg. VM	Top 10%	Top 1%	Tail 1%	Tail 2%
EVA80	9697.80	36445.72	93544.97	771887.17	714610.95
EVA95	10365.37	38968.28	101700.19	626119.07	558398.03
Neoprene 50A	10760.57	41873.86	123395.72	491121.74	467566.59
Neoprene 60A	11265.17	44346.20	133391.12	544109.15	525576.90
Polyurethane 50A	11578.38	46136.62	142849.40	583370.92	556895.68

the more compressible EVA-based surrogates reduce the typical stress transmitted to the vase relative to the near-incompressible elastomeric presets. Within the EVA family, EVA95 produces larger stresses than EVA80 in all three interval-averaged measures, which is consistent with the higher assigned stiffness of the EVA95 surrogate. The difference between the neoprene 50A and neoprene 60A cases is comparatively small, but neoprene 60A produces slightly larger averaged stress measures than neoprene 50A, which is consistent with its higher nominal stiffness.

A different pattern is observed in the temporal upper-tail metrics reported in the last two columns of Table 6.22. When the analysis is restricted to the highest 1% and highest

2% of timesteps according to the smoothed top-1% hotspot stress history, EVA80 produces the largest values, followed by EVA95. The elastomeric cases, which produce larger interval-averaged stresses overall, fall below the EVA cases in these temporally conditioned metrics. This indicates that EVA80 yields the lowest typical stress levels over the shaking interval as a whole, but also the most severe intermittent hotspot events during a relatively small subset of timesteps. A mechanically plausible interpretation is that the softer EVA80 insert filters much of the baseline loading but permits larger excursions of the vase before re-contact, thereby generating sharper localized impacts during a limited number of extreme events. By contrast, the less compressible elastomeric inserts produce larger sustained stress levels but a less pronounced temporal tail amplification.

These observations suggest that the choice of protective insert material influences the vase response in at least two distinct ways. First, increasing the effective compliance of the insert lowers the average stress transmitted to the vase over the shaking interval. Second, excessive compliance can increase the severity of rare re-contact events, which become visible only when the analysis is restricted to the temporal upper tail of the hotspot stress history. In this sense, the EVA80 configuration appears favorable with respect to mean transmitted stress, but less favorable with respect to extreme-event hotspot severity. The EVA95 configuration occupies an intermediate position, providing somewhat higher average stress than EVA80 while also reducing the temporal tail amplification. The elastomeric cases, especially polyurethane 50A, exhibit the largest average and hotspot stress levels, indicating that greater resistance to local compression under confinement leads to higher sustained transmitted loading.

Taken together, the benchmark provides a controlled contact-rich setting in which geometry, contact parameters, loading history, and solver configuration are fixed, while the insert material is varied systematically. The ceramic vase is represented as a stiff elastic body with parameters characteristic of a porcelain-like material, whereas the insert materials range from softer elastomeric response to relatively firmer foam-like response. The inclusion of both neoprene and EVA-type presets reflects two relevant aspects of protective packaging, namely

elastomer-like compliance at finite strain and density-dependent foam stiffness. The resulting comparison isolates the influence of cushioning compliance on the stress transmitted to the vase during gravity settling, lateral excitation, and subsequent relaxation, while also serving as a representative demonstration of multi-body finite-strain contact simulation in the present computational framework.

6.5.2 Mixed Item Dropping

This benchmark examines a cluttered, large-scale contact scenario in which nine deformable tire bodies are dropped into a fixed open container and allowed to interact through repeated impact, rolling, and re-contact with one another, with the container walls, and with a compliant cantilevered ANCF3443 shell beam positioned inside the box as a deformable support surface. Unlike the controlled two-body vase–foam interaction of Section 6.5.1, this configuration is designed primarily as a stress test of the complete simulation pipeline: it simultaneously exercises the multi-block augmented Lagrangian Newton solver, the asynchronous GPU collision detection pipeline described in Chapter 4, and the coupled ANCF3443 shell element and T10 tetrahedral solid element formulations under rapidly evolving many-body contact with mixed element discretizations.

Problem setup. The container is a rigid open box with all nodes fixed throughout the simulation, discretized with 4,474 nodes and 2,170 T10 tetrahedral elements. A cantilevered ANCF3443 shell beam with planform dimensions $0.40 \times 0.40\,\mathrm{m}$ and thickness $0.02\,\mathrm{m}$ is meshed with 30×12 shell elements (360 elements, 403 nodes, 4,836 displacement DOFs) and clamped along one longitudinal edge, which fixes 156 DOFs. The shell beam is positioned inside the container as a deformable support surface so that descending tire bodies interact with it on contact.

Nine deformable tire bodies, each discretized with the same T10 tetrahedral mesh introduced in Section 6.1.4 (34,414 nodes and 17,167 elements per tire), are arranged in two groups:

six tires in an upper cluster positioned near the beam surface, and three additional tires in a lower row so that the descending upper bodies also interact with an already populated region of the container. This arrangement is designed to exercise the contact pipeline under sustained many-body crowding rather than simply a first-impact event.

The combined system contains 314,603 nodes and 947,436 total displacement DOFs, of which 13,578 are constrained (container walls and beam clamp), leaving 933,858 free degrees of freedom—a nearly one-million-DOF nonlinear transient contact problem with two coupled element technologies operating simultaneously. Mesh statistics for all components are collected in Table 6.23. Material parameters for the tire bodies and the shell beam are given in Table 6.24, and the solver, time-integration, and contact settings are summarized in Table 6.25. Both subsystems are modeled with a Saint Venant–Kirchhoff constitutive law with Kelvin–Voigt viscous damping, and the SVK model is chosen here because the primary purpose of the test is to stress the contact pipeline and solver robustness rather than to study material nonlinearity. Representative snapshots of the simulation at three mechanically distinct instants are shown in Figure 6.29.

Table 6.23: Mesh statistics for the mixed item-dropping benchmark. The open box is treated as rigid; all its nodes are constrained throughout.

Component	Nodes	Elements	DOFs	Constrained DOFs
ANCF3443 shell beam	403	360	4,836	156
T10 tire (×9)	309,726	154,503	929,178	—
T10 open box	4,474	2,170	13,422	13,422
Total	314,603	157,033	947,436	13,578

Diagnostics over the first 4,500 steps. The following analysis focuses on the first 4,500 time steps (0.45 s of physical time), which capture the transition from a pre-contact free-fall phase to a sustained multi-contact settling regime. The full simulation runs for 50,000 steps (5.0 s total), but the early window is the most diagnostically informative because it contains the contact-onset transient and the subsequent buildup toward a crowded steady state. The

Table 6.24: Material parameters for the mixed item-dropping benchmark. Both subsystems use the Saint Venant–Kirchhoff (SVK) constitutive model with isotropic Kelvin–Voigt viscous damping ($\eta_{\text{damp}} = \lambda_{\text{damp}}$).

Parameter	Tire bodies (×9)	Shell beam
E (Pa)	5.0×10^6	8.0×10^6
ν	0.40	0.33
ρ_0 (kg/m^3)	900	1200
$\eta_{\text{damp}} = \lambda_{\text{damp}}$ (Pa s)	2.0×10^4	2.0×10^4

Table 6.25: Solver, time-integration, and contact parameters for the mixed item-dropping benchmark.

Category	Parameter	Value
Time integration	Δt	1×10^{-4} s
	Steps (total)	50,000 (5.0 s)
	Gravity	-9.81 m/s^2
Nonlinear solve	Inner absolute tolerance	1×10^{-4}
	Inner relative tolerance	1×10^{-6}
	Outer tolerance	1×10^{-5}
	Line search	enabled
	ALM penalty ρ	1×10^{14}
	Max outer iterations	5
	Max inner iterations	20
	Solver blocks	2
Contact	μ_s	0.6
	μ_k	0.5
	Contact stiffness	1×10^7 Pa/m
	Restitution coefficient	0.5
	Self-collision	disabled

corresponding overview diagnostics are shown in Figure 6.30 and convergence metrics in Figure 6.31.

No contacts are detected until step 1,455, after which the active contact count rises rapidly and remains nonzero for approximately 67.6% of the analyzed interval. The contact population reaches a maximum of 23 active contacts at step 2,620, indicating that the most geometrically crowded configuration occurs in the middle of the analyzed window rather than at first impact. This behavior is consistent with the intended role of the test as a cluttering

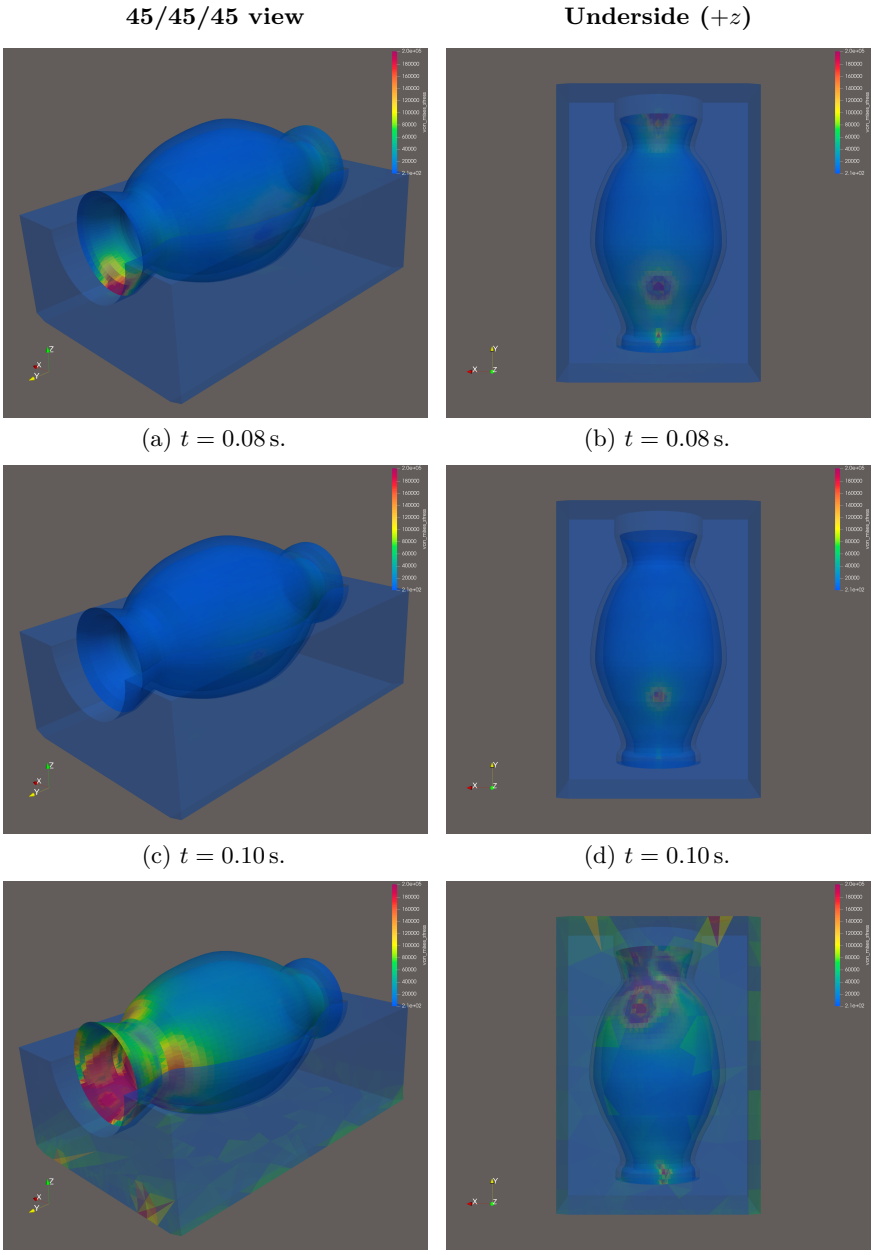
experiment: the dominant challenge is not the first touchdown event, but the subsequent sequence of rearrangements as bodies compete for space inside the container.

Contact onset produces a marked increase in nonlinear-solve cost without loss of numerical control. Excluding the initialization step, the mean solver wall time rises from approximately 645 ms before first contact to approximately 1,969 ms afterward—a roughly threefold increase. The mean inner-iteration count follows the same trend, growing from 5.34 in the pre-contact regime to 7.91 during impact buildup, 8.52 in the dense-contact interval (steps 2,500–3,499), and 8.93 during late rearrangement (steps 3,500–4,499). Notably, the most expensive steps do not coincide with the maximum contact count: the largest non-initialization solver wall time occurs at step 4,487 (3,781.9 ms) with only 10 active contacts, whereas the contact-count peak at step 2,620 is comparatively less costly. This observation indicates that solver difficulty is governed more by unfavorable local contact geometry and repeated reorganization than by the raw number of active contact pairs.

Convergence indicators remain bounded throughout the analyzed interval. The maximum constraint norm stays in the range 10^{-13}–10^{-11}, many orders of magnitude below the outer ALM tolerance, while the maximum residual norm remains controlled; its largest excursion ($\approx 2.0 \times 10^{-4}$ at step 4,490) appears near the end of the window during late-stage rearrangement rather than at first impact. The most prominent numerical signature is the line-search pattern: many steps either accept the full Newton step immediately or reach the backtracking cap, suggesting strongly bimodal globalization behavior under dense frictional contact. This is qualitatively consistent with the step-acceptance behavior observed in the joint-constraint benchmarks of Section 6.4.

Taken together, the results support two conclusions. First, the framework successfully advances a geometrically rich, many-body, mixed shell–solid deformable-contact scene over 0.45 s of physical time without catastrophic instability or divergence, validating the robustness of the pipeline at close to one million free degrees of freedom. Second, the test reveals that late rearrangement and re-contact—not peak contact count alone—constitute the dominant

source of nonlinear difficulty in cluttered dropping scenarios. This finding complements the more controlled unit tests and structured scaling benchmarks reported in Sections 6.3 and 6.1, and extends the robustness evidence from isolated geometric validation toward a regime representative of practical many-body engineering simulation.



(a) $t = 0.08\,\mathrm{s}$. (b) $t = 0.08\,\mathrm{s}$.

(c) $t = 0.10\,\mathrm{s}$. (d) $t = 0.10\,\mathrm{s}$.

(e) $t = 0.25\,\mathrm{s}$. (f) $t = 0.25\,\mathrm{s}$.

Figure 6.27: Representative von Mises stress fields for the EVA80 insert case at three stages of the vase–foam interaction. The left column shows a three-quarter (45°/45°/45°) view, and the right column shows an underside view looking upward along the positive z-axis. The rows correspond to $t = 0.08\,\mathrm{s}$ (post-drop, pre-settling), $t = 0.10\,\mathrm{s}$ (settled in the cavity, just before shaking), and $t = 0.25\,\mathrm{s}$ (during lateral shaking).

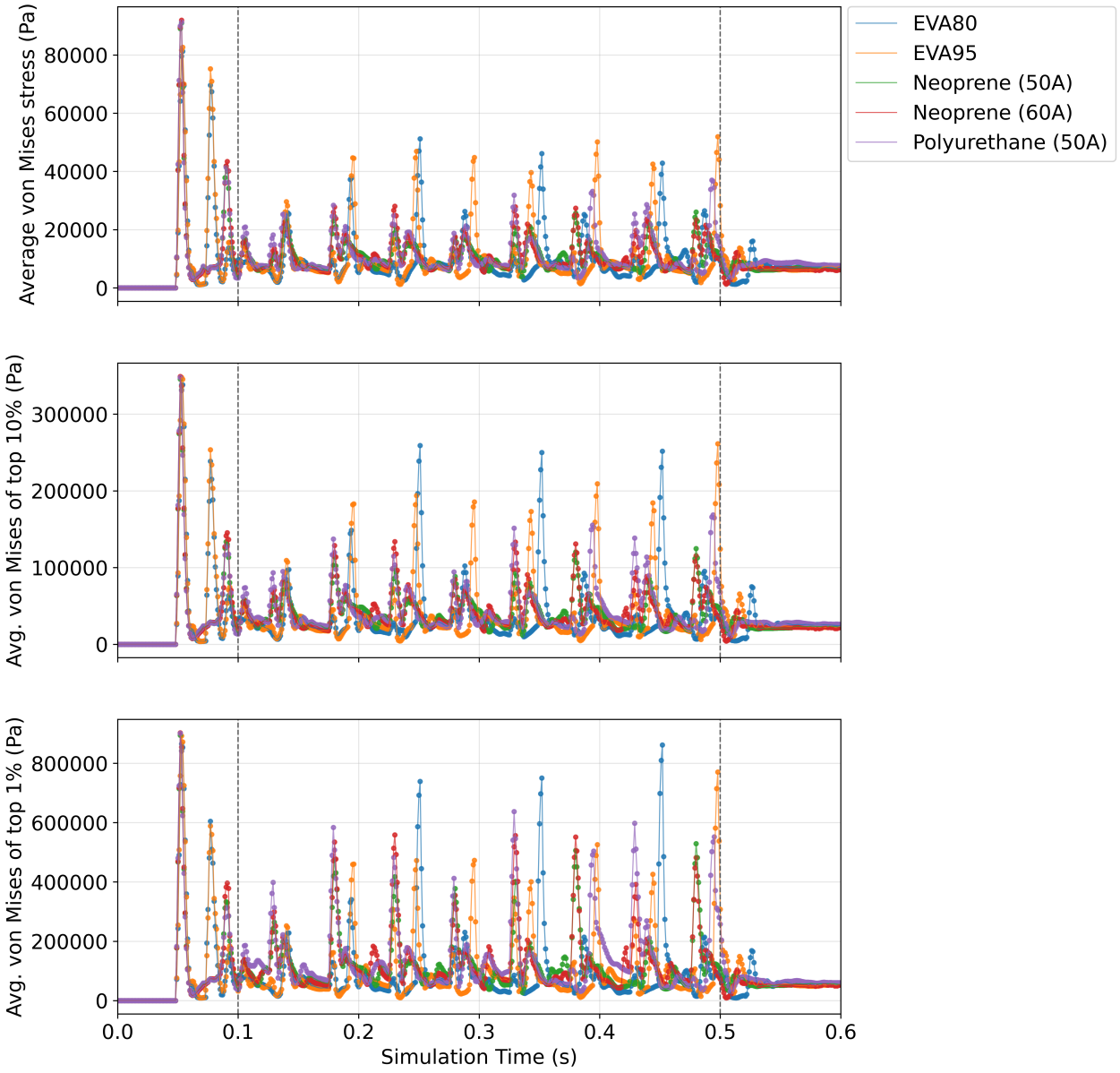
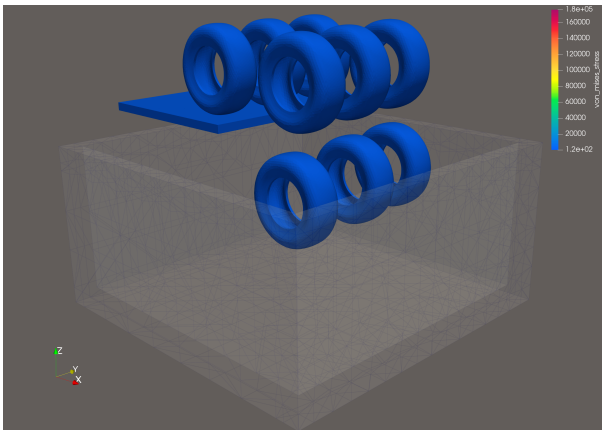
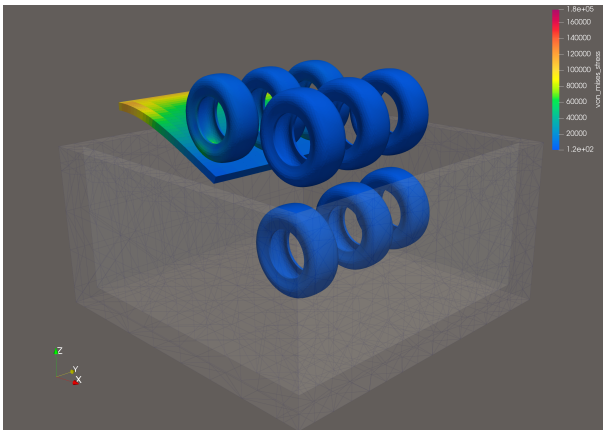


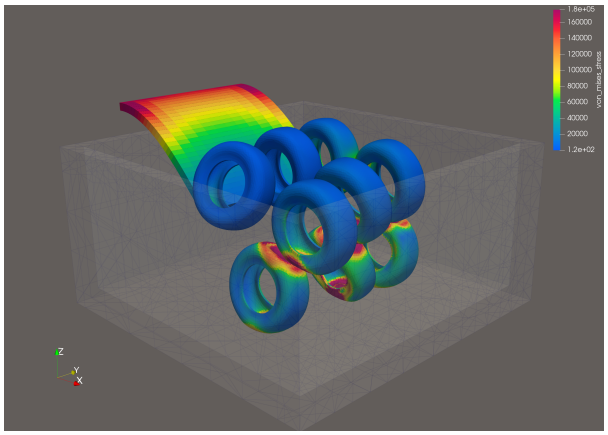
Figure 6.28: Comparison of vase von Mises stress histories for the five foam material presets during the full simulation. The three panels report the whole-body average von Mises stress, the average von Mises stress over the top 10% of cells, and the average von Mises stress over the top 1% of cells, respectively. Phase 1 ($t \in [0.0, 0.1]$ s) corresponds to the vase dropping onto the foam under gravity, Phase 2 ($t \in [0.1, 0.5]$ s) corresponds to foam shaking, and Phase 3 ($t \in [0.5, 0.6]$ s) corresponds to settling.



