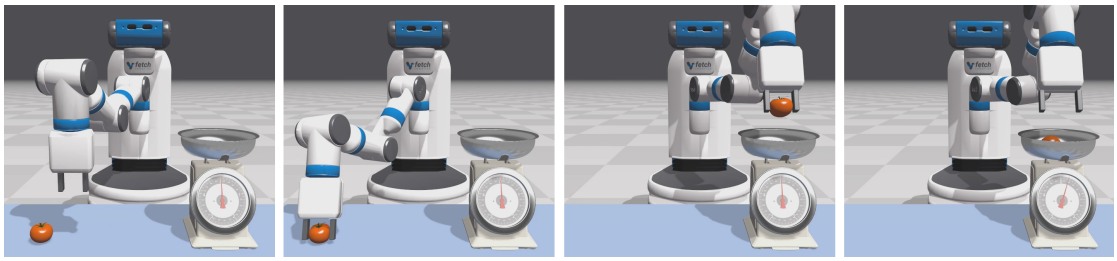
UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE



**PhD Thesis**

Miles Macklin

# Simulation for Learning and Robotics

## Numerical Methods for Contact, Deformation, and Identification

Advisor: Kenny Erleben

Handed in: September 29, 2020

# Abstract

Robots have the potential to improve the quality of our lives by automating and assisting us in our work. Unfortunately, the design, manufacture, and testing, of real-world robots and their controllers remains time consuming and expensive. Simulation provides a potential solution to this problem, as it can allow gathering data in a safe and relatively inexpensive manner. However, there are many challenges in designing simulation methods for robotics. Modern robots consist of both rigid and deformable parts that may interact in strongly coupled ways through contact, for example, in soft robotic manipulation. Thus, to broaden their applicability, simulators must be capable of modeling a wide range of materials. A second challenge in applying simulation is the large scale data required to drive machine learning techniques, for example, reinforcement learning-based controller design. To generate this data, simulators must be scalable, low-latency, and robust to unseen inputs over a wide range of parameters. Sensitivity to ill-conditioning, e.g.: high mass ratios, can cause fast numerical methods to fail to converge in a reasonable time. On the other hand, accurate solvers that can deal with ill-conditioning are often not fast enough to be used for online/interactive control. Designing numerical methods that overcome these limitations is important to expand simulation capabilities. Finally, a major challenge in simulation is crossing the *reality gap*. This refers to the ability of a simulator to provide outputs that are predictive of real-world behavior. Since physical models are approximations, and model parameters are often not known, crossing the reality gap is difficult, and methods that can help transfer simulations to the real world are desirable.

In the first part of this thesis, I present new simulation techniques focusing on contact modeling between rigid and deformable objects in a unified mathematical formulation. I then develop parallel numerical methods to solve the resulting nonlinear problems. The proposed methods have improved convergence rates to address the issue of sensitivity to ill-conditioning, while maintaining scalability and robustness. The second part of this thesis addresses the sim-to-real problem. I include work that performs classical system identification and validation, followed by stochastic, and differentiable simulation methods to perform automatic parameter estimation. The presented work bridges many aspects of computer graphics and robotics, and provides a framework for future studies to enable the design and control of the next generation of robots.

# Contents

# Acknowledgements

First and foremost I would like to thank my advisor Dr. Kenny Erleben who has been a constant source of encouragement, insight, and wisdom. I purchased Dr. Erleben's book on Physics-Based Animation ten years before starting this degree and never imagined I would have the opportunity to work together with him. I also give my deepest thanks to my co-supervisor Dr. Matthias Müller. Without his initial support I would not have pursued a career in research. His enthusiasm, and desire for simplicity has always been an inspiration. I owe a large debt to my colleagues at NVIDIA, particularly Nuttapong Chentanez, Stefan Jeschke, Viktor Makoviychuk, Renato Gasoto, Richard Tonge, Simon Schirm, Mike Skolones, Andrew Reidmeyer, and Hammad Mazhar. NVIDIA is a fantastic place for research due to the diversity of groups. I have been fortunate to collaborate closely with the members and interns at Seattle Robotics Lab, in particular Ankur Handa, Yevgen Chebotar, Jacky Liang, and Dieter Fox. The collaborations between graphics and robotics have opened my eyes to new areas and new problems, that continue to be some of the most exciting to me. I want to acknowledge Jos Stam for introducing me to the adjoint method which resulted in the differentiable simulation work in this thesis in collaboration with Krishna Murthy and Sanja Fidler. I thank Ignacio Llamas and the Omniverse team for their help preparing the renderings, and I'm particularly grateful for the freedom and mentorship given to me by Adam Moravanszky, Tae-Yong Kim, and Rev Lebaredian.

My external collaborators have been a great source of interesting discussions, thank you to Mihai Francu, Sheldon Andrews, and Paul Kry for including me in the world of contact. Many administrators at UCPH have accommodated me during the course of this degree, I especially wish to thank Anja Fløytrup, who has patiently guided me through the Ph.D. process.

To my parents, Janet and Trevor, thank you for their belief in me and for providing my first computer, which sparked a passion for computing. Finally, my biggest thanks go to my wife Rachael and my two sons, Lucas and Alfie. I would not have been able to complete this thesis without their tremendous love and support. They are a constant source of happiness, and I am incredibly grateful to them.

Table 1: Notation

| Symbol | Meaning |
|---|---|
| $\mathbf{q}$ | Generalized system coordinates |
| $\dot{\mathbf{q}}$ | First time derivative of generalized coordinates |
| $\ddot{\mathbf{q}}$ | Second time derivative of generalized coordinates |
| $\mathbf{f}$ | Generalized force function |
| $\mathbf{u}$ | Generalized velocity vector |
| $\tilde{\mathbf{u}}$ | Predicted or inertial system velocity |
| $\mathbf{M}$ | System mass matrix |
| $\mathbf{K}$ | Geometric stiffness matrix |
| $\mathbf{E}$ | Compliance matrix |
| $\mathbf{D}$ | Friction force basis |
| $\mathbf{W}$ | Friction compliance matrix |
| $\mathbf{G}$ | Kinematic mapping transform |
| $\Psi$ | Strain energy density function |
| $\mathbf{c}_b$ | Bilateral constraint vector |
| $\mathbf{c}_n$ | Normal contact constraint vector |
| $\mathbf{c}_f$ | Frictional contact constraint vector |
| $\mathbf{n}$ | Contact normal |
| $d$ | Separation distance for a contact |
| $\psi$ | Friction cone level set |
| $\boldsymbol{\phi}_n$ | Normal contact NCP constraint vector |
| $\boldsymbol{\phi}_f$ | Frictional contact NCP constraint vector |
| $\mathbf{J}_b$ | Jacobian of bilateral constraints |
| $\mathbf{J}_n$ | Jacobian of contact normal NCP constraints |
| $\mathbf{J}_f$ | Jacobian of contact friction NCP constraints |
| $\boldsymbol{\lambda}_b$ | Lagrange multiplier vector for bilateral constraints |
| $\boldsymbol{\lambda}_n$ | Lagrange multipliers for normal contact constraints |
| $\boldsymbol{\lambda}_f$ | Lagrange multipliers for friction contact constraints |
| $n_d$ | Number of system degrees of freedom |
| $n_b$ | Number of bilateral constraints |
| $n_c$ | Number of contacts |

# 1 Introduction

Enabling the next-generation of robots that can, for example, enter a kitchen and prepare dinner, requires new control algorithms that are capable of navigating complex environments and performing dexterous manipulation of real-world objects. Modern machine-learning techniques hold the promise of unlocking this capability, but they require large amounts of data to work effectively [8, 116]. Although there have been some attempts at learning through large-scale data collected directly on real robots [168, 233, 106], this approach remains challenging, as it is laborious and often prohibitively expensive. In contrast, simulation is relatively inexpensive, safe, and has been used to learn and transfer behaviors such as walking and jumping to real robots [204, 179]. However, transfer learning is also challenging. Due to incomplete simulation models, and imprecise parameters, the policies learned in simulation often cannot be directly applied on real world systems, a phenomenon also known as the *reality gap* [100]. In this work, our objective is to develop simulation methods that enable engineers and scientists to create new capabilities for robots. To do this we focus on the three core areas: modeling, robust and efficient simulation, and transfer to the real-world.

**Modeling** Extending transfer learning to new scenarios requires the simulation of rich environments that incorporate multiple physical models. For example, although dexterous manipulation is often treated as a rigid-body problem [80, 139], many real-world use cases require interaction with deformable objects, such as picking up a soft tomato, cracking an egg, or squeezing a ball, as illustrated in Figure 1. In the case of soft robotics, the manipulator itself may be deformable, as illustrated by the PneuNet gripper, shown in Figure 2. This design incorporates a flexible elastic finger, an inextensible paper layer, and a chamber that is pressurized to cause the finger to curve [97]. These types of multiphysics scenarios are challenging for simulation because the internal dynamics may be complex, and they must interact strongly through contact to allow for robust grasping. Contact is particularly challenging to model accurately. Surfaces may contain a grain pattern, or a fiber microstructure that influences their behavior, yet existing contact models, such as Coulomb friction, make simplifications such as assuming a uniformly rough surface [198]. In this work, we develop a framework that allows for the coupled simulation of both rigid and deformable bodies in a unified manner. We address the limitations of existing contact models by proposing a new friction model that can represent arbitrary combinations of anisotropic surfaces.

**Robustness and Efficiency** To broaden the range of scenarios robots can operate, not only must simulators provide rich modeling capability, they must be efficient. Control optimization

techniques such as reinforcement learning require many training examples, and data generation is often the bottleneck [8]. In addition, some robotics scenarios involve haptic feedback, imitation learning, or human-assisted control that requires real-time, low-latency response [127]. To meet these performance constraints, existing real-time simulation methods often make sacrifices in robustness or accuracy. For example, iterative methods such as Gauss-Seidel have low overhead and scale to large problems, but are less robust to ill-conditioned problems, such as when large mass-ratios are present [57]. Robustness becomes increasingly important with the rising popularity of stochastic optimization methods used in machine learning [200]. In many cases these methods generate essentially random inputs to the simulator. For example, noisy control signals in the case of reinforcement learning [201], or diverse physical parameters in the case of domain randomization [210]. To improve robustness, we focus on developing numerical methods that are less sensitive to ill-conditioning, but retain the ability to scale with low-latency. We predict that modern parallel architectures like the graphics processing units (GPUs) will become increasingly important to achieve these goals, and we focus our efforts on methods that are able to exploit large-scale parallelism.

**Skill Transfer** Simulators that provide physical plausibility may be useful for visual effects, however, to be useful for robotics they must have some predictive power in the real world. Closing the reality gap remains a challenging problem, as evidenced by the large number of recent publications and conference workshops on the topic [204, 237, 86]. A core issue for simulation transfer is that the physical parameters, e.g.: mass, geometry, and stiffness of an object are often only known approximately. Parameter estimation, or system identification, is the process of determining these model parameters, typically through a measurement, fitting, and validation loop [189]. This process can be time-consuming, and so we propose a data-driven method that uses stochastic optimization to adapt model parameters based on a distribution of parallel simulations. Furthermore, given the widespread availability of digital cameras we propose a system that uses *differentiable* simulation and rendering to perform system identification directly from images and video.

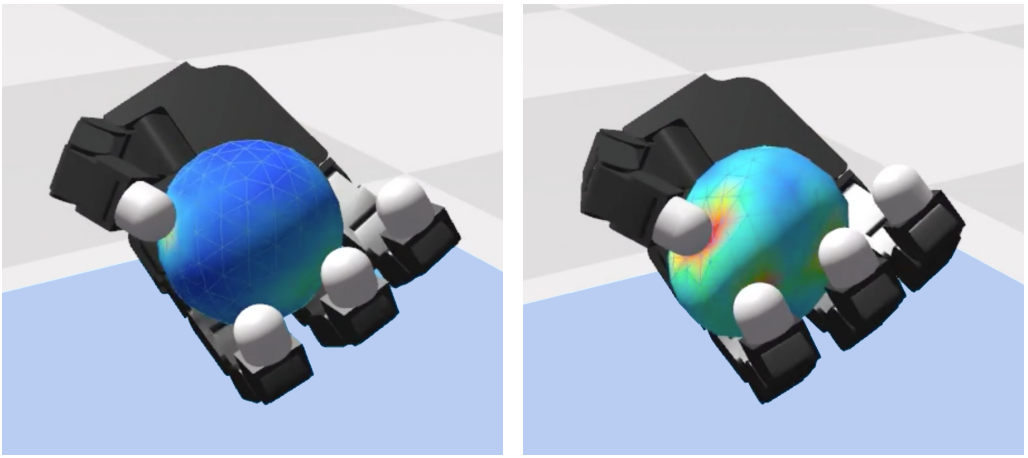


Figure 1: **Deformable Manipulation**: The Allegro hand squeezing a ball. Our framework supports full coupling between the articulated fingers and the ball's internal dynamics. The color field visualizes volumetric strain. Model provided courtesy of SimLab Co., Ltd.

## 1.1 Organization

Conceptually this thesis is divided into two main parts. The first part addresses the modeling, and numerical methods required for efficient and robust multiphysics simulation.

- Chapter 2 discusses mathematical background on physical simulation, and introduces the basics of frictional contact modeling.
- Chapter 3 considers how to model contact between surfaces with anisotropic structure. We present a novel extension of isotropic Coulomb friction model that takes into account the relative structure directions of two surfaces in a way that respects the principle of maximum dissipation (PMD).
- In Chapter 4, we ask how we can leverage the parallel nature of GPUs to solve multiphysics contact problems. We hypothesize that, since the main computational operation in Newton methods is the solution of a sparse linear system, these methods may be able to take advantage of fast linear solvers already developed for the GPU architectures. We show that, with a novel complementarity preconditioner, this approach leads to better handling of ill-conditioned problems than iterative relaxation-based methods. In addition, by treating deformable and rigid bodies in a unified manner, our framework allows for fully coupled simulation of multiphysics scenarios, which we demonstrate on soft robotic grasping tasks.
- Chapter 5 considers the relationship between two variational formulations of implicit time-integration. The question we attempt to answer is the degree to which methods such as Projective Dynamics and Position-based Dynamics differ in terms of their sensitivity to ill-conditioning. While constraint-based (dual) methods are commonly used for contact simulation, we hypothesize that primal optimization methods offer complementary trade-offs. We develop a common mathematical basis for both methods that provides new insight into their relationship, and find that while dual methods are sensitive to mass-ratios, primal methods are not. In addition, we introduce a differentiable contact model that forms the basis for our parameter optimization work in later chapters.
- In Chapter 6, we consider how to improve the robustness of collision detection. We observe that point-sampled collision detection is often insufficient to capture the sharp features of shapes, and present a method based on local optimization over mesh features, that can more accurately prevent interpenetration between objects. We demonstrate the effectiveness and efficiency of this approach on real-time clothing and rigid body contact problems.

The second major topic of this work is the transfer of simulation results to the real-world. We leverage the simulation frameworks developed in the first part of this thesis to address the reality gap problem in robotics.

- In Chapter 7.1, we consider the simulation and modeling of a soft, pneumatically actuated robotic snake. We take a classical approach to system identification through measurement and validation. This work acts as a verification of the simulation work performed in preceding chapters, and pushes the state-of-the-art in real-time simulation of such complex constructions.
- In Chapter 7.2, we ask if it is possible to extend the reinforcement learning framework to not only optimize control policies, but to also perform automatic identification of simulation parameters, using minimal real-world observations. We present *SimOpt*, a stochastic optimization method, that learns and updates model parameters over parallel simulation distributions.
- In Chapter 7.3, we further explore the problem of parameter estimation and ask whether meaningful gradients can be propagated through an end-to-end differentiable simulation and rendering pipeline, from images back to model parameters. Combining the work performed in Chapter 5 with a differentiable rasterizer, we present $\nabla Sim$ (GradSim), a framework for inverse simulation that demonstrates rigid and deformable parameter estimation directly from rendered images and video.
- Chapter 8 concludes the thesis by reflecting on the work and suggesting directions for future research.

## 1.2 Contributions

To summarize, the primary contributions of the work in this thesis are:

- An extension of the Coulomb friction model that considers the interaction of multiple anisotropic surfaces through combinations of ellipsoidal friction cones.
- A non-smooth solver framework for contact problems. The presented work includes new nonlinear complementarity (NCP) preconditioners that allow solving rigid and deformable contact problems in a fully coupled way, while leveraging parallel architectures such as the GPU.
- A derivation and analysis of force-based and constraint-based formulations of contact dynamics. This leads to a unified view of two common simulation methods, Projective Dynamics and Position-based Dynamics, as *primal* and *dual* variational approaches respectively, and gives new insights into the sensitivities of both methods.
- An optimization-based approach to continuous surface contact generation with signed distance fields (SDFs) that overcomes the limitations of sampling-based contact approaches, improving robustness and efficiency.
- A real-time simulator and validated physical model for a soft robotic snake that can make predictive simulations of snake locomotion.
- A reinforcement learning approach that can adapt simulation parameters on the fly using stochastic optimization. This approach automates the typically time-consuming task of system identification.
- A simulation framework that combines differentiable physics and differentiable rendering to achieve state-of-the-art soft and rigid body parameter estimation directly from video.

Together, these technical and scientific innovations enable new applications in robotic modeling, real-time simulation, and control. The presented works have been integrated into industrial simulation platforms such as NVIDIA's Isaac Robotics SDK, and have been used as the basis for many follow-up research projects and publications.

## 1.3 Publications

Over the course of my PhD I was fortunate to have published works with many collaborators. For this thesis I have chosen to present seven papers that most closely relate to the topic of robotics. The first topic of this thesis is covered in Chapters [3-6] and includes material from the following papers on numerical methods for simulation:

- Erleben, K., Macklin, M., Andrews, S., & Kry, P. G. (2020, February). *The Matchstick Model for Anisotropic Friction Cones*. In Computer Graphics Forum (Vol. 39, No. 1, pp. 450-461). Ch. 3
- Macklin, M., Erleben, K., Müller, M., Chentanez, N., Jeschke, S., & Makoviychuk, V. (2019). *Non-smooth Newton Methods for Deformable Multi-body Dynamics*. ACM Transactions on Graphics (TOG), 38(5), 1-20. Ch. 4
- Macklin, M., Erleben, K., Müller, M., Chentanez, N., Jeschke, S, & Kim, T. Y. (2020, October). *Primal/Dual Descent Methods for Dynamics*. In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Ch. 5
- Macklin, M., Erleben, K., Müller, M., Chentanez, N., Jeschke, S., & Corse, Z. (2020). *Local Optimization for Robust Signed Distance Field Collision*. Proceedings of the ACM on Computer Graphics and Interactive Techniques, 3(1), 1-17. Ch. 6

In Chapter 7 I focus on papers that apply the developed methods to specific problems in robotics and machine learning:

- Gasoto, R., Macklin, M., Liu, X., Sun, Y., Erleben, K., Onal, C., & Fu, J. (2019, May). *A Validated Physical Model for Real-Time Simulation of Soft Robotic Snakes*. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 6272-6279). IEEE. Ch. 7.1 [1]
- Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., & Fox, D. (2019, May). *Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience*. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 8973-8979). IEEE. Ch 7.2
- Murthy, K. Macklin, M. Golemo, F. Voleti, V. Petrini, L. Weiss, M. Considine, B. Parent-Lévesque, J. Xie, K. Erleben, K. Paull, L. Shkurti, F. Fidler, S. Nowrouzezahrai, D. *$\nabla$Sim: Differentiable Physics and Rendering Engines for Parameter Estimation from Video* Neural Information Processing Systems (NeurIPS), 2020 (submitted) Ch 7.3[1]

[1] Joint first author

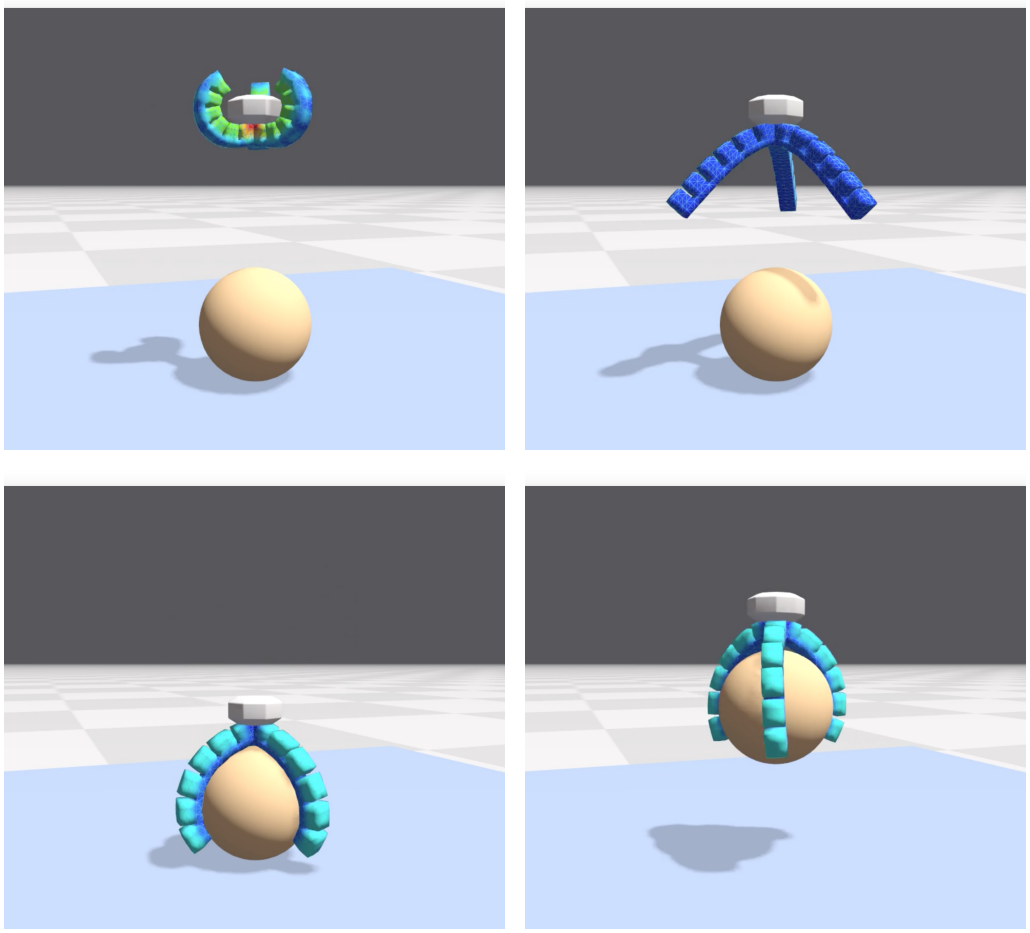


Figure 2: **Soft Robotic Grasping**: A simulation of a soft-robotic grasping mechanism based on the pneumatic networks (PneuNet) design. The three fingers are modeled using tetrahedral FEM, while the sphere is a rigid body. When the chambers on the back of the fingers are pressurized they cause the inextensible lower layer to curve. Our method provides robust contact coupling between the two dynamics models. The element color visualizes the volumetric strain field.

# 2 Background

In this section we give some mathematical background on physical modeling. At the most general level, the dynamics of the systems we wish to simulate can be described through the following second order differential equation:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} - \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{0}, \tag{1}$$

where for a system with $n_d$ degrees of freedom, $\mathbf{M} \in \mathbb{R}^{n_d \times n_d}$ is the system mass matrix, $\mathbf{f} \in \mathbb{R}^{n_d}$ is a generalized force vector, and $\mathbf{q} \in \mathbb{R}^{n_d}$ is a vector of generalized coordinates describing the system configuration with $\dot{\mathbf{q}}, \ddot{\mathbf{q}}$ the first and second time derivatives respectively. We refer the reader to Table 1 for a list of symbols used in this paper.

## 2.1 Physical Modeling

The continuous equations of motion (1) are expressed purely in terms of our generalized coordinates $\mathbf{q}$ and their derivatives. Although it is possible to discretize and solve these equations for $\mathbf{q}$ directly, when dealing with rigid bodies it is convenient to introduce an additional reparameterization in terms of a generalized velocity vector $\mathbf{u}$:

$$\dot{\mathbf{q}} = \mathbf{G}(\mathbf{q})\mathbf{u} \tag{2}$$

$$\ddot{\mathbf{q}} = \dot{\mathbf{G}}(\mathbf{q})\mathbf{u} + \mathbf{G}(\mathbf{q})\dot{\mathbf{u}}. \tag{3}$$

Here $\mathbf{G}$ is referred to as the *kinematic map*, and its components depend on the choice of system coordinates. For simple particles $\mathbf{G}$ is an identity transform, however in Section 2.1.2 we discuss the mapping of a rigid body's angular velocity to the corresponding quaternion time-derivative.

### 2.1.1 Particles

We represent soft bodies, such as cloth or deformable solids, as sets of particles where a particle with index $i$ contributes 3 degrees of freedom,

$$\mathbf{q}_i = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \mathbf{u}_i = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}. \tag{4}$$

The associated mass matrix is $\mathbf{M}_i = m\mathbf{1}_{3x3}$, where $m$ is the particle mass.

### 2.1.2 Rigid Bodies

To describe the state of a rigid body with index $i$ we use a maximal coordinate representation consisting of the position of the body's center of mass, $\mathbf{x}_i$, its orientation expressed as a quaternion $\boldsymbol{\theta}_i = [\theta_1, \theta_2, \theta_3, \theta_4]$, and a generalized velocity vector $\mathbf{u}_i$

$$\mathbf{q}_i = \begin{bmatrix} \mathbf{x}_i \\ \boldsymbol{\theta}_i \end{bmatrix}, \quad \mathbf{u}_i = \begin{bmatrix} \dot{\mathbf{x}}_i \\ \boldsymbol{\omega}_i \end{bmatrix}, \tag{5}$$

where $\boldsymbol{\omega}$ is the body's angular velocity. The *kinematic map* from generalized velocities to the system coordinate time-derivatives is then given by $\dot{\mathbf{q}}_i = \mathbf{G}\mathbf{u}_i$ where

$$\mathbf{G} = \begin{bmatrix} \mathbf{1}_{3\times 3} & \mathbf{0} \\ \mathbf{0} & \frac{1}{2}\mathbf{Q} \end{bmatrix}, \tag{6}$$

and $\mathbf{Q}$ is the matrix that performs a truncated quaternion rotation,

$$\mathbf{Q}(\boldsymbol{\theta}) = \begin{bmatrix} -\theta_2 & -\theta_3 & -\theta_4 \\ \theta_1 & \theta_4 & -\theta_3 \\ -\theta_4 & \theta_1 & \theta_2 \\ \theta_3 & -\theta_2 & \theta_1 \end{bmatrix}. \tag{7}$$

The corresponding generalized mass matrix is then

$$\mathbf{M} = \mathbf{G}^T\tilde{\mathbf{M}}\mathbf{G}, \tag{8}$$

where

$$\tilde{\mathbf{M}}_i = \begin{bmatrix} m\mathbf{1}_{3\times 3} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{9}$$

and $m$ is the body mass and $\mathbf{I}$ is the inertia tensor. When using quaternions to represent rotations there is an implicit constraint that $\|\boldsymbol{\theta}\| = 1$. Rather than include this constraint explicitly it is common to periodically renormalize the quaternion to unit-length [15].

### 2.1.3 Hard Kinematic Constraints

We impose hard bilateral (equality) constraints on the system configuration through constraint functions of the form $C_b(\mathbf{q}) = 0$. Using D'Alembert's principle [113] the force due to such a constraint is of the form $\mathbf{f}_b = \nabla C_b(\mathbf{q})^T \lambda_b$ where $\nabla C_b$ is the constraint's gradient with respect to the system coordinates, and $\lambda_b$ is a Lagrange multiplier variable used to enforce the constraint. Combining these algebraic constraints with the differential equation (1) gives the following Differential Algebraic Equation (DAE):

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} - \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) - \nabla \mathbf{c}_b^T \boldsymbol{\lambda}_b = \mathbf{0} \tag{10}$$

$$\mathbf{c}_b(\mathbf{q}) = \mathbf{0}. \tag{11}$$

For a system with $n_b$ equality constraints we define the constraint vector $\mathbf{c}_b(\mathbf{q}) = [C_{b,1}, C_{b,2}, \cdots, C_{b,n_b}]^T \in \mathbb{R}^{n_b}$, $\nabla \mathbf{c}_b \in \mathbb{R}^{n_b \times n_d}$ its gradient with respect to system coordinates, and $\boldsymbol{\lambda}_b \in \mathbb{R}^{n_b}$ the vector of Lagrange multipliers. In general bilateral constraints may depend on velocity and time, however we assume scleronomic constraints for the remainder of the paper to simplify the exposition.

### 2.1.4 Soft Kinematic Constraints

In addition to hard constraints, we make extensive use of the compliance form of elasticity [185, 215]. We derive this by considering a quadratic energy potential defined in terms of a constraint function $C_b(\mathbf{q})$ and stiffness parameter $k > 0$

$$U(\mathbf{q}) = \frac{k}{2} C_b(\mathbf{q})^2. \tag{12}$$

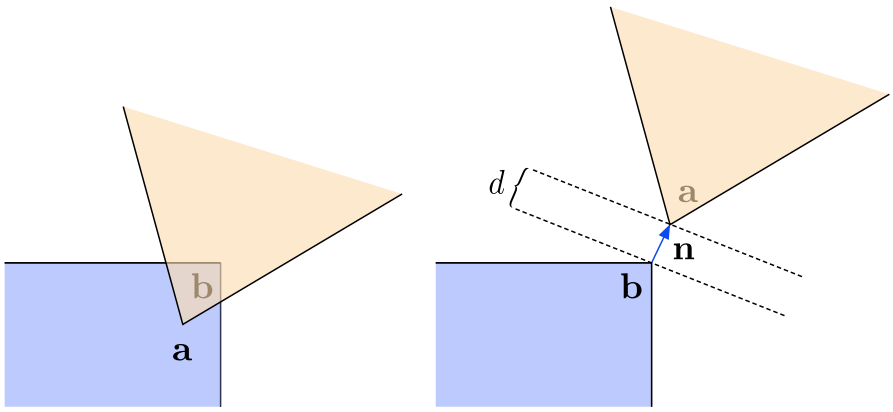



Figure 3: **Contact Constraint**: We model contact as a constraint on the distance between two points $\mathbf{a}$ and $\mathbf{b}$ as measured along a fixed normal direction $\mathbf{n}$ being greater than some minimum $d$.

The force due to this potential is then given by

$$\mathbf{f}_b = -\nabla U^T = -k\nabla C_b^T C_b(\mathbf{q}) = \nabla C_b^T \lambda_b. \tag{13}$$

Here we have introduced the variable $\lambda_b$, defined as $\lambda_b = -kC_b(\mathbf{q})$. We can move all terms to one side to write this as a constraint equation,

$$C_b(\mathbf{q}) + e\lambda_b = 0, \tag{14}$$

where $e = k^{-1}$ is the *compliance*, or inverse stiffness coefficient. We can incorporate this into our equations of motion by defining the diagonal compliance matrix $\mathbf{E} = \text{diag}(e_1, e_2, \cdots, e_{n_b}) \in \mathbb{R}^{n_b \times n_b}$, and updating (11) as follows,

$$\mathbf{c}_b(\mathbf{q}) + \mathbf{E}\boldsymbol{\lambda}_b = \mathbf{0}. \tag{15}$$

This form is mathematically equivalent to including quadratic energy potentials defined in terms of a stiffness. The benefit of the compliance form is that it remains numerically stable even as $k \to \infty$ [215].

### 2.1.5 Continuum Materials

The above process may be extended to more complex energies by extending the scalar quantities to higher dimensions. This allows modeling deformable solids as finite elements (FEM) using compliant constraints [185]. We now show how to extend the simple 1D spring potential presented above to a 3D Hookean constitutive model. Assuming a constant strain element, the strain energy can be written as:

$$U(\mathbf{q}) = V_e \frac{1}{2} \mathbf{s}^T \mathbf{K}_e \mathbf{s}, \tag{16}$$

where $\mathbf{s} = [\epsilon_{xx}, \epsilon_{yy}, \epsilon_{zz}, \epsilon_{yz}, \epsilon_{xz}, \epsilon_{xy}]$ is a vector of strain elements, $\mathbf{K}_e \in \mathbb{R}^{6\times6}$ is the element's material stiffness matrix, and $V_e$ the element rest volume. The corresponding compliance matrix is $\mathbf{E} = (V_e \mathbf{K}_e)^{-1}$ which is a constant that may be precomputed. In Section 4.7.4 we show how to generalize this form to hyperelastic materials where the strain energy density is not a simple quadratic form.

## 2.2 Contact Modeling

In this section we cover some mathematical background on contact modeling. We focus on complementarity formulations and give a derivation of isotropic Coulomb friction from a principle of maximal dissipation. In Section 3 we generalize this, and present a novel friction model for contact between two anisotropic materials inspired by sliding two Matchsticks against one another.

### 2.2.1 Non-Penetration

We initially treat contacts as inelastic and prevent interpenetration between bodies through hard inequality constraints of the form

$$C_n(\mathbf{q}) = \mathbf{n}^T \left[\mathbf{a}(\mathbf{q}) - \mathbf{b}(\mathbf{q})\right] - d \geq 0. \tag{17}$$

Here $\mathbf{n} \in \mathbb{R}^3$ is the contact plane normal created by normalizing the direction vector between closest points of e.g.: triangle-mesh features. The points $\mathbf{a}$ and $\mathbf{b} \in \mathbb{R}^3$ may be points defined in

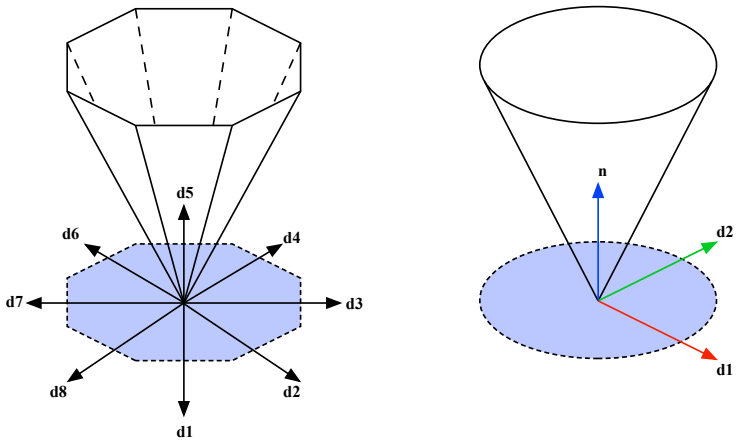



Figure 4: **Left**: A Coulomb friction cone approximated by linearizing into 8 directions leads to a larger LCP problem and introduces bias in some directions. **Right**: We project frictional forces directly to the smooth friction cone, and require only 2 tangential direction vectors.

terms of a rigid body frame, or particle positions in the case of a deformable body. The constant $d$ is a thickness parameter that represents the distance along the contact normal we wish to maintain. Non-zero values of $d$ can be used to model shape thickness, as illustrated in Figure 3. The normal force for a contact is given by $\mathbf{f}_n = \nabla C_n^T \lambda_n$, with the additional Signorini-Fichera complementarity condition

$$0 \leq C_n(\mathbf{q}) \perp \lambda_n \geq 0, \tag{18}$$

which ensures contact forces only act to separate objects [198]. We treat the contact normal $\mathbf{n}$ as fixed in world-space. For a system with $n_c$ contacts, we define the vector of contact constraints as $\mathbf{c}_n = [C_{n,1}, \cdots, C_{n,n_c}] \in \mathbb{R}^{n_c}$, their gradient $\nabla \mathbf{c}_n \in \mathbb{R}^{n_c \times n_d}$, and the associated Lagrange multipliers as $\boldsymbol{\lambda}_n \in \mathbb{R}^{n_c}$. In actual implementation code contact constraints are typically created when body features come within some fixed distance of each other. This approach works well for reasonably small time-steps, but can lead to over-constrained configurations. More sophisticated non-interpenetration constraints can be formulated to avoid this problem [227].

### 2.2.2 Isotropic Friction

We now give a derivation of Coulomb friction from a principle of maximal dissipation that requires the frictional forces remove the maximum amount of energy from the system while having their magnitude bounded by the normal force [198]. For each contact we define a two-

dimensional basis $\mathbf{D} = \mathbf{\Gamma}(\mathbf{q})[\mathbf{d}_1\mathbf{d}_2] \in \mathbb{R}^{n_d \times 2}$. The contact basis vectors $\mathbf{d}_1, \mathbf{d}_2 \in \mathbb{R}^{3\times 1}$ are lifted from spatial to generalized coordinates by the transform $\mathbf{\Gamma} \in \mathbb{R}^{n_d \times 3}$ using the notation of Kaufman et al. [109]. The generalized frictional force for a single contact is then $\mathbf{f}_f = \mathbf{D}\boldsymbol{\lambda}_f$, where $\boldsymbol{\lambda}_f \in \mathbb{R}^2$ is the solution to the following minimization

$$\begin{aligned} \underset{\boldsymbol{\lambda}_f}{\text{argmin}} \quad & \dot{\mathbf{q}}^T \mathbf{D}\boldsymbol{\lambda}_f \\ \text{subject to} \quad & |\boldsymbol{\lambda}_f|_2^2 \leq \mu^2 \lambda_n^2. \end{aligned} \tag{19}$$

Here $\mu$ is the coefficient of friction, and $\lambda_n$ is the Lagrange multiplier for the normal force at this contact which for the moment we assume is known. The inequality constraint on the frictional Lagrange multipliers defines an admissible cone that the total contact force must lie in, as illustrated in Figure 4. The Lagrangian associated with this minimization is

$$\mathcal{L}(\boldsymbol{\lambda}_f, \lambda_n) = \dot{\mathbf{q}}^T \mathbf{D}\boldsymbol{\lambda}_f + \lambda_s(\mu^2\lambda_n^2 - |\boldsymbol{\lambda}_f|_2^2). \tag{20}$$

where $\lambda_s$ is a slack variable used to enforce the Coulomb constraint that the friction force magnitude is bounded by the $\mu$ times the normal force magnitude. When $\mu\lambda_n > 0$ the problem satisfies Slater's condition [23] and we can use the first-order Karush-Kuhn-Tucker (KKT) conditions for (19) given by

$$\mathbf{D}^T\dot{\mathbf{q}} + 2\lambda_s\boldsymbol{\lambda}_f = \mathbf{0} \tag{21}$$

$$0 \leq \lambda_s \quad \perp \quad \lambda_s(\mu^2\lambda_n^2 - |\boldsymbol{\lambda}_f|_2^2) \geq 0. \tag{22}$$

Equation (21) requires that the frictional force act in a direction opposite to any relative tangential velocity. Equation (22) is a complementarity condition that describes the sticking and slipping behavior characteristic of dry friction. The quantity $\lambda_s$ is a slack variable that governs stick/slip transitions. When $\lambda_s = 0$ the friction force lies inside the Coulomb cone, and we must have zero tangential motion. When $\lambda_s > 0$ there is sliding, and the friction force must lie on the boundary of the cone.

# 3 Advanced Friction Modeling

We now explore friction modeling beyond the isotropic Coulomb model presented in the previous chapter. Before introducing a novel anisotropic friction model, we first generalize the derivation from the previous section to support arbitrary friction cones. To do this, we introduce a function $\psi(\boldsymbol{\lambda}_f) : \mathbb{R}^n \rightarrow \mathbb{R}$ that defines an admissible region the frictional forces must lie inside. The generalized optimization associated with the maximal dissipation principle is then given by,

$$\begin{aligned} \underset{\boldsymbol{\lambda}_f}{\text{argmin}} \quad & \dot{\mathbf{q}}^T \mathbf{D} \boldsymbol{\lambda}_f \\ \text{subject to} \quad & \psi(\boldsymbol{\lambda}_f) \geq 0, \end{aligned} \tag{23}$$

and the Lagrangian associated with this minimization is

$$\mathcal{L}(\boldsymbol{\lambda}_f) = \dot{\mathbf{q}}^T \mathbf{D} \boldsymbol{\lambda}_f + \lambda_s \psi(\boldsymbol{\lambda}_f), \tag{24}$$

which has the following optimality conditions:

$$\mathbf{D}^T \dot{\mathbf{q}} + \lambda_s \frac{\partial \psi}{\partial \boldsymbol{\lambda}_f}^T = \mathbf{0} \tag{25}$$

$$0 \leq \lambda_s \quad \perp \quad \psi(\boldsymbol{\lambda}_f) \geq 0. \tag{26}$$

From this we can see that isotropic Coulomb friction corresponds to the specific choice of $\psi(\boldsymbol{\lambda}_f) = \mu^2 \lambda_n^2 - |\boldsymbol{\lambda}_f|_2^2$. In the following section we review some previous work on anisotropic friction modeling, and then introduce a new choice of $\psi$ for modeling combinations of anisotropic surfaces. In Section 4 we discuss solution strategies for solving the complementarity problems arising from this generalized friction model.

## 3.1 Related Work

Friction measurements in mechanical engineering have for a long time demonstrated the shortcomings of the standard isotropic Coulomb model. Liley et al. [126] observed asymmetric anisotropic friction due to molecular tilt of a mica surface, and Umbanhowar et al. [219] used

direction-dependent surface friction properties to help design friction-induced velocity fields on a vibrating plate. Their experiments clearly show anisotropic behavior due to microscale orthographic rough geometric features. Yu and Wang [236] show that very rough microscale geometry gives rise to anisotropic friction strongly correlated to the microscale structure. Conversely, fine roughness appears to give near isotropic behavior. Hence, a certain sufficient roughness scale and directional structure is needed to get strong anisotropic response. This observation leads us to a model that can smoothly blend between isotropic and anisotropic cones.

The work addressing anisotropic friction modeling is relatively sparse [15]. Among the exceptions is the use of the friction tensors by Pabst et al. [161], who propose an additive model that uses nine parameters to describe the material for planar dynamic friction. Our Matchstick model uses the angle between structure directions (much like the tensor eigenvectors) to interpolate cone orientations in a common frame, and only uses a small number of intuitive parameters. In addition, the friction tensors of Pabst et al. do not provide an immediate cone description, but give a direct equation for the dynamic friction force that does not rely on maximum dissipation. Using the generalized derivation in the previous section, we show how to derive anisotropic friction from a principle of maximum dissipation, that to the best of our knowledge has not been published outside this thesis.

A pragmatic solution to generating anisotropic friction effects is to align the first axis of the contact frame with the relative sliding direction, as done in many physics engines and past work [15]. This approach however, does not consider the surface properties and orientations of both the contacting objects. In this work we propose a model uses the mean of the intrinsic material structure directions to estimate the direction of greatest resistance.

## 3.2 The *Matchstick* Model for Anisotropic Friction Cones

Often surfaces have a predominant direction in which friction occurs. For example, due to a grain, or fibers in the case of wood, or some other anisotropic microstructure in the material. To capture this effect, we can generate an ellipsoidal friction cone by choosing a rotation matrix

$$\mathbf{R} = \begin{bmatrix} \mathbf{t} & \mathbf{b} \end{bmatrix} \tag{27}$$

where $\mathbf{t}, \mathbf{b} \in \mathbb{R}^3$ are the tangent and bitangent basis vectors that define the principal axes of the ellipsoid. We use the convention that $\mathbf{t}$ is aligned with the major axis of primary resistance. Then, we can define the ellipsoidal cone using the quadratic form:

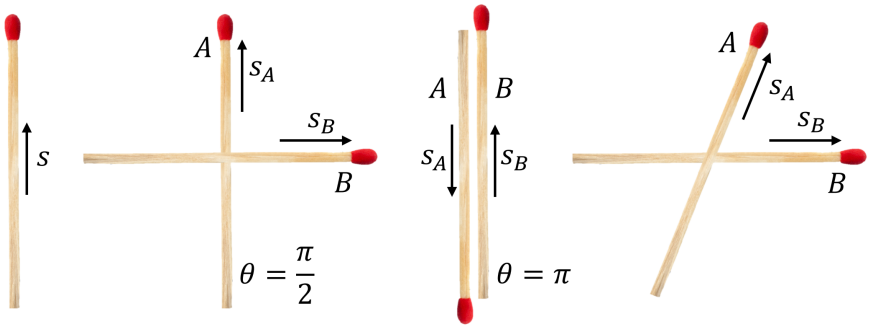



Figure 5: **The Matchstick Model**: Inspired by the anisotropic friction between matchsticks, our model interpolates between isotropic and anisotropic extremes based on the angle between the predominant structure directions $\mathbf{s}_a$, and $\mathbf{s}_b$.

$$\psi(\boldsymbol{\lambda}_f) = \lambda_n^2 - \boldsymbol{\lambda}_f^T \mathbf{R}^T \mathbf{C} \mathbf{R} \boldsymbol{\lambda}_f \geq 0, \tag{28}$$

where $\mathbf{C}$ is a diagonal matrix that determines the cone anisotropy. Given two contacting objects, potentially with different microstructure, there is a question of how to choose $\mathbf{R}$ and $\mathbf{C}$. Since frictional forces arising between two surfaces are dependent on the microscopic geometry of *both* surfaces, it does not make sense to use a predetermined values. Rather, the shape of friction cone should be a function of the combined contacting surface's material properties, and their alignment. In this section we propose a novel approach for generating friction cones for two anisotropic surfaces.

We begin by observing that, in the real world, matchsticks typically have a predominant direction of friction. When these directions are aligned, frictional forces between them are maximized. When they are orthogonal it is at a minimum. This observation suggests an interpolation between the two surface frames, and friction coefficients. We define an interpolation quantity $d$ as follows:

$$d \equiv 1 - \frac{2}{\pi} \cos^{-1} |\|\mathbf{s}_a^T \mathbf{s}_b\||, \tag{29}$$

where $\mathbf{s}_a, \mathbf{s}_b$ are the world-space *structure directions*, defining the predominant direction of friction as illustrated in Figure 5. We construct an interpolated basis $\mathbf{R}$ using the following definition, where the tangent $\mathbf{t}$ is obtained through a normalized average, and the bitangent, $\mathbf{b}$, is chosen to be an orthogonal in the plane:

$$\mathbf{t} = \frac{\mathbf{s}_a + \mathbf{s}_b}{|\mathbf{s}_a + \mathbf{s}_b|} \tag{30}$$

$$\mathbf{b} = \mathbf{n} \times \mathbf{t}. \tag{31}$$

The material matrix $\mathbf{C}$ defining the anisotropy of the friction cone along the tangent and bitangent directions is given by,

$$\mathbf{C} = \begin{bmatrix} \frac{1}{\mu_t}^2 & 0 \\ 0 & \frac{1}{\mu_b}^2 \end{bmatrix} \tag{32}$$

where the effective coefficients of friction are obtained by interpolating the two coefficients of friction according to $d$

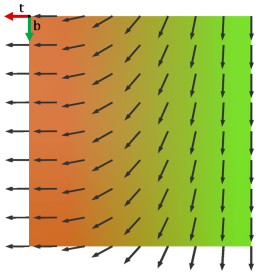
$$\mu_t = d\mu_a + (1-d)\mu \tag{33}$$

$$\mu_b = d\mu_b + (1-d)\mu. \tag{34}$$

Here $\mu$ is the isotropic parameter that may be chosen e.g.: through a simple averaging of two surface defined coefficients. Our proposed combination strategy smoothly varies the friction cone from anisotropic to isotropic as the structure directions become less aligned.

