ROBUST AND ACCURATE SIMULATION OF ELASTODYNAMICS AND CONTACT

Minchen Li

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2020

Chenfanfu Jiang, Assistant Professor
Computer and Information Science
Supervisor of Dissertation

Mayur Naik, Professor
Computer and Information Science
Graduate Group Chairperson

Dissertation Committee

Norman I. Badler, Professor
Computer and Information Science
University of Pennsylvania

Stephen H. Lane, Professor of Practice
Computer and Information Science
University of Pennsylvania

Cynthia Sung, Assistant Professor
Mechanical Engineering
University of Pennsylvania

Danny M. Kaufman, Senior Research Scientist
Creative Intelligence Lab
Adobe Research

ROBUST AND ACCURATE SIMULATION OF ELASTO-DYNAMICS AND CONTACT

2020

Minchen Li

To my family and friends.

Acknowledgements

First, I would like to deeply thank my parents who always understand and support me.

Thank Prof. Kun Zhou, Dr. Menglei Chai, and Prof. Jijun Li for introducing computer graphics to me when I was an undergraduate student at Zhejiang University. Thank Prof. Victor C.M. Leung and Prof. Wei Cai for providing me the opportunity to intern in a foreign country at the first time of my life which shaped my decision on pursuing a PhD.

Thank Prof. Alla Sheffer, Prof. Eitan Grinspun, Dr. Danny M. Kaufman, Dr. Vladimir G. Kim, Prof. Justin Solomon, Dr. Xinxin Zhang, and Prof. Robert Bridson. Having the opportunity to work with you when I was a master student at University of British Columbia provide me a solid research background, which contributes a lot to my PhD.

Thank Dr. Danny M. Kaufman for guiding me through my four internships at Adobe Research and introducing elastodynamics and contact problems to me. Without you we would never come up with the brilliant ideas.

Thank Dr. Ming Gao, Dr. Timothy Langlois, Yu Fang, Joshuah Wolper, Jiecong Lu, Yupeng Jiang, Xuan Li, Yue Li, Dr. Yixin Zhu, Prof. Bo Zhu, Dr. Xinlei Wang, Yunuo Chen, Ziyin Qu, Yuxing Qiu, Zachary Ferguson, Dr. Teseo Schneider, Prof. Denis Zorin, Prof. Daniele Panozzo, Jessica McWilliams, Dr. Yufeng Zhu, Dr. Evan Peng for being wonderful collaborators and consultants. Thank Prof. Norman I. Badler, Prof. Stephen H. Lane, Prof. Cynthia Sung, and Dr Danny M. Kaufman for being on my committee.

Last but not least, I would like to express my greatest gratitude to my advisor Prof. Chenfanfu Jiang who is both a best friend and a role model. It would be impossible for me to complete all these great works without your support and guidance.

ABSTRACT

ROBUST AND ACCURATE SIMULATION OF ELASTODYNAMICS AND CONTACT

Minchen Li

Chenfanfu Jiang

Simulating elastodynamics and contact in a robust and accurate way not only benefits designing realistic and intriguing animations and visual effects in computer graphics, but is also essential for industrial design, robotics, mechanical engineering analysis, etc. However, existing methods are constructed under strong assumptions that limit their application scenarios to a relatively narrow range, which also make the methods sensitive to algorithmic parameters such that extensive parameter tuning often needs to be performed for nearly each different example to obtain consistent quality results. To tackle these challenges, we propose a robust, accurate, and differentiable elastodynamics and contact simulation framework that can always reliably produce consistent quality results for any codimensional solids (volumes, shells, rods, and particles) in a wide range of material, time step size, boundary condition, and resolution settings with interpenetration-free guarantees but do not require algorithmic parameter tuning. Based on solid theoretical foundations, our methods provide controllable trade-off between efficiency and accuracy for different application scenarios. All the proposed features of our methods are thoroughly verified by performing extensive experiments and analyses including comparisons to state-of-the-art methods and ablation study on multiple design choices. Our framework frees designers from extensive parameter tuning as when traditional methods are used, enables simulating brand new phenomena that are never achieved before, and already demonstrate effectiveness in a broader range of application scenarios like robotics design and engineering analysis.

Contents

List of Figures

Chapter 1

Introduction

Solid is one of the four fundamental states of matter, which can be seen everywhere in the world, from food, furniture, and garments, to cars, plants, and animals. Being able to simulate the dynamic behavior of solids with complex self-contact configurations under various external conditions not only allow us to design realistic and intriguing animations and visual effects in computer graphics, but also can benefit industrial design, robotics, civil and mechanical engineering, material science, and bio-mechanics, etc.

Extensive researches Bergou et al. [2008], Gast et al. [2015], Grinspun et al. [2003], Kane et al. [1999], Kaufman et al. [2008], Liu et al. [2017], Teran et al. [2005], Verschoor and Jalba [2019] have been established for modeling, analyzing, and numerically simulating elastodynamics and/or contact. However, robust and accurate simulation of these effects is still challenging because mathematically, elasticity is highly nonlinear and non-convex, and contact is non-smooth. Simulating elastodynamics with contact in a coupled system then brings in more challenges. These pose significant difficulties to standard numerical methods, making them inefficient and unpredictable under potentially extreme conditions and sometimes even results in numerical explosions. In addition, many existing methods are constructed under strong assumptions that limit their application scenarios to a relatively narrow range and also make the methods sensitive to algorithmic parameters such that extensive parameter tuning often needs to be performed for nearly each different example

to obtain consistent quality results. These make the methods inconvenient to use, and also insufficient for a broader range of applications like robotics design, shape optimization, etc, where the reliability of methods is essential.

In this dissertation, we aim at innovating robust, accurate, and differentiable elastodynamics and contact simulation methods that can always reliably produce consistent quality results for any codimensional solids (volumes, shells, rods, and particles) in a wide range of material, time step size, boundary condition, and resolution settings with interpenetration-free guarantees but do not require algorithmic parameter tuning. Based on solid theoretical foundations, we design our methods to provide controllable trade-off between efficiency and accuracy for different application scenarios. We also perform extensive experiments and analyses to thoroughly verify all the above features. This work includes

- 7 ACM SIGGRAPH papers Fang et al. [2019, 2020], Li et al. [2019, 2020b], Wang et al. [2020b], Wolper et al. [2019, 2020],
- 1 ACM Transactions on Graphics (ToG) article Wang et al. [2020a] presented at SIGGRAPH 2020,
- 1 International Journal for Numerical Methods in Engineering (IJNME) article Jiang et al. [2020], and
- 3 submissions Li et al. [2020c,d,e] to top tier conferences or journals under review.

Here we focus on Decomposed Optimization Time Integrator (DOT) Li et al. [2019], Incremental Potential Contact (IPC) Li et al. [2020b], and Constitutive Strain Limiting and Thickness Modeling Li et al. [2020c].

Decomposed Optimization Time Integrator Simulation methods are rapidly advancing the accuracy, consistency and controllability of elastodynamic modeling and animation. Critical to these advances, we require efficient time step solvers that reliably solve all implicit time integration problems for elastica. While available time step solvers succeed

admirably in some regimes, they become impractically slow, inaccurate, unstable, or even divergent in others. Towards addressing these needs, in Chapter 3 we present the Decomposed Optimization Time Integrator (DOT), a new domain-decomposed optimization method for solving the per time step, nonlinear problems of implicit numerical time integration. DOT is especially suitable for large time step simulations of deformable bodies with nonlinear materials and high-speed dynamics (Figure 1.1). It is efficient, automated, and robust at large, fixed-size time steps, thus ensuring stable, continued progress of high-quality simulation output. Across a broad range of extreme and mild deformation dynamics, using frame-rate size time steps with widely varying object shapes and mesh resolutions, we show that DOT always converges to user-set tolerances, generally well-exceeding and always close to the best wall-clock times across all previous nonlinear time step solvers, irrespective of the deformation applied.

Incremental Potential Contact Contacts weave through every aspect of our physical world, from daily household chores to acts of nature. Modeling and predictive computation of these phenomena for solid mechanics is important to every discipline concerned with the motion of mechanical systems, including engineering and animation. Nevertheless, efficiently time-stepping accurate and consistent simulations of real-world contacting elastica remains an outstanding computational challenge. To model the complex interaction of deforming solids in contact, in Chapter 4 we propose Incremental Potential Contact (IPC) – a new model and algorithm for variationally solving implicitly time-stepped nonlinear elastodynamics. IPC maintains an intersection- and inversion-free trajectory regardless of material parameters, time step sizes, impact velocities, severity of deformation, or boundary conditions enforced (Figure 1.2).

Constructed with a custom nonlinear solver, IPC enables efficient resolution of time-stepping problems with separate, user-exposed accuracy tolerances that allow independent specification of the physical accuracy of the dynamics and the geometric accuracy of surface-to-surface conformation. This enables users to decouple, as needed per application,

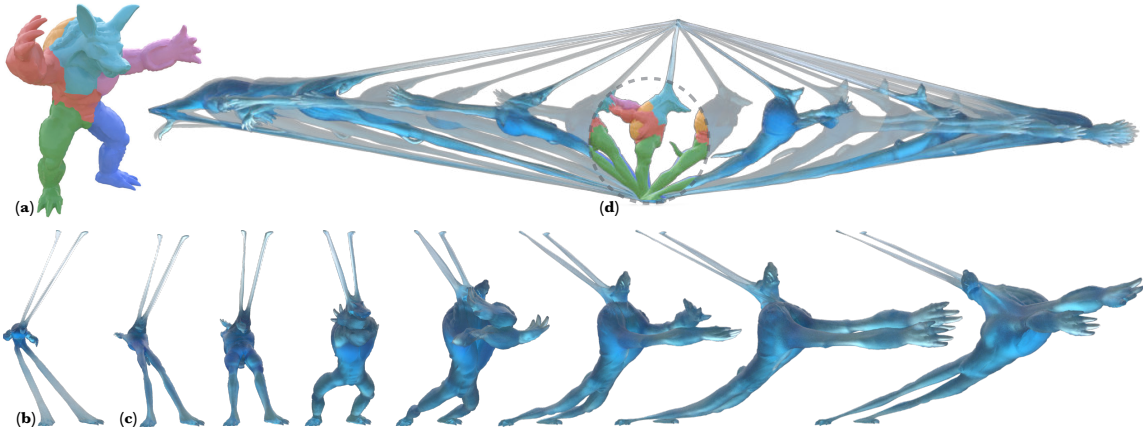


Figure 1.1: ***Severe deformation dynamics simulated with large steps.*** **(a)** The Decomposed Optimization Time Integrator (DOT) decomposes spatial domains to generate high-quality simulations of nonlinear materials undergoing large-deformation dynamics. In **(b)** we apply DOT to rapidly stretch and pull an Armadillo backwards. We render in **(c)** a few frames of the resulting slingshot motion right after release. DOT efficiently solves time steps while achieving user-specified accuracies — even when stepping at frame-rate size steps; here at 25 ms. In **(d)** we emphasize the large steps taken by rendering all DOT-simulated time steps from the Armadillo's high-speed trajectory for the first few moments after release.

desired accuracies for a simulation's dynamics and geometry.

The resulting time stepper solves contact problems that are intersection-free (and thus robust), inversion-free, efficient (at speeds comparable to or faster than available methods that lack both convergence and feasibility), and accurate (solved to user-specified accuracies). To our knowledge this is the first implicit time-stepping method, across both the engineering and graphics literature that can consistently enforce these guarantees as we vary simulation parameters.

In an extensive comparison of available simulation methods, research libraries and commercial codes we confirm that available engineering and computer graphics methods,

Figure 1.2: **Squeeze out:** Incremental Potential Contact (IPC) enables high-rate time stepping, here with $h = 0.01$s, of extreme nonlinear elastodynamics with contact that is intersection- and inversion-free at all time steps, irrespective of the degree of compression and contact. Here a plate compresses and then forces a collection of complex soft elastic FE models (181K tetrahedra in total, with a neo-Hookean material) through a thin, codimensional obstacle tube. The models are then compressed entirely together forming a tight mush to fit through the gap and then once through they cleanly separate into a stable pile.

while each succeeding admirably in custom-tuned regimes, often fail with instabilities, egregious constraint violations and/or inaccurate and implausible solutions, as we vary input materials, contact numbers and time step. We also exercise IPC across a wide range of existing and new benchmark tests and demonstrate its accurate solution over a broad sweep of reasonable time-step sizes and beyond (up to $h = 2$s) across challenging large-deformation, large-contact stress-test scenarios with meshes composed of up to 2.3M tetrahedra and processing up to 498K contacts per time step. For applications requiring high-accuracy we demonstrate tight convergence on all measures. While, for applications requiring lower accuracies, e.g. animation, we confirm IPC can ensure feasibility and plausibility even when specified tolerances are lowered for efficiency.

Constitutive Strain Limiting and Thickness Modeling In Chapter 5 IPC is extended to consider codimensional degree-of-freedoms. This enables a unified, interpenetration-free, robust, and stable simulation framework that couples codimension-0,1,2,3 geometries seamless through frictional contact (Figure 1.3). Extending the incremental potential contact onto thin structures necessitates several new contributions. We introduce a C^2-continuous smooth barrier-based energetically-consistent strain limiting model for cloth and show that a constitutive model perspective of such inequality constraints can be used to strictly enforce strain limits without compromising any inconsistency with elastodynamics and contact in the same implicit procedure. We formulate our approach for both isotropic and anisotropic strain limiting requirements. To capture geometrically convincing finite thickness effects of thin structures, we further devise a new offset barrier formulation for the unsigned distance constraint in IPC with details that are key to robust and efficient implementations. To fulfill the extreme accuracy requirement of our offset barrier that fails any existing continuous collision detection (CCD) methods, we design our own additive CCD (ACCD) method based on iteratively accumulating a rigorously derived theoretical lower bound of time of impact. We perform extensive benchmark experiments to validate the efficacy of our method in capturing intricate behaviors of thin structures without suffering from artifacts or numerical difficulties even in extreme deformations and large time steps.

Our research brings in significant contributions to computer graphics in both academia and industry. We at the first time propose a robust and accurate self-contact handling framework with friction and fully implicit nonlinear elastodynamics that can guarantee interpenetration-free trajectories with codimension-0,1,2,3 objects modeled with geometric thickness and fully coupled exact strain limiting. In practice, these technical pieces effectively turns simulation from an extensive tuning procedure to a plug-and-play process where after setting up, the predictive results will come out in a finite amount of time, there will be no need to worry about algorithm failures. What is more exciting is that our methods

Figure 1.3: Our method facilitates robust, stable, intersection-free, and strictly strain-limited large time step simulations of Lagrangian codimensional hyperelastic objects (volumetric bodies, shells, rods, and particles) in a unified, frictional contact-enabled framework.

also enable creating phenomena that are never being able to be produced by prior methods, which opens up many new possibilities for the current computer graphics community. Likewise, our methods have also shown their effectiveness in helping mechanical engineering analysis Jiang et al. [2020], topology optimization Li et al. [2020e], and robotices desgin Li et al. [2020d], and we have multiple ongoing projects to keep pushing this further.

In what follows, we will first introduce the background knowledge of elastodynamics and contact simulation in Chapter 2. Then we present our methods in detail from Chapter 3 to 5 for Decomposed Optimization Time Integrator (DOT), Incremental Potential Contact (IPC), and Constitutive Strain Limiting and Thickness Modeling, respectively. Finally, we conclude our research and discuss a wide range of future work to further extend our work and spread its impact to more research and industry communities in Chapter 6.

Chapter 2

Background

2.1 Time Integration

The dynamic simulation of deformable solids is conducted by time integration, which is usually discretized into a sequence of discrete time steps that compute internal (e.g. elasticity) and external (e.g. gravity) forces on an object and integrate them over time to obtain velocity and position updates to proceed.

2.1.1 Implicit Euler

Taking the most popular time integration scheme, implicit Euler, in computer graphics as an example, in time step t when object has its nodal positions and velocities stacked up into vectors x^t and v^t (Figure 2.1), we have the following update rules:

$$\begin{cases} x^{t+1} = x^t + hv^{t+1} \\ v^{t+1} = v^t + hM^{-1}(f_{\text{int}}(x^{t+1}) + f_{\text{ext}}) \end{cases} \tag{2.1}$$

, where h is the time step size in unit s, M is the nodal mass matrix computed using density and volume of incident elements of the nodes, f_{ext} is the external force, and $f_{\text{int}}(x^{t+1})$ is the internal force evaluated using nodal position x^{t+1}. As shown in Figure 2.1, usually a

volumetric object is represented by a tetrahedral mesh with both its internal and surface nodes as the degree-of-freedoms in the simulation.

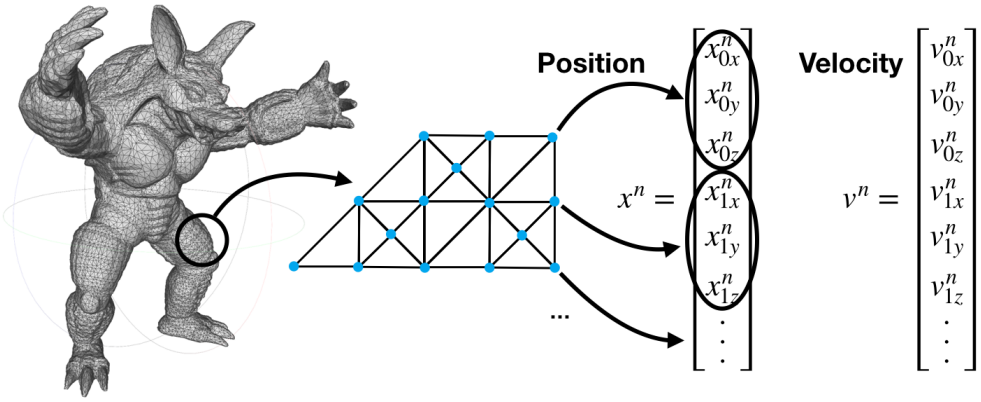


Figure 2.1: **Spatial discretization of deformable solids.**

Since the above update rules for x^{t+1} and v^{t+1} are implicitly defined upon each other, the problem is often solved by first eliminating one of the variables (e.g. v^{t+1}), solve a (non)linear system of equations for the other one (e.g. x^{t+1}), and then compute v^{t+1} using the solved x^{t+1}. One of the most intriguing features of implicit Euler is that it is unconditionally stable even with arbitrarily large time step sizes. This means the simulation with implicit Euler will never explode like explicit time integration schemes even when there are challenging boundary conditions that triggers large deformation of objects. However, to really achieve unconditionally stable simulation in practice without numerical issues, the time integration problem posed by implicit Euler scheme is often reformulated as a smooth minimization problem, and then solved via gradient-based methods with stabilization techniques like line search to guarantee global convergence[1].

[1]Global convergence: the method is guaranteed to converge to a local optimum starting from any initial configuration.

2.1.2 Optimization Time Integrator

To reformulate the problem in Equation 2.1, we eliminate v^{t+1}, gather all terms on one side of the equation and integrate this side w.r.t. x^{t+1} to obtain an incremental potential

$$E(x) = \frac{1}{2}||x - \tilde{x}^t||_M^2 + h^2\Psi(x) \tag{2.2}$$

where $\tilde{x}^t = x^t + hv^t + h^2M^{-1}f_{\text{ext}}$ is called the predictive position, and $\Psi(x) = -\int f_{\text{int}}(x)dx$ is the elasticity potential, which will be discussed in Section 2.2. Then minimizing this $E(x)$ effectively give us the updated position:

$$x^{t+1} = \arg\min_x E(x) \tag{2.3}$$

. This is because by solving this minimization problem we obtain $\nabla E(x^{t+1}) = 0$, which is equivalent to our update rule.

A standard way to robustly solve the reformulated minimization problem with global convergence is to apply Newton-type method with line search. Newton-type method is an iterative approach that approximate the nonlinear objective function using a quadratic function and solve for a search direction to update the iterate in each iteration. Specifically, in each Newton iteration i towards solving the minimization problem in time step t, with the current iterate x^i, the quadratic approximation to the problem given by Taylor expansion is

$$\min_p E(x^i) + p^T\nabla E(x^i) + \frac{1}{2}p^T H(x^i)p \tag{2.4}$$

where $H(x)$ is a symmetric positive definite (SPD) proxy matrix containing 2nd-order information of $E(x)$, and the search direction p can be obtained by setting the gradient of the objective in Equation 2.4 to 0 and solving the linear system

$$p = -H(x^i)^{-1}\nabla E(x^i) \tag{2.5}$$

. After solving for p, x^{i+1} can be obtained as

$$x^{i+1} = x^i + \alpha p \tag{2.6}$$

by line search that ensures $E(x^{i+1}) \leq E(x^i)$, and then we proceed to the next iteration until convergence (Algorithm 1).

Algorithm 1 Newton-type Method for Optimization Time Integrator

procedure NEWTONTYPEMETHOD(x^t, v^t, h, x^{t+1}, v^{t+1})
  $x \leftarrow x^t$
  $E_{\text{prev}} \leftarrow E(x)$, $x_{\text{prev}} \leftarrow x$
  do
    $H \leftarrow$ computeProxyMatrix(x)
    $p \leftarrow -H^{-1}\nabla E(x)$
    $\alpha \leftarrow 1$
    do
      $x \leftarrow x_{\text{prev}} + \alpha p$
      $\alpha \leftarrow \alpha/2$
    while $E(x) > E_{\text{prev}}$
    $E_{\text{prev}} \leftarrow E(x)$, $x_{\text{prev}} \leftarrow x$
  while not converged
  $x^{t+1} \leftarrow x$, $v^{t+1} \leftarrow (x - x^t)/h$

Proxy Matrix The proxy matrix $H(x^i)$ needs to be SPD to ensure that the computed search direction p is a descent direction ($p^T \nabla E(x^i) < 0$) that can guarantee line search success. Here figuring out a way to efficiently compute a SPD $H(x^i)$ per iteration that can also be factorized efficiently and ensures fast convergence is essential to the practical performance of Newton-type method. As we show in Li et al. [2019] (Section 3), accurate 2nd order information of $E(x^i)$ is essential for fast convergence. The widely used Projected Newton (PN) method directly computes $\nabla^2 E(x^i)$ per iteration i and efficiently perform positive semi-definite (PSD) projection per tetrahedron elasticity Hessian to assemble a SPD proxy matrix. PN achieves fast convergence with its bottleneck at computing and factorizing the Hessian in each iteration.

For elastodynamics simulation without contact (what we are discussing up to now), we propose Decomposed Optimization Time Integrator (DOT) Li et al. [2019] to decompose the simulation domain into smaller ones that could be easily factorized in parallel, and utilize quasi-Newton L-BFGS update to perform cheap low-rank update to the proxy matrix so that it does not need to be recomputed per Newton iteration and can still reach super linear convergence. DOT has an overall best timing performance over all alternatives on our test set containing challenging examples undergoing large deformation with large frame-rate time step sizes. See Chapter 3 for more detail.

Linear solve To solve the SPD linear system $Hp = -\nabla E(x)$ to obtain search direction, Cholesky factorization combined with backward substitution can be applied. Eigen Guennebaud et al. [2010] LDLT and CHOLMOD Chen et al. [2008] are two popular SPD linear solvers based on Cholesky factorization, where the latter one supports multi-threading to achieve even better performance. For large scale elastodynamics simulations with around 300K to 500K or a even larger number of nodes, these direct factorization methods may easily run out of memory on nowadays personal desktop/laptop computers because the factors often have much more nonzero entries than the original sparse proxy matrix. In these situations, iterative linear solvers like conjugate gradient methods can be applied.

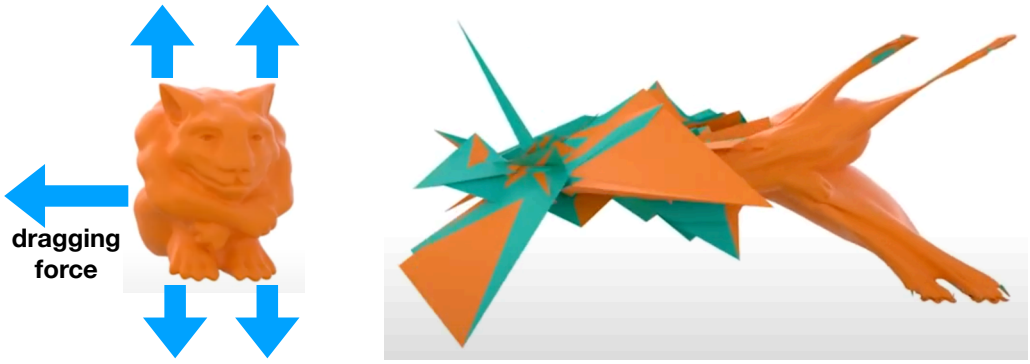
Iterative linear solvers solve the linear systems based on applying matrix-vector multiplications iteratively, so it does not consume more memory than storing the matrix like in direct factorization methods. In some situations like fluids simulation, the matrix-vector multiplications can even be implemented in a matrix-free style, which directly computes the matrix-vector multiplication product without explicitly computing and storing every entry of the matrix, thus achieve even better memory efficiency. However, to achieve nice convergence behavior for the iterative linear solve, suitable preconditioners that could scale down the condition number of the system is essential for often ill-conditioned elastodynamics problems, which is not an issue for direct factorization methods.

In Wang et al. Wang et al. [2020a], we propose Hierarchical Optimization Time Integration (HOT) to solve elastodynamics problem in the material point method (MPM) Stomakhin et al. [2013] setting. Similar to DOT, HOT builds the proxy matrix under L-BFGS framework, and applies conjugate gradient method to solve the linear system with a customized efficient Galerkin multigrid as the preconditioner. HOT achieves the best timing performance compared to all existing implicit MPM solvers. Even for challenging examples with ill-conditioned system and a large number of degree-of-freedoms where applying direct factorization is impractical, HOT can still maintain efficient performance and generate high quality results. See Wang et al. [2020a] for more details.

Line Search Line search is the key to ensure robustness of the time integration. By making sure that each iteration will only decrease the incremental potential that we are minimizing, numerical explosions can never happen within any time step when solved to convergence. Then combined with the implicit Euler update rule which guarantees stability across time steps, the entire simulation can be totally free from explosion. Since we have made sure that the search direction p is always a descent direction, which means given small enough α in each iteration there must be a point with lower energy, we do not need and should not add any iteration cap to the backtracking/bisection line search loop because inproper termination of line search can possibly bring back the numerical explosion issues.

With PN search directions, usually 0 to 2 halfing of α in each iteration would be enough for elastodynamics simulation without contact.

Convergence Criteria How to properly determine convergence of the minimization is in fact an essential component to ensure high quality results (no jittering, damping, or softening artifacts) but it is often lack of attention in traditional methods. To reach better timing performance, some methods in computer graphics often terminate the minimization early by simply setting an iteration cap at e.g. 10. As we show in DOT Li et al. [2019], this absolute number does not work generally for all examples, which could even lead to explosion for challenging scenes (Figure 2.2). Other methods stop the minimization by checking the relative energy decrease (e.g. $|(E(x^i) - E(x^{i+1}))/E(x^i)|$) per iteration. This seemingly good measurement is not adaptive either since different scenes can have very different energy value. In addition, a small relative energy decrease does not always indicate convergence. For a nonconvergent method, it can have vanishing relative energy decreases as it fails to make any significant progress at a certain point, but the iterate can still be faraway from the optimum.



(a) initial frame (b) a middle frame

Figure 2.2: Methods running with a fixed iteration cap can easily running into numerical explosion issues for different examples.

To come up with a termination criteria that appropriately measures the distance from the current iterate to the optimal configuration, and to consider the different scaling in

different examples for generality at the same time, we start by extending the Characteristic Norm (CN) Zhu et al. [2018] that provides this kind of measure for distortion minimization into elastodynamics simulation in DOT Li et al. [2019] (Section 3). Then we observed that CN only characterizes the scale of a single elasticity stiffness per scene, so in HOT Wang et al. [2020a] we extend CN to support complex scenes composed of multiple objects with different elasticity stiffness by applying different CN scaling per dimension on the gradien vector. In IPC Li et al. [2020b] (Section 4), as contact and friction potentials are introduced which have very different form than elasticity, instead of keep exploring CN we found that the norm of search direction p is in fact a nice measure for the distance to convergence up to 2nd order approximation accuracy when PN proxy matrix is used. These appropriate convergence criteria not only allow us to easily set a general tolerance for very different examples to obtain consistently high quality results, but also avoid performing too accurate solves that does not make a big difference for visual applications. See our papers for more details.

2.1.3 Discussion

With the optimization time integration framework, physical simulation of solids could be formulated and realized in a consistent way by introducing new forces like contact and friction either in a potential form or as constraints. In Section 2.2 we introduce hyperelasticity constitutive models in linear finite element setting with a full spectrum of codimension-0,1,2 elements for simulating volumetric objects, shells, and rods. There are different spatial discretization choices for simulating solids that has different trade-offs on multiple aspects of quality, timing performance, and accuracy such as MPM Jiang et al. [2017a], Stomakhin et al. [2013], Wang et al. [2020a], Wolper et al. [2019] and Eulerian-on-Lagrangian methods Fan et al. [2013], Weidner et al. [2018]. In this thesis we focus on finite element method (FEM) and introduce the background knowledge of contact and friction on points, edges, and triangle elements in Section 2.3. In certain cases Dirichlet or Neumann boundary conditions will be needed for prescribing part of the physical system.

Please see the appendix for more details.

When time step size is large, the implicit Euler time integration scheme we focus on here can have noticeable numerical damping. Although this is useful for quickly obtaining static equilibrium with progressive dynamic time stepping that has much better conditioning than direct static solve, improving implicit Euler with better energy conservation while maintaining its stability is a meaningful future work. Other than Newton-type method, there are alternative methods for solving the nonlinear optimization time integration problem such as ADMM Boyd et al. [2011], Overby et al. [2017] that trade accuracy for better efficiency. In this thesis we focus on Newton-type methods for its generality and fast convergence, and encourage future works on accelerating Newton-type methods through smart solving schemes, GPU optimizations, etc.

2.2 Hyperelasticity

Elasticity describes the ability of solids to retain rest shape under external forces. When the object deforms, the elasticity force appears and points to the direction for object to recover rest shape. The essential parts for computing elasticity force and also the force Jacobian (energy Hessian) as needed by our optimization time integrator is to first figure out how deformation is described using nodal positions (our simulation DoFs) and what relation does deformation and elasiticity force has. This depends on the element we use to discretize the continuum body, which split this section into 3 subsections, i.e. tetrahedral elements for volumes (Section 2.2.1), triangle and hinge elements for shells (Section 2.2.2), and segment and ”corner” elements for rods (Section 2.2.3).

2.2.1 Volume

Here we focus on a volume represented by a tetrahedral mesh.

Deformation Gradient The deformation can be described using a tensor field called deformation gradient

$$F = \frac{\partial x}{\partial X}$$

where x is the current nodal position (world space) and X is the rest shape nodal position (material space). For tetrahedra meshes with piecewise linear displacement field per tetrahedron, this tensor field is then piecewise constant, and it can be easily computed as the affine transformation matrix of a tetrahedron from material space to world space:

$$F = \Big[x_2 - x_1, x_3 - x_1, x_4 - x_1\Big] \Big[X_2 - X_1, X_3 - X_1, X_4 - X_1\Big]^{-1} \in R^{3\times 3}$$

Here x_1, x_2, x_3, x_4 represent the current nodal position of the 4 nodes on a tetrahedron and similarly X_1, X_2, X_3, X_4 for rest shape. With the deformation gradient, we can now think of elasticity energies as a measurement of how faraway our deformation gradient F on the tetrahedra are from their closest rotation matrix since rotation only change orientation but not shape. Note that by using F we already ignored the translational motion which does not change shape either.

Neo-Hookean Elasticity Neo-Hookean is one of the most common nonlinear hyperelasticity models for predicting large deformations of elastic mateirals. The energy density function for this model is

$$\psi(F) = \frac{\mu}{2}(\text{tr}(F^T F) - 3) - \mu \ln J + \frac{\lambda}{2}(\ln J)^2$$

for 3D problems where $J = \det F$ and μ and λ are lame parameters computed using Young's modulus $E \in (0, +\infty)Pa$ and Poisson's ratio $\nu \in [0, 0.5)$ of the material via

$$\mu = \frac{E}{2(1+\nu)}, \quad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$$

Here E describes how stiff the material is, and ν measures the ability of a material to preserve volume when deforming. Then to compute the elasticity energy of the entire mesh we integrate $\psi(F)$ over the whole domain which in discrete setting end up with a weighted

sum

$$\Psi(x) = \sum_t V_t \psi(F_t(x))$$

where V_t is the volume of tetrahedra t in material space.

With the definition of the elasticity potential energy, elasticity force and force Jacobians can be easily computed using the derivatives of the elasticity potential via chain rule:

$$\begin{aligned} -f &= \frac{\partial \Psi}{\partial x} = \frac{\partial \Psi}{\partial F}\frac{\partial F}{\partial x} \\ \nabla^2 E(x) &= (\frac{\partial F}{\partial x})^T \frac{\partial^2 \Psi}{\partial F^2}\frac{\partial F}{\partial x} \end{aligned} \tag{2.7}$$

Note that here we flatten F as a vector for simplicity, and since F is a linear function of x, we have $\frac{\partial^2 F}{\partial x^2} = 0$, which leads to a Hessian $\nabla^2 E(x)$ with this simpler form after applying the chain rule. To efficiently compute $\frac{\partial^2 \Psi}{\partial F^2} \in R^{9\times 9}$, singular value decomposition (SVD) could be utilized to take advantage of the inherent sparsity structure of $\frac{\partial^2 \Psi}{\partial F^2}$ due to its rotation invariant property Jiang et al. [2016].

One thing to be careful for FEM simulation with Neo-Hookean is that when element gets degenerated or inverted ($J \leq 0$) during the optimization iterates, the energy density function with $\ln J$ term will be undefined, thus crash the program. To avoid this issue and always ensure positive J for every element, line search filtering Smith and Schaefer [2015] is needed. Before energy decrease line search, we compute a large feasible step size to start from by solving a polynomial equation for each element t to find out the largest step size α_t that first brings its volume to 0:

$$J_t(x_i + \alpha_t p_i) = 0$$

where x_i and p_i are nodal positions and search directions in iteration i, and the smallest positive real root of the above equation is α_t. If there is no smallest positive real root for a certain tetrahedron t, it means that even taking a full step at 1 will not make J_t zero or negative. Given all α_t we can now compute a large feasible step size α_0 as

$$\alpha_0 = (1 - s) \min_t \{\alpha_t\}$$

where s is a slackness coefficient for numerical robustness usually set to 0.2, and then the energy decrease line search can start from α_0. This line search filtering should really be performed at any time when nodal position x gets changed. For example, before moving Dirichlet nodes at the beginning of each time step, line search filtering should also be performed when Neo-Hookean is used. If moving the Dirichlet nodes to target position triggers element inversion, then augmented Lagrangian method can be applied to progressively move the Dirichlet nodes during the optimization. Please see our appendix for more details. One may be curious about whether using an elasticity energy that is also defined for inverted element can avoid this issue. It does, but this kind of invertible elasticity energy cannot always capture the correct elastic behavior, especially when there are extreme compression as we show in Chapter 4.

2.2.2 Shell

Shells refer to thin objects like papers and cloth where their thickness is nearly negligible compared to the other two dimensions. Therefore, for simplicity shells are usually represented by surfaces discretized as triangle meshes. However, to correctly model the elasticity of shells, only applying elasticity on each triangle element is not enough. Because even if shells are thin, they are still volumetric in practice. Thus there should also be elasticity modeled for the thickness direction.

In computer graphics, shells are usually simulated with the discrete shell model Grinspun et al. [2003], where the elasticity is decomposed into two parts residing in the tangent space (membrane) and the normal space (bending) of the shell surface.

Membrane The membrane part of shells' elasticity models the resistence to shearing, stretching, compression, and area changes in the tangent space, or on each triangle in the discrete sense. It is simply the 2D version of the general hyperelastic model we talked about for tetrahedral volume elements in Section 2.2.1. But there are several detailed differences to pay attention.

First the deformation gradient in the tangent space is represented using a 3×2 matrix per triangle:

$$F = \begin{bmatrix} x_2 - x_1, x_3 - x_1 \end{bmatrix} \begin{bmatrix} ||X_2 - X_1|| & \frac{(X_3 - X_1) \cdot (X_2 - X_1)}{||X_2 - X_1||} \\ 0 & \frac{||(X_3 - X_1) \times (X_2 - X_1)||}{||X_2 - X_1||} \end{bmatrix}^{-1} \in R^{3 \times 2}$$

Intuitively, the rest shape of the triangle is reparameterized in a 2D space by placing X_1 at the origin, and then aligning edge $X_2 - X_1$ to the x-axis.