$t = 0\,\mathrm{s}$: nine tires in free fall; shell beam unloaded.



$t = 0.145\,\mathrm{s}$: first contact; localized bending stress at the contact patch.



$t = 0.262\,\mathrm{s}$: dense contact; shell beam heavily deformed with stress concentrations at inter-tire contact points.

Figure 6.29: von Mises stress field (Pa) at three representative instants of the mixed item-dropping simulation. Container walls are shown as a transparent wireframe. The color scale is shared across all panels.

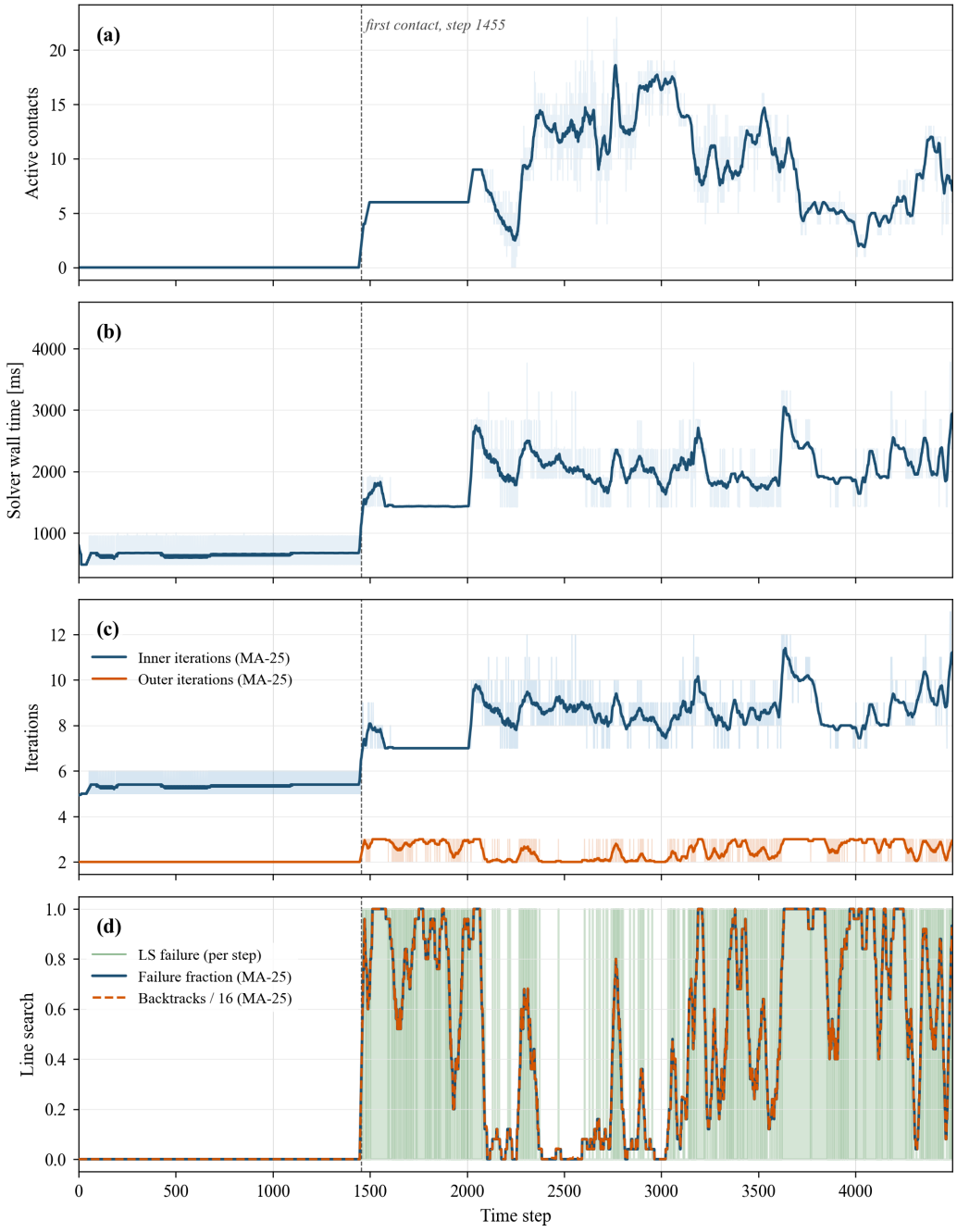


Figure 6.30: Overview diagnostics for the first 4,500 steps (0.45 s) of the mixed item-dropping test. Thin traces are per-step values; heavy lines are 25-step moving averages; the dashed vertical line marks first contact at step 1,455. (a) Active contact count, peaking at 23 (step 2,620). (b) Solver wall time: mean rises from 645 ms to 1,969 ms after contact onset; the costliest step (3,782 ms, step 4,487) has only 10 contacts, confirming that solver cost tracks local geometry rather than contact count. (c) Iteration counts: inner grows from 5.3 to 8.9; outer stays between 2 and 3. (d) Line-search: bimodal pattern—most post-contact steps either accept the full Newton step immediately or exhaust the backtracking budget.

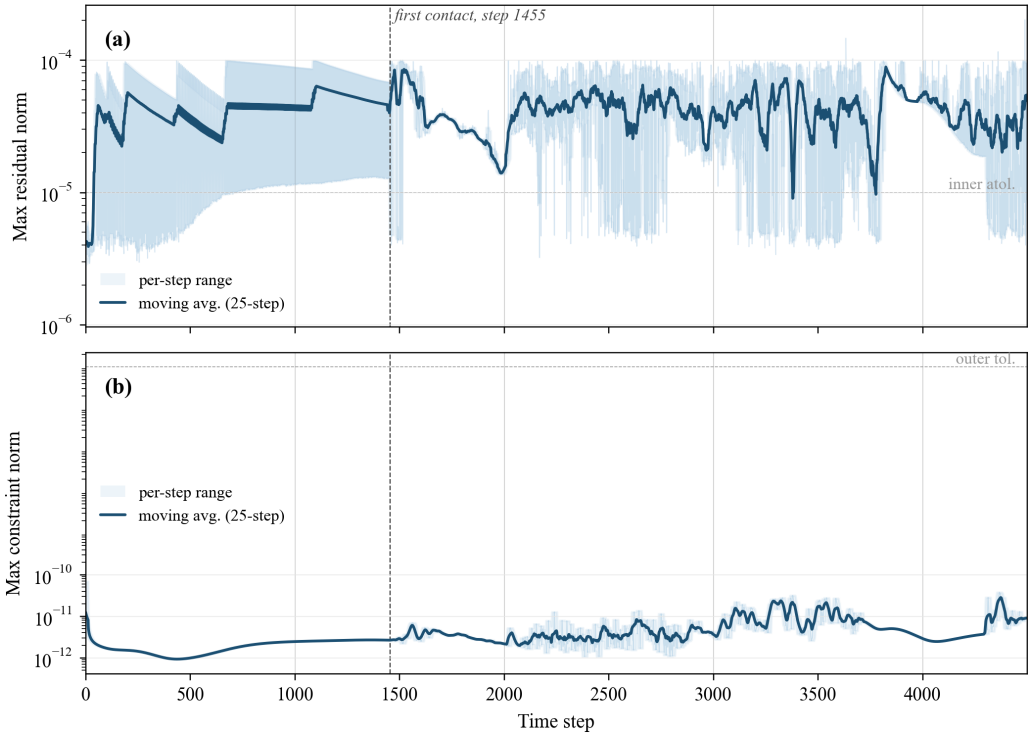


Figure 6.31: Convergence diagnostics for the first 4,500 steps (0.45 s). Shaded bands are per-step ranges; solid lines are 25-step moving averages; dashed lines mark solver tolerances. (a) Maximum residual norm: stays at or below $\varepsilon_{\text{in}} = 10^{-4}$ except for five steps (0.1%), with the largest excursion of 2.0×10^{-4} occurring at step 4,490. (b) Maximum constraint norm: confined to 10^{-13}–10^{-11}, far below the outer ALM tolerance $\varepsilon_{\text{out}} = 10^{-5}$, confirming negligible constraint drift throughout the simulation.

7 TOWARDS NEXT GENERATION OF SIMULATORS - CONTACT-AWARE WORLD MODELS

7.1 Background and Motivation

Despite impressive visual fidelity, state-of-the-art video generative models routinely violate first-principles physics: objects interpenetrate, trajectories ignore gravity, and energy is not conserved. For robotics, scientific visualization, and simulation-in-the-loop content creation, these are not cosmetic artifacts but fundamental limitations that undermine downstream use. A productive lens for addressing them is the *world model* framework [35, 37, 38]—models that maintain an internal state representation and transition dynamics sufficient to predict the consequences of actions over time. Whereas classic world-model agents (Dreamer, DreamerV3 [37, 38]) operate in relatively smooth, low-dimensional environments, video generative models face the full complexity of high-dimensional appearance, non-smooth contact events, and intermittent friction. The central question motivating this chapter is how to endow large-scale video generators with physically lawful transition dynamics, particularly in contact-rich regimes where non-smooth mechanics dominate.

Four complementary strategies have emerged. The first injects physics structure into the model architecture as an inductive bias. PhyDNet [146] introduced a two-branch design that separates a PDE-inspired recurrent cell—modeling smooth spatiotemporal evolution—from a residual appearance branch. The explicit physics subspace constrains the learned dynamics, stabilizing long-horizon rollouts by enforcing locally conservative evolution that resists drift. PhyLoNet [147] extends this design with a relative-flow loss and demonstrates improved robustness over longer prediction horizons. The key insight is that a lightweight, structurally constrained latent update can sharply reduce blatant physical violations at modest computational cost, without requiring simulator calls or model retraining.

The second strategy couples a physics simulator directly to a generative renderer, delegating

state evolution to a mechanistic model. PhysGen [148] exemplifies this staged approach: material and geometric parameters are inferred from a single image; a rigid-body simulator propagates the trajectory under contact and friction; a diffusion renderer then synthesizes temporally consistent video conditioned on the simulated motion. This separation of concerns—physics for dynamics, neural rendering for appearance—yields precise controllability and strong constraint satisfaction. Variants extend the approach to deformable or continuum solvers (e.g., MPM), and emerging sim-projection methods periodically project denoising intermediates back onto the simulator's feasible set, tightening constraints while preserving generative diversity.

The third strategy employs multimodal large language models (MLLMs) or vision-language models (VLMs) to infer physical constraints from a prompt and embed them as guidance objectives during sampling. Zhang et al. [149] extract entities, forces, contacts, and qualitative constraints from text prompts and use these as objectives to guide video diffusion without embedding a full simulator. Because this reasoning step is amortized over the prompt and does not require per-sample simulation, the approach scales to open-domain descriptions and can encode commonsense priors (e.g., gravity direction, liquid behavior) that a generator may not reliably internalize from training data alone. VideoREPA [150] pursues a related goal by distilling physics understanding from video recognition models into text-to-video generators via relational token alignment. The principal risk of reasoning-based methods is hallucination; practical systems therefore pair the planner with lightweight kinematic validators or fallback simulation hints.

The fourth strategy introduces training-free guidance during the denoising process to steer samples away from physical violations without modifying the base model. Hao et al. [151] construct counterfactual, physics-violating prompts and apply synchronized decoupled guidance to repel the sample trajectory from those modes, improving gravity- and contact-related plausibility. Frame-level guidance methods [152] show how per-frame signals—depth maps, keyframes, or constraint scores—can be injected into the denoising loop with low overhead;

replacing these signals with physics-violation scores (e.g., interpenetration estimates, momentum residuals) converts the same mechanism into a practical physics corrector applicable to any pretrained backbone.

Despite this progress, a common limitation across all four strategies is the treatment of contact as a peripheral or implicit phenomenon. Friction coefficients are typically fixed or weakly randomized, contact forces are rarely included as model inputs or outputs, and evaluation protocols measure task reward or perceptual quality rather than contact-mechanics accuracy. This gap is consequential for robotics and manipulation, where the stick–slip transition, normal force distribution, and contact state sequence are primary determinants of whether a planned motion succeeds. The remainder of this chapter addresses this gap through three contributions: a contact-aware encoding scheme that represents per-contact forces as a spatially grounded image aligned with the camera view (§7.3.1); a simulator-generated benchmark dataset—DreamerBench—that exposes synchronized RGB observations, contact visualizations, proprioception, and physics annotations for training and evaluating world models in contact-rich regimes (§7.2); and a VLM-as-Judge evaluation protocol that uses structured rubric scoring aggregated via AUC to evaluate contact-physics fidelity in learned world model predictions (§7.4).

7.2 DreamerBench: Dataset for Evaluating World Models Under Frictional, Contact-Rich Dynamics

World models are typically evaluated on a set of standard benchmarks: continuous-control domains in the DeepMind Control Suite [153], vision-based Atari games via the Arcade Learning Environment (ALE) [154], offline RL datasets such as D4RL [155], robotic manipulation benchmarks like RLBench [156], and large-scale embodied corpora such as Open X-Embodiment [157]. More recent work introduces world-model-oriented benchmarks, for example DrivingDojo for interactive driving world models [158] and WorldPrediction for

high-level world modeling and long-horizon procedural planning [159]. These environments span a range of partially observable Markov decision processes (POMDPs) and are widely used in the development of latent dynamics models. However, they generally treat contact as a hidden or weakly supervised phenomenon: friction coefficients are fixed or weakly randomized, contact forces are rarely logged, and the dynamics are evaluated through reward-based task performance rather than direct examination of the underlying non-smooth, hybrid dynamics.

DreamerBench is designed to occupy a complementary role. It is a simulator-generated, multi-modal dataset that targets frictional, contact-rich dynamics, with the goal of evaluating world models in regimes where the dynamics are discontinuous and governed by non-smooth mechanics [160]. DreamerBench exposes time-synchronized RGB observations, contact visualizations, proprioception, low-level control actions, and physics annotations (contact forces, friction coefficients, and slip signals), together with pre-computed visual latents suitable for token-based world models. The dataset is publicly available at `https://huggingface.co/datasets/zzhou292/DreamerBench`; this section describes its design, simulation pipeline, and evaluation protocol, and places it within the broader set of world-model benchmarks.

7.2.1 Problem Setting and Design Objectives

From a modeling perspective, the domains represented in DreamerBench can be viewed as non-smooth hybrid dynamical systems with intermittent contact, Coulomb friction, and penalty or constraint forces. At the level of an MDP, the transition kernel $p(s_{t+1} \mid s_t, a_t)$ is piecewise smooth but exhibits mode switches when contacts are created or destroyed and when sticking transitions to sliding. Standard benchmarks typically do not make this structure explicit.

The design of DreamerBench is guided by the following objectives:

1. **Contact-centric supervision.** Contact modes (e.g., no contact / sticking / sliding), normal and tangential forces, and friction parameters are exposed as labels. This enables

supervised learning of contact predictors and system-identification-style estimation of friction laws.

2. **Multi-modal but computationally tractable.** DreamerBench provides synchronized RGB, contact "splat" images, and proprioception, and also includes pre-computed vector-quantized latents from the Cosmos-0.1-Tokenizer-DI8x8 encoder [161]. This allows world models either in pixel space or in a discrete latent space without re-running the encoder.

3. **Controlled structural variation.** DreamerBench varies friction coefficients, object inertial parameters, initial conditions, and control signals within a small family of scenarios. This makes it possible to study generalization under distribution shift in physical parameters and identification of latent material properties.

The current version focuses on tabletop interaction motifs involving a hand-held tool (e.g., a flashlight-like object) and one or more rigid objects such as a box, a tall cylindrical container, or a bottle. Typical scenarios include: `flashlight-box` (sliding and pushing a box), `flashlight-coca` (interaction with a tall cylinder with rolling and curved contact patches), and `waterbottle-coca` (combined sliding–rolling between two cylindrical bodies). For each scenario, multiple random seeds perturb surface friction coefficients, initial object poses and velocities, and the control trajectory, producing families of related but distinct contact sequences.

7.2.2 Simulation Pipeline and Logged Modalities

Trajectories are generated using Project Chrono, configured as a multi-physics engine with rigid and (optionally) flexible bodies, Coulomb friction, and penalty-based or constraint-based contact handling. At a fixed simulation time step Δt, the underlying continuous-time dynamics can be interpreted as a hybrid ODE/DAE system with a non-smooth right-hand

side; DreamerBench records a discretized version of these dynamics suited for sequence modeling.

At each simulation step t, a state tuple

$$s_t = \left(o_t^{\text{rgb}}, o_t^{\text{contact}}, q_t, \dot{q}_t, f_t^{\text{contact}}, \mu_t, c_t\right) \tag{7.1}$$

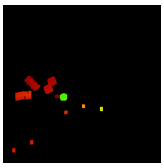
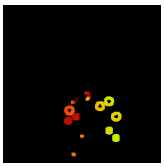
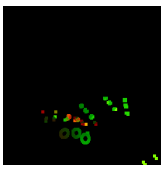
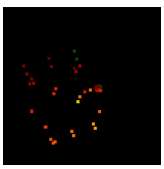
is logged, together with a low-level action a_t and, optionally, a task reward r_t, where:

- o_t^{rgb} collects RGB images from one or more cameras (e.g., ego and side views), at a resolution suitable for vector-quantized autoencoders.
- o_t^{contact} is a "contact splat" image: a workspace-aligned grid that encodes the spatial footprint and magnitude of contact (e.g., approximated contact pressure).
- $q_t, \dot{q}_t$ are robot joint positions and velocities, forming a proprioceptive observation channel.
- f_t^{contact} aggregates per-contact 3D normal and tangential forces (either per contact point or grouped per body), corresponding to impulses in non-smooth mechanics discretized over Δt.
- μ_t denotes scalar or per-body friction coefficients used by the contact model (e.g., static and kinetic coefficients $(\mu_{\text{s}}, \mu_{\text{k}})$), enabling variation of friction regimes.
- c_t is a discrete contact-mode flag (no contact, sticking, sliding, separating), which can be interpreted as a latent discrete state in a hybrid latent variable model.

Each episode (trajectory) is represented as

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T), \tag{7.2}$$

Table 7.1: Simulation scenarios in DreamerBench with example RGB and contact-splat images.

Scenario	Description	Ego camera	Side camera 0	Contact-splat image
`flashlight box`	Tool interacting with a box on a planar surface with sliding and pushing.			
`flashlight coca`	Tool interacting with a tall cylindrical container with sliding and rolling contact.			
`waterbottle coca`	Bottle–cylinder interaction with combined sliding and rolling along the table.			
`fea flashlight`	Four tetrahedron-defined FEA beams and two standing flashlights represented as rigid bodies.			

with horizon T chosen so that objects undergo multiple contact mode transitions (impact, stick–slip, rolling, and settling). This setting is suitable for evaluating long-horizon rollout stability of world models under hybrid dynamics.