## 3.3 Structure Fields

From an implementation point of view, a method is required to store and query the structure directions of a surface at a point. Our method is agnostic to how this is performed, however a convenient representation is to store them as a texture map. The inline figure on the right shows an example of one such texture map used in our experiments, along with structure directions. Here, the red and green channels of the image are used to store the direction information.



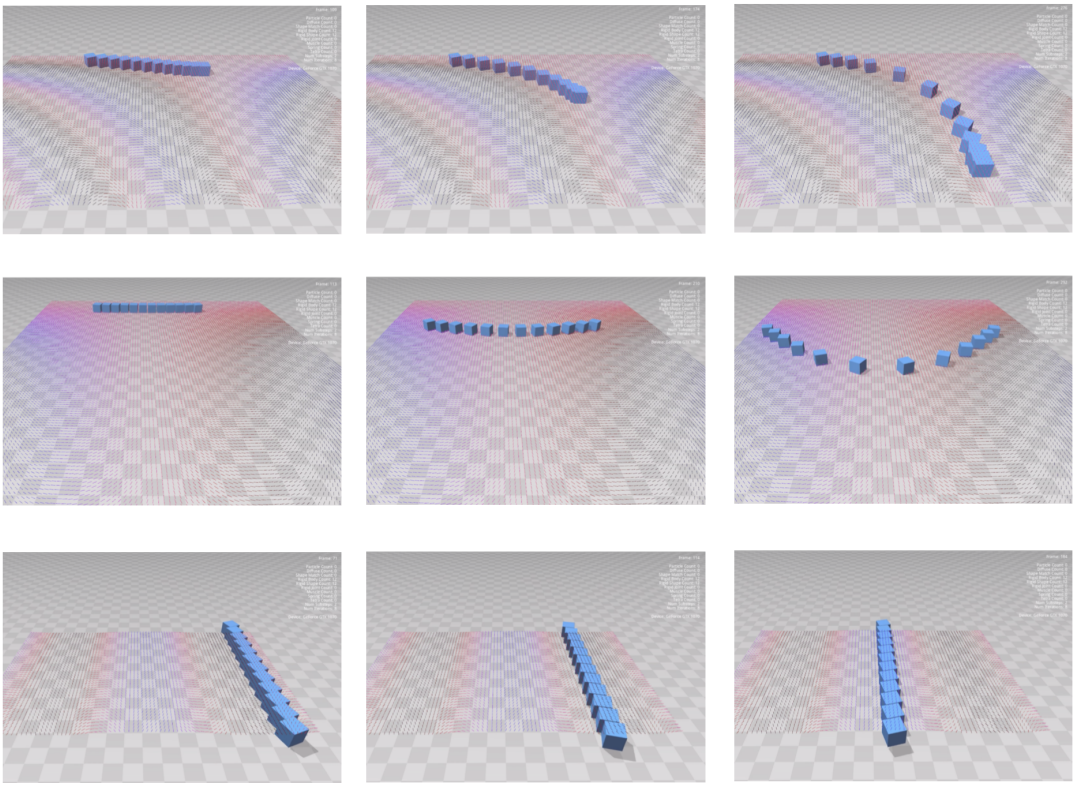
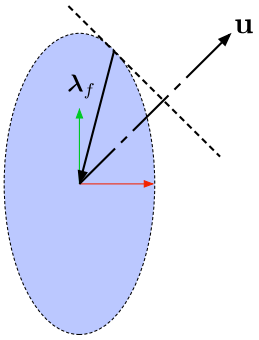


Figure 6: **Friction Control**: Procedurally generated structural directions provide intuitive interactive control. These didactic examples demonstrate channeling (top), spreading (middle), and slaloming (bottom).

## 3.4 Results

Using the solver developed in the following section, Figure 6 shows simple cases where the behavior of sliding boxes can be altered easily by generating a varying structure field for the plane. A surprising result of anisotropic friction is the generation of curved sliding trajectories. This is a result of the principle of maximal dissipation which requires the frictional force to no longer be aligned with the body velocity as illustrated in the top-down view on the right. This can be used for artist effect to 'steer' the trajectory of objects along predefined paths. On a sloped plane, Figures 7 and 8 show control of a log slide in a ravine.




We demonstrate changing structure directions on a soft object with the robotic grasping example shown in Figure 9. Interactive changes to the structure fields immediately change the behavior as the Allegro gripper strokes the soft gel-like material.

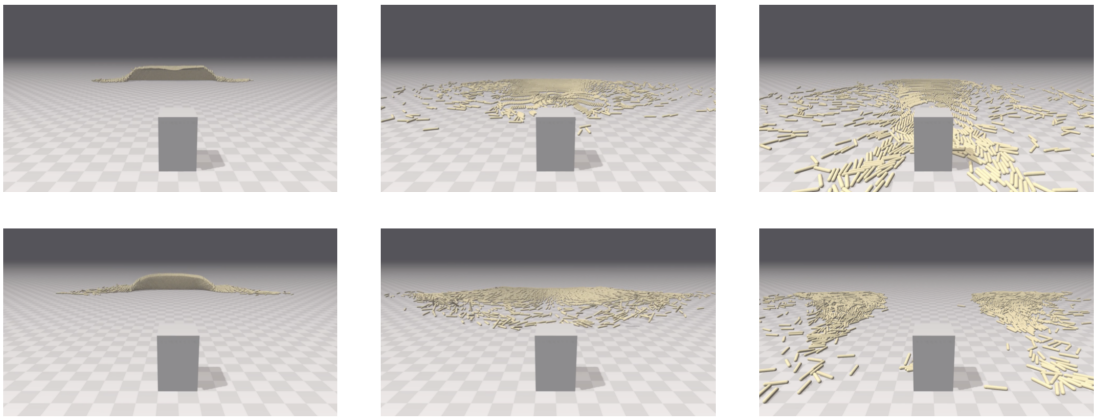


Figure 7: **Log Slide:** A structure field on a ravine slope can be used to steer a log slide towards (top) or around (bottom) a cube-shaped building.

## 3.5 Conclusion and Future Work

We have presented a new phenomenological anisotropic friction model for structured surfaces. The novelty of considering cones as state dependent parametric functions that can change dynamically opens up a doorway for a multitude of modeling possibilities not yet seen. We believe it will be straightforward to extend our work to address friction models with other dependencies, such as sliding velocity (Stribeck effect) or nonlinear scaling with normal forces as employed in recent work on cloth simulation [34]. We note that anisotropic friction is generally in demand for cloth simulation [117] and believe our model may be useful for self-collision between anisotropic fabrics.

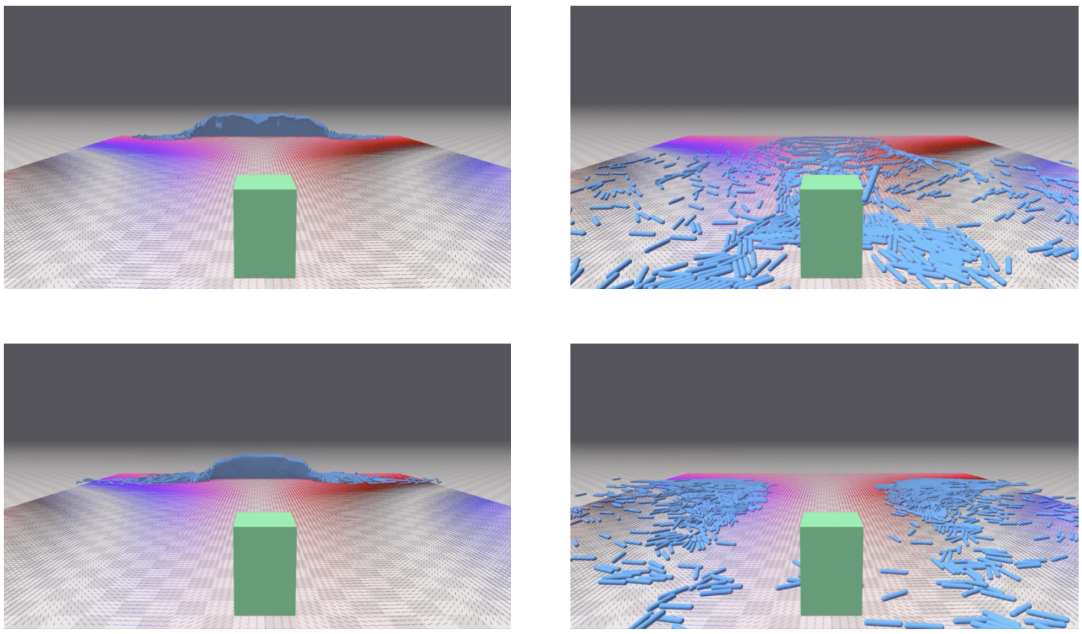


Figure 8: **Log Slide Visualization**: Visualization of the ravine example structure fields. Logs collide with the box (top), and spread prior to collision (bottom).

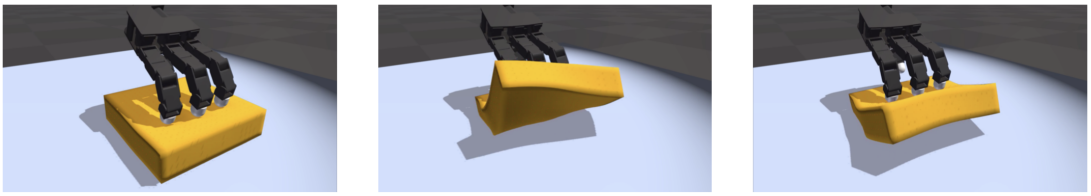


Figure 9: **Anisotropic Manipulation**: Grasping with a robotic gripper demonstrates how our model affects the friction behavior during interaction with a deformable object. Left: We align the minor friction axis in the horizontal direction, which leads to minimal deformation with a sideways sliding motion. Middle, Right: Forward/backwards motion in the primary structure direction causes the elastic sheet to pull up dramatically.

# 4 Non-Smooth Newton Methods

In this section we develop a framework based on Newton's method that solves the underlying nonlinear complementarity problems (NCPs) arising in multi-body dynamics. We combine a nonlinear contact model, articulated rigid-body model, and a hyperelastic material model as a system of semi-smooth equations, and show how it can be solved efficiently. A key advantage of our Newton-based approach is that it allows the use of off-the-shelf linear solvers as the fundamental computational building block. This flexibility means we can choose to use accurate direct solvers, or take advantage of highly-optimized iterative solvers available for parallel architectures such as graphics-processing units (GPUs).

Methods for simulating multi-body systems in the presence of contact and friction most commonly formulate a linear complementarity problem (LCP) that is solved in one of two ways: relaxation methods such as projected Gauss-Seidel (PGS), or direct methods such as Dantzig's pivoting algorithm. Relaxation methods are popular due to their simplicity, but suffer from slow convergence for poorly conditioned problems [57]. In contrast, direct methods can achieve greater accuracy, but are typically serial algorithms that scale poorly with problem size. Finding methods that combine the simplicity and robustness of relaxation methods with the accuracy of direct methods remains a challenge in computer graphics and robotics. While solving LCP problems efficiently is still the subject of active research, as a model they may not capture all of the dynamics we wish to simulate. For example, hyperelastic materials have highly nonlinear forces that significantly affect behavior compared to linear models [193]. In addition, contact models may be nonlinear, for example the smooth and possibly anisotropic friction cones presented in the previous section. This motivates the use a nonlinear solver capable of capturing these effects.

## 4.1 Related Work

### 4.1.1 Contact

The foundational work of Jean and Moreau [103] introduced an implicit time-stepping scheme for contact problems with friction. This work was further popularized by Stewart and Trinkle [199] who linearize the Coulomb friction cone and solve LCPs using Lemke's method in a fixed-point iteration to handle nonlinear forces. We also make use of a fixed-point iteration, but in contrast to their work we use non-smooth functions to model nonlinear friction cones. Kaufman et al. [108] proposed a method for implicit time-stepping of contact problems by solving two separate quadratic programs (QPs) for normal and frictional forces in a staggered manner. In

our work we do not stagger our system updates but solve for contact forces and friction forces in a combined system, and without using a linearized cone. In addition, rather than using QP solves our method requires only the solution of a symmetric linear system as a building block, allowing for the use of off-the-shelf linear solvers.

Relaxation methods such as projected Gauss-Seidel are popular in computer graphics thanks to their simplicity and robustness [16]. While robust, these methods suffer from slow convergence for poorly conditioned problems, e.g.: those with high-mass ratios [57]. In addition, Gauss-Seidel iteration suffers from order dependence and is challenging to parallelize, while Jacobi methods require modifications to ensure convergence [214]. Daviet et al. [43] used a change of variables to restate the Coulomb friction cone into a self-dual complementarity cone problem followed by a modified Fischer-Burmeister reformulation to obtain a local non-smooth root search problem. In comparison, we work directly with the friction cone as limit surfaces as we believe this provides us with more modeling freedom, e.g.: for anisotropic or non-symmetric friction cones. Another difference is that we solve for all contacts simultaneously rather than one-by-one.

Otaduy et al. [159] presented an implicit time-stepping scheme for deformable objects that solves a mixed linear complementarity problem (MLCP) using a nested Gauss-Seidel relaxation over primal and dual variables. Prior work on simulating smooth friction models has used proximal-map projection operators [105, 102, 58] which work by projecting contact forces to the friction cone one contact at a time until convergence. There has been considerable work to address the slow convergence of relaxation methods, Mazhar et al. [146] use a convexification of the frictional contact problem [9] to obtain a cone complementarity problem (CCP) and solve it using an accelerated version of projected gradient descent. Silcowitz et al. [191, 192] developed a method for solving LCPs based on non-smooth nonlinear conjugate gradient (NNCG) applied to a PGS iteration. Francu et al. [66] proposed an improved Jacobi method based on a projected conjugate residual (CR) method. We also make use of Krylov space linear solvers, however our Newton-based iteration is decoupled from the underlying linear backend, allowing the application of matrix-splitting relaxation methods, or even direct solvers.

Early work on Newton methods for contact problems used a formulation based on a generalized projection operator and an augmented Lagrangian approach to unilateral constraints [4, 40]. Newton-based approaches found in libraries such as PATH [50] have proved successful in practice, and have been applied to smooth Coulomb friction models in fiber assemblies in computer graphics [18, 43, 109]. These approaches formulate the complementarity problem in terms of non-smooth functions, and solve them with a generalized version of Newton's method. This approach can yield quadratic convergence, although with a higher per-iteration cost than relaxation methods.

Todorov [211] observed that if solving a nonlinear time-stepping problem, e.g.: due to an implicit time-discretization, it makes little sense to perform the friction cone linearization, since the smooth contact model can be treated simply as an additional set of nonlinear equations in the system. This observation is at the heart of our method, but in contrast to their work we formulate friction in terms of arbitrary NCP-functions, and extend our framework to handle deformable bodies. Our approach is based on Newton's method, and combined with our proposed preconditioner we show that it enables handling considerably more ill-conditioned problems than relaxation methods, while naturally accommodating nonlinear friction models. For a review of numerical methods for linear complementarity problems we refer to the book by Niebe and Erleben [154]. For a review of non-smooth methods applied to dynamics problems we refer to the book by Acary & Brogliato [2].

### 4.1.2 Coupled Systems

There has been considerable work in computer graphics on coupling between rigid and deformable bodies. Shinar et al. [187] proposed a method for the coupled simulation of rigid and deformable bodies through a multi-stage update that peforms collisions, contacts, and stabilization in separate passes. In contrast, we formulate a system update that includes elastic and contact dynamics in a single phase. Duriez [52] showed real-time control of soft-robots using a co-rotational FEM model and Gauss-Seidel based constraint solver. Servin et al. [185] introduced a compliant version of elasticity that fits naturally inside constrained rigid body simulators. Their work was extended by Tournier et al. [215] who include a geometric stiffness term to improve stability. They use temporal coherence of Lagrange multipliers to build the system Jacobian and compute a Cholesky decomposition, followed by a projected-Gauss Seidel solve for contact. For smaller problems our approach is compatible with direct solvers, however we avoid the requirement of dense matrix decompositions by using a diagonal geometric stiffness approximation inspired by the work of Andrews et al. [7], this improves stability and allows us to apply iterative methods.

In this work we propose a generalized view of compliance, and give a recipe for constructing the compliance form of an arbitrary material model given its strain-energy density function in terms of principle stretches, strains, or other parameterization. As an example we show how to formulate the stable Neo-Hookean material proposed by Smith et al. [193]. Liu et al. [132] propose a quasi-Newton method for hyperelastic materials based on Projective Dynamics [22] and model contact through stiff penalty forces. In contrast we model contact through complementarity constraints that naturally fit into existing multi-body simulations.

## 4.2 Governing Equations

Assembling the components presented in Chapter 2, our continuous equations of motion are given by the following nonlinear system of equations:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} - \mathbf{f}(\mathbf{q},\dot{\mathbf{q}}) - \nabla\mathbf{c}_b^T(\mathbf{q})\boldsymbol{\lambda}_b - \nabla\mathbf{c}_n^T(\mathbf{q})\boldsymbol{\lambda}_n - \nabla\mathbf{c}_f^T(\mathbf{q})\boldsymbol{\lambda}_f = \mathbf{0} \tag{35}$$

$$\mathbf{c}_b(\mathbf{q}) + \mathbf{E}\boldsymbol{\lambda}_b = \mathbf{0} \tag{36}$$

$$\mathbf{0} \leq \mathbf{c}_n(\mathbf{q}) \perp \boldsymbol{\lambda}_n \geq \mathbf{0} \tag{37}$$

$$i \in \mathcal{A}, \quad \mathbf{D}_i^T\dot{\mathbf{q}} + \lambda_{s,i}\frac{\partial\psi_i(\boldsymbol{\lambda}_{f,i})}{\partial\boldsymbol{\lambda}_{f,i}}^T = \mathbf{0} \tag{38}$$

$$i \in \mathcal{A}, \quad 0 \leq \lambda_{s,i} \perp \psi_i(\boldsymbol{\lambda}_{f,i}) \geq 0 \tag{39}$$

$$i \in \mathcal{I}, \quad \boldsymbol{\lambda}_{f,i} = \mathbf{0} \tag{40}$$

where $\mathcal{A} = \{i \in (1,\cdots,n_c) \mid \mu_i\lambda_{n,i} > 0\}$ is the set of all contact indices where the normal contact force is active, and $\mathcal{I} = \{i \in (1,\cdots,n_c) \mid \mu_i\lambda_{n,i} \leq 0\}$ is its complement. This combination of a differential equation with equality and complementarity conditions is known as Differential Variational Inequality (DVI) [198]. The coupling between normal and frictional complementarity problems makes the problem non-convex, and in the case of implicit time-integration leads to an NP-hard optimization problem [108]. In the next section we develop a practical method to solve this problem.

## 4.3 Nonlinear Complementarity

One successful approach [62] to solving nonlinear complementarity problems is to reformulate the problem in terms of a *NCP-function* whose roots satisfy the original complementarity conditions, i.e.: functions where the following equivalence holds

$$\phi(a,b) = 0 \iff 0 \leq a \perp b \geq 0. \tag{41}$$

Combined with an appropriate time-discretization, such NCP-functions turn our DVI problem into a root finding one. In general the functions $\phi$ are non-smooth, but allow us to apply a wider range of numerical methods [152].

### 4.3.1 Minimum-Map Formulation

The first NCP function we consider is the *minimum-map* defined as

$$\phi_{\text{min}}(a, b) = \min(a, b) = 0. \tag{42}$$

The equivalence of this function to the original NCP can be verified by examining the values associated with each conditional case [37]. We now show how this reformulation applies to our unilateral contact constraints. Recall that the complementarity condition associated with a contact constraint $C_n(\mathbf{q})$ and its associated Lagrange multiplier $\lambda_n$ is

$$0 \leq C_n(\mathbf{q}) \perp \lambda_n \geq 0. \tag{43}$$

We can write this in the equivalent minimum-map form as

$$\phi_n(\mathbf{q}, \lambda_n) \equiv \min(C_n(\mathbf{q}), \lambda_n) = 0, \tag{44}$$

which has the following derivatives,

$$\frac{\partial \phi_n}{\partial \mathbf{q}} = \begin{cases} \nabla C_n(\mathbf{q}), & C_n(\mathbf{q}) \leq \lambda_n \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{45}$$

$$\frac{\partial \phi_n}{\partial \lambda_n} = \begin{cases} 0, & C_n(\mathbf{q}) \leq \lambda_n \\ 1, & \text{otherwise.} \end{cases} \tag{46}$$

From these cases we can see that the minimum-map gives rise to an active-set style method where a contact is considered active if $C_n(\mathbf{q}) \leq \lambda_n$. Active contacts are treated as equality constraints, while for inactive contacts the minimum-map enforces that the constraint's Lagrange multiplier is zero.

### 4.3.2 Fischer-Burmeister Formulation

An alternative NCP-function is given by Fischer-Burmeister [63], who observe the roots of the following equation satisfy complementarity:

$$\phi_{\text{FB}}(a, b) \equiv a + b - \sqrt{a^2 + b^2} = 0. \tag{47}$$

The Fischer-Burmeister function is interesting because, unlike the minimum-map, it is smooth everywhere apart from the point $(a, b) = (0, 0)$. For $(a, b) \neq (0, 0)$ the partial derivatives of the Fisher-Burmeister function are given by:

$$\alpha(a, b) = \frac{\partial \phi_{\text{FB}}}{\partial a} = 1 - \frac{a}{\sqrt{a^2 + b^2}} \tag{48}$$

$$\beta(a, b) = \frac{\partial \phi_{\text{FB}}}{\partial b} = 1 - \frac{b}{\sqrt{a^2 + b^2}}. \tag{49}$$

At the point $(a, b) = (0, 0)$ the derivative is set-valued (see Figure 11). For Newton methods it suffices to choose any value from this subgradient. Erleben [57] compared how the choice of derivative at the non-smooth point affects convergence for LCP problems and found no overall best strategy. Thus, for simplicity we make the arbitrary choice of

$$\alpha(0, 0) = 0 \tag{50}$$

$$\beta(0, 0) = 1. \tag{51}$$

For a contact constraint $C_n$, with Lagrange multiplier $\lambda_n$ we may then define our contact function alternatively as,

$$\phi_n(\mathbf{q}, \lambda_n) \equiv \phi_{\text{FB}}(C_n(\mathbf{q}), \lambda_n) = 0, \tag{52}$$

with derivatives given by

$$\frac{\partial \phi_n}{\partial \mathbf{q}} = \alpha(C_n, \lambda_n) \nabla C_n \tag{53}$$

$$\frac{\partial \phi_n}{\partial \lambda_n} = \beta(C_n, \lambda_n). \tag{54}$$

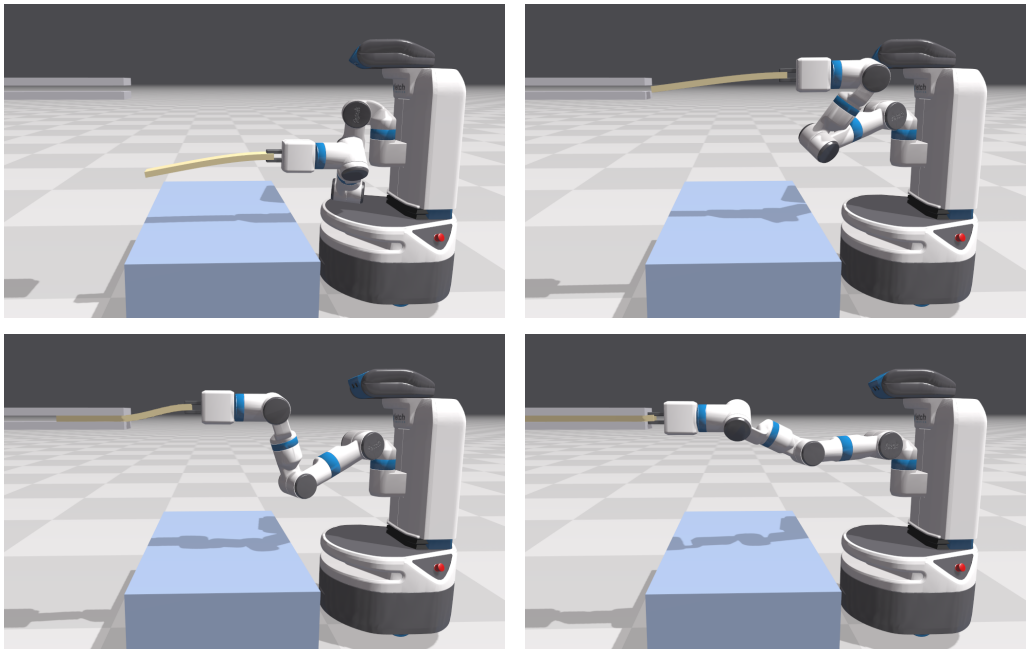


Figure 10: **Flexible Beam Insertion**: The Fetch robot performing a flexible beam insertion task. The beam is modeled as 16 rigid bodies connected by joints with a bending stiffness of 250 N·m. Relaxation methods such as Jacobi struggle to achieve sufficient stiffness on the joints, while direct methods struggle with large contacting systems near the end of simulation. Our iterative method based on PCR achieves high stiffness and robust behavior in contact.

### 4.3.3 Frictional Constraints

As we did for contact above, we can convert our frictional complementarity conditions into non-smooth equations. For each contact we define the following two constraint functions to represent the friction conditions,

$$\boldsymbol{\phi}_f \equiv \mathbf{D}^T \dot{\mathbf{q}} + \lambda_s \frac{\partial \psi}{\partial \boldsymbol{\lambda}_f}^T = \mathbf{0} \tag{55}$$

$$\phi_s \equiv \phi_{\text{min}}(\lambda_s, \psi(\boldsymbol{\lambda}_f, \lambda_n)) = 0, \tag{56}$$

where we may replace the minimum map with any valid NCP function. Before discretizing our equations of motion we first group the normal and friction NCP-functions for all contacts into three vectors,

$$\boldsymbol{\phi}_n = [\phi_{n,1}, \cdots, \phi_{n,nc}]^T \tag{57}$$

$$\boldsymbol{\phi}_f = [\boldsymbol{\phi}_{f,1}, \cdots, \boldsymbol{\phi}_{f,nc}]^T \tag{58}$$

$$\boldsymbol{\phi}_s = [\phi_{s,1}, \cdots, \phi_{s,nc}]^T, \tag{59}$$

and define the following Jacobians with respect to the generalized velocity as

$$\mathbf{J}_b = \frac{\partial \mathbf{c}_b}{\partial \mathbf{q}} \mathbf{G}, \quad \mathbf{J}_n = \frac{\partial \boldsymbol{\phi}_n}{\partial \mathbf{q}} \mathbf{G}, \quad \mathbf{J}_f = \frac{\partial \boldsymbol{\phi}_f}{\partial \dot{\mathbf{q}}} \mathbf{G}. \tag{60}$$

## 4.4 Implicit Time-Stepping

Using a first-order backwards time-discretization of $\dot{\mathbf{q}} = \mathbf{G}\mathbf{u}$ gives the following update for the system's generalized coordinates in terms of generalized velocities over a time-step of length $\Delta t$,

$$\mathbf{q}^+ = \mathbf{q}^- + \Delta t \mathbf{G}(\mathbf{q}^+)\mathbf{u}^+. \tag{61}$$

The superscripts $+, -$ represent a variable's value at the end and beginning of the time-step respectively. Discretizing our continuous equations of motion gives the following implicit time-stepping equations,

$$\tilde{\mathbf{M}}\left(\frac{\mathbf{u}^+ - \tilde{\mathbf{u}}}{\Delta t}\right) - \mathbf{J}_b^T(\mathbf{q}^+)\boldsymbol{\lambda}_b^+ - \mathbf{J}_n^T(\mathbf{q}^+)\boldsymbol{\lambda}_n^+ - \mathbf{J}_f^T(\mathbf{q}^+)\boldsymbol{\lambda}_f^+ = \mathbf{0} \tag{62}$$

$$\mathbf{c}_b(\mathbf{q}^+) + \mathbf{E}\boldsymbol{\lambda}_b^+ = \mathbf{0} \tag{63}$$

$$\boldsymbol{\phi}_n(\mathbf{q}^+, \boldsymbol{\lambda}_n^+) = \mathbf{0} \tag{64}$$

$$\boldsymbol{\phi}_f(\mathbf{u}^+, \boldsymbol{\lambda}_s^+, \boldsymbol{\lambda}_f^+) = \mathbf{0} \tag{65}$$

$$\boldsymbol{\phi}_s(\boldsymbol{\lambda}_s^+, \boldsymbol{\lambda}_f^+, \boldsymbol{\lambda}_n^+) = \mathbf{0} \tag{66}$$

$$\mathbf{q}^+ - \mathbf{q}^- - \Delta t\mathbf{G}\mathbf{u}^+ = \mathbf{0}. \tag{67}$$

Here $\mathbf{u}^+, \boldsymbol{\lambda}^+$ are the unknown velocities and multipliers at the end of the time-step. The constant $\tilde{\mathbf{u}} = \mathbf{u}^- + \Delta t\mathbf{G}^T\mathbf{M}^{-1}\mathbf{f}(\mathbf{q}^-, \dot{\mathbf{q}}^-)$ is the unconstrained velocity that includes the external and gyroscopic forces integrated explicitly. Observe that through this time-discretization the original force level model of friction has changed into an impulsive model. This means friction impulses are able to prevent sliding and interpenetration instantaneously (over a single time-step in the discrete setting). This avoids the inconsistency raised by Painlevé in the continuous setting.

We highlight a few differences from common formulations. First, the equality and inequality constraints have not been linearized through an index reduction step [79]. Index reduction is a common practice that reduces the order of the DAE by solving only for the constraint time-derivatives, e.g.: $\dot{\mathbf{c}}_b = \mathbf{0}$. Index reduction results in a linear problem, but requires additional stabilization terms to combat drift and move the system back to the constraint manifold. These stabilization terms are often applied as the equivalent of penalty forces [11] and are known as a source of instability and tuning issues. Work has been done to add additional post-stabilization passes [36], however these require solving nonlinear projection problems, which gives up the primary benefit of performing the index reduction step. Our discrete equations of motion are also nonlinear, but they require no additional stabilization terms or projection passes.

A second point we highlight is that the friction cone defined through (19) has not been linearized into a faceted cone as is common [199, 108]. The faceted cone approximation leads to a simpler linear complementarity problem (LCP), but increases the number of Lagrange multipliers required (one per-facet), and introduces an approximation bias where frictional effects are stronger in some directions than others. In the next section we propose a method that solves the NCP corresponding to smooth isotropic friction using a fixed number of Lagrange multipliers.

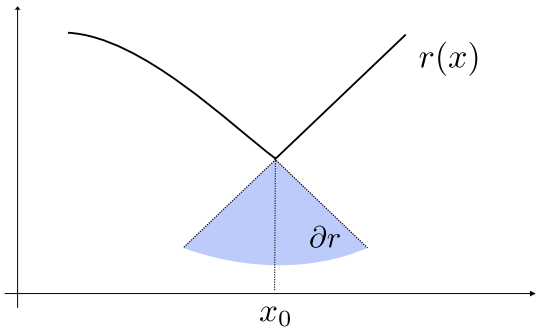



Figure 11: **Non-Smooth Derivatives**: The derivative of a non-smooth function is set valued at discontinuities. The shaded area represents the generalized Jacobian $\partial r(\mathbf{x}_0)$, defined as the convex hull of directional derivatives at $x_0$.

## 4.5 Non-Smooth Newton

We now develop a method to solve the discretized equations of motion (62)-(67). Our starting point is Newton's method which we briefly review here. Given a set of nonlinear equations $\mathbf{r}(\mathbf{x}) = \mathbf{0}$ in terms of a vector valued variable $\mathbf{x}$, Newton's method gives a fixed point iteration in the following form:

$$\mathbf{x}^{n+1} = \mathbf{x}^n - \mathbf{A}^{-1}(\mathbf{x}^n)\mathbf{r}(\mathbf{x}^n), \tag{68}$$

where $n$ is the Newton iteration index. Newton's method chooses $\mathbf{A}$ specifically to be the matrix of partial derivatives evaluated at the current system state, i.e.:

$$\mathbf{A} = \frac{\partial \mathbf{r}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{r}_1}{\partial \mathbf{x}_1} & \cdots & \frac{\partial \mathbf{r}_1}{\partial \mathbf{x}_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{r}_n}{\partial \mathbf{x}_1} & \cdots & \frac{\partial \mathbf{r}_n}{\partial \mathbf{x}_m} \end{bmatrix}. \tag{69}$$

Although Newton's method is most commonly used for solving systems of smooth functions it may also be applied to non-smooth functions by generalizing the concept of a derivative [164]. Qi and Sun [172] showed that Newton will converge for non-smooth functions provided $\mathbf{A} \in \partial \mathbf{r}(\mathbf{x}^k)$, where $\partial \mathbf{r}$ is the *generalized Jacobian* of $\mathbf{r}$ at $\mathbf{x}^k$ defined by Clarke [35]. Intuitively, this is the convex hull of all directional derivatives at the non-smooth point, as illustrated in Figure 11. The derivatives of our NCP-functions given in the previous section belong to this

**Algorithm 1:** Simulation Loop

```
while Simulating do
    Perform Collision Detection;
    for N Newton Iterations do
        Assembly M, H, J, C, g, h, b;
        for M Linear Iterations do
            Update Solution to [JH^{-1}J^T + C]Δλ = b;
        end
        Perform Line Search to find α (optional)
        λ^{n+1} = λ^n + αΔλ
        u^{n+1} = u^n + αΔu
        q^{n+1} = q^n + ΔtGu^{n+1}
    end
end
```

subgradient, and we can use them in the fixed-point iteration of (68) directly. Algorithms of this type are sometimes referred to as *semismooth* methods, we refer the reader to the article by Hintermüller [89] for a more mathematical introduction and the precise definition of semismoothness. In many cases $\mathbf{A}$ is not inverted directly, and the linear system for $\Delta\mathbf{x}$ may only be solved approximately. When this is the case we refer to the method as an inexact Newton method. Additionally, when the partial derivatives in $\mathbf{A}$ are also approximated we refer to it as a quasi-Newton method. In our method we make use of both approximations.

### 4.5.1 System Assembly

Linearizing our discrete equations of motion (62)-(67) each Newton iteration would yield the following linear system in terms of the change in system variables:

$$\begin{bmatrix} \tilde{\mathbf{M}} - \Delta t^2 \mathbf{K} & -\mathbf{J}_b^T & -\mathbf{J}_n^T & -\mathbf{J}_f^T & \mathbf{0} \\ \mathbf{J}_b & \frac{\partial \phi}{\partial \boldsymbol{\lambda}_b} p & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_n & \mathbf{0} & \frac{\partial \phi_n}{\partial \boldsymbol{\lambda_f}} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_f & \mathbf{0} & \mathbf{0} & \frac{\partial \phi_f}{\partial \boldsymbol{\lambda_f}} & \frac{\partial \phi_f}{\partial \boldsymbol{\lambda_s}} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \phi_s}{\partial \boldsymbol{\lambda_n}} & \frac{\partial \phi_s}{\partial \boldsymbol{\lambda_f}} & \frac{\partial \phi_s}{\partial \boldsymbol{\lambda_s}} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u} \\ \Delta \boldsymbol{\lambda}_b \Delta t \\ \Delta \boldsymbol{\lambda}_n \Delta t \\ \Delta \boldsymbol{\lambda}_f \Delta t \\ \Delta \boldsymbol{\lambda}_s \Delta t \end{bmatrix} = - \begin{bmatrix} \mathbf{g} \\ \boldsymbol{\phi}_b \\ \boldsymbol{\phi}_n \\ \boldsymbol{\phi}_f \\ \boldsymbol{\phi}_s \end{bmatrix}. \tag{70}$$

This matrix is clearly non-symmetric due to the inclusion of the function $\boldsymbol{\phi}_s$ that couples

together the normal and frictional Lagrange multipliers. This asymmetry may be avoided by lagging, or staggering the Lagrange multiplier updates and treating the slack variables as constant during each solve [199, 108]. To do this in the context of a Newton update we propose a separate fixed-point iteration to linearize the relationship between $\lambda_s$ and $\boldsymbol{\lambda}_f$. We motivate this by inspecting the first friction condition,

$$\boldsymbol{\phi}_f \equiv \mathbf{D}^T\dot{\mathbf{q}} + \lambda_s \frac{\partial \psi}{\partial \boldsymbol{\lambda}_f}^T = \mathbf{0}, \tag{71}$$

and observe that $\lambda_s$ is a scaling factor that must force the two vector-valued terms to be zero i.e.: $\lambda_s = \frac{|\mathbf{D}^T\dot{\mathbf{q}}|}{|\frac{\partial \psi}{\partial \boldsymbol{\lambda}_f}|}$. Starting with this relationship, we include the complementarity constraint using the following fixed-point iteration,

$$\lambda_s^{n+1} \leftarrow \frac{|\mathbf{D}^T\dot{\mathbf{q}}|^n - \phi_s^n}{|\frac{\partial \psi}{\partial \boldsymbol{\lambda}_f}|^n + \phi_s^n}. \tag{72}$$

By construction, a fixed-point of this iteration satisfies $\phi_s = 0$. We note that conical equivalents of the Fischer-Burmeister function exist and have been used to model smooth isotropic friction [70, 43]. One advantage of our method being based on a fixed-point iteration is that it can be extended to arbitrary friction surfaces for e.g.: anisotropic or even non-symmetric friction cones. The derivatives of $\boldsymbol{\phi}_f$ are then given by,

$$\frac{\partial \boldsymbol{\phi}_f}{\partial \dot{\mathbf{q}}} = \mathbf{D}^T, \quad \frac{\partial \boldsymbol{\phi}_f}{\partial \boldsymbol{\lambda}_f} = \lambda_s \frac{\partial^2 \psi}{\partial \boldsymbol{\lambda}_f^2}^T. \tag{73}$$

### 4.5.2 Final System

We can now write our symmetric Newton system as:

$$\begin{bmatrix} \tilde{\mathbf{M}} - \Delta t^2 \mathbf{K} & -\mathbf{J}_b^T & -\mathbf{J}_n^T & -\mathbf{J}_f^T \\ \mathbf{J}_b & \frac{\mathbf{E}}{\Delta t^2} & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_n & \mathbf{0} & \frac{\mathbf{S}}{\Delta t^2} & \mathbf{0} \\ \mathbf{J}_f & \mathbf{0} & \mathbf{0} & \frac{\mathbf{W}}{\Delta t} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u} \\ \Delta \boldsymbol{\lambda}_b \Delta t \\ \Delta \boldsymbol{\lambda}_n \Delta t \\ \Delta \boldsymbol{\lambda}_f \Delta t \end{bmatrix} = - \begin{bmatrix} \mathbf{g} \\ \mathbf{h}_b \\ \mathbf{h}_n \\ \mathbf{h}_f \end{bmatrix} \tag{74}$$

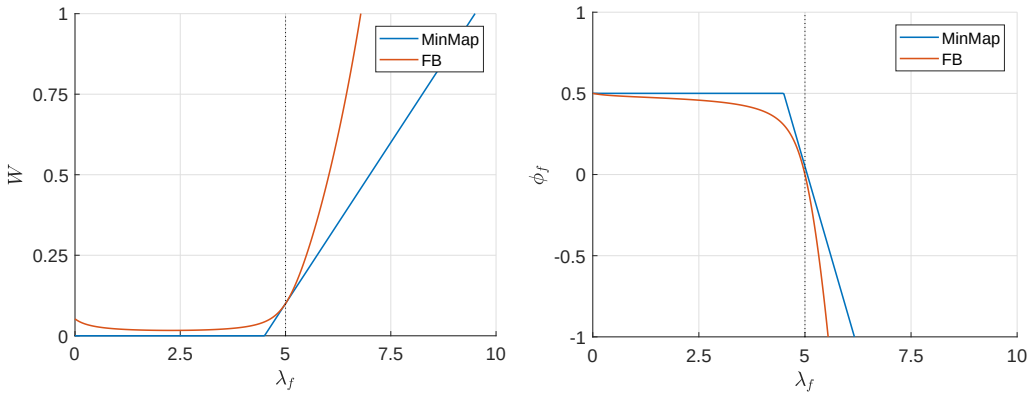



Figure 12: **Frictional Constraints**: Left: We plot the value of our frictional compliance term $W$ for a 1-dimensional particle sliding on a plane with velocity $u_0 = 0.5$m/s, $\lambda_n = 10$N, $\mu = 0.5$. The dashed line represents the friction cone limit at $\lambda_f = 5$N, after this point $W$ acts to strongly penalize the Lagrange multiplier. Right: The frictional error function $\phi_f$ for the same scenario. We see that both the Fischer-Burmeister and Minimum-Map functions are zero at the cone limit which indicates sliding.

where $\mathbf{K}$ is the geometric stiffness matrix arising from the spatial derivatives of constraint forces discussed in Section 4.7.3. The lower-diagonal blocks are the derivatives of our contact functions with respect to their Lagrange multipliers.

$$\mathbf{S} = \frac{\partial \boldsymbol{\phi}_n}{\partial \boldsymbol{\lambda}_n}, \quad \mathbf{W} = \frac{\partial \boldsymbol{\phi}_f}{\partial \boldsymbol{\lambda}_f}. \tag{75}$$

Here $\mathbf{W}$ may be interpreted as a frictional compliance term that acts to project the friction forces back onto the friction cone as illustrated in Figure 12. Grouping the sub-block components such that

$$\mathbf{H} = \left[\tilde{\mathbf{M}} - \Delta t^2 \mathbf{K}\right], \quad \mathbf{J} = \begin{bmatrix} \mathbf{J}_b \\ \mathbf{J}_n \\ \mathbf{J}_f \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \frac{\mathbf{E}}{\Delta t^2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\mathbf{S}}{\Delta t^2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\mathbf{W}}{\Delta t} \end{bmatrix}, \quad \boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda}_b \\ \boldsymbol{\lambda}_n \\ \boldsymbol{\lambda}_f \end{bmatrix} \tag{76}$$

we can write the system compactly as,

$$\begin{bmatrix} \mathbf{H} & -\mathbf{J}^T \\ \mathbf{J} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u} \\ \Delta \boldsymbol{\lambda} \Delta t \end{bmatrix} = - \begin{bmatrix} \mathbf{g} \\ \mathbf{h} \end{bmatrix}. \tag{77}$$

The right-hand side is given by evaluating our discrete equations of motion (62)-(67) at the current Newton iterate. Here, $\mathbf{g}$ is our momentum balance equation,

$$\mathbf{g} = \tilde{\mathbf{M}}\left(\mathbf{u}^n - \tilde{\mathbf{u}}\right) - \Delta t \mathbf{J}^T \boldsymbol{\lambda}^n, \tag{78}$$

and $\mathbf{h}$ is a vector of constraint errors,

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}_b \\ \mathbf{h}_n \\ \mathbf{h}_f \end{bmatrix} = \begin{bmatrix} \frac{1}{\Delta t}\mathbf{1} & & \\ & \frac{1}{\Delta t}\mathbf{1} & \\ & & \mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{c}_b(\mathbf{q}^n) + \mathbf{E}(\mathbf{q}^n)\boldsymbol{\lambda}_b^n \\ \boldsymbol{\phi}_n(\mathbf{q}^n, \boldsymbol{\lambda}_n^n) \\ \boldsymbol{\phi}_f(\mathbf{u}^n, \boldsymbol{\lambda}_f^n) \end{bmatrix}. \tag{79}$$

Here the left-hand side matrix should be considered acting block-wise on the constraint error. Since the frictional constraints are measured at the velocity level they are not scaled by $\frac{1}{\Delta t}$ like the positional constraints.

The mass block matrix $\mathbf{M}$ is evaluated at the beginning of the time-step using $\mathbf{q}^-$, while the $\mathbf{H}$ matrix is evaluated each Newton iteration as described in Section 4.7.3. The friction compliance block, $\mathbf{W}$, is evaluated at each Newton iteration using the current Lagrange multipliers. For inactive contacts with $\lambda_n \geq 0$ we conceptually disable their frictional constraint equation rows by removing them from the system. In practice this can be performed by zeroing the corresponding rows to avoid changing the system matrix structure.

### 4.5.3 Schur-Complement

The system (77) is a saddle-point problem [17] that is indefinite and possibly singular. To obtain a reduced positive semi-definite system, we take the Schur-complement with respect to the mass-block to obtain

$$\left[\mathbf{J}\mathbf{H}^{-1}\mathbf{J}^T + \mathbf{C}\right]\Delta\boldsymbol{\lambda} = \frac{1}{\Delta t}\left(\mathbf{J}\mathbf{H}^{-1}\mathbf{g} - \mathbf{h}\right). \tag{80}$$

After solving this system for $\Delta\boldsymbol{\lambda}$ the velocity change $\Delta\mathbf{u}$ may be evaluated directly,

$$\Delta\mathbf{u} = \mathbf{H}^{-1}\left(\mathbf{J}^T\Delta\boldsymbol{\lambda}\Delta t - \mathbf{g}\right) \tag{81}$$

and the system updated accordingly,

$$\boldsymbol{\lambda}^{n+1} = \boldsymbol{\lambda}^{n} + \alpha\Delta\boldsymbol{\lambda} \tag{82}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^{n} + \alpha\Delta\mathbf{u} \tag{83}$$

$$\mathbf{q}^{n+1} = \mathbf{q}^{n} + \Delta t\mathbf{G}\mathbf{u}^{n+1}. \tag{84}$$

Here $\alpha$ is a step-size determined by line-search or other means (see Section 4.7). We refer to (80) as the Newton *compliance formulation*. Under certain conditions we could alternatively have taken the Schur-complement of (77) with respect to the compliance block $\mathbf{C}$, instead of the mass block $\mathbf{H}$. This transformation is only possible if $\mathbf{C}$ is non-singular, but it leads to what we call the Newton *stiffness formulation*,

$$\left[\mathbf{H} + \mathbf{J}^{T}\mathbf{C}^{-1}\mathbf{J}\right]\Delta\mathbf{u} = -\mathbf{g} - \mathbf{J}^{T}\mathbf{C}^{-1}\mathbf{h}. \tag{85}$$

This form is closely related to that of Projective Dynamics [22], and arises from a linearization of the elastic forces due to a quadratic energy potential. Having $\mathbf{C}$ be invertible corresponds to having compliance everywhere, or in other words, no perfectly hard constraints. For stiff materials, where $\mathbf{C}$ is poorly conditioned, this approach leads to numerical problems in calculating $\mathbf{C}^{-1}$. However, if the system has fewer degrees of freedom than constraints this transformation can result in a smaller system. In this work we are interested in methods that combine stiff constraints with deformable bodies, so we use the compliance form which accommodates both. We further explore the relationship between these two formulations of dynamics in Chapter 5.