The second difference is that the lame parameter λ is computed differently here for shells as the thickness approaches zero Barber [2002]

$$\lambda = \frac{E\nu}{1 - \nu^2}$$

For integration over the domain, notice that the volume weighting V_t should be computed by multiplying triangle area with thickness but not just using triangle area as V_t.

Finally, a triangle can never get inverted in 3D space because any configuration of the triangle can be achieved by combining rotations with stretches or compressions. Therefore even when Neo-Hookean is used, there is no need to perform line search filtering. The only case that could make Neo-Hookean's log term undefined is when a triangle degenerates to a single line or point. But this rarely happens in floating point operations and can be easily avoided by halfing the step size even when detected.

To model the complex behaviors of cloth with discrete shell model, anisotropic membrane elasticity energy which provides different stiffnesses in different directions is also widely applied. Even with anisotropic models that well capture the directionally dependent strain-stress relation of real cloth, real-world cloth material parameters cannot be directly used in cloth simulation like for volumetric bodies. This is because for piecewise linear displacement field, the elasticity model on a mesh without infinitely large resolution can have stiffer behavior due to discretization error. This behavior is called membrane locking Chen et al. [2019]. In this situation a smaller than real material stiffness is used for cloth simulation, together with a strain limiting constraint to avoid over stretchy artifact. In Chapter 5 we propose a barrier-based constitutive strain limiting model which at the first

time guarantees an upper bound of stretch and perform a fully coupled implicit solve together with accurately resolved contact and elastodynamics. Please see Chapter 5 for more details.

Bending Essentially, bending resistence comes from the different scales of membrane resistences at multiple layers in a shell. Since we represent shells using a single surface in the middle of the volume and assumes that all layers have the same membrane deformation, we need to model bending separately. Note that the normal direction shearing resistences are ignored here as they do not contribute significantly to the dynamics of the shell when thickness is tiny (below around $0.001\times$ than tangent dimensions).

Grinspun et al. [2003] discretize bending energy on every interior edge (hinge) of the triangle mesh. The dihedral angle θ between the two incident triangles of the edge is computed, and a measurement on the difference of this angle between rest shape and current configuration is used as the bending energy:

$$\Psi_{\text{bend}}(x) = \sum_i k \frac{3||\bar{e}_i||^2}{\bar{A}_i}(\theta_i - \bar{\theta}_i)^2$$

. Here $||\bar{e}_i||$ is the rest length of edge i, $\bar{A}_i$ is the sum of the rest areas of the two triangles incident to the hinge, $\bar{\theta}_i$ is the rest dihedral angle, and k is the bending modulus computed as

$$k = \frac{E\xi^3}{24(1-\nu^2)}$$

where ξ is the thickness. Computing the dihedral angle and the derivatives are more complicated than in standard hyperelasticity. We refer to Tamstorf and Grinspun [2013] for detailed derivations.

There are several important points to the robustness of the discrete bending model. First, when the incident triangles are degenerated, the dihedral angle can be undefined. Thus barrier-type membrane elasticity like Neo-Hookean is recommended to prevent this case. Using large enough membrane stiffness that does not trigger membrane locking would often also avoid this issue. In addition, when the two incident triangles get coplanar,

the computation of dihedral angle and its derivatives can also experience degeneracies. Luckily this issue could be prevented by handling contact and ensures interpenetration-free trajectories for the simulated objects. With IPC (Chapter 4), this issue can be safely ignored at the first time in cloth simulation.

2.2.3 Rod

Rods refer to another set of thin objects like hairs and threads where two of their dimensions are much smaller than their length. Thus, rods are often represented simply by a polyline located at their centerline, and simulted using discrete rod model Bergou et al. [2008] in computer graphics. Similar to discrete shells, discrete rod model decomposes rods' elasticity into stretching, bending, and twisting. Here we briefly introduce the stretching and bending energy, while refer to Bergou et al. [2008] for the twisting energy.

Stretching The stretching energy of rods is often simply modeled as a cluster of springs that share the change in length, which can be viewed as a one-dimensional hyperelasticity model measuring the inline stress for rods to keep rest length:

$$\Psi_{\text{stretch}}(x) = \sum_i \bar{A}_i \bar{l}_i \frac{E}{2} \left(\frac{||x_1 - x_2||}{\bar{l}_i} - 1\right)^2$$

where $\bar{A}_i = \pi \bar{r}_i^2$ is the cross section area of segment i with $\bar{r}_i$ being its radius in material space, $\bar{l}_i$ is the rest length of segment i, E is the Young's modulus, and x_1 and x_2 are the current nodal positions of the two end nodes of segment i. Here note that we are still integrating over the volume of rods so the volume weighting writes $\bar{A}_i \bar{l}_i$.

Bending Similar to discrete shell bending model, for rods the bending part of elasticity is also based on angle measurement but at each interior node i:

$$\Psi_{\text{bend}}(x) = \sum_i E r^4 \frac{\pi}{4} \frac{(\kappa b_i)^2}{\bar{l'}_i} \tag{2.8}$$

where E is the Young's modulus, r is the radius, $\bar{l'}_i = \bar{l}_i + \bar{l}_{i-1}$ is the sum of the rest length of the two incident edges on point i, and κb_i is a vector with magnitude $2 \tan \frac{\phi}{2}$ where ϕ is

the angle between the two incident edges:

$$\kappa b_i = \frac{2e^{i-1} \times e^i}{|e^{i-1}||e^i| + e^{i-1} \cdot e^i}$$

Note that the rod bending energy we describe here are only for rods with a straignt rest shape, and we refer to Bergou et al. [2008] for rod bending energy supporting curvy rest shapes. From the expression of κb_i, we can see that it becomes ill-defined when a segment gets degenerated into a single point. Again this issue is also fine if we apply our IPC model (Chapter 4) to guarantee an interpenetration-free simulation.

2.2.4 Discussion

So far we have covered hyperelasticity models from volumes to shells and rods. One may note that there is an important idea throughout – even we use codimensional surface and polyline meshes for geometric representation, we still treat them as continuum regions when formulating the elasticity to correctly capture their elastic behaviors. Using codimensional representations has multiple advantages. First, there are less DoFs on codimensional meshes so the simulation can be more efficient. The other advantage is a little bit tricker, which is that when using volumetric meshes to represent thin objects like cloth and hair, and discretize the elasticity with piecewise linear displacement field, there will be shear locking issues when the resolution is not infinitely high such that the object will be harder to bend than reality due to discretization errors. Using codimensional representations and fully ignore the shearing resistence effectively solve this issue and keeps the simulation efficient. However, one difficulty comes out for codimensional objects – there is no geometric thickness modeling. Even if thin objects has nearly negligible thickness, when they gather together and form a pile, e.g. a deck of cards, thickness will matter for correctly modeling the geometric behaviors. In Chapter 5 we discuss this issue in detail and propose an offset version of IPC Li et al. [2020b] to consistently model thickness in a controllable way for mixed dimensional simulations. This also brings up our next topic for the background chapter – contact modeling.

2.3 Contact and Friction

Up to now we have discussed all the background knowledge about how to simulate elasto-dynamics in a robust and accurate way with codimension-0,1,2 meshes ignoring contact. In this section, we will briefly cover how to accurately model contact in an optimization time integration framework while keep its unconditional stability.

2.3.1 Contact

Contact is a hard problem not only because of its nonsmooth nature that contact forces only exist when objects collide, but also that describing contact between discrete surfaces is challenging. In optimization time integrators, contact can be treated as an inequality constraint which limits the trajectory of the simulated objects in an interpenetration-free admissible set:

$$\min_x E(x) \quad s.t. \quad g(x) \geq 0$$

Here $E(x)$ is our incremental potential discussed in Section 2.1.2, $g(x) \geq 0$ defines the admissible set, but the definition of $g(x)$ has been challenging over decades.

Traditional methods Kane et al. [1999], Kaufman et al. [2008], Verschoor and Jalba [2019] define contact constraints by first finding out all close primitive pairs (point-triangle and edge-edge), and then either use the signed volume of the tetrahedron formed by each primitive pair, or the signed distance between each primitive pair along an estimated normal direction in the constraint function. However, these definitions are all localized, and can be invalid when there are large displacement in a time step (Figure 2.3), which thus limits the time step sizes that can be used for solvers based on these constraints.

Contact Constraints based on Precise Distances In Chapter 4, we propose to formulate contact constraint between triangulated surfaces by computing the precise unsigned distance between each primitive pair, and ensures that in every time step, these distances always

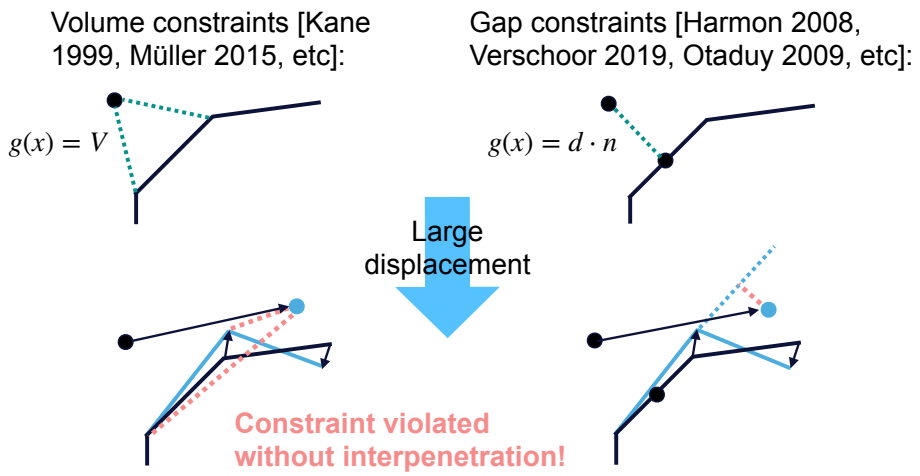


Figure 2.3: **Traditional contact constraints definitions easily get invalid under large displacements.**

stay positive at any point on the optimization update trajectory in each iteration i:

$$\min_x E(x) \quad s.t. \quad \forall k \in \mathscr{C}, a \in [0,1], i \;\; D_k((1-a)x^i + ax^{i+1}) > 0$$

where $\mathscr{C}$ contains all non-incident point-triangle and non-adjacent edge-edge pairs on the surface of the simulated objects, and $D_k(x)$ is defined as

$$\begin{aligned} D_k^{\text{PT}} =& \min_{\beta_1,\beta_2} ||x_P - (x_{T1} + \beta_1(x_{T2} - x_{T1}) + \beta_2(x_{T3} - x_{T1}))|| \\ & s.t. \quad \beta_1 \geq 0, \quad \beta_2 \geq 0, \quad \beta_1 + \beta_2 \leq 1. \end{aligned} \tag{2.9}$$

for point(x_P)-triangle($x_{T1}x_{T2}x_{T3}$) pairs, and

$$\begin{aligned} D_k^{\text{EE}} =& \min_{\gamma_1,\gamma_2} ||x_{11} + \gamma_1(x_{12} - x_{11}) - (x_{21} + \gamma_2(x_{22} - x_{21}))|| \\ & s.t. \quad 0 \leq \gamma_1, \gamma_2 \leq 1. \end{aligned} \tag{2.10}$$

for edge($x_{11}x_{12}$)-edge($x_{21}x_{22}$) pairs. These small optimization problems can be written as piecewise smooth functions depending on the relative positions of the primitives, see Chapter 4 for more details. Since our D_k precisely measure the distance of each primitive pair under any configuration, it is always consistent and valid to describe whether the simulation has interpenetration. With continuous collision detection (CCD) to directly

ensure that in each optimization iteration, the update to x^i is without any interpenetration by taking a step size smaller than the "time of impact", our problem formulation can be simplified to

$$\min_x E(x) \quad s.t. \quad \forall k \in \mathscr{C} \;\; D_k(x) > 0$$

together with a CCD line search filtering in each optimization iteration similar to the one that prevents element inversion.

Barrier Method with Smooth Clamping Now with our rigorous mathematical definition of the constraints, next thing is how to solve it. A common way to tackle inequality constrained optimization problems is to apply sequential quadratic programming (SQP), where the objective and constraints are approximated and then a quadratic optimization problem with linear constraints is solved in each iteration to progressively move towards the local optimum. This is a direct extension to Newton-type methods without constraints, but conducting line search in SQP is tricky, which makes it hard to maintain robustness. What is worse, the linearized constraints can even form infeasible problems, making the approximated subproblems unsolvable. Therefore, we apply barrier method to turn the constrained problem into an "unconstrained" one, and then apply the same projected Newton method to solve it:

$$\min_x E(x) + \kappa \sum_k b(D_k(x))$$

where b is barrier function that goes to infinity when the input distance approaches zero. Apparently, considering all candidates from $\mathscr{C}$ in the barrier is too expensive as its size grows quadratically with the number of DoFs. In addition, this is also unnecessary as faraway primitive pairs may never collide even if we do not explicitly constrain them. Thus we construct a special barrier function with only a local support but is everywhere at least C^2-smooth to keep our smooth optimization method (PN) effective. Please see Chapter 4 for more details.

2.3.2 Friction

Intuitively, friction forces exist in tangent space of each contact point, and they are in the opposite direction of the relative sliding displacement or the moving tendency if there is no relative sliding. The challenge of simulating friction lies in figuring out the static friction force and accurately capturing the transition between static and dynamic modes. Mathematically, friction forces are defined by Maximum Dissipation Principle (MDP) Moreau [1973]. For each active contact primitive pair k, with $u_k = T_k(x)^T(x - x^t)$ being the local relative sliding displacement where $T_k(x) \in R^{3n\times 2}$ is the tangent operator, friction force F_k can be defined as

$$F_k = \begin{cases} -\mu\lambda_k T_k \frac{u_k}{||u_k||} & ||u_k|| > 0 \\ -\mu\lambda_k T_k f & ||u_k|| = 0 \end{cases} \tag{2.11}$$

. Here λ_k is the normal force magnitude, and f can take any vector with $||f|| \leq 1$, for which we clearly see the difficulties here. When $||u_k|| = 0$, F_k can have non-smooth jumps, and displacement alone is not enough to define F_k because for static friction, the force magnitude and direction depends on other forces and the momentum balance that will be reached. This means that static friction force and nodal displacement are implicitly coupled and needs to be solved simultaneously with F_k also being an unknown variable. In addition, unlike hyperelasticity models, there is no well-defined $D(x)$ such that $-\frac{\partial D_k(x)}{\partial x} = F_k$, so friction cannot be easily involved in our optimization time integration framework.

In Chapter 4, we propose a mollified friction model that transit between static and dynamic friction smoothly. This introduces the approximation error for static friction that it can act on primitive pairs with a tiny nonzero sliding displacment. Our model expose an upper bound of the velocity magnitude that experience static friction as a controllable parameter to let users weigh between smoothness (solver efficiency) and accuracy of the friction. Then we discretize friction temporally in a semi-implicit way, such that λ_k and T_k are fixed as their values at the end of last time step. These make the rest of the terms in F_k integrable, and thus we can come up with a potential energy for this approximated friction

model and easily incorporate it into our optimization time integrator. Please see Chapter 4 for more details.

Chapter 3

Decomposed Optimization Time Integrator

3.1 Introduction

Simulation of deformable-body dynamics is a fundamental and critical task in animation, physical modeling, and design. Algorithms for computing these simulations have rapidly advanced the accuracy, consistency, and controllability of generated elastodynamic trajectories. Key to these advances are a long-standing range of powerful implicit time stepping models to numerically time-integrate semi-discretized PDEs Ascher [2008], Butcher [2016], Hairer and Wanner [1996], Hairer et al. [2008]. With a few restrictions, *incremental potentials* Ortiz and Stainier [1999] can be constructed for these models. These are energies whose local minimizers give an implicit time step model's forward map at each time step Hairer et al. [2006], Kane et al. [2000], Kharevych et al. [2006]. Adopting this variational perspective has enabled the application and development of powerful optimization methods to minimize these potentials and efficiently forward step dynamic simulations Chao et al. [2010], Liu et al. [2017], Martin et al. [2011], Overby et al. [2017].

For deformable nonlinear materials, the incremental potential is the sum of a convex, quadratic discrete kinetic energy and a generally nonconvex, strongly nonlinear deformation

energy weighted by time step size; see e.g. (3.3) below. Then, as step size and/or simulated distortions increase, the influence of this latter deformation energy term dominates. This, in turn, requires the expensive modeling of nonlinearities. As we will soon see, too weak an approximation of nonlinearity leads to large errors, artifacts, and/or instabilities, while too large or frequent an update makes other methods impractically slow — reducing progress and imposing significant memory costs.

To address these challenges we propose the Decomposed Optimization Time Integrator (DOT), a new domain-decomposed optimization method for minimizing per time step incremental potentials. DOT builds a novel quadratic matrix penalty decomposition to couple non-overlapping subdomains with weights constructed from missing subdomain Hessian information. We use this decomposition's Hessian, evaluated once at start of time step, as an inner-initializer to perform undecomposed L-BFGS time step solves on a domain mesh with a single copy of the simulation vertices. Advantages of DOT are then:

No vertex mismatch. While the decomposed Hessian is built per subdomain it is evaluated with a consistent set of vertices taken from the full, undecomposed mesh. Penalty weights thus add missing second-order Hessian data to subdomain vertices from neighbors across decomposition boundaries. This gives an initializer to our global L-BFGS solve. We use it to perform descent on the full mesh coordinates. This means vertices on interfaces are never separated, by construction.

Convergence. DOT adds no penalty forces nor gradients. DOT penalty terms only supplement our preconditioning matrix. Descent steps then precondition the undecomposed and unaugmented incremental potential's gradient and converge directly to the underlying undecomposed system's solution.

Efficiency. Subdomain Hessians are parallel evaluated and factorized once per time step. We show that they can also be applied (via small backsolves) in parallel as initializer at each iteration. Results then simply need to be blended together. This process is inserted

between first and second efficient, low-rank updates of each quasi-Newton step. This couples individual, per-domain backsolves together for a global descent step. Resulting iterations are then more effective than L-BFGS approaches and yet faster than Newton.

Contributions In summary, DOT converges at large, fixed-size time steps, ensuring stable continued progress of high-quality simulation output. DOT is an automated and robust optimization method especially suited for simulations with nonlinear materials, large deformations, and/or high-speed dynamics. By automated we mean users need not adjust algorithm parameters or tolerances to obtain good results when changing simulation parameters, conditions, or mesh sizes. By robust we mean a method should solve every reasonable time step problem to any requested accuracy given commensurate time, and only report success when the accuracy has been achieved. To achieve these goals we

- construct a quadratic penalty decomposition to couple non-overlapping subdomains with weights constructed from missing subdomain Hessian information;
- propose a resulting method using this domain-decomposition as inner initializer for undecomposed, full mesh, quasi-Newton time step solves;
- develop a line-search initialization method for reduced evaluations;
- extend Zhu et al.'s [2018] characteristic norm for consistent elastodynamic simulations; and
- perform extensive comparisons of recent performant nonlinear methods for solving large-deformation time stepping.

3.2 Problem Statement and Preliminaries

We focus on solving one-step numerical time-integration models with variational methods. Minimizing an incremental potential, E, we update state from time step t to $t+1$ with a

local minimizer

$$x^{t+1} = \underset{x \in \mathbb{R}^{dn}}{\operatorname{argmin}} E(x, x^t, v^t), \tag{3.1}$$

for n vertex locations in d-dimensional space stored in vector x, with corresponding velocities v. Here $E(x, x^t, v^t)$ is a combined local measure of deformation and discrete kinetic energy, and x is potentially subject to boundary and collision constraints.

The deformation energy, $W(x)$, is (e.g. with linear finite elements) expressed as a sum over elements e in a triangulation T (triangles or tetrahedra depending on dimension),

$$W(x) = \sum_{e \in T} v_e w\big(F_e(x)\big), \tag{3.2}$$

where $v_e > 0$ is the area or volume of the rest shape of element e, w is an energy density function taking the deformation gradient as its argument, and F_e computes the deformation gradient for element e.

Concretely, as recent papers on solvers for time stepping in graphics Bouaziz et al. [2014], Gast et al. [2015], Liu et al. [2013, 2017], Narain et al. [2016], Overby et al. [2017] have almost exclusively focused on the implicit Euler time stepping model, our results in the following sections will do so as well in order to provide side-by-side comparisons. For implicit Euler the incremental potential is

$$E(x, x^t, v^t) = \frac{1}{2} x^T M x - x^T M x^p + h^2 W(x). \tag{3.3}$$

Here $x^p = x^t + h v^t + h^2 M^{-1} f$, f collects external and body forces, M is the finite element mass matrix, and velocity is updated by the implicit Euler finite difference stencil $v^{t+1} = \frac{x^{t+1} - x^t}{h}$. In the next sections we restrict our attention to solving a single time step and so, unless otherwise indicated, we simplify by specifying the incremental potential as $E(x) = E(x, x^t, v^t)$.

Notation. Throughout we will continue to apply superscripts t to indicate time step, while reserving subscripts incrementing i for inner solver iteration indices, and correspondingly subscripts with increments of j for quantities associated with subdomains.

3.3 Related Work

Domain Decomposition Domain decomposition strategies have long been an effective means of efficiently parallelizing large-scale linear problems Quarteroni et al. [1999]. Direct connections with the Schur complement method and block-decomposed iterative solvers for linear algebra are then easily established for faster solves. These methods offer exciting opportunities for scalable performance that have also been applied in graphics Huang et al. [2006], Kim and James [2012], Liu et al. [2016], Sellán et al. [2018], but can also suffer from slower convergence or gapping between imperfectly joined interfaces Kim and James [2012]. Domain decompositions, including the classic Schwartz methods, have also been extended to the nonlinear regime Dolean et al. [2015], Xiao-Chuan and Maksymilian [1994]. Here parallelization is easily obtained; however, methods generally offer linear convergence rates Dolean et al. [2015]. While speeds can potentially be further improved by incorporating ADMM-type strategies Parikh and Boyd [2012], overall application is still hampered by ADMM's underlying first-order convergence and the overhead of working with additional dual variables Boyd et al. [2011]. For DOT, we design a domain decomposition without dual variables that employs energy-aware coupling penalty terms in the subdomain Hessian proxy that can be efficiently and easily parallelized for evaluation. We then integrate this model as an inner component of a customized limited memory BFGS to gain higher-order coupling across domains and so regain super-linear convergence with a small, additional fixed linear overhead.

Optimization-based time integrators in graphics. Position-based dynamics methods have become an increasingly attractive option in animation, being fast and efficient, but not controllable nor consistent Müller et al. [2007]. Both Projective Dynamics (PD) Bouaziz et al. [2014] and extended position-based dynamics Macklin et al. [2016] observe that with small but critical modifications, iterated local and global solves can be brought more closely into alignment with implicit Euler time stepping, although various limitations in terms of consistency, controllability, and/or materials remain. Liu and colleagues [2017] observe

that, in the specific case of the ARAP energy density function, PD is exactly a Sobolev-preconditioning, or in other words, an inverse-Laplacian-processed gradient descent of the incremental potential for implicit Euler. Note that beyond ARAP this analogy breaks down. Following on this observation they propose extending PD's Sobolev-preconditioning of the implicit Euler potential for models with a range of hyperelastic materials and thus varying distortion energies. They further improve convergence of this implicit-Euler solver by wrapping the Sobolev-preconditioner as an initializer for the limited-memory BFGS algorithm (L-BFGS) to minimize the incremental potential. In the following we refer to this algorithm as LBFGS-PD.

Concurrently, Overby et al. [2016, 2017] observe that PD can alternately be interpreted as a particular variant of ADMM with local variables formulated in terms of the deformation gradient. Overby and colleagues then show that an algorithm constructed in this way can likewise be extended to minimize the implicit Euler potential over a range of hyperelastic materials. In the following we will refer to this algorithm as ADMM-PD. Both ADMM-PD and LBFGS-PD improve upon PD both in extending to a wide range of hyperelastic materials and to increasing efficiency for ARAP-based deformation Liu et al. [2017], Overby et al. [2017].

Optimization-based time integration In brief, albeit by a circuitous path, time stepping solvers in computer graphics, starting from position-based methods, come full circle back to minimizing the standard incremental potential for fully nonlinear materials. In this setting, traditional time stepping in computational mechanics generally focuses on preconditioned gradient-descent methods, often augmented with line-search Ascher [2008], Deuflhard [2011]. These methods find local descent directions p_i, at each iterate i, by preconditioning the incremental potential's gradient with a Hessian proxy H_i, so that $p_i = -H_i^{-1}\nabla E(x_i)$. For example, Sobolev-preconditioning Neuberger [1985] for implicit Euler gives a fixed efficient preconditioner built with the Laplacian L so that $H_i = M + h^2 L$ for all time steps Liu et al. [2017].

Newton-type methods Preconditioning with the Hessian $H_i = \nabla^2 E(x_i)$, gives Newton-stepping. To gain descent for large-time stepping this generally requires both line search and a positive-definite fix of the stiffness matrix which otherwise can make the total Hessian indefinite Nocedal and Wright [2006]. Many closely related positive-definite corrections are suggested Nocedal and Wright [2006], Shtengel et al. [2017], Teran et al. [2005]. Here, following Liu and colleagues, we employ Teran et al.'s per-element projection and refer to this method as Projected Newton (PN). Newton-type methods like these have rapid convergence but solution of the Hessian, either directly or via Newton-Krylov variations, are generally reported as too costly per-iteration to be competitive with less-costly preconditioners and scale poorly Brown and Brune [2013], Liu et al. [2017], Overby et al. [2017]. We revisit and analyze this assumption in Section 4.7 with a suite of well-optimized solvers. We observe that in some ranges PN with a direct solve is actually competitive and can even outperform LBFGS-PD and ADMM-PD, while in others the situation is indeed often reversed. In order to retain Newton's superlinear convergence with improved efficiency, lagged updates of the Hessian preconditioner every few time steps have long been proposed Deuflhard [2011], Hecht et al. [2012], Shamanskii [1967]. While in some cases this can be quite effective, the necessary frequency of these updates can not be pre-determined as it depends on local simulation state, e.g., transitions, collisions, etc, and so generally leads to unsightly artifacts and inaccuracies, including ghost forces and instabilities Brown and Brune [2013], Liu et al. [2017].

Quasi-Newton methods Alternately, quasi-Newton BFGS methods have long been applied for simulating large-deformation elastodynamics Deuflhard [2011]. L-BFGS can be highly effective for minimizing potentials. However, an especially good choice of initializer is required and makes an enormous difference in convergence and efficiency Nocedal and Wright [2006]; e.g., Liu et al.'s [2017] Sobolev-initializer. Directly applying the Hessian is of course clearly an ideal initializer for L-BFGS; unfortunately, it is generally a too costly option. Lazy updates of the Hessian, for example at start of time step, have also been

considered Brown and Brune [2013], Liu et al. [2017]; but again have been discarded as too costly and limiting in terms of scalability Liu et al. [2017]. In the following we will refer to this latter method as LBFGS-H. Our development of DOT leverages these start of time step Hessians at the beginning of each incremental potential solve, applying a new, inner domain decomposition strategy to build an efficient, per-time step re-initialized method that outperforms or closely matches best-per-example prior methods across all tested cases.

Trade-offs across time-integration solvers As both LBFGS-PD and ADMM-PD appeared concurrently and, to our knowledge, have not previously been analyzed side-by-side, we do so here for the first time along with PN and LBFGS-H, across a range of examples. We show how all these methods alternately excel or fall-short in varying criteria and examples; see Section 4.7. ADMM-PD has fast initial convergence that rapidly trails off, characteristic of ADMM methods, and thus is generally slowest, often failing to meet even moderate accuracy tolerances. LBFGS-PD on the other hand performs admirably when a

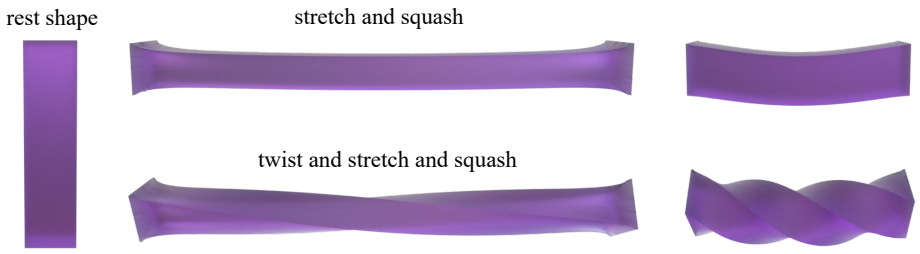


Figure 3.1: ***Uniform deformation examples.***

global, uniform deformation is applied on a uniform mesh such as in bar stress-tests, see e.g., Fig. 3.1. For these cases the Laplacian preconditioner effectively smoothes global error generating rapidly converging simulations Zhu et al. [2018]. However, for everyday non-uniform deformations on unstructured meshes common in many application, LBFGS-PD will rapidly lose efficiency. Often, as deformation magnitude grows, LBFGS-PD becomes slower than a direct application of PN and LBFGS-H, despite LBFGS-PD's per-iteration efficiency; see Section 4.7.

Based on our analysis of where these prior methods face difficulties we propose DOT

which efficiently, robustly, and automatically converges to user-designated accuracy tolerances with timings that outperform or closely match these previous methods across a wide range of practical stress-test deformation scenes on unstructured meshes. DOT combines the advantages of per-time step updates of second-order information with an efficient quasi-Newton update of per-iteration curvature information. The two are integrated together by a novel domain-decomposition we construct that avoids slow coupling convergence challenges faced by traditional domain decomposition methods. Together in DOT these components form the basis for a scalable, highly effective, domain-decomposed, limited memory quasi-Newton minimizer of large time step incremental potentials with super-linear convergence; see Section 4.7.

3.4 Method

We solve each time step by iterated descent of the incremental potential. At each inner iteration i, we calculate a descent direction p_i and a line search step, $\alpha_i \in \mathbb{R}_+$. We then update our current estimate of position at time $t+1$ by $x_{i+1}^{t+1} \leftarrow x_i^{t+1} + \alpha p_i$.

3.4.1 Limited memory quasi-Newton updates

We begin with a quasi-Newton update. At each iterate i, the standard BFGS approach Nocedal and Wright [2006] exploits the secant approximation from the difference in successive gradients, $y_i = \nabla E(x_{i+1}) - \nabla E(x_i)$, compared to the difference in positions $s_i = x_{i+1} - x_i$. This approximation is applied as low-rank updates to an inverse proxy matrix $D_i = H_i^{-1}$ (roughly approximating $\nabla^2 E^{-1}$) so that $D_{i+1} y_i = s_i$. Limited memory BFGS (L-BFGS) then stores for each iteration i just an initial, starting matrix, D_1 and the last m $\{s, y\}$ vector pairs (we use $m = 5$). Joint update and application of D_{i+1} is then applied *implicitly* with just a few, efficient vector dot-products and updates, with the application of the matrix D_1 sandwiched in between the first and second low-rank updates of each step.

Given iterate i's implicitly stored proxy D_i, and the initial proxy, D_1, we set Q_i :

$\mathbb{R}^{dn\times dn} \times \mathbb{R}^{dn} \to \mathbb{R}^{dn}$ to apply the limited-memory quasi-Newton update and application of D_{i+1}. Then

$$p_i = Q_i\big(D_1, x_i\big) = -D_{i+1}\nabla E(x_i). \tag{3.4}$$

L-BFGS's performance depends greatly on choice of initializer D_1 Nocedal and Wright [2006]. Setting D_1 to a weighted diagonal, for example, offers some improvement, as do various inverses of block diagonals taken from the Hessian. Similarly we can, as discussed above, apply Liu et al.'s [2017] inverse Laplacian or even, as proposed by Brown and Bune [2013], could use lagged updates of the Hessian inverse itself — just once every few time steps so as to not perform too many expensive updates and solves.

3.4.2 Initalizing L-BFGS

To consider the relative merits of potential initializers for time stepping we adopt a simple perspective. We observe that each of the above initializers corresponds to the application of a single iteration of a different, nonlinear method as an inner step in the L-BFGS loop Zhu et al. [2018]. The better the convergence of the inner method generally the better performance of the outer L-BFGS method. From this perspective Liu et al. [2017], for example, can be seen to apply an inner iterate of inverse-Laplacian preconditioned descent and so gain the well-appreciated smoothing of the parent Sobolev Gradient Descent method Neuberger [1985]. Or alternately we could consider applying the start of time step inverse Hessian $D_1 = \nabla^2 E(x^t)^{-1}$. The resulting optimization applies an iteration of the inexact Newton method within L-BFGS that could potentially augment quasi-Newton curvature with second-order information via the tangent stiffness matrix.

While this latter strategy could be highly effective to improve convergence Liu et al. [2017], it is expensive both to factorize at every time step and to backsolve at every iteration. Moreover, we observe that initializing with the full Hessian inverse may be an unnecessarily global approach. In many cases large deformations are concentrated locally and their effects are only communicated across the entire material domain over multiple

time steps. Likewise, initializing with the full inverse Hessian limits scalability as we must work with its factors in every quasi-Newton iterate; see Section 4.7.

Motivated by these observations we instead design a method to update L-BFGS with Hessian information using a decomposition that allows us to efficiently store and apply local second-order information at each iterate. To do so we first construct a simple quadratic penalty decomposition that connects non-overlapping subdomains with automatically determined stiffness weights. Our decomposed optimization time integrator is then formed by applying a single inexact-Newton descent iterate of this method as an inner initializer of an undecomposed, full mesh L-BFGS step. The resulting method, as we see Section4.7, then obtains well-improved, scalable convergence for large time step, large deformation simulations.

3.4.3 Decomposition

An illustration of our decomposition is in Fig. 3.2. We decompose our full domain Ω into s non-overlapping subdomains $\{\Omega_1, ..., \Omega_s\}$ partitioned by interfaces $\Gamma = \{\gamma_1, ..., \gamma_{|\Gamma|}\}$ along element boundaries with duplicate copies of each interface vertex assigned to each subdomain participating at that interface. We construct decompositions that both approximately minimize number of edges along interfaces (and so cross-subdomain communication) and balance numbers of nodes per subdomain Karypis and Kumar [1998]. For each subdomain j we then likewise store its interfaces in $\Gamma_j \subseteq \Gamma$.

Each subdomain j then has n_j vertices stored in vector $x_j = (y_j^T, z_j^T) \in \mathbb{R}^{dn_j}$, formed by the concatenation of a copy of its interface vertices z_j, and its remaining subdomain vertices, y_j. Concatenation of all subdomain vertices $\hat{x} = (x_1^T, ..., x_s^T)^T \in \mathbb{R}^{d\hat{n}}$ then includes unshared interior vertices and duplicated interface vertices so that $\hat{n} > n$. We then collect interface vertices from the fully connected mesh as $x_\Gamma \subset x$. Note that x_Γ are then distinct from their duplicated interface copies in the decomposition and can serve as consensus variables for subdomain boundaries. Finally, per subdomain j, we construct interface restriction matrices R_{Γ_j} that extract vertices from x_Γ participating in

that subdomain's interfaces Γ_j.

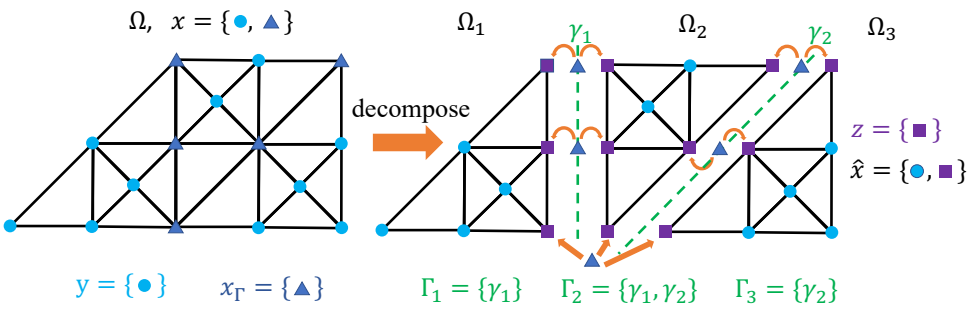


Figure 3.2: ***DOT's decomposition.*** Layout and notation for a mesh (left) which is decomposed into three subdomains (right).

3.4.4 Penalty Potential

Building time stepping incremental potentials on this decomposed system we get a nicely separable sum $\sum_{j\in[1,s]} E_j(x_j)$, where $E_j = E|_{\Omega_j}$ is the restriction of the incremental potential to subdomain j. This separable potential could be efficiently optimized as it allows us to minimize each subdomain independently. Doing so, however, would erroneously decouple subdomains.