File organization and pre-computed latents. On disk, DreamerBench is organized into scenario-specific archives (e.g., `flashlight_box_00.zip`), each containing multiple episodes and auxiliary files:

- arrays for RGB and contact-splat images, proprioception, actions, and physics annotations;
- pre-computed discrete visual tokens $\mathbf{z}_t \in \{0, \ldots, K-1\}^S$ for RGB and contact streams, obtained from a vector-quantized autoencoder and stored in binary files (e.g., `video.bin`, `contact_splat.bin`);
- metadata (JSON) describing token grid size S, codebook size K, frame count, frame rate, and segmentation into episodes.

This layout is compatible with common training stacks (PyTorch, JAX) that either stream episodes as variable-length sequences or treat the dataset as a concatenated sequence with associated *segment IDs*.

7.2.3 Stochastic Joystick Excitation via an Ornstein–Uhlenbeck Process

To generate temporally correlated yet nontrivial end-effector motion for data collection, the joystick command channel in *DreamerBench* is driven by an Ornstein–Uhlenbeck (OU) process. The OU process is a stationary Gaussian Markov process originally introduced as a mean-reverting model for Brownian velocity fluctuations [162]. In continuous time, a d-dimensional OU process $X_t \in \mathbb{R}^d$ satisfies the Langevin-type stochastic differential equation

$$\mathrm{d}X_t = \theta(\mu - X_t)\,\mathrm{d}t + \sigma\,\mathrm{d}W_t, \tag{7.3}$$

where $\theta > 0$ is the mean-reversion rate, $\mu \in \mathbb{R}^d$ is the long-term mean, $\sigma > 0$ is the diffusion scale, and W_t is d-dimensional standard Brownian motion. The process has an exponential

autocorrelation function with correlation time $\tau_c = 1/\theta$ and a Gaussian stationary distribution; efficient numerical schemes for OU trajectories and related functionals are standard in the SDE literature [163].

OU noise is widely used in continuous-control reinforcement learning as a model of colored action noise, in particular to drive exploration in physical systems with inertia without introducing unrealistically high-frequency jitter [164, 165]. Here, the same construction is used not for exploration during learning, but to synthesize joystick-like excitation that repeatedly engages contact, sliding, and rolling during data collection. We instantiate a three-dimensional OU process

$$X_t = \left(X_t^{(x)}, X_t^{(y)}, X_t^{(z)}\right)^\top \tag{7.4}$$

representing a latent control signal over Cartesian directions (x, y, z) in the workspace of the robot arm.

The implementation mirrors an Euler–Maruyama discretization of (7.3) with time step Δt,

$$X_{k+1} = X_k + \theta(\mu - X_k)\,\Delta t + \sigma\sqrt{\Delta t}\,\xi_k, \qquad \xi_k \sim \mathcal{N}(0, I_3), \tag{7.5}$$

which is encoded in the `OrnsteinUhlenbeckProcess.sample()` method. The parameters used in the script correspond to relatively large θ and σ, yielding high-variance but strongly time-correlated excitation, broadly consistent with common practice in continuous-control benchmarks [164, 165].

After each update (7.5), the process state is post-processed before being interpreted as a joystick command. First, a minimum-magnitude constraint is applied: letting $m_k = \|X_k\|_2$ and denoting a user-specified threshold $m_{\min} > 0$, the state is rescaled according to

$$X_k \leftarrow \begin{cases} \dfrac{m_{\min}}{m_k} X_k, & 0 < m_k < m_{\min}, \\[2ex] v\, m_{\min}, & m_k = 0, \end{cases} \tag{7.6}$$

where v is a random unit vector drawn from an isotropic Gaussian and normalized. This step enforces a lower bound on the norm of X_k while preserving the current direction whenever possible, avoiding extended intervals of near-zero actuation.

Next, each component is passed through a deadzone and mapped to a Cartesian increment. If $X_k^{(i)}$ denotes the i-th component, a threshold $\varepsilon > 0$ is applied via

$$X_k^{(i)} \leftarrow \begin{cases} 0, & |X_k^{(i)}| < \varepsilon, \\ X_k^{(i)}, & \text{otherwise}, \end{cases} \tag{7.7}$$

and the resulting vector is converted into an update for the desired end-effector position $p_k \in \mathbb{R}^3$:

$$p_{k+1} = \Pi_{\mathcal{P}}\Big(p_k + v_{\text{scale}}\, X_k\, \Delta t\Big), \tag{7.8}$$

where v_{scale} is a scalar step size corresponding to `movement_speed` in the code, and $\Pi_{\mathcal{P}}$ denotes component-wise clipping to a predefined axis-aligned workspace box $\mathcal{P}$. The updated target position p_{k+1} is then passed to an inverse kinematics solver, which produces joint angles applied to the robot at the control rate.

Overall, the joystick control law in *DreamerBench* can be viewed as an OU-driven colored-noise excitation filtered through a norm constraint, a deadzone nonlinearity, and a constrained kinematic integrator. This construction yields temporally coherent, reproducible stochastic motion that systematically excites contact-rich interactions while avoiding degenerate trajectories with negligible motion, and it admits straightforward ablations (e.g., modifying τ_c or replacing OU noise with i.i.d. Gaussian noise) for future studies.

7.3 Proposed Methods for Contact-Aware World Models

7.3.1 Contact Encoding via Depth-Weighted Gaussian Splats

A central design choice in DreamerBench is to represent contact as a camera-centric image rather than as a sparse list of contact points. Existing work explores several alternatives. Neural Contact Fields represent contact as a continuous probability field over 3D space conditioned on tactile inputs, enabling estimation of complex extrinsic contact patches from visuotactile signals [166]. Vision-only approaches such as Im2Contact recover dense contact maps from RGB observations by learning a pixel-wise contact likelihood without explicit access to forces or tactile measurements [167]. Large-scale visuotactile datasets including FreeTacMan encode contact as image-like tactile fields synchronized with external views for contact-rich manipulation [168]. More recently, DyTact models dynamic contacts using 2D Gaussian surfels attached to articulated meshes, rasterized into time-varying contact fields [169]. These lines of work suggest that contact fields expressed in an image-like domain are a convenient interface for modern vision backbones and world models.

In DreamerBench, each simulation step produces a *contact splat* image that aggregates all active contacts into a single RGB frame aligned with a reference camera. For each contact i, we log a 3D contact position $\mathbf{p}_i \in \mathbb{R}^3$ and a contact force $\mathbf{f}_i \in \mathbb{R}^3$. Given a camera with world-frame position $\mathbf{c} \in \mathbb{R}^3$ and rotation $\mathbf{R}_{cw} \in \mathrm{SO}(3)$ (rotation from world to camera), the camera-frame coordinates of the contact point are

$$\mathbf{x}_i = \mathbf{R}_{cw}(\mathbf{p}_i - \mathbf{c}) = \Big(X_i,\ Y_i,\ Z_i\Big)^\top. \tag{7.9}$$

We interpret X_i as depth along the camera forward axis (with $X_i > 0$ in front of the camera) and apply near-plane clipping by discarding contacts with $X_i < X_{\min}$ or clipping line segments to the plane $X = X_{\min}$ to avoid numerical instabilities in the projection.

Projection to the discrete image plane uses a pinhole model with intrinsics (f_x, f_y, c_x, c_y):

$$u_i = c_x - f_x \frac{Y_i}{X_i}, \qquad v_i = c_y - f_y \frac{Z_i}{X_i}, \tag{7.10}$$

followed by rounding to the nearest pixel center $(\tilde{u}_i, \tilde{v}_i) \in \mathbb{Z}^2$. To encode force *direction* in the image plane, the implementation also projects a second point displaced along the force vector,

$$\mathbf{x}_i^{\text{end}} = \mathbf{R}_{cw}\Big(\mathbf{p}_i + s_i \mathbf{f}_i - \mathbf{c}\Big), \tag{7.11}$$

with a scale s_i chosen as an affine or quantile-based function of the force magnitude $\|\mathbf{f}_i\|$. The displacement between the two projections in pixel space,

$$\Delta\mathbf{d}_i = \Big(\Delta u_i,\ \Delta v_i\Big)^\top = \Big(u_i^{\text{end}} - u_i,\ v_i^{\text{end}} - v_i\Big)^\top, \tag{7.12}$$

defines a 2D direction vector. After normalization $\hat{\mathbf{d}}_i = \Delta\mathbf{d}_i / \|\Delta\mathbf{d}_i\|$, the two components are mapped from $[-1, 1]$ to $[0, 1]$ and stored in the green and blue channels, respectively:

$$G_i = \frac{\hat{d}_{i,x} + 1}{2}, \qquad B_i = \frac{\hat{d}_{i,y} + 1}{2}. \tag{7.13}$$

The red channel encodes a clipped and normalized force magnitude,

$$m_i = \|\mathbf{f}_i\|, \qquad R_i = \frac{\min(m_i, m_{\max})}{m_{\max}} \in [0, 1], \tag{7.14}$$

so that R_i saturates for very large contact forces.

Each contact is then rendered as an isotropic Gaussian kernel centered at $(\tilde{u}_i, \tilde{v}_i)$, with a radius that depends on the normalized magnitude $m_i / m_{\max}$. Specifically, we define a radius r_i through

$$t_i = \left(\frac{m_i}{m_{\max}}\right)^\gamma, \qquad r_i = r_{\min} + (r_{\max} - r_{\min}) t_i, \tag{7.15}$$

with a shaping exponent $\gamma \geq 1$ and bounds $0 < r_{\min} \leq r_{\max}$. The corresponding Gaussian

kernel over integer pixel offsets $(\delta u, \delta v)$ is

$$k_i(\delta u, \delta v) = \exp\left(-\frac{\delta u^2 + \delta v^2}{2\sigma_i^2}\right), \quad \sigma_i \approx \frac{r_i}{3}, \tag{7.16}$$

cropped to a finite window of radius $\mathcal{O}(r_i)$. To encourage nearer contacts to dominate farther ones, the kernel is multiplied by a depth-dependent weight

$$w_i^{\text{depth}} = \exp\left(-\frac{X_i}{\tau_{\text{depth}}}\right), \tag{7.17}$$

with a tunable depth scale $\tau_{\text{depth}} > 0$. The total weight for contact i at pixel (u, v) is then

$$w_i(u, v) = w_i^{\text{depth}}\, k_i\Big(u - \tilde{u}_i,\, v - \tilde{v}_i\Big). \tag{7.18}$$

Let $\mathbf{c}_i = (R_i, G_i, B_i)^\top$ denote the per-contact color vector. The image is assembled by weighted accumulation over all contacts, followed by a per-pixel normalization:

$$\tilde{\mathbf{I}}(u, v) = \sum_i w_i(u, v)\, \mathbf{c}_i, \tag{7.19}$$

$$W(u, v) = \sum_i w_i(u, v), \tag{7.20}$$

$$\mathbf{I}(u, v) = \begin{cases} \tilde{\mathbf{I}}(u, v)/W(u, v), & \text{if } W(u, v) > 0, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{7.21}$$

Here $\mathbf{I}(u, v) \in [0, 1]^3$ is the final contact splat image at the given time step. The normalized averaging behaves like a soft z-buffer over Gaussian "surfels": overlapping contributions are blended rather than summed unboundedly, and nearer contacts are given higher effective weight through w_i^{depth}. The resulting image compactly encodes contact spatial footprint, approximate pressure (via red intensity and kernel size), and projected force direction (via green–blue hue) in a form directly consumable by convolutional networks or video-tokenization modules.

7.3.2 Network Architecture and Loss Design

This section describes ChronoDreamer, a spatial-temporal transformer architecture for action-conditioned video prediction that operates on discrete latent representations obtained through finite-scalar quantization. The model employs a MaskGIT-style training objective for next-frame prediction and jointly predicts future video frames, contact maps, and joint angles conditioned on historical observations and action sequences.

7.3.2.1 Architecture Overview

Figure 7.1 depicts the data pipeline. Raw RGB frames and contact splat images at 256×256 resolution are tokenized by the Cosmos DI8×8 encoder [161] via finite-scalar quantization, yielding 32×32 discrete token grids per frame. These tokens, along with continuous action commands and joint angle observations, are loaded via memory-mapped I/O and passed to the spatial-temporal dynamics model, which outputs predicted video tokens, contact tokens, and regressed joint angles.

The architecture comprises four components. The *video encoder* (NVIDIA Cosmos Tokenizer [161]) maps $H \times W$ RGB frames to $h \times w$ discrete token grids with vocabulary size $V = 65{,}536$ via finite-scalar quantization. The *spatial-temporal transformer* consists of L blocks, each containing factorized spatial and temporal self-attention followed by a feed-forward network. The *MaskGIT module* performs masked token prediction with iterative unmasking governed by a cosine schedule. The *video decoder* reconstructs RGB frames from quantized latents; separate linear heads project to contact map logits and joint angle outputs.

Throughout this section, we denote the batch size by B, temporal sequence length by T, input frame dimensions by $H \times W$, latent token grid dimensions by $h \times w$ (with $S = h \times w$ spatial tokens per frame), model hidden dimension by D, number of attention heads by N_h, vocabulary size by V, and number of transformer layers by L.

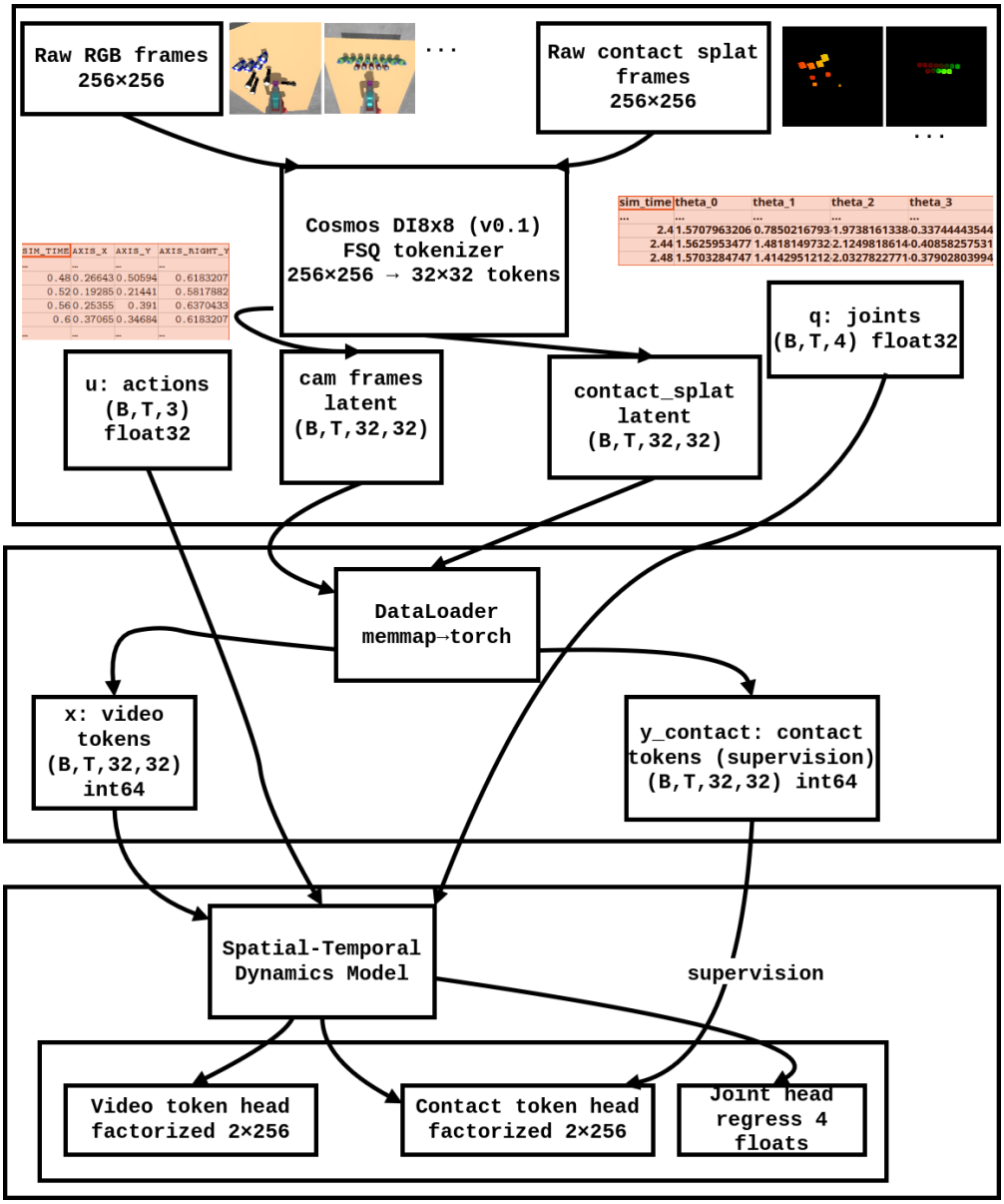


Figure 7.1: Data pipeline. Cosmos encoder tokenizes RGB and contact frames into 32×32 grids. The dynamics model outputs factorized video/contact logits and joint angles.

7.3.2.2 Video Encoder: NVIDIA Cosmos Tokenizer with Finite-Scalar Quantization

Each input RGB frame is encoded into a compact sequence of discrete tokens using the NVIDIA Cosmos Tokenizer [161], a hierarchical convolutional network composed of residual

blocks and 2-level Haar wavelet downsampling that maps a 256×256 frame to a 16×16 spatial grid of discrete latent tokens (a spatial compression factor of 16). The continuous latent representation produced by the convolutional backbone is discretized via Finite-Scalar Quantization (FSQ) [170], a non-learned, deterministic quantization scheme that independently maps each of $C = 6$ continuous latent channels to a fixed set of discrete levels; the resulting per-channel indices are combined into a single token index drawn from a vocabulary of $V \approx 65{,}536$ values, with no explicit learnable codebook. This discrete spatial representation serves as the fundamental unit of the sequence model: each video frame is reduced to $S = 256$ tokens, enabling the downstream transformer to operate over compact integer sequences rather than high-dimensional pixel arrays while retaining sufficient visual fidelity for contact-rich scene prediction.

7.3.2.3 Input Representation and Conditioning

Input Modalities The model receives four input modalities: *history video tokens* $\mathbf{Z}_{\text{hist}} \in \{0, \ldots, V\}^{T_h \times S}$, discrete tokens for T_h history frames with $S = h \times w$ spatial tokens per frame; *history actions* $\mathbf{a}_{\text{hist}} \in \mathbb{R}^{T_h \times 3}$, continuous vectors encoding commanded velocities; *future actions* $\mathbf{a}_{\text{fut}} \in \mathbb{R}^{T_f \times 3}$, action vectors for frames to be predicted; and *history joint angles* $\boldsymbol{\theta}_{\text{hist}} \in \mathbb{R}^{T_h \times N_j}$ with $N_j = 4$ joint channels.

Embedding Strategy *Video Token Embedding.* Video tokens are embedded using the factorized embedding scheme:

$$\mathbf{X}_{\text{video}} = \text{FactorizedEmbed}(\mathbf{Z}) \in \mathbb{R}^{B \times T \times S \times D} \tag{7.22}$$

A special mask token embedding $\mathbf{e}_{\text{mask}} \in \mathbb{R}^D$ is used for tokens that are masked during training:

$$\mathbf{X}_{\text{video}}[i, t, s] = \begin{cases} \text{FactorizedEmbed}(z_{t,s}) & \text{if } z_{t,s} \neq z_{\text{mask}} \\ \mathbf{e}_{\text{mask}} & \text{otherwise} \end{cases} \tag{7.23}$$

Action Embedding. Actions are projected to the model dimension through a linear layer followed by layer normalization:

$$\mathbf{A} = \text{LayerNorm}(\mathbf{W}_a \mathbf{a} + \mathbf{b}_a) \in \mathbb{R}^{B \times T \times D} \tag{7.24}$$

where $\mathbf{W}_a \in \mathbb{R}^{D \times 3}$ and $\mathbf{b}_a \in \mathbb{R}^D$.