## 4.6 Complementarity Precondtioning

An interesting property of the non-smooth complementarity formulations is that their solutions are invariant up to a constant positive scale $r$ applied to either of the arguments. Specifically, a solution to $\phi(a, b) = 0$, is also a solution to the scaled problem, $\phi(a, rb) = 0$. Alart [3] presented an analysis of the optimal choice of $r$ in the context of linear elasticity in a quasistatic setting. Erleben [58] also explored this free parameter in the context of proximal-map operators for rigid bodies. In this section we propose a new complementarity preconditioning strategy to improve convergence by choosing $r$ based on the system properties and discrete time-stepping equations. To motivate our method, we make the observation that the two sides of the complementarity condition typically have different physical units. For example, a contact NCP function

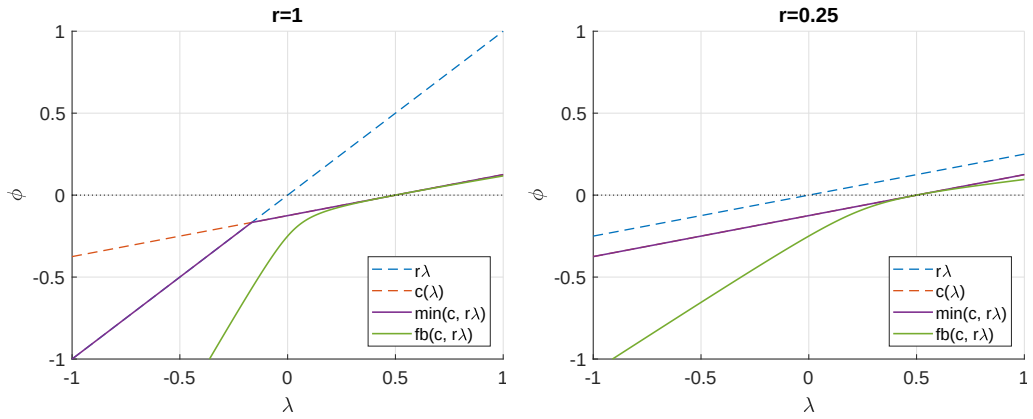



Figure 13: **NCP Preconditioning**: Left: The minimum-map of an NCP function has a kink in it at the cross over point. Right: We propose a preconditioner that removes this discontinuity by forcing both terms to be parallel. For a single constraint this results in a straight-line error function that can be solved in one step regardless of starting point (right). In the case of Fischer-Burmeister (green) the error function's curvature is reduced. For illustration purposes we have shown the constraint function $c = \frac{1}{4}\lambda - \frac{1}{8} \geq 0$, which has a unique solution at $\lambda = \frac{1}{2}$.

$$\phi(C_n, \lambda_n) = 0 \tag{86}$$

has the units of meters (m) for the first parameter, and units of Newtons (N) for the second parameter. This mismatch can lead to poor convergence in a manner similar to the effect of row-scaling in traditional iterative linear solvers.

Inspired by the use of diagonal preconditioners for linear solvers, our idea is to use the effective system mass and time-stepping equations to put both sides of the complementarity condition into the same units. The appropriate $r$-factor to perform this scaling comes from the relation between $\boldsymbol{\lambda}$ and our constraint functions, given through the discrete equations of motion. Specifically, for a unilateral constraint with index $i$ we choose $r_i$ to be,

$$r_i = \Delta t^2 \left[\mathbf{J}\tilde{\mathbf{M}}^{-1}\mathbf{J}^T\right]_{ii}. \tag{87}$$

Intuitively, this is the time-step scaled effective mass of the system, and relates how a change in Lagrange multiplier affects the corresponding constraint value. This has the effect of making both sides of the complementarity equation have the same slope, as illustrated in Figure 13. For a friction constraint $\phi_f$ the correct scaling factor is $r_i = \Delta t \left[\mathbf{J}\tilde{\mathbf{M}}^{-1}\mathbf{J}^T\right]_{ii}$ since this is a velocity-force relationship. When using a Jacobi preconditioner the diagonal of the system matrix is

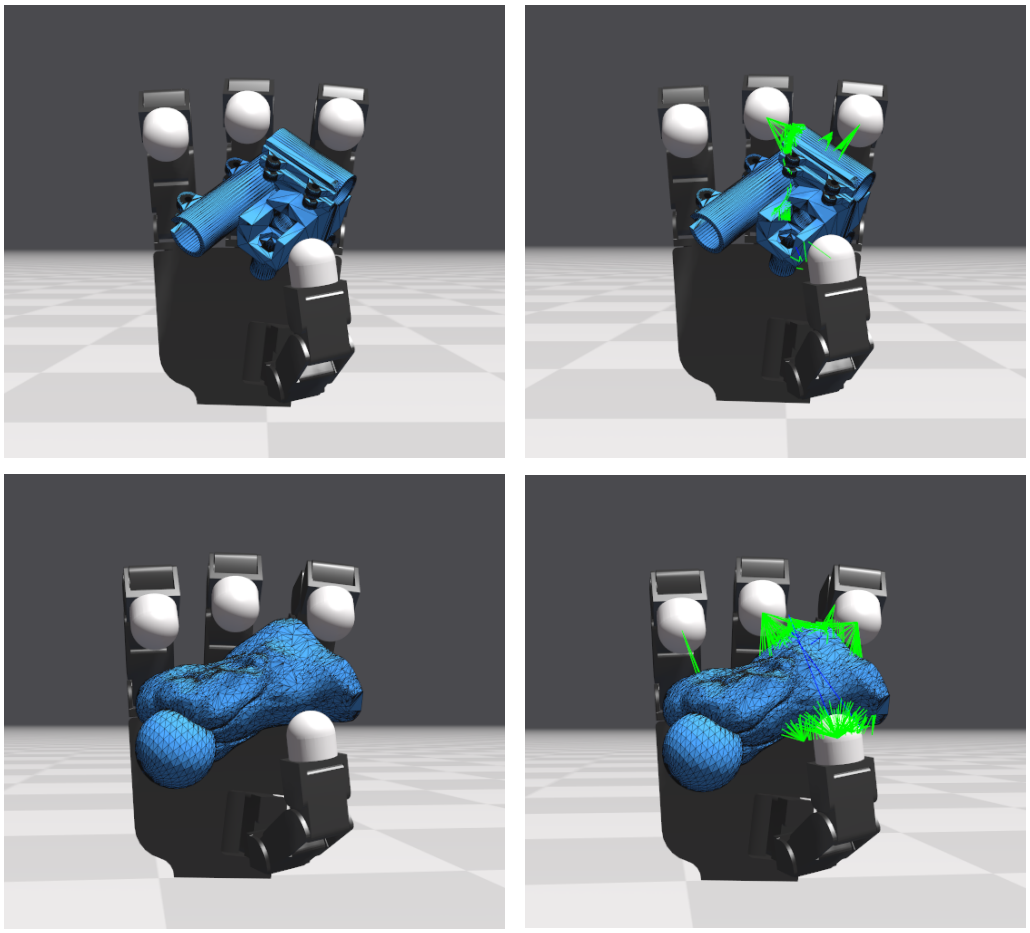


Figure 14: **Grasping Example**: A selection of grasping tests using the Allegro hand and objects from the DexNet adversarial mesh collection [139]. Our method simulates stable grasps around the irregular objects, and is robust to the over-specified contact sets generated by the high-resolution and often non-manifold meshes (right).

already computed, so applying $r_i$ to the NCP function incurs little overhead. We discuss the effect of preconditioning strategies in Section 4.8.1. In Appendix A we derive the gradients for our NCP functions with r-scaling factors included.

## 4.7 Robustness

### 4.7.1 Line Search and Starting Iterate

We implemented a back-tracking line-search based on a merit function defined as the $L_2$ norm of our residual vector. For frictionless contact problems we found this worked well to globalize the

solution. However, for frictional problems we found line search would often cause the iteration to stall and make no progress. We believe this is related to the fact that the frictional problem is non-convex and our search direction is not necessarily a descent direction. For our results, we found a simple damped Newton strategy that accepts a constant fraction of the full step with a factor $\alpha \in (0, 1)$ worked well. This strategy may cause a temporary increase in the merit function, but can result in overall better convergence [141]. Watchdog strategies [157] may be employed to make stronger convergence guarantees, however we did not find them necessary. We have used a value of $\alpha = 0.75$ for all examples unless otherwise specified. This value is ad-hoc, but we have not found our method to be particularly sensitive to its setting.

Starting iterates have a strong effect on most optimization methods, and ours is no different. Although it is common to initialize solvers with the unconstrained velocity at the start of the time-step we found the most robust method was to use the zero-velocity solution as a starting iterate, i.e.:

$$\mathbf{u}^0 = \mathbf{0} \tag{88}$$

$$\boldsymbol{\lambda}^0 = \mathbf{0}. \tag{89}$$

This choice is robust in the case of reinforcement-learning, where large random external torques are applied to bodies that can lead to initial points far from the solution. Warm-starting constraint forces is possible, and our tests indicate this can give a good improvement in efficiency, however due to the additional book-keeping we have not used warm-starting in our reported results.

### 4.7.2 Preconditioned Conjugate Residual

A key advantage of our non-smooth formulation is that it allows black-box linear solvers to be used for nonlinear complementarity problems. Nevertheless, the choice of solver is still an important factor that affects performance and robustness. A common issue we encountered is that even simple contact problems may lead to singular Newton systems. Consider for example a tessellated cylinder resting flat on the ground. In a typical simulator a ring of contact points would be created, leading to an under-determined system, i.e.: there are infinitely many possible contact force magnitudes that are valid solutions. This indeterminacy results in a singular Newton system and causes problems for many common linear solvers. The ideal numerical method should be insensitive to these poorly conditioned problems.

In addition, we seek a method that allows solving each Newton system only inexactly. Since

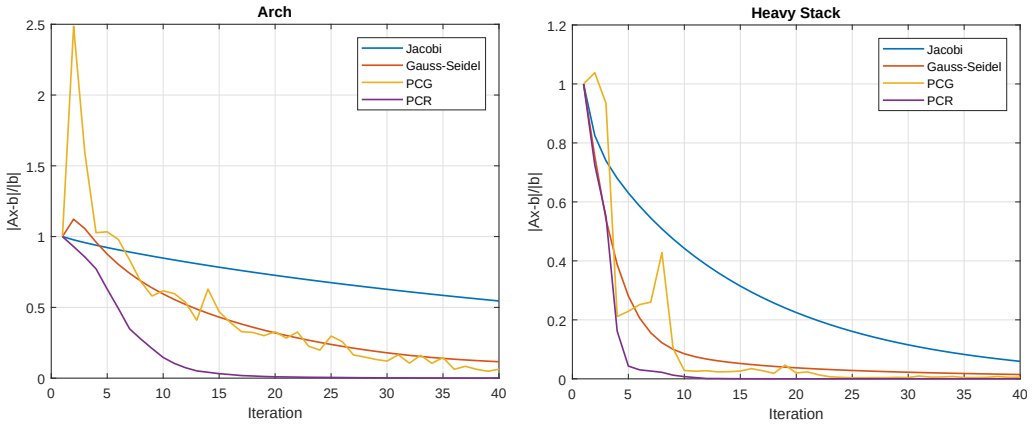



Figure 15: **Linear Solver Convergence**: Convergence of different iterative methods on a single linear sub-problem for two different examples. Preconditioned Conjugate Gradient (PCG) shows characteristic non-monotone behavior that causes problems with early termination. Preconditioned Conjugate Residual (PCR) is monotone, which ensures a useful result even at low iteration counts.

the linearized system is only an approximation it would be wasteful to solve it accurately far from the solution. Thus, another trait we seek is a method that smoothly and monotonically decreases the residual so that it can be terminated early e.g.: after a fixed computational time budget. Finally, since our applications often requires real-time updates we also look for a method that is amenable to parallelization.

The conjugate gradient (CG) method [88] is a popular method for solving linear systems in computer graphics. It is a Krylov space method that minimizes the quadratic energy $e = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$. One side effect of this structure is that the residual $r = |\mathbf{A}\mathbf{x} - \mathbf{b}|$ is not monotonically decreasing. This behavior leads to problems with early termination since, although a given iterate may be closer to the solution, the true residual may actually be larger. This manifests as constraint error changing unpredictably between iterations. A related method that does not suffer from this problem is the conjugate residual (CR) algorithm [177]. It is a Krylov space method similar to conjugate gradient (CG), however, unlike CG each CR iteration $k$ minimizes the residual

$$r_k = ||\mathbf{A}\mathbf{x} - \mathbf{b}||_{\mathcal{K}_k} \tag{90}$$

within the Krylov space $\mathcal{K}_k = \text{span}\{\mathbf{b}, \mathbf{A}\mathbf{b}, \mathbf{A}^2\mathbf{b}, \cdots, \mathbf{A}^{k-1}\mathbf{b}\}$. A remarkable side effect of this minimization property is that CR is monotonically decreasing in the residual norm $r = |\mathbf{A}\mathbf{x} - \mathbf{b}|$, and monotonically *increasing* in the solution variable norm [65]. These are exactly

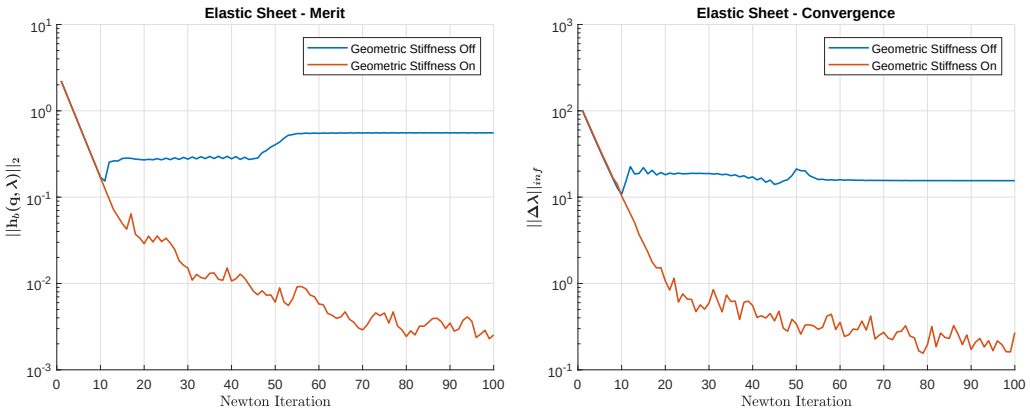



Figure 16: **Geometric Stiffness**: A plot of the merit function and convergence with and without our geometric stiffness approximation on the stretched elastic sheet example. With geometric stiffness disabled we observe overshooting and oscillating iterates at large strains.

the properties we would like for an inexact Newton method. From a computational viewpoint it is nearly identical to CG, requiring one matrix–vector multiply, and two vector inner products per-iteration. CR requires an additional vector multiply-add per-iteration, but each stage is fully parallelizable, and in our experience we did not observe any performance difference to standard CG.

In Figure 15 we compare the convergence of four common linear solvers, Jacobi, Gauss-Seidel, preconditioned conjugate gradient (PCG), and preconditioned conjugate residual (PCR). We use a diagonal preconditioner for both PCR and PCG. Our surprising finding is that PCR often has an order of magnitude lower residual for the same iteration count compared to other methods. A similar result was reported by Fong et al. [65] in the numerical optimization literature. For symmetric positive definite (SPD) systems CR will generate the same iterates as MINRES, however it does not handle semidefinite or indefinite problems in general. In practice we observed that CR will converge for close to singular contact systems when CG and even direct Cholesky solvers may fail. We evaluate and discuss the effect of linear solver on a number of test cases in Section 4.8.3.

### 4.7.3 Geometric Stiffness

The upper-left block of the system matrix, $\mathbf{H} = \mathbf{M} - \Delta t^2 \mathbf{K}$, consists of the mass matrix and a second term $\mathbf{K}$, referred to as the *geometric stiffness* of the system. This is defined as:

Table 2: **Statistics**: Parameters and statistics for different examples. All scenarios have used a time-step of $\Delta t = 0.0083$s. Contact counts are for a representative step of the simulation. Performance cost is typically dominated by the linear solver step, while the matrix assembly cost is small. We have reported solver times for a single time-step of our GPU-based PCR solver. We note that for small problems the performance of iterative methods is limited by fixed cost kernel launch overhead and that scaling with problem size is typically sub-linear up to available compute resources. For contacts we also report the maximum number of contacts in a single simulation island in brackets.

| **Example** | **Bodies** | **Joints** | **Tetra** | **Contacts** | **Newton Iters** | **Linear Iters** | **Assembly (ms)** | | **Solve (ms)** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | # | # | # | # (island) | # | # | Avg | Std. Dev. | Avg | Std. Dev. |
| Allegro DexNet | 21 | 21 | 0 | 104 (104) | 8 | 20 | 0.4 | 0.1 | 10.0 | 1.5 |
| Allegro Ball | 21 | 21 | 427 | 409 (409) | 6 | 50 | 0.35 | 0.05 | 11.4 | 1.2 |
| PneuNet | 1 | 0 | 2241 | 532 (532) | 4 | 80 | 0.3 | 0.05 | 22.25 | 1.4 |
| Fetch Tomato | 29 | 28 | 319 | 371 (371) | 4 | 50 | 0.45 | 0.05 | 12.2 | 1.25 |
| Fetch Beam | 42 | 42 | 0 | 300 (300) | 6 | 50 | 0.2 | 0.1 | 18.6 | 2.25 |
| Arch | 20 | 0 | 0 | 134 (134) | 6 | 20 | 0.15 | 0.05 | 7.75 | 1.4 |
| Table Pile | 54 | 0 | 0 | 936 (736) | 4 | 20 | 0.1 | 0.05 | 3.7 | 0.95 |
| Humanoid Run | 5200 | 4800 | 0 | 10893 (33) | 4 | 10 | 0.8 | 0.05 | 8.9 | 0.25 |
| Yumi Cabinet | 6400 | 6600 | 0 | 4082 (55) | 5 | 25 | 1.8 | 0.25 | 56.0 | 6.4 |

$$\mathbf{K} = \frac{\partial}{\partial \mathbf{u}}\left(\mathbf{J}^T \boldsymbol{\lambda}_i\right). \tag{91}$$

$\mathbf{K}$ is not a physical quantity in the sense that it does not appear in the continuous or discrete equations of motion. It appears only as a side-effect of the numerical method, in this case the linearization performed at each Newton iteration. In practice $\mathbf{K}$ is often dropped from the system matrix, leading to a quasi-Newton method. Tournier et al. [215] observed that ignoring geometric stiffness can lead to instability, as the Newton search direction is free to move the iterate away from the constraint manifold in the transverse direction, eventually causing the iteration to diverge. However, including $\mathbf{K}$ directly is also problematic because the mass block is no longer simple block-diagonal, and in addition, the possibly indefinite constraint Hessian restricts the numerical methods that can be applied. Inspired by the work of Andrews et al. [7] we introduce an approximation of geometric stiffness that improves the robustness of our Newton method without these drawbacks.

We now present a method for approximating $\mathbf{H}$ that is related to the method of Broyden [28]. This method builds the Jacobian iteratively through successive finite differences. We apply this idea to build a diagonal approximation to the geometric stiffness matrix. Using the last two Newton iterates, we can define the following first-order approximation:

$$\mathbf{H}\Delta\mathbf{u} = \left(\frac{\partial \mathbf{g}}{\partial \mathbf{u}}\right)\Delta\mathbf{u} \approx \mathbf{g}(\mathbf{u}^{n+1}) - \mathbf{g}(\mathbf{u}^n) \tag{92}$$

where $\mathbf{H}$ is the unknown matrix we wish to find, and $\Delta\mathbf{u} = \mathbf{u}^{n+1} - \mathbf{u}^n$ is the known difference between the last two iterates. This problem is under-determined since $\mathbf{H}$ has $n_d \times n_d$ entries, while (92) only provides $n_d$ equations, however if we assume $\mathbf{H}$ has the special form:

$$\tilde{\mathbf{H}} \approx \mathbf{M} - \tilde{\mathbf{K}} \tag{93}$$

where $\tilde{\mathbf{K}} = \text{diag}[c_1, c_2, \cdots, c_n]$ is a diagonal approximation to $\mathbf{K}$, then the individual entries of $\tilde{\mathbf{K}}$ are easily determined from (92) by examining each entry:

$$c_k = -\left[\frac{\mathbf{g}(\mathbf{u}^{n+1})_k - \mathbf{g}(\mathbf{u}^n)_k + (\mathbf{M}\Delta\mathbf{u})_k}{\Delta\mathbf{u}_k}\right]. \tag{94}$$

Each Newton iteration we then update the mass block's diagonal entries to form our approximate $\tilde{\mathbf{H}}$ as

$$\tilde{\mathbf{H}}_{ii} = \tilde{\mathbf{M}}_{ii} - \min(0, c_k). \tag{95}$$

For most models $\mathbf{M}$ is block-diagonal, e.g.: 3x3 blocks for rigid body inertias, and so $\mathbf{H}^{-1}$ is may be easily computed to form the Schur complement. To ensure $\mathbf{H}$ remains positive-definite we clamp the shift to be positive. (80). This diagonal approximation is inspired by the method presented by Andrews et al. [7] however this approach has several advantages. First, we do not have to derive or explicitly evaluate the constraint Hessians for different constraint types. This means our method works automatically for contacts and complex deformable models. Second, we do not need to track the Lagrange multipliers from the previous frame, as our method updates the geometric stiffness at each Newton iteration. Finally, since our method does not change the solution to the problem it does not introduce any additional damping provided the solver is run to sufficient convergence. Our approach bears considerable resemblance to a L-BFGS method [156], however the use of a diagonal update means the system matrix structure is constant, allowing us to use the efficient block-diagonal inverse for the Schur complement.

### 4.7.4 Hyperelastic Materials

Deformable bodies with linear material models may exhibit significant volume loss during large deformations, which can lead to element inversion. Hyperelastic constitutive models, where

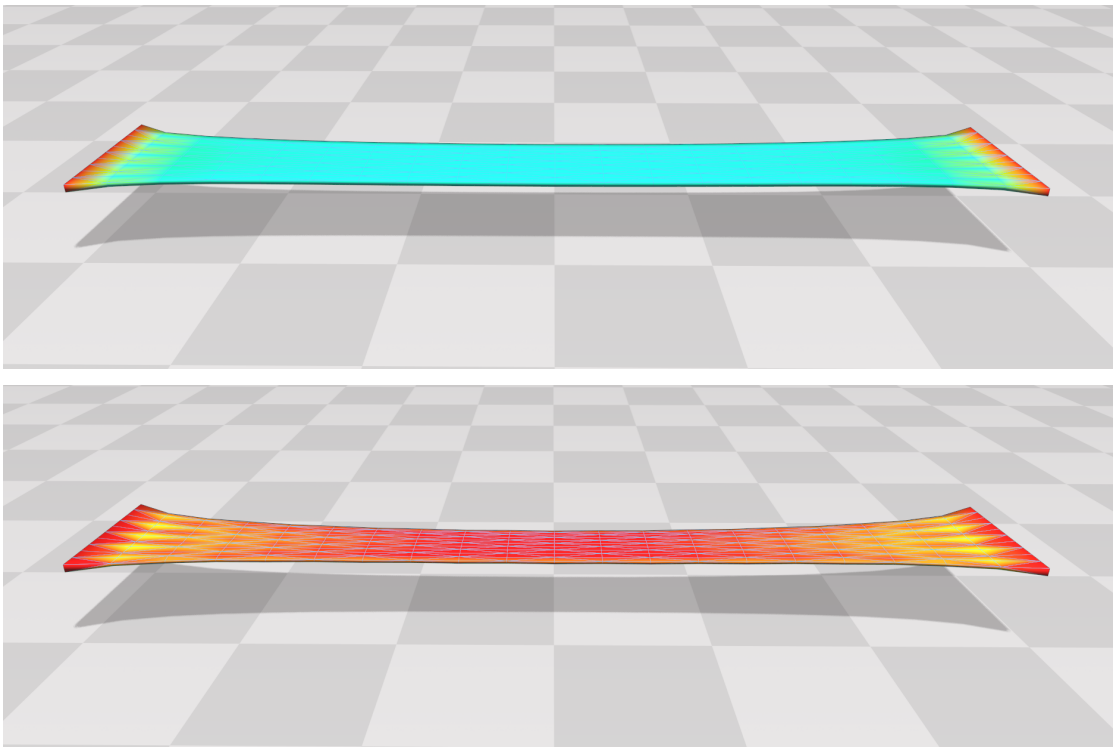


Figure 17: **Material Extension Test**: Our generalized compliance formulation accommodates hyperelastic material models that strongly resist volume changes (top), while linear models show characteristic volume loss at large strains (bottom).

stiffness increases as a function of strain, are less prone to this artifact, as illustrated in Figure 17. However, unlike the Hookean material presented in Section 2, for hyperelastic materials the strain energy density is no longer a simple quadratic form. We now show how to incorporate hyperelastic constitutive models in our constrained dynamics solver. Given a strain energy density function parameterized in terms of principal stretches $\mathbf{s}(\mathbf{q}) = [s_1, s_2, s_3]$ assumed constant over the element, the elastic potential energy is

$$U(\mathbf{q}) = V_e \Psi(\mathbf{s}(\mathbf{q})), \tag{96}$$

and the resulting force on the system is

$$\mathbf{f} = -\frac{\partial U}{\partial \mathbf{q}}^T = -\left(\frac{\partial U}{\partial \mathbf{s}}\frac{\partial \mathbf{s}}{\partial \mathbf{q}}\right)^T. \tag{97}$$

To obtain the compliance form of this force we factorize it as $\mathbf{f} = \mathbf{J}^T\boldsymbol{\lambda}$, where $\mathbf{J} = \frac{\partial \mathbf{s}}{\partial \mathbf{q}}$ and $\boldsymbol{\lambda} = -\frac{\partial U}{\partial \mathbf{s}}^T$. As before, $\boldsymbol{\lambda}$ is a variable introduced to the system, with the associated constraint:

$$\boldsymbol{\phi}_b \equiv \frac{\partial U}{\partial \mathbf{s}}^T + \boldsymbol{\lambda} = 0. \tag{98}$$

Initially this factorization may not appear to have achieved a great deal. However, when we consider the linearization of this constraint we have

$$\frac{\partial^2 U}{\partial \mathbf{s}^2}\mathbf{J}\Delta\mathbf{q} + \mathbf{I}\Delta\boldsymbol{\lambda} = -\boldsymbol{\phi}_b, \tag{99}$$

from which we can identify the compliance matrix as

$$\mathbf{E}(\mathbf{q}) = \left(\frac{\partial^2 U}{\partial \mathbf{s}^2}\right)^{-1} \in \mathbb{R}^{3\times 3}. \tag{100}$$

Pre-multiplying by this matrix we obtain the familiar form below, where the compliance matrix appears on the diagonal.

$$\mathbf{J}\Delta\mathbf{q} + \mathbf{E}\Delta\boldsymbol{\lambda} = -\mathbf{E}\boldsymbol{\phi}_b. \tag{101}$$

The effect of this transformation is to factor out the component containing the material parameters that is potentially ill-conditioned, and to invert it directly. Unlike linear materials the compliance matrix is no longer a constant, but is a function of the system state $\mathbf{q}$, and so $\mathbf{E}$ must be computed for each Newton iteration. Another practical consideration is that, as $\mathbf{E}$ is the inverse of a Hessian it may be indefinite, in which case it may be necessary to project it back to the positive definite cone before including it in our system matrices [207]. We have also found that a simple diagonal approximation to $\mathbf{E}$ is often sufficient. In Appendix B we give the derivation of $\mathbf{E} = (\frac{\partial^2 U}{\partial \mathbf{s}^2})^{-1}$ for the stable Neo-Hookean model presented by Smith et al [193]. To the best of our knowledge this derivation has not been presented previously in the literature.

## 4.8 Analysis

### 4.8.1 Effect of Complementarity Preconditiong

In Figure 22 we look at the distribution of convergence behavior over 132 simulation steps during the flexible beam insertion scenario in Figure 10. We use our proposed complementarity preconditioner with 40 PCR iterations per-Newton iteration and observe super-linear convergence in significantly more steps using our preconditioning strategy than with identity scaling. In addition, the median error with our strategy is often an order of magnitude lower for the same iteration count. We use the rigid body contact test scenarios to evaluate the effectiveness of our complementarity preconditioner, and compare the convergence of different strategies in Figure 18. We observe that identity scaling with $r = 1$ may fail badly when there are large masses, or large contact sliding velocities. Constant global scaling by $r = \Delta t^2$ improves convergence, but still suffers from problems with large mass-ratios. The combination of time-step and effective mass leads to the most reliable observed convergence. Based on the poor performance of identity scaling we believe that a preconditioning strategy such as ours is a necessity to make such non-smooth formulations practical.

### 4.8.2 Effect of NCP-Function

We found that although both the minimum-map and Fischer- Burmeister functions can achieve high accuracy given enough iterations, the minimum-map tends to produce noisy results where the contact forces are not distributed evenly around a contact area. This is primarily a problem

when the contact set is redundant, and a small change in the problem may lead to a large change in the active-set. We found the Fischer-Burmeister function was less sensitive to this problem, and would produce smoother contact force distributions even for redundant contact sets. We suspect this is because the minimum-map has many non-smooth points while Fischer-Burmeister has only one, however further investigation to verify this is needed. Due to the improved stability in interactive environments we have used the Fischer-Burmeister function for all examples unless otherwise stated. For scenarios where force distributions are critical, e.g.: force feedback based controllers, it may be appropriate to use a combination of warm-starting to provide temporal coherence, and redistribution of contact forces as a post-process after the contact solve, as proposed by Zheng & James [238]. Yet another option is to change the contact model itself by introducing compliance. This makes the problem well-posed by allowing some interpenetration, and may be supported in our formulation by augmenting the compliance block on the contact constraints. We further investigate the effect of compliance on force distributions in Chapter 5.

### 4.8.3 Effect of Linear Solver

In Figure 19 we compare convergence for different linear solvers over the course of a Newton solve during a single time-step. For performance sensitive applications it is typically not practical or desirable to run each Newton solve to convergence, so for this test we fix the number of linear solver iterations per-step to 40. This early termination is generally no problem for relaxation and PCR methods, but it can cause problems for solvers like PCG which decrease the error non-monotonically, leading to erratic convergence. In Figure 15 we plot the behavior of each iterative method on a single linear subproblem.

The convergence of Jacobi and Gauss-Seidel with our contact formulation is consistent with the behavior observed in other engines such as Bullet [38] and XPBD [138]. While relaxation methods perform quite well for reasonably well-conditioned problems, they are very slow to converge for situations involving high mass-ratios as shown in Figure 19. This is highlighted in the heavy-stack example which shows catastrophic interpenetration.

### 4.8.4 Error Analysis

To better understand our results we perform an error analysis to establish a baseline accuracy limit given finite precision floating point. Our analysis is based on that given by Tisseur [209] who show that Newton's method applied to the problem of solving $\mathbf{r}(\mathbf{x}) = \mathbf{0}$ will have a limiting step-size, or solution accuracy of

$$\|\Delta \mathbf{x}\| \approx \|\mathbf{A}_*^{-1}\|\nu + \|\mathbf{x}_*\|\delta. \tag{102}$$

Here $\mathbf{x}_*$ is the true solution, $\mathbf{A}_*$ is the system Jacobian at the solution, $\nu$ is an upper bound on the residual error, and $\delta$ is the machine epsilon. In general we do not know the true solution $\mathbf{x}_*$ so we use the lowest error solution as an approximation. Likewise we use $\nu = \delta\|\mathbf{r}\|$ as the residual error bound, where $\mathbf{r}$ is the measured residual at end of the Newton solve. As with the solution variables, the lowest achievable residual is also limited by available precision. Tisseur show that the predicted minimum residual bound is

$$\|\mathbf{r}\| \approx \|\mathbf{A}_*\|\|\mathbf{x}_*\|\delta. \tag{103}$$

We use 32-bit floating point for all calculations which has a machine epsilon of $\delta \approx 10^{-7}$, and we plot the limiting values for the $L_\infty$ norm as dashed lines in Figures 18-19. We find that our method comes close to achieving the predicted limiting accuracy in most cases, and in some cases surpasses it, as in the Flexible Beam example.

## 4.9 Results

We implemented our algorithm in CUDA and run it on an NVIDIA GTX 1070 GPU. The assembly of the system matrix is performed on the GPU in compressed sparse row (CSR) format for maximum flexibility. We build $\mathbf{H}, \mathbf{M}, \mathbf{J}$, and $\mathbf{C}$ separately and perform the matrix multiply operations necessary for Krylov methods through successive multiplications of individual matrices. Collision detection is performed between triangle-mesh features once per-time step using the system's unconstrained velocity to generate candidate pairs. We define contact normals as the normalized vector between each feature pair's closest points using the configuration at the start of the time-step and note that constraint manifold refinement (CMR) could be used to further improve robustness [159].

We now discuss the experimental test scenarios we have used to evaluate our method. We report scene statistics and performance numbers in Table 2. When running in an interactive setting we use a display rate of 60hz. For robust collision detection we use two simulation time-steps per visual frame, each with $\Delta t = 0.0083$s. If the simulation computation takes longer than this the effect for the user is a slightly slower than real-time update rate.

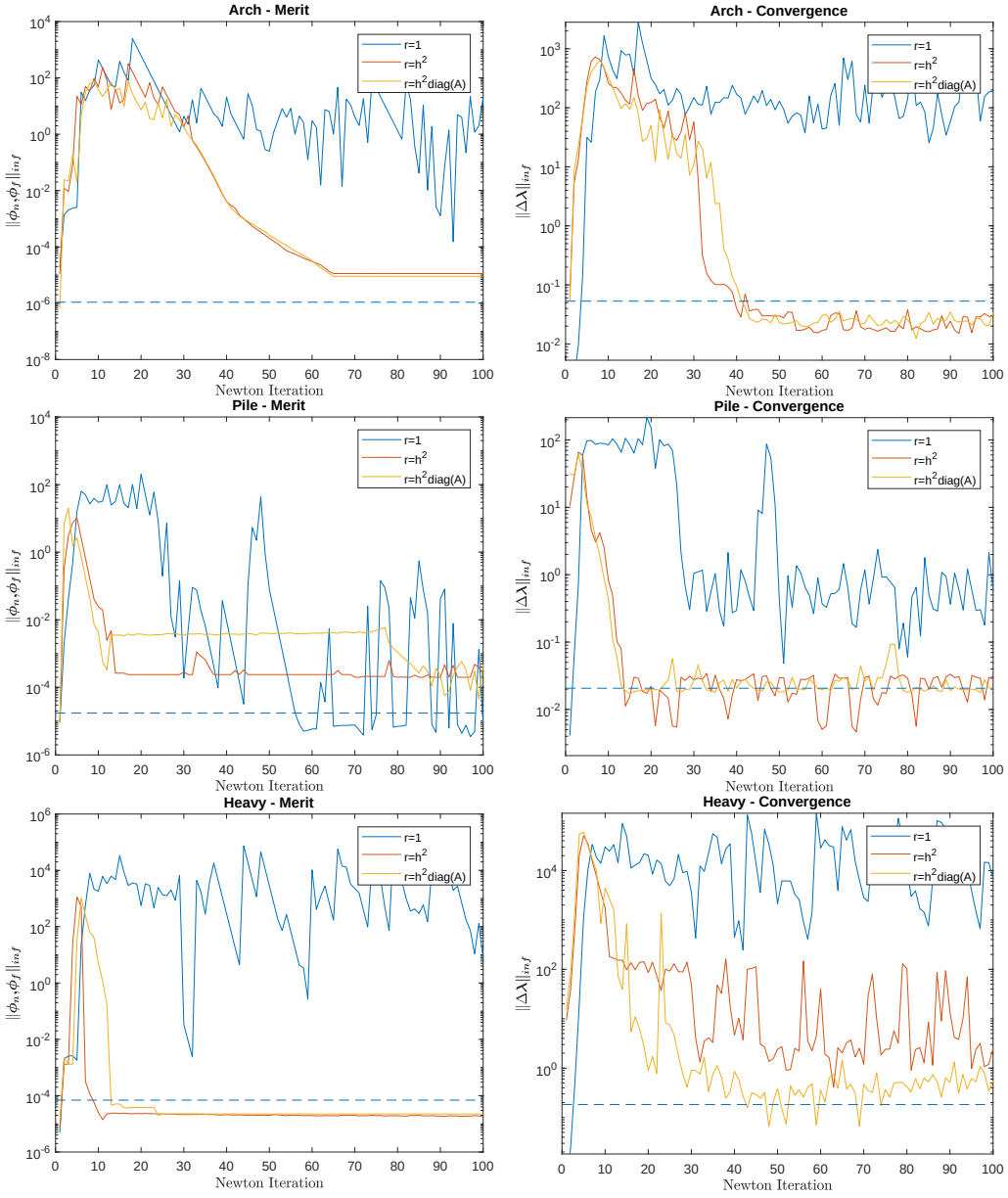



Figure 18: **Effect of NCP Preconditioner**: An evaluation of different complementarity preconditioning strategies on three contact problems. We plot the maximum complementarity error for normal and frictional forces (upper row), and the step size (lower row) for each Newton iteration over a single time-step. Identity scaling (blue) often fails to converge. A time-step scaled strategy with (red) performs well when mass-ratios are small, such as in the arch and pile scenes (left, middle). For situations with high-mass ratios (right) we also take into account the system's effective mass (yellow).

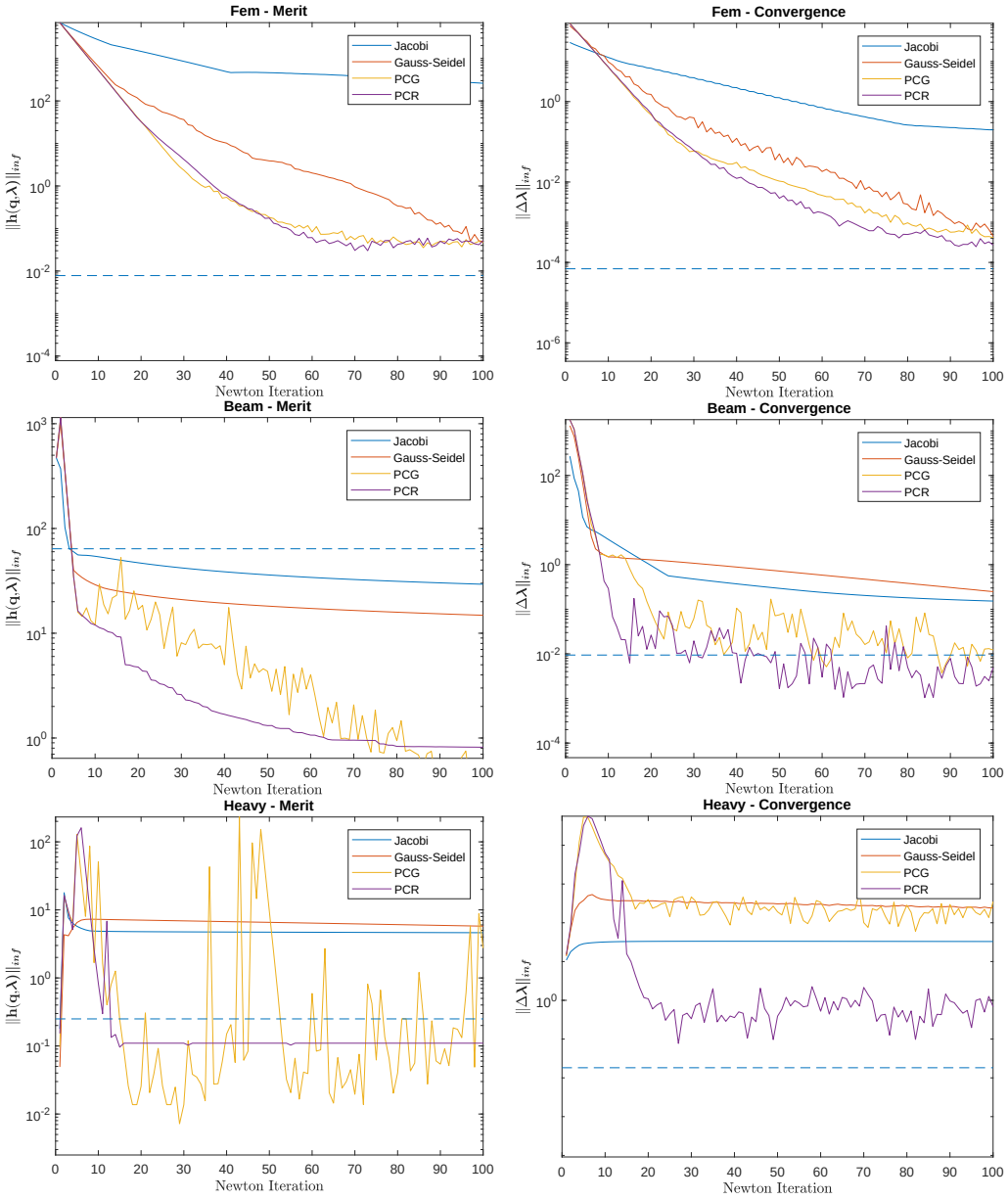



Figure 19: **Effect of Linear Solver**: A comparison of different iterative methods on three test cases. We plot the maximum constraint error (upper row) and the step size (lower row) for each Newton iteration over a single time-step. Jacobi and Gauss-Seidel methods show characteristic stagnation for ill-conditioned problems (blue, red). Conjugate Gradient (yellow) is less sensitive to conditioning but has large residual fluctuations. Conjugate Residual (purple) provides fast convergence and smooth error reduction. The dashed lines represent the limiting accuracy and residual bounds predicted by an error analysis assuming 32-bit floating point.

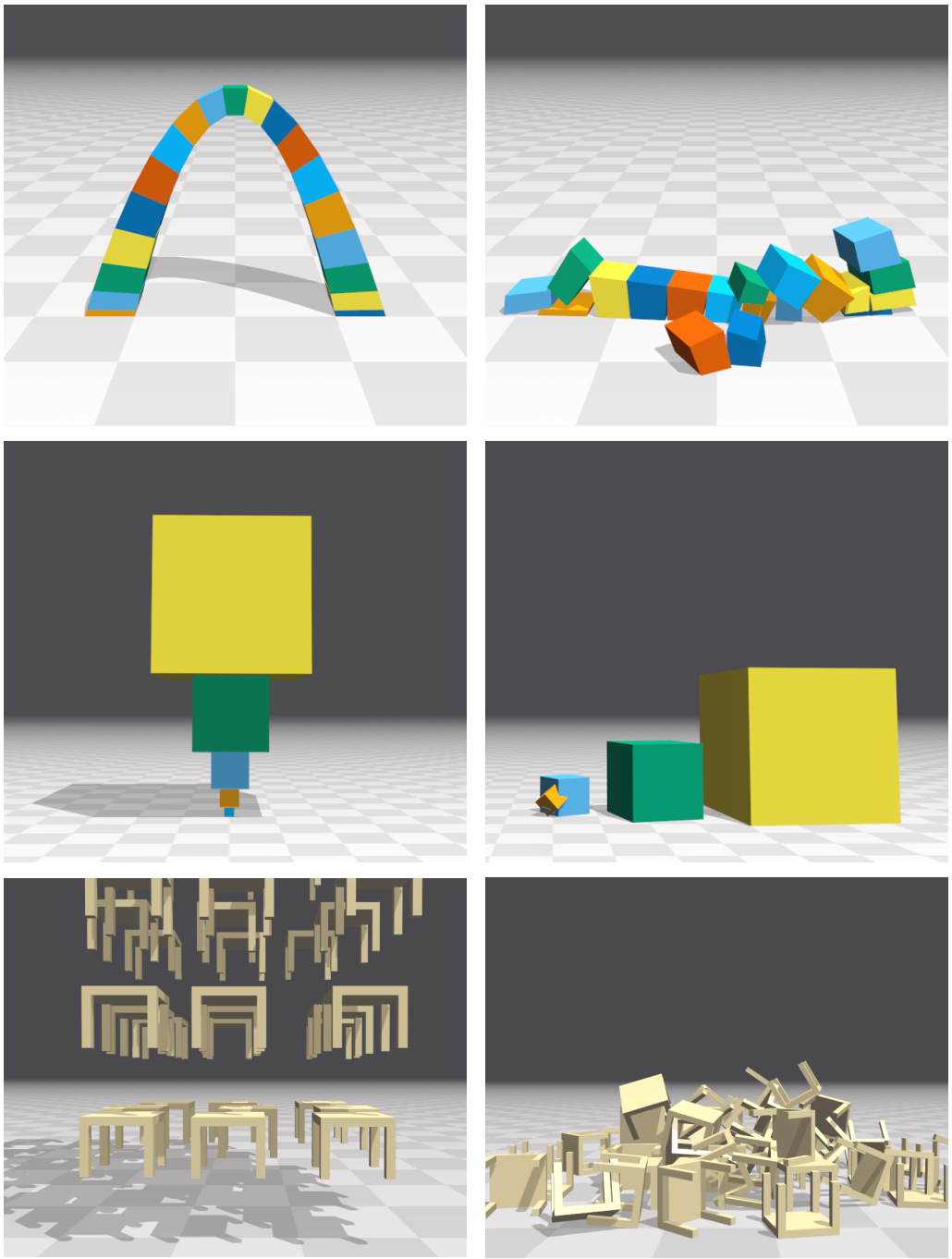


Figure 20: **Contact Examples**: Top: A self-supporting parabolic arch. Direct or Krylov linear solvers can reduce the error sufficiently to form stable structures without drift (left). Relaxation methods like Jacobi and Gauss-Seidel eventually collapse even with hundreds of iterations (right). Middle: A stack of increasingly heavy boxes with a total mass ratio of 4096:1, such poorly conditioned problems are difficult for relaxation methods, leading to significant interpenetration (right). Bottom: An unstructured piling test, our method captures stick/slip transitions and forms stable piles of non-convex objects.

**Fetch Tomato** We test our method on a pick-and-place task using the Fetch robot as shown in the title figure. The robot consists of rigid bodies connected by joints as defined by the Unified Robot Description Format (URDF) file. The tomato is modeled using tetrahedral FEM with Young's modulus of $Y = 0.1$MPa, Poisson's ratio of $\nu = 0.45$, density of $\rho = 1000\text{kg}/\text{m}^3$, and a coefficient of friction of $\mu = 0.75$ between the grippers and the tomato. The robot is controlled by a human operator who directs the arm and the grippers to grasp the object and transfer it to the mechanical scales. The scales are modeled through rigid bodies connected by prismatic and revolute joints that drive the needle and accurately reads the weight of the tomato. The most challenging part of this scenario is the grasp and transfer of the tomato, which requires tight coupling between frictional contact and the internal dynamics of the tomato. Our method forms stable grasps while undergoing large translational and rotational motion.

**Fetch Beam** We test our method on a flexible beam insertion task by modeling the beam as a series of connected rigid bodies with finite bending stiffness as shown in Figure 10. The lightweight beam is modeled by 16 rigid bodies each with mass of $m = 0.003$kg, and connected through joints with a bending stiffness of 250N.m. This example highlights the limitations of traditional relaxation-based approaches that cannot achieve the desired stiffness on the beam's joints even with hundreds of iterations as shown in the convergence plot of Figure 19.

**DexNet** We evaluate our method on the problem of dexterous grasping using the DexNet adversarial object database [139]. These models are highly irregular with many concave areas that make forming stable grasps difficult. The underlying triangle meshes are high resolution and non-manifold which tends to generate many redundant contacts, a challenge for most complementarity solvers. We use the Allegro hand with a coefficient of friction $\mu = 0.8$ to perform grasping using a human control interface, and find stable grasps for the objects in collection as shown in Figure 14.