One possibility for reconnecting subdomains would be to add explicit coupling constraints. However, this would also require dual variables in the form of Lagrange-multipliers. As we are designing our decomposition for insertion within a quasi-Newton loop, additional dual variables are undesirable. Instead, we adopt a simple quadratic penalty for each interface in Γ to pull subdomains together, with consensus variables x_Γ as the bridge. Augmenting the above separable potential with this penalty gives

$$L(\hat{x}, x_\Gamma) = \sum_{j\in[1,s]} \left(E_j(x_j) + \frac{1}{2}\|z_j - R_{\Gamma_j} x_\Gamma\|^2_{K_j} \right). \tag{3.5}$$

Here each K_j is a penalty stiffness matrix that pulls subdomain j's interface vertices towards globally shared consensus positions stored in x_Γ. By driving $|K_j|$ towards ∞ as we optimize L we could hypothetically remove interface mismatch. However, finding

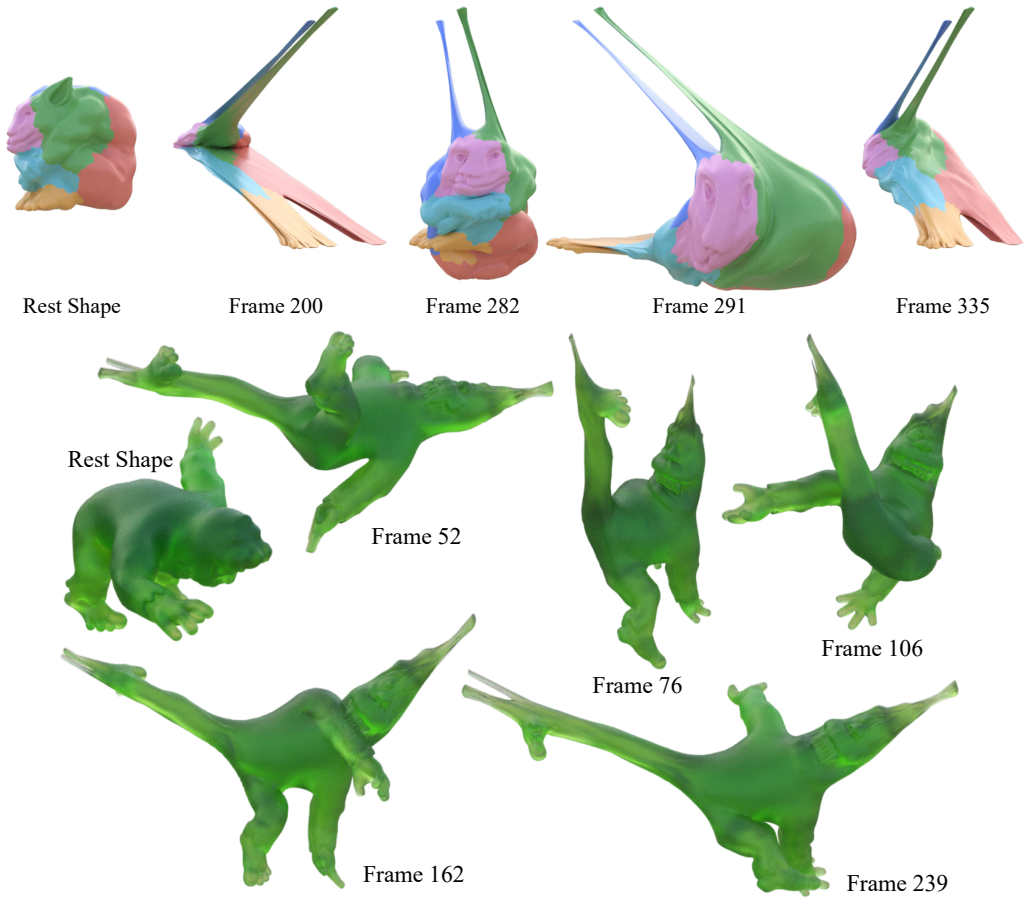


Figure 3.3: ***Deformation stress-tests.*** DOT simulation sequences of hollow cat (top) and monkey (bottom) large-deformation, high-speed, stress-tests.

practical values for K_j that sufficiently pull interface edges together, while avoiding ill-conditioning, is generally challenging. For example, if we choose a penalty that is too stiff it would pull interface domains together too tightly at the expense of overwhelming elasticity and inertia terms in the potential. On the other hand, if we choose a penalty that is too soft, subdomain potentials dominate and interface consensus would be underestimated.

3.4.5 Interface Hessians

We instead seek a penalty that automatically balances out the missing elastic and inertial information along interfaces. In our setting we observe that this missing information is

directly available and forms a natural, automated weighting function for the augmented potential in (3.5) above. Consider that each subdomain's penalty term is effectively just a proxy for the missing energies from neighboring subdomains. Specifically, the Hessian for subdomain Ω_j is

$$\frac{\partial^2 L}{\partial {x_j}^2} = \begin{bmatrix} \frac{\partial^2 E_j(x_j)}{\partial {y_j}^2} & \frac{\partial^2 E_j(x_j)}{\partial y_j \partial z_j} \\ \frac{\partial^2 E_j(x_j)}{\partial z_j \partial y_j} & \frac{\partial^2 E_j(x_j)}{\partial {z_j}^2} + K_j. \end{bmatrix} \tag{3.6}$$

Here, the upper and off-diagonal (symmetric) terms nicely match the unrestricted Hessian on the full domain Ω,

$$\frac{\partial^2 E_j(x_j)}{\partial {y_j}^2} = \frac{\partial^2 E(x)}{\partial {y_j}^2} \text{ and } \frac{\partial^2 E_j(x_j)}{\partial y_j \partial z_j} = \frac{\partial^2 E(x)}{\partial y_j \partial (R_{\Gamma_j} x_\Gamma)}. \tag{3.7}$$

However, our lower diagonal does not as

$$\frac{\partial^2 E_j(x_j)}{\partial {z_j}^2} \neq \frac{\partial^2 E(x)}{\partial (R_{\Gamma_j} x_\Gamma)^2}, \tag{3.8}$$

irrespective of whether or not we have agreement on subdomain interface vertex locations. This is due to the absence of element stencils connecting interface nodes to neighboring subdomains. While this missing information is not currently present in $\partial^2 L / \partial {x_j}^2$, it gives us a natural definition for penalty weights. We assign weights K_j so that they generate the missing interfacial Hessian information from adjacent elements bridging across neighboring subdomains:

$$K_j(x) = \frac{\partial^2 E(x)}{\partial (R_{\Gamma_j} x_\Gamma)^2} - \frac{\partial^2 E_j(x_j)}{\partial {z_j}^2}. \tag{3.9}$$

Our quadratic penalty then recovers otherwise missing components of kinetic and elastic energy stencils across interface boundaries. Note that, as in PN, we project all computed Hessian stencils.

3.4.6 Discussion

To minimize (3.5) we could apply alternating iterations solving first for optimal $\hat{x}$ and then for x_Γ. Indeed, when we add a constraint Lagrangian term, $\sum_{j \in [1,s]} \lambda_j^T (z_j - R_{\Gamma_j} x_\Gamma)$ to (3.5), this alternating process generates a custom ADMM Boyd et al. [2011] algorithm. We

initially considered this approach and find that it significantly outperforms classic ADMM. We observe that automatic weighting with (3.9) nicely smooths error at boundaries; see e.g. Fig. 3.7. However, we also find that this strategy is still not competitive with undecomposed methods like PN, as too much effort is exerted to close gaps between subdomains. Instead, we apply a single iteration of our penalty decomposition as an efficient, inner initializer with second-order information. We then insert it within an outer, undecomposed quasi-Newton step solved on the full mesh. This ensures unnecessary effort is not spent pulling interfaces together, and updates our decomposition with global, full-mesh, curvature information.

3.4.7 Initializer

We now have all necessary ingredients to construct DOT. We initialize an L-BFGS update with a single Newton iteration of our quadratic penalty in (3.5) as follows. At start of time step $t+1$ we define subdomain variables from current positions x^t as $\hat{x}^t = Sx^t$. Here $S \in \mathbb{R}^{d\hat{n}\times dn}$ is a separation matrix that maps the n full-mesh vertices x to subdomain coordinates with duplicated copies of interfacial vertices. Application of the inverse Hessian is then

$$D_1^{t+1} = BS^T\left(\partial^2 L(\hat{x}^t, x_\Gamma^t)/\partial\hat{x}^2\right)^{-1} S \in \mathbb{R}^{dn\times dn}. \tag{3.10}$$

Here $B \in \mathbb{R}^{dn\times dn}$ is a diagonal averaging matrix with diagonal entries corresponding to vertex v set to $1/n_v$ where n_v is the number of duplicate copies for the corresponding vertex v in the decomposition. Iteration i of DOT is then applied by application of

$$p_i = Q_i(D_1^{t+1}, x_i), \tag{3.11}$$

followed by our custom line search and update detailed below.

3.4.8 Construction

Construction and application of the per time step DOT is efficient. At start of solve we first compute and store factors of the augmented Hessians per subdomain,

$$H_j = \nabla^2 E_j(x_j^t) + K_j(x_j^t). \tag{3.12}$$

We then observe that

$$D_1^{t+1} = BS^T \text{diag}(H_1^{-1}, ..., H_s^{-1})S, \tag{3.13}$$

where $\text{diag}(\cdot)$ constructs a block diagonal $\mathbb{R}^{d\hat{n}\times d\hat{n}}$ matrix. Application of D_1^{t+1} as initializer in L-BFGS is then applied in parallel computation to any global vector $q \in \mathbb{R}^{dn}$. We first separate q to repeated subdomain coordinates $(q_1^T, ..., q_s^T)^T = Sq \in \mathbb{R}^{d\hat{n}}$. Then we independently backsolve subdomains with their factors to obtain $r_j = H_j^{-1}q_j$. Finally, we lift all r_j back to full mesh coordinates by blending with $r = BS^T(r_1^T, ..., r_s^T)^T$ to average duplicate coordinates appropriately. See Algorithm 1 below for the full DOT pseudocode.

3.4.9 Line Search

After our quasi-Newton update we next perform backtracking line search on p_i to ensure sufficient descent. For quasi-Newton methods rule-of-thumb Nocedal and Wright [2006] is to always initialize line search with unit step length, i.e. $\alpha_{\text{start}} = 1$, to ensure large steps will take advantage of rapid convergence near solutions. In the large time step, large deformation setting however, we observe that the situation is nonstandard. We often start far from solutions and so need to balance large, initial step estimates against costs of repeated energy evaluations. For this purpose we apply an alternative initializer for line search. For each search direction p_i, DOT initializes with the optimal length of the one-dimensional quadratic model,

$$\alpha_{\text{start}} = \max\left(10^{-1}, \frac{-p_i^T \nabla E(x_i)}{p_i^T \nabla^2 E(x^t) p_i}\right). \tag{3.14}$$

Here the lower bound handles the rare instances where this local fit is too conservative.

3.4.10 Algorithm

Algorithm 1 contains the full DOT algorithm in pseudocode. The dominant costs for runtime are energy costs: evaluations, gradients, Hessians, and SVDs; and subdomain augmented Hessian costs: assembly, factorization and backsolves. Otherwise, costs for our quasi-Newton loop itself are linear (dot products, vector updates, etc). Memory cost is primarily the once per-timestep Cholesky factorization of the subdomain augmented Hessians and their corresponding backsolves per iteration.

3.5 Evaluation

3.5.1 Implementation and Testing

We implemented a common test-harness code to enable the consistent evaluation of all methods with the same optimizations for common tasks. In comparisons of our ADMM-PD and LBFGS-PD implementations with their release codes Liu et al. [2017], Overby et al. [2017] we observe an overall 2-3X speed-up across examples.

Our code is implemented in C++, parallelizing assembly and evaluations with Intel TBB, applying CHOLMOD Chen et al. [2008] with AMD reordering for all linear system solves, and METIS Karypis and Kumar [1998] for all decompositions. Note that for LBFGS-PD and ADMM-PD we perform their global Laplacian backsolves in parallel, per-dimension, and likewise factorize only the scalar Laplacian, one-time as a precompute. Following Overby et al. [2017], ADMM-PD's per-element energy minimizations are performed in diagonal space for efficiency. Common energy evaluations and gradients are optimized with AVX2 parallelization to achieve roughly $4\times$ speedup. To do so we extended the open-source SIMD SVD library McAdams et al. [2011] to support double precision for our framework.

Unless otherwise indicated, all experiments below were performed on a six-core Intel 3.7GHz CPU and were simulated with times step sizes of either 10, 25 (majority), or 40

Algorithm 2 Decomposed Optimization Time Integrator (DOT)

Given: x^t, E, S, B, ϵ

Initialize and Precompute:

$i = 1$

$H_j^{-1} \leftarrow \left(\nabla^2 E_j(x_j^t) + K_j(x^t)\right)^{-1}, \forall j \in [1, s]$ // get Cholesky factors

$g_1 \leftarrow \nabla E(x_1)$

// quasi-Newton loop to solve time step $t+1$:

while $\|g_i\| > \epsilon h^2 \langle W \rangle \|\ell\|$ **do** // termination criteria (Section3.5.2)

$q \leftarrow -g_i$

for $a = i-1, i-2, .., i-m$

$s_a \leftarrow x_{a+1} - x_a, \quad y_a \leftarrow g_{a+1} - g_a, \quad \rho_a \leftarrow 1/(y_a^T s_a)$

$\alpha_a \leftarrow \rho_a s_a^T q$

$q \leftarrow q - \alpha_a y_a$

end for

$(q_1^T, ..., q_s^T)^T \leftarrow Sq$

$r_j \leftarrow H_j^{-1} q_j, \forall j \in [1, s]$

$r \leftarrow BS^T (r_1^T, ..., r_s^T)^T$

for $a = i-m, i-m+1, .., i-1$

$\beta \leftarrow \rho_a y_a^T r$

$r \leftarrow r + (\alpha_a - \beta) s_a$

end for

$p_i \leftarrow r$

$\alpha_{\text{start}} \leftarrow \max\left(10^{-1}, -\left(p_i^T \nabla E(x_i)\right) / \left(p_i^T \nabla^2 E(x^t) p_i\right)\right)$

$\alpha \leftarrow \text{LineSearch}(x_i, \alpha_{\text{start}}, p_i, E)$

$x_{i+1} \leftarrow x_i + \alpha p_i$

$g_{i+1} \leftarrow \nabla E(x_{i+1})$

$i \leftarrow i + 1$

end while

ms (as indicated). For consistent comparison with prior work we focus our analysis on examples with the implicit Euler using the fixed co-rotational material Stomakhin et al. [2012]. Below in Section 3.5.6 we also explore DOT with the Stable Neo-Hookean material Smith et al. [2018].

We compile CHOLMOD with MKL LAPACK and BLAS, supporting multi-threaded linear solves, and set the number of threads per linear solver to take full advantage of multi-core architecture per method. Specifically, the single, full linear systems in each PN and LBFGS-H iteration are solved with 12 threads per solver, while the multiple smaller systems in each LBFGS-PD, ADMM-PD, and DOT iteration are solved simultaneously with 1 thread per solver.

Finally, note that we summarize detailed statistics from all of our experiments in tables in our supplemental document Li et al. [2019]. All tables referred to in the following are found there.

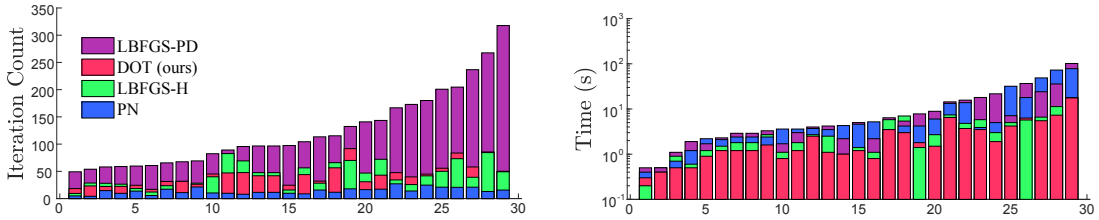


Figure 3.4: ***Convergence and Timings.*** Per-example comparisons of iteration (top) and timing (bottom) costs, per frame, achieved by each time step solver. Note that the bars are overlaid, not stacked.

3.5.2 Termination Criteria

We next focus on the important question of when to stop iterating an individual time step solve. Clearly, for most simulation applications, it is not reasonable to manually monitor the quality of each individual iterate, within every individual time step solve, in order to decide when to stop. Analogously, applying the same fixed number of iterations for all time steps, no matter the method, will not suffice as different steps in a simulation

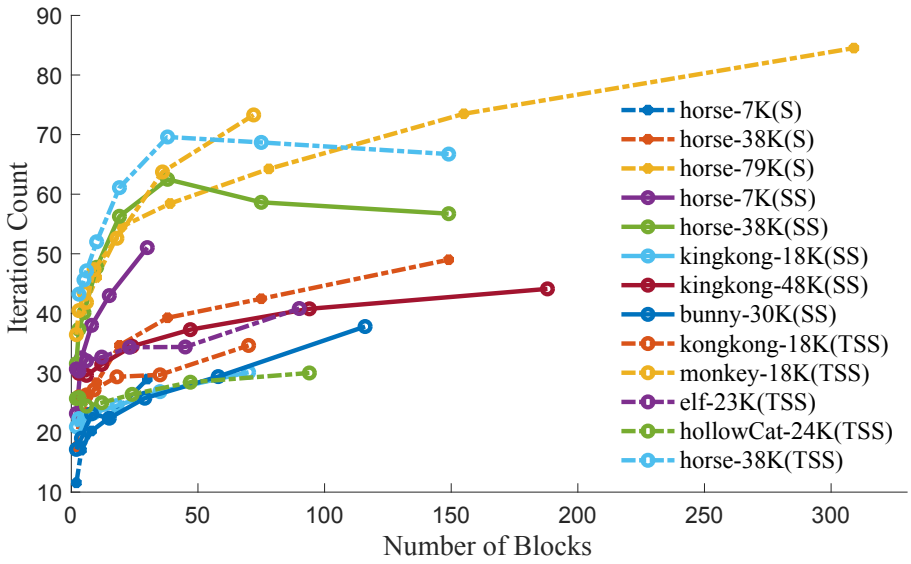


Figure 3.5: ***Subdomain to iteration growth.*** We plot iterations per DOT simulation example as the size of the decomposition increases. We observe a trend of sublinear-growth in iteration count with respect to the number of subdomains, revealing promising opportunities for parallelization.

will have more or less nonlinearity involved and so will require correspondingly different amounts of work to maintain consistent simulation quality. Without this simulations can and will accrue inconsistencies, artifacts and instabilities. Similarly, relative error measures for termination are not satisfactory because they are fulfilled when an algorithm is simply unable to make further progress and so has stagnated far from a working solution. On the other hand, the gradient norm provides an excellent measure of termination for scientific computing problems where we seek high-accuracy. However, for animation and many other applications we seek a way to reliably stop at stable, good-looking, consistent simulations with reasonably low error, *but* not necessarily with vanishingly small gradients. As observed by Zhu et al [2018], in these cases even gradient norm tolerances are not suitable for choosing a termination criteria.

To address this problem for minimizing deformation energies in statics problems, Zhu and colleagues propose a dimensional analysis of the deformation energy gradient.

They derive a characteristic value for the norm of the deformation gradient over the mesh. Then, applying it as a scaling factor to the gradient norm obtains consistent quality solutions across a wide range of problems, object shapes and energies. Here we observe that a small modification of this analysis gives a corresponding scaling factor for the incremental potential. We begin with the base case of the characteristic value for the norm of the deformation gradient over the mesh as $\langle W \rangle ||\ell||$. Here $\langle W \rangle$ is the norm of the deformation energy Hessian of a single element at rest, ℓ is a vector in $\mathbb{R}^n$ with each entry the surface area of each simulation node's one-ring stencil. For dynamic time stepping we are then minimizing with the incremental potential (3.3) and so have a weighted sum of discrete kinetic and deformation energies. At stationarity of (3.3) we then have $M(x - x^t - x^p) + h^2 \nabla W(x) = 0$. We then have proportional measures $-\frac{1}{h^2} M(x - x^t - x^p)$ at corresponding scale with $\nabla W(x)$, and similarly $\frac{1}{h^2} M(x - x^t - x^p) + \nabla W$. Then, for consistent convergence checks across examples and methods at each iteration we simply check the rescaled incremental potential

$$||\nabla E|| < \epsilon h^2 \langle W \rangle ||\ell||. \tag{3.15}$$

In the following evaluations we refer to this measure as the characteristic gradient norm (CN). After extensive experimentation across a wide range of examples, using Projected Newton (PN) as a baseline, we find consistent quality solutions across methods using (3.15) at increments of ϵ; see the supplemental videos Li et al. [2019]. Moreover, we find that solutions satisfying $\epsilon = 10^{-5}$ are the first to avoid visual artifacts that we consistently observe below this tolerance, including variable material softening, damping, jittering and explosions. For all experiments in the remainder of this section, unless otherwise indicated, we thus set our termination criteria with $\epsilon = 10^{-5}$ using (3.15). All CN convergence measures thus apply rescaling of the gradient norm. Convergence in CN per method then confirms convergence to the same (reference) solution provided, e.g., by Newton's method.

3.5.3 Iteration Growth with Domain Size

We next investigate the scalability of DOT as the number of subdomains in the decomposition grows. We apply DOT across thirteen simulation examples ranging in mesh resolution from 7K to 136K vertices with a range of moderate to extreme deformation test scenes. Each scene is solved to generate ten simulated seconds, time stepped at $h = 25$ms. For each simulation example we create a sequence of decompositions by requesting target subdomain sizes starting at 16K vertices (where possible) down by halves to 256 vertices. We then simulate all of the resulting decompositions, spanning from 2 to 309 subdomains, with DOT. In figure 3.5 we plot the the number of DOT iterations per simulation example as the size of its decomposition increases. We observe a trend of sublinear-growth in iteration count (per simulation) as the number of subdomains increases, revealing promising opportunities for parallelization of DOT.

3.5.4 Performance

Fig. 3.5 suggests parallelization opportunities for DOT across a wide range of decomposition sizes. Of course, how to best exploit these opportunities will vary greatly with platform. Here we begin with two modest exercises starting with a six-core Intel 3.7GHz CPU, 64GB memory. We script a set of increasingly challenging dynamic deformation stress-test scenarios across a range of mesh shapes and resolutions. See, for example, Fig.s 1.1 and 3.3, our supplemental document and video Li et al. [2019] for example details. For each simulation we target DOT to simply utilize available cores and so set the number of subdomains in all simulations in this first exercise to six. In Fig. 3.4 we summarize runtime statistics for these examples with DOT, PN, LBFGS-H, and LBFGS-PD across the full set of these examples. We also test ADMM-PD on the full set of examples but find it unable to converge on any in the set. See our convergence analysis below for more details on ADMM-PD's behavior here.

Timings Across this set we observe DOT has the fastest runtimes, for all but three examples (see below for discussion of these), over the best timing for each example across all converging methods: PN, LBFGS-H, and LBFGS-PD. In general DOT ranges from 10X to 1.1X faster than PN, from 2.5X to 1X faster than LBFGS-H, and from 11.4X to 1.6X faster than LBFGS-PD. The one exception we observe is in the three smallest meshes of the horse stretch scalability example where the deformation is slow, so that the solves are close to statics. Here we observe that LBFGS-H is, on average 1.3X faster than DOT on smaller meshes up to 79K vertices. Then, as mesh size increases to 136K and beyond, here too DOT becomes faster. Finally, for even larger meshes LBFGS-H can not fit in memory; see *Scaling* below. Importantly, across examples, we observe that PN, LBFGS-PD and LBFGS-H alternate as fastest as we change simulation example. Here PN ranges from 2X slower to 4.6X faster than LBFGS-PD, while LBFGS-H often seems to be a best choice among the three, but also can be slowest, i.e., ranging from 1.1X to 1.7X slower. Trends here suggest that LBFGS-PD tends to do better for more moderate deformations while PN and LBFGS-H often pull ahead for more extreme deformations but this is not entirely consistent and it is challenging to know which will be the better method per example, a priori. Finally, as we see below, both PN and LBFGS-H do not scale well to larger systems.

Scaling To examine scaling we successively increase mesh resolution for the horse TSS example. On this machine (recall 64GB memory) both PN and LBFGS-H can not run models beyond 308K vertices while DOT and LBFGS-PD can continue for examples up to and including 754K vertices. We compare the performance among methods at these two extremes and find that DOT is 1.9X faster than LBFGS-H at 308K nodes, while it is 2.7X faster than LBFGS-PD at 754K nodes, where both PN and LBFGS-H can not run.

Changing Machines Next, in Table 4, we report statistics as we exercise DOT on both our six-core machine and a sixteen-core Xeon 2.4GHz CPU. Here we correspondingly set the number of subdomains in all simulations to six and sixteen respectively. Although overall timings of course change, we see that DOT similarly maintains the fastest runtimes

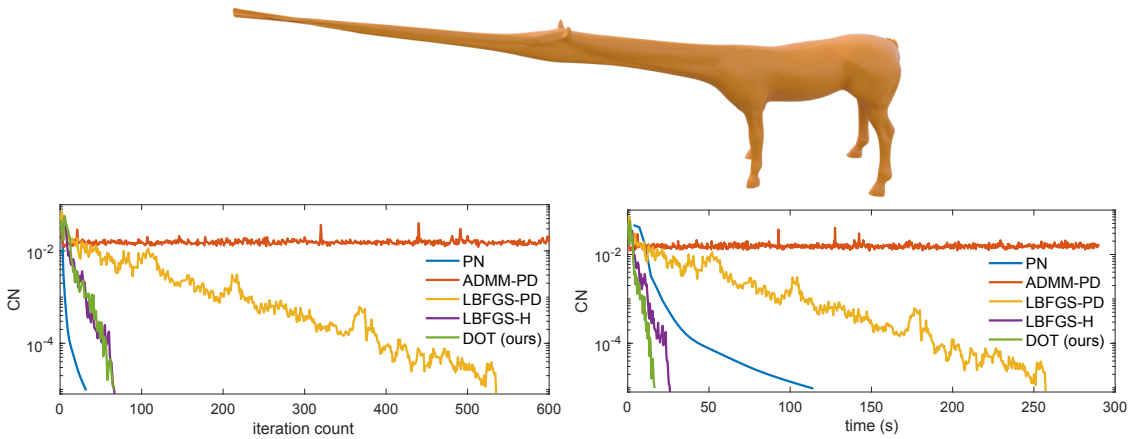


Figure 3.6: ***Convergence comparisons.*** **Top:** We compare the convergence of methods for a single time step midway through a stretch script with a 138K vertex mesh; measuring error with the characteristic gradient norm (CN) see Section 3.5.2. **Middle:** We observe DOT's super-linear convergence matches LBGHS-H and closely approaches Project Newton's (PN), while LBFGS-PD lags well behind and ADMM-PD does not converge. **Bottom:** comparing timing, DOT pulls ahead of PN and LBFGS-H, with lower per-iteration costs.

across both machines, over the best timing for each example between PN, LBFGS-H and LBFGS-PD. Here these three latter methods again swap one another in speeds per example.

Changing Decompositions Then, to confirm that there is a wide range of viable subdomains settings for DOT, we examine performance as we vary subdomain sizes using the same simulation example set from Section 3.5.3 above. In Table 5 we summarize statistics for these simulations and observe that all simulations match or out-perform the best result timing between PN, LBFGS-H and LBFGS-PD; the only exceptions being the smallest 7K mesh horse simulations are slightly outperformed by PN and LBFGS-H.

3.5.5 Convergence

DOT balances efficient, local second-order updates with global curvature information from gradient history. In Fig. 3.6 we compare convergence rates and timings across methods

for a single time step midway through the large stretch example of the 138K vertex horse mesh. We observe super-linear convergence for DOT, matching LBFGS-H's and closely approaching PN's, while LBFGS-PD lags well behind, and ADMM-PD characteristically does not converge to even a much lower tolerance than the one requested. In turn, comparing timings, DOT out-performs PN and LBFGS-H with lower per-iteration cost. In Fig. 3.7 we visualize DOT's characteristic process: in the first few iterations error is concentrated at interfaces and is then quickly smoothed out by the successive iterations.

In addition to PN, LBFGS-PD, LBFGS-H and ADMM-PD, we also investigated standard Jacobi and Gauss-Seidel decomposition methods Quarteroni et al. [1999], and experimented with applying incomplete Cholesky as initializer for L-BFGS. Convergence rates for the former two methods are slow, making them impractical compared to the above methods. The latter method, interestingly, sometimes performs well, but is inconsistent as it also can be slow and even fails to converge in other cases; see our supplemental document Li et al. [2019] for details.

3.5.6 Varying Time Step, Material Parameters and Model

Here we compare behavior as we change material parameters and vary over a range of frame-size time steps. We apply a twist, stretch and squash (TSS) script to an 18K vertex monkey mesh. For the same example we apply time step sizes of 10, 25 and 40 ms respectively and vary material parameters comparing across a range of Young's modulus and Poisson ratio. As summarized in Table 2, across all examples DOT obtains the fastest runtimes with trends showing timings increasing for all methods as we increase time step size and Poisson ratio. For varying stiffness, timings for DOT stay largely flat, while here PN, LBFGS-H, and LBFGS-PD become slower with softer materials. Finally, we consider changing the material model. We apply the Stable Neo-Hookean model to run the monkey TSS example. Relative timings stays consistent as with the FCR model, where DOT ranges from 1.5X to 2.4X faster than all alternatives.

3.6 Summary

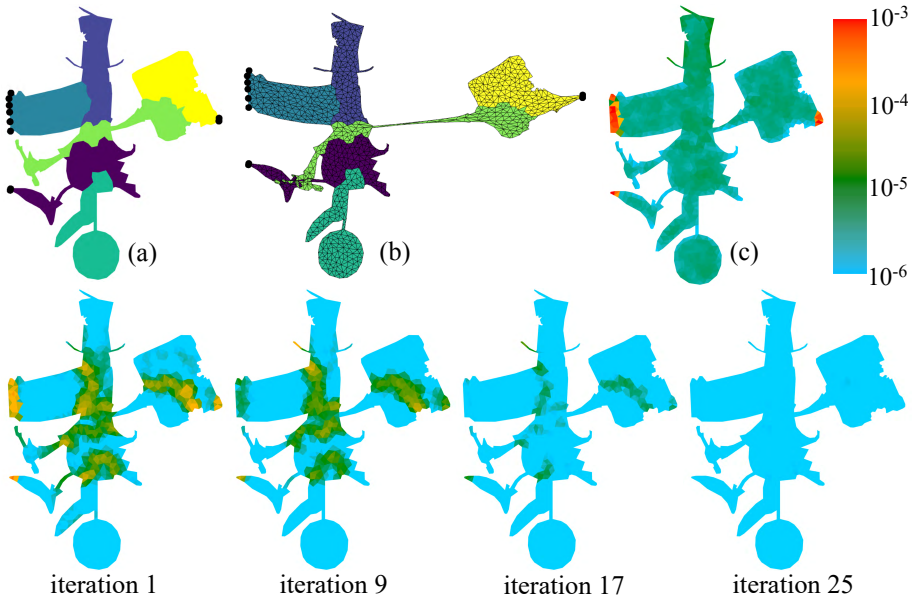


Figure 3.7: ***Residual Visualization.*** With a decomposition (a), current deformation in (b), and starting error in (c), we visualize DOT's characteristic convergence process. In the first few iterations error is concentrated at interfaces and is then rapidly smoothed by the successive iterations.

In this work we developed DOT, a new time step solver that enables efficient, accurate and consistent frame-size time stepping for challenging large and/or high-speed deformations with nonlinear materials. So far we have focused on CPU parallelization on moderate commodity machines with medium-scale meshes ranging from 31K to 4.2M tetrahedra. However, as we see in Section 3.5.3 above, DOT's sub-linear scaling of iterations for more decompositions makes extensions of DOT to large-scale systems exceedingly promising to pursue. Concurrently for meshes at all scales we observe that while rule-of-thumb matching domain count to available cores already exposes significant speed-up and robustness we have also seen in Section 4.7 that across a wide range of decomposition sizes we maintain a significant and consistent advantage. Thus we are also excited to explore DOT with recent advances in batch-processed factorizations and solves on the GPU, e.g., with MAGMA

Abdelfattah et al. [2017], where L-BFGS low-rank updates can be efficiently performed via map reduce. Likewise, while DOT offers speed and robust convergence at large time step, decomposition also offers other promising opportunities. One exciting direction is applying recent mesh-adaptation strategies Schneider et al. [2018] which can now be performed on-the-fly, independently per subdomain.

When deformations are mild, uniformly distributed, at slow speeds and/or on small meshes we see that the win for DOT is sometimes not as significant. If such cases are to be expected then certainly alternatives such as PN, LBFGS-H, LBFGS-PD and ADMM-PD may be reasonable choices as well. Our experience suggests that most scenarios are not likely to limit the scope of a simulation tool to these cases. If so, then we propose DOT as a one-size-fits-all method that improves or closely matches performance in these easier and gentler cases and then shines for the challenging scenarios were previous methods become stuck and/or exceedingly slow.

Our evaluation of DOT has so far focused on invertible energies. Efficiency with noninverting energies, e.g., Neo-Hookean, may require custom-handling of the elasticity barrier, e.g, by line search curing Zhu et al. [2018], and is an interesting future direction.

DOT solves elastodynamics for both rapidly moving and fixed boundary conditions over a wide range of simulation conditions. As in ADMM-PD and LBFGS-PD, integrating standard collision resolution methods using penalty or hard-constraints, can be directly encorporated in DOT. However, custom-leveraging DOT's structure for efficient contact processing remains exciting future work.

Finally, while our decompositions from METIS have been effective they are certainly not optimal for optimization-based time stepping. It would be interesting to explore custom decompositions which specifically take advantage of the incremental potential's structure and even adaptive decompositions per-time step. Likewise, while the simple strategy of matching subdomain count to cores already enables simple and easy-to-implement advantages, it is of course in no way optimal. An exciting future investigation is exploring per-task custom decompositions based on compute resources available.

Chapter 4

Incremental Potential Contact

4.1 Introduction

Contact is ubiquitous and often unavoidable and yet modeling contacting systems continues to stretch the limits of available computational tools. In part this is due to the unique hurdles posed by contact problems. There are several intricately intertwined physical and geometric factors that make contact computations hard, especially in the presence of friction and nonlinear elasticity.

Real-world contact and friction forces are effectively discontinuous, immediately making the time-stepping problems very stiff, especially if the contact constraints are enforced exactly. On the other hand, even small violations of exact contact constraints (which are nonconvex) can lead to impossible to untangle geometric configurations, with a direct impact on physical accuracy and stability. In addition, stiff contact forces often lead to extreme deformations, resulting in element inversions for mesh-based discretization. Friction modeling then introduces further challenges with asymmetric force coupling and rapid switching between sliding and sticking modes.

In this work, our goal is to achieve very high robustness (by which we mean the absence of catastrophic failures or stagnation) for contact modeling even for the most challenging elastodynamic contact problems with friction. Robustness should be obtained independent

of user-controllable accuracy in time-stepping, spatial discretization and contact resolution, while maintaining efficiency required to solve large-scale problems. At the same time we wish to also ensure that all accuracies – across the board – are efficiently attainable (of course with additional cost) when required.

With these goals in mind, we reexamine the contact problem formulation, discretization and numerical methods from scratch, building on numerous ideas and observations from prior work.

Our Incremental Potential Contact (IPC) solver is constructed for mesh-based discretizations of nonlinear volumetric elastodynamic problems supporting large nonlinear deformations, implicit time-stepping with contact, friction and boundaries of arbitrary codimension (points, curves, surfaces, and volumes). A key principle we follow is that while the physics and shape can be approximated arbitrarily coarsely, the geometric constraints (absence of intersections of the approximate geometry and inversions of elements) are maintained exactly at all times. We achieve this for essentially arbitrary target time steps and spatial discretization resolution.

The key element of our approach is the formulation of the contact problem and the customized numerical method to solve it. As a starting point, we use an exact contact constraint formulation, described in terms of an *unsigned* distance function, and rate-based maximal dissipation for friction.

For every time step, we solve the discrete nonlinear contact problem with a given tolerance using a smoothed barrier method, ensuring that the solution remains intersection-free at all intermediate steps. We use a comparably smoothed, arbitrarily close, approximation to static friction, also eliminating the need for an explicit Coulomb constraint, and cast friction forces at every time step in a dissipative potential form, using an alternating lagged formulation. All forces can then be solved by unconstrained minimization.

Our barrier formulation for contact has several important properties: 1) it is an almost everywhere C^2 function of the unsigned distances between mesh elements, C^1 continuous for a measure-zero set of configurations; 2) its support is restricted to a small part of the

configuration space close to configurations with contact. The former property makes it possible to use rapidly converging Newton-type unconstrained optimization methods to solve the barrier approximation of the problem, the latter ensures that additional contact forces are applied highly locally *and* that only a small set of terms of the barrier function need to be computed explicitly during optimization. Jointly they enable stable, conforming contact between geometries.