Joint Angle Embedding. Joint angles are similarly projected:

$$\mathbf{J} = \text{LayerNorm}(\mathbf{W}_j \boldsymbol{\theta} + \mathbf{b}_j) \in \mathbb{R}^{B \times T \times D} \tag{7.25}$$

where $\mathbf{W}_j \in \mathbb{R}^{D \times N_j}$.

Combined Control Token. The action and joint angle embeddings are combined additively to form a single control token per frame:

$$\mathbf{C}_t = \mathbf{A}_t + \mathbf{J}_t \in \mathbb{R}^D \tag{7.26}$$

This combined control token is prepended to the video tokens for each frame, yielding an augmented sequence:

$$\mathbf{X}_t = [\mathbf{C}_t; \mathbf{X}_{\text{video},t}] \in \mathbb{R}^{(S+1) \times D} \tag{7.27}$$

The full input tensor has shape $(B, T, S+1, D)$, where the first position in the spatial dimension corresponds to the control token.

Positional Encoding Learnable positional embeddings encode both temporal and spatial positions:

$$\mathbf{P} \in \mathbb{R}^{1 \times T \times (S+1) \times D} \tag{7.28}$$

The positional embedding is added to the input representation:

$$\mathbf{X}' = \mathbf{X} + \mathbf{P} \tag{7.29}$$

7.3.2.4 Spatial-Temporal Transformer Decoder

Architecture Overview Figure 7.2 details the transformer internals. The ST-Transformer decoder comprises $L = 24$ identical blocks, each applying factorized spatial and temporal attention followed by a feed-forward network. This factorization reduces complexity from $\mathcal{O}(T^2S^2)$ for joint spatiotemporal attention to $\mathcal{O}(TS^2 + T^2S)$.

ST-Block Structure Each ST-Block contains three sub-layers applied in sequence: spatial self-attention across all $S + 1$ tokens within each frame, temporal self-attention across all T frames for each spatial position, and a position-wise feed-forward network.

Spatial Self-Attention. Given input $\mathbf{X} \in \mathbb{R}^{B\times T\times(S+1)\times D}$, spatial attention operates independently on each frame. The input is first reshaped to $(BT, S + 1, D)$:

$$\mathbf{X}_{\text{spatial}} = \text{reshape}(\mathbf{X}, (BT, S + 1, D)) \tag{7.30}$$

Standard multi-head self-attention is applied:

$$\mathbf{X}'_{\text{spatial}} = \mathbf{X}_{\text{spatial}} + \text{SelfAttn}(\text{Norm}(\mathbf{X}_{\text{spatial}})) \tag{7.31}$$

In spatial attention, the control token (at position 0) attends to and is attended by all video tokens (positions 1 to S), enabling action-video interaction within each frame.

Temporal Self-Attention with Causal Masking. For temporal attention, the tensor is reshaped to $(B(S + 1), T, D)$:

$$\mathbf{X}_{\text{temporal}} = \text{reshape}(\mathbf{X}'_{\text{spatial}}, (B(S + 1), T, D)) \tag{7.32}$$

Causal self-attention is applied with a lower-triangular attention mask:

$$\mathbf{X}'_{\text{temporal}} = \mathbf{X}_{\text{temporal}} + \text{CausalSelfAttn}(\mathbf{X}_{\text{temporal}}) \tag{7.33}$$

The causal mask ensures that tokens at time t can only attend to tokens at times $\leq t$:

$$M_{ij} = \begin{cases} 0 & \text{if } i \geq j \\ -\infty & \text{otherwise} \end{cases} \tag{7.34}$$

Self-Attention Mechanism The self-attention operation computes queries, keys, and values through linear projections:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q \in \mathbb{R}^{N \times D} \tag{7.35}$$

$$\mathbf{K} = \mathbf{X}\mathbf{W}_K \in \mathbb{R}^{N \times D} \tag{7.36}$$

$$\mathbf{V} = \mathbf{X}\mathbf{W}_V \in \mathbb{R}^{N \times D} \tag{7.37}$$

Multi-head attention splits these into $N_h = 8$ heads with dimension $d_k = D/N_h = 32$:

$$\text{head}_i = \text{softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V}_i \tag{7.38}$$

The heads are concatenated and projected:

$$\text{MultiHead}(\mathbf{X}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_{N_h})\mathbf{W}_O \tag{7.39}$$

QK-Normalization. Layer normalization is applied to queries and keys before computing attention scores:

$$\tilde{\mathbf{Q}}_i = \text{LayerNorm}(\mathbf{Q}_i) \tag{7.40}$$

$$\tilde{\mathbf{K}}_i = \text{LayerNorm}(\mathbf{K}_i) \tag{7.41}$$

This stabilizes training, particularly for large models.

μP Scaling. When using maximal update parameterization (μP), attention logits are

scaled by $8/d_k$ instead of $1/\sqrt{d_k}$:

$$\alpha = \frac{8}{d_k} \quad (\mu\text{P}) \tag{7.42}$$

This ensures consistent training dynamics across model widths.

Feed-Forward Network The feed-forward network consists of two linear transformations with GELU activation:

$$\text{FFN}(\mathbf{x}) = \mathbf{W}_2 \cdot \text{Dropout}\left(\text{GELU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)\right) + \mathbf{b}_2 \tag{7.43}$$

The hidden dimension is $D_{\text{ff}} = 4D = 1024$ (controlled by mlp_ratio = 4.0).

The complete ST-Block forward pass is:

$$\mathbf{X}^{(1)} = \mathbf{X} + \text{SpatialAttn}(\text{Norm}(\mathbf{X})) \tag{7.44}$$

$$\mathbf{X}^{(2)} = \mathbf{X}^{(1)} + \text{TemporalAttn}(\mathbf{X}^{(1)}) \tag{7.45}$$

$$\mathbf{X}^{(3)} = \mathbf{X}^{(2)} + \text{FFN}(\text{Norm}(\mathbf{X}^{(2)})) \tag{7.46}$$

Attention Pattern Analysis Spatial attention is bidirectional within each frame: the control token $\mathbf{C}_t$ attends to all video tokens $\{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,S}\}$, and each video token $\mathbf{x}_{t,s}$ attends to both the control token and all other video tokens in frame t. No causal mask is applied.

Temporal attention enforces causality across frames. Each position attends only to the same position at earlier times: the control token at time t aggregates $\{\mathbf{C}_0, \ldots, \mathbf{C}_t\}$; the video token at position s and time t attends to $\{\mathbf{x}_{0,s}, \ldots, \mathbf{x}_{t,s}\}$. The causal mask prevents information flow from future frames.

7.3.2.5 MaskGIT: Masked Generative Image Transformer

ChronoDreamer adopts the masked generative image transformer (MaskGIT) [171] training objective for video token prediction. During training, a subset of future-frame tokens is masked and the model predicts the originals. Each masked frame samples a mask ratio from

a cosine schedule,

$$p_t = \cos\left(\frac{\pi}{2} \cdot u\right), \quad u \sim \text{Uniform}(0, 1), \tag{7.47}$$

and with probability $\rho_{\text{non-mlm}} = 0.5$ an autoregressive-style boundary frame t^* is chosen instead, fully corrupting all frames at or after t^*. A small random token replacement ($r = 0.2$) is applied prior to masking to improve robustness.

Loss Computation Three prediction heads share the transformer backbone. The video and contact heads each output logits over the K-factor FSQ vocabulary and are trained with factorized cross-entropy over masked positions $\mathcal{M}$:

$$\mathcal{L}_{\text{video}} = \frac{1}{|\mathcal{M}|} \sum_{(t,s)\in\mathcal{M}} \sum_{k=1}^{K} \text{CE}\Big(\boldsymbol{\ell}_{t,s}^{(k)},\, z_{t,s}^{(k)}\Big)\,, \tag{7.48}$$

$$\mathcal{L}_{\text{contact}} = \frac{1}{|\mathcal{M}|} \sum_{(t,s)\in\mathcal{M}} \sum_{k=1}^{K} \text{CE}\Big(\boldsymbol{\ell}_{t,s}^{\text{contact},(k)},\, c_{t,s}^{(k)}\Big)\,. \tag{7.49}$$

The control token predicts joint angles via MSE over future frames:

$$\mathcal{L}_{\text{joint}} = \frac{1}{T_f} \sum_{t=T_h}^{T-1} \|\hat{\boldsymbol{\theta}}_t - \boldsymbol{\theta}_t\|_2^2. \tag{7.50}$$

The total loss is

$$\mathcal{L} = \mathcal{L}_{\text{video}} + \lambda_{\text{contact}}\mathcal{L}_{\text{contact}} + \lambda_{\text{joint}}\mathcal{L}_{\text{joint}}, \tag{7.51}$$

with $\lambda_{\text{contact}} = 2.0$ and $\lambda_{\text{joint}} = 1.0$.

Inference Future frames are decoded autoregressively. For each frame t, all tokens are initialized as mask tokens and unmasked over N_{steps} iterations following the MaskGIT cosine schedule [171]. At each iteration, every position is scored by confidence

$$c_s = \prod_{k=1}^{K} \max_{v} \text{softmax}(\boldsymbol{\ell}_s^{(k)})_v, \tag{7.52}$$

the highest-confidence positions are committed, and the remainder are re-masked for the next iteration. Once a frame is fully decoded, its tokens serve as context for subsequent frames.

7.3.2.6 Video Decoder

Decoder Architecture The decoder mirrors the encoder, mapping quantized latents to RGB images. Given quantized features $\hat{\mathbf{e}} \in \{-1, +1\}^{C_z \times h \times w}$, an initial 3×3 convolution projects to $C_{\text{base}} \cdot 4$ channels, followed by mid-block residual layers. Upsampling stages apply residual blocks with channel multipliers $(4, 2, 2, 1, 1)$ in reverse order. The output projection applies group normalization and Swish activation before a 3×3 convolution produces three RGB channels.

Upsampling Operation Spatial resolution is increased using depth-to-space (pixel shuffle) operations:

$$\text{Upsample}(\mathbf{h}) = \text{DepthToSpace}(\text{Conv}_{3\times3}(\mathbf{h}; 4C), \text{block\_size} = 2) \tag{7.53}$$

This rearranges a tensor of shape $(B, 4C, H, W)$ to $(B, C, 2H, 2W)$.

Token-to-Latent Mapping Given predicted token indices $z \in \{0, \ldots, V-1\}$, the decoder recovers the continuous latent representation via the inverse FSQ de-quantization operation. This produces a tensor $\mathbf{e} \in \mathbb{R}^{C \times h \times w}$ with $C = 6$ channels, which is then passed to the decoder network for image reconstruction. The inverse Haar wavelet transform is applied during decoding to upsample the spatial resolution by a factor of 4.

Factorization Strategy For computational efficiency during training and inference, token indices are factorized into $K = 2$ factors, each with vocabulary size $V_f = 256$, such that $V_f^K = 65{,}536 \approx V$. This reduces the output logit dimension from V to $K \cdot V_f$ and enables parallel prediction across factors.

7.3.2.7 Model Output Specification

Forward Pass Outputs During training, the forward pass returns the total loss $\mathcal{L}$ and its components $\mathcal{L}_{\text{video}}$, $\mathcal{L}_{\text{contact}}$, $\mathcal{L}_{\text{joint}}$, along with token-level accuracies on masked positions. The full logit tensor of shape $(B, K \cdot V_f, T, H, W)$ is retained for downstream analysis.

Generation Outputs At inference, `generate()` returns predicted video token indices of shape $(B, T \cdot H \cdot W)$. Optional outputs include contact map tokens (B, T_f, H, W), predicted joint angles (B, T_f, N_j), and factored logits for uncertainty quantification.

7.3.2.8 Model Configuration

Default Hyperparameters The model uses $L = 24$ transformer layers with hidden dimension $D = 256$, $N_h = 8$ attention heads (head dimension $d_k = 32$), FFN hidden dimension $D_{\text{ff}} = 1024$, temporal sequence length $T = 16$, and spatial sequence length $S = 1024$ (32×32 tokens per frame). The image vocabulary size is $V = 65536$, factorized into $K = 2$ sub-vocabularies of size $V_f = 256$ each. Joint angle inputs have $N_j = 4$ channels.

Computational Complexity Per ST-Block, spatial attention requires $\mathcal{O}(BT(S+1)^2 D)$ operations while temporal attention requires $\mathcal{O}(B(S+1)T^2 D)$ operations. For L layers, the total complexity is:

$$\mathcal{O}(L \cdot BTS(S+T)D) \tag{7.54}$$

With default parameters ($T = 16$, $S = 1024$, $D = 256$, $L = 24$), this is approximately $\mathcal{O}(10^{11})$ operations per forward pass for batch size 1.

Parameter Count. The model contains approximately:

$$\text{Params} \approx L \cdot (12D^2 + 8D^2) + V_f \cdot K \cdot D + T \cdot (S+1) \cdot D \tag{7.55}$$

yielding roughly 30-50M parameters depending on configuration.

7.3.3 Results and Discussion

7.3.3.1 Quantitative Analysis

Training performance of ChronoDreamer is monitored via two complementary signals. The primary signal is the masked video token cross-entropy loss $\mathcal{L}_{\text{video}}$, which measures how accurately the model recovers masked discrete tokens in future frames; lower values indicate that the predicted token distribution concentrates on the correct codebook entry. As a perceptual quality check, LPIPS [172] is computed between frames decoded from predicted tokens and the corresponding ground-truth frames, providing a measure of appearance fidelity that is less sensitive to minor spatial misalignment than pixel-level MSE.

Table 7.2 summarizes final evaluation metrics across the four DreamerBench scenarios. Including the contact modality yields a richer multi-modal output: in addition to RGB video predictions, the model simultaneously predicts contact force maps encoded as depth-weighted Gaussian splats, providing an explicit supervisory signal for friction and contact geometry that the video-only model entirely lacks. This additional predictive burden is reflected in modestly higher video cross-entropy loss and LPIPS across all scenarios—a natural trade-off, as the model must allocate representational capacity across two output streams rather than one. The contact modality is nevertheless the key contribution: it enables downstream applications such as contact-aware planning and force estimation that are simply out of reach for a video-only world model.

Table 7.2: Quantitative evaluation of ChronoDreamer across DreamerBench scenarios. Lower is better for all metrics. "With contact" includes the contact-encoding modality; "Video only" ablates it.

Dataset	Mode	Video Loss	Video LPIPS	Joint MSE
`fea_flashlight`	With contact	8.6429	0.1644	0.005884
	Video only	8.6183	0.1532	0.004316
`flashlight_coca_eval`	With contact	8.4509	0.1793	0.003468
	Video only	8.3706	0.1566	0.002590
`flashlight_box_eval`	With contact	7.9173	0.1477	0.004819
	Video only	7.8388	0.1318	0.003946
`waterbottle_coca_eval`	With contact	8.6861	0.1816	0.004710
	Video only	8.5922	0.1598	0.003721

7.3.3.2 Qualitative Analysis

A qualitative analysis has been conducted by a human to gain an understanding of the model performance after training for 4 epochs using the following parameters.

We have noticed the spatial and object coherence, the objects' relative positions on the table, can be preserved easily by the model, across multiple scenarios, when contact has not happened. Examples have been provided in Figure 7.3. The spatial coherence is a combination of camera rendering and positioning, lower level inverse kinematic controller, and multibody dynamics physics.

Figures 7.4a and 7.4b show cases in which contact happens between the end-effector and an object on the table. The model is able to predict the contact event and generate corresponding contact splat frames. However, the predicted video and camera frames tend to show artifacts and blurriness after contact happens, and certain spatial coherence is lost. This behavior is quite consistent across multiple scenarios, tasks, and given prompts. This might be an indication that the current DreamerBench dataset does not contain enough contact-rich data, as indeed due to the nature of action sampling and simulation setup, contact events take a relatively small portion of the entire dataset.

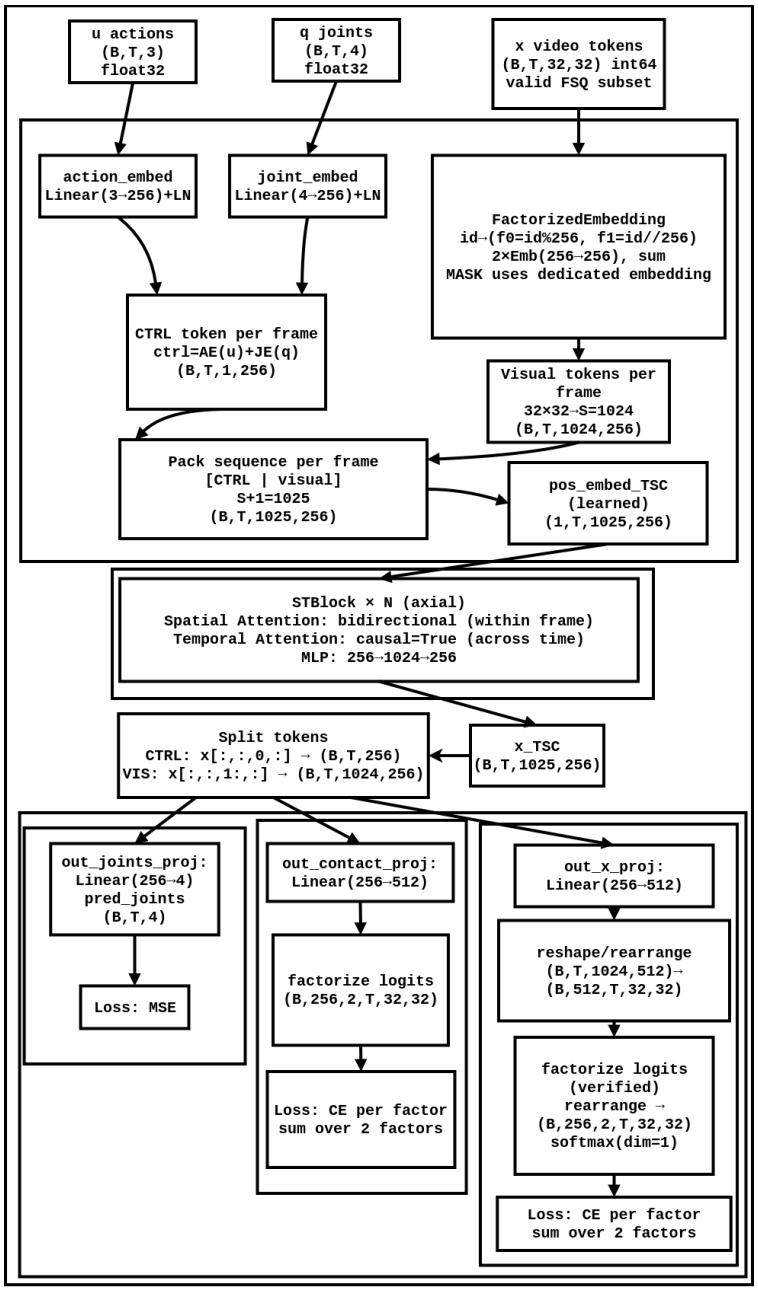
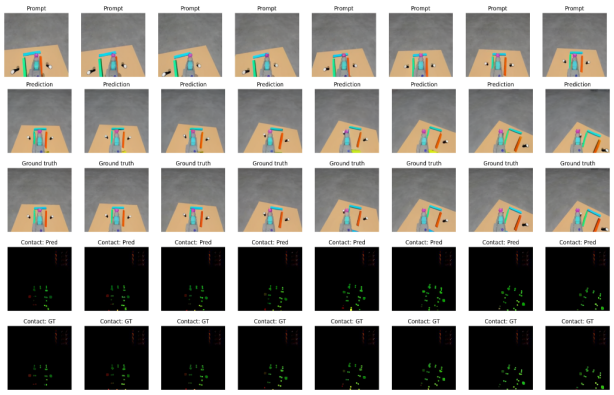
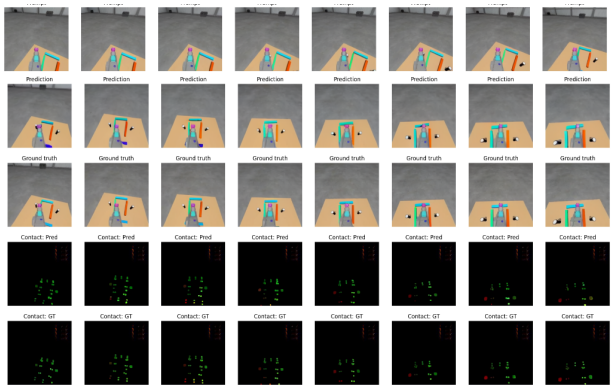


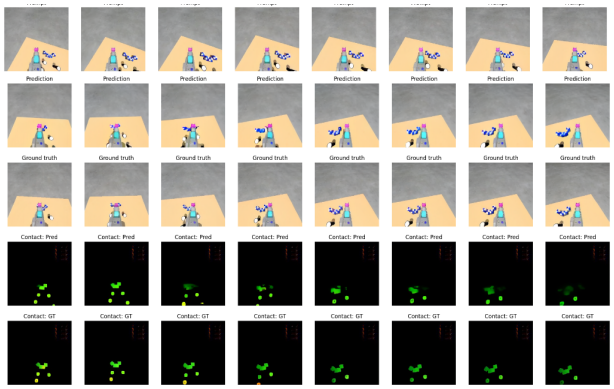
Figure 7.2: ST-Transformer architecture. Control tokens (action + joint embeddings) are concatenated with video tokens and processed by N axial ST-Blocks. Three output heads: joint regression (MSE), contact tokens, and video tokens (factorized CE).