**PneuNet** To test coupling between deformable and rigid bodies we simulate a three-fingered gripper based on the PneuNet design [97]. We model the deformable finger using tetrahedral FEM with a linear isotropic material model and parameters for silicone rubber of $Y = 0.01\text{GPa}, \nu = 0.47, \rho = 1200\text{kg}/\text{m}^3$. To model inflation we use an activation function that uniformly adds an internal volumetric stress to each tetrahedra in the finger arches. We do not model the chamber cavity explicitly, however we found this simple activation model was sufficient to reproduce the characteristic curvature of the gripper. We observe robust coupling through contact by picking up a ball with mass $m = 0.32$kg, using a friction coefficient of $\mu = 0.7$ as shown in Figure 2.

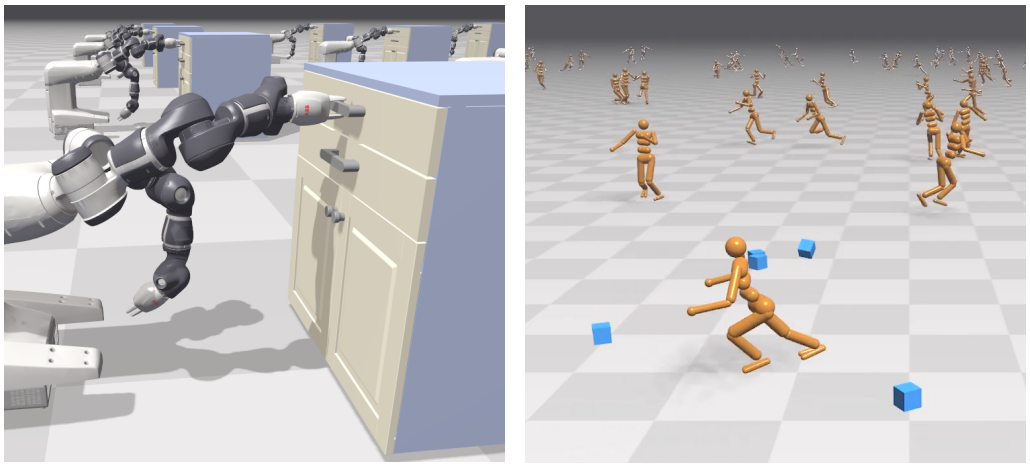


Figure 21: **Reinforcement Learning**: Left: The Yumi robot trained to open a cabinet drawer. The network learns a policy that reaches the handle, performs a grasp, and opens it through frictional forces from the fingers alone. Right: RL locomotion example based on the OpenAI Roboschool Humanoid Flagrun Harder environment. Using our simulator the network learns a robust policy that takes advantage of stick-slip transitions to change direction quickly, and recover from external disturbances.

**Rigid Body Contact** We test our method on a variety of rigid-body contact problems as illustrated in Figure 20. Our method achieves stable configurations for difficult problems including self-supported structures, and stacks with extreme mass ratios. For the self-supported arch we use $\mu = 0.6$ with masses in the range $m = [15, \cdots, 110]$. For the heavy stack of boxes we use $\mu = 0.5$, with masses that increase geometrically as $m = [8, 64, \cdots, 32768]$kg. For the table piling scene each table has a mass of $m = 4.7$kg with $\mu = 0.7$. Particularly on the scenes with high-mass ratios we observe that relaxation methods struggle to reduce error, while Krylov methods form stable structures and successfully prevent interpenetration.

**Material Extension** We perform a material extension test and compare the behavior between a linear co-rotational model and the hyperelastic model of Smith et al. [193] with Young's modulus set to $E = 10^5$Pa and Poisson's ratio of $\nu = 0.45$. We visualize volumetric strain and observe high volume loss for the linear model as shown in Figure 17. We also found geometric stiffness to be essential to obtain a stable simulation when strains are large. This is illustrated in in Figure 16 which shows the iterate oscillating around the solution.

**Reinforcement Learning** Reinforcement learning (RL) is a good test of robustness for a simulator since it generates many random inputs in the form of forces, torques, and constraint configurations. We apply our simulator to two scenarios using reinforcement learning. The first is the

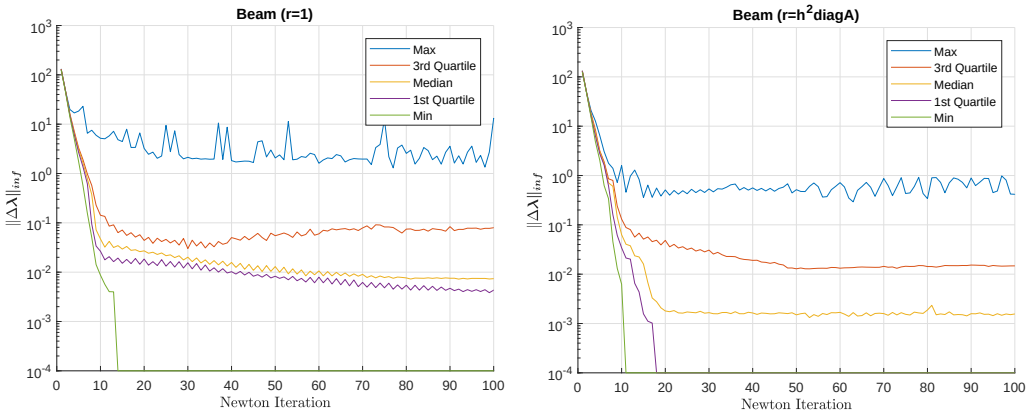



Figure 22: **Convergence Analysis**: Quartile analysis of the flexible beam-insertion example with identity scaling (left), and our proposed NCP preconditioning strategy (right). We plot statistics for 100 Newton iterations taken over 132 simulation-steps. Using our preconditioner we observe super-linear convergence and lower final residual for significantly more cases than an identity scaling.

problem of training the ABB Yumi robot to grasp a cabinet handle and open a drawer as shown in Figure 21 (left). We use the PPO algorithm [182] to train the network and find it quickly (less than 100 training iterations) learns a policy to extend, grasp and open the drawer through implicit PD controls applied through the joints. Our second RL example is the Humanoid Flagrun Harder scene adapted from the OpenAI Roboschool [158]. In this task a humanoid model must learn to stand up and run in a randomly assigned direction that changes periodically as shown in Figure 21 (right). The learned actions are torques, applied as external forces. Using our simulator we are able to achieve good running results and note the agent taking advantage of stick-slip transitions on the feet during fast turns.

## 4.10 Limitations

Krylov methods such as PCG and PCR use a globally optimal line-search step at each iteration. This has the sometimes unexpected effect that error in one row can affect the rate of residual reduction in other rows, leading to slower than necessary convergence for independent parts of the simulation. A natural solution would be to include an island-detection step and solve these independent systems separately.

We found our geometric stiffness approximation to be effective on particle-based objects, but less so on rigid-articulated mechanisms, and in the worst case can cause some jitter at joint limits. To avoid this we apply geometric stiffness only to the particle degrees of freedom. In the future we plan to explore more advanced quasi-Newton methods e.g.: those based on symmetric

rank-1 (SR1) updates.

Our method's primary computational cost is the solution of a symmetric linear system at each step. All the iterative solvers considered here may be implemented in a matrix-free way which would likely improve performance. Because the linear solver is treated as a black box it would be interesting to apply more advanced methods, such as algebraic multi-grid (AMG), to accelerate convergence for larger-scale linear subproblems.

We do not have a convergence proof for the fixed-point iteration used to update the frictional slack variables, and it is possible that better choices exist. Similarly, the design space for choosing the $r$ factor in the complementarity preconditioner is large, and we think heuristics based on global information may provide significant performance improvements. We think these two areas are rich for further research.

## 4.11 Conclusion and Future Work

We have presented a GPU framework for multi-body dynamics that allows off-the-shelf linear solvers to be used for rigid and deformable contact problems. We evaluated our method on a variety of scenarios in robotics and reinforcement learning and found that it performs well on grasping and dexterous manipulation of deformable objects, as well as ill-conditioned rigid body contact problems. We believe our complementarity preconditioner makes non-smooth formulations of contact practical for interactive applications for the first time, and hope that this opens the door for future works to apply new contact models, preconditioners, and linear solver methods to multi-body problems.

# 5 Primal/Dual Descent Methods for Dynamics

In the previous section we presented a numerical method that performs implicit time-integration by solving for the roots of a nonlinear system of equations. An alternative to root finding is to view implicit time-integration as the solution to an energy minimization problem. These *variational* methods are popular in elasticity simulation due to their robustness and efficiency. One such method is Projective Dynamics (PD) [22], in this method forces are considered to arise from an energy potential, and minimized using a local/global optimization over the *primal* variables, namely position and velocity. In contrast, constraint-based methods such as extended Position-based Dynamics (XPBD) [138] are typically solved in terms of *dual* variables, i.e.: the Lagrange multipliers of each constraint.

In this chapter, we show that these primal and dual methods may both be derived from a common variational basis. This provides us with an insight into their relationship, and allows us to perform a sensitivity analysis to identify potential sources of ill-conditioning. Based on this analysis, we perform numerical experiments and show that these methods offer complementary trade-offs in terms of sensitivity to mass ratios and stiffness ratios. In addition, while primal descent methods have been used successfully for elasticity simulation [224, 225], they have not found wide-spread use in contact-rich scenarios, such as rigid body simulation. We extend these methods to dry frictional contact by deriving a Coulomb friction model from a variational basis, and provide an efficient preconditioner suitable for parallelization. Our primal formulation of contact possesses a number of desirable traits. First, it is differentiable, and has well-defined inverse dynamics, an important property for trajectory optimization [212]. Second, it is insensitive to mass-ratios, allowing it to stably simulate scenarios involving both small and large bodies, as shown in Figure 23. Third, it does not require the tracking of auxiliary variables such as Lagrange multipliers. This means that the system size remains constant over the course of a simulation, which may be desirable if using the system state as an input to e.g.: a neural network controller [162]. Finally, we find that the force distributions obtained from relaxed, or compliant contact models, are smoother than the result from hard-contact models, something that is desirable when contact forces act as input to control algorithms. In summary, our technical contributions are as follows:

- A derivation of primal and dual formulations of contact dynamics from a common variational basis. Our derivation shows the underlying connection between methods such as Projective Dynamics and XPBD, and extends the Projective Dynamics-based method of [224] to rigid body simulation.
- A primal frictional contact model derived from a variational basis. Our model is simple

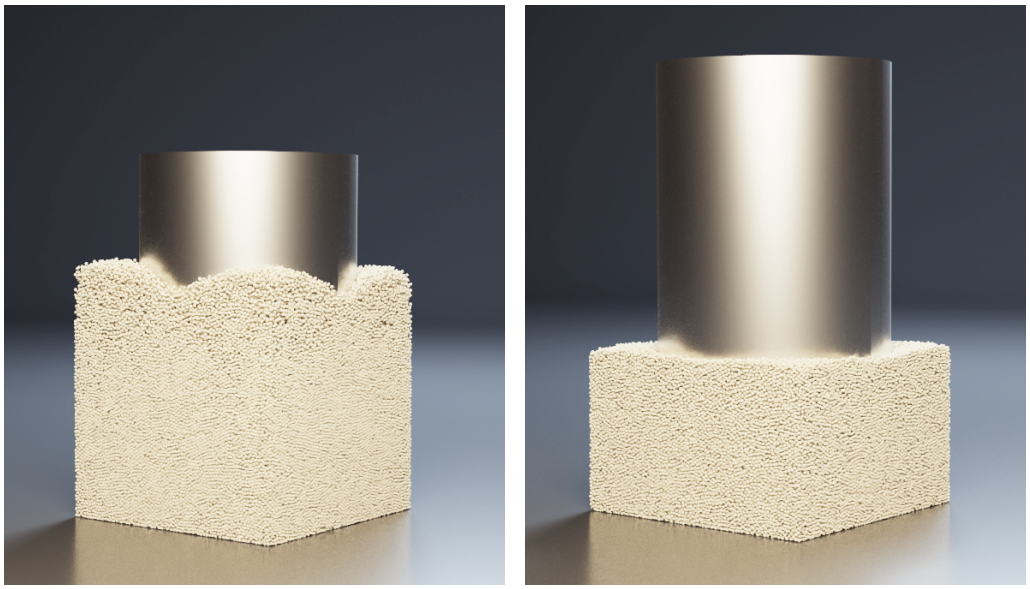


Figure 23: **Granular Material**. In this example the granular medium consists of 256k rigid bodies with an average radius of 5mm. The resulting mass ratio between the grains and cylinder is $80000 : 1$, which results in an ill-conditioned system for dual, or constraint-based solvers (left). Primal formulations on the other hand are relatively unaffected by this ratio (right).

to implement, supports differentiability, and is well-suited to GPUs.

- A numerical analysis of the sensitivity of both primal and dual descent methods to ill-conditioned problems. We test the relative strengths / weaknesses of each method in the presence of large mass ratios, and identify the lesser known problem of stiffness ratios.
- An experimental comparison of both optimization methods on a number of scenarios including unstructured piling, robotic grasping, cloth simulation, and trajectory optimization.

## 5.1 Related Work

We now discuss some previous simulation work in both the primal and dual space with a focus on elasticity and contact simulation.

### 5.1.1 Elasticity

There has been a large amount of work on implicit simulation of elastic bodies in computer graphics. Baraff & Witkin [12] proposed a single-step Newton method for solving the discrete

implicit equations of motion in cloth simulation. They use a linearization of forces that results in significant artificial damping, and has led to the development of a number of nonlinear implicit solvers. Many of these are derived from variational principles, which formulate the problem as one of energy minimization [71, 144]. In particular, Projective Dynamics (PD) [22, 131], proposes a splitting-based method where elastic potentials are handled through a local projection, and inertial potentials are handled through a global step. Wang et al. [224, 225] showed that descent methods applied to variational implicit Euler may be considered as a special case of Projective Dynamics where the local and global solve are performed using one preconditioned Jacobi iteration. In this work we also focus on preconditioned descent methods for their simplicity and simple GPU parallelization. We extend the work of Wang et al. to include rigid bodies, and propose a contact model based on a variational energy function that correctly models Coulomb friction.

Constraint-based, or dual methods are also popular for elastic and multi-body dynamics. Servin et al. [185] formulated a linear finite element method (FEM) as a set of compliant constraints. This approach was made robust by the inclusion of geometric stiffness terms that include second order constraint information [215, 7]. Goldenthal et al. [74] proposed a fast constraint projection method using direct solvers, while Position-based Dynamics employs iterative, local, nonlinear constraint projections on the same constraint-based formulation [151, 197]. This approach was extended to correctly handle quadratic energy potentials in XPBD [138], and we show how these methods may be seen as solving a dual variational problem, which we derive in Section 5.2.3.

### 5.1.2 Contact

Penalty methods are a common primal model of contact in graphics and robotics [143, 234, 51, 205]. While explicit penalty methods tend to require small time-steps for stability, *implicit* penalty methods have been used successfully [232]. A notable example is in MuJuCo [213, 212], which uses a relaxed contact model inside an acceleration-based framework with direct solver methods. We present a primal contact model inspired by this work and combine it with a descent-based solver well-suited for GPU implementations and capable of scaling to hundreds of thousands of simultaneous contacts. While relaxed methods of contact generally permit some interpenetration or slip, we find this is not a significant limitation, as illustrated in Figure 25.

Pan et al. [162] used a smooth contact model for differentiable trajectory optimization. They propose a viscous friction model that, does not make a distinction between stick and slip regimes. In this work we propose a relaxed friction model that also captures the Coulomb constraint that friction forces should lie inside the friction cone. Hybrid methods for contact have also been

Figure 24: **Unstructured Piling**. A sequence of frames from a large-scale piling example inspired by Xu & Barbic [232]. Despite having 393k (40x more) contacts, our parallel preconditioned gradient descent solver runs at real time rates.

used successfully. An example of this is the work by Tang et al. [206], who used the augmented Lagrangian method (ALM) to resolve cloth self-collisions. Kaufman et al. [108] proposed staggered projections for frictional contact where interpenetration is prevented through interleaved quadratic programming (QP) solves. Their resulting optimization problem may be solved using primal or dual methods. Mazhar et al. [147] compared primal and dual formulations of contact modeled as hard constraints and their effect on solution methods.

Simultaneous to our work, Li et al. [119] proposed a lagged primal formulation of contact that uses barrier methods to guarantee interpenetration free states. While barrier methods prevent touching contact, we focus on implicit penalty methods, which permit some interpenetration. We propose a similar smoothed friction model derived from a variational dissipation potential and compare primal and dual methods for solving the resulting optimization problem. Recent work has extended Projective Dynamics to handle nodal frictional contact for cloth and thin objects [136, 42]. Since our method is based on the descent-based form of Projective Dynamics [225] it is not limited to nodal contact. Brown et al. [27] presented a non-smooth dissipation potential to model friction in the Projective Dynamics framework, we present a smooth extension of this model that has continuous derivatives.

As discussed in Section 4., implicit time-stepping schemes for rigid body contact often use a linear complementarity (LCP) formulation. This model has become popular in graphics, especially when combined with iterative LCP solvers such as projected Gauss-Seidel (PGS) and projected Jacobi [159, 43, 57, 15, 154, 214, 221]. Dual formulations of contact naturally handle hard contact and static friction constraints. On the other hand, hard models of contact may become overdetermined through incompatible contact constraints, for example a body being squeezed between two immovable objects. In this case numerical methods may return an arbitrary answer, or in the case of direct methods may fail to produce any answer at all. Hard contact problems may also become underdetermined, for example a tessellated object resting on the ground with multiple contact points. In this case there exist many possible solutions, and the result will typically depend on the constraint order given to the solver. In contrast, our relaxed primal model of contact generates well-posed problems that result in smooth contact force distributions as we demonstrate in Figure 33.

## 5.2 Optimization-based Time Integration

In this section we introduce our implicit time-stepping scheme. We show how this may be formulated as a discrete variational optimization problem, and solved by either primal or dual numerical methods. We then analyze the sensitivity and relative strengths of both methods in Section 5.3.

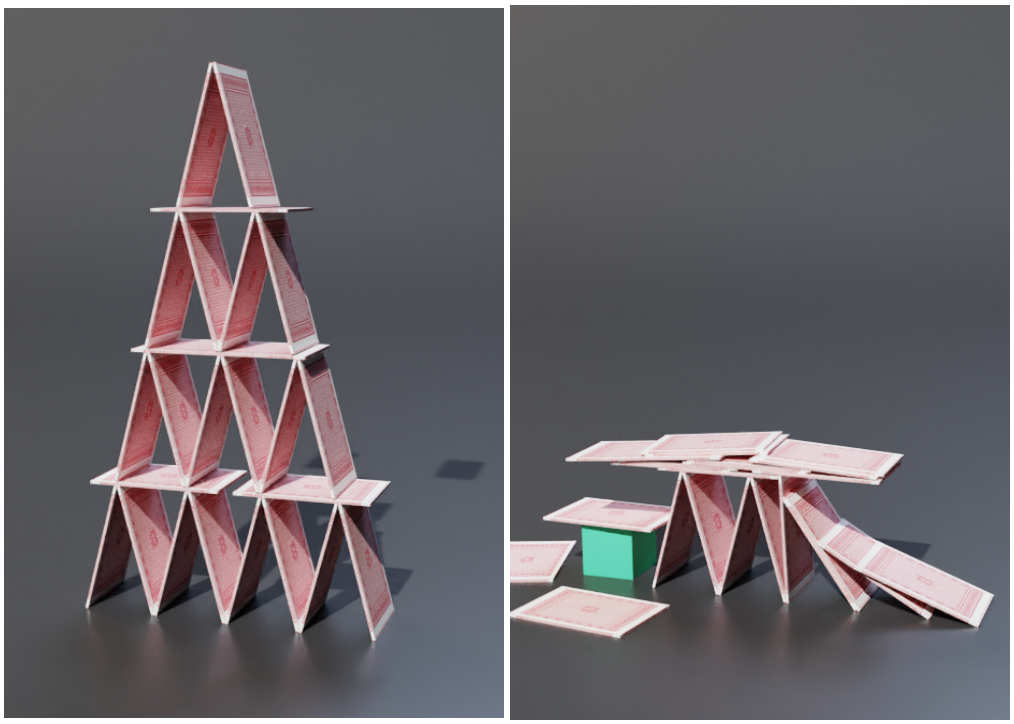


Figure 25: **Structured Stacking**. A classic stacking test involving a house of cards. We find that, until knocked down by an external body, implicit primal contact is able to achieve similarly stable structures to traditional dual methods.

To begin, we define the generalized system coordinates and their time derivatives as $\mathbf{q}$ and $\dot{\mathbf{q}}$ respectively. We use the same notation presented in Section 4, and re-parameterize the system by introducing the discrete velocity $\mathbf{u}^+$, and the relationship $\mathbf{q}^+ = \mathbf{q}^- + \Delta t \mathbf{G}\mathbf{u}^+$, where the superscripts $+/-$ are shorthand to indicate the state at $t$ and $t + \Delta t$, i.e.: the beginning and end of the time-step, respectively. As in the preceding section, the matrix $\mathbf{G}$ is the *kinematic map* that maps spatial velocities to system coordinate time derivatives, i.e.: $\dot{\mathbf{q}} = \mathbf{G}\mathbf{u}$. This velocity re-parameterization allows us to treat rigid bodies and particles in a unified manner. The discrete equations of motion are then

$$\mathbf{M}\left(\mathbf{u}^+ - \tilde{\mathbf{u}}\right) - \Delta t \mathbf{f}(\mathbf{q}^+, \mathbf{u}^+) = \mathbf{0}. \tag{104}$$

Where the constant $\tilde{\mathbf{u}} = \mathbf{u}^- + \Delta t\, \mathbf{M}^{-1}\left(\mathbf{f}_{\text{ext}} + \mathbf{f}_{\text{gyro}}\right)$ is the unconstrained velocity that includes the external and gyroscopic forces integrated explicitly. As shown in previous works, we can formulate implicit time integration as an optimization problem [14, 22, 71]. First, we define the objective function:

$$g(\mathbf{u}) = \frac{1}{2}\left(\mathbf{u} - \tilde{\mathbf{u}}\right)^T \mathbf{M}\left(\mathbf{u} - \tilde{\mathbf{u}}\right) + \sum_i U_i\left(\mathbf{q}^+(\mathbf{u})\right), \tag{105}$$

where $U_i$ are arbitrary energy potentials that give rise to the forces $\mathbf{f}$ on the system. The optimization problem is then,

$$\mathbf{u}^+ \equiv \underset{\mathbf{u}}{\operatorname{argmin}}\, g(\mathbf{u}). \tag{106}$$

The benefit of stating implicit time integration in this variational form is that many robust methods exist to solve such optimization problems, allowing a more unified treatment. We are primarily interested in first order methods, i.e.: those that use only information about the gradient of $g$ since they are simple to implement, and well suited for parallelization [225]. In the following section we will show how to use second order information when available. Note that the gradient of the objective (105) is simply given by (104), i.e.:

$$\mathbf{d}\big|_{\mathbf{u}^+} \equiv \left.\frac{\partial g}{\partial \mathbf{u}}^T\right|_{\mathbf{u}^+} = \mathbf{M}\left(\mathbf{u}^+ - \tilde{\mathbf{u}}\right) - \Delta t \mathbf{f}(\mathbf{q}^+, \mathbf{u}^+) \tag{107}$$

where the generalized force is $\mathbf{f} = -\sum_i \mathbf{G}^T \frac{\partial U_i^T}{\partial \mathbf{q}^+}$.

### 5.2.1 Gradient Descent

Perhaps the simplest approach to solving the minimization (106) is gradient descent. In this scheme we repeatedly update the solution $\mathbf{u}^+$ and $\mathbf{q}^+$ as follows:

$$\mathbf{u}^+ \leftarrow \mathbf{u}^+ - \alpha \mathbf{d} \tag{108}$$

$$\mathbf{q}^+ \leftarrow \mathbf{q}^- + \Delta t \mathbf{G}\mathbf{u}^+ \tag{109}$$

where $\alpha$ is a step-length parameter. In practice gradient descent converges very slowly and a line search may be necessary to avoid overshooting and divergence. We can improve the convergence of gradient descent by defining a preconditioning matrix $\mathbf{P}$. Provided an appropriate choice of $\mathbf{P}$ such that $\mathbf{d}^T \mathbf{P}\, \mathbf{d} > 0$, our descent update for $\mathbf{u}^+$ is then

$$\mathbf{u}^+ \leftarrow \mathbf{u}^+ - \alpha \mathbf{P}\mathbf{d}. \tag{110}$$

A common choice for $\mathbf{P}$ is the Hessian inverse, i.e.: $\mathbf{P} \approx \mathbf{H}^{-1} \equiv \frac{\partial^2 g}{\partial \mathbf{u}^2}^{-1}$, which corresponds to Newton's method. Due to potential indefiniteness, and the complexity of evaluating the Hessian, many approaches exist to approximate $\mathbf{P}$ leading to a range of quasi-Newton methods. In the following section we review and compare some common choices and discuss their relationship.

### 5.2.2 Quadratic Potentials

The primal descent method presented above is applicable to any nonlinear conservative force. However, to draw comparisons to other methods we first consider the special case of quadratic energy potentials. Consider a single potential of the form,

$$U = \frac{1}{2} k\, C(\mathbf{q})^2, \tag{111}$$

where $k$ is a stiffness parameter, and $C(\mathbf{q})$ a constraint function that can be either a scalar or vector function. We define the corresponding generalized force arising from $U$ as,

$$\mathbf{f} = -\mathbf{G}^T \frac{\partial U}{\partial \mathbf{q}}^T = -\, k\, \mathbf{J}^T C(\mathbf{q}), \tag{112}$$

where the constraint Jacobian is given by $\mathbf{J} = \frac{\partial C}{\partial \mathbf{q}}\mathbf{G}$. For a Newton style preconditioner we need the Hessian, $\mathbf{H}$, of our objective function $g$ with respect to the solution variable $\mathbf{u}$,

$$\mathbf{H} = \frac{\partial^2 g}{\partial \mathbf{u}^2} = \mathbf{M} - \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{u}}. \tag{113}$$

Assuming the mass $\mathbf{M}$ is known, the term to be computed is the force Jacobian $\frac{\partial \mathbf{f}}{\partial \mathbf{u}}$, which, for a quadratic potential, is given by,

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}} = -\Delta t k \left[ \mathbf{J}^T \mathbf{J} + \frac{\partial \mathbf{J}}{\partial \mathbf{u}} C \right]. \tag{114}$$

Here, the second term corresponds to geometric stiffness [215, 7]. Using just first-order terms, the preconditioner is

$$\mathbf{P}^{GN} \equiv \left[ \mathbf{M} + \Delta t^2 \, k \, \mathbf{J}^T \mathbf{J} \right]^{-1} \approx \left[ \frac{\partial^2 g}{\partial \mathbf{u}^2} \right]^{-1}, \tag{115}$$

which corresponds to a Gauss-Newton iteration on $g$. To avoid computing the inverse, or solving a system of equations, we use a simple diagonal approximation, where each entry is the reciprocal of the diagonal of $\mathbf{P}^{GN}$, i.e.:

$$\mathbf{P}^{D}_{dd} \equiv \frac{1}{\mathbf{M}_{dd} + \Delta t^2 \, k \, \mathbf{J}_d^2}. \tag{116}$$

Note that $d$ is the index of the degree of freedom, not the constraint. In Section 5.4 we show how to extend preconditioners of this form to contact and friction.

### 5.2.3 Dual Ascent

Given a constrained optimization problem it is possible to construct a *dual* optimization problem over Lagrange multipliers [23]. In this section we derive the dual problem for the case of quadratic potentials, and show how it leads naturally to constrained dynamics methods such as extended position-based dynamics (XPBD).

To construct the dual of our primal optimization problem we introduce the auxiliary variables $\boldsymbol{\lambda} = -\mathbf{K}\,\mathbf{c}$ where $\mathbf{K} = \text{diag}[k_1, \ldots, k_n]$ is a matrix of stiffness values and is the inverse of compliance, i.e.: $\mathbf{K}^{-1} = \mathbf{E}$, and $\mathbf{c} = [C_1, \ldots, C_n]$ is a vector of constraint functions. This allows us to rewrite the system potential energy as $U = -\frac{1}{2}\mathbf{c}^T\boldsymbol{\lambda}$, and define the following Lagrangian,

$$\mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}) \equiv \frac{1}{2}(\mathbf{u} - \tilde{\mathbf{u}})^T\mathbf{M}(\mathbf{u} - \tilde{\mathbf{u}}) - \boldsymbol{\lambda}^T\,\mathbf{c}(\mathbf{q}^+) - \frac{1}{2}\boldsymbol{\lambda}^T\mathbf{K}^{-1}\boldsymbol{\lambda}. \tag{117}$$

It can be verified that the stationarity conditions for this Lagrangian correspond to the original problem (106) with quadratic potentials [23]. While both primal and dual optimization can be applied to arbitrary energy potentials, the dual formulation requires finding a splitting into Lagrange multipliers that is not always straightforward, as shown in Section 4.7.4.

The corresponding Lagrange dual function for (117) is $h(\boldsymbol{\lambda}) = \inf_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \boldsymbol{\lambda}) = \mathcal{L}(\mathbf{u}^*, \boldsymbol{\lambda})$. In general, the constraint functions are nonlinear, and so we cannot obtain a closed form expression for $\mathbf{u}^*$ in terms of $\boldsymbol{\lambda}$. However, assuming constraint linearity we can make the following approximation $\mathbf{u}^* \approx \tilde{\mathbf{u}} + \Delta t\mathbf{M}^{-1}\mathbf{J}^T\boldsymbol{\lambda}$. Inserting this into the Lagrangian, the dual function is then

$$h(\boldsymbol{\lambda}) \approx \frac{\Delta t^2}{2}\boldsymbol{\lambda}^T\left(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\right)\boldsymbol{\lambda} - \boldsymbol{\lambda}^T\mathbf{c}(\mathbf{q}^+) - \frac{1}{2}\boldsymbol{\lambda}^T\mathbf{K}^{-1}\boldsymbol{\lambda}, \tag{118}$$

with a corresponding dual *maximization* problem

$$\boldsymbol{\lambda}^+ = \underset{\boldsymbol{\lambda}}{\text{argmax}}\; h(\boldsymbol{\lambda}). \tag{119}$$

To derive the optimality conditions for (119) we take the derivative of $h$, keeping in mind that $\mathbf{q}^+$ is implicitly a function of $\mathbf{u}^*$ and in turn $\boldsymbol{\lambda}$, to obtain

$$\frac{\partial h}{\partial \boldsymbol{\lambda}} = -\left[\mathbf{c}(\mathbf{q}^+) + \mathbf{K}^{-1}\boldsymbol{\lambda}\right] = \mathbf{0}. \tag{120}$$

This set of nonlinear equations corresponds to the form in the XPBD algorithm [138]. To build a preconditioner we evaluate the Hessian with respect to $\boldsymbol{\lambda}$, again differentiating through the definition of $\mathbf{q}^+$, giving:

$$\frac{\partial^2 h}{\partial \boldsymbol{\lambda}^2} = -\left[\Delta t^2 \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T + \mathbf{K}^{-1}\right], \tag{121}$$

with the diagonal preconditioner for the dual ascent given by

$$\mathbf{P}_{ii}^D = \frac{1}{\Delta t^2 \mathbf{J}_i \mathbf{M}^{-1} \mathbf{J}_i^T + \mathbf{K}_{ii}^{-1}}. \tag{122}$$

Note that for maximization the sign of the preconditioner is reversed to ensure an ascent direction. The update step is then

$$\boldsymbol{\lambda}^+ \leftarrow \Pi\left(\boldsymbol{\lambda}^+ + \alpha \mathbf{P} \frac{\partial h}{\partial \boldsymbol{\lambda}}\right), \tag{123}$$

where $\Pi$ is a projection operator that is used to enforce bound and friction constraints on the dual variables. This is followed by an update of the primal variables,

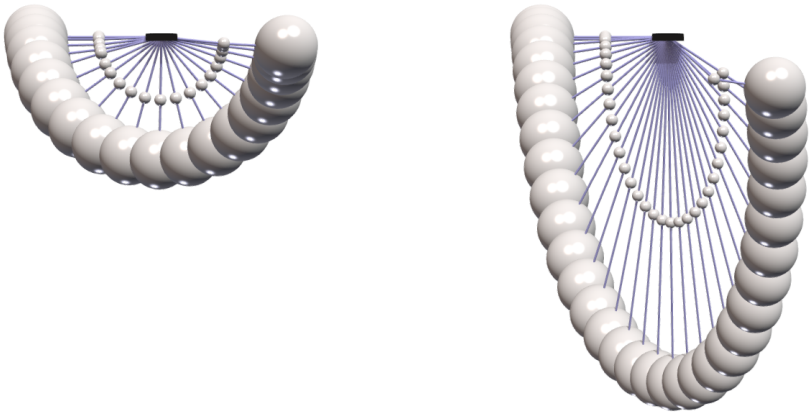
$$\mathbf{u}^+ \leftarrow \tilde{\mathbf{u}} + \Delta t \mathbf{M}^{-1} \mathbf{J}^T \boldsymbol{\lambda}^+ \tag{124}$$

$$\mathbf{q}^+ \leftarrow \mathbf{q}^- + \Delta t \mathbf{G} \mathbf{u}^+. \tag{125}$$

This derivation shows how we may obtain dual-space algorithms such as XPBD from the starting point of a primal optimization problem. When using a diagonal preconditioner, the above update for $\boldsymbol{\lambda}^+$ is identical to that of a Jacobi XPBD iteration. However, in (124)-(125) the primal variable update differs from XPBD by applying updates from the initial state $\mathbf{q}^-, \mathbf{u}^-$ rather than the current descent iterate. A similar observation was made by Daviet [42]. While this modification ensures the method converges to the same solution as the primal form, we found using the *Fast Projection* update of XPBD where position modifications are applied incrementally was more robust [74]. We outline both primal and dual methods in Listings (2-3).

## 5.3 Conditioning

Regardless of the preconditioner used for our numerical method, we can analyze the conditioning of both the primal and dual problems by inspecting their Hessian side-by-side:



(a) Primal (b) Dual

Figure 26: **Mass Ratio Test**. A double pendulum consisting of two spheres with a mass ratio of $10^4$. High mass ratios cause ill-conditioning for dual methods, which manifests as excessive stretching when using fixed iteration counts. Primal formulations are insensitive to mass ratios and show the correct behavior.

(a) Primal (b) Dual

Figure 27: **Stiffness Ratio Test**. We simulate an elastic double pendulum where the lower spring is $10^4$ times stiffer than the upper one. Both springs are stiff enough to easily support the attached weights, however the high stiffness ratio causes ill-conditioning for primal formulations and leads to significant error (stretching). In contrast, dual formulations are insensitive to stiffness ratios, and show the correct behavior.

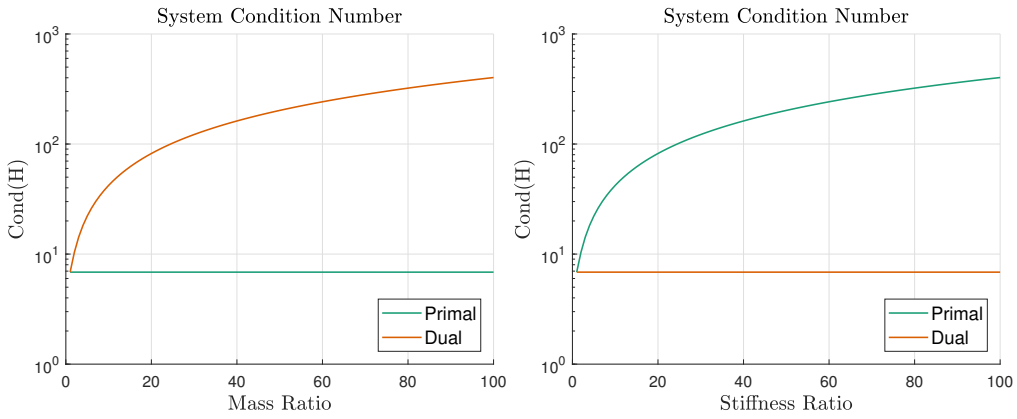



Figure 28: **Conditioning**. A plot of the system condition number for a 1D chain of particles with a large mass attached, as shown in Figure 26. As the mass of the weight is increased the condition number of the dual system increases (left). The situation is exactly reversed for the case of stiffness ratios shown in Figure 27, where increasing stiffness leads to poor conditioning (right).

$$\frac{\partial^2 g}{\partial \mathbf{u}^2} = \left[\mathbf{M} + \Delta t^2 \mathbf{J}^T \mathbf{K} \mathbf{J}\right] \tag{126}$$

$$\frac{\partial^2 h}{\partial \boldsymbol{\lambda}^2} = \left[\Delta t^2 \mathbf{J} \mathbf{M}^{-1} \mathbf{J}^T + \mathbf{K}^{-1}\right]. \tag{127}$$

Inspecting the first (primal) case, when $\mathbf{K}$ has a large norm and is poorly conditioned (e.g.: there are high stiffness ratios), then this will dominate the mass term and primal descent methods will converge slowly, leading to error, as illustrated in Figure 27. The situation is reversed for the dual form, when $\mathbf{M}$ has a large relative norm and is poorly conditioned (e.g.: high mass ratios) then the system will be hard to solve for iterative dual methods as shown in Figure 26. In Figure 28 we see how, for a simple 1D chain, the condition number of the systems exactly mirror each other for mass/stiffness ratios in each form. While common wisdom states that high stiffness values lead to poorly conditioned systems, this analysis shows that it is actually the *stiffness ratio* that is problematic rather than the absolute stiffness values. We further analyze the effect of this on iterative methods in Section 5.6.

## 5.4 Contact

We now present a novel primal contact model that incorporates slip and stick regions with a robust preconditioner suitable for implicit integration with descent-based solvers. We first restate our non-interpenetration constraints using inequalities as follows:

**Algorithm 2: Primal Descent Simulation Loop.**

**while** *Simulating* **do**
  Perform collision detection;
  $\mathbf{u}^+ \leftarrow \tilde{\mathbf{u}}$;
  $\mathbf{q}^+ \leftarrow \mathbf{q}^- + \Delta t \mathbf{G}\mathbf{u}^+$;
  **for** *n descent iterations* **do**
    Initialize force $\mathbf{f}$, and Jacobian diagonal $\mathbf{p}$;
    $\mathbf{f} \leftarrow \mathbf{0}$;
    $\mathbf{p} \leftarrow \mathbf{0}$;
    Evaluate forces and derivatives;
    **for** *i forces* **do**
      $\mathbf{f} \leftarrow \mathbf{f} + \mathbf{f}_i$;
      $\mathbf{p} \leftarrow \mathbf{p} + \text{diag}\left(\Delta t k_i \mathbf{J}_i^T \mathbf{J}_i\right)$ ;
    **end**
    Build preconditioner;
    **for** *d degrees of freedom* **do**
      $\mathbf{P}_{dd}^D = (\mathbf{M}_{dd} + \Delta t \mathbf{p}_d)^{-1}$;
    **end**
    Compute gradient;
    $\mathbf{d} \leftarrow \mathbf{M}(\mathbf{u}^+ - \tilde{\mathbf{u}}) - \Delta t \mathbf{f}$;
    Update state;
    $\mathbf{u}^+ \leftarrow \mathbf{u}^+ - \alpha \mathbf{P}^D \mathbf{d}$;
    $\mathbf{q}^+ \leftarrow \mathbf{q}^- + \Delta t \mathbf{G}\mathbf{u}^+$;
  **end**
**end**

$$C_n(\mathbf{q}) \equiv \mathbf{n}^T \left[\mathbf{a}(\mathbf{q}) - \mathbf{b}(\mathbf{q})\right] - d \geq 0, \tag{128}$$

where $\mathbf{n} \in \mathbb{R}^3$ is the contact plane normal given by the direction vector between closest points of triangle-mesh features, and $d$ is a separation distance to maintain that may be used to model surface thickness. Although we treat the contact normal as fixed over the course of the time step, it is also possible to use a nonlinear constraint on the object motion [119].

Penalty methods of contact associate a stiff potential with the contact constraint (128). This may be viewed as relaxation of the complementarity condition so that some interpenetration is allowed [198]. In the simplest case, the penalty force is a function of the clamped constraint

**Algorithm 3: Dual Ascent Simulation Loop**

**while** *Simulating* **do**
  Perform collision detection;
  $\mathbf{u}^+ \leftarrow \tilde{\mathbf{u}}$;
  $\mathbf{q}^+ \leftarrow \mathbf{q}^- + \Delta t \mathbf{G}\mathbf{u}^+$;
  **for** *n ascent iterations* **do**
    Initialize Lagrange multipliers $\boldsymbol{\lambda}$, and dual gradient $\mathbf{h}$;
    $\boldsymbol{\lambda} \leftarrow \mathbf{0}$;
    $\mathbf{h} \leftarrow \mathbf{0}$;
    Evaluate constraints and derivatives;
    **for** *i constraints* **do**
      $\mathbf{h}_i = -C_i(\mathbf{q}^+) - k_i^{-1}\boldsymbol{\lambda}_i$;
      $\mathbf{P}_{ii}^D = \left(\Delta t^2 \mathbf{J_i} \mathbf{M}^{-1} \mathbf{J_i}^T + \mathbf{K}_{ii}^{-1}\right)^{-1}$;
    **end**
    Compute dual update;
    $\boldsymbol{\lambda} \leftarrow \Pi\left(\boldsymbol{\lambda} + \alpha \mathbf{P}^D \mathbf{h}\right)$;
    Update state;
    $\mathbf{u}^+ \leftarrow \mathbf{u}^+ + \Delta t \mathbf{M}^{-1} \mathbf{J}^T \Delta\boldsymbol{\lambda}$;
    $\mathbf{q}^+ \leftarrow \mathbf{q}^- + \Delta t \mathbf{G}\mathbf{u}^+$;
  **end**
**end**

error,

$$U_n(\mathbf{q}) \equiv \frac{k_n}{p} \min(0, C_n(\mathbf{q}))^p. \tag{129}$$

where $p$ is a constant exponent (often chosen to be 2). The associated force due to this potential is

$$\mathbf{f}_n(\mathbf{q}) \equiv -k_n \mathbf{J}_n^T \min(0, C_n(\mathbf{q}))^{p\text{-}1}, \tag{130}$$

where $k_n$ controls the stiffness of the contact. One advantage of penalty based approaches is that they can easily support nonlinear contact force models [104]. In addition, by varying $p$ we can obtain smoother contact forces that provide continuous derivatives. As shown in Figure 29, when $k_n \to \infty$ the force approaches a hard constraint limit, and as $p$ increases, so does the

smoothness of contact forces. To construct a preconditioner for the contact normal force we use the following Hessian approximation:

$$\frac{\partial \mathbf{f}_n}{\partial \mathbf{u}} \approx -\begin{cases} k_n \mathbf{J}_n^T \mathbf{J}_n (p\text{-}1) \min(0, C_n(\mathbf{q}))^{p\text{-}2} & C_n(\mathbf{q}) < 0 \\ \mathbf{0}, & \text{otherwise,} \end{cases} \tag{131}$$

where we have dropped higher order terms corresponding to the geometric stiffness [215]. This ensures the preconditioner remains positive definite and that the step is in a descent direction. This approximation is justified since, unlike single-step Newton schemes, we repeatedly re-evaluate the constraint gradients throughout the nonlinear optimization.

## 5.5 Friction

To introduce friction forces we first define the slip velocity at a contact as $\mathbf{u}_s = \mathbf{D}^T \mathbf{u} \in \mathbb{R}^2$, where $\mathbf{D} \in \mathbb{R}^{n_d \times 2}$ is a basis that projects the bodys' relative velocity to the tangent plane, defined by two orthogonal vectors, perpendicular to the normal $\mathbf{n}$.