To guarantee a collision-free state at every time step, feasibility is maintained throughout all nonlinear solver iterations: the line search in our customized Newton-based solver is augmented with efficient, filtered continuous collision detection (CCD) accelerated by a conservative CFL-type contact bound on line search step sizes. Friction forces are resolved directly in the same solver via our lagged potential with geometric accuracy improved by alternating updates.

4.1.1 Contributions

In summary, IPC solves nonlinear elastodynamic trajectories that are intersection- and inversion-free, efficient and accurate (solved to user-specified accuracies) while resolving collisions with both nonsmooth and codimensional obstacles. To our knowledge, this is the first implicit time-stepping method, across both the engineering and graphics literature, with these properties.

We demonstrate the efficacy of IPC with stress tests containing large deformations, many contact primitive pairs, large friction, tight collisions as well as sharp and codimensional obstacles. Our technical contributions include

- A contact model based on the unsigned distance function;
- An almost everywhere C^2, C^1-continuous barrier formulation, approximating the contact problem with arbitrary accuracy, with barrier support localized in the configuration space, enabling efficient time-stepping;

- Contact-aware line search that continuously guarantees penetration-free descent steps with CCD evaluations accelerated by a conservative-bound contact-specific CFL-inspired filter;

- A new variational friction model with smoothed static friction, formulated as a lagged dissipative potential, robustly resolving challenging frictional contact behaviors; and

- A new benchmark of simulation test sets with careful evaluation of constraint and time stepping formulations along with an extensive evaluation of existing contact solvers.

4.2 Contact Model

We focus on solving numerical time-integration for nonlinear volumetric elastodynamic models with contact. These models can interact with fixed and moving obstacles which can be of arbitrary dimension (surfaces, curves and points). The simulation domain is discretized with finite elements. Given n nodal positions, x, finite-element mass matrix, M, and a hyper-elastic deformation energy, $\Psi(x)$, the contact problem extremizes the extended-value action

$$S(x) = \int_0^{\mathrm{T}} \Big(\frac{1}{2} \dot{x}^T M \dot{x} - \Psi(x) + x^T (f_e + f_d) \Big) dt.$$

on an *admissible set of trajectories* $\mathscr{A}$, which we discuss below. Here f_e are external forces and f_d are dissipative frictional forces. We assume, for simplicity, that all object geometry is discretized with n-dimensional piecewise-linear elements, $n = 1, 2, 3$.

Admissible trajectories We construct a new definition of admissibility based on unsigned distance functions that has a number of advantages. Most importantly, in the context of our work, it naturally allows us to formulate exact contact constraints in terms of constraints on collisions between pairs of primitives (triangles, vertices and edges), and can be

defined in exactly the same way for objects of any dimensions (points, curves, surfaces and volumes).

Specifically we define trajectories $x(t)$, with $x \in \mathbb{R}^{3n}$ as *intersection-free*, if for all moments t, $x(t)$ ensures that the distance $d(p, q)$ between any distinct points p and q on the boundaries of objects is positive. In the space of trajectories, the set of intersection-free trajectories forms an open set $\mathscr{A}_I$, as it is defined by strict inequalities. Optimization problems may not have solutions in this set; for this reason, we add the limit trajectories to it, which involve contact. Specifically, we define the *set of admissible trajectories* $\mathscr{A}$ as the *closure* of $\mathscr{A}_I$. In other words, a trajectory is admissible, if it is intersection-free, *or* there is an intersection-free trajectory arbitrarily close.

Note that this closure is not equivalent to replacing the constraint $d(p, q) > 0$ with $d(p, q) \geq 0$; the latter is always satisfied for unsigned distances, so that all trajectories would be admissible. This is not the case for our definition. Consider for example, a point moving towards a plane. If its trajectory touches the plane and then turns back, an arbitrarily small perturbation makes it intersection-free, and the trajectory is in $\mathscr{A}$. However, if the trajectory crosses the plane small perturbations do not make it intersection-free. This highlights the need for our treatment even in the volumetric setting as the boundaries of our mesh upon which we impose constraint are exactly surfaces whose potential collisions include the point-face case above.

We can describe $\mathscr{A}_I$ directly in terms of constraints on unsigned distances d between surface primitives (vertices, edges, and faces in the simulation surface mesh and domain boundaries). We denote this set of mesh primitives $\mathscr{T}$. Equivalently to the more general definition above, a piecewise-linear trajectory $x(t)$ starting in an intersection-free state x_0 is admissible, if for all times t, the configuration $x(t)$ satisfies positive distance constraints $d_{ij}(x(t)) > 0$ for all $\{i, j\} \in \mathscr{B}$, where $\mathscr{B} \subset \mathscr{T} \times \mathscr{T}$ is the set of all non-adjacent and non-incident surface mesh primitive pairs.

We then observe that the distance between any pair of primitives is bounded from below by the distance for triangle-vertex and edge-edge pairs, *if* there are no intersections. For

this reason, it is sufficient to enforce constraints $d_k(x(t)) > 0$ *continuously* in time, for all $k \in \mathscr{C} \subset \mathscr{B}$ where $\mathscr{C}$ contains all *non-incident point-triangle and all non-adjacent edge-edge pairs* in the surface mesh.

Time discretization Discretizing in time, we can directly construct discrete energies whose stationary points give an unconstrained time step method's update Ortiz and Stainier [1999]. Concretely, given nodal positions x^t and velocities v^t, at time step t, we formulate the time step update for new positions x^{t+1} as the minimization of an Incremental Potential (IP) Kane et al. [2000], $E(x, x^t, v^t)$, over valid $x \in \mathbb{R}^{3n}$. For example the IP for implicit Euler is then simply

$$E(x, x^t, v^t) = \tfrac{1}{2}(x - \hat{x})^T M(x - \hat{x}) - h^2 x^T f_d + h^2 \Psi(x), \tag{4.1}$$

where h is the time step size and $\hat{x} = x^t + hv^t + h^2 M^{-1} f_e$. IPs for implicit Newmark (see Section 4.7) and many other integrators follow similarly by simply treating their update rule as stationarity conditions of a potential with respect to variations of x^{t+1}.

Addition of contact constraints restricts minimization of the IP to admissible trajectories Kane et al. [1999], Kaufman and Pai [2012] and so yields for our model the following time step problem:

$$x^{t+1} = \underset{x}{\operatorname{argmin}}\, E(x, x^t, v^t),\ x^\tau \in \mathscr{A}, \tag{4.2}$$

where x^τ, $\tau \in [t, t+1]$, is the linear trajectory between x^t and x^{t+1}.

Our goal is to define a numerical method for approximating the solution of this problem in (4.2). Solving it is challenging due to the complex nonlinearity of the admissibility constraint, especially in the context of large deformations.

In turn, when frictional forces in (4.2) include frictional contact, solving the time step problem becomes all the more challenging as f_d is now governed by the Maximal Dissipation Principle Moreau [1973] and so must satisfy further challenging, asymmetric and strongly nonlinear consistency conditions Simó and Hughes [1998]. We present a friction formulation in Section 4.5 that is naturally integrated into this formulation via a lagged dissipative potential.

We further define a set of piecewise-linear surfaces as ϵ-separated, if the distance between two boundary points of the set is at least ϵ, unless these are on the same element of the boundary. An ϵ-separated trajectory is then a trajectory for which surfaces stay ϵ-separated. We denote the set of such trajectories $\mathscr{A}_\epsilon$.

To handle contact constraints, in our algorithm, we use the following overall approach: (a) the IP function E is unmodified on $\mathscr{A}_\epsilon$ – the set of trajectories for which any ϵ-separated trajectory extremizes the action are preserved; (b) we introduce a barrier term that vanishes for trajectories in $\mathscr{A}_\epsilon$ and diverges as the distance between any two boundary points vanishes, converting the problem to an unconstrained optimization problem. This barrier, together with continuous collision detection within minimization steps, ensure that trajectories remain in $\mathscr{A}_I$.

This algorithm then guarantees that the trajectories are modified, compared to the exact solution, in an arbitrarily small, user-controlled (by ϵ) region near object boundaries and, at the same time, always remain admissible.

4.2.1 Trajectory accuracies

A discrete contacting trajectory is accurate if it satisfies 1) admissibility, 2) discrete momentum balance, 3) positivity, 4) injectivity and 5) complementarity.

In the discrete setting, *momentum balance* requires that the gradient of the incremental potential, $\nabla_x E(x)$, balance against the time-integrated contact forces. Its accuracy, is then exactly measured by the residual error in the optimization of the constrained incremental potential. We simply and directly control accuracy of momentum balance by setting stopping tolerance in our nonlinear optimization; see Section 4.4.5.

In turn *positivity* means that contact forces' signed magnitudes, λ_k, per contact pair $k \in \mathscr{C}$, are always non-negative and so push surfaces but do not pull. Our method guarantees exact positivity.

Combined with *admissibility*, *injectivity* requires positive volumes for all tetrahedra in the simulation mesh. This invariant is enforced when a non-inverting energy density

function, e.g. neo-Hookean, is modeled [1].

Finally, the classic definition of *complementarity* in contact mechanics Kikuchi and Oden [1988] is the requirement that contact forces enforcing admissibility can only be exerted between surfaces if they are touching with no distance between them. We do not allow $d_k(x) = 0$, and so define a comparable measure of discrete ϵ-complementarity requiring

$$\lambda_k \max(0, d_k(x) - \epsilon) = 0, \ \forall k \in \mathscr{C} \tag{4.3}$$

to measure how well contact accuracy is achieved. Discrete complementarity is then satisfied whenever distances between all contact pairs, defined as surface pairs with nonzero contact forces, are less than the ϵ and converge to complementarity as we reduce ϵ.

4.3 Related Work

Computational contact modeling is a fundamental and long studied task in mechanics well covered from diverse perspectives in engineering, robotics and computer graphics Brogliato [1999], Kikuchi and Oden [1988], Stewart [2001], Wriggers [1995]. At its core the contact problem combines enforcement of significant and challenging *geometric* non-intersection constraints with the resolution of a deformable solid's momentum balance. The latter task is well-explored, often independent of contact Belytschko et al. [2000], Stuart and Humphries [1996]. We focus below on related works in defining contact constraints, implicitly time stepping with contact and friction, and barriers.

4.3.1 Constraints and constraint proxies

Contact simulation requires a computable model of admissibility and so a choice of contact constraint representation. For volumetric models, admissibility generally begins with description of a signed distance function. This allows a clean formulation of the continuous

[1] When an invertible deformation model, e.g. fixed corotational, is modeled, injectivity need not be preserved in computation. We primarily focus on non-inverting neo-Hookean but will also demonstrate the weaker invertible case with fixed corotational.

model. However, when it comes to computing non-intersection on deformable meshes, choices for representing non-intersection must be made and a diversity of constraint representations exist. Contact constraints for deformable meshes, in both engineering Belytschko et al. [2000], Wriggers [1995] and graphics Bridson et al. [2002], Daviet et al. [2011], Harmon et al. [2008, 2009], Otaduy et al. [2009], Verschoor and Jalba [2019] are most commonly defined pairwise between matched surface primitives.

Existing methods most often define a local, signed distance evaluation using a diverse array of nonlinear proxy functions as well as their linearizations. These include linear gap functions, linearized constraints built from geometric normals, as well as a number of oriented volume constraints Kane et al. [1999], Sifakis et al. [2008]. These nonlinear proxies, such as the tetrahedral volumes formed between surface point-face and edge-edge pairs, are only locally valid. They can introduce artificial ghost contact forces when sheared, false positives when rotated (e.g. for edge-edge tetrahedra), discontinuities when traversing surface element boundaries, and, in many methods, must still be further linearized and so introduce additional levels of approximation in order to solve a constrained time step.

Alternately gap functions and other related methods approximate signed distance functions for pairs of primitives by locally projecting a linearized distance measure between pairwise surface primitives onto a fixed geometric normal Otaduy et al. [2009], Wriggers [1995]. As discussed in Erleben Erleben [2018] these "contact point and normal" based constraint functions can be inconsistent over successive iterations and so are highly sensitive to surface and meshing variations with well known failure modes if care is not taken. Indeed, as we investigate in Section 4.8.3, even with iterative updates of these linear constraints inside SQP-type methods, time stepping with gap functions and related representations produces highly varied results whose success or failure is largely dependent on the scene simulated. In turn all of these challenges are only further exacerbated when simulations encounter the sharp, nonsmooth and even codimensional collisions imposed by meshed obstacles Kane et al. [1999]; see e.g. Figure 4.1.

Recent fictitious domain methods Jiang et al. [2017b], Misztal and Bærentzen [2012],

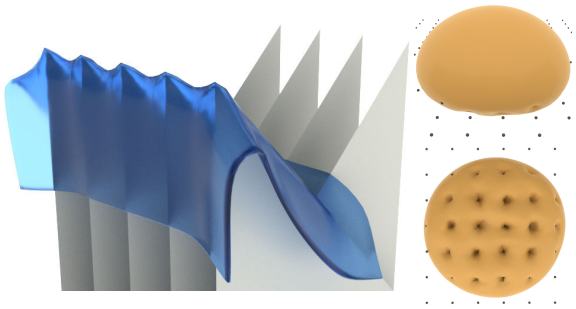


Figure 4.1: **Nonsmooth, codimensional collisions.** Left: thin volumetric mat falls on codimensional (triangle) obstacles. Right: a soft ball falls on a matrix of point obstacles, front and bottom views.

Müller et al. [2015], Pagano and Alart [2008], Zhang et al. [2005] offer a promising alternative. In these methods, motivated by global injectivity conditions Ball [1981] negative space is separately discretized by a compatible discretization sometimes called an air-mesh Müller et al. [2015]. Maintaining a non-negative volume on elements of this mesh then guarantees non-inversion. However, as with locally defined proxy volumes, the globally defined mesh introduces increasingly severe errors, e.g., shearing and locking forces, as it distorts with the material mesh. In 2D this can be alleviated by local Müller et al. [2015] or global Jiang et al. [2017b] remeshing, however this is highly inefficient in 3D, does not provide a continuous constraint representation for optimization, nor, even with remeshing, can it resolve sliding and resting contact where air elements must necessarily be degenerate Li et al. [2018b].

Alternately, discrete signed distance fields (SDF) representations can be constructed via a number of approximation strategies over spatial meshes Jones et al. [2006]. However, while state of the art adaptive SDF methods now gain high-resolution accuracy for sampling against a fixed meshes Koschier et al. [2017], they can not yet be practically updated at rates suitable for deformable time steps, much less at rates suitable for querying deformations at every iterate within a single implicit time step solve Koschier et al. [2017]. At the same time, discontinuities across element boundaries, while improved in recent works, still preclude smooth optimization methods.

We observe that while approximating signed distance pairwise between surface mesh elements is problematic, unsigned distance is well defined. We then design a new contact model for exact admissibility constraints in terms of unsigned distances between mesh-element pairs. This model of constraint is constructed sufficiently smooth to enable efficient, super-linear Newton-type optimization, maintains exact constraint satisfaction guarantees throughout all steps (time steps and iterations) and requires evaluation of just mesh-surface primitive pairs.

4.3.2 Implicit Time Step Algorithms for Contact

With choice of contact constraint proxy, $g(x) \geq 0$, the solve for the implicit time-step update is then the minimization of the contact-constrained IP Doyen et al. [2011], Kane et al. [1999], Kaufman and Pai [2012],

$$\min_x E(x, x^t, v^t) \quad \text{s.t.} \quad g(x) \geq 0. \tag{4.4}$$

The variational problem (4.4), or its approximation is then minimized to compute the configuration at each time step.

This is typically done with off-the-shelf Nocedal and Wright [2006] or customized constrained optimization techniques. In engineering, commonly used methods include sequential quadratic programming (SQP) Kane et al. [1999], augmented Lagrangian and occasionally interior point methods Belytschko et al. [2000]. All such methods iteratively linearize constraint functions and elasticity. However, both nonlinear constraint functions and their linearizations are generally valid only in local regions of evaluation and so can lead to intersections due to errors at larger time steps, faster speeds and/or larger deformations. For example Kane et al.'s [1999] volumes are only valid under a strong assumption of the relative position of contact primitive pairs.

In turn linearization of the full constraint set can also introduce additional error, result in infeasible sub-problems, locking and/or constraint drift Erleben [2018]. This often requires complex and challenging (re-)evaluations of constraints in inter-penetrating states. Even

when such obstructions are not present, iterated constraint linearization generally can not guarantee interpenetration-free state except upon convergence and so often must resort to small time steps and non-physical fail-safes in order to limit damage caused by missed constraint enforcement.

Although SQP- Kane et al. [1999] and LCP/QP-based contact solvers Kaufman et al. [2008] support and generally employ a variety of constraint-set culling and active-set update strategies, e.g., incrementally adding newly detected collisions at each iteration Otaduy et al. [2009], Verschoor and Jalba [2019], they also can become infeasible and generate constraint drift when linearizing and filtering constraints.

Irrespective of how the contact-IP is solved and constraints are enforced, we then remain faced with combinatorial explosion in the number of contact constraints to handle. Determining the active set, i.e., finding which constraints are necessary for admissibility and so can not be ignored, remains an outstanding computational challenges. At the same time, to take large time steps or handle large deformation, we must resolve strongly nonlinear deformation energies in balance with contact forces. This requires line search. However, for constrained optimization methods, e.g., SQP, efficient line search in the presence of large numbers of active constraints remains an open problem Bertsekas [2016], Nocedal and Wright [2006]. For this reason, existing methods in graphics currently avoid line search altogether and are, as a consequence, mostly restricted to quadratic energy models per time step Otaduy et al. [2009], Verschoor and Jalba [2019] and, often, small time step sizes for even moderate material stiffness Verschoor and Jalba [2019].

4.3.3 Friction

The addition of accurate friction with stiction only increases the computational challenge for time stepping deformation Wriggers [1995]. Friction forces are tightly coupled to the computation of both deformation and the contact forces that prevent intersection. These side conditions are generally formulated by their own governing variational Maximal Dissipation Principle (MDP) Goyal et al. [1991], Moreau [1973] and thus introduce additional nonlinear,

nonsmooth and asymmetric relationships to dynamics. In transitions between sticking and sliding modes large, nonsmooth jumps in both magnitude and direction are made possible by frictional contact model. Asymmetry, in turn, is a direct consequence of MDP: frictional forces are not uniquely defined by the velocities they oppose, and are also determined by additional consistency conditions and constraints, e.g., Coulomb's law. One critical consequence is that there is no well-defined potential that we can add to an IP to directly produce contact friction via minimization.

To address these challenges, frictional contact is often solved by seeking a joint solution to the optimality conditions of MDP together with the discretized equations of motion (the latter are equivalent to optimality conditions for E). This requires, however, simultaneously solving for primal velocity unknowns together with a large additional number of dual contact and friction force unknowns. These latter variables then scale in the number of active contacts and their number grows large for even moderately sized simulation meshes.

To solve these systems it is generally standard to apply iterative per-contact, nonlinear Gauss-Seidel-type methods Alart and Curnier [1991], Bridson et al. [2002], Daviet et al. [2011], Jean and Moreau [1992], Kaufman et al. [2014]. Here elasticity is again often, but not always, linearized per time step, while contact and friction constraints are similarly often approximated per iteration with a range of linear and nonlinear proxies. Alternate iteration strategies Kaufman et al. [2008], Otaduy et al. [2009] have also been applied. However, as in the frictionless setting, all such splittings remain challenging to solve with guarantees for complex, real-world scenarios. Most recently, the same discrete formulation has been solved with new custom-designed algorithms – both with nonsmooth Newton-type strategies Bertails-Descoubes et al. [2011], Macklin et al. [2019] and an extension of the Conjugate Residual method Verschoor and Jalba [2019] with improved accuracy and efficiency.

4.3.4 Barrier Functions

Barrier functions are commonly applied in nonlinear optimization, especially in interior-point methods Nocedal and Wright [2006]. Here primal-dual interior point methods are generally favored with Lagrange multipliers as additional unknowns for improved convergence. For contact problems, this impractically enlarges system sizes by orders-of-magnitude. Here we focus on a primal solution suited for contact problems. Similarly, the vast majority of the literature focuses on globally supported functions, which are not viable for contact due to the quadratic set (collision primitive pairs) of constraints that must be considered. Recently, a few works have begun exploiting *locally supported* barriers Harmon et al. [2009], Schüller et al. [2013], Smith and Schaefer [2015]. Harmon et al. [2009] propose a set of layered discrete penalty barriers that grow unbounded as the configuration reaches toward contact. While well-suited for small time-step explicit methods, the incremental construction of the barriers challenge application in implicit time integration with Newton-type optimization. Most recently methods in geometry processing Schüller et al. [2013], Smith and Schaefer [2015] propose locally supported barriers in the context of 2D mesh parametrization to prevent element inversion and overlap. Our formulation builds on a similar idea. Here we design smoothed, local barriers custom-constructed for the challenges of resolving contact-response and preventing intersection between 3D mesh-primitives.

4.3.5 Summary

In summary, state of the art methods for contact simulation are often highly effective per example. However, in order to do so they generally require significant hand-tuning per simulation set-up in order to obtain successful simulation output, i.e., stable, nonintersecting, plausible, or predictive output. Currently many of the tuned parameters, as we discuss in Section 4.8.3, are not physical but rather guided by expected intersection constraint violation errors and stability needs, and so need to be experimentally determined by

many simulation test runs. Thus, to date, direct, fully automated simulation has not been available for contact simulation – despite contact's fundamental role in many design, engineering, robotics, learning and animation tasks. Towards a direct, "plug-and-play" contact simulation framework we propose IPC . Across a wide range of mesh resolutions, time step sizes, physical conditions, material parameters and extreme deformations we confirm IPC performs and completes simulations to requested accuracy without algorithm parameter tuning.

4.4 Primal Barrier Contact Mechanics

In this section, we describe how we solve our time step problem (4.2) formulated in Section 4.2. We defer consideration of friction to Section 4.5, focusing on handling contact dynamics here. We solve the minimization problem (4.2), with primitive-pair admissibility constraints using a carefully designed barrier-augmented incremental potential that can be evaluated efficiently. In turn, to solve this potential we design a custom, contact-aware, Newton-type solver, outlined in Algorithm 3, with constraint culling for efficient evaluation of objectives, gradients and Hessians (Section 4.4.3).

4.4.1 Barrier-Augmented Incremental Potential

To enforce distance constraints $d_k(t) > 0$, for all $k \in \mathscr{C}$, we construct a continuous barrier energy b (Section 4.4.2), that creates a highly localized repulsion force, affecting motion only when primitives are close to collision, and vanishing if primitives are a small user-specified distance apart. We then augment the time step potential $E(x, x^t, v^t)$ with a sum of these barriers over all possible pairs in $\mathscr{C}$. The barrier-augmented IP is then

$$B_t(x) = E(x, x^t, v^t) + \kappa \sum_{k \in \mathscr{C}} b\big(d_k(x)\big), \tag{4.5}$$

with $\kappa > 0$ an adaptive conditioning parameter automatically controlling the barrier stiffness (see Section 4.4.3 and our appendix for details.).

Algorithm 3 Barrier Aware Projected Newton

procedure BARRIERAWAREPROJECTEDNEWTON($x^t, \boldsymbol{\epsilon}$)
  $x \leftarrow x^t$
  $\hat{\mathscr{C}} \leftarrow$ComputeConstraintSet$(x, \hat{d})$ ▷ Section 4.4.6, 4.6.1
  $E_{\text{prev}} \leftarrow B_t(x, \hat{d}, \hat{\mathscr{C}})$
  $x_{\text{prev}} \leftarrow x$
  do
    $H \leftarrow \text{SPDProject}(\nabla_x^2 B_t(x, \hat{d}, \hat{\mathscr{C}}))$ ▷ Section 4.4.3
    $p \leftarrow -H^{-1} \nabla_x B_t(x, \hat{d}, \hat{\mathscr{C}})$
    // CCD line search: ▷ Section 4.4.4
    $\alpha \leftarrow \min(1, \text{StepSizeUpperBound}(x, p, \hat{\mathscr{C}}))$
    do
      $x \leftarrow x_{\text{prev}} + \alpha p$
      $\hat{\mathscr{C}} \leftarrow$ComputeConstraintSet$(x, \hat{d})$
      $\alpha \leftarrow \alpha/2$
    while $B_t(x, \hat{d}, \hat{\mathscr{C}}) > E_{\text{prev}}$
    $E_{\text{prev}} \leftarrow B_t(x, \hat{d}, \hat{\mathscr{C}})$
    $x_{\text{prev}} \leftarrow x$
    Update κ, BCs and equality constraints ▷ appendix
  while $\frac{1}{h}\|p\|_\infty > \boldsymbol{\epsilon}_d$
  return x

Minimizing (4.5) enables the solution of contact-constrained dynamics with unconstrained optimization. Computing the energy naively, however, would require evaluation of the barrier functions for all $O(|\mathscr{T}|^2)$ pairs. To address similar challenges many simulation methods simply remove constraints corresponding to distant primitives that are hoped to be unnecessary for the current solution. However, this tempting operation is dangerous, as significant errors and instabilities can be introduced when constraint sets are modified and critical collisions can also be missed (see Sections 4.3 and 4.8). Instead, we design smooth barrier functions that allow us to compute the barrier energy exactly and efficiently for all constraints while evaluating distances only for a small subset of pairs of primitives that are close and simultaneously ensuring that the rest smoothly evaluate to zero.

4.4.2 Smoothly Clamped Barriers

We begin by defining a smooth barrier function composed of terms that are *local* for every primitive pair, that is each term is exactly zero if the two primitives are far away, enabling reliable and efficient pruning of pairs in $\mathscr{C}$ without change to the solution.

We start by defining a computational distance accuracy target, $\hat{d} > 0$ (corresponding to ϵ in Section 4.2) that specifies the maximum distance at which contact repulsions can act. We then construct a barrier potential that approaches infinity at zero distance, initiates contact forces for pairs closer than the target distance, $\hat{d}$, and applies no repulsion at distances greater than $\hat{d}$.

Considering the smooth log-barrier function commonly applied in optimization Boyd and Vandenberghe [2004] gives $\ln(d/\hat{d})$, where d is the unsigned distance evaluation between a primitive pair. However, simply truncating this function produces an unacceptably non-smooth energy which cannot be efficiently optimized and is effectively no better than simply discarding constraints. Some examples of problems this generates in optimization are covered in the appendix. We thus propose a smoothly clamped barrier to regain

superlinear convergence for Newton-type methods

$$b(d,\hat{d}) = \begin{cases} -(d-\hat{d})^2 \ln\left(\frac{d}{\hat{d}}\right), & 0 < d < \hat{d} \\ 0 & d \geq \hat{d}. \end{cases} \tag{4.6}$$

Our barrier function is now C^2 at the clamping threshold, and it is exactly zero for pairs beyond the target accuracy (see Figure 4.2). Now, without harm, at any configuration x, we only need to evaluate barrier terms for the *culled constraint set*

$$\hat{C}(x) = \{k \in C \ : \ d_k(x) \leq \hat{d}\},$$

composed of barriers between close primitives. As we increase accuracy by specifying smaller $\hat{d}$ we then need to evaluate increasingly smaller numbers of contact barriers, albeit with increased cost in nonlinearity.

Next, while the barrier function $b(d,\hat{d})$ itself is now C^2, the distance function it evaluates between primitives will be C^0 for certain unavoidable configurations; i.e., parallel edge-edge collisions – see Figure 4.8. For this reason, we multiply the barrier terms for edge-edge collisions by a mollifier that ensure our distance function is C^1 (and piecewise C^2) for all primitive pair types. Distance evaluation and mollifier are discussed in detail in Section 4.6. Additional important considerations related to numerical stability and roundoff error in distance evaluation are then detailed further in the appendix.

4.4.3 Newton-Type Barrier Solver

Projected Newton (PN) methods are second-order unconstrained optimization strategies for minimizing nonlinear nonconvex functions where the Hessian may be indefinite. Here we apply and customize PN to the barrier-augmented IP (4.5). At each iteration, we project each local energy stencil's Hessian to the cone of symmetric positive semi-definite (PSD) matrices (see SPDProject function in Algorithm 3) prior to assembly. Specifically, following Teran et al. [2005] we project per-element elasticity Hessians to PSD. We then comparably project the Hessian of each barrier to PSD. Each barrier Hessian has the form

$$\frac{\partial^2 b}{\partial d^2}\nabla_x d(\nabla_x d)^T + \frac{\partial b}{\partial d}\nabla_x^2 d \tag{4.7}$$

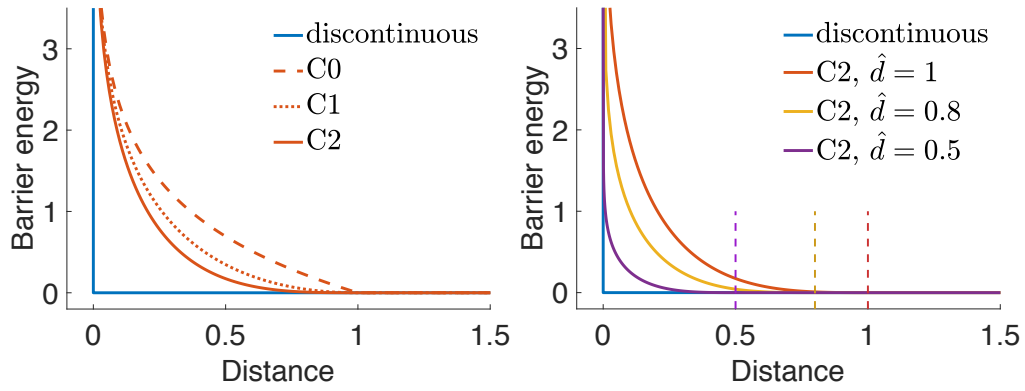


Figure 4.2: **Barriers.** Left: log barrier function clamped with varying continuity. We can augment the barrier to make clamping arbitrarily smooth (see our appendix). We apply our C^2 variant for best tradeoff: smoother clamping improves approximation of the discontinuous function while higher-order continuity introduces more computational work. Right: our C^2 clamped barrier improves approximation to the discontinuous function as we make our geometric accuracy threshold, $\hat{d}$, smaller.

and so can be constructed as a small matrix restricted to the vertices in the stencil of the barrier's primitive pair. The addition of mass matrix terms then ensures that the assembled total IP Hessian is symmetric positive definite (SPD). Originally we also investigated a Gauss-Newton approximation to the above barrier Hessian, taking only the first, SPD term in the sum. However, we find that resulting search directions are far less efficient than using the full projected barrier Hessian.

Termination For termination of the solver we check the infinity norm of the Newton search direction scaled by time step (but *unscaled* by line-search step size). Specifically we solve each time step's barrier IP to an accuracy satisfying $\frac{1}{h}\|H^{-1}\nabla B_t(x)\|_\infty < \boldsymbol{\epsilon}_d$. This provides affine invariance and a characteristic measure using the Hessian's natural scaling as metric. Accuracy is then directly defined by $\boldsymbol{\epsilon}_d$ in physical units of velocity (and so is independent of time-step size applied) and consistently measures quadratically approximated distance to local optima across examples with varying scales and conditions.

Barrier stiffness adaptation We automatically adapt our barrier stiffness to provide repulsive scaling that balances necessary distances against conditioning from the barrier stiffness. Our barrier-augmented potential, B_t, has two key parameters: $\hat{d}$ and κ, that jointly scale the effective stiffness of each contact barrier. The strength of our barriers' contact forces (equivalently Lagrange multipliers) are directly determined during minimization by evaluating distances, d_k, and stiffness, κ. When κ is too small, contact-pair distances must become tiny to exert sufficient repulsion. On the other hand, when κ is too large, contact-pair distances must be below $\hat{d}$ in order to exert non-zero force, but at the same time remain exceedingly close to $\hat{d}$ so as to not exert too large a repulsion. Both cases thus generate unnecessary ill-conditioning and nonsmoothness that challenge efficiency. As we directly control geometric accuracy by setting $\hat{d}$, this frees κ to adaptively condition our Newton-solver to improve convergence. While conceptually one could imagine finding improved scalings of κ by hand, per example, this is unacceptable and inefficient for an automated simulation pipeline. Instead, in our appendix, we derive our stiffness update algorithm that automatically adapts barrier stiffness per iterate for improved conditioning.

Relation to homotopy solves While in IPC we directly set and solve for a desired target accuracy $\hat{d}$, a natural alternative is to solve with a homotopy as is typical in interior point methods. We initially experimented with this approach: solving for larger distances (and so less stiff systems) and then decreasing to the target distance, $\hat{d}$, over successive nonlinear solves. We find, however, that this is unnecessary for elastodynamics where the direct barrier solves we employ are much more efficient. In part, this is because we typically have a good warm start available from the prior time step.

4.4.4 Intersection-Aware Line Search

While our barrier energy is infinite for contact, this by itself does not guarantee that constraints $d_k(t) > 0$ are not violated by the solver. Standard line search Nocedal and Wright [2006], e.g, back-tracking with Wolfe conditions, can find an energy decrease

in configurations that have passed through intersection, resulting in a step that takes the configuration out of the admissible set.

Smith and Schaefer's [2015] line-search filter computes the largest step size in 2D per triangle and per boundary point-edge pair that first triggers inversion or overlap, and then take the minimum as a step size upper bound for the current Newton iteration to stay feasible. Taking inspiration from this line-search filter we propose a continuous, intersection-aware line search filter for 3D meshes. In each line search we first apply CCD to conservatively compute a large feasible step size along the descent step. We then apply back-tracking line search from this step size upper bound to obtain energy decrease. CCD then certifies that each step taken is always valid. When we apply barrier-based energy densities (our default) for our elasticity potential, Ψ, i.e., neo-Hookean, we combine the inversion-aware line search filter Smith and Schaefer [2015] with our intersection-aware filter to obtain descent steps. In combination this guarantees that every step of position update in our solver (and so simulation) maintains an inversion- and intersection-free trajectory.

4.4.5 IPC Solution Accuracy

Revisiting accuracy we confirm *momentum balance* is directly satisfied by IPC after convergence. For example, for implicit Euler we have

$$\nabla_x B_t(x, \hat{d}) = 0 \implies M(\frac{x - \hat{x}}{h^2}) = -\nabla\Psi(x) + \sum_{k \in \mathscr{C}} \lambda_k \nabla d_k(x), \tag{4.8}$$

where our contact forces λ_k are given by barrier derivatives

$$\lambda_k = -\frac{\kappa}{h^2}\frac{\partial b}{\partial d_k}. \tag{4.9}$$

Comparable discrete momentum balance follows when we apply alternate time integration methods, e.g. implicit Newmark. *Positivity* is then confirmed directly by (4.9) and observing that our barrier function definition guarantees $\frac{\partial b}{\partial d_k} \leq 0$. In turn our above line-search filters guarantee *admissibility* and, when applicable for barrier-type elasticity energy densities,

injectivity. Finally, our barrier definition ensures *discrete complimentarity* is always satisfied as contact forces can not be applied at distance more than $\epsilon = \hat{d}$ away.

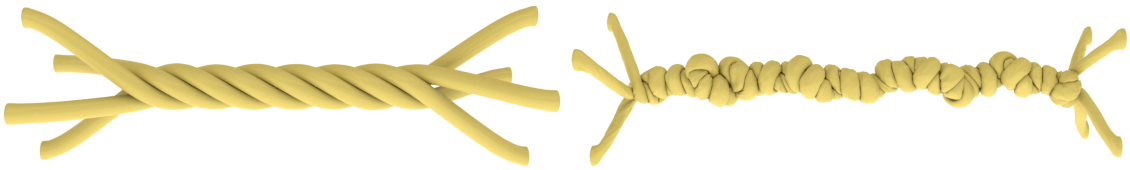


Figure 4.3: **Extreme stress test: rod twist for 100s.** We simulate the twisting of a bundle of thin volumetric rod models at both ends for 100s. IPC efficiently captures the increasingly conforming contact and expected buckling while maintaining an intersection- and inversion-free simulation throughout. Top: at 5.5s, before buckling. Bottom: at 73.6s, after significant repeated buckling is resolved.

4.4.6 Constraint Set Update and CCD Acceleration

Every line search, prior to backtracking, performs CCD to guarantee non-intersection, while every evaluation of energies and their derivatives compute distances to update the culled constraint set, $\hat{\mathscr{C}}(x)$. To accelerate these computations, we construct a combined spatial hash and distance filtering structure to efficiently reduce the number of primitive-pair distance checks. Then, to further accelerate intersection-free stepping along each Newton iterate's descent direction, p, we derive an efficient conservative bound motivated by CFL conditions Courant et al. [1967]. As in force evaluations we aim to avoid unnecessary and expensive CCD computation on primitive pairs *not in* $\hat{\mathscr{C}}$. We leverage the fact that all contact pairs *not* in $\hat{\mathscr{C}}$ are at distances greater than $\hat{d}$, and use the maximal relative search step in p of each such pair to obtain a conservative upper bound on step size. We then need only perform the CCD tests on primitive pairs in $\hat{\mathscr{C}}$. This CCD culling generally provides an average 50% speed-up for all CCD costs across our simulations, with negligible increase in Newton iterations and an overall impact of 10% improvement in simulation times. Details on these accelerations and our adaptive application of this bound (to avoid taking overly conservative steps) are detailed in our appendix.