(a) End-effector moving left.

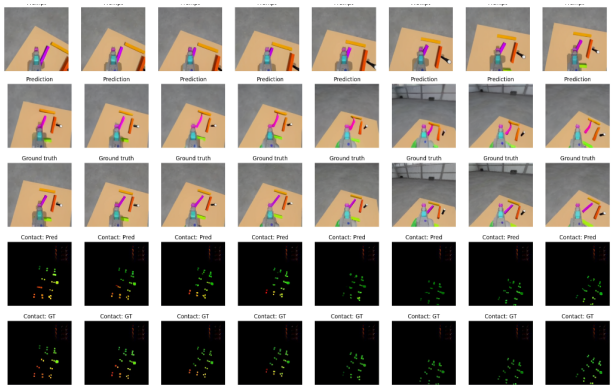


(b) End-effector moving right, hovering above FEA beam.

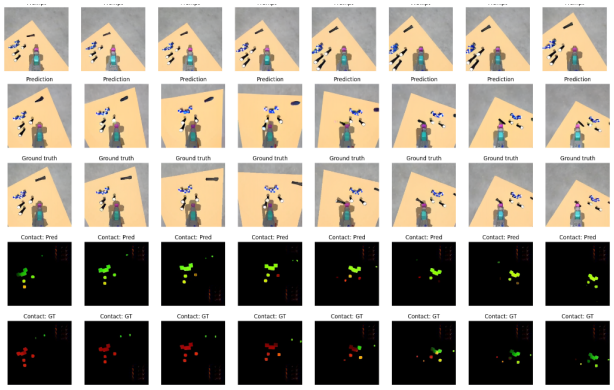


(c) End-effector moving right.

Figure 7.3: Spatial coherence in non-contact scenarios (`fea_flashlight`). Row 1: prompt; Rows 2–3: predicted RGB; Rows 4–5: predicted contact splat. Time progresses left to right.



(a) `fea_flashlight`: gripper collides with deformable beam.



(b) `flashlight_box`: gripper pushes flashlight leftward.

Figure 7.4: Predictions during contact events. Layout as in Figure 7.3. Post-contact frames exhibit increased blurriness.

7.4 VLM-as-Judge: Evaluating Physics Fidelity in World Model Predictions

Standard pixel-level metrics — Fréchet Video Distance (FVD) [173], perceptual similarity (LPIPS) [172], and SSIM — are the dominant tools for evaluating learned world models and video generative models [38, 174, 175]. These metrics measure appearance fidelity but are blind to physical plausibility: a world model that hallucinates an object sliding through a wall can achieve low LPIPS while exhibiting physically catastrophic behavior. Brooks et al. (Sora) documented this gap directly, noting that their model sometimes violates contact plausibility despite producing visually compelling frames [176]. Huang et al. (VBench [177]) argued for decomposing video evaluation into semantic dimensions including "physical correctness," but their benchmark targets open-ended video generation rather than action-conditioned world model predictions against ground-truth simulation.

We propose a *VLM-as-Judge* evaluation protocol to assess whether a learned world model preserves contact physics in its predictions. Rather than measuring pixel distances, the protocol queries a Vision-Language Model directly: given a strip of frames showing a predicted future, does the sequence faithfully represent the contact behavior of the underlying simulation? This approach draws on the LLM-as-judge methodology of Zheng et al. [178] and the structured rubric design of Prometheus [179, 180], and proposes a rubric-based VLM scoring scheme aggregated via AUC as an offline metric for world model contact-physics fidelity.

The evaluation protocol presented here is an *offline metric* applied at the dataset level to compare world model checkpoints and track training progress, complementary to any online collision-checking used during planning.

7.4.1 Limitations of Standard Pixel-Level Metrics for Contact Evaluation

FVD [173] compares distributions of I3D features across video clips; LPIPS [172] uses deep VGG or AlexNet features to measure perceptual similarity between frame pairs. Both are insensitive to contact plausibility by construction. Consider a predicted clip in which a rigid object smoothly penetrates a table surface: if pixel texture is sharp and the trajectory is smooth, FVD and LPIPS may be low despite the physical violation. Conversely, a physically correct prediction that involves rapid contact-induced blurring — a common artifact in discrete-token world models — may be penalized.

VBench [177] acknowledged this by proposing multi-dimensional evaluation that includes "physical correctness," but targets generative models with no ground-truth dynamics. PhyCoBench [181] explicitly includes collision as one of seven physical-principle evaluation categories, and VideoPhy [182] examines solid-solid contact, establishing precedent that VLMs can assess collision-related phenomena. Neither paper addresses the *paired-fidelity task* — evaluating a world model against ground-truth simulation trajectories. Table 7.3 summarizes where the proposed contribution sits relative to prior work.

7.4.2 VLM-AUC Evaluation Metric and Experimental Setup

7.4.2.1 16-Frame Strip

Each evaluation sample is a single image containing 16 frames arranged left to right in temporal order. The first 8 frames are ground-truth context frames rendered from the DreamerBench simulator. The last 8 frames are the frames under evaluation — either additional ground-truth frames (the ceiling condition) or world model predictions. This layout is compact enough to fit within a VLM context as a single image while providing enough temporal coverage to observe contact onset, sustained displacement, and settling.

Table 7.3: Novelty of the VLM-as-Judge evaluation protocol relative to prior work.

Research area	What exists	What is missing
World model evaluation	FVD, LPIPS, task success rate	Semantic / physical plausibility metrics
VLM-as-judge	NLP evaluation, video generation quality, robot task success	Application to world model prediction fidelity
Contact-rich simulation	Trajectory error, contact force error	Visual semantic evaluation of learned predictions

7.4.2.2 Collision Severity Score

The VLM returns a collision severity score on a 1–5 integer scale anchored symmetrically around 3 (genuinely uncertain):

1. Confidently no collision: clear spatial separation throughout, no contact evidence.
2. Probably no collision: no definitive contact evidence, though some ambiguity exists.
3. Genuinely uncertain: evidence for and against roughly balanced; insufficient to classify.
4. Probably collision: moderate contact evidence (displacement, deformation, or sustained pressing).
5. Confidently collision: unambiguous, sustained contact with clear physical consequence.

This recentered rubric avoids a systematic false-positive bias that arises when 3 out of 5 scale levels imply a positive outcome: under such a scale, a genuinely uncertain VLM defaults

to 3, which by that rubric design counts as a positive detection even though a score of 3 semantically reflects uncertainty rather than collision evidence. Recentering to a symmetric scale makes 3 the explicit neutral abstention and requires affirmative physical evidence for scores of 4 or 5.

7.4.2.3 VLM-AUC Metric

Each evaluation sample is scored independently by five prompt variants (Section 7.4.2.4), and the five integer scores are averaged to yield a continuous ensemble score in $[1, 5]$. This mean score is the VLM's evidence estimate for collision presence in that sample.

Human annotators label each sample using a binary interface: $y \in \{0, 1\}$, where 1 denotes collision and 0 denotes no collision. Each evaluation set contains 72 human-labeled samples; due to the stochastic nature of the data collection process, the split between collision-positive and collision-negative cases is not guaranteed to be equal across scenarios.

The Area Under the ROC Curve (AUC) is computed between the continuous VLM ensemble scores and the binary human labels. AUC measures how often the VLM correctly ranks a collision sample above a non-collision sample: given one sample from each class, AUC equals the probability that the collision sample receives a higher ensemble score, and takes values in $[0, 1]$. A key motivation for adopting AUC over accuracy-based metrics is its invariance to class imbalance: because AUC is equivalent to the normalized Wilcoxon–Mann–Whitney statistic [183], it measures rank-discriminability between the two classes independently of their relative proportions, making it a principled choice when the collision-to-non-collision ratio varies across scenarios.

Formally, let $\mathcal{P} = \{i : y_i = 1\}$ and $\mathcal{N} = \{j : y_j = 0\}$ denote the collision-positive and collision-negative index sets, respectively, and let $s_i \in [1, 5]$ be the ensemble score for sample i. The AUC is computed as the normalized Wilcoxon–Mann–Whitney statistic,

$$\text{AUC} \;=\; \frac{1}{|\mathcal{P}|\,|\mathcal{N}|} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} \Big[\mathbf{1}(s_i > s_j) + \tfrac{1}{2}\,\mathbf{1}(s_i = s_j) \Big], \tag{7.56}$$

where ties receive half-credit. Because the ensemble score s_i is a continuous average of five integer responses, strict ties are rare in practice, and Eq. (7.56) is equivalent to the trapezoidal area under the empirical ROC curve swept by thresholding s_i.

The intended ordering of the three conditions is:

$$\text{VLM-AUC(GT)} \;\geq\; \text{VLM-AUC(Epoch 10)} \;\geq\; \text{VLM-AUC(Epoch 1)}. \tag{7.57}$$

Ground-truth frames are sharp and physically correct, so the VLM discriminates collision from non-collision well — VLM-AUC(GT) is the ceiling. Epoch-10 world model predictions are slightly blurry after 10 training epochs, slightly compressing scores and reducing AUC. Epoch-1 predictions are severely degraded (trained on only 12 hours of data in a single epoch), triggering the skepticism clause and compressing scores toward the neutral band, further reducing AUC. To the extent that Eq. (7.57) holds across model–scenario pairs, the metric provides signal about training progress and may serve as a principled model-selection criterion.

7.4.2.4 Prompt Ensemble and Engineering

To reduce sensitivity to prompt wording — a well-documented failure mode in LLM-as-judge settings [178, 184] — five prompt variants (P1–P5) are run per sample and their scores averaged. Averaging across multiple prompt phrasings stabilizes the ensemble score and reduces variance attributable to any single elicitation strategy, a practice standard in LLM evaluation methodology [178]. Figure 7.5 summarizes the full prompt composition: a shared template supplies the scene context, collision semantics, physical filters, and few-shot calibration examples common to all variants; each of the five variant-specific bodies then applies a distinct elicitation strategy to the same visual evidence at runtime.

All five variants share a common specification structure containing the following components:

- **Scene and camera description.** Identifies the robot arm, tabletop items, and the ego-mounted camera. Explicitly instructs the VLM to distinguish real item displacement from global frame shifts caused by camera motion alone: if all items shift together while maintaining their relative arrangement, only camera motion is present; if some items move while others remain stationary, real displacement has occurred.

- **Collision definition.** Restricts evidence to manipulator-to-item contact; item-to-item and item-to-table contact do not count. Requires displacement, deformation, or bending to occur in proximity to the manipulator; peripheral scene changes are excluded. A high-clearance pass-through — where the manipulator moves near but clearly above an item with no new displacement, rotation, or deformation — is explicitly not a collision.

- **Physical consistency requirement.** Real displacement follows a smooth trajectory; an item pushed in frame 10 should continue moving in frames 11–12, not snap back or vanish. Flickering, snapping, or randomly appearing and disappearing items are prediction artifacts, not collision evidence.

- **Degraded-frame skepticism.** When future frames are noticeably blurrier or noisier than context frames, the VLM must raise its evidence threshold and require stronger physical evidence before scoring above 3.

- **Few-shot examples.** Six annotated examples span the main ambiguity cases: no object response, camera-motion-only appearance, rigid-object displacement, deformable-object deflection, apparent 2D overlap without contact, and prediction artifact.

- **Default no-collision assumption.** The VLM is instructed to start from the assumption that no collision occurred and to increase the score only when affirmative evidence contradicts this default.

The five variants differ in elicitation strategy:

- **P1 (Baseline).** Direct scoring with the standard rubric and no additional elicitation instruction.
- **P2 (Evidence-first).** Requires the VLM to complete a three-step chain of thought — (1) describe the manipulator trajectory, (2) list and assess contact signs for physical consistency, (3) evaluate evidence clarity — before assigning a score. Encourages grounded reasoning rather than holistic impression.
- **P3 (Conservative).** Adds an explicit constraint: scores of 4 or 5 are permitted only when the collision evidence is visually unambiguous and physically consistent across multiple frames; all ambiguous cases must be scored 3 or below.
- **P4 (Future-only).** Reinforces the temporal boundary: context frames (1–8) are for scene understanding only; all contact evidence must come exclusively from future frames (9–16).
- **P5 (Counterfactual).** After scoring, the VLM must state what evidence would need to be absent for the score to drop by one level, verifying that the assigned score is not inflated by incidental features.

The ensemble design reflects two systematic failure modes identified during development, both specific to VLM-as-judge evaluation of physical reasoning.

Rubric anchoring bias. An earlier prompt version used a rubric in which three of five scale levels (3, 4, 5) implied a positive (collision) outcome. When the VLM was genuinely uncertain, it defaulted to a score of 3, which under that rubric design counted as a positive detection — despite 3 semantically representing uncertainty rather than collision evidence — thereby inflating false-positive rates. The fix is a symmetric rubric in which 3 is the explicit neutral/uncertain anchor, scores of 4–5 require affirmative physical evidence, and scores of 1–2 indicate negative confidence. This eliminates the positive skew without changing the underlying VLM capability.

Artifact conflation. Early-epoch world model predictions frequently produce blurry or flickering frames. Without explicit guidance, VLMs may interpret rendering artifacts as motion evidence and over-detect collisions. Two countermeasures are embedded in the common prompt header: (a) a *physical consistency* clause requiring smooth, sustained trajectories as evidence of real contact and explicitly excluding flickering or snapping items; and (b) a *degraded-frame skepticism* clause that instructs the VLM to raise its evidence threshold when future frames are noticeably blurrier than context frames. Together, these produce the expected score compression in early-epoch predictions: if the future is severely degraded, the VLM abstains near 3 for both collision and non-collision samples, appropriately signaling low model quality rather than false detections.

These failure modes are analogous to the verbosity and position biases documented in the NLP LLM-as-judge literature [178, 184], and suggest that rubric design for physical-reasoning judges requires domain-specific calibration rather than direct transfer of general-purpose evaluation prompts.

7.4.2.5 Models and Evaluation Datasets

We evaluate three open-weight vision-language models:

- **Llama 3.2 90B Vision Instruct** [185] (Meta, released September 2024) — a dense 88.8B-parameter vision-language model built on the Llama 3 architecture, extended with a vision adapter to support image understanding. The model provides a 128K-token context window and is instruction-tuned for strong visual grounding across a broad range of multimodal tasks.

- **Gemma 3 27B IT** [186] (Google DeepMind, released March 2025) — a dense 27.4B-parameter instruction-tuned vision-language model trained on 14 trillion tokens. It supports a 130K-token input context and is designed to balance cost and capability for multimodal evaluation.

- **Qwen3.5 122B A10B** [187] (Alibaba, released February 2026) — a mixture-of-experts vision-language model with 122B total parameters and approximately 10B active parameters per forward pass, distributing computation across 256 experts with 8 routed per token. The model supports a 262K-token context window and is designed for efficient multimodal inference at large scale.

All models are queried via API at temperature 0.2, introducing a small amount of entropy to encourage broader reasoning paths while maintaining near-deterministic outputs. Full prompt text is logged for reproducibility.

Evaluation covers four DreamerBench scenarios: `fea_flashlight`, `flashlight_coca`, `flashlight_box`, and `waterbottle_coca`. Each scenario contributes 72 human-labeled samples; the split between collision-positive and collision-negative cases reflects the natural frequency of contact events in the collected data and is not enforced to be equal.

Three evaluation conditions are compared for each model–scenario pair. **Ground Truth (GT)**: the VLM judges 16-frame strips drawn entirely from ground-truth simulator frames; this is the ceiling condition. **Epoch 10**: strips whose last 8 frames are world-model predictions after 10 full training epochs on the DreamerBench dataset; the model has undergone substantial training and produces visually improved rollouts. **Epoch 1**: strips whose last 8 frames are world-model predictions after a single training epoch on approximately 12 hours of collected data; the model is undertrained and produces noticeably degraded, artifact-laden frames.

7.4.3 Results and Discussion

Table 7.4 reports VLM-AUC across all three evaluation conditions—ground-truth frames (*GT*, ceiling), Epoch-10 world-model predictions, and Epoch-1 world-model predictions—for all three VLMs and all four DreamerBench scenarios, with five independent runs per model–scenario pair. GT values above 0.5 indicate that a VLM can discriminate collision quality from RGB frames alone, without access to the contact splat channel. The *Overall* rows at the bottom of each model block pool all 4 tasks $\times$ 5 runs = 20 raw values per condition; at this aggregate

level, the expected ranking VLM-AUC(GT) > VLM-AUC(Ep. 10) > VLM-AUC(Ep. 1) holds for both Llama 3.2 90B and Qwen3.5 122B, providing encouraging support for VLM-AUC as a training-progress metric. Gemma 3 27B, the smallest of the three models, maintains the ceiling property—GT achieves the highest overall AUC—though the epoch-level ordering between Ep. 10 and Ep. 1 is less consistent, as discussed below.

The *Overall* rows aggregate all 4 tasks × 5 runs = 20 raw values per condition and provide the clearest aggregate signal. All three models place GT at the top of the overall ranking, confirming the ceiling interpretation across model families, scales, and architectures.

Llama 3.2 90B shows the most consistent pattern. The expected ordering VLM-AUC(GT) > VLM-AUC(Ep. 10) > VLM-AUC(Ep. 1) holds cleanly at the aggregate level (0.664 > 0.608 > 0.572), and is maintained in three of four individual scenarios. The sole task-level exception is `waterbottle_coca`, where Ep. 1 (0.595) edges Ep. 10 (0.573) by less than one standard deviation — a minor inversion that does not disturb the overall trend. The GT-to-Epoch-1 margin spans roughly 0.08–0.09 AUC points across all four scenarios, and run-to-run variance stays modest (std ≤ 0.065), providing solid empirical support for the ordering criterion in Eq. (7.57).

Qwen3.5 122B also satisfies the overall ordering (0.717 > 0.673 > 0.659), with GT achieving the highest aggregate AUC among all three models. Task-level orderings are less uniform: `fea_flashlight` sees Ep. 10 slightly exceed GT (0.786 vs. 0.755), and `flashlight_box` yields the pattern Ep. 1 > GT > Ep. 10. These per-task fluctuations reflect modest epoch-level effect sizes relative to within-task variance; pooling across all 20 values surfaces the underlying trend reliably. The high overall GT score confirms that Qwen3.5 122B is a sensitive discriminator of collision quality in ground-truth frames, and the aggregate ordering demonstrates that VLM-AUC tracks model improvement over training for this architecture as well.