We now present our novel primal formulation of frictional contact. We postulate a variational energy giving rise to frictional dissipation forces similar to Pandolfi & Ortiz [163]. However, to address the issue of indeterminancy in the static $\mathbf{u}_s = \mathbf{0}$ case we relax the Coulomb model to include a stiff quadratic region around the origin,

$$U_f(\mathbf{u}) \equiv \begin{cases} \frac{1}{2} k_f |\mathbf{u}_s|^2 & k_f |\mathbf{u}_s| < \mu |\mathbf{f}_n| \\ \mu |\mathbf{f}_n| |\mathbf{u}_s| - \gamma, & \text{otherwise.} \end{cases} \tag{132}$$

Here the parameter $k_f$ controls stiffness in the 'stick' regime, where the Coulomb condition $k_f |\mathbf{u}_s| < \mu |\mathbf{f}_n|$ requires that the friction force lie inside the normal cone. We treat $\mathbf{f}_n$ as a constant parameter to the potential, which in the discrete setting corresponds to staggering, or lagging the update of normal forces in the friction calculations between iterations [108]. The constant $\gamma = \frac{\mu^2 |\mathbf{f}_n|^2}{2k_f}$ is chosen to make the potential have $C^0$ continuity when $k_f |\mathbf{u}_s| = \mu |\mathbf{f}_n|$. This potential is quadratic around the origin and is linear past a certain point (in the slip regime). It gives rise to the following forces:

$$\mathbf{f}_f(\mathbf{u}) \equiv -\min\left(k_f, \mu \frac{|\mathbf{f}_n|}{|\mathbf{u}_s|}\right) \mathbf{D}^T \mathbf{u}_s, \tag{133}$$

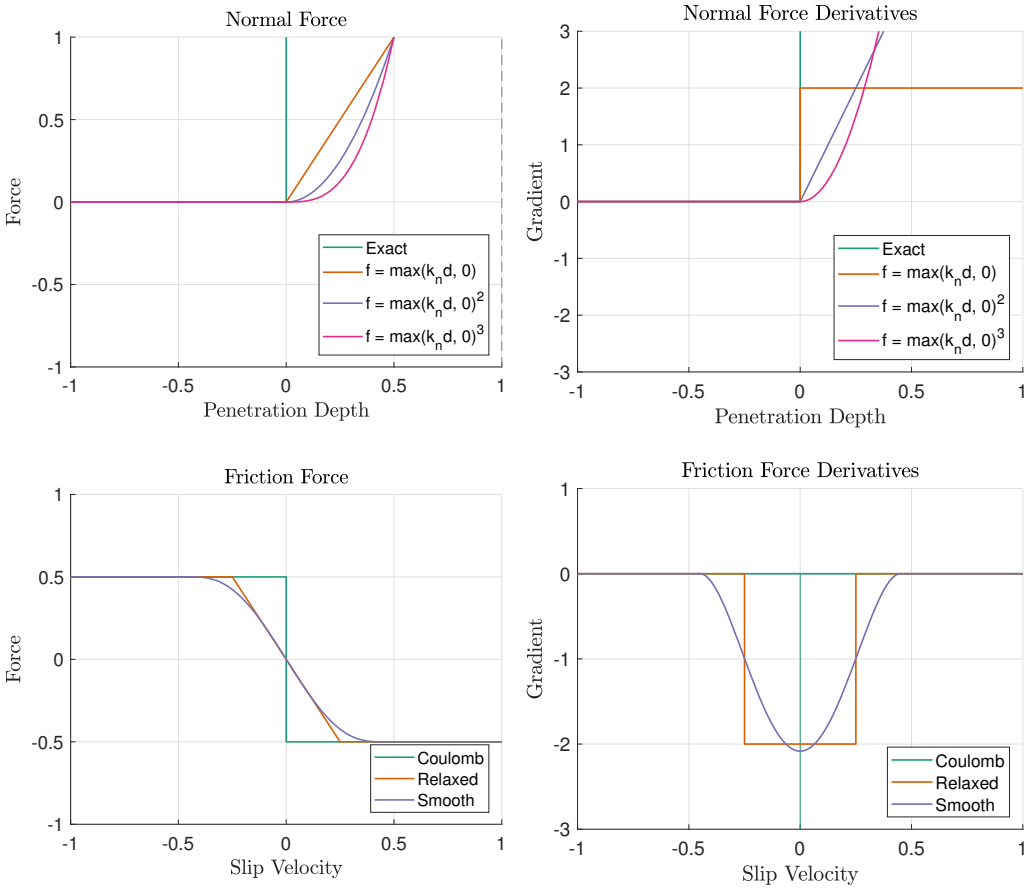



Figure 29: **Contact Forces**. Relaxed contact models approximate hard contact by replacing the step function with a linear hinge (top left). By exponentiating this function we can obtain $C^1$/$C^2$ continuity, with analytic derivatives (top right). Coulomb friction may also be relaxed to obtain invertible contact models (bottom left). The relaxed friction model may then be smoothed to obtain second order differentiability (bottom right).

which in 1D looks like the relaxed step function as illustrated in Figure 29. To construct a preconditioner for this frictional force we require the potential Hessian, which has the form,

$$\frac{\partial \mathbf{f}_f}{\partial \mathbf{u}} \equiv -\mathbf{D}^T \mathbf{\Lambda} \mathbf{D}, \tag{134}$$

with $\mathbf{\Lambda}$ given by

$$\mathbf{\Lambda} \equiv \begin{cases} k_f \mathbf{I} & k_f |\mathbf{u}_s| < \mu |\mathbf{f}_n| \\ \mu \frac{|\mathbf{f}_n|}{|\mathbf{u}_s|} \left( \mathbf{I} - \frac{\mathbf{u}_s \mathbf{u}_s^T}{|\mathbf{u}_s|^2} \right) & \text{otherwise.} \end{cases} \tag{135}$$

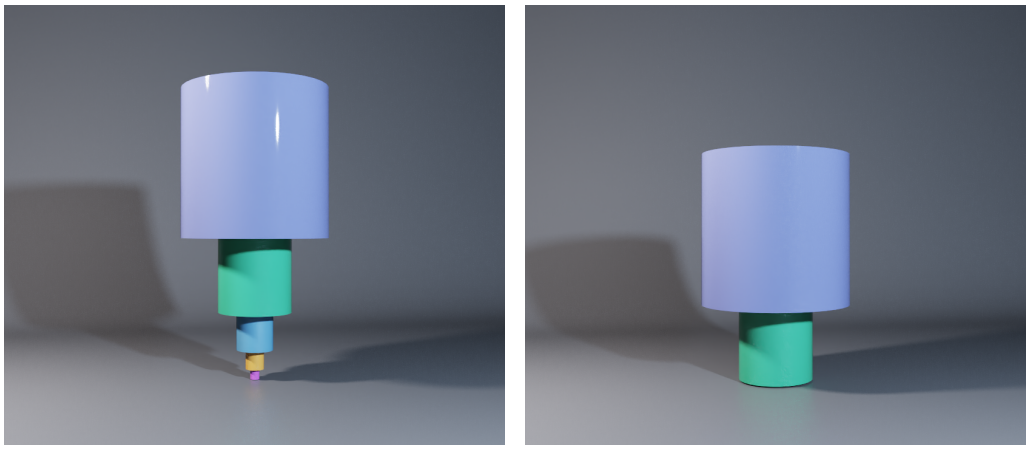
The term $\mathbf{I} - \frac{\mathbf{u}_s \mathbf{u}_s^T}{|\mathbf{u}_s|^2}$ comes from the derivative of a normalized vector, and accounts for the turning of constraint directions. We found this term can cause gradient descent to fail to converge reliably. Instead we use the simpler, and slightly more conservative scalar Hessian approximation:

$$\Lambda \approx \begin{cases} k_f & k_f |\mathbf{u}_s| < \mu |\mathbf{f}_n| \\ \mu \frac{|\mathbf{f}_n|}{|\mathbf{u}_s|}, & \text{otherwise.} \end{cases} \tag{136}$$

Similar to non-penetration constraints, relaxed friction models may also be smoothed to provide continuous derivatives. This is illustrated in Figure 29, where the quadratic no-slip region in our friction potential is replaced with a 5th degree interpolating polynomial [54]. Alternatively functions such as the pseudo-Huber norm offer higher order continuity and may also serve as a smooth friction approximation [30].

## 5.6 Results

To investigate the relationship between primal and dual formulations we implement the Projective Dynamics (primal) method of Wang et al. [225], and the XPBD (dual) method of Macklin et al. [138]. Both methods are amenable to parallelization, and so we use CUDA and run them on an NVIDIA Geforce 2080 Ti. For collision detection we use triangle-mesh based contact generation between point-face and edge-edge feature pairs in proximity at start of each time-step. We use a per-frame time step of $\Delta t = 0.0166$ seconds, with a varying number of substeps depending on the example as reported in Table 3. For rigid body simulation we have a block-diagonal



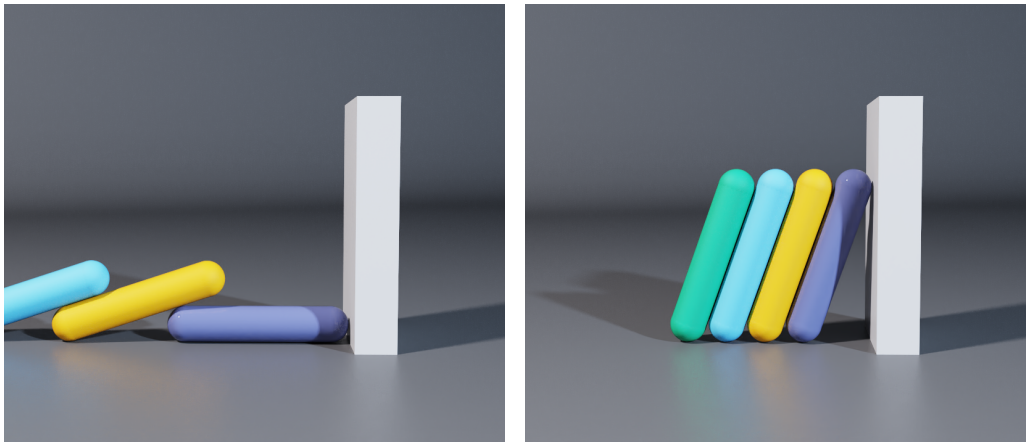
(a) Primal (b) Dual

Figure 30: **Contact Mass Ratio**. A contact stacking scenario with a mass ratio of 4096:1. Primal solvers are insensitive to this ratio and stack stably in 20 iterations. Conversely, this scenario leads to ill-conditioning for dual methods which fail to stack with 500 iterations.

mass matrix, which we invert blockwise to obtain the preconditioner. We use a collision thickness $d \in [10^{-3}, 10^{-2}]$m. As our focus is on real-time applications we use a fixed number of iterations per-time step. Line search may be necessary to make gradient-based methods robust. However, for the primal descent method we found that using a fixed value of $\alpha = 0.5$ was sufficient to ensure convergence for all cases we tested. We found dual methods to be more sensitive to the choice of step length, especially when many contacts influence a single body. To ensure convergence in the dual case we have used the mass-splitting approach of Tonge et al [214]. This amounts to treating $\alpha$ as a diagonal matrix that provides a varying step size for each dual variable.

**Mass Ratio Tests** We measure the sensitivity of both primal and dual methods to mass ratios using a simple double pendulum with a mass ratio of 10000:1 as shown in Figure 26. In this case, optimization on the dual problem proceeds slowly, leading to large stretching. In contrast, primal-space optimization is relatively unaffected. An equivalent result occurs in the contact scenario shown in Figure 30. Here the stack has a mass ratio of 4096:1 between the lower cylinder and the top one. For the primal solver we use contact parameters of $k_n = k_f = 10^8$ which is sufficient to stably support the stack. In contrast, dual-space optimization converges slowly for this case, leading to large interpenetrations.

**Stiffness Ratio Tests** A simple test to illustrate the effect of stiffness ratios is shown in Figure 27. Here two point masses are connected by springs with stiffness coefficents that vary by a ratio



(a) Primal (b) Dual

Figure 31: **Contact Stiffness Ratio**. An example of a high stiffness ratio contact scenario. In this case we have chosen contact stiffness coefficients of $k_n = 10^8$, and $k_f = 10^6$ creating a stiffness ratio of 100:1. This leads to an ill-conditioned system that results in artificial slip for primal descent solvers.

(a) $k_f = 10^2$ (b) $k_f = 10^3$ (c) $k_f = 10^4$ (d) Dual

Figure 32: **Grasp Stability**. We measure the effect of frictional stiffness on grasp stability. Here the Yumi robot picks up a cube, attempts to lift it to a height of 8cm, and remain stationary. Each image shows the final state of a grasp after 15s. We show the effect of varying friction stiffness with $k_f$ increasing from left to right. Low stiffness results in visible slipping, but with high stiffness and implicit integration grasps can be made nearly as stable as hard contact (dual) models over long periods of time.

of 10000:1. Although both springs are stiff enough to easily support the masses, when combined with a descent-based solver, the much stiffer lower spring has the effect of slowing convergence for the top spring, resulting in significant stretching. Dual-space solvers do not suffer from any ill-conditioning in this case, showing the correct (unstretched) behavior.

An equivalent, but less obvious, stiffness ratio problem occurs in contacting scenarios where the normal and friction stiffness coefficients $k_n$ and $k_f$ differ by a large magnitude. In Figure 31 we see the effect of raising the contact stiffness while leaving the friction stiffness fixed. When combined with an iterative method, this has the effect of reducing friction convergence, leading to artificial slip.

**Rigid Piling** To investigate the performance of each method on large scale unstructured piling we simulate a granular material consisting of 256k rigid bodies as shown in Figure 23. The grains consist of spherical bodies with an average radius of 5mm and a mass of 5g. We use contact parameters of $k_n = k_f = 10^4$ and $\mu = 0.3$. A large cylindrical weight is dropped onto the pile creating a mass ratio of 80000:1. After 1.25s one of the walls is removed, allowing the grains to flow out. For primal methods we see that the grains support the weight easily, while dual optimization with the same iteration count shows significant compression.

A second example is illustrated in Figure 24. In this case, 512 bowls each represented by a triangle mesh with 1160 faces are dropped from a height. We use contact parameters of $k_n = k_f = 10^6$, and $\mu = 0.7$. This example is inspired by Xu & Barbic [232], who simulated large plate stacks with a direct method. They used distance field based collision detection that generated 8.5k maximum contacts. Our triangle-based representation generates 393k contacts when settled (40x larger), however our parallel gradient-based solver still runs at real time rates and forms a stable pile. In this example we see similar behavior with both primal and dual based solvers. This is expected since there are no significant sources of ill-conditioning. Nevertheless, this example shows that the use of primal contact models and descent based optimization is practical as an alternative to iterative dual methods that may traditionally be used for such simulations.

**Force Distributions** In real world scenarios, for example a tessellated cylinder resting on the ground, there are often many redundant contact constraints. This creates an underdetermined problem, with many possible contact force distributions that are all valid solutions. In this case, hard-contact solvers may produce an unpredictable distribution that is dependent on the ordering of constraints. In some applications uniform force distributions are desirable, for example, in sound synthesis Zheng et al. [238] used a secondary per-pair optimization to generate smooth contact forces. A benefit of the relaxed primal model presented here is that it generates evenly

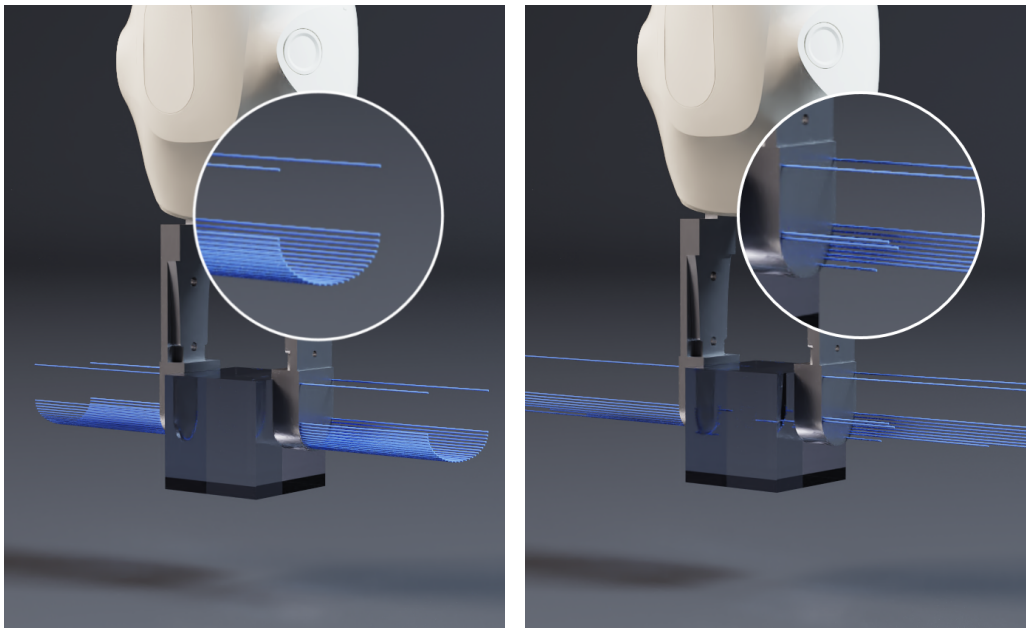
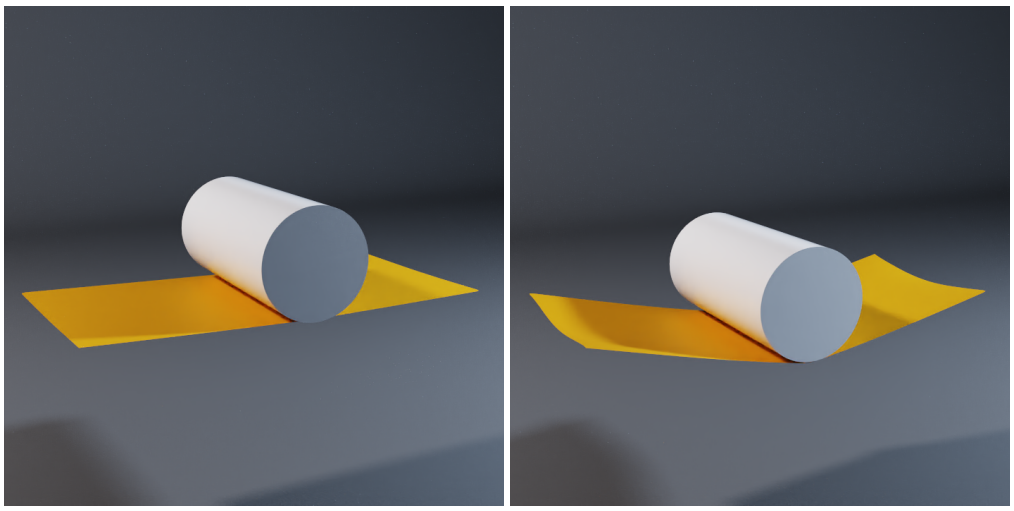


Figure 33: **Force Distributions**. We visual contact normal forces in blue. The distribution of forces generated by a relaxed primal contact model are smooth (left). For hard contact models the problem is underdetermined, leading to a solution that depends on the ordering of contacts (right).

distributed forces without additional post-processing. An example of this is shown in Figure 33. Here the ABB Yumi robot grips a cube. The gripper is tessellated somewhat non-uniformly, but in a relaxed primal model of contact the system remains well-posed and leads to a smooth distribution of contact forces over the vertices.

**Cloth** To investigate the behavior of each method on deformable simulation, we suspend a piece of cloth modeled by 800 triangles and drop a rigid body onto it from a height, as shown in Figure 34. Surprisingly, the dual solution does not become stiffer after some elastic stiffness threshold value around $k = 10^8$. The same effect was observed by Soler et al. [194] in the context of Cosserat rods. We found this effect was consistent even in the absence of contact, and believe this behavior can be explained by the use of an approximate solution to the primal optimization subproblem in the dual update. In this case a hybrid method like the one used by Narain et al. [153] for cloth strain-limiting may perform better.

**Grasp Stability** Without some additional history-based contact tracking, relaxed models of friction do not generate frictional forces at zero-velocity. This means some non-zero slip is expected. We measure the error induced by the this approximation on a robotic manipulation



(a) Primal (b) Dual

Figure 34: **Cloth Simulation**. For an object suspended by inextensible cloth ($k = 10^{10}$) we find that primal descent (left) is able to achieve higher effective stiffness than dual ascent (right) in an equivalent number of iterations.

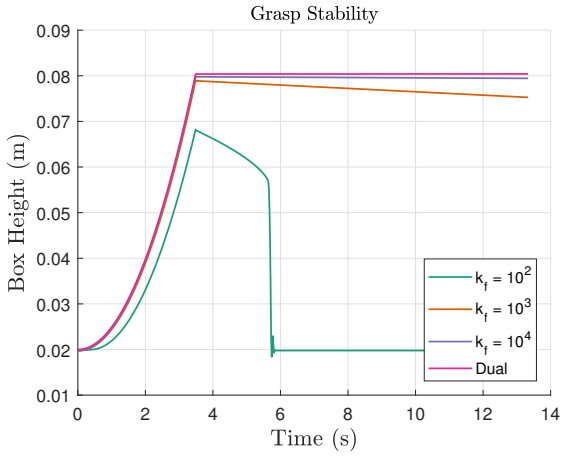



Figure 35: **Grasp Stability**. A plot of the box height over time for the Yumi grasp example shown in Figure 32. We consider the solution given by a hard-contact dual solver as the baseline. At low values of $k_f$ the relaxed friction model visibly slips. For high values of $k_f$ the relaxed model is stable over the experiment's duration.

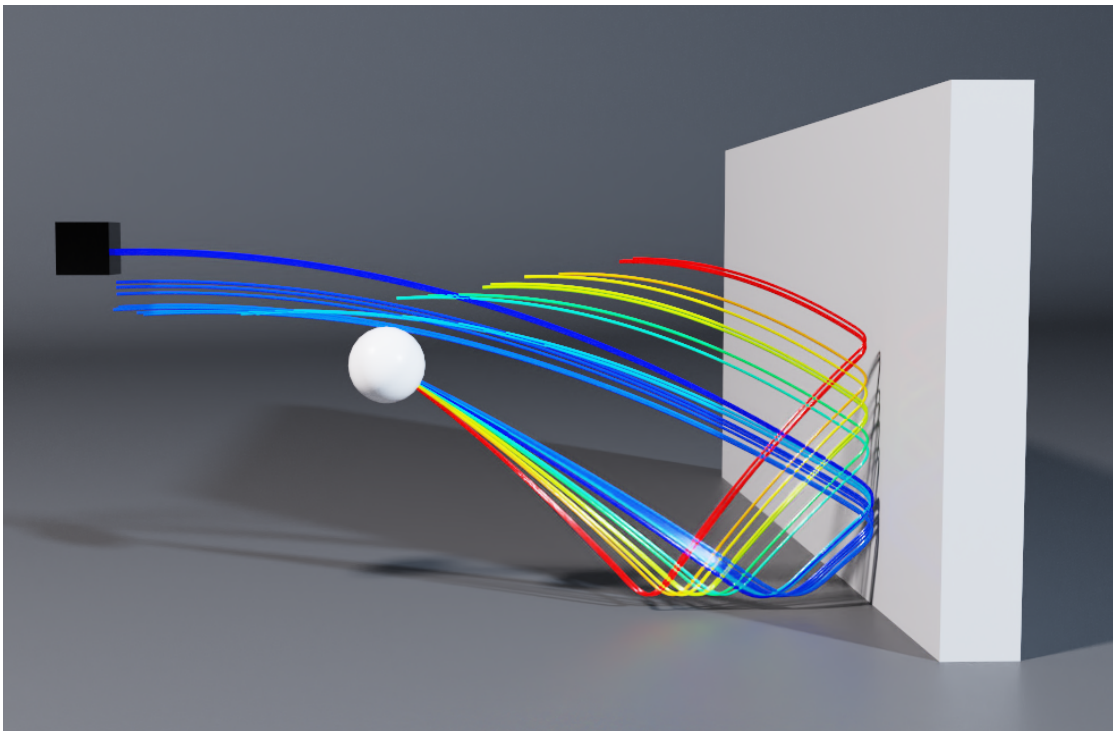


Figure 36: **Differentiability**. A trajectory optimization test using differentiable simulation to minimize the distance to a target through two contact events. Our primal contact model is well-suited to differentiability since it may be smoothed, and the system size remains fixed regardless of the number of contacts.

task shown in Figure 32. We found that for low values of $k_f$ the robot was unable to hold the cube stationary. However, for high enough coefficients we found that primal contact was competitive with the dual baseline.

**Differentiability** We show an example of trajectory optimization in Figure 36. Given an initial trajectory that involves impacts with two surfaces, the goal is to find a starting impulse such that a bouncing ball hits a target at $t = 0.5$s. We use a discrete adjoint method [114] to perform reverse mode differentiation through the primal contact solver and optimize the loss function using an L-BFGS optimizer [128]. Primal formulations of contact are well-suited to differentiability for two reasons. First, the contact forces may be smoothed to provide $C^2$ continuity required for some optimization methods. Second, when performing reverse mode differentiation the state of the system must be saved at each forward step to compute the correct gradients during the backwards pass. For primal contact the size of the system is fixed regardless of the number of contacts. In contrast, the dual system must store a varying, and potentially large amount of contact information at each forward step.

Table 3: Simulation statistics and performance numbers for the examples in this paper. The per-iteration work done by primal and dual descent methods is similar so we expect timings to be consistent between methods. We report the timings in milliseconds per-frame for both methods.

| **Example** | **Steps** | **Iters.** | **Contacts** | **Primal** | **Dual** |
|---|---|---|---|---|---|
| | # | # | # (Avg) | ms | ms |
| Cylinder Stack | 2 | 20 | 189 | 0.9 | 0.9 |
| Capsule Lean | 2 | 20 | 11 | 0.5 | 0.4 |
| Yumi Grasp | 4 | 80 | 60 | 6.4 | 6.8 |
| Granular Material | 8 | 30 | 1510k | 1396.8 | 1136.7 |
| Bowl Pile | 8 | 15 | 393k | 16.6 | 18.1 |
| House of Cards | 4 | 100 | 1.2k | 9.3 | 8.6 |

## 5.7 Limitations

The lack of hard constraints in the primal form would appear to be a major limitation, however, it is often possible to design a reduced system where constrained degrees of freedom are removed completely, e.g.: for articulated rigid bodies [61, 118]. Our derivation in terms of generalized velocities naturally supports these types of re-parameterizations. In our experience, the biggest limitation of primal solvers is the effect of stiffness ratios on convergence. Stiffness ratios may manifest themselves in unexpected ways, for example in the leaning capsule scene in Figure 31. While it is often sufficient to simply set the contact and friction stiffness to the same value, some situations may require more careful authoring.

Our method is designed to work with the descent-based optimization form of Projective Dynamics presented by Wang et al. [225] that does not require pre-factorized matrices. The extension to prefactorized methods is not obvious, and we leave this as future work. An advantage of the primal form is that it does not require the inversion of the mass matrix $\mathbf{M}$. This means it is possible to use consistent mass matrices, which are able to produce more accurate results for FEM-based simulations [33]. Primal formulations also make it particularly easy to perform implicit integration of arbitrary force models. For example, materials with nonlinear constitutive equations, or activation models with complex dynamics such as muscle-tendon units. Our primal formulation is applicable to any nonlinear energy (with the caveat that it may not find a global optimum for nonconvex models), however the dual form requires finding a suitable variable splitting which is most easily performed for quadratic energies.

## 5.8 Conclusion and Future Work

We have presented a unified derivation and analysis of primal and dual formulations of implicit integration from a variational perspective. In addition, we propose a novel primal contact model with a robust preconditioner that is easy to incorporate into existing solvers such as Projective Dynamics. Our contributions extend these frameworks to large scale rigid body and robotics simulations that may have otherwise been treated with a separate method.

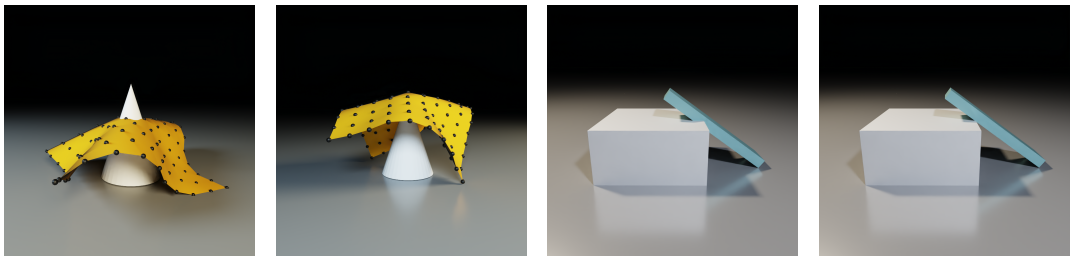
We have focused on iterative descent-based methods, however we expect some of the effects described here will change when used with other methods, e.g.: direct solvers. For future work we plan to explore techniques to address the stiffness ratio problem in primal formulations. We believe accelerated descent and nonlinear conjugate gradient methods are promising methods to improve convergence, while maintaining scalability.

Figure 37: **Cloth Contact**: Our face-based optimization method accurately captures contacts between cloth and sharp features in a signed distance function without discrete sampling of mesh geometry.

# 6 Collision Detection

The simulation methods developed in the previous sections are only as good as their inputs. This is especially true when considering the contact constraints between bodies, which must be accurate enough to prevent object interpenetrations. In this chapter we turn to the problem of collision detection. We propose a new method for contact generation between bodies represented by signed distance fields (SDFs) that robustly captures sharp features, and efficiently generates contacts between rigid bodies and thin-shells such as cloth.

While triangle-meshes are able to compactly and accurately represent the surface detail of an object, they do not explicitly represent the inside/outside regions of a shape, and require significant computation to recover this information [98]. As such, simulators that use triangle meshes for collision typically use complex continuous collision detection (CCD) methods that aim to strictly prevent interpenetration [171, 25, 159]. Alternatively, signed distance fields (SDFs) implicitly represent the surface of an object, but explicitly represent the interior and exterior regions



(a) (b) (c) (d)

Figure 38: **Limitations of Sampling**: Point sampling may miss contacts between sharp features in the SDF cone and the cloth geometry (a). Our method adaptively finds points of maximum penetration along each face, generating accurate face-vertex contacts (b). Point sampling also fails to capture edge-edge contacts between geometry and the SDF (c), our method naturally generates edge-edge contacts by optimizing over each face (d).

*and* the distance to the surface. Since interpenetrations may occur as a result of compliant contact models, or through numerical inaccuracy, having a robust way to recover from overlapping states makes SDFs particularly well suited for collision detection.

In general a signed distance field (SDF) may be considered as a function $\phi(\mathbf{x}) : \mathbb{R}^3 \rightarrow \mathbb{R}$ where $\mathbf{x}$ is a point in some previously defined coordinate frame. The value $\phi(\mathbf{x})$ represents the signed Euclidean distance of $\mathbf{x}$ to a point on the surface, where $\phi = 0$. By convention we define the interior of an object as the set of points where $\phi(\mathbf{x}) < 0$. A consequence of this definition is that the gradient $\nabla\phi$ points in the direction of maximum distance increase away from the surface. Given an SDF, the closest point on the surface to $\mathbf{x}$ may be calculated directly as $\mathbf{y} = \mathbf{x} - \nabla\phi(\mathbf{x})\phi(\mathbf{x})$. Determining collision between a point $\mathbf{x}$ against an object represented by an SDF is then as simple as evaluating $\phi(\mathbf{x})$ and checking the sign. Depending on the choice of contact model, a contact normal may be defined as $\mathbf{n} = \nabla\phi^T$, which can be used as input for a penalty or constraint-based contact solver such as the ones developed in the previous sections.

Although point-based contact is well suited for use cases such as particle systems, it is less clear how to extend the method to handle continuous surfaces such as triangle meshes. Previous work has proposed performing an offline sampling of the surface geometry at discrete points. Then, at runtime, checking each point for overlap as described above. While this approach is conceptually simple, it suffers from the problem that any discrete sampling may be insufficient to detect overlap for particularly sharp features, for example the spikes of a dragon, as shown in Figure 37, or the apex of a cone interpenetrating a piece of cloth, as shown in Figure 38 (a). In addition, point-sampling fails to capture the case of edge-edge contacts between bodies, for example between two boxes resting on each other, as in Figure 38 (c). While it is possible to increase point-sampling density, the contact locations remain fixed at discrete points on the

surface which can cause jumps as contacts shift from one point to the next. Furthermore, as the number of samples increases, so does the number of contacts generated. This shifts the computational burden to the contact solver, which may have high computational complexity.

In this work we propose a method to generate contacts between triangle mesh faces and edges in an adaptive and continuous manner. Our method is based on a local optimization over mesh edges or faces using constrained convex optimization. Unlike fixed, discrete sample points, this approach allows contacts to vary smoothly over the mesh elements, capturing sharp point-face and edge-edge contacts.

In summary, we make the following contributions:

- A method for generating smoothly varying contacts between mesh features and shapes represented by signed distance fields (SDFs)
- An analysis and comparison of three numerical methods for solving the local constrained optimization
- The application of our method to contact generation between thin-shells, rigid bodies, and deformable solids

## 6.1 Related Work

Signed distance fields, and implicit surfaces have a long history in computer graphics for shape modeling and rendering [20, 83, 21, 230]. In this paper we focus on the use of SDFs for collision detection or contact generation, and discuss some previous work below.

**Collision Detection** Fuhrmann et al [69] leveraged SDFs for collision detection between cloth and complex geometry. They recognized the limitations of point-sampling the cloth surface, and proposed adding samples at the midpoint of each edge to minimize the chance of missed collisions. Our method generalizes this idea to adaptive and smoothly varying face and edge contacts. Guendelman [77] showed how SDFs may be used for collision detection between non-convex rigid bodies with an impulse-based contact solver. They combined vertex sampling with edge-based isosurface intersection. We extend their work by capturing point-face contacts where the SDF may not have an embedded mesh representation. In addition, our method allows generating edge-edge contact constraints before intersection occurs through a local minimization. Xu et al. [232] also used an SDF representation with point-sampling for rigid body contact, but with an implicit penalty-based contact model. Their work was later extended to the case of continuous

collision detection (CCD) between shapes represented by SDFs [231]. Volume contact models [5, 223] share some similarities with SDFs representations. Volume contact methods work by identifying the overlapping volume of shapes and introducing constraints to remove/minimize it. In this work we are concerned with how to generate contacts between an arbitrary mesh and a solid represented by an SDF. In contrast to volume based approaches, our method also handles the case of thin-shells, e.g.: cloth, colliding against a rigid body. Weidner et al. [226] propose a hybrid Eulerian-on-Lagrangian with remeshing [153] to allow cloth to slide over sharp features and conform closely to the collision shape. For simplicity we do not perform remeshing, but generate smoothly varying contacts along the simulation mesh faces.

**Deformable Bodies** There is a large body of work on collision detection between deformable bodies [208]. Surface-based methods typically aim to detect and prevent interpenetration between surface elements using a combination of continuous collision detection (CCD) and fail-safe methods [171, 25, 183, 82, 197]. One advantage of SDFs is that, they provide robust inside/outside information, thus if penetration occurs it can often be recovered from. SDFs are usually considered static, although some authors have extended them to handle runtime deformations. Seyb et al. [186] recently applied deformed sphere tracing for particle collisions. Fisher and Lin [64] proposed a method to warp SDFs based on a tetrahedral embedding for collision against deformable objects. McAdams et al. [148] use a similar idea with point sampling of a surface mesh to perform character self-collision. We apply our optimization-based contact generation method to these approaches for accurate and robust contact between deformable bodies and cloth.

**Representation** SDFs are commonly discretized and stored on a grid, or volume texture. To avoid the memory overhead for dense volumes, adaptive storage methods have been proposed [68, 129]. Non-manifold and sparse representations have also been proposed to allow representing thin features efficiently [149]. Koschier et al. [112] proposed a highly accurate SDF representation based on hp-adaptive grids. They used the resulting representation for collision detection by point-sampling the surface. Our method is agnostic with regard to the underlying SDF representation, and may be used with analytic, dense, or sparse representations.

## 6.2 Face Contact

We first consider the case of colliding a single triangle against a solid body represented as an SDF. We use the *closest point* methodology for generating contact points [59] which requires finding the closest points on the face and the SDF isosurface. For a triangle with vertices

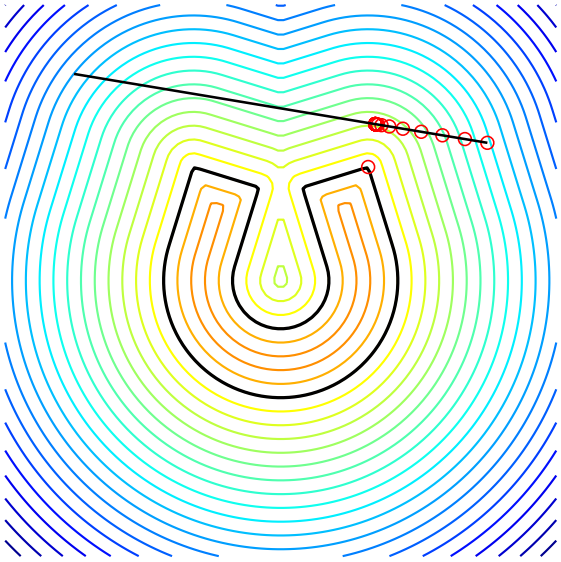


Figure 39: **Optimization Procedure**: We find the closest point between a mesh element (a line segment), and the surface of a shape represented by a signed distance field (horseshoe). Here we visualize the iterates (red dots) to show the progress of our optimization-based procedure.

$\mathbf{p}, \mathbf{q}, \mathbf{r} \in \mathbb{R}^3$, the closest point on the triangle to the rigid body is given by the solution to the following minimization over the barycentric coordinates $u, v, w$,

$$\underset{u,v,w}{\text{argmin}} \quad \phi(u\mathbf{p} + v\mathbf{q} + w\mathbf{r}) \tag{137}$$

$$\text{s.t.} \quad u, v, w \geq 0 \tag{138}$$

$$u + v + w = 1. \tag{139}$$

The constraints ensure that the solution to this problem lies in the triangle interior or on its boundary. In the following sections we examine some numerical methods for solving this optimization problem.

### 6.2.1 Projected Gradient Descent

SDFs may represent arbitrary shapes, which means the objective (137) in the preceding minimization is in general a nonlinear, and non-convex function of arbitrary complexity. Global optimization of such functions is difficult, and may require sophisticated methods such as branch and bound approaches [91]. An alternative method that often works well to find a *local minimum* (or at least a stationary point) is projected gradient descent [175, 176]. To apply gradient

descent we need the derivative of the distance field with respect to the barycentric coordinates, which is given by:

$$\mathbf{d} = \begin{bmatrix} \frac{\partial \phi}{\partial u} & \frac{\partial \phi}{\partial v} & \frac{\partial \phi}{\partial w} \end{bmatrix}^T = \begin{bmatrix} \nabla \phi(\mathbf{x})\mathbf{p} \\ \nabla \phi(\mathbf{x})\mathbf{q} \\ \nabla \phi(\mathbf{x})\mathbf{r} \end{bmatrix} \tag{140}$$

where the spatial point $\mathbf{x}(u, v, w) = u\mathbf{p} + v\mathbf{q} + w\mathbf{r}$ is simply a barycentric interpolation of the triangle vertices. The SDF gradient $\nabla \phi(\mathbf{x})$ w.r.t. spatial coordinates may be computed on-demand using finite differences, analytic gradients, or precomputed and stored on a grid. After computing the gradient $\mathbf{d}$ w.r.t. barycentric coordinates, we iteratively update our candidate solution $\mathbf{c}$ according to a fixed step size $\alpha$, and project onto the constraint manifold, using the fixed-point iteration

$$\mathbf{c}_{i+1} \leftarrow \mathbb{P}(\mathbf{c}_i - \alpha \mathbf{d}). \tag{141}$$

where $\mathbf{c} = [u, v, w]^T$ is the vector of barycentric coordinates, and $i$ the iteration index. We visualize the progress of this method on a 2D problem in Figure 39. The projection operator $\mathbb{P}$ maps a point to the closest point on the constraint manifold described by (138)-(139). Geometrically, these constraints represent a triangle centered around the origin in 3D with vertices $[1, 0, 0], [0, 1, 0], [0, 0, 1]$. The projection therefore, amounts to finding the closest point on this triangle from our current iterate, for which efficient codes exist [56]. We note that it is also possible to simplify the problem by expressing $w$ in terms of $u$ and $v$ such that the closest point projection can now be performed in 2D instead of 3D.

### 6.2.2 Frank-Wolfe

The Frank-Wolfe, or *conditional gradient method* is an iterative algorithm for solving constrained convex optimization problems [67, 99]. Similar to gradient descent, Frank-Wolfe only requires access to first order derivatives, however it has the appealing property that it does not require a projection step onto the constraint manifold. The Frank-Wolfe algorithm proceeds by iteratively solving the following minimization for the point $\mathbf{s}_i \in \mathbb{R}^3$,

$$\underset{\mathbf{s}_i}{\text{argmin}} \quad \mathbf{s}_i^T \nabla \phi(\mathbf{x}_i) \tag{142}$$

$$\text{s.t.} \quad \mathbf{s}_i \in \mathcal{D}, \tag{143}$$

followed by the update:

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \alpha(\mathbf{s}_i - \mathbf{x}_i). \tag{144}$$

here $\mathcal{D}$ is the domain to optimize over, in our case the triangle itself. Unlike projected gradient descent which optimizes over barycentric coordinates $\mathbf{c}$, for Frank-Wolfe it is more natural to optimize over spatial coordinates ($\mathbf{x}$) directly. Due to convexity of the triangle, the solution $\mathbf{s}_i$ must be one of the vertices $\mathbf{p}, \mathbf{q}, \mathbf{r}$, which can be computed by direct enumeration. This is similar to the problem of finding the support point for a given direction in the Gilbert-Johnson-Keerthi (GJK) algorithm [72]. Once this extremal point $\mathbf{s}_i$ has been identified, the Frank-Wolfe method performs a step (144) where the step length has the particular form $\alpha = \frac{2}{i+2}$ to ensure convergence. Although Frank-Wolfe is designed for convex functions we found it to be effective for non-convex optimization.

### 6.2.3 Culling and Starting Iterate

To quickly reject triangles from processing we evaluate the distance at the centroid $\mathbf{x}_c$ of the triangle and check if it is less than the radius $r$ of the bounding sphere centered at this point, i.e.: $\phi(\mathbf{x}_c) < r$. The same optimization applies for mesh edges as discussed in Section 6.3. We found this to be a highly effective early out for rigid body contact, however as may be expected, it is less effective for cloth in close contact with a body as illustrated in Figure 49.

The choice of starting iterate is important to the convergence of iterative methods. A simple heuristic we found effective is to evaluate $\phi$ at each triangle vertex and choose the point with the smallest value. This is also helpful in avoiding local minima, although it is not sufficient to guarantee a global solution.

### 6.2.4 Termination Conditions

Since a lower bound on the distance for each element is not known in advance we cannot define an absolute tolerance on the distance function $\phi(\mathbf{x})$. However, for projected gradient descent we may define a stopping criteria based on the magnitude of the gradient $\mathbf{d}$. For Frank-Wolfe the subproblem objective $\mathbf{s}_i^T \nabla\phi(\mathbf{x}_i)$ may be used, since for convex functions this is an upper bound on the primal error [99]. After termination, if the minimum distance lies within some threshold margin $\phi(\mathbf{x}) < \delta$ then we create a contact constraint. Otherwise, the element is deemed not to collide and no contact is generated.

Figure 40: **Edge Contact**: An example showing collision of 128 ropes consisting of a string of 1D elements. Point-based sampling misses edge-edge contacts leading to interpenetration across the sharp SDF edge (left). We use golden-section search to find the closest point along each segment to the isosurface (right).

## 6.3 Edge Contact

The face-based optimization presented in the previous section will naturally find contacts that lie on edges. However in many cases we wish to optimize over edges directly, for example when simulating one dimensional objects such as hair, or rope, as in Figure 40. Considering an edge (line segment) defined between two points $\mathbf{p}, \mathbf{q} \in \mathbb{R}^3$ we can define the closest point to the isosurface of an SDF as the solution to the single variable optimization problem:

$$\begin{aligned} \underset{u}{\text{argmin}} \quad & \phi(u\mathbf{p} + (1-u)\mathbf{q}) && (145) \\ \text{s.t.} \quad & u \geq 0 && (146) \\ & u \leq 1. && (147) \end{aligned}$$

Projected gradient descent and Frank-Wolfe can be applied directly to this problem, however the lower dimensionality makes search based methods attractive. We describe one such scheme in the next section.

Figure 41: **Cloth Net**: A suspended piece of cloth supporting sharp non-convex objects shows visible penetrations for simple point sampling approaches (left), while optimized face contacts were robust (right). In this example the cloth net consists of 4.8k faces, the rigid branches are represented as SDFs with embedded triangle meshes consisting of between 2-27k faces each. Collision detection time is around 350us per-timestep.

### 6.3.1 Golden-Section Search

A simple and robust method for solving interval-constrained optimization problems is *golden-section search* [110]. Golden-section search is a generalization of the root finding bisection method for minimizing unimodal functions. Although our SDF function may not be unimodal over the interval $[0, 1]$, golden-section search will terminate on one of the minima, or the boundary of the interval. An advantage of golden-section search is that it does not require gradients of the objective function, making it particularly efficient.

## 6.4 Models

We now discuss how to apply our presented methods to the case of collision detection for a range of common simulation models in computer graphics.

### 6.4.1 Cloth Collision

To collide deformable cloth represented by a triangle mesh against shapes represented as SDFs, we perform the local face optimization of Section 6.2 for each triangle in the mesh. This ap-

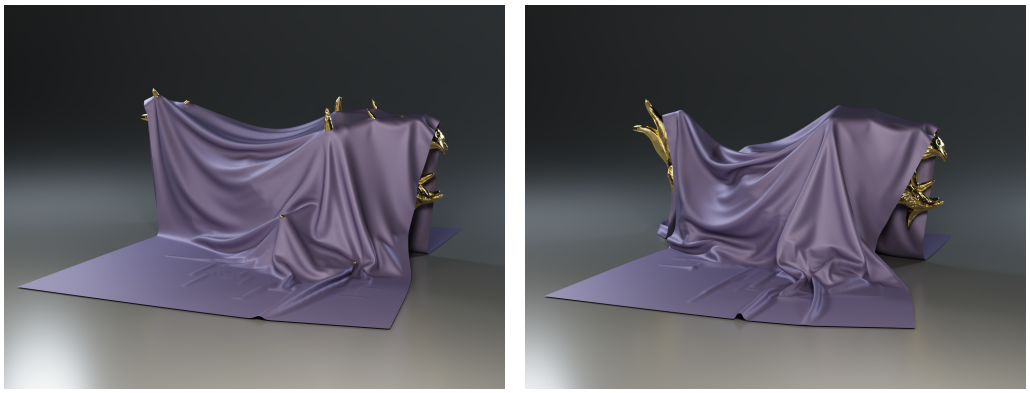


Figure 42: **Complex Geometry**: Point sampling shows visible penetrations through the sharp features of a dragon (left), our method prevents interpenetration by creating optimized local contacts (right). In this example the cloth consists of 20k triangles, collision detection against the SDF dragon takes around 80us per-timestep.

proach generates one contact per-triangle, which we found was sufficient for our tests, as illustrated in Figures 41 & 42. To avoid creating repeated contacts between connected faces we use the representative triangle approach, which uniquely assigns mesh features (vertices, edges, faces) to each element as a greedy preprocess [150, 41]. After optimization a contact may be discarded if it lies on a feature not assigned to the element.

### 6.4.2 Rigid Body Collision

For collision between rigid bodies that are well tessellated e.g.: the shells in Figure 43, we found that optimization over faces was sufficient to generate good contact manifolds. However, for shapes where this was not true, e.g.: the elongated boxes in Figure 38, we found that descent based methods would converge slowly, particularly for edge-edge contacts with high curvature. To avoid this problem we recommend to instead test vertices of both meshes directly as in Guendelman et al. [77], followed by the search based optimization of Section 6.3 over each edge.

### 6.4.3 Soft Body Collision

SDFs may be used for collision against deformable shapes by embedding the distance field inside a deformable cage [64, 148]. Given a point in world space coordinates $\mathbf{x}_s$, we associate a material space point $\mathbf{x}_m = \mathbf{g}(\mathbf{x}_s)$ where $\mathbf{g} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is an operator that maps from spatial coordinates back to the material pose, i.e.: an inverse displacement function. Given such a

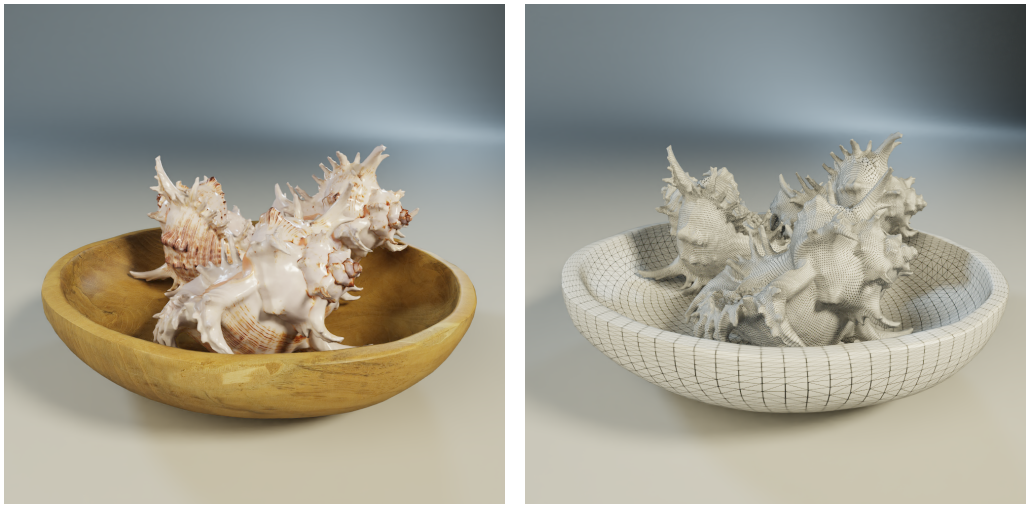
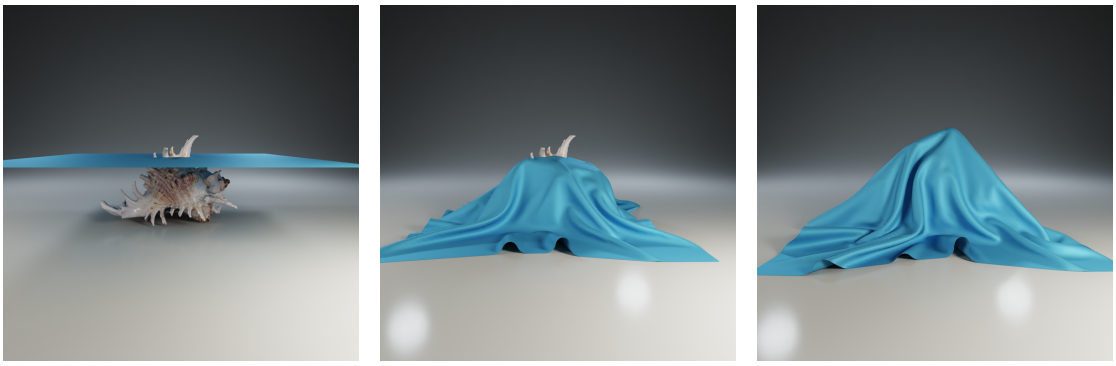


Figure 43: **Rigid Body Simulation**: A high resolution rigid body simulation where each shell consists of 129k triangles and contains many sharp features. Collision detection time using per-face optimization is 0.4ms on average.