4.5 Variational Friction Forces

Frictional contact adds contact-dependent dissipative forcing to our system. At macroscale these friction forces are modeled by the Maximal Dissipation Principle (MDP) Moreau [1973]. MDP posits that frictional forces maximize rate of dissipation in relative motion directions orthogonal to contact constraints up to a maximum magnitude imposed by limit surfaces, e.g. as modeled by Coulomb's constraint Goyal et al. [1991].

4.5.1 Discrete Friction

To include frictional contact in our time stepping, we add local friction forces F_k for every active surface primitive pair, $k \in \hat{C}(x)$. For each such pair k, at current state x, we construct a consistently oriented sliding basis $T_k(x) \in \mathbb{R}^{3n\times 2}$. Each T_k is built so that $u_k = T_k(x)^T(x - x^t) \in \mathbb{R}^2$ gives the local relative sliding displacement at contact k, in the frame orthogonal to the distance vector between closest points on the two primitives defining $d_k(x)$. See Section 4.6 and our appendix for details on construction of $T_k(x)$. The corresponding sliding velocity is then $v_k = u_k/h \in \mathbb{R}^2$.

Maximizing dissipation rate subject to the Coulomb constraint defines friction forces variationally

$$F_k(x,\lambda) = T_k(x) \operatorname*{argmin}_{\beta\in\mathbb{R}^2} \beta^T v_k \quad \text{s.t.} \quad \|\beta\| \le \mu\lambda_k \tag{4.10}$$

where λ_k is the contact force magnitude and μ is the local friction coefficient.

Friction forces governed by (4.10) are bimodal. If $\|v_k\| > 0$, there is sliding and the corresponding friction force opposes it with $F_k = -\mu\lambda_k T_k(x)\frac{u_k}{\|u_k\|}$. If $\|v_k\| = 0$, there is sticking and the corresponding static friction force is $F_k = -\mu\lambda_k T_k(x) f$, where the friction direction f can take any value in the closed 2D unit disk.

4.5.2 Challenges to Computation

Friction forces F_k are then challenging to solve for in three interconnected ways. First, F_k is *nonsmooth*. In transitions between sticking and sliding modes, nonsmooth jumps in both magnitude and direction are possible. Second, because of sticking modes, F_k in MDP is not uniquely defined by displacements until we have found a solution satisfying stationarity:

$$\nabla B_t(x) - h^2 \sum_{k \in \mathscr{C}} F_k(x, \lambda) = 0. \tag{4.11}$$

Third, there is no well defined dissipation potential whose spatial gradient will generate friction forces. As a consequence, frictional contact forces do not naturally fit into variational time-stepping frameworks.

To tackle these challenges, we first examine F_k as a nonsmooth function of u_k. Next, as in our barrier treatment of contact, we smooth the friction function with controlled and bounded accuracy. Then, in order to apply friction as an energy potential in our variational solve, we lag updates of the sliding bases T_k and contact forces λ_k over nonlinear solves within each time step (or over time steps). This allows us to define a smooth dissipative potential for friction that can be consistently integrated into our Newton-type solver.

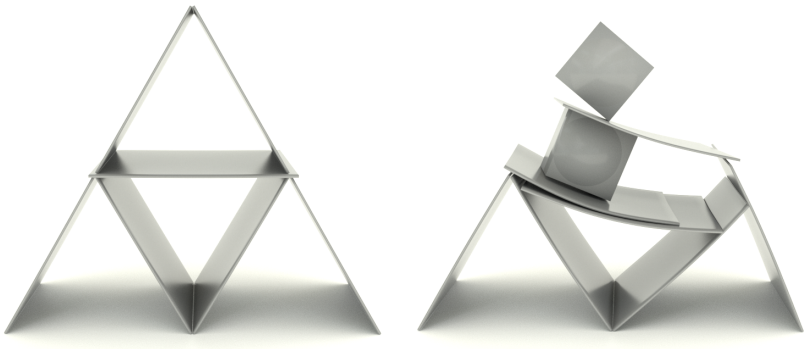


Figure 4.4: **Friction benchmark: Stiff card house.** Left: we simulate a frictionally stable "card" house with $0.5m \times 0.5m \times 4mm$ stiff boards ($E = 0.1GPa$). Right: we impact the house at high-speed from above with two blocks; elasticity is now highlighted as the thin boards rapidly bend and rebound.

4.5.3 Smoothed Static Friction

During each of our Newton iterations any transitions of sliding displacements to or from sticking conditions will introduce large and sudden jumps in friction forces, F_k. These discontinuities, if left unmollified, would severely slow and even break convergence of gradient-based optimization; see Section 4.7. To enable efficient and stable optimization, we smooth the friction-velocity relation in the transition to static friction.

We start with a useful and equivalent (re-)expression for friction forces:

$$F_k = -\mu\lambda_k T_k(x) f(\|u_k\|)\, s(u_k), \tag{4.12}$$

with $s(u_k) = \frac{u_k}{\|u_k\|}$ when $\|u_k\| > 0$, while $s(u_k)$ takes any 2D unit vector when $\|u_k\| = 0$. The friction magnitude function, f, is then correspondingly nonsmooth with $f(\|u_k\|) = 1$ when $\|u_k\| > 0$, and $f(\|u_k\|) \in [0, 1]$ when $\|u_k\| = 0$.

To smooth f and so (4.12) with bounded approximation error, we first define a velocity magnitude bound ϵ_v (in units of m/s) below which sliding velocities $v_k = u_k/h$ are treated as static. Then, we define a smoothed approximation of f with f_1. We maintain $f_1(y) = 1$ for all $y > h\epsilon_v$, (sliding) while for $y \in [0, h\epsilon_v]$, we require $f_1(y)$ to smoothly and monotonically transition from 1 to 0 over a finite range. This forms a bijective map from velocity magnitudes to friction magnitudes for velocities below the ϵ_v limit. For smoothing we experiment with satisfying interpolating polynomials ranging from C^0 to C^2. Increased continuity order introduces greater smoothing and faster error reduction for decreasing ϵ_v, at the cost of introducing greater nonlinearity into the IP solve. In the end, we find that our C^1 interpolant

$$f_1(y) = \begin{cases} -\frac{y^2}{\epsilon_v^2 h^2} + \frac{2y}{\epsilon_v h}, & y \in (0, h\epsilon_v) \\ 1, & y \geq h\epsilon_v, \end{cases} \tag{4.13}$$

provides best balance – yielding a continuous force Jacobian while introducing less nonlinearity and so fewer overall iterations in testing. See Figure 4.5 and our discussion in the appendix.

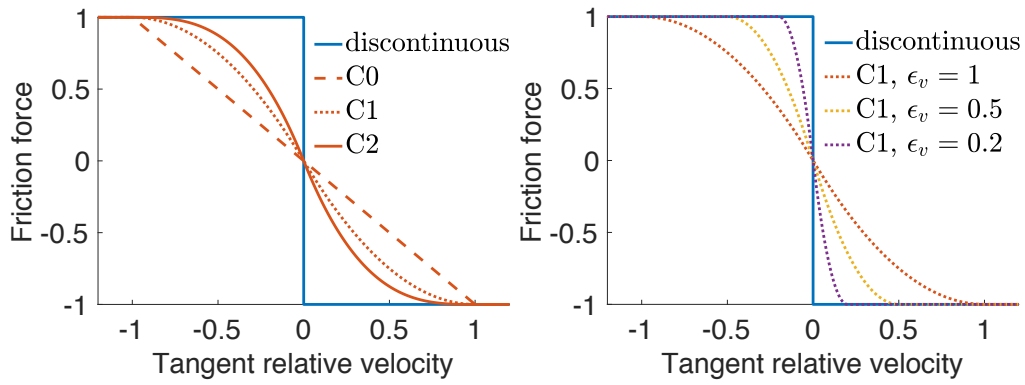


Figure 4.5: **Friction smoothing in 1D.** Left: increasing orders of our polynomials better approximate the friction-velocity relation with increasing smoothness. Right: Our C^1 construction improves approximation to the exact relation as we make our frictional accuracy threshold, ϵ_v, and so the size of static friction zone, smaller.

4.5.4 Variationally Approximated Friction

With a smooth and uniquely defined F_k for each u_k, we are now able to define friction forces solely based on nodal displacement unknowns. A next natural step would then be to define a so-called dissipative potential Kane et al. [2000], Pandolfi et al. [2002] for inclusion in our optimization. An ideal potential would be a scalar function with respect to x whose gradient returns F_k. However, even with our smoothing, no well-defined displacement-based potential for friction exists, and F_k cannot be approximated by a potential force without introducing significant approximation errors. In other words, we do not have a variational form of friction that we can yet minimize.

We start by making dependence of our friction on both $T_k(x)$ and $\lambda_k(x)$ explicit:

$$F_k(x, \lambda_k, T_k) = -\mu \lambda_k T_k f_1(||u_k||) \frac{u_k}{||u_k||}. \tag{4.14}$$

Now, if we set $T_k = T_k(x)$ and $\lambda_k = \lambda_k(x)$ this friction evaluation is exact. However, if we decouple dependence of the evaluated sliding basis and contact force from x and instead lag them to values, λ^n, T^n, from a prior nonlinear solve (or previous time step) n, then all remaining terms in the expression for friction are integrable. The lagged friction force is

then $F_k(x, \lambda_k^n, T_k^n)$ and provides a simple and compact friction potential,

$$D_k(x) = \mu \lambda_k^n f_0(||u_k||). \tag{4.15}$$

Here f_0 is given by the relations $f_0' = f_1$ and $f_0(\epsilon_v h) = \epsilon_v h$ so that $F_k(x) = -\nabla_x D_k(x)$. This potential provides easy-to-compute Hessian, $\nabla_x^2 D_k(x)$, and energy contributions to the barrier potential, described in detail in the appendix. Our full friction potential is then $D(x) = h^2 \sum_{k \in \mathscr{C}} D_k(x)$, and the frictional barrier-IP potential for the time step $t + 1$ is

$$B_t(x, \hat{d}) + D(x). \tag{4.16}$$

Friction Hessian projection For our Newton method (Section 4.4.3), we again need to project the friction potential Hessian to the space of PSD matrices. The friction Hessian structure is similar to that of elasticity, in that it can be written as a product of the T_k matrices. This allows us to apply the same strategy as used for elasticity Hessians, and so we need only perform a 2×2 PSD projection for each friction term per primitive pair. This is detailed in our appendix.

4.5.5 Frictional Contact Accuracy

Accuracy of friction forces generated by each solution of our IP (4.16) are defined by the static threshold, sliding basis and contact force magnitudes.

Static friction threshold As we apply smaller ϵ_v we decrease the range of sliding velocities that we exert static friction upon and correspondingly sharpen the friction forces towards the exact nonsmooth friction relation. Decreasing ϵ_v thus reduces stiction error while increasing compute times as we introduce a sharper nonlinearity in a tighter range; see Figure 4.5. For accurate reproduction of dynamic behaviors with friction and for visually plausible results, we observed that $\epsilon_v = 10^{-3} \ell m/s$, where ℓ is characteristic length (i.e. bounding box size), works well as a default across a wide range of examples with friction coefficients. See e.g., Figure 4.7. As static accuracy becomes important, we then

find solutions with $\epsilon_v = 10^{-5}m/s$ work well. We have further confirmed IPC convergence down to $\epsilon_v = 10^{-9}m/s$. See, for example, our reproduction of the stable frictional contact structures in the masonry arch and card house examples in Figures 4.4 and 4.6.

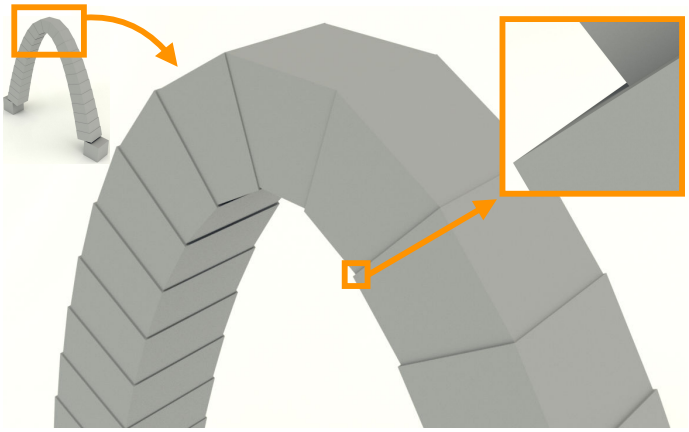


Figure 4.6: **Friction benchmark: Masonry arch.** IPC captures the static stable equilibrium of a 20m high cement ($\rho = 2300kg/m^3$, $E = 20GPa$, $\nu = 0.2$) arch with tight geometric, $\hat{d} = 1\mu m$, and friction, $\boldsymbol{\epsilon}_v = 10^{-5}m/s$ accuracy. Decreasing μ then obtains the expected instability and the stable arch does not form (see the supplemental videos Li et al. [2020b]). Inset: zoomed $100\times$ (orange) highlights the minimal gaps with a geometric accuracy of small $\hat{d}$.

Friction direction and magnitude We improve accuracy of the direction and magnitude of the friction forces by solving successive minimizations of (4.16) within each time step. For each solve we update the lagged T^n and λ^n (warm-starting from the previous time step) with results from the last nonlinear solve. Convergence of lagged iterations is then achieved when we reach approximate momentum balance with

$$\|\nabla B_t(x^{t+1}) - h^2 \sum_{k \in \mathscr{C}} F_k(x^{t+1}, \lambda^{t+1}, T^{t+1})\| \leq \epsilon_d, \tag{4.17}$$

where ϵ_d is the targeted dynamics accuracy.

We confirm lagged iterations rapidly converge over nonlinear solves with our FE models for the well-known, standard frictional benchmarks, e.g., block-slopes, catenary arches and

card houses. See Figures 4.6 and 4.4 and Section 4.7. However, we emphasize that we do not have convergence guarantees for lagging. In particular, we have identified cases with large deformation and/or high speed impacts where we do not reach convergence for T and λ in the friction forces. Thus, in our large-deformation frictional examples we apply just a single lagging iteration. In these cases, sliding directions and contact force magnitudes in the friction force evaluation may not match. However, even in these cases, all other guarantees, including non-intersection, momentum balance (as in the frictionless case) and accurate stiction are maintained. More generally, we observe high-quality, predictive frictional results for large deformation examples independent of the number of lagging iterations applied; see e.g. Figure 4.15. We also emphasize that for frictionless models, IPC continues to guarantee convergence for contact and elasticity with just a single nonlinear solve per time step.

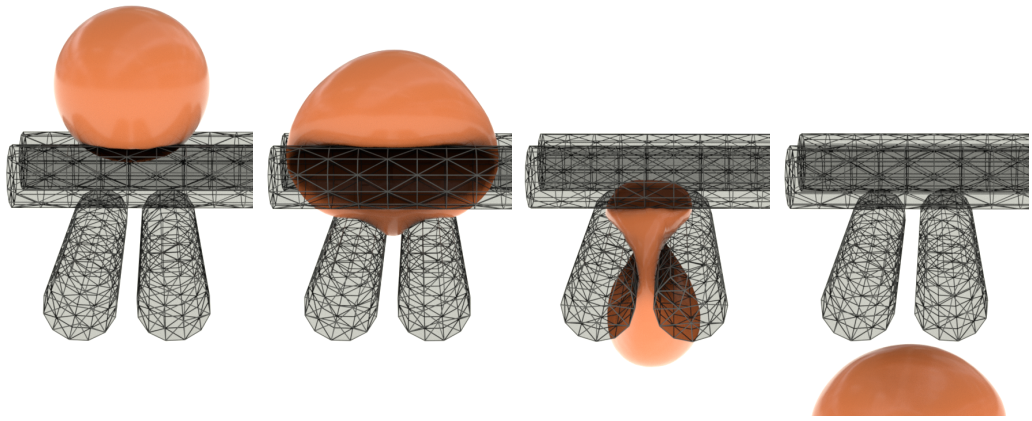


Figure 4.7: **Large deformation, frictional contact test.** We drop a soft ball ($E = 10^4 Pa$) on a roller (made transparent to highlight friction-driven deformation). Here IPC simulates the ball's pull through the rollers with extreme compression and large friction ($\mu = 0.5$).

4.6 Distance Computation

Evaluating unsigned distance functions between point-triangle and edge-edge pairs requires care as closed-form distance formulas change with relative position of surface primitives.

4.6.1 Combinatorial Distance Computation

Unsigned distances are given by the closest points on the two primitives evaluated.

Distance between a point v_P and a triangle $T = (v_{T1}, v_{T2}, v_{T3})$ can be formulated as a constrained optimization problem,

$$\begin{aligned} \mathscr{D}^{\text{PT}} = & \min_{\beta_1,\beta_2} ||v_P - (v_{T1} + \beta_1(v_{T2} - v_{T1}) + \beta_2(v_{T3} - v_{T1}))|| \\ & s.t. \quad \beta_1 \geq 0, \quad \beta_2 \geq 0, \quad \beta_1 + \beta_2 \leq 1. \end{aligned} \tag{4.18}$$

Similarly the distance between edges v_{11}-v_{12} and v_{21}-v_{22} is

$$\begin{aligned} \mathscr{D}^{\text{EE}} = & \min_{\gamma_1,\gamma_2} ||v_{11} + \gamma_1(v_{12} - v_{11}) - (v_{21} + \gamma_2(v_{22} - v_{21}))|| \\ & s.t. \quad 0 \leq \gamma_1, \gamma_2 \leq 1. \end{aligned} \tag{4.19}$$

Each possible *active set* of these two minimizations corresponds to a closed-form distance formula. In each, at most two constraints can be active at the same time.

- When *two* constraints are active in either (4.18) or (4.19), the distance between primitives is a point-point distance evaluation:

$$d^{PP} = ||v_a - v_b||. \tag{4.20}$$

  Here v_a and v_b correspond to v_P and a v_{Ti} for (4.18), or to the two endpoints of the edges in the edge-edge pair for (4.19).

- When a *single* constraint is active in either (4.18) or (4.19), the distance in both cases becomes a point-edge distance evaluation:

$$d^{PE} = \frac{||(v_a - v_c) \times (v_b - v_c)||}{||v_a - v_b||}. \tag{4.21}$$

  Here (v_a, v_b) corresponds to one of the triangle edges of T and $v_c = v_P$ for (4.18), or else (v_a, v_b) corresponds to one of the edges in the edge-edge pair and v_c corresponds to an endpoint of the other edge for (4.19).

- When *no* constraints are active in either (4.18) or (4.19), distance computations are simply parallel-plane distance evaluations. For the point-triangle pairing in (4.18) this is

$$d^{PT} = |(v_P - v_{T1}) \cdot \frac{(v_{T2} - v_{T1}) \times (v_{T3} - v_{T1})}{||(v_{T2} - v_{T1}) \times (v_{T3} - v_{T1})||}|, \tag{4.22}$$

  while for the edge-edge pairing in (4.19) it is

$$d^{EE} = |(v_{11} - v_{21}) \cdot \frac{(v_{12} - v_{11}) \times (v_{22} - v_{21})}{||(v_{12} - v_{11}) \times (v_{22} - v_{21})||}|. \tag{4.23}$$

For evaluations of d, ∇d, and $\nabla^2 d$, we apply the currently valid, closed-form distance formula (either PP, PE, PT, or EE above) and its analytic derivatives. The formula to apply, at each evaluation of a surface pair, is determined by the active constraint subset defined by the current relative positions of the pair's primitives. This information is computed and stored together with our culled constraint set $\hat{\mathscr{C}}$ data, and so is then available for direct use whenever computing barrier energies and derivatives. This treatment is analogous to storing and reusing singular value decompositions of deformation gradients for elasticity computations. As in elasticity, our distance state and evaluations can efficiently be reused for all energy and derivative evaluations at the same nodal positions. Correspondingly, having now reduced general point-triangle and edge-edge distance evaluations to the above closed-form formulas, we can directly compute and store our sliding bases, $T_k(x)$, for friction computation with respect to each case; please see our appendix for details.

4.6.2 Differentiabilty of d

In collision-resolution methods, close-to-parallel edge-edge contacts are notorious failure modes – to the extent that existing methods often ignore this case by throwing out all corresponding constraints Harmon et al. [2008]. However, despite the challenges imposed, these constraint cases cannot be removed, as doing so would lead to intersection. The reason for the difficulty in these cases is the (lack of) differentiability of the distance function for some configurations. Each above analytic formula for distances corresponds to a subset of the relative configuration space of a primitive pair. For example, for vertex-triangle pairs,

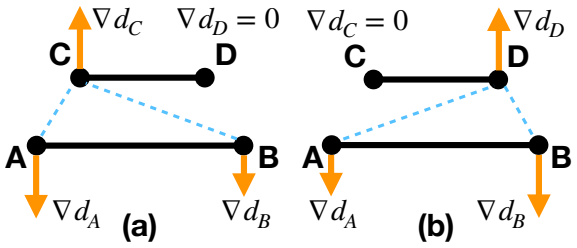


Figure 4.8: **Nonsmoothness of parallel edge-edge distance.** When edge AB and CD are parallel, the distance computation can be reduced to either (a) $C - AB$ point-edge or (b) $D - AB$ point-edge. Then for the trajectory of C moving down from above D, the distance gradient is not continuous at the parallel point even though the distance is always continuously varying.

relative configurations are completely characterized by fixing the triangle and varying v_P positions. If the projection of v_P to T is in the triangle interior, no constraints are active, while if the projection lies on the interior of a triangle edge then one constraint is active. Otherwise, two constraints are active.

Each of these geometric criteria defines a subset of $\mathbf{R}^3$, where one of the three analytic formulas is valid. The distance function is C^∞ inside each such domain, and, in general, is C^1 at the boundaries between domains. However, the critical exception is in parallel edge-edge configurations: at these points, the distance function is not differentiable (see Figure 4.8). Configurations close to these parallel edge-edge conditions, when reached, lead to unacceptably slow convergence of Newton iterations or even convergence failures altogether. Numerically, the issue is similar to the C^0-continuous friction problem we faced in Section 4.5.2. To resolve this issue, we once again apply a local smoothing solution to mollify the barrier corresponding to nearly parallel edge-edge contact conditions.

We smooth by multiplying all edge-edge barrier terms by a piecewise-polynomial mollifier closely analogous to our static-friction smoother; recall Figure 4.5. Here, for each edge-edge contact pair k, we define $e_k(x)$ to vanish when edges $(v_{11}v_{12} - v_{21}v_{22})$ are

parallel and to smoothly grow to 1 as the edge-pair become far from parallel,

$$e_k(x) = \begin{cases} -\frac{1}{\epsilon_\times^2}c^2 + \frac{2}{\epsilon_\times}c & c < \epsilon_\times, \\ 1 & c \geq \epsilon_\times, \end{cases} \tag{4.24}$$

where $c = ||(v_{12} - v_{11}) \times (v_{22} - v_{21})||^2$ and $\epsilon_\times = 10^{-3}||v'_{12} - v'_{11}||^2||v'_{22} - v'_{21}||^2$ is defined with respect to edge-edge vertex-pair rest positions v'.

Our mollified edge-edge barriers are then $e_k(x)b\big(d_k(x)\big)$ and so now extend our barrier potentials to a piecewise C^∞, *everywhere* C^1-continuous (for nonintersecting configurations) barrier formulation. At the same time our barriers now remain sufficient to guarantee that no collisions are missed: there are always point-triangle contact pairs at distance no more than the parallel edge-edge distance; see our appendix for details on this. In turn, our construction of the parallel-edge mollifier then minimizes its impact on edge-edge pair barriers as they move away from degeneracy. While in principle increasing smoothness to C^1 is sufficient to avoid most dramatic degeneracy failures, there are additional numerical stability issues to be addressed related to nearly parallel edges. Please see our appendix for details.

Now, with this third and last smoothing in place we have an overall time-stepping potential for contact and friction that can leverage superlinear convergence and robustness of Newton-type stepping. As we analyze in Sections 4.7 and 4.8 below (see especially 4.7.1) this gains robust simulation against failure – even when simulating challenging conditions with unavoidable numbers of degenerate evaluations.

4.7 Evaluation

Our IPC code is implemented in C++, parallelizing assembly and evaluations with Intel TBB, applying CHOLMOD Chen et al. [2008] with AMD reordering for linear system solves in all examples (with the exception of the squishy ball example – see below) and Eigen Guennebaud et al. [2010] for linear algebra routines. We run most experiments on a 4-core 3.5GHz Intel Core i7, a 4-core 2.9 GHz Intel Core i7, and a 8-core 3.0 GHz Intel

Xeon machine. Machine use per example is summarized along with performance statistics and problem parameters in Figure 4.22 and in our supplemental document Li et al. [2020b]. The reference implementation, scripts used to generate these results and our benchmarks are released as an open-source project.

Linear system computations and solves We compute elasticity and barrier Hessians (with PSD projections) in parallel, and have designed and implemented a custom multi-threaded, sparse matrix data structure construction routine that, given the connectivity graph of nodes, efficiently builds the CSR format with index entries ready. While we utilize efficient symbolic factorization and parallel numerical factorization routines in CHOLMOD Chen et al. [2008] compiled with MKL LAPACK and BLAS, we also tested IPC with AMGCL Demidov [2019] – a multigrid preconditioned solver. Here, we found behavior is as might be expected, less memory overhead and faster linear solves by avoiding direct factorization. However, for majority of examples the large deformations and many contacts generate poorly conditioned systems. We then found AMGCL requires extensive parameter tuning to perform well and still can not compete, in general, with the parallel direct solver. All examples in the following then apply CHOLMOD for linear solves, with the exception of our largest, squishy ball example (Figure 4.21), where we apply AMGCL.

Models and practical considerations We primarily employ the non-inverting, neo-Hookean (NH) elasticity model and implicit Euler time stepping. In the following examples we also apply and evaluate implicit Newmark time stepping, as well as the invertible fixed corotational elasticity (FCR) elasticity model. While for clarity in the preceding we derive IPC with unmodified distance evaluations, for numerical accuracy and efficiency our implementation applies squared distances for evaluations of the barrier, we use $b(d^2, \hat{d}^2)$, and related computations, thus avoiding squared roots. In turn expressions for contact forces, λ_k, and related terms must be modified, from our direct exposition and derivations above. To do so we rescale for consistent dimensions and units in our implementation; see our appendix for details. Finally and importantly we note that IPC's barrier formulation requires

nonzero separation distances to be strictly satisfied at initialization and then guarantees it throughout simulation. Exact initialization at zero distance is neither possible (as the barrier of course diverges) nor for that matter physically meaningful. Contact, including resting contact, instead occurs around the specified geometric distance accuracy given by the user. Here we demonstrate simulated configurations with distances down to $10^{-8}m$ reached in simulation (e.g. arch in Figure 4.6) or initialized by users.

Evaluation and tests Below we first introduce a set of unit tests for seemingly simple yet challenging scenarios with nonsmooth, aligned and close contacts (Section 4.7.1), stress tests involving large deformation and high velocities (Section 4.7.2), and friction (Section 4.7.3). We next study IPC 's scaling, run time, and accuracy behavior as we vary simulation problem parameters (Section 4.7.4). Finally, we present an extensive, quantitative comparison with previous works in Section 4.8.

4.7.1 Unit tests

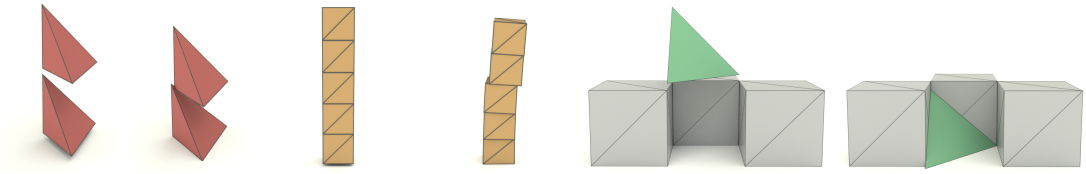


Figure 4.9: **Aligned, close and nonsmooth contact tests.** Pairs of before and after frames of deformable geometries initialized with exact alignment of corners and/or asperities; dropped under gravity. We confirm both nonsmooth and conforming collisions are accurately and stably resolved.

Aligned, close and nonsmooth contact We apply a set of unit tests exercising closely aligned, conforming and nonsmooth contact known to stress contact algorithms. We build them with two simple models: a single tetrahedron and an 8-node unit cube; see Figure 4.9 and Figures 4.10. For contact handling, these seemingly simple tests are designed to

trigger degenerate edge cases that often cause failure in existing methods (see Section 4.8). IPC resolves all cases including those in which we exercise *exact* parallel edge-edge (e.g., Figure 4.9 middle) and point-point (e.g., Figure 4.9, left) collisions. For unit tests like Figure 4.9 right we drop objects into slotted obstacles so that they fit tightly with tiny gaps; here IPC retrieves a tight conforming fit into a 1μm gap.

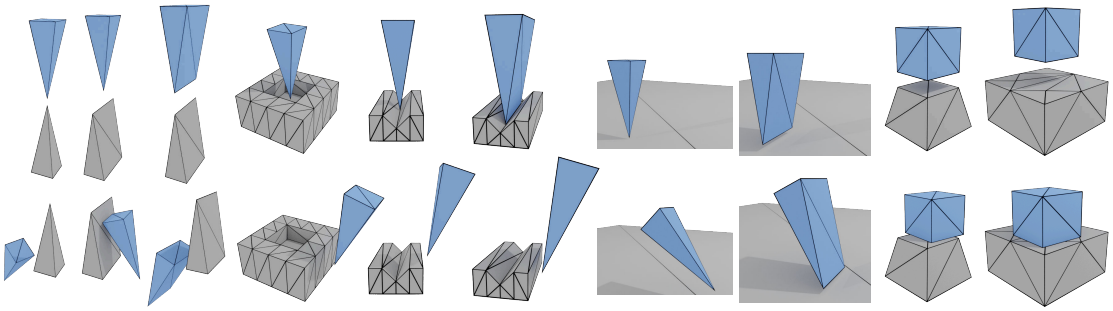


Figure 4.10: **Erleben tests** Top: fundamental test cases to challenge mesh-based collision handling algorithms proposed by Erleben [2018]. Bottom: IPC robustly passes all these tests even when stepped at frame-rate size time steps.

Erleben fundamental cases Erleben [2018] proposes unit tests (see Figure 4.10 top row) for contact constraint failure testing. Here these tests are again simple but designed to challenge mesh-based collision-handling algorithms. IPC again resolves all tests robustly (see Figure 4.10, bottom row), even when stepped at frame-rate size time steps.

Tunneling Tunneling through obstacles when simulating high-speed velocities is a common failure mode in dynamic contact modeling. We thus add an example to our unit tests: we fire an elastic ball (diameter 0.1m) at a fixed 0.02m thin board at successive speeds of 10m/s, 100m/s, and 1000m/s stepped at $h = 0.02s$. IPC accurately rebounds at large time step without tunneling in all cases.

Large mass and stiffness ratio tests Contact resolution between objects with largely varying scale, mass, and/or stiffness ratios has long-challenged time stepping methods due

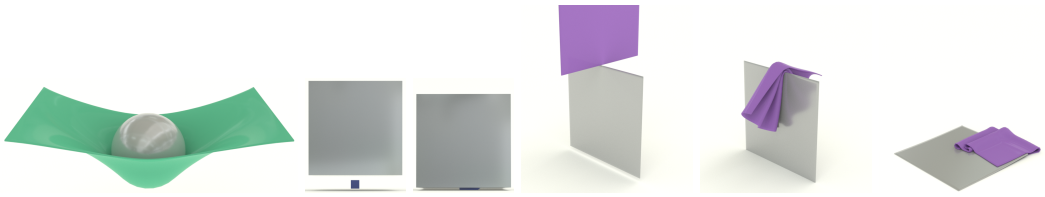


Figure 4.11: **Large Mass and Stiffness ratios.**

to ill-conditioning. In Figure 4.11, we simulate IPC dropping of a range of objects upon each other with widely varying weight and stiffness. Here we apply $E = 0.1GPa$ for the sphere, board, and large cube, $E = 1MPa$ for the small cube and the mat holding the sphere, and $E = 10KPa$ for the mat dropped on boards. For the stiff ball and large cube, we set their respective densities to $2\times$ and $10\times$ that of softer objects ($1000kg/m^3$) to add large mass ratios to the challenge. Regardless of these different ill-conditioned settings, IPC simulates all scenes robustly and efficiently without any artifacts; see also our supplemental videos Li et al. [2020b].

Chains While resolving transient collisions exercises stability, large numbers of persistent, coupled contacts, as in a long chain of elastic links, exercises contact constraint accuracy. A small amount of constraint error integrated over time will cause such chains to break. We simulate chains of 100 elastic links under gravity, observe stable oscillations and shock-propagation while shorter chains stably bounce – all preserve constraints; see our supplemental videos Li et al. [2020b].

4.7.2 Stress tests

We next consider IPC's ability to resolve a range of extreme stress-test examples motivated by well-known pre-existing challenges and previously proposed benchmarks.

Funnel To confirm contact resolution under strong boundary conditions, extreme compression, and elongation, we pull a stiff NH material dolphin model through a codimensional funnel mesh obstacle. We step IPC at large time steps of $h = 0.04$s with up to 32.3K

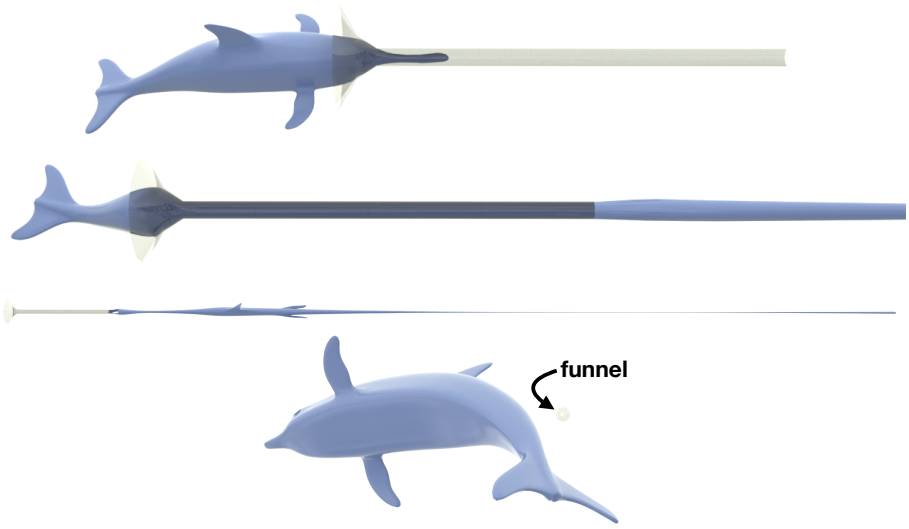


Figure 4.12: **Funnel test.** Top: the tip of a stiff neo-Hookean dolphin model is dragged through a long, tight funnel (a codimensional mesh obstacle). Middle top: due to material stiffness and tightness of fit the tip of the model is elongated well before the tail pulls through. Middle bottom: extremity of the deformation is highlighted as we zoom out immediately after the model passes through the funnel. Bottom: finally, soon after pulling through, the dolphin safely recovers its rest shape. We confirm that the simulation is both intersection- and inversion-free throughout all time steps.

contacts per step. The resulting simulation is intersection- and inversion-free throughout with the model regaining its rest shape once pulled through (Figure 4.12).

Thin volumetric meshes Thin geometries notoriously stress contact simulations. Likewise, as more simulation nodes are involved in collision stencils, simulation challenges grow. Here we test IPC's handling of extreme cases with both challenges, by simulating single layer meshes of tetrahedra. Here IPC robustly handles the contacts with accurate solutions at all time steps across a range of large deformation contact examples (Figures 4.4, 4.11 and 4.13).