Gemma 3 27B, the smallest model evaluated at 27B parameters compared to 90B and 122B for the other two, presents a more nuanced picture. GT achieves the highest overall AUC (0.565), confirming that the ceiling property holds even at this scale, but the epoch-level

Table 7.4: Per-run VLM-AUC across all models, scenarios, and prompting conditions. *GT*: ground-truth frames (ceiling condition); *Ep. 10*: world-model predictions after 10 training epochs; *Ep. 1*: world-model predictions after 1 training epoch (12 hours of data). Bold *Mean±Std* rows give per-task mean ± std over 5 runs; italic *Overall* rows aggregate all 20 raw values (4 tasks × 5 runs) per condition.

Model	Run	fea_flashlight			flashlight_box			flashlight_coca			waterbottle_coca		
		GT	Ep. 10	Ep. 1	*GT*	Ep. 10	Ep. 1	*GT*	Ep. 10	Ep. 1	*GT*	Ep. 10	Ep. 1
Llama 3.2 90B	Run 1	0.537	0.610	0.488	0.710	0.554	0.594	0.640	0.636	0.525	0.631	0.512	0.531
	Run 2	0.604	0.556	0.485	0.667	0.667	0.549	0.703	0.616	0.623	0.699	0.531	0.574
	Run 3	0.652	0.511	0.541	0.792	0.682	0.677	0.664	0.655	0.563	0.725	0.575	0.568
	Run 4	0.583	0.660	0.534	0.614	0.644	0.587	0.715	0.620	0.606	0.716	0.640	0.676
	Run 5	0.583	0.604	0.467	0.687	0.669	0.605	0.705	0.617	0.622	0.648	0.608	0.624
	Mean±Std	**0.592±0.042**	**0.588±0.057**	**0.503±0.033**	**0.694±0.065**	**0.643±0.052**	**0.602±0.047**	**0.685±0.032**	**0.629±0.017**	**0.588±0.043**	**0.684±0.042**	**0.573±0.053**	**0.595±0.056**
	Overall					GT = **0.664±0.061**	Ep. 10 = **0.608±0.052**	Ep. 1 = **0.572±0.059**					
Qwen3.5 122B	Run 1	0.722	0.782	0.630	0.650	0.532	0.660	0.805	0.624	0.654	0.643	0.695	0.689
	Run 2	0.746	0.810	0.608	0.692	0.625	0.702	0.810	0.600	0.635	0.758	0.660	0.683
	Run 3	0.765	0.794	0.612	0.595	0.612	0.748	0.706	0.612	0.648	0.795	0.694	0.682
	Run 4	0.722	0.778	0.637	0.706	0.652	0.644	0.679	0.664	0.657	0.585	0.620	0.639
	Run 5	0.819	0.766	0.643	0.619	0.631	0.664	0.816	0.608	0.649	0.704	0.706	0.692
	Mean±Std	**0.755±0.040**	**0.786±0.017**	**0.626±0.015**	**0.652±0.047**	**0.610±0.046**	**0.684±0.042**	**0.763±0.065**	**0.622±0.025**	**0.649±0.008**	**0.697±0.085**	**0.675±0.035**	**0.677±0.022**
	Overall					GT = **0.717±0.073**	Ep. 10 = **0.673±0.077**	Ep. 1 = **0.659±0.033**					
Gemma 3 27B	Run 1	0.699	0.460	0.582	0.505	0.441	0.541	0.485	0.499	0.404	0.632	0.500	0.685
	Run 2	0.629	0.465	0.568	0.521	0.464	0.598	0.459	0.543	0.357	0.632	0.528	0.669
	Run 3	0.622	0.434	0.558	0.506	0.434	0.522	0.466	0.488	0.344	0.620	0.494	0.630
	Run 4	0.655	0.432	0.568	0.453	0.363	0.547	0.508	0.515	0.356	0.619	0.518	0.651
	Run 5	0.673	0.437	0.549	0.519	0.358	0.410	0.471	0.514	0.356	0.634	0.507	0.657
	Mean±Std	**0.656±0.032**	**0.446±0.016**	**0.565±0.012**	**0.501±0.028**	**0.412±0.048**	**0.524±0.069**	**0.478±0.019**	**0.512±0.021**	**0.363±0.023**	**0.627±0.007**	**0.509±0.014**	**0.658±0.021**
	Overall					GT = **0.565±0.082**	Ep. 10 = **0.470±0.051**	Ep. 1 = **0.528±0.115**					

ordering is reversed at the aggregate level: Ep. 1 (0.528) exceeds Ep. 10 (0.470). Examining individual tasks, `fea_flashlight` shows a clear GT > Ep. 1 > Ep. 10 pattern consistent with training progress, while `flashlight_coca` is the most challenging scenario, with GT falling slightly below chance (0.478), suggesting this contact configuration is less visually discriminable. The substantial run-to-run variance (std = 0.115 for the Ep. 1 aggregate) indicates that Gemma 3 27B's scoring distributions are more sensitive to prompt-response variation than the larger models, which is consistent with the general finding in the VLM literature that larger models produce more stable and calibrated judgments on fine-grained visual discrimination tasks. Despite the noisy epoch-level ordering, the ceiling property is robustly satisfied, and Gemma 3 27B successfully identifies ground-truth quality as the highest-scoring condition overall.

Comparing ceiling scores across models highlights both the strength and the breadth of the VLM-AUC signal. The GT condition represents the best-case discrimination achievable by each model: world-model degradation is entirely absent, and the VLM judges strips drawn from the most physically accurate frames available. Qwen3.5 122B achieves the highest ceiling VLM-AUC (0.717), followed by Llama 3.2 90B (0.664) and Gemma 3 27B (0.565). All three values exceed 0.5, confirming that each model can correctly rank collision-positive clips above collision-negative clips from ground-truth RGB frames alone, without access to the contact splat channel. That none of the ceiling scores approaches 1.0, however, reflects two compounding sources of limitation that are inherent to the evaluation design rather than to the metric itself. First, the vision and physical reasoning capability of current-generation VLMs is bounded: contact physics involves subtle spatial relationships, brief deformations, and transient displacements that can challenge even a capable vision-language model, and a model's ability to reliably distinguish collision-positive from collision-negative samples ultimately depends on how well its visual encoder and reasoning chain can parse these cues from a static composite image without simulator-level physical grounding. Second, the evaluation pipeline introduces information compression on two fronts: spatially, 16 frames

are tiled side by side into a single composite image, substantially reducing the per-frame resolution available to the judge; and temporally, frames are subsampled at an effective rate below 2 Hz before strip assembly, so brief or rapidly evolving contact events may span only one or two frames in the evaluation window, limiting the temporal evidence available for discrimination. Together, the bounded reasoning capacity of the VLM judge and the spatial–temporal compression of the pipeline impose a metric ceiling that is independent of world-model quality; the observed gap between the best ceiling VLM-AUC (0.717) and the theoretical maximum of 1.0 should therefore be interpreted as a joint property of the evaluation modality and the judge's capability, rather than a shortcoming of the metric itself. The lower ceiling score of Gemma 3 27B is consistent with its reduced parameter count relative to the other two models, but the fact that it remains above chance demonstrates that the collision discrimination signal embedded in ground-truth simulator frames is accessible to models well below the 90B–122B scale. This cross-model consistency at the ceiling level provides confidence that VLM-AUC is a broadly applicable evaluation tool rather than one that depends on choosing a particularly capable judge.

Taken together, the five-run evaluation across all three models provides an encouraging validation of VLM-AUC as a training-progress metric. The aggregate GT $>$ Ep. 10 $>$ Ep. 1 ordering holds for both Llama 3.2 90B and Qwen3.5 122B, and the ceiling property holds for all three models. The results demonstrate that VLM-AUC can track world-model improvement over training, with signal strength correlating with model scale.

In the interest of reproducibility, the complete evaluation artifacts are archived in the DreamerBench dataset repository [160]. These include, for every query: the full inference log, wall-clock request time, backend endpoint and model identifier, the VLM's chain-of-thought reasoning trace, the raw text output, and the parsed integer score assigned to each sample. Releasing these artifacts enables independent replication of the reported VLM-AUC values, facilitates comparison against alternative prompting strategies or judge models, and supports future ablation studies on rubric design and ensemble composition.

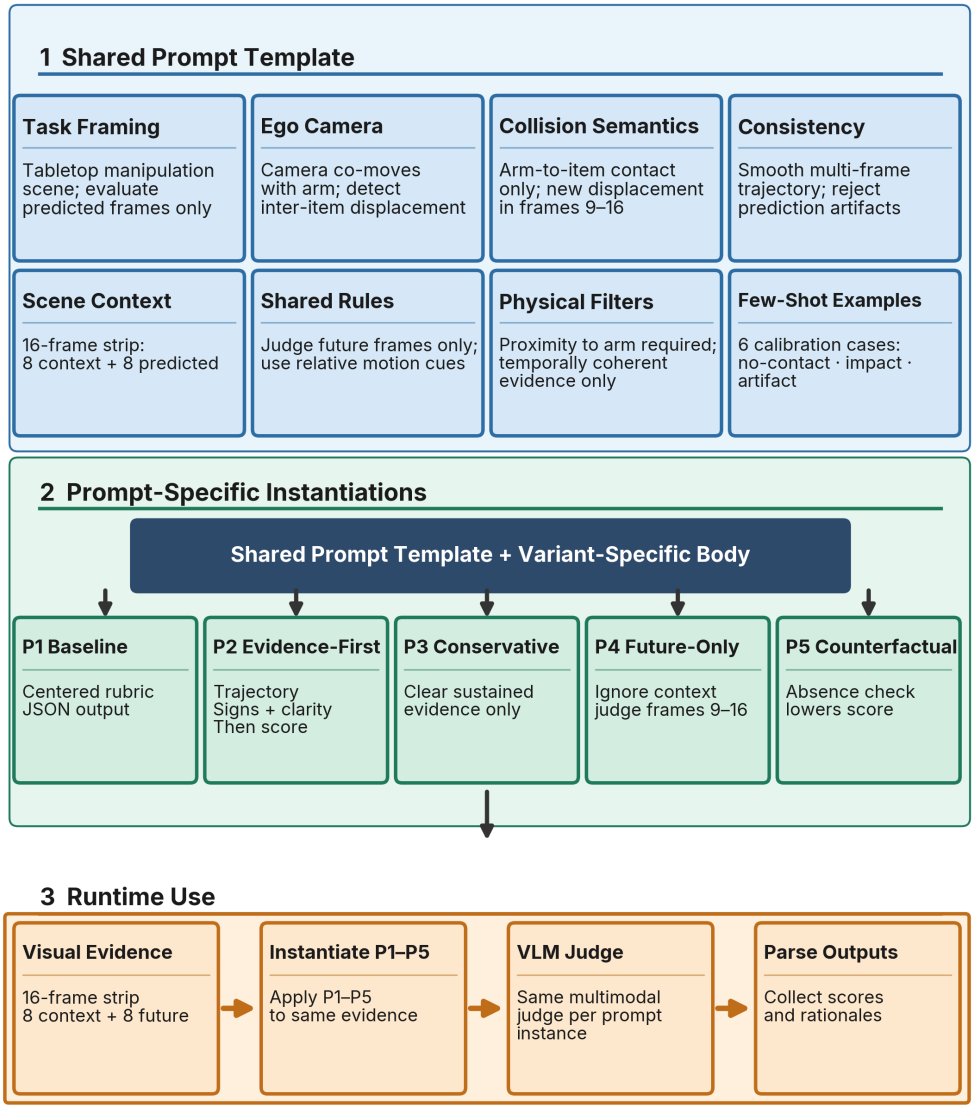


Figure 7.5: Prompt composition for the VLM-as-judge ensemble. **(1) Shared Prompt Template:** all five variants share a common header specifying scene context, ego-camera handling, collision semantics, physical consistency requirements, and six few-shot calibration examples. **(2) Prompt-Specific Instantiations:** five variants (P1–P5) each apply a distinct elicitation strategy—baseline direct scoring, evidence-first chain-of-thought, conservative high-score gating, strict future-only temporal boundary, and counterfactual score verification—to the same shared template. **(3) Runtime Use:** a 16-frame strip is assembled, all five prompts are instantiated and sent to the same VLM judge, and the returned scores and rationales are parsed and averaged into the ensemble score.

8 CONTRIBUTION AND FUTURE WORK

8.1 Summary of Contributions

- We developed a TL deformable-body dynamics formulation in which the kinematics and constitutive response are written consistently in terms of the deformation gradient $\mathbf{F}$ and its linearization, while adopting the standard structure of finite-strain continuum mechanics [12, 188].

- We expressed the position field as $\mathbf{r}(\mathbf{u};t) = \mathbf{N}(t)\mathbf{s}(\mathbf{u})$, enabling a factorization of the deformation gradient $\mathbf{F}(\mathbf{u};t) = \mathbf{N}(t)\mathbf{H}(\mathbf{u})$ that cleanly separates time-dependent nodal unknowns from precomputable, element-specific geometric quantities. This compact formulation carries through the derivation of internal forces, constraint Jacobians, and consistent linearizations [43, 84].

- We formulated deformable-body kinematic joints by composing scalar primitives of DP1, DP2, CD, and DIST type directly on isoparametric finite elements. Within this setting, the associated element-level Jacobian blocks were derived, their accumulation into composite-joint operators was discussed, and the main linearization issues associated with dot-product constraints were identified, including row scaling and the contribution of constraint curvature to the Newton system.

- We expressed the constitutive-discretization interface through the first Piola–Kirchhoff stress $\mathbf{P}(\mathbf{F})$ and its material derivative $\partial\mathbf{P}/\partial\mathbf{F}$, which makes the internal-force and tangent constructions largely independent of element topology. In this form, adding a new hyperelastic or viscoelastic material model amounts to specifying $\mathbf{P}(\mathbf{F})$ and its derivative, while the surrounding integration and assembly machinery remains unchanged [82, 83].

- We derived, within one virtual-work framework, the contributions of inertia, mass-

distributed force fields, internal forces, concentrated loads, prescribed surface tractions, frictional contact forces, and bilateral constraint reactions. The corresponding quadrature construction was written in a form compatible with isoparametric finite elements and with the deformation-gradient-based constitutive interface.

- We cast the backward-Euler constrained TL-FEA step as a velocity-level augmented-Lagrangian optimization problem, in which the stationarity conditions recover the discrete equations of motion while bilateral constraints are enforced through multiplier and penalty terms. This provides a formulation-level link between the mechanics model and the nonlinear solution procedure used in practice.

- We precomputed all reference-configuration quantities — shape-function values and gradients at quadrature points, element Jacobians, and the consistent mass matrix — once and cached them in device memory, and constructed the nonzero sparsity pattern of every global operator (mass matrix, constraint Jacobian, Newton Hessian) once at initialization, holding it fixed for the duration of the simulation. This fixed-sparsity strategy eliminates repeated symbolic analysis, allows cuDSS to reuse its fill-reducing permutation and factor buffers, and reduces each subsequent Newton iteration to a cheaper numerical refactorization.

- We introduced a two-stage GPU parallelization for internal force and tangent stiffness and coupled it with a Newton solver within an augmented-Lagrangian outer loop. Stage 1 assigns one GPU thread per element–quadrature-point pair to evaluate and cache the first Piola–Kirchhoff stress; Stage 2 accumulates force contributions per element–node pair via atomic scatter, with tangent stiffness reusing the Stage 1 cache to eliminate redundant constitutive evaluations. Bilateral constraints are enforced by alternating the inner Newton solve with a dual-ascent multiplier update; the Newton system is assembled on the GPU and solved with cuDSS. Systematic benchmarks across three element types (T10, ANCF3243, ANCF3443) and six mesh resolutions

show approximately one order of magnitude reduction in real-time factor relative to CPU baselines at the largest resolutions tested.

- We developed a BVH-free GPU collision detection pipeline in which triangle meshes are decomposed into soups and contact candidates are identified through a bin-based spatial partitioning that maps directly to GPU thread blocks, avoiding per-step tree updates. Narrow-phase resolution uses projection-based triangle–triangle overlap queries, and contact normals are smoothed by patch-level area-weighted reduction. We structured the algorithm as two asynchronous CPU-managed threads — a kinematics thread for broad-phase detection and a dynamics thread for narrow-phase resolution — that communicate through shared buffers, so that collision detection cost overlaps with physics integration.

- We validated the frictional contact model against closed-form rigid-body predictions through a quasi-static brick-on-slope experiment spanning four Coulomb regimes and a dynamic oblique-impact sweep comparing the tangential coefficient of restitution against analytical theory across a range of impact angles. We validated the bilateral constraint and joint-force recovery machinery through deformable double-pendulum simulations with revolute and spherical joints, comparing against Project Chrono rigid-body references and verifying ALM-recovered joint forces against analytical quasi-static equilibrium predictions.

- We curated and released DreamerBench [160], a 12-hour human-supervised contact-rich robot interaction dataset, on Hugging Face. We introduced a contact-encoding method that represents friction and contact signals as continuous image-like data, enabling a world model to learn this modality alongside RGB video generation while keeping LPIPS losses in the same range as video-only generation. We also presented and open-sourced ChronoDreamer [189], a world-model implementation that learns underlying dynamics from encoded video and contact-splat frames, as demonstrated through quantitative

and qualitative analysis.

- We proposed VLM-AUC, an evaluation metric that uses a vision-language model as an automated judge to assess contact-physics fidelity in world model predictions. The VLM scores each predicted video rollout against a structured rubric anchored to collision evidence, and the resulting scores are aggregated via AUC against binary human collision labels, providing a complementary means of assessing generation quality beyond pixel-level reconstruction losses. We validated the metric by evaluating ChronoDreamer checkpoints at two training stages and on ground-truth data, across three open-weight VLMs — Llama 3.2 90B, Gemma 3 27B, and Qwen3.5 122B A10B — demonstrating that the metric provides an interpretable signal of model improvement over the course of training.

8.2 Limitations and Future Work

- **Explicit contact coupling.** The frictional contact forces are treated as external loads within the augmented Lagrangian formulation — assembled into $\mathbf{f}_{\text{ext}}$ and held fixed during the inner Newton solve rather than being incorporated into the Hessian $\mathbf{H}$. This explicit operator-splitting treatment of contact decouples the contact constitutive response from the structural Newton system, avoiding the complexity of computing and assembling a contact Jacobian, but at the cost of conditional stability: large contact stiffness values or rapidly evolving contact states can induce oscillations or divergence unless the time step is sufficiently small. A fully implicit treatment — one that linearizes the contact force model and contributes its Jacobian to $\mathbf{H}$ — would remove this restriction and is a natural direction for future work.

- **AdamW solver hyperparameter sensitivity.** The AdamW solver was implemented and explored as a first-order alternative to Newton precisely because, in the Total Lagrangian FEA setting, it requires only gradient evaluations — no Hessian assembly

and no sparse factorization. Each inner iteration reduces to an embarrassingly parallel per-DOF update: every thread independently evaluates its gradient entry and applies the bias-corrected AdamW rule without any inter-thread communication, making the solver a natural fit for GPU execution. However, the first-order convergence rate causes the gradient norm to plateau within any practical iteration budget at large problem sizes, making tight tolerances unattainable without a prohibitive increase in iteration count. Beyond convergence rate, the solver is sensitive to the choice of hyperparameters — in particular the learning rate and weight decay coefficient — whose optimal values depend on element type, material model, mesh resolution, and time step size. The benchmarks in Chapter 6 demonstrate competitive performance for the structured beam-sagging cases, but these parameters were tuned specifically for those configurations and do not transfer automatically to new problem setups. Developing adaptive or self-tuning strategies for the AdamW hyperparameters is a natural direction for future work.

- **Shared memory staging for GPU assembly kernels.** During implementation, several strategies employing GPU shared memory as an intermediate buffer within the internal force and Hessian assembly kernels were explored, motivated by the expectation that staging intermediate results in on-chip shared memory would reduce global memory traffic and improve kernel throughput. In practice, however, these approaches yielded negative or negligible performance gains in benchmarks. A key observation was that each element carries a substantial amount of element-specific data — precomputed reference-configuration quantities, quadrature-point stress tensors, and shape function gradient arrays — and loading all of this into shared memory requires a large shared memory allocation per thread block. On the RTX 5090, this high per-block shared memory footprint directly reduces the number of thread blocks that can reside concurrently on a streaming multiprocessor (SM), lowering SM occupancy and offsetting any latency benefit from on-chip staging. A more targeted investigation — identifying which specific intermediate quantities benefit from shared memory staging without significantly

increasing the per-block footprint — is left as future work.