(a) (b) (c)

Figure 44: **Interpenetration Robustness**: Given an initially interpenetrating state (a), triangle-based collision will not resolve intersections (b). SDFs provide inside/outside information that allows them to reliably separate objects (c).

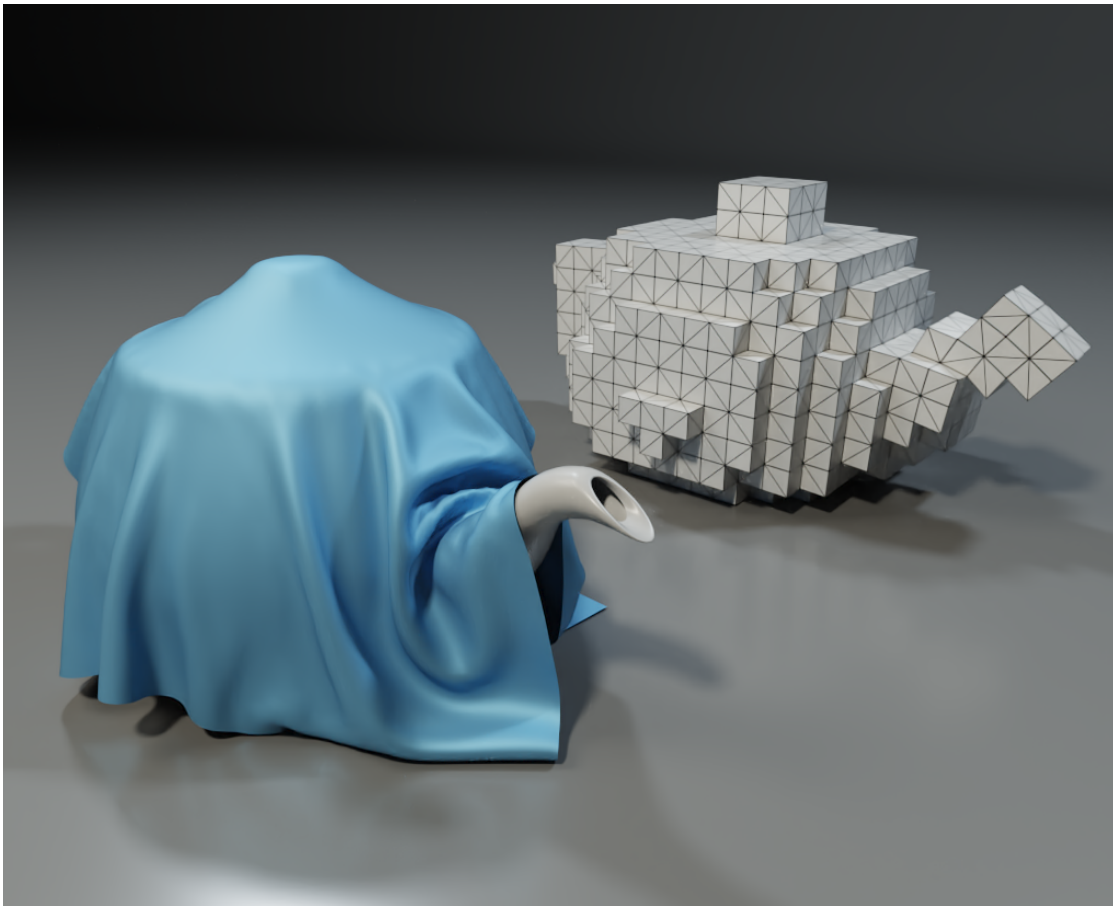


Figure 45: **Deformable Bodies**: Our method can be used with deformable SDF embeddings. Here cloth is collided against a deformable teapot using a tetrahedral FEM model. While the tetrahedral cage is relatively coarse (background), collision against the teapot is performed against the embedded high resolution SDF (foreground).

mapping we can define a deformed distance field,

$$\phi_d = \phi_m(\mathbf{g}(\mathbf{x}_s)), \tag{148}$$

where $\phi_m$ is the (static) distance field defined in material space. We note that the deformed field may no longer satisfy the distance field property $\|\nabla\phi_d\| = 1$, however the isosurface is not changed by this transformation, and it is sufficient to find a local minimum and define contact constraints. To apply our method to deformable bodies we follow the approach of McAdams et al. [148]. Given a deformed tetrahedral mesh, we first find the tetrahedron enclosing a world space point, and then compute the material space point associated with it. The enclosing tetrahedron with vertices $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ may be found efficiently using a bounding volume hierarchy (BVH) or other spatial data structure. Assuming linear shape functions, the projection back to material space is given by,

$$\mathbf{x}_m = \mathbf{g}(\mathbf{x}_s) = \mathbf{M}\mathbf{D}^{-1}(\mathbf{x}_s - \mathbf{v_0}_s) + \mathbf{v_0}_m. \tag{149}$$

We use the subscript $s$ and $m$ to denote world space and material space quantities respectively. Here $\mathbf{D} \in \mathbb{R}^{3\times3}$ is a basis for the enclosing deformed tetrahedron in world space, and $\mathbf{M} \in \mathbb{R}^{3\times3}$ is a basis for the undeformed tetrahedron in material space:

$$\mathbf{M} = \begin{bmatrix} \mathbf{v_1}_m - \mathbf{v_0}_m & \mathbf{v_2}_m - \mathbf{v_0}_m & \mathbf{v_3}_m - \mathbf{v_0}_m \end{bmatrix} \tag{150}$$

$$\mathbf{D} = \begin{bmatrix} \mathbf{v_1}_s - \mathbf{v_0}_s & \mathbf{v_2}_s - \mathbf{v_0}_s & \mathbf{v_3}_s - \mathbf{v_0}_s \end{bmatrix}. \tag{151}$$

Given the mapping function $\mathbf{g}$, and a world space point on a triangle defined by

$$\mathbf{x}_s(u, v, w) = u\mathbf{p} + v\mathbf{q} + w\mathbf{r}, \tag{152}$$

the gradient of $\phi_d(\mathbf{x}_s)$ w.r.t. the barycentric coordinates for optimization is given by,

$$\mathbf{d} = \begin{bmatrix} \frac{\partial\phi_d}{\partial u} & \frac{\partial\phi_d}{\partial v} & \frac{\partial\phi_d}{\partial w} \end{bmatrix}^T = \begin{bmatrix} \nabla\phi_m(\mathbf{x_m})\mathbf{G}(\mathbf{x_s})\mathbf{p} \\ \nabla\phi_m(\mathbf{x_m})\mathbf{G}(\mathbf{x_s})\mathbf{q} \\ \nabla\phi_m(\mathbf{x_m})\mathbf{G}(\mathbf{x_s})\mathbf{r} \end{bmatrix}, \tag{153}$$

where $\mathbf{G} = \nabla \mathbf{g} = \mathbf{M}\mathbf{D}^{-1}$. The matrix $\mathbf{G}$ may also be identified as the inverse of the deformation gradient commonly computed in tetrahedral FEM simulations [190]. As $\mathbf{D}$ is a basis for the deformed tetrahedron, if the element is extremely ill-conditioned calculating the inverse may not be possible. A robust implementation should detect this case and may choose to ignore such elements.

We note that $\mathbf{G}$ is a function of the world space point $\mathbf{x}_s$, and typically involves traversal of a BVH to determine the enclosing tetrahedron. To avoid performing this work for each optimization step, we evaluate $\mathbf{G}$ at the centroid of a triangle and treat it as constant over the course of the optimization. Following [148], once we have found a minimum of $\phi_d$ we project the minimum point back onto the surface (in material space), and apply the deformation mapping back to world-space to define the contact point. We illustrate this approach in Figure 45, where cloth is draped over a deformable teapot. A key advantage of this method is that it decouples the elastic discretization from the collision representation. This allows us to use a regular and relatively coarse hexahedral grid (decomposed into tetrahedra) for the computation of internal elastic dynamics, while using a high resolution SDF representation to resolve collision.

## 6.5 Discrete Distance Fields

SDFs are often stored as discrete samples on a regular or sparse grid [112]. This approach allows representing complex non-linear functions. However using a discretization with finite extents requires some special handling during contact generation. Since optimization can only proceed where there is valid SDF data, a problem occurs when there is not a sufficient margin between the surface of the shape and the extents of the SDF volume data. A possible solution to this is to project samples lying outside the volume's bounding box to the closest point on the box's surface. However we found it was sufficient to use a conservative sampling margin around the shape's isosurface, and for simple shapes to use the analytic approach suggested below.

## 6.6 Analytic Distance Fields

Closed form functions for the distance to simple shapes can often be found, and these can be efficiently combined to model complex geometry. Our method is independent of the SDF representation, so we can naturally support analytic field definitions. This allows us to collide meshes against arbitrarily smooth surfaces, without the need to form a surface triangulation. In addition, parameterized shapes can be animated to deform collision geometry over time as illustrated in Figure 46.

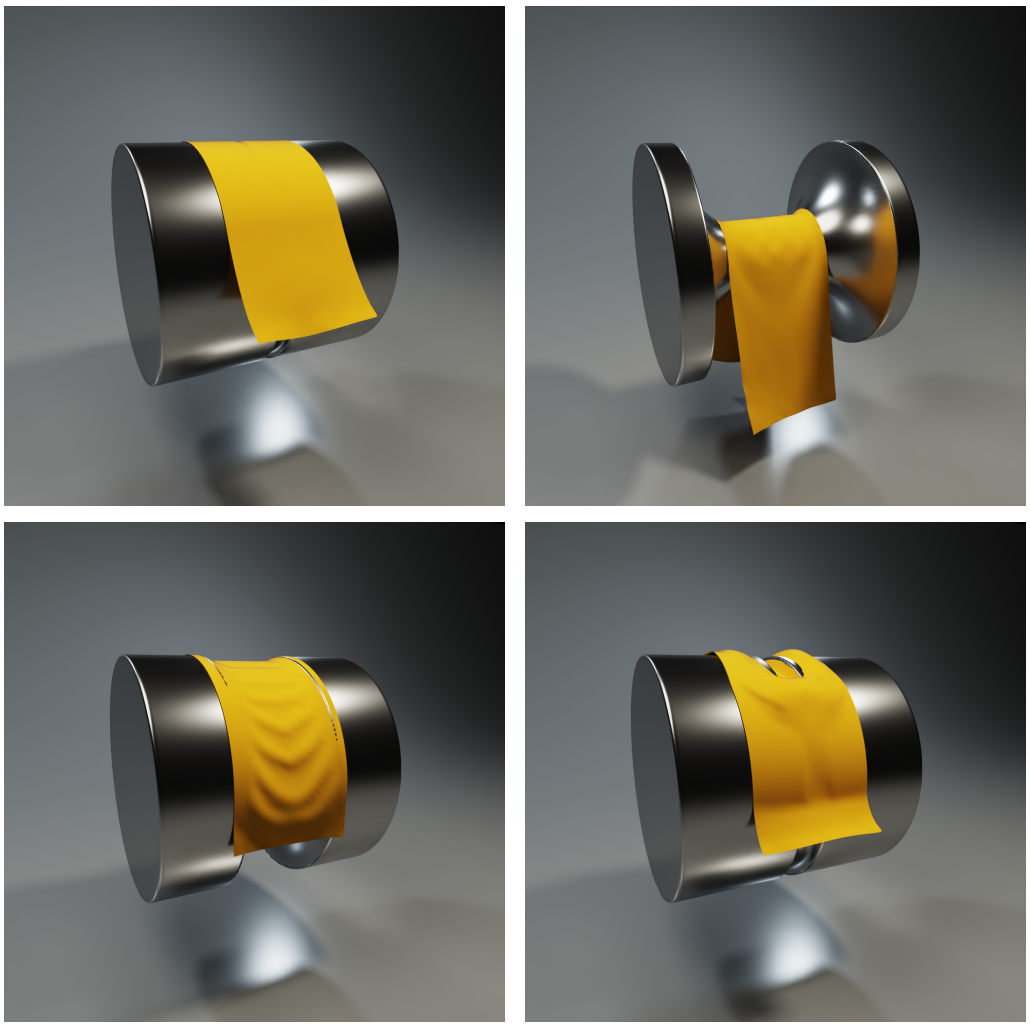


Figure 46: **Analytic Distance Fields**: Our method naturally supports collision against analytic SDF functions without the requirement for an embedded mesh representation. Top Row: Optimization-based contact generation against an analytic spool SDF. Bottom Row: Point-based sampling misses edge contacts, and eventually leads to cloth becoming separated and tangled.

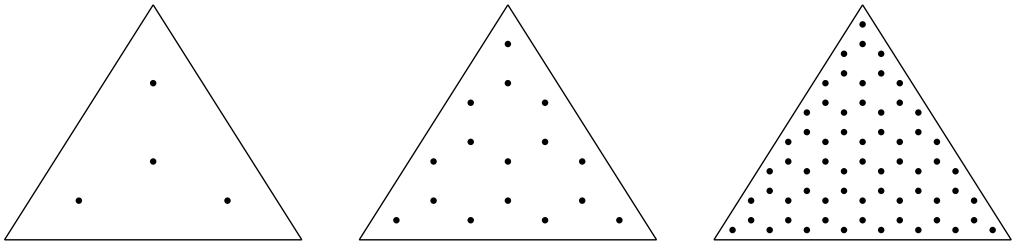


Figure 47: **Sampling Patterns**: Triangle point sampling for 4, 16, and 64 samples (left to right) using a low discrepancy sampling pattern. Despite providing well distributed coverage, discrete sampling may miss contacts between sharp features, and cannot provide smooth transitions for edge-edge contacts.

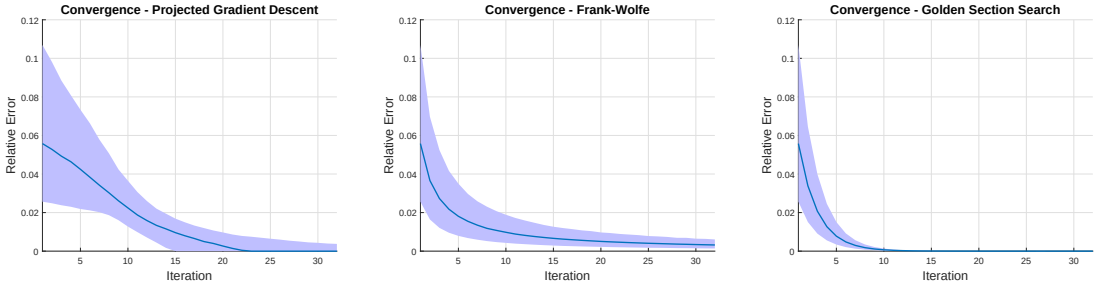



Figure 48: **Convergence Analysis**: We analyze the convergence of projected gradient descent (left), Frank-Wolfe (middle), and Golden-Section Search (right) over a random distribution of 1000 closest point problems. We randomly position a 2D segment over the domain (possibly intersecting the SDF) and plot the median, first and third quartile (shaded area) of the relative forward error at each iteration. To compute the relative error we use a ground truth solution obtained using high resolution brute-force sampling over the entire element.

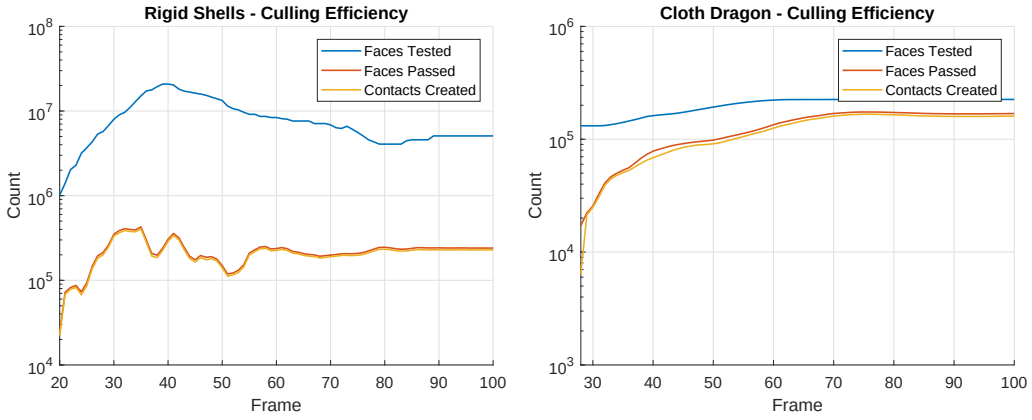



Figure 49: **Culling Efficiency**: Sphere culling faces against the SDF is highly effective for rigid body scenes (left). It is less effective for scenes where cloth is draped closely over an object (right). In both cases we see that the final number of contacts created is close to the number passing bounding sphere checks, indicating good culling efficiency.

## 6.7 Results

Since our method naturally parallelizes over mesh features we have implemented it in CUDA and run it on an NVIDIA GTX 2080 Ti. For dynamics simulation we use the extended position-based dynamics (XPBD) method [151, 138] where contacts are handled as hard unilateral constraints.

### 6.7.1 Convergence

To compare the convergence of the three methods we position a line segment randomly over the domain shown in Figure 39, and record the error at each iteration. Figure 48 shows the error distribution over 1000 random tests. Comparing the gradient-based methods, we see that projected gradient descent often reaches a lower error after the same number of iterations as Frank-Wolfe. However we note that the behavior of gradient descent is quite sensitive to the step-length used. In contrast, Frank-Wolfe uses a decreasing step length, which slows convergence but removes the need to set parameters, making it attractive from an authoring perspective. Observing the error plot for golden-section search in Figure 48 (right), we see it converges quickly on average, and has the tightest error distribution. Hence, if we have a 1D optimization problem where search methods are applicable we believe they should be preferred over descent based methods.

Descent-based optimization occurs using fixed step lengths in barycentric coordinates, this means that convergence is independent of the world-space size of the triangle. Rather, convergence rate primarily depends on the complexity and conditioning of the SDF function over the mesh feature. As is typical for descent based methods, minima in flat regions of low curvature will cause slower convergence.

### 6.7.2 Performance

In Table 4 we report the per-time step average collision times for our examples. Despite performing a local optimization loop per-triangle, collision detection times are still a fraction of a millisecond (we report timings in microseconds) for complex geometry. A surprising finding is that in many cases optimization-based collision detection performance was not significantly more costly than simple point sampling. We attribute this partly to effective early culling as shown Figure 49, but also to the fact that once SDF data enters the GPU cache it is fast to sample. This is even more apparent with the analytic test case shown in Figure 46, which has identical performance for all three methods. This is explained by the fact that, in this example, the optimization is purely compute-based (no memory fetches) and so total cost is small

Table 4: Performance timings comparing a simple point-based approach (Simple), to our face-based optimization. Here we have fixed the number of iterations to 32 for all methods, and report Golden-section search numbers for the rope example only, since it is a 1D problem, while the other examples have used face-based optimization.

| **Example** | **SDF** | **Points** | **Elements** | **Simple** | **PGD** | **FW** | **GS** |
|---|---|---|---|---|---|---|---|
| | # | # | # | $\mu$s | $\mu$s | $\mu$s | $\mu$s |
| Dragon | $256^3$ | 20,000 | 39,402 | 20 | 85 | 77 | - |
| Net | $256^3$ | 51,161 | 81,194 | 336 | 373 | 321 | - |
| Teapot | $128^3$ | 5,281 | 6,962 | 112 | 123 | 115 | - |
| Shells | $512^3$ | 635,020 | 1,269,880 | 401 | 445 | 437 | - |
| Rope | $256^3$ | 4224 | 4096 | 98 | 102 | 99 | 99 |
| Analytic | - | 10,000 | 19,502 | 48 | 48 | 48 | - |

compared to the rest of the kernel and falls below our ability to measure it.

### 6.7.3 Comparison To Sampling Approaches

One way to improve point sampled collision is to simply increase the number of samples. To compare our optimization method against this approach we generate face samples using the low-discrepancy sampling scheme of Basu and Owen [13] as illustrated in Figure 47. Using 64 face samples we were unable to obtain an interpenetration free result for the cone-cloth example. This illustrates how, for sharp features, even high sampling frequencies are insufficient to guarantee an interpenetration free state. On the cloth dragon example, which has a higher-resolution base cloth mesh, we found 4 additional face samples were sufficient to prevent interpenetration, however this approach generated approximately 80k contacts, compared with the 20k generated by face-based optimization.

### 6.7.4 Comparison To Surface Approaches

A popular method for collision detection is to perform contact generation between mesh surface features by examining triangle pairs, either in proximity, or using CCD checks. We compared against a surface-based contact generation scheme where point-face and edge-edge pairs are tested for proximity and a contact generated if the closest points lie below a distance threshold. We employ a GPU AABB tree to cull triangle pairs [107]. On the rigid shells example this mesh-based collision took approximately 15ms per-step, compared to less than 0.5ms using SDF-based contact. The performance difference may be explained by the fact each shell contains approximately 129k triangles, which results in deep hierarchy traversals. This is not

a perfect comparison, since SDFs implicitly resample the surface geometry (possibly losing detail), and because our approach generates exactly one contact per-triangle rather than one per-feature pair. Nevertheless, we found this result to be representative on a range of high resolution meshes. A more concerning limitation of surface-based approaches is that once interpenetration has occurred it is difficult to recover from. A primary advantage of SDFs is their ability to provide inside/outside information, allowing points to be projected to the surface. In Figure 44 we illustrate this by embedding cloth in a complex shape. Triangle-based contact remains entangled, while SDF-based contact can reliably separate both objects.

## 6.8 Limitations

While our method improves the robustness of mesh contact against SDFs, some issues remain. Since our method is local, it does not generate a global minimum translation distance (MTD) that would guarantee separation of already interpenetrating elements. In practice we found this was not a significant problem, and that deep penetrations are typically resolved within a few time-steps. A second limitation occurs when a triangle or edge is resting on multiple features (repeated local minima). Since we generate one contact per-element this may result in interpenetration. A possible solution would be to perform an implicit subdivision of each face. If local minima are isolated they can be used to bracket the element into smaller pieces and recursively search on these smaller subspaces.

## 6.9 Conclusion and Future Work

Due to their efficiency, flexibility, and robustness, we believe SDFs are well suited to shape representation for simulation. We have presented a local optimization-based technique that improves the quality of contact generation for meshes against SDFs. We have focused on first-order optimization methods for their simplicity. However, second-order methods such as Newton, quasi-Newton, or accelerated methods could also be employed to improve convergence rate. For future work we plan to explore contact generation between two solids both represented purely as SDFs, without the requirement for any mesh-based surface representation. Producing the intersection of two SDFs is straightforward, which suggests they may be well suited for volume-based contact approaches.

# 7 Robotics & Machine Learning

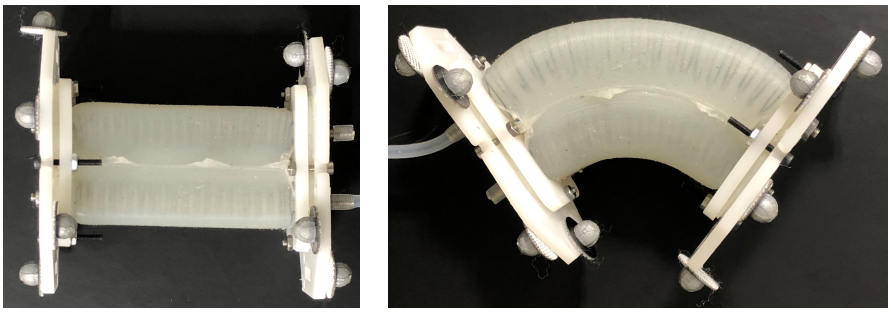
We now move our focus away from the development of simulation methods and on to their application in the domains of robotics and machine learning. In Section 7.1 we apply the techniques developed in previous chapters to the simulation and modeling of a soft robotic snake from a classical perspective of measurement and validation. In Section 7.2 we discuss an approach to closing the Sim-to-Real gap that relies on stochastic optimization, and in 7.3 we discuss an approach to system identification and control using model-based optimization that takes advantage of a fully differentiable simulator.

## 7.1 A Validated Physical Model For Real-Time Simulation of Soft Robotic Snakes

In this section we present a compliant constrained dynamics model for a soft snake robot [160, 135, 134]. Our model is able to accurately represent the deformations of the snake links while being efficient enough for real-time simulations. To achieve low-latency simulation, we use the GPU-based physics simulator based on the work presented in Section 4, that allows us to simulate coupled soft and rigid links, and leverages large-scale parallel iterative solvers to efficiently solve large systems. The resulting simulation is validated against a real robotic snake to verify that the deformation model is accurate and that the dynamics of the simulation match that of the real system. In summary, our main contributions are:

- A constrained dynamics model for a modular soft robotic snake that accurately represents a large range of deformations
- Model validation in static deformation and dynamic locomotion tests
- A simulation framework suitable for performing real-time control of soft robots.

This work was jointly lead by myself and Renato Gasoto. I was responsible for the development of the simulator and aspects of the physical modeling of the robot, while Renato was responsible for the physical experiments and validation against the real snake. I have omitted some details regarding modeling actuator latency that I was not directly involved with.



(a) Unpressurized (b) Pressurized

Figure 50: **Single Soft Link**: Each deformable snake chamber is made from silicone and wrapped with a fiber reinforcement, preventing it from increasing in radius when pressure is applied. The center of the link contains an inextensible layer that prevents it from expanding in length [160]. The spheres attached to the rigid plates are markers used for tracking the curvature in the experiments. Left: No pressure applied. Right: 8 psi applied on left chamber.

### 7.1.1 Related Work

The field of soft robots includes a large and varied range of designs that incorporate compliant materials and actuators. Soft robots may have a few flexible regions in them and may be driven by tendons [140], while completely soft pneumatic actuators can be driven by exerting a range of pressures within deformable bodies. The use of pressure leads to many intricate designs that exploit the material geometry to achieve the desired actuation [97, 222]. To allow a large range of pressures and material deformations, hybrid materials are often used on the pressure chambers, such as inextensible layers and fiber reinforcements [169]. Our modular soft snake robots use hybrid materials [135, 134] to achieve highly efficient 2D terrain locomotion.

The diversity of soft materials and difficulty in accurate modeling of soft robots lead to an increasing need for building a realistic simulation for deformable robots. Duriez et al. [53] presented a framework for simulating soft robots using a quasistatic approach based on FEM. In this work, we also use FEM with tetrahedral elements as a fundamental building-block. In addition, we combine these elements in a multiphysics system with spring networks, frictional contact, attachment constraints for soft/hybrid materials and articulated rigid bodies. We perform implicit time-integration to simulate dynamic trajectories. Recent work by Pozzi et al. [170] used a rigid-body model, fitted to an offline FEM simulation augmented with stiff springs to achieve real-time updates. In this work, we simulate the finite-element models and spring networks directly, using the large-scale parallelism of graphics processing units (GPUs) to achieve online update rates, thus avoiding an expensive offline simulation and data fitting stage.

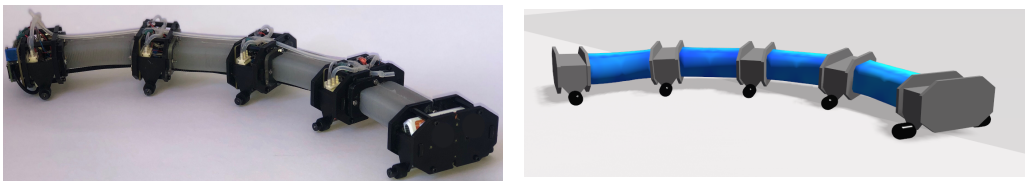


Figure 51: **Full Assembly**: The robotic snake with four links. The main controller receives wireless commands from the computer, and passes it over to each slave controller, which activates the solenoid valves that release the pressure to the soft actuator, on the right we show the snake in simulation.

### 7.1.2 Modeling

The snake is made of soft bending actuation modules, as shown in Figure 50a. The soft links of the snake robot are made of Ecoflex™ 00-30 silicone rubber which has material parameters $Y = 66.243$KPa, and $\nu = 0.4999$ [49]. To discretize the soft links we construct a triangular mesh of the surface and tetrahedralize it using TetGen [188].

In the center of the link, between the two chambers, there is a custom integrated curvature sensor, and a plastic film that inhibits linear extension. This set of constraints results in bending the entire soft module when one chamber is pressurized, as seen in Figure 50b. Since the inextensible layer in the center of the link has a deformation threshold that is beyond the range of forces to be applied on the soft links, it is acceptable to model it as a non-deformable constraint between particles along the center plane, which we achieve with zero-compliance springs. Similarly, the radial constraint on the chambers are defined as a set of inter-particle constraints over coplanar particles along the link. Although it would be possible to drive each link's expansion using surface pressure forces directly, the other constraints in the link allow us to simply control the chamber volume using constraints between particles along the primary axis of expansion. Only one chamber on the link is active (i.e.: pressurized) at a time, so this set of actuation constraints only applies to the expanding chamber. Figure 52 displays the constraints overlaid on the link. The link mesh was subdivided in 13 cross-sections along its length, in order to allow real-time computation, while maintaining good accuracy on the material deformation.

Caps are attached to both ends of the actuator to seal the chambers and allow modular connections with other segments. The caps are made of two ABS plates sandwiching the rim of the silicone. The links are then connected to each other through the rigid bodies that contain the electronics necessary to control the snake robot. In addition, the rigid bodies are attached to the wheels via. hinge joints. The wheels provide contact with the floor and model the anisotropic friction that a real snake has from its scales. The full assembly of a snake consisting of four links is shown in Figure 51, and the number of DOFs and constraints are given in Table 5.

Table 5: The number of elements of each type in a full snake assembly consisting of 4 soft links.

| **Type** | **Quantity** |
|---|---|
| Rigid Bodies | 15 |
| Rigid Joints | 10 |
| Particles | 1504 |
| Tetrahedra | 4536 |
| Attachments | 217 |
| Springs | 1460 |

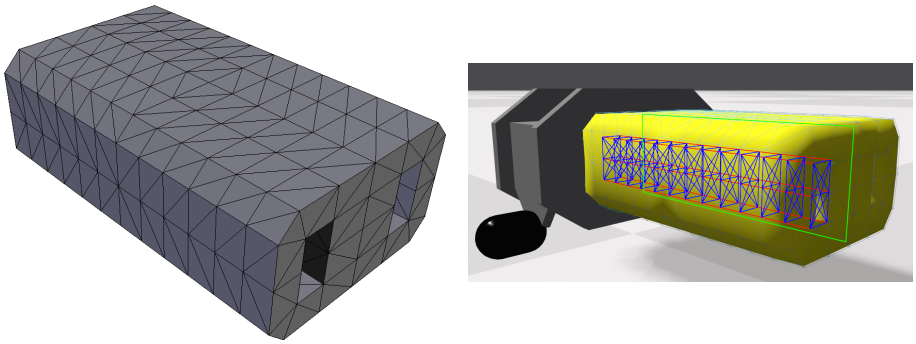


Figure 52: **Soft Link Discretization**: Left: Tetrahedral mesh for the soft-link. Right: Our constraint network. Green constraints represent stiff springs to limit chamber expansion. Blue constraints ensure the chamber radial perimeter is constant. Red constraints are used to expand chamber as overpressure is applied.

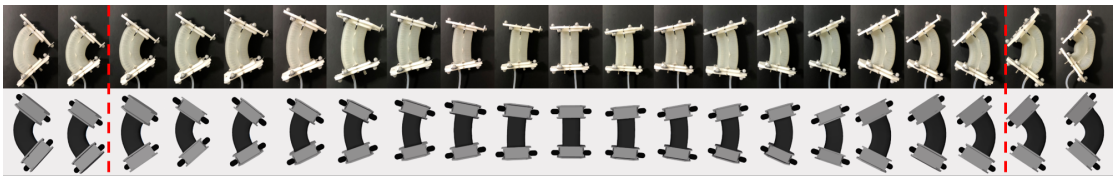


Figure 53: **Curvature Validation**: Measuring the link expansion after settling from -10 to 10 psi overpressure. Negative values mean the left chamber is inflated, while positive values are for the right chamber. The simulation displays high accuracy on the curvature up to 8 $psi$, where the dashed lines were traced. After that the pressure becomes excessive and the real link stops following the linear model.

### 7.1.3 Results

For all tracking experiments, the poses of the rigid links were captured using the motion capture system (MOCAP) by placing four markers on each rigid extremity so that we can collect their full poses. The MOCAP system contains 11 cameras surrounding the observable space of $4 \times 3$ m$^2$. Figure 50 shows the markers on the top corners of the rigid plates. This redundancy allows the information of links to be collected with high precision and minimal loss of tracking during the experiments. In order to eliminate remaining outliers, every experiment is repeated for 10 times unless otherwise mentioned.

We use a friction coefficient of $\mu = 1$ for all experiments, although this is relatively high, we observe little slipping between the wheels and the ground in the real world and did not find it necessary to optimize this value. We implement our simulator in CUDA using an NVIDIA GTX 1080 Ti GPU. We use a fixed time-step of $\Delta t = 0.0083s$, each time-step performs 4 Newton iterations, with each linear system solved approximately using 20 PCR iterations to ensure a fixed computational cost.

**Quasi-static Verification** The first experiment on the simulation is to verify whether the pressure actuator follows the same geometrical behavior as the real link. For this experiment, the curvatures of the real links were obtained by subtracting the yaw of the rigid connectors attached to each soft link, for the varying over-pressures (the pressure that exceeds the resting atmospheric pressure) from 0 to 10 psi, moving up with steps of 1 psi for both directions on the link, and averaged over 300 samples. Negative values show the inflation of left chamber and positive values are for the right chamber as visualized in Figure 53.

From Figure 54, it can be seen that the curvature increases linearly with the pressure within a range. Particularly, the spring actuation model is able to closely match the real curvature,

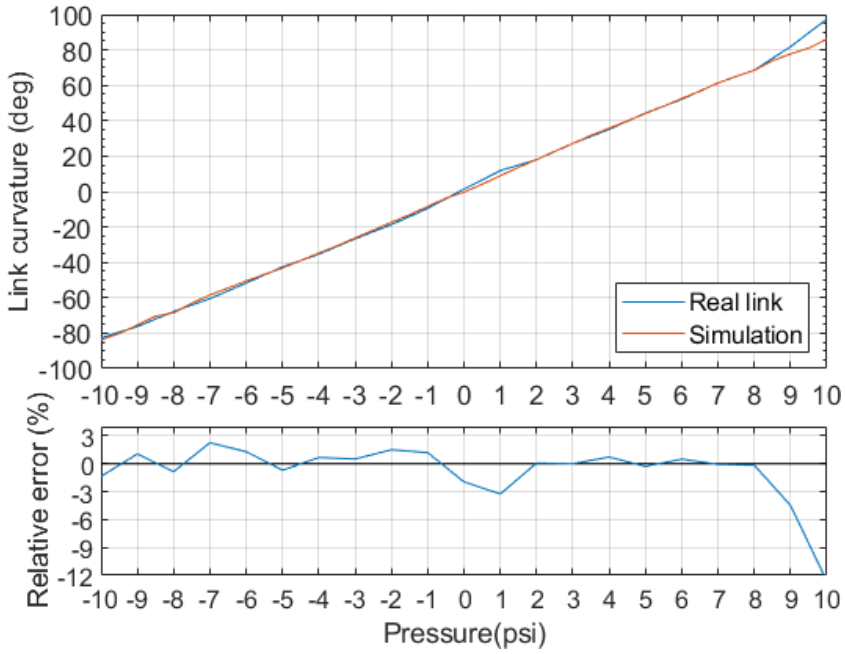



Figure 54: **Curvature Under Pressure**: The top plot shows that the curvature follows a linear relationship with the pressure applied. The bottom plot is the relative error.

and accurately follows the linear model up to 8psi. Ecoflex 00-30 Young modulus is 66 kPa, at 9 psi (62 kPa) it nears 2x expansion, which is the limit at which the material is linear. This behavior is also clearly observed by the relative error plot. When the pressure exceeds 8 psi, the real link starts to bulge over imperfections in the manufacturing, and on the opposite side, it folds in itself, resulting in a deviation from the linear model. At this pressure range there is a potential risk of damaging the links in the long term. For these reasons, it was deemed that the safe pressure threshold shall be 8 psi, and all the remaining tests were restricted to up to that range.

**Dynamic Verification** The simulator trajectory is then tested with an open-loop control and compared with the real snake robot using the following parameters. Since it's open-loop, it is expected that due to model inaccuracies and unmodeled dynamics the simulator will have error accumulated along the trajectory. As a result, the simulated trajectory may diverge from the actual trajectory over time. However, from Figure 55, it can be seen that starting from the same initial condition, the simulated trajectory closely matches the real trajectory for the execution time. The inclusion of an actuator latency model for the pressure update further improves the tracking of the center of mass with the real snake.

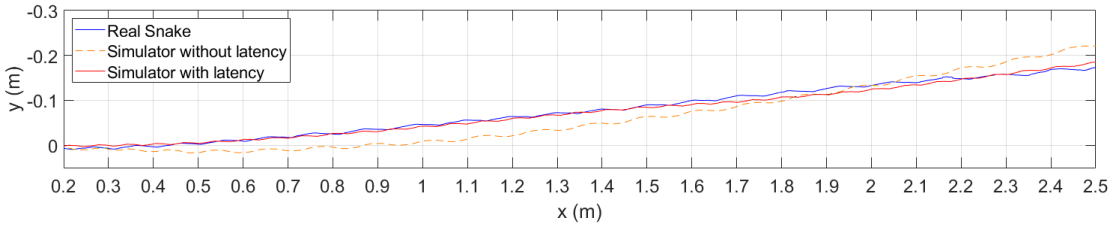



Figure 55: **Trajectory Comparison**: Motion capture data (blue), simulator without actuator latency (orange), and with latency modeling (red). We find our model is able to closely predict the trajectory of the snake over long distances.

Table 6: Timings for our simulator broken down into linear system assembly, and solver time. We report times in milliseconds (ms) as we increase the number of robot instances. Due to the complexity of the assembly a single instance of the snake is enough to almost saturate the GPU, nevertheless we see some weak-scaling which is optimal at 7 parallel copies.

| **Snakes** (#) | 1/4 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Assembly | 0.74 | 1.11 | 1.41 | 2.33 | 3.41 | 5.59 | 6.19 | 6.45 | 7.37 | 8.48 | 10.31 |
| Solve | 4.96 | 10.52 | 25.06 | 31.90 | 43.12 | 50.41 | 58.63 | 66.95 | 79.73 | 86.89 | 95.57 |
| Total | 5.70 | 11.63 | 26.47 | 34.23 | 46.53 | 56.00 | 64.82 | 73.4 | 87.10 | 95.37 | 105.88 |
| Total/Snake | - | **11.63** | 13.23 | 11.41 | 11.63 | 11.2 | 10.80 | **10.48** | 10.88 | 10.59 | 10.58 |

**Benchmarking** In order to test scalability of the system, we benchmark simulation times for a single soft link, a full snake with 4 links, and up to 10 snakes, each with 4 links. The linear system for a single Newton iteration corresponding to a single snake is a sparse matrix of roughly $30000 \times 30000$ with approximately $3.5 \times 10^5$ non-zeros. The total per-frame simulation times are in Table 6. From the results, it can be seen that simulation time increases linearly with the number of links and snakes. Linear scaling is expected, since a single link saturates the GPU. On average, it takes less than 12ms to simulate each snake, with optimal performance obtained when simulating seven snakes together. When compared to the CPU we found one full snake takes 8.9s per frame on a 3.3GHz Intel Core i7-5820K using the single-threaded Eigen Preconditioned Conjugate Gradient implementation [78].

### 7.1.4 Conclusion and Future Work

We have presented a dynamics model and simulation framework for a pneumatically actuated soft robotic snake robot. We validated the simulation against a real world robot, and found it produces real-time, high-fidelity results even in complex scenarios involving a mixture of hybrid soft bodies, rigid bodies, and frictional contacts. In the open loop control analysis our dynamic simulation remains in close agreement to the real snake trajectory. Our future research is to

develop a learning-based closed-loop controller to generate fast and stable snake gaits in a range of terrains, as well as obstacle-aided navigation, and learning specific motion primitives from demonstration [95, 96].

## 7.2 Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience

In the previous section we presented the results of a classical modeling, simulation, and validation approach on a specific robot using open-loop control. In this work, we focus on closing the reality gap by learning control policies over distributions of simulated scenarios. We present a reinforcement learning framework, *SimOpt*, that allows the joint optimization of control *and* physical parameters, improving policy transfer to the real-world.

In this work Yevgen Chebotar has been the primary author, and responsible for the majority of the paper and physical experiments. My contribution was the parallel GPU physical simulator, additional modeling, experimental set up, and the framework for performing domain randomization. I have omitted some algorithmic details I was not responsible for, and have focused on the results obtained using the simulator developed in Section 4.

### 7.2.1 Related Work

The problem of finding accurate models of the robot and the environment to facilitate the design of robotic controllers in the real world dates back to the original works on system identification [115, 73]. In the context of reinforcement learning (RL), model-based RL explored optimizing policies using learned models [46]. In [47, 48], the data from real world policy executions is used to fit a probabilistic dynamics model, which is then used for learning an optimal policy. Although our work follows the general principle of model-based reinforcement learning, we aim at using a simulation engine as a form of parameterized model that can help us to embed prior knowledge about the world.

The combination of system identification and dynamics randomization has been used in the past to learn locomotion for a real quadruped [204], non-prehensile object manipulation [133] and in-hand object pivoting [10]. In our work, we recognize domain randomization and system identification as powerful tools for training general policies in simulation. However, we address the problem of automatically learning simulation parameter distributions that improve policy transfer, as it remains challenging to do it manually. The closest work to ours are the methods that propose to iteratively learn simulation parameters and train policies in tandem. Tan et al. [203] propose an iterative system identification framework that is used to optimize trajectories of a bipedal robot in simulation, and calibrate the simulation parameters by minimizing the discrepancy between the real world and simulated execution of the trajectories. Zhu et al. [239] suggests optimizing the simulation parameters such that the value function is well approximated in simulation without replicating the real world dynamics. We also recognize that exact

replication of the real world dynamics might not be feasible, however a suitable randomization of the simulated scenarios can still lead to a successful policy transfer. Some works consider grounding the simulator using real world data. However, they may require a human in the loop to select the best simulation parameters [60], or require fitting of additional models for the real robot forward dynamics and simulator inverse dynamics [81]. Our work is closely related to the adaptive *EPOpt* framework of Rajeswaran et al. [174], who optimize a policy over an ensemble of models and adapt the model distribution using data from the target domain. *EPOpt* updates the model distribution by employing Bayesian inference with a particle filter, whereas we update the model distribution using an iterative KL-divergence constrained procedure. More importantly, they focus on simulated environments while in our work, we develop an approach that is shown to work in the real world and apply it to two real robot tasks.

### 7.2.2 Reinforcement Learning

We now briefly review the background of reinforcement learning for control policy optimization. Let $\tau = (s_0, a_0, \ldots, s_T, a_T)$ be a trajectory of states and actions, $P(s_t, s_{t+1}, a_t) \in \mathbb{R}^+$ a state-transition probability, and $R(\tau) = \sum_{t=0}^{T} \gamma^t R(s_t, a_t) \in \mathbb{R}$ the trajectory reward with discount parameter $\gamma$. The goal of reinforcement learning methods is to find parameters $\theta$ of a policy $\pi_\theta(a|s)$ that maximize the expected discounted reward over trajectories induced by the policy: $\mathbb{E}_{\pi_\theta}[R(\tau)]$ where $s_0 \sim p_0, s_{t+1} \sim P(s_{t+1}|s_t, a_t)$ and $a_t \sim \pi_\theta(a_t|s_t)$.

We define a distribution of simulation parameters $\xi \sim p_\phi(\xi)$ parameterized by $\phi$. As it was shown in [210, 178, 8], it is possible to design a distribution of simulation parameters $p_\phi(\xi)$, such that a policy trained on $P_{\xi \sim p_\phi}$ would perform well on a real world dynamics distribution. This approach is also known as *domain randomization*, and the policy training maximizes the expected reward under the dynamics induced by the distribution of simulation parameters $p_\phi(\xi)$:

$$\max_\theta \mathbb{E}_{P_{\xi \sim p_\phi}} \left[ \mathbb{E}_{\pi_\theta}[R(\tau)] \right]. \tag{154}$$

Domain randomization requires significant expertise and tedious manual fine-tuning to design the simulation parameter distribution $p_\phi(\xi)$. Furthermore, as we show in our experiments, it is often disadvantageous to use overly wide distributions of simulation parameters as they can include scenarios with infeasible solutions that hinder successful policy learning, or lead to exceedingly conservative policies. Instead, in the next section, we present a way to automate the learning of $p_\phi(\xi)$ that makes it possible to shape a suitable randomization without the need to train on very wide distributions.