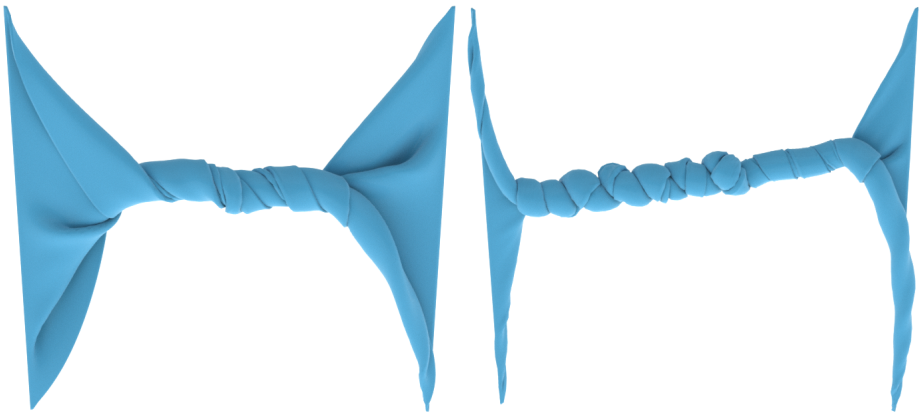


Figure 4.13: **Stress test: extreme twisting of a volumetric mat for 100s.** Left: IPC simulation at 10s after 2 rounds of twisting at both ends. Right: at 40s after 8 rounds of twisting. This model, designed to stress IPC, has all of its 45K simulation nodes lying on the mesh surface.

Extreme and extended twisting As large deformation high-contact examples, we twist thin mats (Figure 4.13), rods (Figure 4.3), and Armadillos (Figure 4.20, bottom) with rotating speeds of $72^\circ/s$ at both ends. We simulate the twist of both the rods and mats for 100s – efficiently capturing increasingly tight conforming contact and expected buckling in all simulations.

Compactor test In Figure 4.14 we test "trash" compactor-type examples from Harmon et al. [2009]. After releasing the compactor from the extreme compression point we clearly see that the tentacles of the octocat model and correspondingly the sphere, mat, and bunnies models are all cleanly separated.

Rollers compression and stick-slip instability To combine extreme deformation with friction, we match the set-up of the kinematic roller test from Vershoor and Jalba [2019] with the same originally applied, high friction coefficient $\mu = 0.5$ (Figure 4.15). This scene is highly challenging due to the competing large magnitude of the friction and the large compression induced by the rollers. Here, with a moderately stiff material ($E = 5 \times 10^5 Pa$

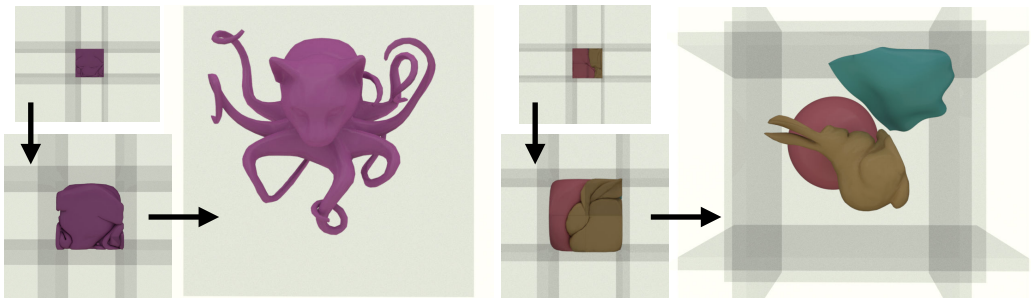


Figure 4.14: **Trash Compactor.** An octocat (left) and a collection of models (right) are compressed to a small cube by 6 moving walls and then released. Here, under extreme compression IPC remains able to preserve intersection- and inversion- free trajectories solved to requested accuracies.

Young's modulus) we observe that IPC with our friction model obtains the expected stick-slip instability effects that such competition should generate. In simulation we observe deformation grows in opposition to static friction in the rollers until stress overcomes static friction and we observe slip – this process is then repeated. This stick-slip effect is captured by our Armadillo with moderate stiffness when tested with both the NH and FCR elasticity models (see our supplemental videos Li et al. [2020b] for the motion). We also note, as expected, when we subsequently test with softer material, i.e., $E = 10^5 Pa$, we get smooth rolling behavior for the Armadillo, as expected, without stick slip.

Codimensional collision obstacles Collision obstacles, especially in animation and gaming, are often most easily expressed in their default form as triangle meshes or even unorganized triangle soups. While highly desirable in applications, codimensional collision types are not generally supported by available simulation methods, which often suffer tunneling, snagging, and resulting instabilities when exposed to them. To our knowledge IPC is the first algorithm to stably and accurately resolve collisions between volumes and codimensional collision objects. We perform a set of tests dropping different objects on planes, segments, and points, see e.g. Figures 4.1, 4.16 and 4.17. Collisions are stably resolved and we see tight compliance to the sharp poking obstacles in contacting regions.

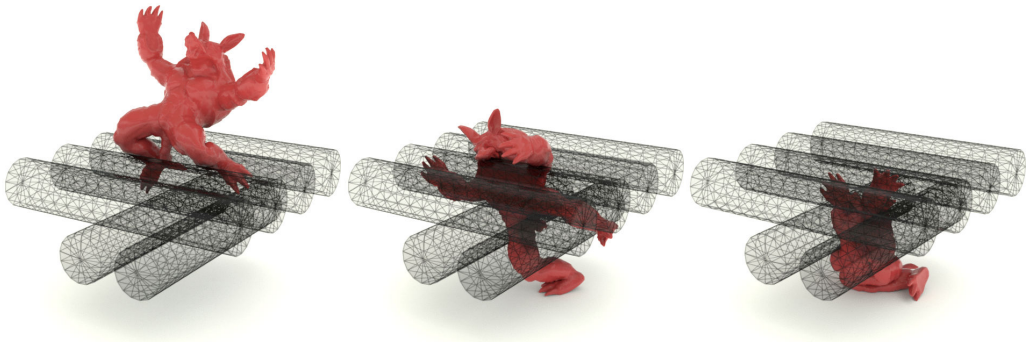


Figure 4.15: **Roller tests.** Simulating the Armadillo roller from Verschoor and Jalba [2019] (same material parameters) in IPC now captures the expected stick-slip behavior for the high-friction, moderate stiffness conditions.

Codimensional rollers What if we modify the roller test in Figure 4.7, leaving only the edges or even only the points for the roller obstacles? This leads to our codimensional roller tests (Figure 4.17). Here, with solely codimensional wire (just edges) and points (just vertices) rollers, the big ball still pulls inwards, forming tightly pressed geometries in the contact regions as it is compressed and pulled against and then out of the codimensional rollers (Figure 4.17). For sharp point rollers we require negligible friction (for points we apply $\mu = 10^{-3}$) to pull the ball inwards as the sharp points directly grab the deforming surface.

Squeeze out stress test A plate compresses and then forces a collection of complex soft material models into a tight conforming mush through a thin co-dimensional tube obstacle. Once through they cleanly separate (Figure 1.2).

High speed impact test To examine IPC's fidelity in capturing high-speed dynamics we match the reported material properties and firing speed of an experiment of foam practice ball fired at high-speed towards a fixed steel wall. In Figure 4.18 top, we show key frames of a high-speed capture of the event. Middle: we visualize velocity magnitudes simulated by IPC , stepped with implicit Newmark and the NH material, at the same corresponding

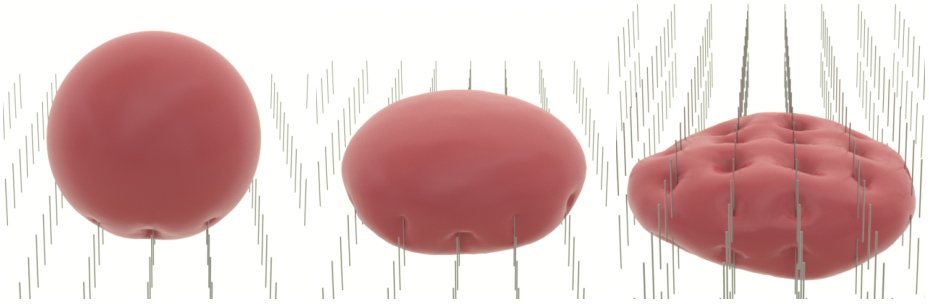


Figure 4.16: **Codimensional collision objects: pin-cushions** We drop a soft ball onto pins composed of codimensional line-segments and then torture it further by pressing down with another set of codimensional pins to compress from above. IPC robustly simulates the ball to a "safe", stable resting state under compression against the pins.

times in the simulation, and bottom the IPC-simulated geometry. Here we observe both the expected shockwave propagating through the sphere during the finite-time collision as well as the overall matching dynamics and shape across the simulation. Please see our supplemental video Li et al. [2020b] for complete simulation moving through the phases of inelastic collision impact: compression (first shockwave), restoration (second shockwave), and release.

4.7.3 Frictional contact tests

To examine IPC's frictional model we simulate a set of increasingly challenging frictional benchmark tests. All utilize a tight accuracy of $\epsilon_v = 10^{-5}m/s$ and apply lagged iterations to update sliding bases and normal forces until the system is confirmed as fully converged by satisfying (4.11).

Block tests We start by placing stiff elastic blocks on a slope with tangent at 0.5. Here for $\mu = 0.5$, IPC generates the expected result of frictional equilibrium – the block does not slide. Switching to $\mu = 0.49$, IPC then immediately sets the block sliding, again matching the analytic solution.

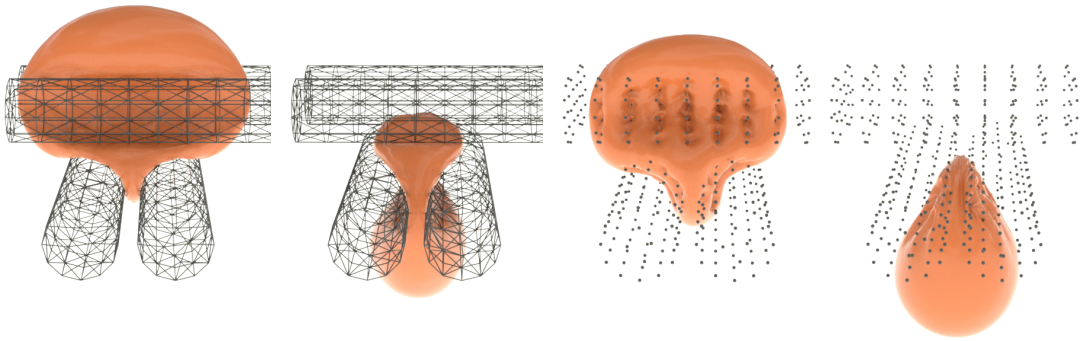


Figure 4.17: **Codimensional collision objects: rollers** We modify the ball roller example from Figure 4.7 by using only the edge segments (left) or even just vertices (right) for the moving roller obstacle. For these extremely challenging tests IPC continues robust simulation exhibiting tight compliant shapes in contact regions pressed by the sharp obstacles.

Frictionally dependent structures We test IPC on the challenging, frictionally dependent stable structure tests from Kaufman et al. Kaufman et al. [2008]. We model both the card house (Figure 4.4) and masonry arch (Figure 4.6) with stiff deformable materials. We further extend the challenge of the arch with a precarious base balanced on sharp edges. We obtain long-term stable structures with $\mu = 0.5$ and $\mu = 0.2$ respectively and confirm that they fall apart as we reduce to $\mu = 0.2$ and $\mu = 0.1$ respectively (see our supplemental document and video Li et al. [2020b] for statistics and animation).

Stick-slip instability Finally, we script the motion of the top of a thin, volumetric elastic rod pushed slightly down towards, and then along a surface ($\mu = 0.35$) to test stick-slip oscillations. As in the Armadillo roller example, large static friction creates a buildup of elastic energy in the rod which is released when the friction force, opposing sliding contact, is exceeded by the tangential stiffness at the contact. This interaction between the friction forces and the sliding velocities becomes periodic, and so induces self-excited oscillations that buildup and dissipate energy; see Figure 4.19 and our supplemental video Li et al. [2020b].

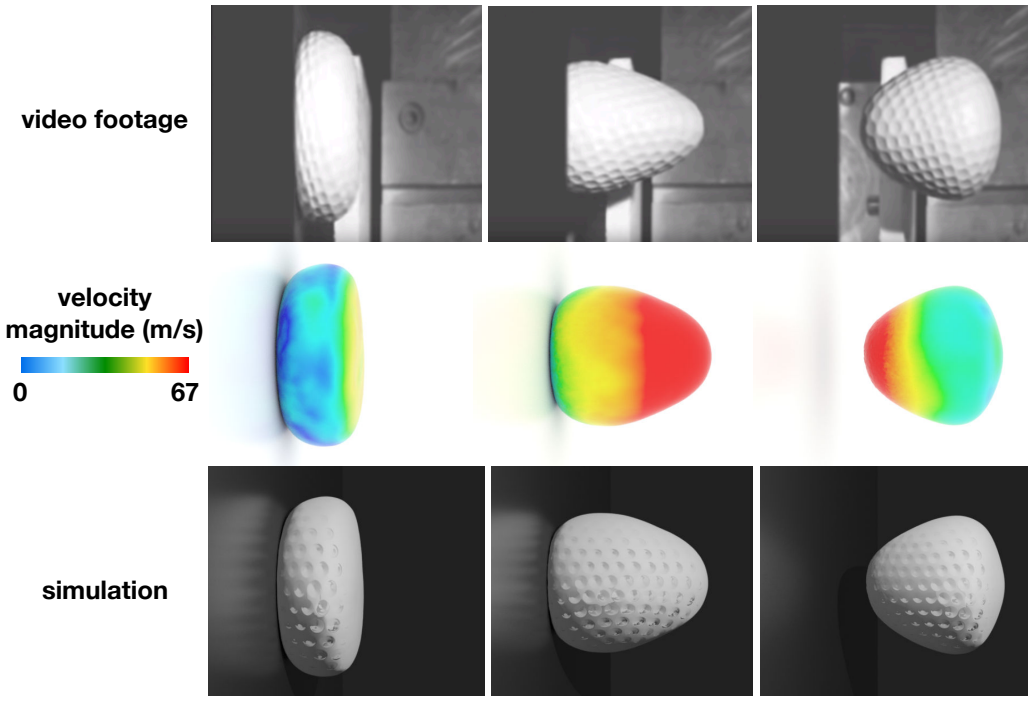


Figure 4.18: **High-speed impact test.** Top: we show key frames from a high-speed video capture of a foam practice ball fired at a fixed plate. Matching reported material properties (0.04m diameter, $E = 10^7$Pa, $\nu = 0.45$, $\rho = 1150\text{kg/m}^3$) and firing speed ($v_0 = 67\text{m/s}$), we apply IPC to simulate the set-up with Newmark time stepping at $h = 2 \times 10^{-5}$s to capture the high-frequency behaviors. Middle and bottom: IPC-simulated frames at times corresponding to the video frames showing respectively, visualization of the simulated velocity magnitudes (middle) and geometry (bottom).

4.7.4 Scaling, Performance, and Accuracy

Varying time step sizes Existing contact-resolution methods generally rely on small time step sizes for simulation success. As demonstrated above, IPC is able to simulate across a wide range of time step sizes h and so can capture a range of different frequency effects. Choice of time step size for IPC is then simply a question of accuracy required per application as balanced against efficiency needed, rather than a predicate required for success. To investigate the effect of varying time step size, h, in IPC we simulate the tight

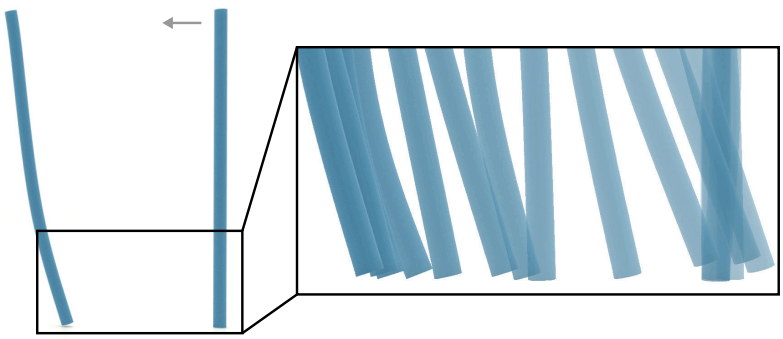


Figure 4.19: **Stick-slip** oscillations with friction simulated with IPC by dragging an elastic rod along a surface.

twisted rods example (Figure 4.3) for 6s. We range h from $0.002s$ to $2s$. In Table 4.1 we observe that transitioning from large to small time step sizes, our method improves its *per time-step* performance – but not by orders of magnitude. This is because the costs of intersection-free time stepping, distance computation and CCD do not change much. Since we do not miss any contacts, the number of constraints we process decrease only sublinearly as we decrease time step sizes. This is a key computational feature to ensure feasibility and robustness. On the other hand, we happily observe that our method is robust even well beyond standard time step sizes. While, in general, such excessively large step sizes beyond frame-rate are not useful for dynamics, this offers a robust opportunity for quasi-statically computing equilibria subject to challenging contact conditions. When we deploy IPC with implicit Euler IP (taking advantage of numerical dissipation), these very-large time steps rapidly compute equilibria with extreme contact conditions in just a few steps.

Scaling In Figure 4.20 top, we study scaling behavior of IPC , with twisting mat (Figure 4.13) and twisting Armadillo (Figure 4.20, bottom) simulations of increasing-resolution meshes ranging from 3K to 219K nodes. Armadillo is a representative volumetric model while the single-layer mat is an extreme example designed to especially stress IPC . The mat meshes importantly have all simulation nodes on their surfaces and so, as contacts tighten in the twisting mat, they can form arbitrarily dense Hessians. For the mat we

Table 4.1: **Increasing time step sizes to frame-rate and beyond.** Here we demonstrate tradeoffs in varying time step sizes for the same tight twisted rods example (2.5K nodes, 6.9K tets, 4.6K faces, $E = 10^4 Pa$) with a 4-core 2.9GHz Intel Core i7 CPU, 16GB memory. # iters is the number of Newton iterations *per time step* or in *total* for the simulation sequences.

h (s)	# constraints	per time step		total	
	avg (max)	t (s)	# iters	t (s)	# iters
0.002	137 (430)	0.29	2.12	862	6351
0.005	194 (584)	0.36	2.37	435	2843
0.01	269 (707)	0.38	2.65	229	1591
0.025	435 (1.0K)	0.38	2.69	91	645
0.05	551 (1.2K)	0.46	3.06	56	368
0.1	597 (1.3K)	0.73	4.75	44	285
0.2	607 (1.2K)	1.79	14.37	54	431
0.5	653 (1.4K)	11.39	100.58	137	1207
1	708 (1.3K)	18.41	188.17	110	1129
2	843 (1.3K)	52.02	522.00	156	1566

observe iteration count, memory and contact counts increase linearly with resolution, while timing increases in a slight superlinear trend. For the more standard volumetric Armadillo model we observe iteration count remains flat as we increase resolution, while timing and memory increase linearly.

In addition, when mesh sizes and contacts grow large, available memory can potentially preclude application of direct linear solvers. To confirm IPC applicability in these settings we simulate the firing of a 688K node, 2.3M tetrahedra, squishy ball model from Zheng and James [2012] at a glass board using AMGCL's Demidov [2019] multigrid-preconditioned iterative linear solver. Here both the large element count and the large numbers of collisions enabled by the toy's many colliding tendrils introduce very large system solves during the

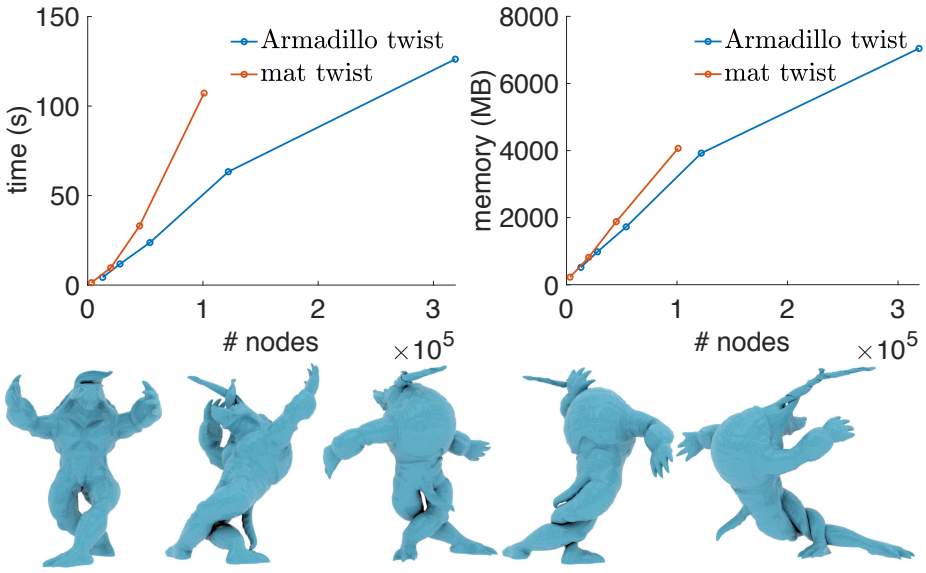


Figure 4.20: **Scaling tests.** Top: applying increasing resolution meshes ranging from 3K to 219K nodes we examine the time (left) and memory (right) scaling behavior of IPC on a range of resolutions of the twisting Armadillo and twisting mat (Figure 4.13) examples. Bottom: frames from the highest-resolution twisting Armadillo example (219K nodes, 928K tets).

most contact-rich steps colliding against the glass (Figure 4.21).

Performance Comprehensive statistics on all simulations, models, parameters and performance are reported in Table 4.22 and in our supplemental document Li et al. [2020b]. For reference dynamics please see our supplemental videos Li et al. [2020b].

Accuracy User-facing parameters in IPC have three accuracies that can be specified: 1) dynamics accuracy (ϵ_d), defining how well dynamics are resolved; 2) geometric accuracy ($\hat{d}$), defining how close objects can come to touching; and 3) stiction accuracy (ϵ_v), defining how well static friction is resolved. All three provide users direct and intuitive control (with meaningful physical units) of the trade-off between accuracy and compute cost.

In our extensive testing, IPC converges to satisfy these requested accuracies while

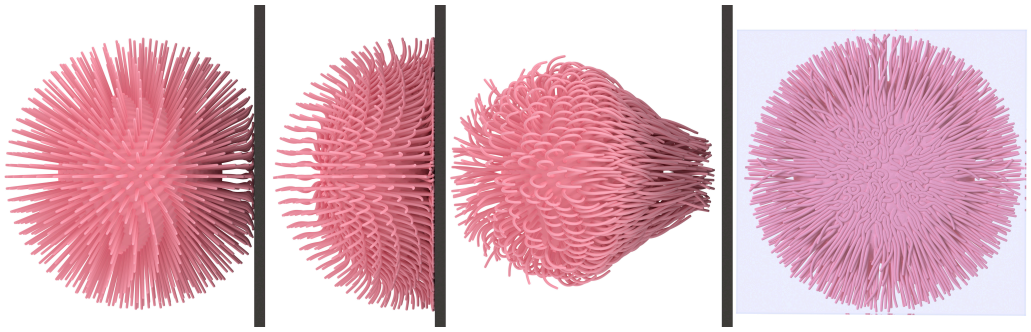


Figure 4.21: **Squishy ball test:** Simulated by IPC an elastic squishy ball toy model (688K nodes, 2.3M tets) is thrown at a glass wall. The left three frames show side views before, at, and after the moment of maximal compression during impact. The right-most frame then shows the view behind the glass during the moment of maximal compression, highlighting how all of the toy's intricately intertwined tendrils remain intersection free.

always maintaining an intersection- and inversion-free state. These guarantees (non-intersection, non-inversion) hold even as we radically increase speed of collision at large time step, apply extreme deformations, and model highly stiff materials. We have tested this across a wide range of test examples with material stiffnesses up to $E = 2 \times 10^{11} Pa$ and have confirmed our IPC implementation's ability to converge to tight tolerances for all these measures when requested with ϵ_d down to 10^{-7}m/s, ϵ_v down to $10^{-8} m/s$, and $\hat{d}$ down to $1 \mu m$.

As we discuss and demonstrate in Sections 4.3 and 4.8, all previously available methods introduce computational error for these accuracy measures; to our knowledge, IPC is the first to provide and expose direct and separable control of them. Our singular exception, as detailed in Section 4.5 above, is the number of frictional lagging iterations applied. When accurate friction is required, e.g., our arch, stick-slip and card house experiments, we set no upper bound on this parameter. Then as discussed above, in these examples IPC fully converges and is entirely parameter-free. However, (as detailed above) we do not have convergence guarantees for lagging, and in our large-deformation frictional examples we apply a single lagging iteration. In these cases, as discussed in Section 4.5.5, sliding

directions and contact forces in the friction may not match. However, even in such cases all other guarantees, including non-intersection are maintained. We observe high-quality results regardless of number of lagging iterations applied or accuracies specified.

Finally, on the other end of the spectrum in many applications, e.g. animation, it can be desirable to trade accuracy for efficiency. We confirm robust, plausible behavior for IPC when we set very large, loose tolerances on all the above parameters, e.g. with $\epsilon_d = 10^{-1}$m/s, while still maintaining feasible (non-inverting, non-intersecting) trajectory guarantees.

Exact CCD admissibility check IPC's collision aware line search ensures intersection-free trajectories. Our implementation applies standard floating-point CCD[2] combined with the conservative advancement strategies detailed in Section 4.4 and our appendix to ensure efficient, intersection-free stepping. Exact CCD then offers the possibility for aggressive advancement of intersection-free steps and so improved efficiency. To this end we tested the robust CCD methods from both Bridson et al. [2002] and Tang et al. [2014] but found the reference implementations for each missed critical intersections in degeneracies. We then reimplemented Bridson et al. [2002] with rationals. While this version now guarantees exactness, it is much slower ($\sim$30x) than our floating-point implementation. Currently we apply this exact CCD just for re-analysis as a post step check after every Newton iterate to test three of our challenging contact stress tests: octocat on codimensional "knives", ball roller and mat twist. We confirm that every step taken in every time step was intersection free in these examples.

Varying material model A general expectation from unconstrained simulation is that modeling with non-invertible materials like NH should be more costly than comparable set-ups with invertible materials like FCR. However, when studying our large deformation examples with contact we find that the picture is more complex. Here the larger bottleneck is generally resolving contact barrier terms. In many examples we then observe that

[2]https://github.com/evouga/collisiondetection

simulations with NH and FCR have comparable cost. In a number of other simulations with extreme contact conditions (e.g. pin-cushion and mat twist) element degeneracies allowed by FCR actually increase overall cost of simulation well over the same simulations with the NH material. Finally, in other cases where stress is most extreme (e.g. armadillo roller and dolphin funnel), NH entails more cost than the comparable simulation with FCR.

4.8 Comparisons

We perform extensive quantitative comparisons with existing algorithms and commercial codes used in both computer graphics (Section 4.8.1) and mechanical engineering (Section 4.8.2). Then, to more fairly compare across a large class of previous contacts algorithms based on SQP-type methods, we implement their core contact resolution procedures in a single framework, and perform a large scale comparison on our benchmark test set (Section 4.8.3). While our implementations are not finely tuned as for the first two sets of comparisons, this approach allows us to compare the core algorithmic components in a common, objective and unbiased context.

4.8.1 Computer Graphics Comparisons

Contact algorithms in graphics often target performance with small compute budgets and so admirably face many efficiency challenges in balancing fidelity against speed. We investigate what happens if we push these methods' settings to be most accurate without regard to speed, e.g., max iteration caps of 1M per step and time steps down to $10^{-5}s$. Here, nevertheless, we still document failures, e.g., tunneling, non-convergence, instabilities and ghost forces, even on very simple test examples.

Verschoor and Jalba [2019] We apply the reference implementation Verschoor and Jalba [2019] to reproduce available scenes from Verschoor and Jalba with their default and reported input parameters. Here we observe that small adjustments to time step

sizes and material parameters lead to divergent simulations. Specifically, the Armadillo roller example does not converge when applying the implementation's default time step of $h = 10^{-3}$ for a range of stiffnesses of $E = 5 \times 10^4, 5 \times 10^5$, and 5×10^6Pa, nor when applying the default material setting $E = 5 \times 10^5 Pa$ for a range of time step sizes of $h = 10^{-3}, 2 \times 10^{-3}, 4 \times 10^{-3}$, and 10^{-2}s. In all these cases the implementation maxes out at its default max-iteration cap of 1M.

We extract the Armadillo mesh, roller models and replicate the same example in IPC with identical scene settings. Here it is noteworthy that IPC applies fully nonlinear NH and FCR models with variational friction while the reference code (matching paper) linearizes elasticity once per time step. As covered in Section 4.7.2, IPC obtains the stick-slip oscillations expected in this setting (see also our video Li et al. [2020b]), when rolling the Armadillo. This does not match the Verschoor and Jalba reference code nor paper video. Artificial material softening due to the per-step linearization of Verschoor and Jalba's elasticity likely explains the difference. We confirm this in Section 4.7.2 where IPC's fully nonlinear simulation of the Armadillo roller, with a softer $E = 10^5 Pa$ ($5\times$ softer) does not, as expected, stick-slip.

SOFA SOFA Faure et al. [2012] is an open-source simulation framework featuring a range of physical models. These include deformable models via FEM. We modify a SOFA demo scene to simulate the five-link chain example with the top link fixed and four free FE links. We use the linear elasticity model (most robust) and found SOFA to provide a stable solution for the chain with large time steps up to $h = 10^{-2}$s. We extend the chain to ten links and are unable to find a converging time step size (tested down to $h = 10^{-4}$s).

Houdini Houdini SideFX [2020] is a widely used VFX tool that provides two performant simulation methods for deformable volumes: 1) a FE solver with co-rotated linear and neo-Hookean materials, and 2) Vellum, a state-of-the-art PBD solver. While capable of producing impressive effects – especially for rapid collision denting and bouncing, we find that both solvers suffer in different ways when enforcing contact constraints accurately is

critical. As a simple demonstration we again apply the chain example.

Trying a simple, lower-stiffness, 5-link chain we aim Houdini's FE solver towards robustness over speed by finely tetrahedralizing the link rings ($\sim$ 8000 tets per ring), applying small time steps (we tried increasing solver substeps to $h \approx 1ms$), and increasing collision passes (up to 16). Up to and including these maximum settings we observe rings tunneling through. We verify the same tunneling with both FE solvers provided in Houdini 18 (GNL,GSL), with both available materials. With similar stretchy material, IPC is able to accurately resolve the chain collisions even with a much coarser mesh ($\sim$ 500 tets per ring), and frame-rate size time steps, e.g. $h = 0.04s$.

For the same 5-link scene, Houdini's Vellum PBD system does better, avoiding tunneling. However, as we increase numbers of links different tradeoffs (expected of PBD) are exposed. For example, a 35-link chain, requires collision passes and/or substeps to be increased quite high to prevent tunneling. However, this unavoidably changes the material (stiffer) and introduces biasing, in this case with sideways ghost forces. Careful experimentation with substep, smoothing, and constraint iteration parameters do not help alleviate these issues. For long chains (e.g. 100 links) we confirm IPC produces stable results, with accurate physical effects (e.g. shockwaves). See our supplemental videos Li et al. [2020b].

4.8.2 Comparison with engineering codes

We compare IPC with two commercial engineering codes, COMSOL [2020] and ANSYS [2020], and one open-source engineering simulation framework Krause and Zulian [2016]. For all three codes we set up exceedingly simple scenes involving small numbers of objects. All three methods generate intersection during simulation and exhibit instabilities highly dependent on parameters and tuning choices. In stark contrast to these three engineering solutions, IPC resolves a range of contact problems, demonstrates robust output across parameters, and ensures feasible trajectories.

4.8.3 Large scale benchmark testing with SQP-type methods

We focus on frictionless contact to compare a wide range of recently developed, implicit time-stepping algorithms. Removing the various and diverse treatments for friction allows us to carefully consider behavior with contact for a broad set of recent methods Daviet et al. [2011], Harmon et al. [2008], Jean and Moreau [1992], Kane et al. [1999], Kaufman et al. [2008, 2014], Macklin et al. [2019], Otaduy et al. [2009], Verschoor and Jalba [2019] in a common test-harness framework. This is because all these methods, once friction is removed, follow a common iterated, Newton-type process to solve each time step as follows: 1) To help reduce constraint violation heuristic distance offsets/thickenings are applied to constraints; 2) at the start of each time step collision detection is performed to update a running estimate of active constraints; 3) The currently determined active (and possibly offset) constraint set and the IP energy are respectively approximated by first and second-order expansions; 4) The resulting quadratic energy is minimized subject to the linearized inequality constraints. This is a QP problem and so a bottleneck. A wide range of algorithms thus focus particularly on the efficient solution of this QP with custom approaches including QP, CR, LCP and nonsmooth-Newton strategies. Given the common sequential QP structure, we will jointly refer to them going forward as SQP-type. 5) A resulting displacement is then found and applied to the current iterate. This entire process is then repeated until a termination criteria is reached.

The above methods then differ in amount of offset, choice of constraint function, active set update strategy, IP approximations – most in graphics use just a fixed quadratic energy approximation (and so linearized elasticity) per time step, and choice of QP solver.

Here we focus on the ability of these methods to achieve convergent and accurate solves on a benchmark composed of our unit tests from Section 4.7.1 and a few additional low-resolution examples. To eliminate uncertainty of errors from the wide range of QP methods, we use the same state-of-the-art, albeit slow, QP solver Gurobi Gurobi Optimization [2019] for all methods and test each simulation method across a grid of variations on an HPC cluster.

We implement three common constraint types: the projected gap function, see e.g., Harmon et al. [2008]; the volume based proxy of Kane et al. [1999]; and the CCD-based gap function, see e.g. Otaduy et al. [2009] and Vershoor and Jalba [2019]. For each constraint type we test on a 3D sweep of (a) time steps (10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}s), (b) constraint offsets (10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}), and (c) both fully nonlinear SQP and the graphics-standard of per time-step fixed quadratic approximation of the elastic energy with nonlinear constraints.

A general pattern appears in our results: for simulations to succeed all methods require small time step and/or large constraint offset. With large time steps accuracy of the constraint linearization diminishes, thus larger constraint offsets are necessary to compensate for constraint violations. A too large constraint offset leads to failures as the local QP may become infeasible. Additionally, with large constraint offset, a constraint pair may initially violate the constraints (a common-case for self-collision due to arbitrarily small distances between elements). While it is possible to recover from such initial constraint violations, this rarely happens in our experiments. In contrast, we (re-)confirm IPC is unconditionally robust across all test cases and time steps in the benchmark.

4.9 Summary

In summary, IPC provides an exceedingly flexible, efficient, and unconditionally feasible solution for volumetric, mesh-based nonlinear elasticity simulations with self or external, volumetric or codimensional contacts. Guaranteeing intersection- and inversion-free output, IPC allows both computer graphics and engineering applications to run simulations by directly specifying just physically and geometrically meaningful parameters and tolerances as required per application.

At the same time much more remains to be done. While we have enabled a first of its kind "plug-and-play" contact simulation framework that provides convergent, intersection- and inversion-free simulation, clearly costs rise as scene complexity (both in contacts enforced and mesh resolutions) increase. There are thus many promising directions for

future improvement that are exciting directions for exploration including further customized Newton-type methods, practical speed exact CCD, extensions to higher-order elements and improved convergence for frictional contact. We emphasize that we have no guarantee for convergence of lagged friction for λ and T (although we do for stiction) and so another meaningful avenue of future development is better exploration and understanding of its behavior.

Our hope is to enable engineers, designers, and artists to utilize predicative, expressive, and *differentiable* simulation, free from having to perform extra per-scene algorithmic tuning or deviation from real-world physical parameters. We look forward to enabling design, machine learning, robotics, and other processes reliant on automated and reliable simulation output across parameter sweeps and iterations and hope to better enable artists to use real-world materials and settings as useful design tools for creative exploration.

Figure 4.22: **Simulation statistics** for IPC on a subset of our benchmark examples. Complete benchmark statistics are summarized in our supplemental document Li et al. [2020b]. For each simulation we report geometry, time step, materials, accuracies solved to ($\hat{d}$, ϵ_d and ϵ_v are generally set w.r.t. to bounding box diagonal length l), number of contacts processed per time step, machine, memory, as well as average timing and number of Newton iterations per time step solve. When applicable, for friction we additionally report number of lagged iterations, with number of iterations set to $*$ indicating lagged iterations are applied until convergence until (4.17) is satisfied. We apply implicit Euler time stepping and the neo-Hookean material by default unless specified in example name; i.e., "NM" for implicit Newmark time stepping and "FCR" for the fixed-corotational material model.