- **World model scale and evaluation coverage.** ChronoDreamer was trained on DreamerBench, which comprises 12 hours of simulator-generated interaction data. While sufficient to demonstrate the contact-encoding and VLM-AUC contributions, this volume is modest relative to the scale of contemporary world foundation models: the NVIDIA Cosmos model, for instance, was trained on 20 million hours of video spanning diverse real-world domains [161]. The gap in data volume directly limits generalization, particularly for rare or complex contact events underrepresented in the current dataset. Compounding this, ChronoDreamer itself is a relatively small model of approximately 30–50M parameters, a deliberate constraint imposed by available compute, whereas production-scale world models operate at orders of magnitude greater capacity. Taken together, the limited data and model size mean that the results reported here should be interpreted as a proof-of-concept demonstration rather than a performance ceiling.

  On the evaluation side, the VLM-AUC metric was validated against two ChronoDreamer training checkpoints and ground-truth simulation frames. The ordering of AUC scores across these three conditions provides an encouraging signal that the metric tracks model improvement, but the ablation covers a narrow portion of the training trajectory and does not systematically vary prompt design, VLM backbone choice, or dataset scenario. A more comprehensive ablation — spanning additional checkpoints, a wider range of prompt variants, and multiple dataset splits — is needed to establish the metric's reliability as a general-purpose evaluation tool for contact-rich world models. The current evaluation also covers only three open-weight VLM architectures — Llama 3.2 90B, Gemma 3 27B, and Qwen3.5 122B A10B — and whether the metric's behavior generalizes to other model families, including closed-weight models, remains untested.

BIBLIOGRAPHY

[1] Zhou, Z., Arivoli, G., and SBEL, University of Wisconsin-Madison, 2025. Total-Lagrangian-FEA: Research code for total-lagrangian finite element method for flexible multibody dynamics. `https://github.com/uwsbel/Total-Lagrangian-FEA`. GitHub repository.

[2] Todorov, E., Erez, T., and Tassa, Y., 2012. "MuJoCo: A physics engine for model-based control". In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp. 5026–5033.

[3] Coumans, E., and Bai, Y., 2016–2019. PyBullet, a Python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`.

[4] Tasora, A., Serban, R., Mazhar, H., Pazouki, A., Melanz, D., Fleischmann, J., Taylor, M., Sugiyama, H., and Negrut, D., 2016. "Chrono: An open source multi-physics dynamics engine". In High Performance Computing in Science and Engineering, T. Kozubek, R. Blaheta, J. Šístek, M. Rozložník, and M. Čermák, eds., Springer International Publishing, pp. 19–49.

[5] Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., and State, G., 2021. Isaac gym: High performance gpu-based physics simulation for robot learning.

[6] Macklin, M., 2022. "Warp: A high-performance Python framework for GPU simulation and graphics". In NVIDIA GPU Technology Conference (GTC).

[7] Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O., 2021. Brax - a differentiable physics engine for large scale rigid body simulation.

[8] Liu, C. K., and Negrut, D., 2021. "The role of physics-based simulators in robotics". *Annual Review of Control, Robotics, and Autonomous Systems,* **4**(Volume 4, 2021), pp. 35–58.

[9] Glotzer, S. C., et al., eds., 2011. *International Assessment of Research and Development in Simulation-Based Engineering and Science*. World Scientific, Singapore.

[10] Ragani, A. F., Symington, I., Keene, R., and Stein, J. P., 2023. "Unveiling the next frontier of engineering simulation: Digital engineering in an AI world". *NAFEMS Benchmark Magazine,* **27**(3). Results of a joint NAFEMS–McKinsey survey on engineering simulation.

[11] Zienkiewicz, O., and Taylor, R., 2005. *The Finite Element Method for Solid and Structural Mechanics*, 6 ed. Butterworth-Heinemann, Oxford, UK.

[12] Bonet, J., and Wood, R. D., 2008. *Nonlinear Continuum Mechanics for Finite Element Analysis*, 2nd ed. Cambridge University Press, Cambridge, UK.

[13] Belytschko, T., Liu, W., and Moran, B., 2000. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons, Chichester, UK.

[14] Oliver, J., and O'nate, E., 1984. "A total lagrangian formulation for the geometrically nonlinear analysis of structures using finite elements. part i. two-dimensional problems: Shell and plate structures". *International Journal for Numerical Methods in Engineering,* **20**(12), pp. 2253–2281.

[15] Shabana, A. A., 1996. An Absolute Nodal Coordinate Formulation for the large rotation and deformation analysis of flexible bodies. Tech. Rep. MBS96-1-UIC, University of Illinois at Chicago. Technical report (ANCF original formulation).

[16] Nachbagauer, K., 2014. "State of the art of ANCF elements regarding geometric description, interpolation strategies, definition of elastic forces, validation and the

locking phenomenon in comparison with proposed beam finite elements". *Archives of Computational Methods in Engineering,* **21**(3), pp. 293–319.

[17] Mazhar, H., Heyn, T., and Negrut, D., 2011. "A scalable parallel method for large collision detection problems". *Multibody System Dynamics,* **26**, pp. 37–55. 10.1007/s11044-011-9246-y.

[18] Montanari, M., Petrinic, N., and Barbieri, E., 2017. "Improving the GJK algorithm for faster and more reliable distance queries between convex objects". *ACM Transactions on Graphics,* **36**(4), pp. 38:1–38:17.

[19] Govender, N., Wilke, D. N., and Kok, S., 2015. "Collision detection of convex polyhedra on the NVIDIA GPU architecture for the discrete element method". *Applied Mathematics and Computation,* **267**, pp. 810–829.

[20] Karras, T., 2012. "Maximizing parallelism in the construction of BVHs, octrees, and k-d trees". In Proceedings of High Performance Graphics 2012, The Eurographics Association, pp. 33–37.

[21] Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M.-P., Faure, F., Magnenat-Thalmann, N., Strasser, W., and Volino, P., 2005. "Collision detection for deformable objects". *Computer Graphics Forum,* **24**(1), pp. 61–81.

[22] Muratore, F., Treede, F., Gienger, M., Peters, J., et al., 2022. "Robot learning from randomized simulations: A review". *Frontiers in Robotics and AI,* **9**, p. 799893.

[23] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P., 2017. "Domain randomization for transferring deep neural networks from simulation to the real world". In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 23–30.

[24] Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P., 2018. "Sim-to-real transfer of robotic control with dynamics randomization". In IEEE International Conference on Robotics and Automation (ICRA), pp. 3803–3810.

[25] Chen, X., Hu, J., Jin, C., Li, L., and Wang, L., 2022. "Understanding domain randomization for sim-to-real transfer". In International Conference on Learning Representations (ICLR).

[26] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J., 2016. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age". *IEEE Transactions on Robotics,* **32**(6), pp. 1309–1332.

[27] Ehsani, K., Tulsiani, S., Gupta, S., Farhadi, A., and Gupta, A., 2020. "Use the force, Luke! Learning to predict physical forces by simulating effects". In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 224–233.

[28] Murthy, J. K., Macklin, M., Golemo, F., Voleti, V., Petrini, L., Weiss, M., Considine, B., Parent-Lévesque, J., Xie, K., Erleben, K., Paull, L., Shkurti, F., Nowrouzezahrai, D., and Fidler, S., 2021. "gradsim: Differentiable simulation for system identification and visuomotor control". In International Conference on Learning Representations.

[29] Byravan, A., Humplik, J., Hasenclever, L., Brussee, A., Nori, F., Haarnoja, T., Moran, B., Bohez, S., Sadeghi, F., Vujatovic, B., and Heess, N., 2023. "NeRF2Real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields". In IEEE International Conference on Robotics and Automation (ICRA), pp. 9362–9369.

[30] Abou-Chakra, J., Rana, K., Dayoub, F., and Suenderhauf, N., 2024. "Physically embodied Gaussian splatting: A realtime correctable world model for robotics". In 8th

Conference on Robot Learning (CoRL), Vol. 270 of *Proceedings of Machine Learning Research*, pp. 513–530.

[31] Torne, M., Simeonov, A., Li, Z., Chan, A., Chen, T., Gupta, A., and Agrawal, P., 2024. "Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation". In Robotics: Science and Systems (RSS).

[32] Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D. J., and Kavukcuoglu, K., 2016. "Interaction networks for learning about objects, relations and physics". In Advances in Neural Information Processing Systems (NeurIPS), Vol. 29, pp. 4502–4510.

[33] Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. W., 2020. "Learning to simulate complex physics with graph networks". In Proceedings of the 37th International Conference on Machine Learning (ICML), Vol. 119, PMLR, pp. 8459–8468.

[34] Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W., 2021. "Learning mesh-based simulation with graph networks". In International Conference on Learning Representations (ICLR).

[35] Ha, D., and Schmidhuber, J., 2018. World models.

[36] Ha, D., and Schmidhuber, J., 2018. "Recurrent world models facilitate policy evolution". In Advances in Neural Information Processing Systems (NeurIPS), Vol. 31, pp. 2450–2462.

[37] Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M., 2020. "Dream to control: Learning behaviors by latent imagination". In International Conference on Learning Representations (ICLR).

[38] Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T., 2023. "Mastering diverse domains through world models". *arXiv preprint arXiv:2301.04104*.

[39] Wu, P., Escontrela, A., Hafner, D., Goldberg, K., and Abbeel, P., 2023. "Daydreamer: World models for physical robot learning". In Proceedings of the Conference on Robot Learning (CoRL), Vol. 205 of *Proceedings of Machine Learning Research*, pp. 2223–2245.

[40] Bonet, J., Gil, A. J., and Wood, R. D., 2016. *Nonlinear solid mechanics for finite element analysis: statics*. Cambridge University Press, Cambridge, UK.

[41] Peng, Q., and Li, M., 2023. "Comparison of finite element methods for dynamic analysis about rotating flexible beam". *Nonlinear Dynamics,* **111**(15), pp. 13753–13779.

[42] Shabana, A. A., 1997. "Definition of the slopes and the finite element absolute nodal coordinate formulation". *Multibody System Dynamics,* **1**(3), pp. 339–348.

[43] Shabana, A. A., and Yakoub, R. Y., 2001. "Three dimensional absolute nodal coordinate formulation for beam elements: Theory". *ASME Journal of Mechanical Design,* **123**, pp. 606–613.

[44] Zhang, J., 2021. "A direct jacobian total lagrangian explicit dynamics finite element algorithm for real-time simulation of hyperelastic materials". *International Journal for Numerical Methods in Engineering,* **122**(20), pp. 5744–5772.

[45] Shabana, A. A., 2020. *Dynamics of Multibody Systems*, fifth ed. Cambridge University Press, Cambridge, England.

[46] Sifakis, E., and Barbic, J., 2012. "Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction". In *Acm siggraph 2012 courses*. pp. 1–50.

[47] Stickle, M. M., Molinos, M., Navas, P., Yagüe, Á., Manzanal, D., Moussavi, S., and Pastor, M., 2022. "A component-free lagrangian finite element formulation for large strain elastodynamics". *Computational Mechanics,* **69**(3), pp. 639–660.

[48] García-Vallejo, D., Mayo, J., Escalona, J. L., and Domínguez, J., 2004. "Efficient evaluation of the elastic forces and the jacobian in the absolute nodal coordinate formulation". *Nonlinear Dynamics,* **35**(4), pp. 313–329.

[49] Maqueda, L. G., and Shabana, A. A., 2007. "Poisson modes and general nonlinear constitutive models in the large displacement analysis of beams". *Multibody system dynamics,* **18**(3), pp. 375–396.

[50] Orzechowski, G., and Fraçzek, J., 2015. "Nearly incompressible nonlinear material models in the large deformation analysis of beams using ANCF". *Nonlinear Dynamics,* **82**, pp. 451–464.

[51] Kissel, A., Taves, J., and Negrut, D., 2022. "Constrained multibody kinematics and dynamics in absolute coordinates: A discussion of three approaches to representing rigid body rotation". *Journal of Computational and Nonlinear Dynamics,* **17**(10), 08, p. 101008. 101008.

[52] Betsch, P., and Steinmann, P., 2003. "Constrained dynamics of geometrically exact beams". *Computational Mechanics,* **31**(1), pp. 49–59.

[53] Sugiyama, H., Escalona, J. L., and Shabana, A. A., 2003. "Formulation of three-dimensional joint constraints using the absolute nodal coordinates". *Nonlinear Dynamics,* **31**(2), pp. 167–195.

[54] Georgii, J., and Westermann, R., 2005. "Interactive simulation of deformable bodies on GPUs". In Proceedings of the Conference on Simulation and Visualization (SimVis), SCS Publishing House, pp. 209–218.

[55] Macklin, M., Müller, M., Chentanez, N., and Kim, T.-Y., 2014. "Unified particle physics for real-time applications". *ACM Transactions on Graphics (TOG),* **33**(4), pp. 1–12.

[56] Hu, Y., Li, T.-M., Anderson, L., Ragan-Kelley, J., and Durand, F., 2019. "Taichi: a language for high-performance computation on spatially sparse data structures". *ACM Transactions on Graphics (TOG),* **38**(6), pp. 1–16.

[57] Faure, F., Duriez, C., Delingette, H., Allard, J., Gilles, B., Marchesseau, S., Talbot, H., Courtecuisse, H., Bousquet, G., Peterlik, I., et al., 2012. "Sofa: A multi-model framework for interactive physical simulation". In *Soft tissue biomechanical modeling for computer assisted surgery.* Springer, Berlin, Heidelberg, pp. 283–321.

[58] Corporation, N., 2025. NVIDIA cuDSS: GPU-Accelerated Direct Sparse Solver Library. `https://developer.nvidia.com/cudss`. Version 0.5.0, Preview Release.

[59] Géradin, M., and Rixen, D., 1994. *Mechanical vibrations : theory and application to structural dynamics*, 2nd ed. Wiley, Chichester, UK.

[60] Brüls, O., Cardona, A., and Arnold, M., 2012. "Lie group generalized-α time integration of constrained flexible multibody systems". *Mechanism and Machine Theory,* **48**, pp. 121–137.

[61] Baumgarte, J., 1972. "Stabilization of constraints and integrals of motion in dynamical systems". *Computer Methods in Applied Mechanics and Engineering,* **1**, pp. 1–16.

[62] Bertsekas, D. P., 1995. *Nonlinear Programming.* Athena Scientific, Belmont, MA.

[63] Nocedal, J., and Wright, S., 1999. *Numerical optimization.* Springer, New York, NY.

[64] Alnæs, M. S., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M. E., and Wells, G. N., 2015. "The FEniCS project version 1.5". *Archive of Numerical Software,* **3**(100), pp. 9–23.

[65] Mazier, A., Bilger, A., Forte, A. E., Peterlik, I., Hale, J. S., and Bordas, S. P. A., 2022. "Inverse deformation analysis: an experimental and numerical assessment using the FEniCS project". *Engineering with Computers,* **38**(5), pp. 4099–4113.

[66] Xue, T., Liao, S., Gan, Z., Park, C., Xie, X., Liu, W. K., and Cao, J., 2023. "JAX-FEM: A differentiable GPU-accelerated 3D finite element solver for automatic inverse design and mechanistic data science". *Computer Physics Communications,* **291**, p. 108802.

[67] Nachbagauer, K., Gruber, P., and Gerstmayr, J., 2012. "Structural and continuum mechanics approaches for a 3D shear deformable ANCF beam finite element: Application to static and linearized dynamic examples". *J. Comput. Nonlinear Dynam,* **8 (2)**, p. 021004.

[68] Yamashita, H., Valkeapää, A. I., Jayakumar, P., and Sugiyama, H., 2015. "Continuum mechanics based bilinear shear deformable shell element using Absolute Nodal Coordinate Formulation". *Journal of Computational and Nonlinear Dynamics,* **10**(5), p. 051012.

[69] Taylor, M. E., 2022. "Efficient implementation strategies for the absolute nodal coordinate formulation with linear and hyperelastic material laws". Ph.D thesis, The University of Wisconsin-Madison.

[70] Taylor, M., Serban, R., and Negrut, D., 2023. "Implementation implications on the performance of ANCF simulations". *International Journal of Non-Linear Mechanics,* **149**, p. 104328.

[71] Lees, A., and Wood, J., 2011. "Skills development and recording in engineering analysis and simulation — industry needs". In Proceedings of the NAFEMS World Congress.

[72] Wood, J., Prinja, N., and Morris, T., 2013. "Transferring simulation skills from other industries to nuclear". *Nuclear Future,* **9**(1), pp. 1–7.

[73] EASIT2: Engineering analysis and simulation innovation transfer. `https://www.nafems.org/about-us/projects/past-projects/easit2/`. Accessed 2025-11-25.

[74] Fadzli, F. E., Rahman, A., Abdullah, W. Z. W., et al., 2023. "A systematic literature review: Real-time 3D reconstruction method for telepresence system". *PLOS ONE,* **18**(11), p. e0287155.

[75] Ingale, A. K., and Bormane, D. S., 2021. "Real-time 3D reconstruction techniques applied in dynamic scenes: A systematic literature review". *Signal Processing: Image Communication,* **96**, p. 116310.

[76] Berger, M., Tagliasacchi, A., Seversky, L. M., Alliez, P., Guennebaud, G., Levine, J. A., Sharf, A., and Silva, C. T., 2017. "A survey of surface reconstruction from point clouds". *Computer Graphics Forum,* **36**(1), pp. 301–329.

[77] Huang, Z., Wen, Y., Wang, Z., Ren, J., and Jia, K., 2024. "Surface reconstruction from point clouds: A survey and a benchmark". *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[78] Geuzaine, C., and Remacle, J.-F., 2009. "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities". *International Journal for Numerical Methods in Engineering,* **79**(11), pp. 1309–1331.

[79] Si, H., 2015. "TetGen, a delaunay-based quality tetrahedral mesh generator". *ACM Transactions on Mathematical Software,* **41**(2), Feb.

[80] Berzeri, M., and Shabana, A., 2000. "Development of simple models for the elastic forces in the absolute nodal coordinate formulation". *J. of Sound and Vibration,* **235**(4), pp. 539–565.

[81] García-Vallejo, D., Mikkola, A. M., and Escalona, J. L., 2007. "A new locking-free shear deformable finite element based on absolute nodal coordinates". *Nonlinear Dynamics,* **50**, pp. 249–264.

[82] Gerstmayr, J., Sugiyama, H., and Mikkola, A., 2013. "Review on the absolute nodal coordinate formulation for large deformation analysis of multibody systems". *Journal of Computational and Nonlinear Dynamics,* **8**(3), p. 031016.

[83] Otsuka, K., Makihara, K., and Sugiyama, H., 2022. "Recent advances in the Absolute Nodal Coordinate Formulation: Literature review from 2012 to 2020". *Journal of Computational and Nonlinear Dynamics,* **17**(8), p. 080803.

[84] Shabana, A. A., 2023. "An overview of the ANCF approach, justifications for its use, implementation issues, and future research directions". *Multibody System Dynamics*.

[85] Gerstmayr, J., and Shabana, A. A., 2005. "Efficient integration of the elastic forces and thin three-dimensional beam elements in the absolute nodal coordinate formulation". In Proceeding of Multibody Dynamics ECCOMAS Thematic Conference.

[86] Bonet, J., et al., 2022. "Limitations of the st. venant–kirchhoff material model in large strain applications". *International Journal of Solids and Structures,* **248**, dec, p. 111618.

[87] Mooney, M., 1940. "A theory of large elastic deformation". *Journal of Applied Physics,* **11**(9), pp. 582–592.

[88] Rivlin, R. S., 1948. "Large elastic deformations of isotropic materials. I. Fundamental concepts". *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences,* **240**(822), pp. 459–490.