### 7.2.3 Learning Simulation Randomization

The goal of our framework is to find a distribution of simulation parameters that brings simulated observations induced by the policy closer to the observations of the real world. Let $\pi_{\theta,p_\phi}$ be a policy trained under the simulated dynamics distribution $P_{\xi\sim p_\phi}$ as in the objective (154), and let $D(\tau_\xi^{ob}, \tau_{real}^{ob})$ be a measure of discrepancy between real world observation trajectories $\tau_{real}^{ob} = (o_{0,real} \ldots, o_{T,real})$ and simulated observation trajectories $\tau_\xi^{ob} = (o_{0,\xi} \ldots, o_{T,\xi})$ sampled using policy $\pi_{\theta,p_\phi}$ and the dynamics distribution $P_{\xi\sim p_\phi}$. The goal of optimizing the simulation parameter distribution is to minimize the following objective:

$$\min_{\phi} \mathbb{E}_{P_{\xi\sim p_\phi}} \left[ \mathbb{E}_{\pi_{\theta,p_\phi}} \left[ D(\tau_\xi^{ob}, \tau_{real}^{ob}) \right] \right]. \tag{155}$$

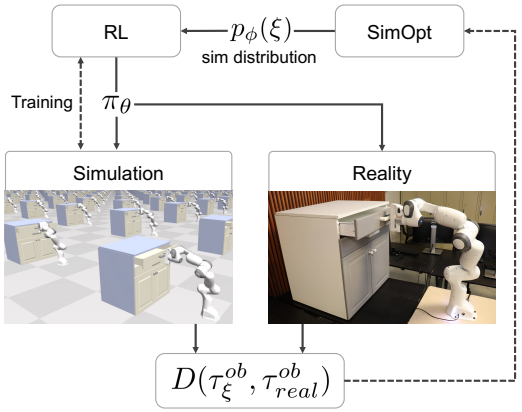
This optimization would entail training and real robot evaluation of the policy $\pi_{\theta,p_\phi}$ for each $\phi$. This would require a large amount of RL iterations and more critically real robot trials. Hence, we develop an iterative approach to approximate the optimization by training a policy $\pi_{\theta,p_{\phi_i}}$ on the simulation parameter distribution from the previous iteration and using it for both, sampling the real world observations and optimizing the new simulation parameter distribution $p_{\phi_{i+1}}$. This gives the following iterative optimization procedure:

$$\begin{aligned} &\min_{\phi_{i+1}} \mathbb{E}_{P_{\xi_{i+1}\sim p_{\phi_{i+1}}}} \left[ \mathbb{E}_{\pi_{\theta,p_{\phi_i}}} \left[ D(\tau_{\xi_{i+1}}^{ob}, \tau_{real}^{ob}) \right] \right] \\ &\text{s.t.} \quad D_{KL}\left(p_{\phi_{i+1}} \| p_{\phi_i}\right) \leq \epsilon, \end{aligned} \tag{156}$$

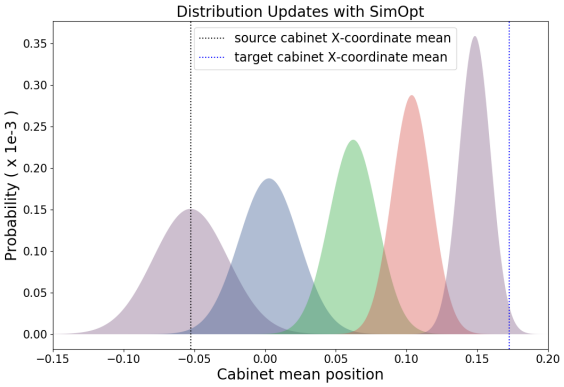
where we introduce a KL-divergence step $\epsilon$ between the old simulation parameter distribution $p_{\phi_i}$ and the updated distribution $p_{\phi_{i+1}}$ to avoid going out of the trust region of the policy $\pi_{\theta,p_{\phi_i}}$ trained on the old simulation parameter distribution. Figure. 56a shows the general structure of our algorithm that we call *SimOpt*.

For this work, we use a non-differentiable simulator and so we employ a sampling-based gradient-free algorithm based on relative entropy policy search [167]. This method is capable of optimizing the objective in Eq. 156 and is able to perform updates of $p_\phi$ with an upper bound on the KL-divergence step. In addition, the simulator can be treated as a black-box, since $p_\phi$ can be optimized directly by only using samples $\xi \sim p_\phi$ and the corresponding costs $c(\xi)$ coming from $D(\tau_\xi^{ob}, \tau_{real}^{ob})$. Sampling of simulation parameters and the corresponding policy roll-outs is highly parallelizable, which we use in our experiments to evaluate large amounts of parameter samples in parallel.

As noted above, single components of our framework can be exchanged. In case of avail-




(a) **Optimization Pipeline**: After training a policy on current distribution, we sample the policy both in the real world and for a range of parameters in simulation. The discrepancy between the simulated and real observations is used to update the simulation parameter distribution in SimOpt.




(b) **Updating Distributions**: The cabinet position distribution in the source environment, located at extreme left, slowly starts to change to the target environment distribution as a function of running 5 iterations of SimOpt.

ability of a differentiable simulator, the objective Eq. 156 can be defined as a loss function for optimizing with gradient descent. Furthermore, for cases where $\ell_1$ and $\ell_2$ norms are not applicable, we can employ other forms of discrepancy functions, e.g. to account for potential domain shifts between observations [217, 218, 184]. Alternatively, real world and simulation data can be additionally used to train $D(\tau_\xi^{ob}, \tau_{real}^{ob})$ to discriminate between the observations by minimizing the prediction loss of classifying observations as simulated or real, similar to the discriminator training in the generative adversarial framework [75, 90, 84]. Finally, a higher-dimensional generative model $p_\phi(\xi)$ can be employed to provide a multi-modal randomization of the simulated environments.

### 7.2.4 Experiments

In our experiments we aim to answer the following questions: First, how does our method compare to standard domain randomization? Second, how does our learned simulation parameter distribution compares to training on a very wide parameter distribution? Third, how many *SimOpt* iterations and real world trials are required for a successful transfer of robotic manipulation policies? And finally, does our method work for different real world tasks and robots?

To answer these questions we choose two robot manipulation tasks for evaluation: cabinet drawer opening and swing-peg-in-hole which we describe below.

Figure 57: **Manipulation Tasks**: Running policies trained in simulation at different iterations for real world swing-peg-in-hole and drawer opening tasks. Left: *SimOpt* adjusts physical parameter distribution of the soft rope, peg and the robot, which results in a successful execution of the task on a real robot after two *SimOpt* iterations. Right: *SimOpt* adjusts physical parameter distribution of the robot and the drawer. Before updating the parameters, the robot pushes too much on the drawer handle with one of its fingers, which leads to opening the gripper. After one *SimOpt* iteration, the robot can better control its gripper orientation, which leads to an accurate task execution.

**Swing-peg-in-hole** The goal of this task is to put a peg attached to a robot hand on a rope into a hole placed at a 45 degrees angle. Manipulating a soft rope leads to a swinging motion of the peg, which makes the dynamics of the task more challenging. The task set up in the simulation and real world using a 7-DoF Yumi robot from ABB is depicted in Figure. 57 (left). Our observation space consists of 7-DoF arm joint configurations and 3D position of the peg. The reward function for the RL training in simulation includes the distance of the peg from the hole, angle alignment with the hole, and a binary reward for solving the task.

**Drawer opening** In the drawer opening task, the robot has to open a drawer of a cabinet by grasping and pulling it with its fingers. This task involves an ability to handle contact dynamics when grasping the drawer handle. For this task, we use a 7-DoF Panda arm from Franka Emika shown in Figure. 57 (right). This task is operated on a 10D observation space: 7D robot joint angles and 3D position of the cabinet drawer handle. The reward function consists of the distance penalty between the handle and end-effector positions, the angle alignment of the end-effector and the drawer handle, the opening distance of the drawer and an indicator function ensuring that both robot fingers are on the handle.

We would like to emphasize that our method does not require the full state information of the real world, e.g. we do not need to estimate the rope diameter, rope compliance etc., to update the simulation parameter distribution in the swing-peg-in-hole task. The output of our policies consists of 7 joint velocity commands and an additional gripper command for the drawer opening task.

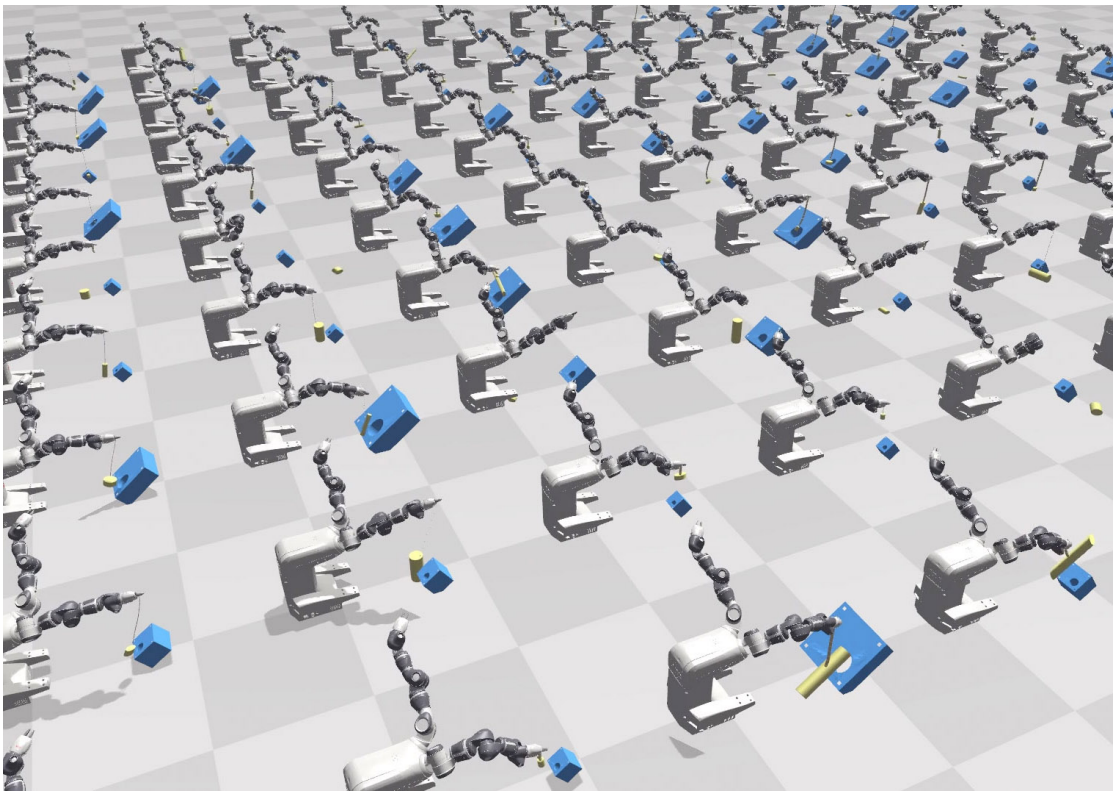


Figure 58: **Domain Randomization**: An example of a wide distribution of simulation parameters in the swing-peg-in-hole task where it is not possible to find a solution for many of the task instances.

### 7.2.5 Results

For simulation we use the framework developed in Section 4, since this is designed for parallelization we can simulate multiple instances of the scene on a single GPU. The RL training and *SimOpt* simulation parameter sampling is performed using a cluster of 64 GPUs for running the simulator with 150 simulated agents per GPU. In the real world, we use object tracking with DART [181] to continuously track the 3D positions of the peg in the swing-peg-in-hole task and the handle of the cabinet drawer in the drawer opening task, as well as initialize positions of the peg box and the cabinet in simulation. DART operates on depth images and requires 3D articulated models of the objects. We learn multi-variate Gaussian distributions of the simulation parameters parameterized by a mean and a full covariance matrix, and perform several updates of the simulation parameter distribution per-*SimOpt* iteration using the same real world roll-outs to minimize the number of real world trials.

Tables 7 and 8 show the initial mean, diagonal values of the initial covariance matrix and the final mean of the Gaussian simulation parameter distributions that have been optimized with *SimOpt* in drawer opening and swing-peg-in-hole tasks.

**Comparison to standard domain randomization** We aim at understanding what effect a wide simulation parameter distribution can have on learning robust policies, and how we can improve the learning performance and the transferability of the policies using our method to adjust simulation randomization. Figure 58 shows an example of training a policy on a significantly wide distribution of simulation parameters for the swing-peg-in-hole task. In this case, peg size, rope properties and size of the peg box were randomized. As we can observe, a large part of the randomized instances does not have a feasible solution, i.e.: when the peg is too large for the hole or the rope is too short. Finding a suitably wide parameter distribution would require manual fine-tuning of the randomization parameters.

**Swing-peg-in-hole** Figure. 57 (left) demonstrates the behavior of real robot execution of the policy trained in simulation over 3 iterations of *SimOpt*. At each iteration, we perform 100 iterations of RL in approximately 7 minutes and 3 roll-outs on the real robot using the currently trained policy to collect real world observations. Then, we run 3 update steps of the simulation parameter distribution with 9600 simulation samples per update. In the beginning, the robot misses the hole due to the discrepancy of the simulation parameters and the real world. After a single *SimOpt* iteration, the robot is able to get much closer to the hole, however not being able to insert the peg as it requires a slight angle to go into the hole, which is non-trivial to achieve using a soft rope. Finally, after two *SimOpt* iterations, the policy trained on a resulting simulation parameter distribution is able to swing the peg into the hole in $90\%$ of the times when evaluated on 20 trials.

We observe that the most significant changes of the simulation parameter distribution occur in the physical parameters of the rope that influence its dynamical behavior and the robot parameters that influence the policy behavior, such as scaling of the policy actions.

**Drawer opening** For drawer opening, we learn a Gaussian distribution of the robot and cabinet simulation parameters. Figure. 57 (right) shows the drawer opening behavior before and after performing a *SimOpt* update. During each *SimOpt* iteration, we run 200 iterations of RL for approximately 22 minutes, perform 3 real robot roll-outs and run 20 update steps of the simulation distribution using 9600 samples per update step. Before updating the parameter distribution, the robot is able to reach the handle and start opening the drawer. However, it cannot exactly replicate the learned behavior from simulation and does not keep the gripper orthogo-

| | $\mu_{init}$ | diag($\Sigma_{init}$) | $\mu_{final}$ |
|---|---|---|---|
| **Robot properties** | | | |
| Joint compliance (7D) | [-8.0 . . . -8.0] | 1.0 | [-8.2 . . . -7.8] |
| Joint damping (7D) | [-3.0 . . . -3.0] | 1.0 | [-3.0 . . . -2.6] |
| Joint action scaling (7D) | [0.5 . . . 0.5] | 0.02 | [0.25 . . . 0.44] |
| **Rope properties** | | | |
| Rope torsion compliance | 2.0 | 0.07 | 1.89 |
| Rope torsion damping | 0.1 | 0.07 | 0.48 |
| Rope bending compliance | 10.0 | 0.5 | 9.97 |
| Rope bending damping | 0.01 | 0.05 | 0.49 |
| Rope segment width | 0.004 | 2e-4 | 0.007 |
| Rope segment length | 0.016 | 0.004 | 0.017 |
| Rope segment friction | 0.25 | 0.03 | 0.29 |
| Rope density | 2500.0 | 8.0 | 2500.12 |
| **Peg properties** | | | |
| Peg scale | 0.33 | 0.01 | 0.30 |
| Peg friction | 1.0 | 0.06 | 1.0 |
| Peg mass coefficient | 1.0 | 0.06 | 1.06 |
| Peg density | 400.0 | 10.0 | 400.07 |
| **Peg box properties** | | | |
| Peg box scale | 0.029 | 0.01 | 0.034 |
| Peg box friction | 1.0 | 0.2 | 1.01 |

Table 7: **Swing-peg-in-hole**: simulation parameter distribution.

nal to the drawer, which results in pushing too much on the handle from the bottom with one of the robot fingers. As the finger gripping force is limited, the fingers begin to open due to a larger pushing force. After adjusting the simulation parameter distribution that includes robot and drawer properties, the robot is able to better control its gripper orientation and by evaluating on 20 trials can open the drawer at all times keeping the gripper orthogonal to the handle.

| | $\mu_{init}$ | diag($\Sigma_{init}$) | $\mu_{final}$ |
|---|---|---|---|
| **Robot properties** | | | |
| Joint compliance (7D) | [-6.0 . . . -6.0] | 0.5 | [-6.5 . . . -6.1] |
| Joint damping (7D) | [3.0 . . . 3.0] | 0.5 | [2.4 . . . 2.7] |
| Gripper compliance | -11.0 | 0.5 | -10.9 |
| Gripper damping | 0.0 | 0.5 | 0.34 |
| Joint action scaling (7D) | [0.26 . . . 0.26] | 0.01 | [0.19 . . . 0.35] |
| **Cabinet properties** | | | |
| Drawer joint compliance | 7.0 | 1.0 | 8.3 |
| Drawer joint damping | 2.0 | 0.5 | 0.81 |
| Drawer handle friction | 0.001 | 0.5 | 2.13 |

Table 8: **Drawer opening**: simulation parameter distribution.

### 7.2.6 Conclusion and Future Work

Closing the simulation to reality transfer loop is an important component for a robust transfer of robotic policies. In this work, we demonstrated that adapting simulation randomization using real world data can help in learning simulation parameter distributions that are particularly suited for a successful policy transfer without the need for exact replication of the real world environment. In contrast to trying to learn policies using very wide distributions of simulation parameters, which can simulate infeasible scenarios, we are able to start with distributions that can be efficiently learned with reinforcement learning, and modify them for a better transfer to the real world scenario. Our framework does not require full state of the real environment and reward functions are only needed in simulation. We evaluated our approach on two real world robotic tasks and showed that policies can be transferred with only a few iterations of simulation updates using a small number of real robot trials.

In this work, we applied our method to learning uni-modal simulation parameter distributions. We plan to extend our framework to multi-modal distributions and more complex generative simulation models in future work. Furthermore, we plan to incorporate higher-dimensional sen-

sor modalities, such as vision and touch, for both policy observations and factors of simulation randomization.

## 7.3 $\nabla$*Sim*: A Unified Treatment of Differentiable Physics and Rendering

Traditionally, physics simulation and rendering have been treated as disjoint, mutually exclusive tasks. In this section, we present $\nabla$*Sim*, a framework that takes a unified view on *simulation* in general, to mean physics simulation and rendering. Mathematically we can represent the entire simulation pipeline as a function $\mathbf{f} : \mathbb{R}^P \rightarrow \mathbb{R}^H \times \mathbb{R}^W; \mathbf{f}(\mathbf{p}) = \mathcal{I}$. Where $\mathbf{f}$ takes a vector $\mathbf{p} \in \mathbb{R}^P$ of input parameters and produces an image of dimensions $H \times W$ pixels. If this function $\mathbf{f}$ were differentiable, then the gradient of a function $\mathbf{f}(\mathbf{p}, t)$ with respect to the simulation parameters $\mathbf{p}$ intuitively tells us that perturbing $\mathbf{p}$ by an infinitesimal $\delta\mathbf{p}$ will change the output image of the simulation from $\mathcal{I}$ to $\mathcal{I} + \nabla\mathbf{f}(\mathbf{p}, t)\delta\mathbf{p}$.

$\nabla$*Sim* comprises two main components: a *differentiable physics engine* that computes and advances the physical state of the world at each time instant, and a *differentiable renderer* that renders the world to a 2D image. While differentiable versions of each component have been considered in isolation [216, 44, 196, 195, 45, 229, 85, 92], to the best of our knowledge we are the first to combine both differentiable simulation and rendering to solve tasks such as system identification, and control optimization. This is significant because in practice ground-truth observations from the real-world (e.g.: particle positions) are not available, or are laborious to gather, and the ability to infer system properties such as mass, inertia and elasticity *purely from video* would significantly reduce data requirements.

Since both physical simulation and rendering often contain discontinuous functions, it is not clear that it should be possible or meaningful to take the gradient of the entire pipeline. However, as we show, it is possible to obtain good quality gradient information over long time horizons, in a way that outperforms less accurate and more inefficient methods based on finite-differences. In addition, we leverage our previous work to simulate multiphysics environments that comprise rigid bodies, deformable solids, and thin-shell FEM.

This work was jointly lead by myself and Krishna Murthy. I was responsible for the development of the differentiable simulator, physical modeling, and experimental setup. I have omitted some details regarding some experimental details that I was not directly involved with.

### 7.3.1 Related Work

Differentiable physics simulators have seen significant recent attention and activity, with efforts centered around embedding physics structure into auto-differentiation frameworks. This has enabled differentiation through contact and friction models [216, 44, 196, 195, 45, 229, 85], latent

state models [76, 180, 101, 87], volumetric soft bodies [94, 93, 92], as well as particle dynamics [180, 122, 121, 92]. In computer graphics, Takahashi & Lin [202] addressed the problem of parameter estimation for fluid simulation using sampling-based optimization. They compare to gradient-based optimization using finite-differencing and report it to be less robust and efficient. In this work we leverage auto-differentiation to obtain analytic gradients, and find this approach provides more accurate results than finite difference methods such as REINFORCE [228], while providing good performance for large dimensional problems.

Bhat et al. [19] used simulated annealing to optimize cloth bending parameters, while Yang et al. [235] recover cloth material parameters from video using a supervised learning over large datasets. In contrast, we use differentiable rendering to recover material properties from single samples. Liang et al. [124] propose an efficient differentiable simulator for cloth dynamics with contact, while Qiao et al. [173] propose a mesh-based simulation framework for coupled rigid and cloth models. We also focus on mesh-based representations, and we extend the physical modeling flexibility by incorporating a tetrahedral FEM-based hyperelasticity models to simulate deformable solids and thin-shells.

### 7.3.2 Differentiable Rendering Engine

A renderer takes a scene description in the form of triangle meshes, materials, lighting and camera properties, and generates color image outputs. The rendering process is generally not differentiable, as visibility and occlusion events introduce discontinuities. Most interactive renderers, e.g.: for real-time applications, employ a rasterization process to project 3D geometric primitives onto 2D pixel coordinates, resolving these visibility events with non-differentiable operations. Our experiments employ two differentiable alternatives to traditional rasterization, SoftRas [130] and DIB-R [32], both of which rely on smoothing triangle edges by replacing their discontinuities with sigmoids. This has the effect of blurring triangle edges into semi-transparent boundaries, thereby removing the non-differentiable discontinuity of traditional rasterization. DIB-R distinguishes between foreground pixels (associated to the principal object being rendered in the scene) and background pixels (for all other objects, if any). The latter are rendered using the same technique as SoftRas while the former are rendered by bilinearly sampling a texture using differentiable UV coordinates. Given its efficiency benefits, we rely preferentially on DIB-R whenever rendering speed becomes a bottleneck.

While several other modern differentiable renderers have been proposed that model more complex light transport phenomenon [120, 155], such renderers are often computationally expensive, and for the purposes of this work we restrict ourselves to rasterization-based differentiable renderers [130, 32].

### 7.3.3 Differentiable Physics Engine

The dynamic evolution of the system is governed by a second order differential equations (ODE) of the form $\mathbf{M}(\mathbf{q}, \boldsymbol{\theta})\ddot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, \boldsymbol{\theta})$, where $\mathbf{M}$ is a mass matrix that may also depend on state and design parameters $\boldsymbol{\theta}$, and $\mathbf{f}$ is a force vector that may also be parameterized by $\boldsymbol{\theta}$, e.g.: stiffness coefficients. Differentiable solutions to ODEs of this type may be obtained through black box numerical integration methods using the continuous adjoint method [31]. However, we instead consider our physics engine as a discrete differentiable operation that provides an implicit relationship between a state vector $\mathbf{q}^- \equiv \mathbf{q}(t)$ at the start of a time step, and the updated state at the end of the time step $\mathbf{q}^+ \equiv \mathbf{q}(t + \Delta t)$. An arbitrary discrete time integration scheme can then be abstracted as the function

$$\mathbf{g}(\mathbf{q}^-, \mathbf{q}^+, \boldsymbol{\theta}) = \mathbf{0}, \tag{157}$$

relating the initial and final system state and the model parameters $\boldsymbol{\theta}$. By the implicit function theorem, if we can specify a loss function $l(\mathbf{q}^+)$ at the output of the simulator, we can compute $\frac{\partial l}{\partial \mathbf{q}^-}$ as $\mathbf{c}^T \frac{\partial \mathbf{g}}{\partial \mathbf{q}^-}$, where $\mathbf{c}$ is the solution to the linear system $\frac{\partial \mathbf{g}}{\partial \mathbf{q}^+}^T \mathbf{c} = -\frac{\partial l}{\partial \mathbf{q}^+}^T$, and likewise for the model parameters $\boldsymbol{\theta}$. This formulation is general enough to represent explicit, multi-step, or fully implicit time-integration schemes, however for this work we employ semi-implicit Euler integration for simplicity.

While the partial derivatives $\frac{\partial \mathbf{g}}{\partial \mathbf{q}^-}$, $\frac{\partial \mathbf{g}}{\partial \mathbf{q}^+}$, $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$ can be computed by hand, or through graph-based auto-differentation frameworks [165, 1, 24], program transformation approaches such as DiffTaichi [92], and Google Tangent [220] are particularly well-suited to simulation code. For our differentiable simulator we define an embedded subset of Python syntax to define kernel code, and at runtime generate C++/CUDA code for both the forward and backwards kernels. In Figure 59 we give an example of a simulation kernel definition.

To generate each kernel's *adjoint* its abstract syntax tree (AST) is parsed using Python's built-in `ast` module. We then generate C++ kernel code for forward and reverse mode, through a simple transformation [142], which may be compiled to a CPU or GPU executable using the PyTorch `torch.utils.cpp_extension` mechanism. This approach allows writing imperative code, with fine-grained indexing and implicit operator fusion (since all operations in a kernel execute as one GPU kernel launch). Each kernel is wrapped as a PyTorch autograd operation so that it fits natively into the larger computational graph. This enables simple interaction with existing differentiable renderers that rely on PyTorch input buffers.

```python
@kernel
def integrate_particles(
    x : tensor(float3),
    v : tensor(float3),
    f : tensor(float3),
    w : tensor(float),
    gravity : tensor(float3),
    dt : float,
    x_new : tensor(float3),
    v_new : tensor(float3)):

    # Get thread ID
    thread_id = tid()

    # Load state variables and parameters
    x0 = load(x, thread_id)
    v0 = load(v, thread_id)
    f0 = load(f, thread_id)
    inv_mass = load(w, thread_id)

    # Load external forces
    g = load(gravity, 0)

    # Semi-implicit Euler
    v1 = v0 + (f0 * inv_mass - g*step(inv_mass))*dt
    x1 = x0 + v1 * dt

    # Store results
    store(x_new, thread_id, x1)
    store(v_new, thread_id, v1)
```

Figure 59: **Particle Integration Kernel**: Using a subset of Python syntax we allow defining differentiable kernels using imperative parallel code that can operate directly on PyTorch tensors.


(a) Triangular FEM element

(b) Tetrahedral FEM element

Figure 60: **Mesh Discretization**: We use triangular (a) and tetrahedral (b) FEM models with angle-based and volumetric activation parameters, $\alpha$. These mesh-based discretizations are a natural fit for our differentiable rasterization pipeline, which is designed to operate on triangles.

### 7.3.4 Physical Models

We adopt a mesh-based discretization for deformable shells and solids. In both cases the surface is represented by a triangle mesh, enabling straightforward integration with our triangle mesh-based differentiable rasterizer.

**Deformable Solids** For solids, we use a tetrahedral FEM model as illustrated in Figure 60b. We use the hyperelastic NeoHookean constitutive model of Smith et al. [193]. This derives from a potential energy that is parameterized by the two Lamé coefficients

$$\Psi(\mathbf{q}, \theta) = \frac{\mu}{2}(I_C - 3) + \frac{\lambda}{2}(J - \alpha)^2, \tag{158}$$

where $I_C, J$ are invariants of strain. We introduce a per-element activation parameter $\alpha$, that controls a per-element volumetric activation that can be used to drive locomotion. We store activation and material parameters for all tetrahedra in differentiable tensors, allowing us to optimize w.r.t. to these properties directly.

**Deformable Shells** To model thin-shells such as clothing, we use constant strain triangular elements embedded in 3D. The same Neo-Hookean constitutive model above is applied to model in-plane elastic deformation, with the addition of a bending energy $\mathbf{f}_b(\mathbf{s}, \theta) = k_b \sin(\frac{\phi}{2} + \alpha)\mathbf{d}$, where $k_b$ is the bending stiffness, $\phi$ is the dihedral angle between two triangular faces, $\alpha$ is a per-edge actuation value that allows the mesh to flex inwards or outwards, and $\mathbf{d}$ is the force direction given in [26]. We also include a lift/drag model that approximates the effect of the surrounding air on the surface of mesh. For both solids and shells we use the strain-rate dissipation potentials of Sanchez & Otaduy [137] to model damping.

**Contact** As discussed in Section 4, implicit contact methods based on complementarity formulations of contact may be used to maintain hard non-penetration constraints [44]. However, we adopt the relaxed primal model of contact presented in Section 5 since this provides well-defined derivatives of arbitrary continuity, allowing us to apply black-box optimizers such as L-BFGS. For contact generation we use an all-pairs collision detection scheme that generates a conservative set of contacts at the start of each time-step. This process is non-differentiable, but provided that a conservative margin is used then the gradient over a single time-step is consistent.

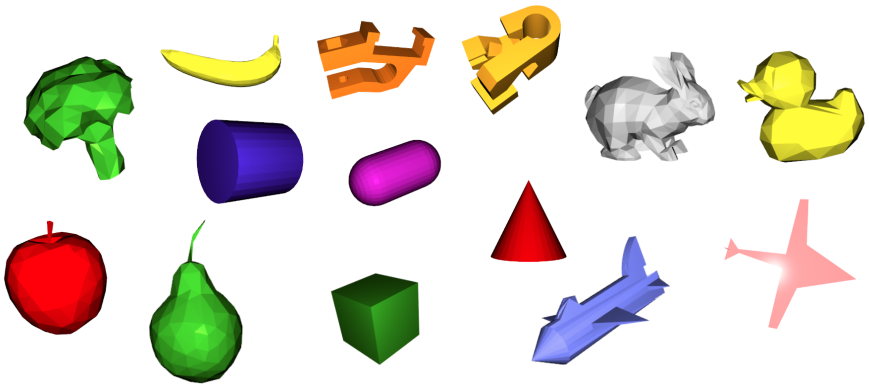


Figure 61: **Rigid Body Database**: Meshes used in our rigid body mass identification experiments. All of these meshes have been simplified to contain 250 vertices or fewer, for faster collision detection and differentiable rendering times.

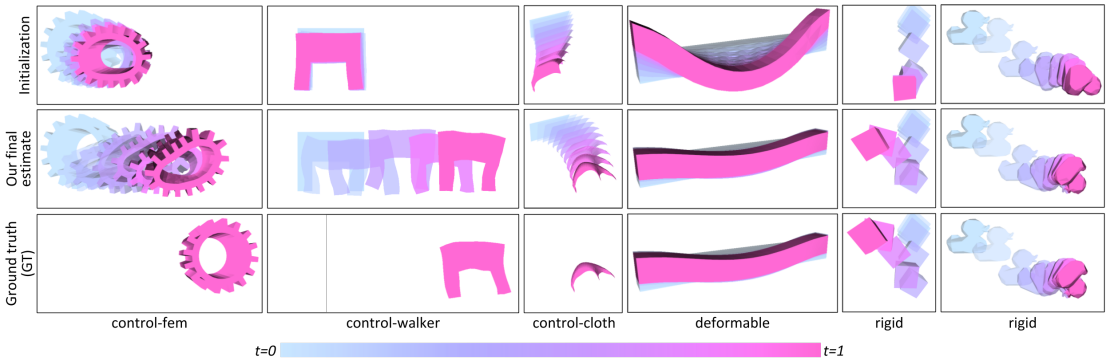



Figure 62: **Qualitative Results**: Our framework accurately estimates physical parameters for diverse, complex environments. For *control-fem* and *control-walker* experiments, we train a neural network to actuate a soft body towards a target image (GT). For *control-cloth*, we optimize the cloth's initial velocity to hit a target pose (GT) in the presence of nonlinear lift/drag forces. For *deformable* experiments, we optimize the material properties of a beam to match a video. In the *rigid* experiments, we estimate contact parameters (elasticity/friction) and object density to match a video (GT). We visualize entire time sequences with color-coded blends.

### 7.3.5 Parameter Estimation

We now discuss the application of $\nabla$*Sim* to tasks in system identification, where the goal is to recover physical model parameters directly from images.

**Rigid Bodies** We curated a dataset comprising of $14$ meshes, comprising primitive shapes such as boxes, cones, cylinders, as well as non-convex shapes from ShapeNet [29] and DexNet [139] as shown in Fig. 61. For each of the experiments, we pick an object at random, and sample its physical attributes from a predefined range: densities are in the range $[2, 12]\,\text{kg/m}^3$, contact parameters $k_e, k_d, k_f$ are in the range $[1, 500]$, and the coefficient of friction $\mu$ is in the range $[0.2, 1.0]$. The positions, orientations, (anisotropic) scale factors, and initial velocities are randomly uniformly sampled in a cube of side-length $13$m centered at the camera. Across all rigid-body experiments, we use $800$ objects for training and $200$ objects for testing.

We apply a *known* impulse to the object, and record a video of the resultant trajectory. Inference with $\nabla$*Sim* is done by picking an initial guess of the mass (at random), unrolling a differentiable simulation using this guess, comparing the rendered out video with the true video using pixelwise mean-squared error (MSE), and updating the initial guess by gradient descent over the unrolled computation graph. Table 9 shows the results for an initial control experiment, designed to evaluate the ability to predict the mass of an object from video, and knowing the impulse applied to it. We compare our image-based supervision to a *strong supervision* baseline where the exact 3D pose (translation, and rotation) of the body is provided to the optimizer. This strong supervision creates an advantage for pure simulation-based optimization. However, we found our combined video-based supervision is also able to very precisely estimate mass, to a relative absolute error of 9.01e-5. The "Average" and "Random" baselines predict the dataset mean and a uniformly sampled random number from the domain respectively.

To investigate whether analytical differentiability is required, we compare performance to a PyBullet [39] baseline, and apply a black-box gradient estimation technique REINFORCE [228]. Compared to using analytic gradients we found this approach to be particularly sensitive to several simulation parameters. In Table 10, we also estimate the parameters of our compliant contact model from video observations alone, and found that $\nabla$*Sim* is able to precisely recover the parameters of the simulation.

**Deformable Solids** One core strength of $\nabla$*Sim* is its ability to handle a rich class of objects, such as deformable solids and thin-shells (cloth). We conduct a series of experiments to investigate the ability to recover physical parameters of deformable solids from images. We demonstrate the ability of $\nabla$*Sim* to accurately recover the parameters of $100$ instances of randomly cho-

| **Approach** | **Mean abs. err. (kg)** | **Abs. Rel. err.** |
|---|---|---|
| Average | 0.2022 | 0.1031 |
| Random | 0.2653 | 0.1344 |
| PyBullet (REINFORCE) | 0.0928 | 0.3668 |
| Video supervision | 2.36e-5 | 9.01e-5 |
| Strong supervision | 1.35e-9 | 5.17e-9 |

Table 9: **Mass estimation**: $\nabla$*Sim*'s video-based approach obtains precise mass estimates, comparing favorably even with approaches that require strong 3D supervision. We report the mean absolute error and absolute relative errors for all approaches evaluated.

| | **mass** | **elasticity** | | **friction** | |
|---|---|---|---|---|---|
| **Approach** | $m$ | $k_d$ | $k_e$ | $k_f$ | $\mu$ |
| Average | 1.7713 | 3.7145 | 2.3410 | 4.1157 | 0.4463 |
| Random | 10.0007 | 4.18 | 2.5454 | 5.0241 | 0.5558 |
| Strong Supervision | 1.70e-8 | 0.036 | 0.0020 | 0.0007 | 0.0107 |
| Video Supervision | 2.87e-4 | 0.4 | 0.0026 | 0.0017 | 0.0073 |

Table 10: **Contact parameter estimation**: $\nabla$*Sim* estimates contact parameters (elasticity, friction) to a high degree of accuracy, despite estimating them from video. Strong supervision requires accurate 3D ground-truth at 30 FPS. We report absolute *relative* errors for each approach evaluated.

| | Deformable solid FEM | | | Thin-shell (cloth) |
|---|---|---|---|---|
| | Per-particle mass | Material properties | | Per-particle velocity |
| | $m$ | $\mu$ | $\lambda$ | $v$ |
| **Approach** | **Rel. MAE** | **Rel. MAE** | **Rel. MAE** | **Rel. MAE** |
| Strong Supervision | 0.032 | 0.0025 | 0.0024 | 0.127 |
| Image supervision | 0.048 | 0.0054 | 0.0056 | 0.026 |

Table 11: **Parameter estimation of deformable objects**: We estimate per-particle masses and material properties (for solid deformable objects) and per-particle velocities for cloth throwing. In the case of cloth, strong 3D supervision (based on the cloth center-of-mass reaching a target) actually performs worse than image-based since in this case the image provides more information on the target geometric configuration.

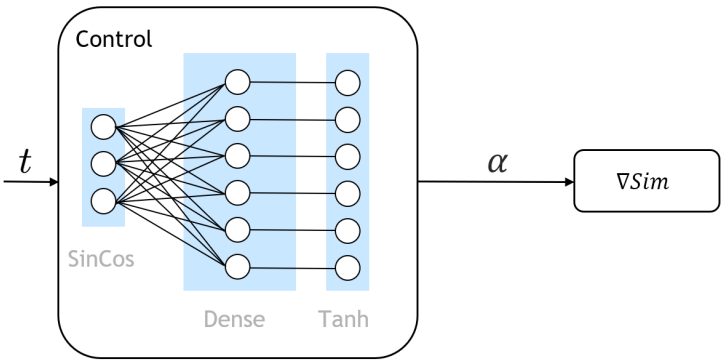



Figure 63: **Open Loop Controller**: A simple network architecture used the for generating time-varying volumetric actuation signals in the 2D and 3D Soft Walker examples.

sen deformable objects (bouncing spheres, beams), and report the mean absolute error (MAE) in Table 11.

### 7.3.6 Model Predictive Control

To investigate whether the gradients computed by $\nabla Sim$ are meaningful for vision-based tasks, we conduct a range of visual Model Predictive Control (MPC) experiments, involving the actuation of deformable objects towards an *image-based* target pose. While traditional *state-based* MPC requires a goal specification in 3D state space, visual MPC specifies a goal by means of an image of the desired end-configuration.

For our model predictive control tests, we design an experiment where a 3-layer neural network is trained to generate periodic activation signals with the goal of maximizing horizontal velocity. The network architecture is shown in Figure 63. With the simulation time $t$ as input we generate $N$ phase-shifted sinusoidal signals which are passed to a fully-connected layer (zero-bias), and a final activation layer. The output is a vector of per-element activation values that control the volumetric expansion and contraction of each element.

**2D Soft Walker** The first example involves a 2D walker model, inspired by DiffTaichi [92]. We represent the walker as a FEM triangle mesh (104 elements), and train a neural network (NN) control policy to actuate the walker to reach a target pose on the right-hand side of an image. Our NN consists of one fully connected layer and a $\tanh$ activation as illustrated in Figure 63. Each triangle uses elasticity parameters of $\mu = 10^4$, $\lambda = 10^4$, and contact parameters of $k_e = 10^4$, $k_d = 10^3$, $k_f = 10^3$, $k_\mu = 0.5$, and is allowed to change its area by a maximum of $20\%$, to

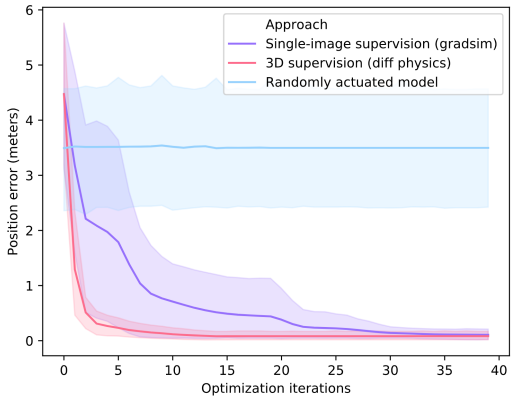
encourage physically-plausible solutions. We include an activation penalty of $l_{cost} = 0.1\|\boldsymbol{\alpha}\|_2$, where $\boldsymbol{\alpha}$ is the vector of scalar activation values for each time-step. $\nabla$*Sim* is able to learn an effective policy in only 3 iterations of gradient descent, by minimizing a pixelwise MSE between the last frame of the rendered video and the goal image as shown in Fig. 62 (column 2, bottom).

**3D Soft Wheel** In our second test, we formulate a more challenging 3D control problem where the goal is to actuate a soft-body FEM object (a *wheel*) consisting of 1152 tetrahedral elements to move to a target position as shown in Fig. 62 (left). We use the same NN architecture as in the 2D walker example, and use the Adam [111] optimizer to minimize our pixelwise MSE metric. We use elasticity paramters of $\mu = 10^3$, $\lambda = 10^3$, and contact parameters of $k_e = 10^4$, $k_d = 10$, $k_f = 10^3$, $k_\mu = 0.5$. To evaluate $\nabla$*Sim* accuracy we train a baseline model that uses strong supervision and minimizes the MSE between the target position and the precise 3D location of the center-of-mass (COM) of the FEM model at each time step (i.e.: a *dense* reward). We test both strong and image-based supervision against a naive baseline that generates random activations and plot convergence in Fig. 64a.

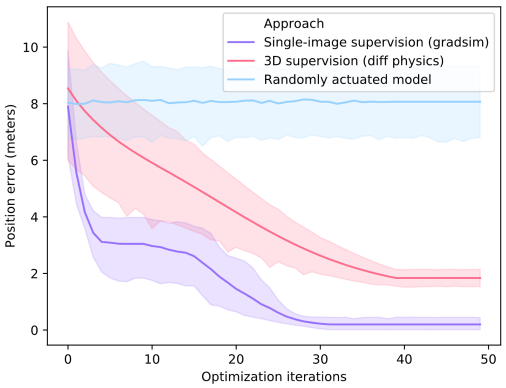
While strong supervision performs well on this task, it is important to note that it uses explicit 3D supervision at each timestep (i.e., 30 fps). In contrast, $\nabla$*Sim* uses a *single image* as an implicit target, and yet manages to achieve the goal state, albeit taking a longer number of iterations.

**Cloth Throw** In the case of cloth, we design an experiment to control a piece of cloth by optimizing the initial velocity such that it reaches a pre-specified target. In each episode, a randomly sized piece of cloth is spawned, comprising between 64 and 2048 triangles, and a new start/goal combination is chosen. Across all episodes, we fix elasticity parameters to $\mu = \lambda = 10^4$. For the strong supervision baseline we minimize the distance between the center-of-mass (COM) of the cloth and the target position. In the visual MPC case ($\nabla$*Sim*), the user provides an image specifying the desired cloth pose at $t = 5$s. The loss function is the pixelwise MSE between the last frame of the rendered video and the goal image.

In this challenging setup, we notice that state-based MPC is often unable to accurately reach the target. We believe this is due to the underdetermined nature of the problem, since, for objects such as cloth, the COM by itself does not uniquely determine the configuration of the object. Visual MPC on the otherhand, provides a more well-defined problem. An illustration of the task is presented in Fig. 62 (column 3), and the convergence of the methods shown in Fig. 64b.




(a) *3D Soft Walker*: Convergence over 6 random seeds varying the walker's goal configuration. While strong supervision performs well, it requires precise 3D observations. In contrast, image-based supervision is able to solve the task by using just a single image of the target configuration.




(b) *Cloth Control*: Convergence over 5 random seeds varying the dimensions and initial/target poses of the cloth. Strong supervision converges to a suboptimal solution due to ambiguity in specifying the pose of a cloth via its center-of-mass. $\nabla Sim$ solves the environment using a single target image.

Figure 64: **Convergence Analysis**: Convergence of $\nabla Sim$ visual MPC compared with strong 3D supervision, and random action policies.

### 7.3.7 Limitations

While our physics framework supports a rich set of material types there are still some missing features. For example, cloth self-collision is not currently handled, which greatly increases the solver complexity [124]. Additionally, the extension to articulated rigid bodies would open the door to many applications in robotics which we hope to explore. The differentiable renderer outputs images that have rather simplistic shading, lighting, and materials. However, we expect any video and images of the real world would be preprocessed using existing segmentation and classification techniques before performing optimization, this means our differentiable renderer does not necessarily need to provide photorealistic imagery.

### 7.3.8 Conclusion and Future Work

We have presented a framework for parameter estimation and model-based control that leverages differentiable simulation and differentiable rendering. We demonstrated that the combination of these two systems can achieve higher accuracy and efficiency than finite difference-based methods. Although we have trained on simulated imagery, our next steps are to connect this framework to imaging devices to enable low-cost identification and simulation of real-world objects.

# 8 Conclusion

This thesis presents new modeling tools, and numerical methods that aim to improve the capabilities of simulation for robotics. The presented methods are amenable to parallelization, and we have demonstrated them on real-time simulation of robots consisting of soft and rigid parts, that cross the reality gap. Already, the methods developed in this thesis have been adopted by industrial simulation frameworks, and used by researchers to perform large-scale robotics simulations [125, 55, 123, 145].

In Chapter 3 we presented a novel frictional contact model. However, it is still an analytic model, designed by hand. As data acquisition devices mature, we anticipate the development of data-driven models, that can adapt to real-world measurements. This is particularly exciting when combined with differentiable simulation, which can perform model-fitting using direct optimization.

Simulators can always be more efficient. While the methods presented in Chapters 4-5 achieve low-latency, we believe there is room to improve this further by adopting tools from the numerical optimization community. In particular, accelerated fixed-point methods such as Anderson acceleration [6], may provide high parallelism with even better convergence properties. In practice, there is often a tension between robustness and efficiency. We found that line-search strategies were unreliable when objectives are nonconvex. This suggests more sophisticated strategies are needed, for example watch-dog, or non-monotonic line-search [157]. However, these approaches typically limit parallelism. Adapting these safe guards to parallel architectures remains a challenge, but if solved could further improve solver robustness.

With regards to contact generation, a fundamental assumption that we have made in all methods is that contact geometry is linear over a time-step. Specifically, the normal direction and local contact point positions do not change during the implicit solve. This is a common simplification, however it can introduce locking artifacts [227], and we believe the solution to these is to consider how the contact geometry evolves over a time-step. This would, in essence, require interleaving of collision detection, and solver stages [159], where the contact constraint becomes a nonlinear function of system state. Our nonlinear solvers and efficient SDF contact generation provide a natural framework to include this effect.

In Chapter 7.3 we have shown how differentiable simulation can achieve excellent performance on parameter estimation and model predictive control tasks. However, gradient-based optimization alone suffers from becoming stuck in local-minima. For example, trajectory optimization through contact will generally not find solutions that include contacts that were not present in the starting guess. We believe the combination of model-based optimization via. auto-

matic differentiation, and stochastic exploration methods, e.g.: via reinforcement learning may provide the efficiency of gradient-based methods, while increasing global accuracy. In addition, the interplay between physical modeling and gradient-based optimization is subtle, and as yet mostly unexplored. For example, in Chapter 5 we have presented a relaxed contact model that has well-defined gradients. However, the sensitivity of different optimizers to these modifications has yet to be thoroughly analyzed.

To conclude, this thesis presents a set of simulation advances that hold the potential to enable new capabilities robotics. We believe simulation will play an increasingly important role in the future of robotics. As access to hardware becomes more challenging, having efficient, robust, and rich simulations is crucial to driving the development of technology that interacts with the real-world.

# References


[1] ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G. S., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., GOODFELLOW, I., HARP, A., IRVING, G., ISARD, M., JIA, Y., JOZEFOWICZ, R., KAISER, L., KUDLUR, M., LEVENBERG, J., MANÉ, D., MONGA, R., MOORE, S., MURRAY, D., OLAH, C., SCHUSTER, M., SHLENS, J., STEINER, B., SUTSKEVER, I., TALWAR, K., TUCKER, P., VANHOUCKE, V., VASUDEVAN, V., VIÉGAS, F., VINYALS, O., WARDEN, P., WATTENBERG, M., WICKE, M., YU, Y., AND ZHENG, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] ACARY, V., AND BROGLIATO, B. *Numerical methods for nonsmooth dynamical systems: applications in mechanics and electronics*. Springer Science & Business Media, 2008.

[3] ALART, P. Méthode de newton généralisée en mécanique du contact. *Journal Africain de Mathématiques pures et Appliquées 76* (1997), 83–108.