Example	nodes, tets, faces	h(s)	ρ(kg/m^3), E(Pa), ν	$\hat{d}$ (m)	μ, ϵ_v(m/s), friction iterations	ϵ_d(m/s)	contacts avg. (max.) (per timestep)	machine	memory (MB)	timing (s), iterations (per timestep)
Ball on points	7K, 28K, 10K	0.04	1000, 1e4, 0.4	1e-3l	N/A	1E-02l	126 (182)	4-core 2.9 GHz Intel Core i7, 16 GB memory	229	2.8, 6.6
Mat on knives	3.2K, 9.1K, 6.4K	0.04	1000 2e4, 0.4	1e-3l	N/A	1E-02l	291 (472)	4-core 2.9 GHz Intel Core i7, 16 GB memory	147	1.4, 5.5
100 chains	20K, 49K, 40K	0.04	500, 1e7, 0.4	1e-3l	N/A	1E-02l	40K (53K)	4-core 2.9 GHz Intel Core i7, 16 GB memory	450	4.0, 2.4
Dolphin funnel	8K, 36K, 10K	0.04	1000 1e4, 0.4	1e-3l	N/A	1E-02l	7K (31K)	4-core 2.9 GHz Intel Core i7, 16 GB memory	357	27.9, 39.7
Pin-cushion compress	9K, 28K, 10K	0.04	1000 1e4, 0.4	1e-3l	N/A	1E-02l	317 (496)	4-core 2.9 GHz Intel Core i7, 16 GB memory	233	3.7, 9.5
Golf ball (NM)	29K, 118K, 38K	2E-05	1150, 1e7, 0.45	1e-3l	N/A	1E-02l	1K (4K)	4-core 2.9 GHz Intel Core i7, 16 GB memory	861	12.1, 9.3
Mat twist (100s)	45K, 133K, 90K	0.04	1000 2e4, 0.4	1e-3l	N/A	1E-02l	264K (439K)	8-core 3.0 GHz Intel Xeon, 32GB memory	4,546	776.2, 34.5
Rods twist (100s)	53K, 202K, 80K	0.025	1000 1e4, 0.4	1e-3l	N/A	1E-02l	243K (498K)	8-core 3.0 GHz Intel Xeon, 32GB memory	2,638	141.5 ,14.1
Trash compactor: ball, mat and bunny	15K, 56K, 22K	0.01	1000 1e4, 0.4	1e-3l	N/A	1E-02l	6K (132K)	8-core 3.0 GHz Intel Xeon, 32GB memory	638	61.9, 29.4
Squeeze out	45K, 181K, 60K	0.01	1000, 5e4, 0.4	1e-3l	N/A	1E-02l	37K (277K)	8-core 3.0 GHz Intel Xeon, 32GB memory	1,700	252, 42.5
Ball mesh roller	7K, 28K, 11K	0.01	1000, 1e4, 0.4	1e-3l	0.5, 1e-3l, 1	1E-02l	2.3K (5.6K)	4-core 2.9 GHz Intel Core i7, 16 GB memory	215	63.3, 58.6
Hit board house	6K, 15K, 11K	0.025	1000, 1e8, 0.4	1e-4l	1.0, 1e-5l, 2	1E-02l	7K (13K)	4-core 2.9 GHz Intel Core i7, 16 GB memory	186	10.0, 16.6
Cement Arch	216, 150, 324	0.01	2300, 2e10, 0.2	1E-06	0.5, 1e-5l, *	1E-04l	101 (118)	4-core 3.6 GHz Intel Core i7, 32 GB memory	54	0.05, 5.7
Stick-slip Armadillo roller (FCR)	67K, 386K, 24K	0.025	1000, 5e5, 0.2	1e-3l	0.5, 1e-3l, 1	1E-02l	8K (33K)	4-core 3.6 GHz Intel Core i7, 32 GB memory	3,651	346, 66.8
Squishy ball (AMGCL)	688K, 2314K, 1064K	1E-03	1000, 7e4, 0.4	1e-4l	N/A	4E-02l	3.6K (105K)	8-core 3.6 GHz Intel Core i9, 64GB memory	19,463	328.3, 12.2

Chapter 5

Strain Limiting and Thickness Modeling

5.1 Introduction

Cloth, papers, hairs, sands. These various kinds of solids have a common feature that one or two or even all their dimensions are nearly negligible when compared to their other dimensions or a cluster of them. To capture the intricate dynamic behaviors of these thin structures with self-contact is challenging in many aspects.

First, codimensional shape representations like surface or segment meshes are often used to avoid numerical stiffening on the shearing modes, that can make the material artificially harder to bend. Even when codimensional shape representations are used, for cloth there are still severe numerical stiffening on the bending mode unless the underlying surface mesh has extremely high resolution. This happens for practical simulations of cloth even when real-world anisotropic membrane stiffnesses are used. Using smaller membrane stiffness avoid this issue, but it usually makes the cloth too stretchy. Therefore, modern cloth simulation methods often apply a soft membrane energy to avoid locking but also add a strain limit constraint to avoid stretchy behaviors. By limiting both the lower and upper bound of the strain, some methods even go without membrane energies for simplicity and

efficiency.

However, existing strain limiting method can hardly guarantee that the solved strains are within the specified range, which potentially results in inconsistent material behaviors in different time steps depending on how close the violation is to the specified bound. In addition, all existing strain limiting methods handle the inequality strain limit constraint via post projection separately from contact processing and elastodynamic solves. This not only cannot guarantee simultaneous satisfaction of both contact and strain limit constraint, but also can result in large error in elastodynamics (inertia and bending), which can lead to severe artifacts.

On the other hand, with codimensional shape representation, the thickness of objects are usually only considered when computing integral of kinetic and elasticity energies, but ignored in contact handling. In fact, contact handling methods in state-of-the-art cloth simulator even rely on an appropriately large thickness parameter as an offset to define and handle the interpenetration-free constraints that could help avoid some interpenetration or jittering artifacts. But as we show in Section 5.6.5, extensive parameter tuning, e.g. time step size h, is required for this mechanism to work, making thickness control tricky for thin materials, especially when simulating scenes where multiple objects gather together. In addition, modeling thickness with constraint offset also poses significant difficulties to existing state-of-the-art continuous collision detection (CCD) solvers on their robustness and accuracy while maintaining efficiency.

CCD is an important component in contact simulation for avoiding interpenetration when advancing nodal positions. It is often conducted by computing the smallest step sizes (or times) that first bring the distances of any surface point-triangle or edge-edge pairs to 0 (first time of impact/collision), and then take the minimum of the computed values among all primitive pairs. An inaccurate CCD return value can either lead to interpenetration when it is larger than the true solution, or stall the simulation or optimization when it is too tiny or even 0. Therefore, extensive researches have been conducted to explore various trade-offs between efficiency and accuracy for different application scenarios. But none of

them can fulfill our need on gauranteeing a finite separation as inelastic thickness ξ while at the same time relying on the distances between offset surfaces that are orders-of-magnitude smaller than ξ for computing contact forces.

5.1.1 Contribution

By posing the limit of strain as a constitutive behavior, we propose constitutive strain limiting, a smooth barrier-based energetically-consistent strain limiting model supporting both isotropic and anisotropic membrane resistance. With a barrier potential energy, our strain limiting model can be accurately solved to satisfy any reasonable strain restriction fully coupled with elastodynamics and contact, ensuring simultaneous strain limit and interpenetration-free guarantees and consistent momentum balance even with frame-rate time step sizes.

We thereby propose a unified framework for simulating mixed-dimensional elastodynamics including elastic volumetric bodies, shells, rods, and particles all coupled together through robust and accurate contact and friction. Unlike traditional mixed-dimensional elastodynamics simulation methods that mainly focus on designing the connections between objects (mostly equality constraints), we here focus on accurately resolving the contact and friction (the more challenging inequality constraints). By consistently maintaining a constraint offset with our local CFL-based iterative CCD approach, our method at the first time achieves controllable thickness effect under codimensional shape representation.

5.2 Related Work

5.2.1 Cloth and Rods

Cloth Baraff and Witkin [1998], Bridson et al. [2002], Harmon et al. [2009], Li et al. [2020a], Narain et al. [2012], Volino and Thalmann [2000] and hair Choe et al. [2005], McAdams et al. [2009], Müller et al. [2012], Selle et al. [2008], Ward and Lin [2003]

simulation has been popular in computer graphics over decades. The most common cloth simulation pipeline is based on implicit time integration with collision response computed using a non-linear impact zone solver Baraff and Witkin [1998], Bridson et al. [2002], Harmon et al. [2008], Li et al. [2020a, 2018a], Narain et al. [2012], Otaduy et al. [2009], Tang et al. [2016, 2018]. As high resolution is essential for cloth simulation to capture intricate behaviors, many existing works focus on improving the timing performance via GPU acceleration Li et al. [2020a], Schmitt et al. [2013], Tang et al. [2013, 2016, 2018] or smart solving schemes like multigrid Tamstorf et al. [2015] or splitting Goldenthal et al. [2007]. Another challenge is to capture the complex anisotropic behaviors of cloth, which triggered works that explores cloth material parameter estimation from real-world data Bhat et al. [2003], Clyde et al. [2017], Miguel et al. [2012]. There are also works that explore different spatial discretizations e.g. material point method Guo et al. [2018], Jiang et al. [2017a], Eulerian-on-Lagrangian method Weidner et al. [2018], to obtain different trade-offs among performance, quality, and accuracy in multiple aspects. Recently, Liang et al. [2019] proposed an efficient differentiable cloth simulator which can be embeded as a layer in deep neural networks, and Casafranca et al. [2020] and Sperl et al. [2020] combined yarn-level simulation into continuum shell based cloth simulation to gain more accuracy and details.

Observing that guaranteed contact resolution with controllable friction, strain limiting, and thickness modeling of shells and rods is still an unsolved but essential problem, we propose our method to tackle these challenges and make shell and rod simulation robust, easy to control, and accurate.

5.2.2 Strain Limiting

Here we provide a literature review on general methods trying to maintain a strain limit or range for cloth simulation, not only limited to standard strain limiting research.

Goldenthal et al. [2007] defines equality constraints on quad mesh edge length and propose a fast projection method to efficiently post project predictive solutions towards

Methods	Constraint type	DoFs constrained	Splitting choice	Constraint solver
Goldenthal et al. [2007]	Equality	Quad edge length	3-step	Fast projection (Implicit Euler)
English and Bridson [2008]	Equality	Triangle edge midpoint distance	3-step	Fast projection (BDF2)
Thomaszewski et al. [2009]	Equality	Lagged corotational small strain on triangles	3-step	Gauss-Seidel or Jacobi
Chen and Tang [2010]	Equality	Triangle edge length	1-step	Least squares
Wang et al. [2010]	Range	SVD of triangle strain	2 or 3-step	Gauss-Seidel
Narain et al. [2012]	Range	SVD of triangle strain	3-step	Augmented Lagrangian
Hernandez et al. [2013]	Range	SVD of triangle strain with angle dependency	2-step	Projected Gauss-Seidel for LCP
Wang et al. [2016]	Range	SVD of triangle strain	2-step	SQP
Jin et al. [2017]	Inequality	Triangle edge length	3-step	Complementary (similar to fast projection)
Chen et al. [2019]	Equality	Green-Lagrangian strain on nodes (MLS)	3-step	Fast projection (Implicit Euler)

Figure 5.1: **Strain limiting methods feature table.** Here 2-step splitting refers to solving elastodynamics before a post constraint projection which simultaneously handle contact and strain limiting, while 3-step splitting means elastodynamics, contact, and strain limiting are all decoupled and solved independently per time step. In the table only Chen and Tang [2010] solve the 3 terms in a fully coupled way, but by using least squares the solution is simply an L2 approximation to the original problem.

satisfying edge length constraints. English and Bridson [2008] applies non-conforming mesh strategy and define equality constraint on triangle edge midpoint distances. They also solve the constraint via post projection by extending the fast projection method in Goldenthal et al. [2007] to support BDF-2 integrator. Thomaszewski et al. [2009] uses lagged corotational small strain on triangles to define equality constraints and apply post projection via Gauss-Seidel or Jacobi solver. Chen and Tang [2010] defines equality constraints on triangle edge length. With equality constraints per edges or triangles, these methods still suffer from some degree of membrane locking when the constraints are accurately satisfied except for English and Bridson [2008] but there the non-conforming mesh issues introduce extra challenges.

Thus, there are methods constraining the singular values of deformation gradient per triangle to a finite range to avoid membrane locking Hernandez et al. [2013], Narain et al. [2012], Wang et al. [2010, 2016]. Moreover, Jin et al. [2017] only defines inequality constraints on triangle edge length. Similar to English and Bridson [2008] that reduce the constraint number by investigating different choice of discretization, Chen et al. [2019] novelly defines equality constraints on the Green-Lagrange strain per node computed by Moving Least Squares (MLS).

Here only Thomaszewski et al. [2009] and Hernandez et al. [2013] explicitly dealt with anisotropic strain limiting, while the latter one is customized for anisotropic but not isotropic cases. Goldenthal et al. [2007], Narain et al. [2012], and Chen et al. [2019] mentioned possible ways that could extend the proposed methods to support anisotropy.

Except for Chen and Tang [2010] that applies Least Squares approximation to the entire system without friction and only for elastostatics, most existing methods solves strain limiting, elastodynamics, and contact separately, where strain limit and contact constraints cannot be simultaneously satisfied. Hernandez et al. [2013], Wang et al. [2016], and Wang et al. [2010] conduct post projection for strain limiting and contact simultaneously, but the errors from elasticity and inertia introduced by splitting the elastodynamics and constraint projection can still potentially lead to artifacts.

5.2.3 Thickness Modeling

To simulate shells or rods using codimensional shape representations, although the thickness of the objects are orders-of-magnitude smaller than other dimensions and it usually can be ignored, in situations where multiple objects gather together through contact, e.g. a pile of cards (Figure 5.18), a bowl of noodles (Figure 5.20), etc, correctly modeling thickness is still important to obtain realistic behaviors.

Directly simulating shells or rods using volumetric meshes easily solve the thickness modeling issue geometrically, where elasticity can also be succinctly modeled with pure translational DoFs Hauptmann and Schweizerhof [1998]. However, obtaining a correct

elastic behavior is challenging in this situation. Linear element volumetric shell has the well-known shear locking issue. Although higher-order elements are much more accurate, they are too expensive. Thus, traditional solid shell methods use either reduced integration with hourglass mode stabilization Reese [2007] or assumed natural strain methods Cardoso et al. [2008] to achieve locking free linear element that maintains the efficiency, which unfortunately brings back the complexity into the model.

Getting back to the discrete shell or rod regime, contact methods usually apply an offset in the constraint definition for modeling thickness to a certain degree Li et al. [2018a], Narain et al. [2012]. However, as shown in Li et al. [2020b], this offset is in fact a relaxation to the contact handling problem that try to balance contact constraint errors against time step size and collision speed for avoiding interpenetration or explosion issues. Relying on it to consistently model thickness for codimensional objects is not practical as we show in Section 5.6.5. In our work, we extend IPC with an offset barrier that can guarantee a minimal separation between the mid-surface/line geometries to consistently model thickness. This new formulation also extends the capability of simulation with codimensional objects to a lot more phenonmena than before (e.g. Figure 5.20, 5.21, etc).

5.2.4 Continuous Collision Detection (CCD)

Here we provide a brief literature review of CCD methods focusing on piece-wise linear surfaces with linear trajectory. Provot [1997] formulated computing first time of impact for point-triangle or edge-edge pairs as first solving a cubic equation for coplanarity and then performing overlap checks to see whether collision is happening. This later becomes a standard formulation of CCD and followed up by many works Harmon et al. [2009], Tang et al. [2011]. However, based on floating-point root-finding, although the computation can be quite efficient, large numerical errors can result in both false positive or false negative output especially when distances are tiny and the configuration is degenerate (e.g. parallel edge-edge, etc). Although performing the root-finding and overlap checks with

rational numbers can avoid any numerical issue, the computational cost becomes orders-of-magnitude more expensive. Therefore, there are methods Brochu et al. [2012], Tang et al. [2014], Wang et al. [2015] applying exact arithmetic for improved robustness while still maintaining efficiency. Some other methods Harmon et al. [2011], Lu et al. [2019] perform CCD by requesting a small separation for better avoiding interpenetration, which is also the strategy applied in IPC Li et al. [2020b] on top of a floating-point root-finding CCD solver[1]. But when we try to maintain a finite separation ξ between midsurface/midlines for inelastic thickness modeling and relying on the orders-of-magnitude smaller distances between the offset surface for computing contact forces, we observed that all existing CCD methods fail hardly, either leading to interpenetration or totally stall our optimization (see Section 5.6.8). Therefore, by deriving a theoretical lower bound of the time of impact, we propose our additive CCD (ACCD) method that stay in the floating-point regime but robustly approximate time of impact monotonically without the error-prone direct root-finding computations and thus works with high accuracy even for extreme cases.

5.2.5 Coupling

Finally we review simulation methods that couple codimensional DoFs. Martin et al. [2010] derive a higher-order integration rule, or elaston, measuring stretching, shearing, bending, and twisting along any axis without distinguishing between forms of different dimension (solids, shells, rods). Their single code accurately models a diverse range of elastoplastic behaviors where collision is simply point-wise penalty model since the objects are represented by a set of spheres. Chang et al. [2019] present a unified method to simulate deformable elastic bodies consisting of mixed-dimensional components by defining all types of connections between objects of different codimensions via equality constraints. They employed collision detection and resolution strategy in Bridson et al. [2002]. On the other hand, the material point method Jiang et al. [2017a] discretizes elasticity on Lagrangian particles while solve the momentum balance on the DoFs of a background

[1] https://github.com/evouga/collisiondetection

Eulerian grid. The contact between any codimensional objects are easily handled via the Eulerian grid in a unified way, but sticky and gap artifacts can be visible for not high enough grid resolution. Our method demonstrate that the recently proposed Incremental Potential Contact (IPC) Li et al. [2020b] model can naturally couple all different codimensional objects together through robust and accurate contact and friction without any artifacts or special treatment. In addition, our framework also at the first time conduct a fully implicit coupled solve of elastodynamics, contact, and strain limiting.

5.3 Formulation and Overview

We focus on the combined solution of meshed, codimensional models simulated jointly and so coupled arbitrarily via contact. To do so we generalize the IPC model to codimensional DoF and so must address the challenges introduced by thin codimensional models both coupled by and stressed via contact and largely imposed boundary conditions. Here we first cover our generalization of IPC to mixed codimensional models and then introduce the key components of our method that enable its simulation.

Contact-aware elastodynamics. For elastodynamics, we perform implicit time stepping with optimization time integration Gast et al. [2015], Kaufman and Pai [2012], Li et al. [2019], Liu et al. [2017], Overby et al. [2017], Wang et al. [2020a] to minimize an Incremental Potential (IP) Kane et al. [2000] with line search, ensuring stability and global convergence. For a wide range of time steppers and assuming hyperelasticity, the IP to update from time step n to $n+1$ is defined as

$$E(x) = \frac{1}{2}||x - \hat{x}^n||_M^2 + \alpha h^2 \Psi(\beta x + \gamma x^n), \tag{5.1}$$

with the timestep update given by $x^{n+1} = \text{argmin}_x E(x)$. Here h is time step size, M is the mass matrix, and Ψ is an elastic energy potential. Scaling factors $\alpha, \beta, \gamma \in [0, 1]$ and explicit predictor $\hat{x}^n$ then determine the time step method applied. For example, here we focus primarily on the graphics-standard implicit Euler with $\alpha, \beta = 1$, $\gamma = 0$ and

$\hat{x}^n = x^n + hv^n + h^2 M^{-1} f_{\text{ext}}$. Alternate implicit time stepping, e.g., with implicit midpoint or Newmark, are similarly matched by applying alternate scalings and predictors Kaufman and Pai [2012].

Incremental Potential Contact (IPC) then augments the IP with contact and dissipative potentials Li et al. [2020b]:

$$E(x) = \frac{1}{2}||x - \hat{x}^n||_M^2 + \alpha h^2 \Psi(\beta x + \gamma x^n) + B(x) + D(x). \tag{5.2}$$

Here $B(x)$ and $D(x)$ are respectively the IPC contact barrier and friction potentials. The contact barrier $B(x)$ enforces strictly positive distances between all primitive pairs. Following Li et al. [2020b], a custom Newton-type solver is applied to solve the IP with Continuous Collision Detection (CCD) filtering used in each line search, at every Newton iteration to ensure intersection-free trajectories throughout. Please see Li et al. [2020b] for details on the IPC algorithm and solver implementation.

Mixed-dimensional hyperelasticity. To enable a unified simulation framework for coupled volumetric bodies, shells, rods and particles, we compute mass and volume for all codimensional elements by treating them as continuum regions with respect to standard discretizations (see Section 5.2 and below), and so construct the total elasticity energy $\Psi(x)$ for the IP as

$$\Psi(x) = \Psi_{\text{vol}}(x) + \Psi_{\text{shell}}(x) + \Psi_{\text{rod}}(x). \tag{5.3}$$

Here, as representative examples, we apply fixed Corotated elasticity Stomakhin et al. [2012] for volumes (Ψ_{vol}); discrete shells hinge bending Grinspun et al. [2003], Tamstorf and Grinspun [2013], combined with isotropic/orthotropic StVK Chen et al. [2018], Clyde et al. [2017] and neo-Hookean membrane models, for shells (Ψ_{shell}); and the quadratic spring energy for stretching, combined with the discrete rods bending model Bergou et al. [2008] for rods (Ψ_{rod}). Here the generalized IPC framework is agnostic to these elasticity and time stepper choices. The above are currently selected as best for comparison and evaluation given the models standard in existing available codes for comparison. For discrete models,

along with properly integrating all lower-dimensional energies with an accurate volume weighting per element, we further parameterize rod bending moduli via Kirchhoff rod theory for direct material setting. As, to our knowledge, this is not previously covered in the literature Bergou et al. [2008], we cover the necessary details for completeness in our appendix. Now, with per-domain elasticity summed, our IPC model couples objects of arbitrary codimensions directly, without splitting, via frictional contact. Specifically all codimensional DoF are now associated with a potential energy and so are free to move by time stepping. In turn they are coupled by IPC barriers including all point-triangle and edge-edge pairs from surfaces (both volumes' and shells'), rods, and particles; point-edge pairs from all rod nodes and particles to rod segments; and finally point-point pairs between particles.

Constitutive strain limiting When it comes to codimensional shell models all but the highest-resolution meshes suffer from locking artifacts for cloth materials. To mitigate these effects strain-limiting has long been critical for simulating cloth materials with realism and accuracy. Inspired by IPC's intersection-free and inversion-free barrier formulation Li et al. [2020b], we construct a new constitutive barrier model that directly enforces strain limiting while maintaining rest-state consistency. To do so we designed a barrier energy that 1) diverges at the strain limit, and 2) has vanishing gradient and Hessian (and so neither force nor preconditioning contribution) at rest shape. This keeps whichever underlying membrane elasticity model we choose to apply energetically consistent. With this barrier construction, we apply our IPC solver and guarantee strain-limit-satisfaction for all practical (and well below) reasonable limits (verified down to 0.1%) for all iterations and so for all time steps in each IPC simulation. See Section 5.6. We show how our formulation augments existing membrane energies as an added strain limiting potential for general isotropic cases (Sec. 5.4.1), and also demonstrate its application as a new extension for orthotropic STVK membranes augmented to include anisotropic strain limits (Sec. 5.4.2). Because of its potential-based model, our isotropic (respectively anisotropic) constitutive strain limiting is

then easily and directly applied in IPC (Eq. 5.2), as just a simple addition to (respectively modification of) the elasticity potential, Ψ, in the IP. This gives a mesh-based simulation framework that produces trajectories simultaneously ensuring intersection-free and strain limit satisfying steps. In turn the resulting dynamics satisfy accurate momentum balance fully coupling frictional contact forces, strain-limiting and elasticity. Thus, as demonstrated in Section 5.6.3, our method then avoids artifacts commonly generated by force-splitting errors in traditional projection-based strain-limiting methods.

Thickness modeling. Thin structures typically have small relative thicknesses whose *elastic* behavior can be indirectly captured by codimensional DoF models. However, correctly and directly modeling thickness is crucial for capturing correct geometric behaviors. Consider, for example, a deck of cards. Each card has a nearly imperceptible thickness when considered individually. However, when stacked, their combined thickness is large and clear, and can not be ignored. While contact-processing strategies often introduce thickness parameters they are generally applied to mitigate inaccuracies. As analyzed in Li et al. [2020b], these thicknesses can not be consistently enforced and moreover must be changed heuristically per scene and example (e.g., based on collision speeds) to avoid simulation failures. To consistently enable thickness modeling for codimensional simulations we extend the IPC formulation Li et al. [2020b] to capture the geometric thickness of codimensional structures modeled by directly enforcing distance offsets. Our treatment imposes a strict guarantee that mid-surfaces (respectively lines) of shells (respectively rods) will not move closer than the applied thickness values even as these thicknesses become characteristically small; see Sec. 5.5. This enables us to account for thickness in the contacting behavior of codimensional structures and so capture effects that are challenging and in some cases have not previously been modeled nor demonstrated.

Additive CCD Contact modeling between thin codimensional models with small *but finite* thicknesses challenges all collision-detection routines. Here then, most importantly, we see unacceptable failures and/or inefficiencies in all available existing CCD methods

and codes. We see this both for standard floating point root-finding CCD methods as well as more recent developments in exact CCD methods. These issues are attributable to the well-known numerical sensitivity of the challenging underlying root-finding problems posed. Here, most specifically, we see that IPC is reliant on a bounded guarantee of non-intersection for all CCD evaluations. Large enough conservative bounds can help Li et al. [2020b] account for these inaccuracies, ensuring non-intersection, but they do so at the cost of progress and so can even stall convergence altogether when it comes to codimensional models. To address these issues we derive a new, simple to compute, numerically robust, additive CCD (ACCD) for approaching requested CCD bounds that is stable for the exceedingly small evaluation distances required. While we most immediately focus here on ACCD's application within our IPC framework, as we show in the following sections, ACCD is exceedingly simple and so easy to implement when compared to all prior CCD routines (exact and floating point) with both improved performance, robustness and guarantees, and so is suitable for easy replacement in all other applications where CCD modules are employed.

5.4 Constitutive Strain Limiting

Here we begin by constructing our constitutive strain-limiting model. We first start with the general, isotropic regime with a strain-limiting potential (Sec. 5.4.1) that augments arbitrary, existing membrane energies. Then we demonstrate extension to the anisotropic regime with an orthotropic StVK membrane that directly applies anisotropic strain limits (Sec. 5.4.2).

5.4.1 Isotropic Constitutive Strain Limiting

We define isotropic strain-limiting constraints per element t as

$$\sigma_i^t < s, \ \ \forall i, \tag{5.4}$$

where the singular value decomposition of each triangle t's deformation gradient, $F^t = U^t \Sigma^t V^{tT}$, gives $\Sigma^t = \text{diag}(\sigma_1^t, \sigma_2^t)$. The imposed constraint bound s is then the requested strain limit with practical bounds generally selected for cloth with $s \in [1.01, 1.1]$ Bridson et al. [2002], Provot et al. [1995].

Inspired by IPC's contact barrier formulation, we then construct a local energy that enforces this unilateral strain-limit. We can begin with a typical log-barrier strategy. This gives energies

$$\kappa_s \sum_i -\ln(s - \sigma_i^t), \tag{5.5}$$

with κ_s defining the barrier stiffness for limit s. While ensuring strain-limits this barrier is problematic. First, from the modeling perspective, it applies forcing everywhere. Specifically, constitutive behavior is unnecessarily changed far from strain limits where the underlying membrane model should be untouched and, worse yet, applies ghost forces at rest. Second, from the computational perspective, because this barrier is globally non-zero, evaluating it (and its derivatives) in optimization is prohibitively expensive.

From these issues we then observe that strain-limit forcing for triangles far from the strain limit is then both unacceptable and unnecessary. Instead, we want local support for nonzero forces just close to where these limits are achieved. One simple and seemingly plausible remedy, to achieve this localized support region, is to clamp the barrier in Eq. 5.5 to zero beyond a small strain threshold $\hat{s}$ (e.g. $\hat{s} = 1$) giving barrier energies

$$\kappa_s \sum_i b_i^{st}(x), \quad b_i^{st}(x) = \begin{cases} -\ln(\frac{s-\sigma_i^t}{s-\hat{s}}) & \sigma_i^t > \hat{s} \\ 0 & \sigma_i^t \leq \hat{s} \end{cases}. \tag{5.6}$$

Unfortunately, while providing local support these energies are only C^0 continuous and so do not allow for practical gradient-based optimization necessary for the efficient solution of the IP.

To smooth our strain-limiting while providing local support we then propose a C^2

smoothly clamped barrier for strain-limiting

$$b_i^{st}(x) = \begin{cases} -(\frac{\hat{s}-\sigma_i^t}{s-\hat{s}})^2 \ln(\frac{s-\sigma_i^t}{s-\hat{s}}) & \sigma_i^t > \hat{s} \\ 0 & \sigma_i^t \leq \hat{s} \end{cases} \tag{5.7}$$

.

Now, with barriers in hand, we could potentially next consider treating these barriers directly as constraints (as in primal barrier and interior point methods). However, doing so would then simply sum these barriers over elements and so would obtain inconsistent behavior as we change meshes. Instead, to provide consistent behavior we impose strain limiting constitutively as an energy density integrated over the volume of cloth to obtain the potential

$$\begin{aligned} \Psi_{\text{SL}}(x) &= \sum_i \kappa_s \int_\Omega b_i^{st}(x) dV \\ &\approx \kappa_s \sum_{t,i} V^t b_i^{st}(x). \end{aligned} \tag{5.8}$$

For our approximation, we use $V^t = A^t \xi^t$, with A^t and ξ^t respectively the area and thickness of triangle t. Our final isotropic strain-limiting potential, Ψ_{SL}, is then C^2 with local support and can simply be added to our total potential in Equation 5.3. In turn, strain limiting is then directly handled at each time step by optimization of the IPC Equation 5.2, while ensuring that strain-limit barriers are not violated in each line-search step. During steps when most triangles remain under the strain limit threshold, little additional computation is then required. While, when stretch increases, we can adjust the necessary nonzero terms from newly activated barriers and so, as we show in Section 5.6, strictly enforce strain limits while balancing all applied forces.

Strain-Limit Stiffness With our barrier potential defined we now specify setting its stiffness. The mollified clamping strategy we apply for our strain-limit model is inspired by IPC's smoothing of contact barrier energies Li et al. [2020b]. Here notice an extra ingredient for strain-limiting is the applied $1/(s-\hat{s})^2$ factor. This scaling enables us to apply our barrier directly to a limit-normalized strain, measured by $y = (\hat{s} - \sigma_i^t)/(s - \hat{s})$

with

$$b_i^{st}(y) = \begin{cases} -y^2 \ln(1+y) & y < 0, \\ 0 & y \geq 0. \end{cases} \tag{5.9}$$

In turn, as our applied strain-limit (s) and/or clamping threshold ($\hat{s}$) is varied with application, the barrier function w.r.t. y remains unchanged. Only the gradient of the linear map from σ_i^t to y varies. This allows us to apply a single, consistent initial barrier stiffness κ_s (we use $1KPa$ for all examples) across all choices of differing strain limits and clamping thresholds. Here the barrier potential curve is then simply linearly rescaled each time, to a different strain range, and so provides consistent conditioning to the system. Finally, to avoid numerical issues introduced by tiny gaps between a current strain and the imposed strain limit (e.g., when extreme boundary conditions are imposed as in Figures 5.14 and 5.15) we adapt barrier stiffness when needed. Starting with our initial κ_s, we increase it by $2\times$ (up to max bound of $\kappa_s = 0.1MPa$) whenever the strain gap $s - \sigma_i^t$ of a triangles t is less than $10^{-4}(s - \hat{s})$ over two consecutive iterations.

5.4.2 Anisotropic Constitutive Strain Limiting

Above we have constructed strain-limiting as an isotropic constitutive model. We formed a barrier energy that can be integrated and so directly added to potential energy in order to augment existing membrane models with hard strain limits. The key to making this work is the application of our C^2 continuous clamping so that the application of strain-limits does not alter rest-shape gradients nor rest-shape Hessians.

Alternately, we can directly apply a similar constitutive strain-limiting strategy to modify membrane elasticity to now include strain limits. To do so we simply substitute an anisotropic membrane model with a barrier energy that prevents violation of strain limits, while matching the original membrane energy gradient and Hessian at rest.

This second strategy is particularly motivated by the need to incorporate strain-limiting into available data-driven models that have been constructed to carefully fit measured cloth data. Here we demonstrate the specific application to the anisotropic, data-driven model of

Clyde and colleaguesClyde [2017], Clyde et al. [2017].

Their constitutive model energy is

$$\begin{aligned}\tilde{\psi}(\tilde{E}_{11}, \tilde{E}_{12}, \tilde{E}_{22}) =& \frac{a_{11}}{2}\eta_1(\tilde{E}_{11}^2) + \frac{a_{22}}{2}\eta_3(\tilde{E}_{22}^2) \\ &+ a_{12}\eta_2(\tilde{E}_{11}\tilde{E}_{22}) + G_{12}\eta_4(\tilde{E}_{12}^2),\end{aligned} \tag{5.10}$$

where $\tilde{E} = D^T E D$ is the reduced and aligned Green-Lagrangian strain, $\tilde{E}^{i3} = \tilde{E}^{3i} = 0$, and column vectors of D are the tangent and normal bases of the shell in material space. Functions η are then a sum of polynomial functions with real-valued exponents α_{ji} that satisfy $\eta(0) = 0$ and $\eta'(0) = 1$. Here the first constraint enforces zero-energy, zero-stress rest configurations, while the second constraint allows a natural correspondence between the parameters $a_{\alpha\beta}$ and G_{12} and linear elasticity at infinitesimal strain. Clyde et al. use

$$\eta_j(x) = \sum_{i=1}^{d_j} \frac{\mu_{ji}}{\alpha_{ji}}((x+1)^{\alpha_{ji}} - 1).$$

Their measured data is then restricted to deformations within the fracture limit or elasticity range of the cloth. Beyond this range meaningful extrapolation is then unlikely, due to possible overfitting inside the range (exponents α_{ji} from the fitting can range up to 10^5). To address this limitation Clyde et al. propose a quadratic extrapolation for simulation. However such extrapolation is not physically meaningful. To usefully apply data-driven modeling either fracture should be captured beyond this regime or else a stable and controllable strain limit imposed to stay in bounds. Here, we focus on controllably imposing strain limit while respecting the underlying model fitting.

Barrier Formulation Starting from the same constitutive model we redefine the η basis with barriers

$$\eta(E) = -E^{max}\log((E^{max} - E)/E^{max}).$$

For consistency note this basis again satisfies both $\eta(0) = 0$ and $\eta'(0) = 1$ and so is valid for elasticity. This barrier also diverges at E^{max} and so ensures that E never exceeds measured strain limit bounds. It thus captures data-driven strain limits and at the same time

avoids extrapolation. Here, as the underlying model is anisotropic, warp, weft, and shearing directions all can impose the have different, measured E^{max} bounds enabling preservation of measured anisotropy for all strain limits.

Free model parameters $a_{\alpha\beta}$ and G_{12}, are then matched at $\nabla^2\psi(0,0,0)$ with the underlying model's fit; see Appendix for details. Note this formulation frees us from computing the sensitive (and potentially expensive) polynomials with large real-valued exponents. See Section 5.6.2 for experiments on the proposed model. Finally we note that, in contrast to the strategy we demonstrated in our isotropic case from the last section (where we worked directly on bounds w.r.t. the deformation gradient), here this second energy is symmetric for stretch and compression and so is not inequality-based.

5.5 Modeling Thickness

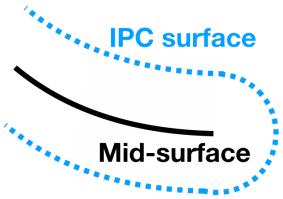
In its original form IPC maintains intersection-free paths for volumetric models by enforcing positivity of unsigned distances d_k, as an invariant between all non-adjacent and non-incident surface primitive pairs k Li et al. [2020b]. This is suitable for volumetric contact where this constraint permits arbitrarily close, but never intersecting, surfaces. For codimensional models, however, this constraint is no longer sufficient.

When the 3D deformation of thin materials is reduced to the deformation of a 2D surface or 1D curve, elasticity can be well-resolved on surfaces and curves, but contact can not. Neglecting to account for finite thickness in codimensional contact generates unacceptable artifacts (see e.g. Figure 5.16) and clearly fails to capture geometries formed by thin structure interactions (see e.g. Figures 5.9 and 5.18).

To model thickness in contact for codimensional objects we equip each midsurface element i with a finite thickness ξ_i. The distance constraint for primitive pairs k on the midsurface, formed between element primitives i and j, is then

$$d_k(x) > \xi_k = \frac{\xi_i + \xi_j}{2}.$$

Boundaries of codimensional materials are then approximated with a rounded cross-section, while for interaction between zero-thickness materials our distance constraints reduce to that of the original IPC constraint (e.g. for volume-to-volume contact). Finally for volume-to-codimensional contact, volumes can continue to maintain zero-thickness boundaries while interacting with finite thickness codimensional boundaries.