[89] Luo, Y., and Peng, B., 2020. "Benchmark problems of hyper-elasticity analysis in evaluation of fem". *Applied Sciences,* **10**(4), p. 1240.

[90] Simo, J. C., 1987. "On a fully three-dimensional finite-strain viscoelastic damage model: Formulation and computational aspects". *Computer Methods in Applied Mechanics and Engineering,* **60**(2), pp. 153–173.

[91] Reese, S., and Govindjee, S., 1998. "A theory of finite viscoelasticity and numerical aspects". *International Journal of Solids and Structures,* **35**(26–27), pp. 3455–3482.

[92] Holzapfel, G. A., 1996. "On large strain viscoelasticity: Continuum formulation and finite element applications to elastomeric structures". *International Journal for Numerical Methods in Engineering,* **39**(22), pp. 3903–3926.

[93] Holzapfel, G. A., and Simo, J. C., 1996. "A new viscoelastic constitutive model for continuous media at finite thermomechanical changes". *International Journal of Solids and Structures,* **33**(20–22), pp. 3019–3034.

[94] Treutenaere, S., Lauro, F., Bennani, B., Matsumoto, T., and Mottola, E., 2015. "Finite strain formulation of viscoelastic damage model for simulation of fabric reinforced polymers under dynamic loading". In EPJ Web of Conferences, Vol. 94, p. 04011.

[95] Drake Developers, 2025. Hydroelastic contact user guide. `https://drake.mit.edu/doxygen_cxx/group__hydroelastic__user__guide.html`. Accessed: 2025-12-14.

[96] Elandt, R., Drumwright, E., and Srinivasa, S. S., 2019. "A pressure field model for fast, robust approximation of net contact force and moment between nominally rigid objects". In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 8238–8245.

[97] Masterjohn, J., Guoy, D., Shepherd, J., and Castro, A., 2022. "Velocity level approximation of pressure field contact patches". *IEEE Robotics and Automation Letters,* **7**(4), pp. 11593–11600.

[98] Lauterbach, C., Garland, M., Sengupta, S., Luebke, D., and Manocha, D., 2009. "Fast BVH construction on GPUs". *Computer Graphics Forum,* **28**(2), pp. 375–384.

[99] Lauterbach, C., Mo, Q., and Manocha, D., 2010. "gproximity: Hierarchical GPU-based operations for collision and distance queries". *Computer Graphics Forum,* **29**(2), pp. 419–428.

[100] Tang, M., Curtis, S., Yoon, S., and Manocha, D., 2011. "Collision-streams: Fast GPU-based collision detection for deformable models". In Proceedings of the 2011 Symposium on Interactive 3D Graphics and Games (I3D '11), ACM, pp. 63–70.

[101] Chitalu, F. M., Dubach, C., and Komura, T., 2018. "Bulk-synchronous parallel simultaneous BVH traversal for collision detection on GPUs". In Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '18), ACM, pp. Article 4, 1–9.

[102] Gilbert, E., Johnson, D., and Keerthi, S., 1988. "A fast procedure for computing the distance between complex objects in three-dimensional space". *IEEE Journal on Robotics and Automation,* **4**(2), pp. 193–203.

[103] Snethen, G., 2008. "XenoCollide: Complex collision made simple". In *Game Programming Gems 7*, S. Jacobs, ed. Charles River Media, Hingham, MA, pp. 165–178.

[104] Pan, J., and Manocha, D., 2012. "GPU-based parallel collision detection for fast motion planning". *The International Journal of Robotics Research,* **31**(2), pp. 187–200.

[105] Hubbard, P. M., 1996. "Approximating polyhedra with spheres for time-critical collision detection". *ACM Transactions on Graphics,* **15**(3), pp. 179–210.

[106] Bradshaw, G., and O'Sullivan, C., 2004. "Adaptive medial-axis approximation for sphere-tree construction". *ACM Transactions on Graphics,* **23**(1), pp. 1–26.

[107] Zhang, R., Tagliafierro, B., Vanden Heuvel, C., Sabarwal, S., Bakke, L., Yue, Y., Wei, X., Serban, R., and Negrut, D., 2024. "Chrono DEM-Engine: A Discrete Element Method dual-GPU simulator with customizable contact forces and element shape". *Computer Physics Communications,* **300**, p. 109196.

[108] Fleischmann, J., Serban, R., Negrut, D., and Jayakumar, P., 2016. "On the importance of displacement history in soft-body contact models". *Journal of Computational and Nonlinear Dynamics,* **11**(4), p. 044502.

[109] Johnson, K. L., 1987. *Contact Mechanics*. Cambridge University Press, Cambridge, UK.

[110] Essoufi, B., Koko, J., and Zafrar, A., 2017. "Alternating direction method of multiplier for a unilateral contact problem in electro-elastostatics". *Computers & Mathematics with Applications*.

[111] Essoufi, B., and Zafrar, A., 2022. "J-admm for a multi-contact problem in electro-elastostatics". *ZAMM*.

[112] Chorfi, A., and Koko, J., 2020. "Alternating direction method of multiplier for the unilateral (frictionless) contact problem with an automatic penalty parameter selection". *Applied Mathematical Modelling*.

[113] Burman, E., Hansbo, P., and Larson, M. G., 2019. "Augmented lagrangian finite element methods for contact problems". *ESAIM: M2AN*.

[114] Burman, E., Hansbo, P., and Larson, M. G., 2025. "Hybridized augmented lagrangian methods for contact problems". *Computer Methods in Applied Mechanics and Engineering*.

[115] Tasora, A., Mazhar, I., and Negrut, D., 2021. "Solving variational inequalities and cone complementarity problems in nonsmooth dynamics using the admm". *International Journal for Numerical Methods in Engineering*.

[116] Zhu, S., Fang, C., Yu, P., Zhai, X., Hao, A., and Pan, J., 2024. "Efficient frictional contacts for soft body dynamics via admm". *The Visual Computer*.

[117] Nishioka, A., and Kanno, Y., 2023. "Inertial projected gradient method for large-scale topology optimization". *Japan Journal of Industrial and Applied Mathematics*.

[118] Oka, T., Misawa, R., and Yamada, T., 2023. "Nesterov's acceleration for level set-based topology optimization using reaction–diffusion equations". *Applied Mathematical Modelling*.

[119] Mazhar, H., Heyn, T., Negrut, D., and Tasora, A., 2015. "Using nesterov's method to accelerate multibody dynamics with friction and contact". *ACM Transactions on Graphics*.

[120] Hien, L. T. K., and Nam, N. M., 2022. An inertial admm for a class of nonconvex optimization problems. arXiv:2212.11336.

[121] Yuhn, C., Sato, Y., Kobayashi, H., Kawamoto, A., and Nomura, T., 2023. "4d topology optimization: Integrated optimization of the structure and self-actuation of soft bodies for dynamic motions". *Computer Methods in Applied Mechanics and Engineering*.

[122] Saad, Y., 2003. *Iterative methods for sparse linear systems*, 2 ed. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

[123] Davis, T. A., 2006. *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

[124] Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F., 1997. "Efficient management of parallelism in object oriented numerical software libraries". In Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser Press, pp. 163–202.

[125] Heroux, M. A., Bartlett, R. A., Howle, V. E., Hoekstra, R. J., Hu, J. J., Kolda, T. G., Lehoucq, R. B., Long, K. R., Pawlowski, R. P., Phipps, E. T., Salinger, A. G., Thornquist, H. K., Tuminaro, R. S., Willenbring, J. M., Williams, A., and Stanley, K. S., 2005. "An overview of the Trilinos project". *ACM Transactions on Mathematical Software,* **31**(3), pp. 397–423.

[126] Anderson, R., Andrej, J., Barker, A., Bramwell, J., Camier, J.-S., Cerveny, J., Dobrev, V., Dudouit, Y., Fisher, A., Kolev, T., Pazner, W., Stowell, M., Tomov, V., Akkerman, I., Dahm, J., Medina, D., and Zampini, S., 2021. "MFEM: A modular finite element methods library". *Computers & Mathematics with Applications,* **81**, pp. 42–74.

[127] Merrill, D., 2015. CUB: CUDA UnBound, a library of warp-wide, block-wide, and device-wide gpu parallel primitives. Available online at `http://nvlabs.github.io/cub/`.

[128] Zhou, Z., Arivoli, G., and Negrut, D., 2026. A Total Lagrangian Finite Element Framework for Multibody Dynamics: Part I – Formulation.

[129] Loshchilov, I., and Hutter, F., 2019. "Decoupled weight decay regularization". In International Conference on Learning Representations (ICLR).

[130] Project Chrono, 2020. Chrono: An open source framework for the physics-based simulation of dynamic systems. `http://projectchrono.org`. Accessed: 2020-03-03.

[131] Ansys, Inc., 2024. Academic usage terms and conditions. `https://www.ansys.com/legal/terms-and-conditions/academic-usage`. Accessed: 2026.

[132] Recuero, A. M., and Negrut, D., 2016. Chrono support for ANCF finite elements: Formulation and validation aspects. Tech. Rep. TR-2016-11, Simulation-Based Engineering Laboratory, University of Wisconsin-Madison.

[133] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Rupp, K., Smith, B. F., and Zhang, H., 2014. PETSc users manual. Tech. Rep. ANL-95/11 - Revision 3.5, Argonne National Laboratory.

[134] Williams, S., Waterman, A., and Patterson, D., 2009. "Roofline: an insightful visual performance model for multicore architectures". *Communications of the ACM,* **52**(4), pp. 65–76.

[135] Wu, C.-Y., Thornton, C., and Li, L.-Y., 2003. "Coefficients of restitution for elastoplastic oblique impacts". *Advanced Powder Technology,* **14**(4), pp. 435–448.

[136] Yu, K., Elghannay, H. A., and Tafti, D., 2017. "An impulse based model for spherical particle collisions with sliding and rolling". *Powder Technology,* **319**, pp. 102–116.

[137] National Park Service, 1993. Packing museum objects for shipment. Conserve O Gram 17/2, July 1993. U.S. Department of the Interior, accessed 2026-03-16.

[138] Húlan, T., Štubňa, I., Ondruška, J., Csáki, Š., Lukáč, F., Mánik, M., Vozár, L., Ozolins, J., Kaljuvee, T., and Trník, A., 2020. "Young's modulus of different illitic clays during heating and cooling stage of firing". *Materials,* **13**(21), p. 4968.

[139] Kojima, S., 2024. "Poisson's ratio of glasses, ceramics, and crystals". *Materials,* **17**(2), p. 300.

[140] Dassault Systèmes Simulia Corp., 2024. *Hyperelastic Behavior of Rubberlike Materials.* Abaqus 2024 Documentation, accessed 2026-03-16.

[141] Han, D., and Che, W., 2021. "Comparison of the shear modulus of an offshore elastomeric bearing between numerical simulation and experiment". *Applied Sciences,* **11**(10), p. 4384.

[142] Mamashli, H., Gerdooei, M., and Ghafourian Nosrati, H., 2026. "Parametric study of piercing force and surface quality in elastomer-assisted tube piercing". *Scientific Reports,* **16**, p. 3612. Published online 2025-12-24; version of record 2026-01-27.

[143] Chen, H., Sun, D., Gao, L., Liu, X., and Zhang, M., 2024. "Mechanical behavior of closed-cell ethylene-vinyl acetate foam under compression". *Polymers,* **16**(1), p. 34.

[144] The Rubber Company, n.d.. 50 shore neoprene rubber sheeting datasheet. Datasheet. Density = 1.35 g/cm^3 (1350 kg/m^3).

[145] Delta Rubber, n.d.. Neoprene rubber sheets / bs2752. 60 Shore datasheet. Density = 1.40 g/cm^3 (1400 kg/m^3).

[146] Le Guen, V., and Thome, N., 2020. "Disentangling physical dynamics from unknown factors for unsupervised video prediction". In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11474–11484.

[147] Ben Zikri, N., and Sharf, A., 2022. "PhyLoNet: Physically-constrained long term video prediction". In Proceedings of the Asian Conference on Computer Vision (ACCV).

[148] Liu, S., Ren, Z., Gupta, S., and Wang, S., 2024. "Physgen: Rigid-body physics-grounded image-to-video generation". In European Conference on Computer Vision (ECCV). OpenAccess version.

[149] Zhang, K., Xiao, C., Xu, J., Mei, Y., and Patel, V. M., 2025. Think before you diffuse: Infusing physical rules into video diffusion.

[150] Zhang, X., Liao, J., Zhang, S., Meng, F., Wan, X., Yan, J., and Cheng, Y., 2025. "VideoREPA: Learning physics for video generation through relational alignment with foundation models". In Advances in Neural Information Processing Systems (NeurIPS).

[151] Hao, Y., Chen, C., Mian, A. S., Xu, C., and Liu, D., 2025. Enhancing physical plausibility in video generation by reasoning the implausibility.

[152] Jang, S., Ki, T., Jo, J., Yoon, J., Kim, S. Y., Lin, Z., and Hwang, S. J., 2025. Frame guidance: Training-free guidance for frame-level control in video diffusion models, June.

[153] Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M., 2018. "Deepmind control suite". *arXiv preprint arXiv:1801.00690*.

[154] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M., 2013. "The arcade learning environment: An evaluation platform for general agents". *Journal of Artificial Intelligence Research,* **47**, pp. 253–279.

[155] Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S., 2020. "D4RL: Datasets for deep data-driven reinforcement learning". *arXiv preprint arXiv:2004.07219*.

[156] James, S., Ma, Z., Arrojo, D. R., and Davison, A. J., 2020. "RLBench: The robot learning benchmark & learning environment". *IEEE Robotics and Automation Letters,* **5**(2), pp. 3019–3026.

[157] Open X-Embodiment Collaboration, 2023. "Open x-embodiment: Robotic learning datasets and RT-X models". *arXiv preprint arXiv:2310.08864*.

[158] Wang, Y., et al., 2024. "DrivingDojo dataset: Advancing interactive and knowledge-enriched driving world models". *arXiv preprint arXiv:2410.10738*.

[159] Chen, D., Chung, W., Bang, Y., Ji, Z., and Fung, P., 2025. "Worldprediction: A benchmark for high-level world modeling and long-horizon procedural planning". *arXiv preprint arXiv:2506.04363*.

[160] Zhou, J., 2025. Dreamerbench: Dataset for evaluating world models under frictional, contact-rich dynamics. `https://huggingface.co/datasets/zzhou292/DreamerBench`. Accessed: 2025-11-27.

[161] Agarwal, N., Ali, A., Bala, M., Balaji, Y., Barker, E., et al., 2025. "Cosmos world foundation model platform for physical AI". *arXiv preprint arXiv:2501.03575*.

[162] Uhlenbeck, G. E., and Ornstein, L. S., 1930. "On the theory of the brownian motion". *Physical Review,* **36**(5), pp. 823–841.

[163] Gillespie, D. T., 1996. "Exact numerical simulation of the ornstein–uhlenbeck process and its integral". *Physical Review E,* **54**(2), pp. 2084–2091.

[164] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., 2015. "Continuous control with deep reinforcement learning". *arXiv preprint arXiv:1509.02971*.

[165] Hollenstein, J., Auddy, S., Saveriano, M., Renaudo, E., and Piater, J., 2022. "Action noise in off-policy deep reinforcement learning: Impact on exploration and performance". *Transactions on Machine Learning Research*. Available at arXiv:2206.03787.

[166] Higuera, C., Dong, S., Boots, B., and Mukadam, M., 2022. "Neural contact fields: Tracking extrinsic contact with tactile sensing". *arXiv preprint arXiv:2210.09297*.

[167] Kim, L., Li, Y., Posa, M., and Jayaraman, D., 2023. "Im2contact: Vision-based contact localization without touch or force sensing". In Proceedings of The 7th Conference on Robot Learning, Vol. 229 of *Proceedings of Machine Learning Research*, PMLR, pp. 1533–1546.

[168] Wu, L., Yu, C., Ren, J., Chen, L., Huang, R., Gu, G., and Li, H., 2025. "Freetacman: Robot-free visuo-tactile data collection system for contact-rich manipulation". *arXiv preprint arXiv:2506.01941*.

[169] Cong, X., Xing, A., Pokhariya, C., Fu, R., and Sridhar, S., 2025. "Dytact: Capturing dynamic contacts in hand-object manipulation". *arXiv preprint arXiv:2506.03103*.

[170] Mentzer, F., Minnen, D., Agustsson, E., and Tschannen, M., 2024. "Finite scalar quantization: VQ-VAE made simple". In International Conference on Learning Representations (ICLR).

[171] Chang, H., Zhang, H., Jiang, L., Liu, C., and Freeman, W. T., 2022. "MaskGIT: Masked generative image transformer". In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

[172] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O., 2018. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In Proceedings

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 586–595.

[173] Unterthiner, T., van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., and Gelly, S., 2019. "Towards accurate generative models of video: A new metric & challenges". *arXiv preprint arXiv:1812.01717*.

[174] Alonso, E., Jelley, A., Micheli, V., Kanervisto, A., Storkey, A., Pearce, T., and Fleuret, F., 2024. "Diffusion for world modeling: Visual details matter in Atari". In Advances in Neural Information Processing Systems (NeurIPS).

[175] Valevski, D., Leviathan, Y., Arar, M., and Fruchter, S., 2024. "Diffusion models are real-time game engines". *arXiv preprint arXiv:2408.14837*.

[176] Brooks, T., Peebles, B., Holmes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C., Wang, R., and Ramesh, A., 2024. Video generation models as world simulators.

[177] Huang, Z., He, Y., Yu, J., Zhang, F., Si, C., Jiang, Y., Zhang, Y., Wu, T., Jin, Q., Chanpaisit, N., Wang, Y., Chen, X., Wang, L., Lin, D., Qiao, Y., and Liu, Z., 2024. "VBench: Comprehensive benchmark suite for video generative models". In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

[178] Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I., 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

[179] Kim, S., Shin, J., Cho, Y., Jang, J., Longpre, S., Lee, H., Yun, S., Shin, S., Kim, S., Thorne, J., and Seo, M., 2024. "Prometheus: Inducing fine-grained evaluation capability in language models". In International Conference on Learning Representations (ICLR).

[180] Kim, S., Suk, J., Longpre, S., Lin, B. Y., Shin, J., Welleck, S., Neubig, G., Lee, M., Lee, K., and Seo, M., 2024. "Prometheus 2: An open source language model specialized

in evaluating other language models". In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).

[181] Chen, Y., Zhu, X., and Li, T., 2025. "A physical coherence benchmark for evaluating video generation models via optical flow-guided frame prediction". *arXiv preprint arXiv:2502.05503*.

[182] Bansal, H., Lin, Z., Xie, T., Zong, Z., Yarom, M., Bitton, Y., Jiang, C., Sun, Y., Chang, K.-W., and Grover, A., 2024. "VideoPhy: Evaluating physical commonsense for video generation". *arXiv preprint arXiv:2406.03520*.

[183] Hanley, J. A., and McNeil, B. J., 1982. "The meaning and use of the area under a receiver operating characteristic (ROC) curve". *Radiology,* **143**(1), pp. 29–36.

[184] Wang, P., Li, L., Chen, L., Cai, Z., Zhu, D., Lin, B., Cao, Y., Liu, Q., Liu, T., and Sui, Z., 2023. "Large language models are not fair evaluators". *arXiv preprint arXiv:2305.17926*.

[185] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J.,

Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X. E., Xia, X., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence,

B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Liu, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N. P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S.,

Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao, Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z., and Ma, Z., 2024. The llama 3 herd of models.

[186] Gemma Team, 2025. Gemma 3 technical report. Tech. rep., Google DeepMind.

[187] Qwen Team, 2026. Qwen3.5: Towards native multimodal agents, February.

[188] Shabana, A. A., 1998. "Computer implementation of the Absolute Nodal Coordinate Formulation for flexible multibody dynamics". *Multibody System Dynamics,* **2**, pp. 307–332.

[189] Zhou, Z., and SBEL, University of Wisconsin-Madison, 2025. ChronoDreamer: World model with physical information for robotic planning task. `https://github.com/uwsbel/ChronoDreamer`. GitHub repository.