[4] ALART, P., AND CURNIER, A. A mixed formulation for frictional contact problems prone to newton like solution methods. *Computer methods in applied mechanics and engineering 92*, 3 (1991), 353–375.

[5] ALLARD, J., FAURE, F., COURTECUISSE, H., FALIPOU, F., DURIEZ, C., AND KRY, P. G. Volume contact constraints at arbitrary resolution. *ACM Trans. Graph. 29*, 4 (2010), 82.

[6] ANDERSON, D. G. Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM) 12*, 4 (1965), 547–560.

[7] ANDREWS, S., TEICHMANN, M., AND KRY, P. G. Geometric stiffness for real-time constrained multibody dynamics. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 235–246.

[8] ANDRYCHOWICZ, M., BAKER, B., CHOCIEJ, M., 'O ZEFOWICZ, R. J., MCGREW, B., PACHOCKI, J. W., PETRON, A., PLAPPERT, M., POWELL, G., RAY, A., SCHNEIDER, J., SIDOR, S., TOBIN, J., WELINDER, P., WENG, L., AND ZAREMBA, W. Learning dexterous in-hand manipulation. *IJ Robotics Res. 39*, 1 (2020).

[9] ANITESCU, M., AND HART, G. D. A constraint-stabilized time-stepping approach for rigid multibody dynamics with joints, contact and friction. *International Journal for Numerical Methods in Engineering 60*, 14 (2004), 2335–2371.

[10] ANTONOVA, R., CRUCIANI, S., SMITH, C., AND KRAGIC, D. Reinforcement learning for pivoting task. *CoRR abs/1703.00472* (2017).

[11] ASCHER, U. M., CHIN, H., PETZOLD, L. R., AND REICH, S. Stabilization of constrained mechanical systems with daes and invariant manifolds. *Journal of Structural Mechanics 23*, 2 (1995), 135–157.

[12] BARAFF, D., AND WITKIN, A. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM, pp. 43–54.

[13] BASU, K., AND OWEN, A. B. Low discrepancy constructions in the triangle. *SIAM Journal on Numerical Analysis 53*, 2 (2015), 743–761.

[14] BATTY, C., BERTAILS, F., AND BRIDSON, R. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. 26*, 3 (July 2007).

[15] BENDER, J., ERLEBEN, K., AND TRINKLE, J. Interactive simulation of rigid body dynamics in computer graphics. *Comput. Graph. Forum 33*, 1 (Feb. 2014), 246–270.

[16] BENDER, J., MÜLLER, M., OTADUY, M. A., TESCHNER, M., AND MACKLIN, M. A survey on position-based simulation methods in computer graphics. In *Computer graphics forum* (2014), vol. 33, Wiley Online Library, pp. 228–251.

[17] BENZI, M., GOLUB, G. H., AND LIESEN, J. Numerical solution of saddle point problems. *Acta numerica 14* (2005), 1–137.

[18] BERTAILS-DESCOUBES, F., CADOUX, F., DAVIET, G., AND ACARY, V. A nonsmooth newton solver for capturing exact coulomb friction in fiber assemblies. *ACM Transactions on Graphics (TOG) 30*, 1 (2011), 6.

[19] BHAT, K. S., TWIGG, C. D., HODGINS, J. K., KHOSLA, P. K., POPOVIC, Z., AND SEITZ, S. M. Estimating cloth simulation parameters from video. In *Symposium on Computer Animation* (2003), The Eurographics Association, pp. 37–51.

[20] BLINN, J. F. A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3 (1982), 235–256.

[21] BLOOMENTHAL, J., AND WYVILL, B. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[22] BOUAZIZ, S., MARTIN, S., LIU, T., KAVAN, L., AND PAULY, M. Projective dynamics: fusing constraint projections for fast simulation. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 154.

[23] BOYD, S. P., AND VANDENBERGHE, L. *Convex optimization*. Cambridge university press, 2004.

[24] BRADBURY, J., FROSTIG, R., HAWKINS, P., JOHNSON, M. J., LEARY, C., MACLAURIN, D., AND WANDERMAN-MILNE, S. JAX: composable transformations of Python+NumPy programs, 2018.

[25] BRIDSON, R., FEDKIW, R., AND ANDERSON, J. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21*, 3 (July 2002), 594–603.

[26] BRIDSON, R., MARINO, S., AND FEDKIW, R. Simulation of clothing with folds and wrinkles. In *Symposium on Computer Animation* (2003), The Eurographics Association, pp. 28–36.

[27] BROWN, G. E., OVERBY, M., FOROOTANINIA, Z., AND NARAIN, R. Accurate dissipative forces in optimization integrators. *ACM Transactions on Graphics (TOG) 37*, 6 (2018), 1–14.

[28] BROYDEN, C. G. A class of methods for solving nonlinear simultaneous equations. *Mathematics of computation 19*, 92 (1965), 577–593.

[29] CHANG, A. X., FUNKHOUSER, T., GUIBAS, L., HANRAHAN, P., HUANG, Q., LI, Z., SAVARESE, S., SAVVA, M., SONG, S., SU, H., ET AL. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* (2015).

[30] CHARBONNIER, P., BLANC-FÉRAUD, L., AUBERT, G., AND BARLAUD, M. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on image processing 6*, 2 (1997), 298–311.

[31] CHEN, T. Q., RUBANOVA, Y., BETTENCOURT, J., AND DUVENAUD, D. K. Neural ordinary differential equations. In *Neural Information Processing Systems* (2018), pp. 6571–6583.

[32] CHEN, W., GAO, J., LING, H., SMITH, E., LEHTINEN, J., JACOBSON, A., AND FIDLER, S. Learning to predict 3d objects with an interpolation-based differentiable renderer. *Neural Information Processing Systems* (2019).

[33] CHEN, Y. J., LEVIN, D. I., KAUFMANN, D., ASCHER, U., AND PAI, D. K. Eigenfit for consistent elastodynamic simulation across mesh resolution. In *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2019), pp. 1–13.

[34] CHEN, Z., FENG, R., AND WANG, H. Modeling friction and air effects between cloth and deformable bodies. *ACM Trans. Graph. 32*, 4 (July 2013), 88:1–88:8.

[35] CLARKE, F. H. *Optimization and nonsmooth analysis*, vol. 5. SIAM, 1990.

[36] CLINE, M. B., AND PAI, D. K. Post-stabilization for rigid body simulation with contact and constraints. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on* (2003), vol. 3, IEEE, pp. 3744–3751.

[37] COTTLE, R. W. Linear complementarity problem. In *Encyclopedia of Optimization*. Springer, 2008, pp. 1873–1878.

[38] COUMANS, E. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses* (New York, NY, USA, 2015), SIGGRAPH '15, ACM.

[39] COUMANS, E., AND BAI, Y. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

[40] CURNIER, A., AND ALART, P. A generalized newton method for contact problems with friction. *Journal de Mecanique Theorique et Appliquee 7*, suppl. 1 (1988), 67–82. Special issue entitled "Numerical Methods in Mechanics of Contact involving Friction".

[41] CURTIS, S., TAMSTORF, R., AND MANOCHA, D. Fast collision detection for deformable models using representative-triangles. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2008), I3D '08, Association for Computing Machinery, pp. 61–69.

[42] DAVIET, G. Simple and scalable frictional contacts for thin nodal objects. *ACM Transactions on Graphics (TOG) 39*, 4 (2020).

[43] DAVIET, G., BERTAILS-DESCOUBES, F., AND BOISSIEUX, L. A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 139.

[44] DE AVILA BELBUTE-PERES, F., SMITH, K., ALLEN, K., TENENBAUM, J., AND KOLTER, J. Z. End-to-end differentiable physics for learning and control. In *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 7178–7189.

[45] DEGRAVE, J., HERMANS, M., DAMBRE, J., AND WYFFELS, F. A differentiable physics engine for deep learning in robotics. *Frontiers Neurorobotics 13* (2019), 6.

[46] DEISENROTH, M. P., NEUMANN, G., AND PETERS, J. *A survey on policy search for robotics*. now publishers, 2013.

[47] DEISENROTH, M. P., AND RASMUSSEN, C. E. PILCO: A model-based and data-efficient approach to policy search. In *ICML* (2011), Omnipress, pp. 465–472.

[48] DEISENROTH, M. P., RASMUSSEN, C. E., AND FOX, D. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In *Robotics: Science and Systems* (2011).

[49] DI, J., YAO, S., YE, Y., CUI, Z., YU, J., GHOSH, T. K., ZHU, Y., AND GU, Z. Stretch-triggered drug delivery from wearable elastomer films containing therapeutic depots. *ACS nano 9*, 9 (2015), 9407–9415.

[50] DIRKSE, S. P., AND FERRIS, M. C. The path solver: a nommonotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software 5*, 2 (1995), 123–156.

[51] DRUMWRIGHT, E. A fast and stable penalty method for rigid body simulation. *IEEE transactions on visualization and computer graphics 14*, 1 (2007), 231–240.

[52] DURIEZ, C. Control of elastic soft robots based on real-time finite element method. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on* (2013), IEEE, pp. 3982–3987.

[53] DURIEZ, C. Control of elastic soft robots based on real-time finite element method. In *IEEE International Conference on Robotics and Automation* (May 2013), pp. 3982–3987.

[54] EBERT, D. S., MUSGRAVE, F. K., PEACHEY, D., PERLIN, K., AND WORLEY, S. *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003.

[55] EPPNER, C., MOUSAVIAN, A., AND FOX, D. A billion ways to grasps - an evaluation of grasp sampling schemes on a dense, physics-based grasp data set. In *Proceedings of the International Symposium on Robotics Research (ISRR)* (Hanoi, Vietnam, 2019).

[56] ERICSON, C. *Real-Time Collision Detection*. CRC Press, Inc., USA, 2004.

[57] ERLEBEN, K. Numerical methods for linear complementarity problems in physics-based animation. In *ACM SIGGRAPH 2013 Courses* (2013), ACM, p. 8.

[58] ERLEBEN, K. Rigid body contact problems using proximal operators. In *Proceedings of the ACM Symposium on Computer Animation* (2017), p. 13.

[59] ERLEBEN, K. Methodology for assessing mesh-based contact point methods. *ACM Transactions on Graphics (TOG) 37*, 3 (2018), 39.

[60] FARCHY, A., BARRETT, S., MACALPINE, P., AND STONE, P. Humanoid robots learning to walk faster: from the real world to simulation and back. In *AAMAS* (2013), IFAAMAS, pp. 39–46.

[61] FEATHERSTONE, R. *Rigid body dynamics algorithms*. Springer, 2014.

[62] FERRIS, M. C., AND MUNSON, T. S. Complementarity problems in gams and the path solver. *Journal of Economic Dynamics and Control 24*, 2 (2000), 165–188.

[63] FISCHER, A. A special newton-type optimization method. *Optimization 24*, 3-4 (1992), 269–284.

[64] FISHER, S., AND LIN, M. C. Deformed distance fields for simulation of non-penetrating flexible bodies. In *Computer Animation and Simulation 2001* (Vienna, 2001), N. Magnenat-Thalmann and D. Thalmann, Eds., Springer Vienna, pp. 99–111.

[65] FONG, D. C.-L., AND SAUNDERS, M. Cg versus minres: An empirical comparison. *Sultan Qaboos University Journal for Science [SQUJS] 17*, 1 (2012), 44–62.

[66] FRÂNCU, M., AND MOLDOVEANU, F. Virtual try on systems for clothes: Issues and solutions. *UPB Scientific Bulletin, Series C 77*, 4 (2015), 31–44.

[67] FRANK, M., AND WOLFE, P. An algorithm for quadratic programming. *Naval research logistics quarterly 3*, 1-2 (1956), 95–110.

[68] FRISKEN, S. F., PERRY, R. N., ROCKWOOD, A. P., AND JONES, T. R. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 249–254.

[69] FUHRMANN, A., SOBOTKA, G., AND GROSS, C. Distance fields for rapid collision detection in physically based modeling. In *International Conference on Computer Graphics and Vision '03. Proceedings* (Moscow, Russia, 01 2003), Eurographics.

[70] FUKUSHIMA, M., LUO, Z.-Q., AND TSENG, P. Smoothing functions for second-order-cone complementarity problems. *SIAM Journal on optimization 12*, 2 (2002), 436–460.

[71] GAST, T. F., SCHROEDER, C., STOMAKHIN, A., JIANG, C., AND TERAN, J. M. Optimization integrator for large time steps. *IEEE transactions on visualization and computer graphics 21*, 10 (2015), 1103–1115.

[72] GILBERT, E. G., JOHNSON, D. W., AND KEERTHI, S. S. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation 4*, 2 (1988), 193–203.

[73] GIRI, F., AND BAI, E.-W. *Block-oriented nonlinear system identification*, vol. 1. Springer, 2010.

[74] GOLDENTHAL, R., HARMON, D., FATTAL, R., BERCOVIER, M., AND GRINSPUN, E. Efficient simulation of inextensible cloth. In *ACM Transactions on Graphics (TOG)* (2007), vol. 26, ACM, p. 49.

[75] GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIR, S., COURVILLE, A., AND BENGIO, Y. Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680.

[76] GUEN, V. L., AND THOME, N. Disentangling physical dynamics from unknown factors for unsupervised video prediction, 2020.

[77] GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. Nonconvex rigid bodies with stacking. *ACM Trans. Graph. 22*, 3 (July 2003), 871–878.

[78] GUENNEBAUD, G., JACOB, B., ET AL. Eigen v3. http://eigen.tuxfamily.org, 2010.

[79] HAIRER, E., AND WANNER, G. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, vol. 14. Springer, 2010.

[80] HAN, L., GUAN, Y.-S., LI, Z., SHI, Q., AND TRINKLE, J. C. Dextrous manipulation with rolling contacts. In *Proceedings of International Conference on Robotics and Automation* (1997), vol. 2, IEEE, pp. 992–997.

[81] HANNA, J. P., AND STONE, P. Grounded action transformation for robot learning in simulation. In *AAAI* (2017), AAAI Press, pp. 4931–4932.

[82] HARMON, D., VOUGA, E., TAMSTORF, R., AND GRINSPUN, E. Robust treatment of simultaneous collisions. *ACM Trans. Graph. 27*, 3 (2008), 23.

[83] HART, J. C. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer 12*, 10 (1996), 527–545.

[84] HAUSMAN, K., CHEBOTAR, Y., SCHAAL, S., SUKHATME, G. S., AND LIM, J. J. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In *NIPS* (2017), pp. 1235–1245.

[85] HEIDEN, E. Tiny differentiable simulator, 2020.

[86] HEIDEN, E., MILLARD, D., COUMANS, E., AND SUKHATME, G. S. Augmenting differentiable simulators with neural networks to close the sim2real gap. *CoRR abs/2007.06045* (2020).

[87] HEIDEN, E., MILLARD, D., ZHANG, H., AND SUKHATME, G. S. Interactive differentiable simulation, 2019.

[88] HESTENES, M. R., AND STIEFEL, E. *Methods of conjugate gradients for solving linear systems*, vol. 49. NBS Washington, DC, 1952.

[89] HINTERMÜLLER, M. Semismooth newton methods and applications. *Department of Mathematics* (2010).

[90] HO, J., AND ERMON, S. Generative adversarial imitation learning. In *NIPS* (2016), pp. 4565–4573.

[91] HORST, R., AND TUY, H. *Global optimization: Deterministic approaches*. Springer Science & Business Media, Berlin Heidelberg, 2013.

[92] HU, Y., ANDERSON, L., LI, T.-M., SUN, Q., CARR, N., RAGAN-KELLEY, J., AND DURAND, F. Difftaichi: Differentiable programming for physical simulation. *International Conference on Learning Representations* (2020).

[93] HU, Y., FANG, Y., GE, Z., QU, Z., ZHU, Y., PRADHANA, A., AND JIANG, C. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans. Graph. 37*, 4 (July 2018).

[94] HU, Y., LIU, J., SPIELBERG, A., TENENBAUM, J. B., FREEMAN, W. T., WU, J., RUS, D., AND MATUSIK, W. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *ICRA* (2019), IEEE, pp. 6265–6271.

[95] IJSPEERT, A. J. Central pattern generators for locomotion control in animals and robots: a review. *Neural networks 21*, 4 (2008), 642–653.

[96] IJSPEERT, A. J., NAKANISHI, J., HOFFMANN, H., PASTOR, P., AND SCHAAL, S. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation 25*, 2 (2013), 328–373.

[97] ILIEVSKI, F., MAZZEO, A. D., SHEPHERD, R. F., CHEN, X., AND WHITESIDES, G. M. Soft robotics for chemists. *Angewandte Chemie 123*, 8 (2011), 1930–1935.

[98] JACOBSON, A., KAVAN, L., AND SORKINE-HORNUNG, O. Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–12.

[99] JAGGI, M. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning* (Atlanta, Georgia, USA, 17–19 Jun 2013), S. Dasgupta and D. McAllester, Eds., vol. 28 of *Proceedings of Machine Learning Research*, PMLR, pp. 427–435.

[100] JAKOBI, N., HUSBANDS, P., AND HARVEY, I. Noise and the reality gap: The use of simulation in evolutionary robotics. In *ECAL* (1995), vol. 929 of *Lecture Notes in Computer Science*, Springer, pp. 704–720.

[101] JAQUES, M., BURKE, M., AND HOSPEDALES, T. Physics-as-inverse-graphics: Unsupervised physical parameter estimation from video. In *International Conference on Learning Representations* (2019).

[102] JEAN, M. The non-smooth contact dynamics method. *Computer methods in applied mechanics and engineering 177*, 3-4 (1999), 235–257.

[103] JEAN, M., AND MOREAU, J. J. Unilaterality and dry friction in the dynamics of rigid body collections. In *1st Contact Mechanics International Symposium* (Lausanne, Switzerland, 1992), pp. 31–48.

[104] JOHNSON, K. L. *Contact Mechanics*. Cambridge University Press, 1985.

[105] JOURDAN, F., ALART, P., AND JEAN, M. A gauss-seidel like algorithm to solve frictional contact problems. *Computer methods in applied mechanics and engineering 155*, 1-2 (1998), 31–47.

[106] KALASHNIKOV, D., IRPAN, A., PASTOR, P., IBARZ, J., HERZOG, A., JANG, E., QUILLEN, D., HOLLY, E., KALAKRISHNAN, M., VANHOUCKE, V., AND LEVINE, S. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRR abs / 1806.10293* (2018).

[107] KARRAS, T. Maximizing parallelism in the construction of bvhs, octrees, and k-d trees. In *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics* (Goslar, DEU, 2012), EGGH-HPG'12, Eurographics Association, pp. 33–37.

[108] KAUFMAN, D. M., SUEDA, S., JAMES, D. L., AND PAI, D. K. Staggered projections for frictional contact in multibody systems. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 164.

[109] KAUFMAN, D. M., TAMSTORF, R., SMITH, B., AUBRY, J.-M., AND GRINSPUN, E. Adaptive nonlinearity for collisions in complex rod assemblies. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 123.

[110] KIEFER, J. Sequential minimax search for a maximum. *Proceedings of the American mathematical society 4*, 3 (1953), 502–506.

[111] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. In *ICLR (Poster)* (2015).

[112] KOSCHIER, D., DEUL, C., AND BENDER, J. Hierarchical hp-adaptive signed distance fields. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Goslar, DEU, 2016), SCA '16, Eurographics Association, pp. 189–198.

[113] LANCZOS, C. *The Variational Principles of Mechanics*, vol. 4. Courier Corporation, 1970.

[114] LAUSS, T., OBERPEILSTEINER, S., STEINER, W., AND NACHBAGAUER, K. The discrete adjoint method for parameter identification in multibody system dynamics. *Multibody system dynamics 42*, 4 (2018), 397–410.

[115] LENNART, L. System identification: theory for the user. *PTR Prentice Hall, Upper Saddle River, NJ* (1999), 1–14.

[116] LEVINE, S., PASTOR, P., KRIZHEVSKY, A., IBARZ, J., AND QUILLEN, D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research 37*, 4-5 (2018), 421–436.

[117] LI, J., DAVIET, G., NARAIN, R., BERTAILS-DESCOUBES, F., OVERBY, M., BROWN, G. E., AND BOISSIEUX, L. An implicit frictional contact solver for adaptive cloth simulation. *ACM Trans. Graph. 37*, 4 (July 2018), 52:1–52:15.

[118] LI, J., LIU, T., AND KAVAN, L. Fast simulation of deformable characters with articulated skeletons in projective dynamics. In *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2019), pp. 1–10.

[119] LI, M., FERGUSON, Z., SCHNEIDER, T., LANGLOIS, T., ZORIN, D., PANOZZO, D., JIANG, C., AND KAUFMAN, D. M. Incremental potential contact: Intersection- and inversion-free large deformation dynamics. *ACM Transactions on Graphics 39*, 4 (2020).

[120] LI, T.-M., AITTALA, M., DURAND, F., AND LEHTINEN, J. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 37*, 6 (2018), 222:1–222:11.

[121] LI, Y., LIN, T., YI, K., BEAR, D., YAMINS, D. L. K., WU, J., TENENBAUM, J. B., AND TORRALBA, A. Visual grounding of learned physical models, 2020.

[122] LI, Y., WU, J., TEDRAKE, R., TENENBAUM, J. B., AND TORRALBA, A. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *ICLR* (2019).

[123] LIANG, J., HANDA, A., VAN WYK, K., MAKOVIYCHUK, V., KROEMER, O., AND FOX, D. In-hand object pose tracking via contact feedback and gpu-accelerated robotic simulation. *arXiv preprint arXiv:2002.12160* (2020).

[124] LIANG, J., LIN, M., AND KOLTUN, V. Differentiable cloth simulation for inverse problems. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 772–781.

[125] LIANG, J., MAKOVIYCHUK, V., HANDA, A., CHENTANEZ, N., MACKLIN, M., AND FOX, D. Gpu-accelerated robotic simulation for distributed reinforcement learning. In *Conference on Robot Learning* (2018), pp. 270–282.

[126] LILEY, M., GOURDON, D., STAMOU, D., MESETH, U., FISCHER, T., LAUTZ, C., STAHLBERG, H., VOGEL, H., BURNHAM, N., AND DUSCHL, C. Friction anisotropy and asymmetry of a compliant monolayer induced by a small molecular tilt. *Science 280*, 5361 (1998), 273–275.

[127] LIN, M. C., AND OTADUY, M. *Haptic rendering: foundations, algorithms, and applications*. CRC Press, 2008.

[128] LIU, D. C., AND NOCEDAL, J. On the limited memory bfgs method for large scale optimization. *Mathematical programming 45*, 1-3 (1989), 503–528.

[129] LIU, F., AND KIM, Y. J. Exact and adaptive signed distance fields computation for rigid and deformable models on gpus. *IEEE transactions on visualization and computer graphics 20*, 5 (2013), 714–725.

[130] LIU, S., LI, T., CHEN, W., AND LI, H. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. *Proceedings of International Conference on Computer Vision* (2019).

[131] LIU, T., BARGTEIL, A. W., O'BRIEN, J. F., AND KAVAN, L. Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 214.

[132] LIU, T., BOUAZIZ, S., AND KAVAN, L. Quasi-newton methods for real-time simulation of hyperelastic materials. *ACM Transactions on Graphics (TOG) 36*, 3 (2017), 23.

[133] LOWREY, K., KOLEV, S., DAO, J., RAJESWARAN, A., AND TODOROV, E. Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system. In *SIMPAR* (2018), IEEE, pp. 35–42.

[134] LUO, M., PAN, Y., TAO, W., CHEN, F., SKORINA, E. H., AND ONAL, C. D. Refined theoretical modeling of a new-generation pressure-operated soft snake. In *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (2015), American Society of Mechanical Engineers, p. V05CT08A023.

[135] LUO, M., SKORINA, E. H., TAO, W., CHEN, F., OZEL, S., SUN, Y., AND ONAL, C. D. Toward modular soft robotics: Proprioceptive curvature sensing and sliding-mode control of soft bidirectional bending modules. *Soft robotics 4*, 2 (2017), 117–125.

[136] LY, M., JOUVE, J., BOISSIEUX, L., AND BERTAILS-DESCOUBES, F. Projective Dynamics with Dry Frictional Contact. *ACM Transactions on Graphics 39*, 4 (2020), 1–8.

[137] M. SÁNCHEZ-BANDERAS, R., AND A. OTADUY, M. Strain rate dissipation for elastic deformations. *Computer Graphics Forum 37*, 8 (2018), 161–170.

[138] MACKLIN, M., MÜLLER, M., AND CHENTANEZ, N. Xpbd: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (2016), ACM, pp. 49–54.

[139] MAHLER, J., LIANG, J., NIYAZ, S., LASKEY, M., DOAN, R., LIU, X., OJEA, J. A., AND GOLDBERG, K. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *CoRR abs/1703.09312* (2017).

[140] MANTI, M., HASSAN, T., PASSETTI, G., D'ELIA, N., LASCHI, C., AND CIANCHETTI, M. A bioinspired soft robotic gripper for adaptable and effective grasping. *Soft Robotics 2*, 3 (2015), 107–116.

[141] MARATOS, N. *Exact penalty function algorithms for finite dimensional and control optimization problems*. PhD thesis, Imperial College London (University of London), 1978.

[142] MARGOSSIAN, C. C. A review of automatic differentiation and its efficient implementation. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 9*, 4 (2019), e1305.

[143] MARHEFKA, D. W., AND ORIN, D. E. Simulation of contact using a nonlinear damping model. In *Proceedings of IEEE international conference on robotics and automation* (1996), vol. 2, IEEE, pp. 1662–1668.

[144] MARTIN, S., KAUFMANN, P., BOTSCH, M., GRINSPUN, E., AND GROSS, M. Unified simulation of elastic rods, shells, and solids. In *ACM SIGGRAPH 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 39:1–39:10.

[145] MATL, C., NARANG, Y., BAJCSY, R., RAMOS, F., AND FOX, D. Inferring the material properties of granular media for robotic tasks, 2020.

[146] MAZHAR, H., HEYN, T., NEGRUT, D., AND TASORA, A. Using nesterov's method to accelerate multibody dynamics with friction and contact. *ACM Transactions on Graphics (TOG) 34*, 3 (2015), 32.

[147] MAZHAR, H., MELANZ, D., FERRIS, M., AND NEGRUT, D. An analysis of several methods for handling hard-sphere frictional contact in rigid multibody dynamics. Tech. Rep. TR-2014-11, Simulation-Based Engineering Laboratory, University of Wisconsin-Madison., 09 2014.

[148] MCADAMS, A., ZHU, Y., SELLE, A., EMPEY, M., TAMSTORF, R., TERAN, J., AND SIFAKIS, E. Efficient elasticity for character skinning with contact and collisions. *ACM Trans. Graph. 30*, 4 (July 2011).

[149] MITCHELL, N., AANJANEYA, M., SETALURI, R., AND SIFAKIS, E. Non-manifold level sets: A multivalued implicit surface representation with applications to self-collision processing. *ACM Trans. Graph. 34*, 6 (2015), 247.

[150] MORAVANSZKY, A., TERDIMAN, P., AND KIRMSE, A. Fast contact reduction for dynamics simulation. *Game programming gems 4* (2004), 253–263.

[151] MÜLLER, M., HEIDELBERGER, B., HENNIX, M., AND RATCLIFF, J. Position based dynamics. *J. Vis. Comun. Image Represent. 18*, 2 (Apr. 2007), 109–118.

[152] MUNSON, T. S., FACCHINEI, F., FERRIS, M. C., FISCHER, A., AND KANZOW, C. The semismooth algorithm for large scale complementarity problems. *INFORMS Journal on Computing 13* (2001), 294–311.

[153] NARAIN, R., SAMII, A., AND O'BRIEN, J. F. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph. 31*, 6 (2012), 152.

[154] NIEBE, S., AND ERLEBEN, K. Numerical methods for linear complementarity problems in physics-based animation. *Synthesis Lectures on Computer Graphics and Animation 7*, 1 (2015), 1–159.

[155] NIMIER-DAVID, M., VICINI, D., ZELTNER, T., AND JAKOB, W. Mitsuba 2: A retargetable forward and inverse renderer. *Transactions on Graphics (Proceedings of SIGGRAPH Asia) 38*, 6 (2019).

[156] NOCEDAL, J. Updating quasi-newton matrices with limited storage. *Mathematics of computation 35*, 151 (1980), 773–782.

[157] NOCEDAL, J., AND WRIGHT, S. *Numerical optimization*. Springer Science & Business Media, 2006.

[158] OPENAI. Roboschool. https://github.com/openai/roboschool, 2017.

[159] OTADUY, M. A., TAMSTORF, R., STEINEMANN, D., AND GROSS, M. Implicit contact handling for deformable objects. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 559–568.

[160] OZEL, S., SKORINA, E. H., LUO, M., TAO, W., CHEN, F., PAN, Y., AND ONAL, C. D. A composite soft bending actuation module with integrated curvature sensing. In *Robotics and Automation (ICRA), IEEE International Conference on* (2016), IEEE, pp. 4963–4968.

[161] PABST, S., THOMASZEWSKI, B., AND STRASSER, W. Anisotropic friction for deformable surfaces and solids. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2009), SCA '09, ACM, pp. 149–154.

[162] PAN, Z., REN, B., AND MANOCHA, D. Gpu-based contact-aware trajectory optimization using a smooth force model. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2019), SCA '19, Association for Computing Machinery.

[163] PANDOLFI, A., AND ORTIZ, M. Finite element analysis of nonsmooth frictional contact. *Solid Mechanics and its Applications 3* (01 2007).

[164] PANG, J.-S. Newton's method for b-differentiable equations. *Mathematics of Operations Research 15*, 2 (1990), 311–341.

[165] PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N., ANTIGA, L., DESMAISON, A., KOPF, A., YANG, E., DEVITO, Z., RAISON, M., TEJANI, A., CHILAMKURTHY, S., STEINER, B., FANG, L., BAI, J., AND CHINTALA, S. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[166] PEREZ, A. G., CIRIO, G., HERNANDEZ, F., GARRE, C., AND OTADUY, M. A. Strain limiting for soft finger contact simulation. In *World Haptics Conference (WHC), 2013* (2013), IEEE, pp. 79–84.

[167] PETERS, J., "U LLING, K. M., AND ALTUN, Y. Relative entropy policy search. In *AAAI* (2010), AAAI Press.

[168] PINTO, L., AND GUPTA, A. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *ICRA* (2016), IEEE, pp. 3406–3413.

[169] POLYGERINOS, P., WANG, Z., OVERVELDE, J. T. B., GALLOWAY, K. C., WOOD, R. J., BERTOLDI, K., AND WALSH, C. J. Modeling of soft fiber-reinforced bending actuators. *IEEE Transactions on Robotics 31*, 3 (June 2015), 778–789.

[170] POZZI, M., MIGUEL, E., DEIMEL, R., MALVEZZI, M., BICKEL, B., BROCK, O., AND PRATTICHIZZO, D. Efficient fem-based simulation of soft robots modeled as kinematic chains. In *IEEE International Conference on Robotics and Automation (ICRA)* (May 2018), pp. 1–8.

[171] PROVOT, X. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation '97* (Vienna, 1997), D. Thalmann and M. van de Panne, Eds., Springer Vienna, pp. 177–189.

[172] QI, L., AND SUN, J. A nonsmooth version of newton's method. *Mathematical programming 58*, 1-3 (1993), 353–367.

[173] QIAO, Y.-L., LIANG, J., KOLTUN, V., AND LIN, M. C. Scalable differentiable physics for learning and control. In *ICML* (2020).

[174] RAJESWARAN, A., GHOTRA, S., RAVINDRAN, B., AND LEVINE, S. Epopt: Learning robust neural network policies using model ensembles. In *ICLR (Poster)* (2017), OpenReview.net.

[175] ROSEN, J. B. The gradient projection method for nonlinear programming. part i. linear constraints. *Journal of the Society for Industrial and Applied Mathematics 8*, 1 (1960), 181–217.

[176] ROSEN, J. B. The gradient projection method for nonlinear programming. part ii. nonlinear constraints. *Journal of the Society for Industrial and Applied Mathematics 9*, 4 (1961), 514–532.

[177] SAAD, Y. *Iterative Methods for Sparse Linear Systems*, 2nd ed. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[178] SADEGHI, F., AND LEVINE, S. CAD2RL: real single-image flight without a single real image. In *Robotics: Science and Systems* (2017).

[179] SADEGHI, F., TOSHEV, A., JANG, E., AND LEVINE, S. Sim2real view invariant visual servoing by recurrent control. *arXiv:1712.07642* (2017).

[180] SCHENCK, C., AND FOX, D. Spnets: Differentiable fluid dynamics for deep neural networks, 2018.

[181] SCHMIDT, T., NEWCOMBE, R. A., AND FOX, D. DART: dense articulated real-time tracking. In *Robotics: Science and Systems* (2014).

[182] SCHULMAN, J., WOLSKI, F., DHARIWAL, P., RADFORD, A., AND KLIMOV, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[183] SELLE, A., SU, J., IRVING, G., AND FEDKIW, R. Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. *IEEE transactions on visualization and computer graphics 15*, 2 (2008), 339–350.

[184] SERMANET, P., LYNCH, C., CHEBOTAR, Y., HSU, J., JANG, E., SCHAAL, S., AND LEVINE, S. Time-contrastive networks: Self-supervised learning from video. In *ICRA* (2018), IEEE, pp. 1134–1141.

[185] SERVIN, M., LACOURSIERE, C., AND MELIN, N. Interactive simulation of elastic deformable materials. In *Proceedings of SIGRAD Conference* (2006), pp. 22–32.

[186] SEYB, D., JACOBSON, A., NOWROUZEZAHRAI, D., AND JAROSZ, W. Non-linear sphere tracing for rendering deformed signed distance fields. *ACM Trans. Graph. 38*, 6 (Nov. 2019), 229:1–229:12.

[187] SHINAR, T., SCHROEDER, C., AND FEDKIW, R. Two-way coupling of rigid and deformable bodies. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008), Eurographics Association, pp. 95–103.

[188] SI, H. Tetgen: A quality tetrahedral mesh generator and three-dimensional delaunay triangulator. *Weierstrass Institute for Applied Analysis and Stochastic, Berlin, Germany* (2006).

[189] SICILIANO, B., AND KHATIB, O. *Springer handbook of robotics*. Springer, 2016.

[190] SIFAKIS, E., AND BARBIC, J. Fem simulation of 3d deformable solids: A practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses* (New York, NY, USA, 2012), SIGGRAPH '12, Association for Computing Machinery.

[191] SILCOWITZ, M., NIEBE, S. M., AND ERLEBEN, K. Nonsmooth newton method for fischer function reformulation of contact force problems for interactive rigid body simulation. In *6th Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS 09* (2009), pp. 105–114.

[192] SILCOWITZ-HANSEN, M., NIEBE, S., AND ERLEBEN, K. A nonsmooth nonlinear conjugate gradient method for interactive contact force problems. *The Visual Computer 26*, 6-8 (2010), 893–901.

[193] SMITH, B., GOES, F. D., AND KIM, T. Stable neo-hookean flesh simulation. *ACM Transactions on Graphics (TOG) 37*, 2 (2018), 12.

[194] SOLER, C., MARTIN, T., AND SORKINE-HORNUNG, O. Cosserat rods with projective dynamics. In *Computer Graphics Forum* (2018), vol. 37, Wiley Online Library, pp. 137–147.

[195] SONG, C., AND BOULARIAS, A. Identifying mechanical models through differentiable simulations. *CoRR abs/2005.05410* (2020).

[196] SONG, C., AND BOULARIAS, A. Learning to slide unknown objects with differentiable physics simulations. *CoRR abs/2005.05456* (2020).

[197] STAM, J. Nucleus: Towards a unified dynamics solver for computer graphics. In *Computer-Aided Design and Computer Graphics, 2009. CAD/Graphics' 09. 11th IEEE International Conference on* (2009), IEEE, pp. 1–11.

[198] STEWART, D. E. Rigid-body dynamics with friction and impact. *SIAM review 42*, 1 (2000), 3–39.

[199] STEWART, D. E., AND TRINKLE, J. C. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering 39*, 15 (1996), 2673–2691.

[200] SUN, S., CAO, Z., ZHU, H., AND ZHAO, J. A survey of optimization methods from a machine learning perspective. *IEEE transactions on cybernetics* (2019).

[201] SUTTON, R. S., AND BARTO, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

[202] TAKAHASHI, T., AND LIN, M. C. Video-guided real-to-virtual parameter transfer for viscous fluids. *ACM Transactions on Graphics (TOG) 38*, 6 (2019), 1–12.

[203] TAN, J., XIE, Z., BOOTS, B., AND LIU, C. K. Simulation-based design of dynamic controllers for humanoid balancing. In *IROS* (2016), IEEE, pp. 2729–2736.

[204] TAN, J., ZHANG, T., COUMANS, E., ISCEN, A., BAI, Y., HAFNER, D., BOHEZ, S., AND VANHOUCKE, V. Sim-to-real: Learning agile locomotion for quadruped robots. In *Robotics: Science and Systems* (2018).

[205] TANG, M., MANOCHA, D., OTADUY, M. A., AND TONG, R. Continuous penalty forces. *ACM Trans. Graph. 31*, 4 (2012), 107:1–107:9.

[206] TANG, M., WANG, T., LIU, Z., TONG, R., AND MANOCHA, D. I-cloth: Incremental collision handling for gpu-based interactive cloth simulation. *ACM Transactions on Graphics (TOG) 37*, 6 (2018), 1–10.

[207] TERAN, J., SIFAKIS, E., IRVING, G., AND FEDKIW, R. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), ACM, pp. 181–190.

[208] TESCHNER, M., KIMMERLE, S., HEIDELBERGER, B., ZACHMANN, G., RAGHUPATHI, L., FUHRMANN, A., CANI, M.-P., FAURE, F., MAGNENAT-THALMANN, N., STRASSER, W., ET AL. Collision detection for deformable objects. In *Computer graphics forum* (USA, 2005), vol. 24, Wiley Online Library, pp. 61–81.

[209] TISSEUR, F. Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications 22*, 4 (2001), 1038–1057.

[210] TOBIN, J., FONG, R., RAY, A., SCHNEIDER, J., ZAREMBA, W., AND ABBEEL, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017), IEEE, pp. 23–30.

[211] TODOROV, E. Implicit nonlinear complementarity: A new approach to contact dynamics. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on* (2010), IEEE, pp. 2322–2329.

[212] TODOROV, E. Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), IEEE, pp. 6054–6061.

[213] TODOROV, E., EREZ, T., AND TASSA, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012), IEEE, pp. 5026–5033.

[214] TONGE, R., BENEVOLENSKI, F., AND VOROSHILOV, A. Mass splitting for jitter-free parallel rigid body simulation. *ACM Trans. Graph. 31*, 4 (July 2012), 105:1–105:8.

[215] TOURNIER, M., NESME, M., GILLES, B., AND FAURE, F. Stable constrained dynamics. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 132.

[216] TOUSSAINT, M., ALLEN, K., SMITH, K., AND TENENBAUM, J. Differentiable physics and stable modes for tool-use and manipulation planning. In *Proceedings of Robotics: Science and Systems* (Pittsburgh, Pennsylvania, June 2018).

[217] TZENG, E., DEVIN, C., HOFFMAN, J., FINN, C., PENG, X., LEVINE, S., SAENKO, K., AND DARRELL, T. Towards adapting deep visuomotor representations from simulated to real environments. *CoRR abs / 1511.07111* (2015).

[218] TZENG, E., HOFFMAN, J., DARRELL, T., AND SAENKO, K. Simultaneous deep transfer across domains and tasks. In *ICCV* (2015), IEEE Computer Society, pp. 4068–4076.

[219] UMBANHOWAR, P., VOSE, T. H., MITANI, A., HIRAI, S., AND LYNCH, K. M. The effect of anisotropic friction on vibratory velocity fields. In *2012 IEEE International Conference on Robotics and Automation* (May 2012), pp. 2584–2591.

[220] VAN MERRIËNBOER, B., WILTSCHKO, A. B., AND MOLDOVAN, D. Tangent: automatic differentiation using source code transformation in python. In *arXiv* (2017).

[221] VERSCHOOR, M., AND JALBA, A. C. Efficient and accurate collision response for elastically deformable models. *ACM Trans. Graph. 38*, 2 (Mar. 2019).

[222] WALKER, I. D. Continuous backbone "continuum" robot manipulators. *ISRN Robotics* (2013).

[223] WANG, B., FAURE, F., AND PAI, D. K. Adaptive image-based intersection volume. *ACM Trans. Graph. 31*, 4 (July 2012), 97:1–97:9.

[224] WANG, H. A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 246.

[225] WANG, H., AND YANG, Y. Descent methods for elastic body simulation on the gpu. *ACM Transactions on Graphics (TOG) 35*, 6 (2016), 212.

[226] WEIDNER, N. J., PIDDINGTON, K., LEVIN, D. I., AND SUEDA, S. Eulerian-on-lagrangian cloth simulation. *ACM Trans. Graph. 37*, 4 (2018), 50.

[227] WILLIAMS, J., LU, Y., AND TRINKLE, J. A geometrically exact contact model for polytopes in multirigid-body simulation. *Journal of Computational and Nonlinear Dynamics 12*, 2 (2017), 021001.

[228] WILLIAMS, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning 8*, 3–4 (May 1992), 229–256.

[229] WU, J., LU, E., KOHLI, P., FREEMAN, W. T., AND TENENBAUM, J. B. Learning to see physics via visual de-animation. In *Advances in Neural Information Processing Systems* (2017).

[230] WYVILL, B., GUY, A., AND GALIN, E. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum 18*, 2 (1999), 149–158.

[231] XU, H., AND BARBIC, J. Continuous Collision Detection Between Points and Signed Distance Fields. In *Workshop on Virtual Reality Interaction and Physical Simulation* (Bremen, Germany, 2014), J. Bender, C. Duriez, F. Jaillet, and G. Zachmann, Eds., The Eurographics Association.

[232] XU, H., ZHAO, Y., AND BARBIC, J. Implicit multibody penalty-based distributed contact. *IEEE transactions on visualization and computer graphics 20*, 9 (2014), 1266–1279.

[233] YAHYA, A., LI, A., KALAKRISHNAN, M., CHEBOTAR, Y., AND LEVINE, S. Collective robot reinforcement learning with distributed asynchronous guided policy search. In *IROS* (2017), IEEE, pp. 79–86.

[234] YAMANE, K., AND NAKAMURA, Y. Stable penalty-based model of frictional contacts. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* (2006), IEEE, pp. 1904–1909.

[235] YANG, S., LIANG, J., AND LIN, M. C. Learning-based cloth material recovery from video. In *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 4393–4403.

[236] YU, C., AND WANG, Q. J. Friction anisotropy with respect to topographic orientation. *Scientific reports 2* (2012), 988.

[237] YU, W., KUMAR, V. C., TURK, G., AND LIU, C. K. Sim-to-real transfer for biped locomotion. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019), IEEE, pp. 3503–3510.

[238] ZHENG, C., AND JAMES, D. L. Toward high-quality modal contact sound. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011) 30*, 4 (Aug. 2011).

[239] ZHU, S., KIMMEL, A., BEKRIS, K. E., AND BOULARIAS, A. Fast model identification via physics engines for data-efficient policy search. In *IJCAI* (2018), ijcai.org, pp. 3249–3256.

# A Contact Preconditioners

In this appendix we give the full form of the contact constraints and derivatives, including the $r$-factor scaling parameter, for the minimum-map and Fischer-Burmeister NCP-functions. First, the contact constraint given in terms of the Fischer-Burmeister function can be written as,

$$\phi_{n\text{FB}} = C_n(\mathbf{q}) + r\lambda_n - \sqrt{C_n(\mathbf{q})^2 + r^2\lambda_n^2}, \tag{159}$$

which has the following derivatives,

$$\frac{\partial \phi_{n\text{FB}}}{\partial \mathbf{q}} = \left(1 - \frac{C_n(\mathbf{q})}{\sqrt{C_n(\mathbf{q})^2 + r^2\lambda_n^2}}\right) \nabla C_n \tag{160}$$

$$\frac{\partial \phi_{n\text{FB}}}{\partial \lambda_n} = \left(1 - \frac{r\lambda_n}{\sqrt{C_n(\mathbf{q})^2 + r^2\lambda_n^2}}\right) r. \tag{161}$$

Likewise, for the minimum-map,

$$\phi_{n\text{min}} = \begin{cases} C_n & C_n(\mathbf{q}) \leq r\lambda_n \\ r\lambda_n & \text{otherwise} \end{cases} \tag{162}$$

with derivatives given by,

$$\frac{\partial \phi_{n\text{min}}}{\partial \mathbf{q}} = \begin{cases} \nabla C_n & C_n(\mathbf{q}) \leq r\lambda_n \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{163}$$

$$\frac{\partial \phi_{n\text{min}}}{\partial \lambda_n} = \begin{cases} 0 & C_n(\mathbf{q}) \leq r\lambda_n \\ r & \text{otherwise} \end{cases}. \tag{164}$$

# B Compliance Form of Stable Neo-Hookean Materials

Here we give the derivatives required for the compliance form of the stable Neo-Hookean material model introduced by Smith et al [193]. The elastic strain-energy density is given by

$$\Psi_E = C_1(I_C - 3) + D_1(J - \alpha)^2. \tag{165}$$

The material constants $C_1$, $D_1$, and $\alpha$ can be related to the Lamé parameters according to the original paper. Defining $\mathbf{s} = [s_1, s_2, s_3]^T$ as the vector of principal stretches of the deformation gradient $\mathbf{F}$ we have $I_C = s_1^2 + s_2^2 + s_3^2$ the first invariant of stretch, and $J = s_1 s_2 s_3$, the relative volume change induced by $\mathbf{F}$. To perform the compliance transformation we require the Jacobian and Hessian of $\Psi$ with respect to $\mathbf{s}$, which we provide below:

$$\frac{\partial \Psi_E}{\partial \mathbf{s}} = 2C_1 \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}^T + 2D_1(J-1) \begin{bmatrix} s_2 s_3 \\ s_1 s_3 \\ s_1 s_2 \end{bmatrix}^T$$

$$\frac{\partial^2 \Psi_E}{\partial \mathbf{s}^2} = 2 \begin{bmatrix} D_1 s_2^2 s_3^2 + C_1 & k_1 & k_2 \\ k_1 & D_1 s_1^2 s_3^2 + C_1 & k_3 \\ k_2 & k_3 & D_1 s_1^2 s_2^2 + C_1 \end{bmatrix}$$

$$\begin{aligned} k_0 &= 2J - \alpha, \\ k_1 &= D_1 s_3 k_0, \\ k_2 &= D_1 s_2 k_0, \\ k_3 &= D_1 s_1 k_0. \end{aligned}$$

For a constant strain tetrahedron with rest volume $V_e$ the elastic potential energy is $U(\mathbf{q}) = V_e \Psi_E(\mathbf{s})$, and the compliance matrix is $\mathbf{E} = \left( V_e \frac{\partial^2 \Psi_E}{\partial \mathbf{s}^2} \right)^{-1}$ which can be obtained through a $3 \times 3$ matrix inverse. The derivatives of the singular values with respect to the vertices, $\mathbf{J} = \frac{\partial \mathbf{s}}{\partial \mathbf{q}}$, are given in [166].