5.5.1 Solving IPC with Thickened Boundaries

For a numerically robust and efficient implementation, IPC applies squared distances (with appropriate rescalings) to compute with equivalent barrier functions. This change swaps the model's derived contact barrier $b(d_k(x), \hat{d})$ to an equivalent rescaled implmentation using $b(d_k^2(x), \hat{d}^2)$ Li et al. [2020b]. Here we directly apply our thickened contact barrier via squared distances

$$b_\xi(d_k(x), \hat{d}) = b(d_k^2(x) - \xi_k^2, 2\xi_k\hat{d} + \hat{d}^2),$$

so that contact forces correctly diverge at $d_k(x) = \xi_k$, and nonzero contact forces are only applied between mid-surface pairs closer than $\hat{d} + \xi_k$.

Here then the barrier function, gradient, and Hessian only need to be evaluated with the modified input distance, while distance gradient and Hessian remain unchanged as

$$\frac{d(d_k^2(x) - \xi_k^2)}{dx} = \frac{dd_k^2(x)}{dx}.$$

Evaluation of contact force magnitudes for computing friction forces as well as adaptive stiffness (κ) updates for contact barriers follow similarly.

Most constraint set computations then require simple and comparable simple modifications. For spatial hashing, bounding boxes of all mid-surface primitive i are extended by $\xi_i/2$ in all dimensions on left-bottom and top-right corners before locating voxels. This enables hash queries with $\hat{d}$ to remain unchanged. Next, for broad-phase contact pair

detection, query distances $\hat{d}$ are now updated to $\hat{d} + \xi_k$ to check for bounding box overlap. Or else, equivalently, bounding box primitives are extended as in the spatial hash. Finally, to accelerate continuous collision detection (CCD) queries, spatial hash construction also requires similar bounding box extension, while for its broad phase, applied gaps are ξ_k (rather than 0).

5.5.2 Challenges for CCD

While the above initial modifications for thickness are straightforward, finite thickness and codimensional DoF introduce new computational challenges for CCD queries. Here we analyze these challenges and develop a new CCD method to address them.

Standard-form IPC applies position updates in each iteration to obtain minimal separation distances of $s\times$ the current separation distances, d_k^{cur}, at time of CCD evaluations. Here $s \in [0.1, 1)$ is a conservative rescaling factor (generally set to $s = 0.2$) that allows CCD queries to avoid intersection, even when surfaces are very close Li et al. [2020b]. To similarly resolve finite thicknesses, our conservative distance bound is now $s(d_{cur} - \xi_k) + \xi_k$.

Concretely, the barrier evaluated parameters, $d_k^2(x) - \xi_k^2$, must always remain positive. In turn we require all CCD evaluations, for each displacement p, to provide us with sufficiently accurate times $t \in [0, 1]$ to satisfy positivity. If impact would occur for a pair along p, we require a time t so that the scaled displacement, tp, ensures distance remains greater than ξ_k and is as close as possible to the target separation distance $\xi_k + s(d_k^{cur} - \xi_k)$.

Doing this with thickness is much more challenging than in volumetric IPC where updated distances are targeting sd_{cur} (and must be greater than 0). With thin materials $d_{cur} - \xi_k$ can be as small as $10^{-8}m$, while in practice ξ_k is regularly at the scale of $10^{-4}m$ (e.g. thickness values of $\sim 3 \times 10^{-4}m$ for cloth and $\sim 1 \times 10^{-4}m$ for hairs are standard). Consider then that without thickness ($\xi_k = 0$) updating distances to sd_{cur} within 100% relative error are then acceptable. On the other hand for standard values of finite ξ_k, distance updates would require relative errors from the CCD evaluation of around $0.1\% = 10^{-8}m/10^{-4}m$ in order to avoid interpenetration.

Obtaining CCD evaluations to this accuracy is extremely challenging for available methods. As a starting example we tested the floating point CCD solver from Li et al. [2020b], requesting a $0.2(d_{cur} - \xi_k) + \xi_k$ minimal separation for two challenging codimensional examples with thickness (see Figures 5.20 and 5.21) with $\xi_k > 0$. Here the CCD solver returns $t = 0$ time of impact (toi) erroneously even if we remove the conservative scaling factor (i.e. set $s = 0$) altogether (Section 5.6.8). Note that in doing so this error stops simulation progress altogether.

5.5.3 CCD Lower Bounding

Next we derive a useful lower bound value for CCD queries that is numerically robust and can be efficiently evaluated in floating point. This lower bound provides a conservative, guaranteed safe step size and also a clear measure to test a CCD query's validity: any CCD-type evaluation with smaller toi has clearly failed. In Section 5.5.4 we then apply this bound to derive a simple, effective and numerically accurate explicit CCD solver that is robust and efficient for progress even in the challenging CCD evaluations we require for thin material simulations.

Here, without loss of generality, we will focus on the edge-edge case between edge pairs (x_0, x_1) and (x_2, x_3) with corresponding displacements p_0, p_1, p_2 and p_3. The distance function between arbitrary points parameterized respectively by γ and β on each edge at any time t is then

$$\begin{aligned} f(t,\gamma,\beta) &= ||d(t,\gamma,\beta)||, \text{with} \\ d(t,\gamma,\beta) &= (1-\gamma)x_0(t) + \gamma x_1(t) - ((1-\beta)x_2(t) + \beta x_3(t)), \\ x_i(t) &= x_i(0) + tp_i, \text{and} \\ t,\gamma,\beta &\in [0,1]. \end{aligned} \tag{5.11}$$

A CCD evaluation then seeks the smallest positive real t satisfying

$$f_1(t) = \min_{\gamma,\beta} f(t,\gamma,\beta) = 0.$$

If such t exists we call it t_a. Parameters $(\gamma_a, \beta_a) = \text{argmin}_{\gamma,\beta} f(t_a, \gamma, \beta)$ in turn give the corresponding points colliding at time t_a on each both edges.

We can then express t_a as

$$t_a = \frac{f(0, \gamma_a, \beta_a)}{||(1-\gamma_a)p_0 + \gamma_a p_1 - ((1-\beta_a)p_2 + \beta_a p_3)||}.$$

Challenges to CCD evaluations then occur in determining γ_a and β_a, when degeneracies and numerical error make floating-point methods both prone to false positives and false negatives.

We can however, directly and accurately perform a distance query w.r.t. the primitives at start (t=0) of evaluation, to find the distance, $f_1(0)$, that certainly satisfies $f_1(0) \leq f(0, \gamma_a, \beta_a)$. Then triangle inequality gives $||(1-\gamma_a)p_0 + \gamma_a p_1 - ((1-\beta_a)p_2 + \beta_a p_3)|| \leq ||(1-\gamma_a)p_0 + \gamma_a p_1|| + ||(1-\beta_a)p_2 + \beta_a p_3||$, and since $\gamma_a, \beta_a \in [0, 1]$ we get $||(1-\gamma_a)p_0 + \gamma_a p_1|| \leq \max(||p_0||, ||p_1||)$ and $||(1-\beta_a)p_2 + \beta_a p_3|| \leq \max(||p_2||, ||p_3||)$. Put together we then have the practical and directly computable bound on toi

$$t_a \geq \frac{f_1(0)}{\max(||p_0||, ||p_1||) + \max(||p_2||, ||p_3||)}. \tag{5.12}$$

We then note that even when there is no smallest positive time satisfying $f_1(t) = 0$ (and so no impact on the interval), our bound remains well-defined as a conservative step size.

We next observe that, perhaps surprisingly, state-of-the-art floating-point CCD solvers can and will return toi results smaller than our lower bound, and so are clearly in error (Section 5.6.8).

Finally, to improve our bound, we observe that it holds independently of the choice of the reference frame. Thus we can further tighten the bound on each CCD query independently by picking frames that reduce the norm of displacement vectors p_i. For example by subtracting each p_i by the average $\frac{1}{4}\sum_i p_i$.

5.5.4 Additive CCD

We now apply our lower bound to build a new CCD algorithm that iteratively updates and adds our lower bound over successive conservative steps towards toi. The resulting

additive CCD (ACCD) method then robustly solves for a bounded toi, monotonically approaching each CCD solution, while only requiring explicit calls to evaluate distances between updated primitive positions.

At the start of each CCD query, to initialize the ACCD algorithm (see Algorithm 4), we first center the collision stencil's displacement at origin to reduce our bound's denominator, $l_p = \max(||p_0||, ||p_1||) + \max(||p_2||, ||p_3||)$, and so increase the step size we can safely take. If there is no relative motion ($l_p = 0$), we of course simply return no collision and so a full unit step is valid. We then compute the requested minimal separation $g = s(\sqrt{d_{sqr}} - \xi)$ to the offset surface based on the current squared distance d_{sqr} and the scaling factor s. For this we use a formula that is more robust to cancellation error; see lines 8-9 in Algorithm 4.

Starting with a most conservative time of impact $t = 0$ (line 10) we create a *local* scratch-pad copy of nodal positions, x_i, and initialize the current lower bound step, t_l, with Equation 5.12 (line 11).

We then enter our iterative refinement loop (lines 12-21) to monotonically improve our toi estimate t. At each iteration, we update our local copy of nodal positions x_i with the current step t_l (lines 13-14). If this new position achieves our target distance to the offset surface (becomes smaller than g) we have converged and the previous t is the time of impact that brings distance just up to g (line 17). If not, we update our toi estimate by adding the current t_l to t (line 18). Note that we always add our first lower bound step to t (line 16) as it is guaranteed to not bring distance closer than g. If our toi is now larger than t_c, the current minimum first time of impact (or can be simply set to 1), we can return no collision (lines 19-20). Otherwise we compute a new local, lower bound, t_l, from the updated configuration (line 21) and begin the next iteration.

ACCD thus provides an exceedingly simple-to-implement, numerically robust CCD evaluation. It requires only explicit calls to distance evaluations and so no numerically challenged root-finding operations. In turn, ACCD is able to support very small thickness offsetting and also controllable accuracy and so flexible tuning for performance vs accuracy

Algorithm 4 Additive CCD

procedure ADDITIVECCD(x_i, p_i, ξ, s, t_c, t)
  $\bar{p} \leftarrow \sum_i p_i/4$
  for i in $\{0, 1, 2, 3\}$ **do**
    $p_i \leftarrow p_i - \bar{p}$
  $l_p \leftarrow \max(||p_0||, ||p_1||) + \max(||p_2||, ||p_3||)$
  if $l_p = 0$ **then**
    return false
  $d_{sqr} \leftarrow$ computeSquaredDistance(x_0, x_1, x_2, x_3)
  $g \leftarrow s(d_{sqr} - \xi^2)/(\sqrt{d_{sqr}} + \xi)$
  $t \leftarrow 0$
  $t_l \leftarrow (1 - s)(d_{sqr} - \xi^2)/((\sqrt{d_{sqr}} + \xi)l_p)$
  while true **do**
    for i in $\{0, 1, 2, 3\}$ **do**
      $x_i \leftarrow x_i + t_l p_i$
    $d_{sqr} \leftarrow$ computeSquaredDistance(x_0, x_1, x_2, x_3)
    if $t > 0$ and $(d_{sqr} - \xi^2)/(\sqrt{d_{sqr}} + \xi) < g$ **then**
      break
    $t \leftarrow t + t_l$
    if $t > t_c$ **then**
      return false
    $t_l \leftarrow 0.9(d_{sqr} - \xi^2)/((\sqrt{d_{sqr}} + \xi)l_p)$
  return true

trade-offs in standard CCD applications. In Section 5.6.8 we compare ACCD with state-of-the-art CCD solvers based on exact arithmetics and conservative strategies and show that ACCD succeeds in all cases with the best timing performance while other methods frequently fail.

Note that our stopping criteria requires $s > 0$ to provide finite termination, meaning that ACCD is never aimed at computing the exact toi, which is also not really needed by contact simulation methods. In our simulation we use $s = 0.1$ for every example.

5.6 Evaluation

We implement our methods in C++, applying Eigen for linear algebra Guennebaud et al. [2010] and CHOLMOD Chen et al. [2008], compiled with Intel MKL LAPACK and BLAS as our linear solver. Details of our computed derivatives and algebraic simplifications for numerical robustness and efficiency are detailed in our appendix. All evaluations are executed on either an Intel 16-Core i9-9900K CPU 3.6GHz $\times$ (32GB memory), an Intel Core i7-8700K CPU @ 3.7GHz $\times$ 12 (64GB memory), or an Intel Core i7-9700K CPU @ 3.6GHz $\times$ 8 (32GB memory) as detailed per experiments below.

Line Search As discussed above, CCD-based filtering is critical in line-search for collision. For strain-limiting we must now additionally ensure line search will not violate any imposed strain limits. In the isotropic case, although 2×2 SVD has a closed form solution, it does not provide a polynomial equation for easy feasible step size computation. Therefore, in line search, after we obtain an interpenetration-free step size, we simply half step sizes whenever we detect a strain limit violation during backtracking until we find a step size satisfying both strain limit and decreases energy. In practice, because our strain-limit barriers are included in the Hessian and gradient computation, we find that Newton search directions are effective at avoiding strain-limits and, in cases when needed, at most 3 bisections for strain-limiting are required. We apply the same strategy here for the anisotropic case. However, we do note that here in our the anisotropic case, it would

be reasonable to formulate quadratic equations amenable to directly computing largest feasible strain-limit satisfying step sizes. We leave this for future work.

Experiments Below we begin with a study evidencing membrane locking behaviors for standard cloth materials and simulation meshes (Section 5.6.1). We then demonstrate our method's ability to strictly satisfy strain-limits while fully coupling all physical forces in Section 5.6.2. To our knowledge our method is the first to enable strict satisfaction of strain-limits and also is the first to fully couple strain-limiting, elasticity and contact forces. To verify this we next consider comparison against prior methods. As discussed in Section 5.2 existing methods in strain-limiting generate artifacts including locking, jittering, and interpenetration. These issues result from two sources: 1) inaccuracies from applied splitting models and 2) inability to satisfy strain-limits in computation. Here, for the first time, we first separately analyze the problems that are created by splitting models (Sec. 5.6.3), and then in Sec. 5.6.4 consider the artifacts and inaccuracies (generated by state-of-the art cloth code) that are then jointly resulting from both splitting errors and inability to enforce requested strain-limits. In Section 5.6.5 and 5.6.8 we then analyze our thickness modeling comparing both with existing cloth simulation codes and prior methods for CCD evaluation. Finally with all pieces in place, we consider our method applied to prior cloth simulation benchmarks (Section 5.6.6); and stress tests demonstrating new challenges for thin materials (Section 5.6.7) and fully coupled systems of arbitrary codimension with consistent thickness modeling (Section 5.6.9).

5.6.1 Cloth Material and Membrane Locking

Here we first examine the impact of membrane locking on real-world materials and strain-limiting's ability to mitigate them. In Penava et al. [2014], density, thickness, and directionally dependent membrane strain-stress curves for multiple varieties of cloth are measured and validated. However, directly applying these real-world cloth parameters for simulation with standard-resolution meshes can produce severe membrane locking.

This is unavoidable unless exceedingly high and so often impractically large mesh sizes are used. Here we examine this locking behavior, first examining our IPC model without strain-limiting. We also note that such locking behaviors are independent of algorithm and, for example, are also easy to demonstrate in ARCSim Narain et al. [2012] with their real-world captured material parameters Wang et al. [2011]; see Figure 5.6d.

We start by considering the behavior of a simple $1m$-by-$1m$ unstructured mesh dropped on a fixed sphere and ground plane (with friction of $\mu = 0.4$ for all). We apply the measured cotton cloth density ($472.6kg/m^3$) and thickness ($0.318mm$) from Penava et al. [2014] and then consider varying membrane stiffness while keeping bending Young's modulus at $0.8MPa$ and Poisson's ratios for both membrane and bending to the measured value for warp direction as 0.243. Specifically Penava et al. [2014], find membrane Young's moduli ranging from $0.8MPa$ to $32.6MPa$ for varying in-plane directions.

For our example, we then find that even when applying an isotropic membrane model, with the smallest determined membrane stiffness ($0.8MPa$), we observe severe locking artifacts (see e.g. Figure 5.2a) where bending is artificially stiffened. If we next lower the smallest measured membrane stiffness by $0.1\times$, we then see that artificial bending artifacts are largely eliminated but we still obtain sharp creasing artifacts forced by the dominating membrane energy (see e.g. Figure 5.2b). Next, if we try an even smaller $0.01\times$ scaling of membrane stiffness, observable membrane locking effects are now gone, but as expected the resulting material is much too stretched and so not even close to the desired material behavior (Figure 5.2c). This simple example nicely demonstrates the challenges of membrane locking when simulating stiff real-world cloth materials. We then note that these artifacts are only exacerbated in more challenging simulations with, for example, moving boundary conditions and tight contact.

Next we apply a strict strain-limiting, here via our constitutive model, to constrain strain within the elasticity range measured by Penava et al. [2014]. The broadest range in all directions for cotton allows a stretch factor up to 6.08%. Here, applying this bound, even with $0.01\times$ scaling of membrane stiffness, we now regain a simulation free from

membrane locking and, since restricted to measured strain limits, also free of unnatural stretching artifacts (Figure 5.2d).

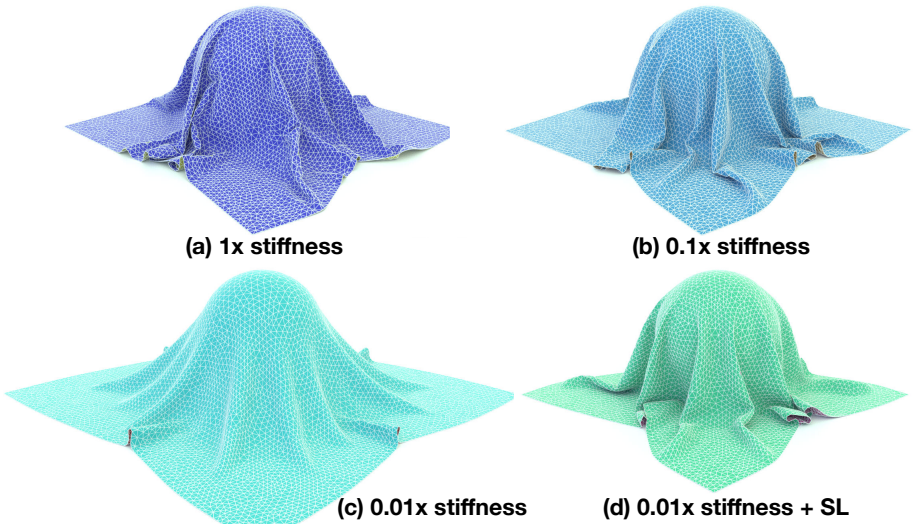


Figure 5.2: **Membrane Locking.** A cloth on sphere simulation (8K-node) with (a) $1\times$ stiff membrane of cotton material ($0.8MPa$) suffers from membrane locking where bending behavior is artificially stiffened and creased. Reducing membrane stiffness to (b) $0.1\times$ that of measured cotton values removes artificial bending stiffness but we still observe overly sharp edge creasing. Next an even softer membrane at (c), $0.01\times$ cotton's, then avoids observable membrane locking artifacts altogether, but results in nonphysical stretching inappropriate for cotton. However, (d) applying a softer $0.01\times$ membrane together with our constitutive strain limiting model to enforce measured cotton strain limits prevents membrane locking while obtaining natural stretch limits for the material.

5.6.2 Exact Strain Limits

Above we have demonstrated the well-known importance and impact of applying strain-limiting for simulating cloth materials. Here we investigate our method's ability to tightly and accurately enforce strain limits. As we will demonstrate this holds both for extreme (e.g. at 0.1%) and anisotropic limits.

Accuracy across decreasing strain limits To confirm accuracy, controllability and robustness of our constitutive strain limiting we apply increasingly severe strain limits of 1% and then 0.1% on both the sphere drape example from above and a standard membrane locking test of a two-corner pinned cloth Chen et al. [2019], Jin et al. [2017]. Here, see Figure 5.3, we observe stable simulation output with stretch on all triangles satisfying the prescribed limit constraints. Closer examination of the draped sphere tests in Figure 5.3

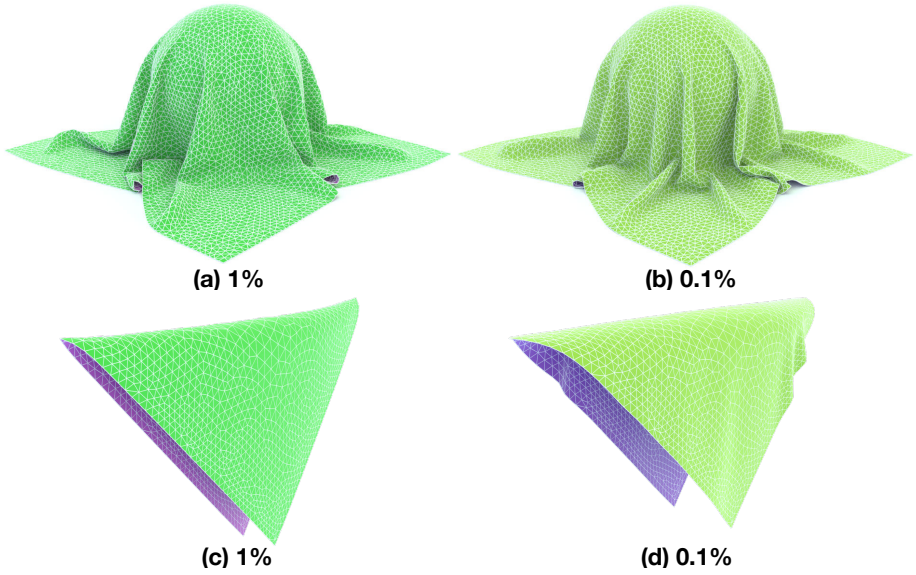


Figure 5.3: **Extreme Strain Limits.** Our constitutive strain limiting model stays robust and guarantees constraint satisfaction even with extreme (and even nonphysical) strain limit constraints (1% in (a, c) and 0.1% in (b, d)). Here the mesh only contains 8K (a, b) or 2K (c, d) nodes. Not in (d) that as strain-limits tighten beyond those physically plausible and so more constraints become active w.r.t. to DoF count, a small amount of membrane locking artifact becomes apparent; see our discussion below.

also confirms that both are free from artificial bending stiffness. For contrast, compare this result with the artificial stiffening in Figure 5.2a, where the maximum stretch exceeds 4.5% despite applying the measured membrane stiffness directly. Here, as we are applying our strain-limits *unilaterally*, this agrees with Jin et al.'s [2017] argument that unilateral strain-limits can allow compression to avoid membrane locking better than bilateral enforcement

– at least to a certain degree.

However, even unilateral enforcement is not a perfect panacea if limits are very tight. For an extreme example as we push strain-limits to a very small 0.1% strain limit we can finally observe some slight but distinct sharp edge-creasing artifacts in Figure 5.3b and similarly locking in Figure 5.3d. Although such an extreme limit is unlikely and generally not encountered for cloth, this experiment does highlight an important point: if most unilateral stain-limit constraints are active then they behave similalrly to the bilateral case. Thus, as discussed in Chen et al. [2019], when most strain limits are at the bound, active constraint numbers can again exceed the number of simulated DOFs, and so locking can once again be encountered. Here we focus on formulating a controllable and robust constitutive model for strain limiting. We then hope to further enable increasing the range of locking-free configurations by exploring alternate discretization for the strain constraints as in Chen et al. [2019] but note that this is only be possible when the underlying method, as proposed here, can accurately guarantee constraint satisfaction.

Anisotropic strain limiting For anisotropy, the story is similar. In Figure 5.4 we demonstrate a cloth hang (two top corners fixed) and our same cloth drape on sphere example. Again applying measured stiffness values, here from Clyde [2017], generates membrane locking (Figure 5.4a). Then scaling down membrane stiffness $0.1\times$ and $0.01\times$ while preserving measured strain limits again provides the expected improved results (Figure 5.4b,c) while in Figure 5.4 (again with $0.01\times$ membrane stiffness), we see both har strain limits and anisotropic wrinkling behavior.

5.6.3 Comparison with Splitting Models

Existing strain-limiting methods introduce errors from two primary sources. First, at the modeling level, they split strain-limiting from the resolution of elasticity. Second, irrespective of the splitting model applied, algorithms applied to solve them are inaccurate and often unable to satisfy even moderate requested strain-limits. All prior methods then

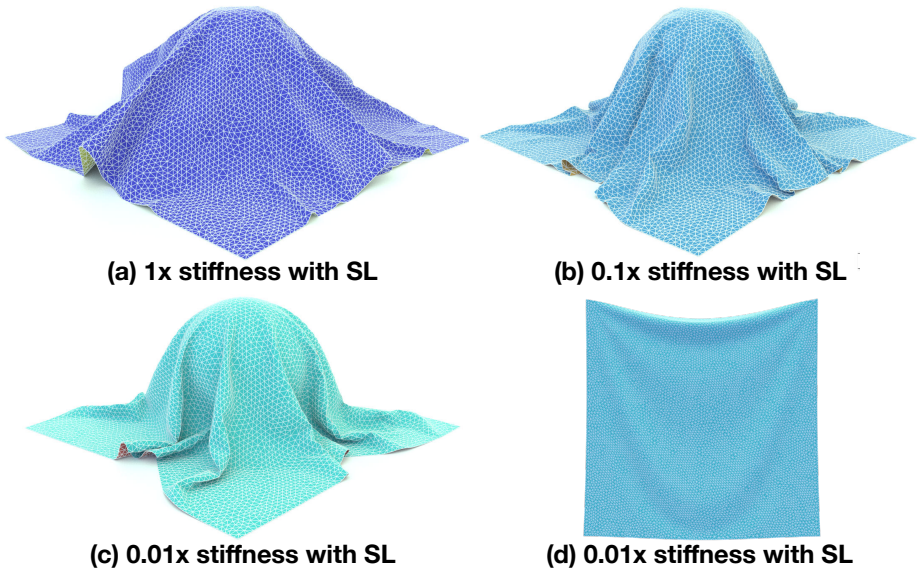


Figure 5.4: **Anisotropic strain limiting.** Cloth on sphere drape with varying scales of real-world membrane stiffness (a-c) and a cloth hang (d) all simulated with our anisotropic strain limiting model at $h = 0.04s$. Here the shown frames are the final static rest shape. We can see that the draped cloth exhibits differing wrinkle patterns when compared to isotropy.

introduce errors from both of these sources and so it has remained unclear what problems originate from each. Here, we first separately analyze the problems that are created by splitting models, by solving each step of the splitting to tight accuracy. This allows us to show that the splitting itself introduces errors that are unavoidable irrespective of how accurately the constraints could be enforced. Then, in the next section, we will examine solution accuracy and see that errors in the strain-limit solve itself then produce inconsistent and so often uncontrollable results for simulation.

To examine splitting errors we follow the standard strain-limiting, splitting strategy and so divide each time step solve into two sequential steps. The first solves a predictor step that includes all forces for the whole system, *with the exception of the strain-limit*, to obtain an intermediate configuration $\hat{x}$. Next, the second step *projects* the predictor to satisfy the strain-limit. To do so we minimize our constitutive strain limiting energy summed with the

mass-weighted L^2 distance from the final timestep solution to $\hat{x}$.

In the simplest case we can consider the effect of this splitting when there are no contact forces. We start with a pinned and pulled square cloth. We fix its two top corners and apply weights to pull its two bottom corners downwards; see Figure 5.5a and b. For comparison, without strain limiting, the cloth is stretched well over $10\times$ (not shown) while with our constitutive strain limits, strains are restricted to the measured elasticity range, and, in this fully coupled solution, we see the expected vertically aligned wrinkles from stretching at the two bottom pulled corners; see Figure 5.5a. On the other hand in Figure 5.5b, we see that splitting strain limiting from the elasticity solve produces obvious biasing artifacts at the two bottom corners where the wrinkles are aligned in non-physical directions. These errors from the decoupled forces are then increasingly severe as time step increases (here we show with $h = 0.04$s).

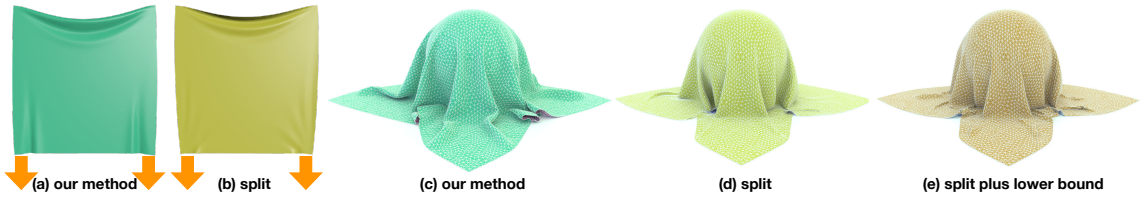


Figure 5.5: **Comparison of our fully coupled strain limiting solution with traditional splitting methods.** Examples with both a cloth pinned at top and pulled on bottom (a, b), and a cloth draped on sphere (c-e) can both demonstrate the difference between fully coupled solutions (a and c) and the artifacts generated by splitting strain limit solves from elastodynamics (b,d and e). Here we show the final configuration of each simulation as it comes to a rest. We can see that with splitting, large errors in elasticity (membrane and bending) can generate incorrect wrinkling directions (b), severe compression artifacts (d), or even numerical softening (d and e).

Next, to consider splitting errors when subject to contact, we again consider the cloth on sphere example. Here we apply a neo-Hookean membrane elasticity to help the splitting method avoid possible triangle degeneracies. Note that with our fully coupled model we

do not require the neo-hookean model as triangle degeneracies are prevented by point-triangle constraints between neighbors. Here in Figure 5.5d we then see that splitting with contact now suffers from severe compression artifacts, which again comes from splitting between membrane and bending energies. For comparison, consider our fully coupled strain-limited solution in Figure 5.5c. In turn the error in the split solution suggests that an additional lower bound on strain (e.g. as applied in ARCSim) might be helpful to avoid these compression errors. However, if we additionally add a strain-limit lower bound (e.g. at 0.7) then the splitting method is indeed now free from severe compression artifacts, but now due to the errors in membrane and bending the cloth still remains unnaturally flat against the floor; see Figure 5.5e.

Of course as with all time-step splitting methods, errors can be reduced by applying increasingly smaller time step sizes. Here, for example, we find that visual errors between the split model and our fully coupled solve disappear at $h = 0.004s$. However, as expected, the resulting large and often unacceptable increase in compute time wipes out any expected gains from splitting (we will see this theme again in the next sections' comparisons with existing cloth codes). Likewise the necessary decrease in time step size then varies with example and scene so that robust, controllable time-stepping with splitting, does not appear possible.

5.6.4 Strain Limiting Comparisons

As we just demonstrated how severe the artifact a two-step splitting solve can result in, what if contact and strain limiting projection is further split that results in 3 substeps, and that the strain limit cannot always be satisfied exactly? This leads to our comparison to ARCSim Narain et al. [2012].

Without contact, we consider the cloth drape example again. Instead of making the two bottom corners heavier, this time we place the initial configuration of the cloth in xz plane, so the swing motion will make the drape challenging in certain frames. In Figure 5.6(a, b) we show the frames at $4s$ as the cloth nearly becomes static. We use the default

strain limit, $[0.9, 1.1]$, in ARCSim for the comparison, and in ARCSim we use their default cotton material. Comparing to our method, even with this wide range of strain limit the result obtained by ARCSim under $h = 0.04s$ has obvious over-stretched triangles near the fixed corners, which forms a larger arc. In Figure 5.7 we can see that ARCSim violate the strain limit a lot both in stretch and compression. Even if we decrease the time step size to $h = 0.01s$ to make the problem easier, ARCSim still cannot exactly satisfy the strain limit constraints, although the violation does get smaller. With $h = 0.001s$, ARCSim finally can closely satisfy the constraints, but it needs $87min$ for the $4s$ simulation sequence, which is more than $10\times$ that of our method running with $h = 0.04s$. In addition, it is unclear how time step size needs to be decreased in different simulations to ensure ARCSim results to satisfy the strain limit, while only our method guarantees constraint satisfaction agnostic to simulation settings.

In fact for the augmented Lagrangian strain limiting solver in ARCSim source code there is an iteration cap set to 100. Even if we increase this number up to 1000, the strain limit satisfaction still cannot be guaranteed.

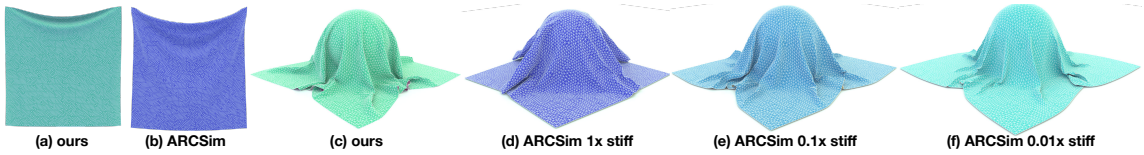


Figure 5.6: **Strain limiting comparisons.** A cloth drape example where initially the cloth is in xz plane (a, b), and a cloth on sphere example (c-f) showing comparison of our fully coupled contact-aware constitutive strain limiting method to ARCSim's augmented Lagrangian based method under $h = 0.04s$. From the shown frames when the scenes become (nearly) static, we can see ARCSim cannot guarantee strain limit satisfaction (b,e,f), and as we decrease membrane stiffness in (e,f) to avoid membrane locking issue in (d), the over- compression or stretching artifact gets more obvious.

With contact, we consider the 8K-node cloth on sphere example again. For ARCSim,

h=0.04s if not specified	Max s_{max} over time steps	Avg s_{max} over time steps	Min s_{min} over time steps	Avg s_{min} over time steps	Total runtime (min)
Cloth drape ours	1.0988	1.0884	0.9801	0.9824	6.5
Cloth drape ARCSim	2.1281	1.4836	0.7686	0.8760	2.0
Cloth drape ARCSim h=0.01s	1.8959	1.3197	0.8127	0.8879	8.0
Cloth drape ARCSim h=0.001s	1.1110	1.1009	0.8959	0.9093	87.0
Cloth on sphere ours	1.0617	1.0535	0.9548	0.9652	7.5
Cloth on sphere ARCSim	1.4176	1.0875	0.2397	0.9220	1.0
Cloth on sphere ARCSim 0.1x stiff	1.3796	1.1587	0.0568	0.8599	1.5
Cloth on sphere ARCSim 0.01x stiff	1.7944	1.3216	0.6714	0.8445	1.0

Figure 5.7: **Strain limiting comparisons.** Strain and runtime statistics of the cloth drape example and the cloth on sphere example in comparison with ARCSim. Here $s_{\max}$ and $s_{\min}$ are the maximum and minimum strain of all triangles in a certain time step. As time step size decreases, ARCSim results satisfy strain limits better, while as membrane stiffness decreaes, which is necessary to avoid membrane locking, ARCSim results violates strain limits more. On the other hand, our method always guarantee strain limit satisfaction agnostic to simulation settings.

we observed very similar membrane locking issue as in Figure 5.2a when we use ARCSim's default cotton material (Figure 5.6d). But with this stiff membrane, the strain limit constraints are well satisfied except for several challenging time steps in the beginning. When we multiply the stretching stiffness by $0.1\times$ (Figure 5.6e), the membrane locking issue gets much better, but now there are more triangles violating the strain limit in the whole simulation. This shows that the performance of ARCSim's strain limiting method is very sensitive to membrane stiffness. Now if we multiply the stiffness by $0.01\times$ as usually used in our method (e.g. Figure 5.2d), the result of ARCSim shown in Figure 5.6f violates the strain limits even more, and it is suffering from the extreme compression issue similar to our splitting ablation study in Figure 5.5e. In addition, with $0.01\times$ membrane stiffness, the animation generated by ARCSim is suffering from jittering issue, which is because

their contact and strain limiting are further split into separate projection steps, making the constraints hard to be simultaneously satisfied. This issue has been discussed in their limitation section.

5.6.5 Thickness Modeling

Here we conduct an experiment on a cloth stack example to demonstrate the thickness modeling challenge in existing cloth simulator, and then show the controllability of our method on intricately modeling the thickness of objects with robustness guarantees (more examples in Section 5.6.9).

We construct a scene with 10 8K-node cloth in xz-plane falling simultaneously at different height and orientation onto a square board (Figure 5.8a). This should form a cloth pile where correctly modeling the thickness of each cloth matters a lot on obtaining a reasonable pile height. This seemingly simple example already exposes challenges with existing shell simulators in handling geometric thickness with contact. Here we compare against two widely used state-of-the-art cloth simulators, ARCSim Narain et al. [2012] and Argus Li et al. [2018a], of which the source codes are available online.

ARCSim ARCSim Narain et al. [2012] computes[2] collision response applying Harmon et al's [2008] non-rigid impact zones and standard "repulsion thickness" parameters in order to both model cloth thickness and to help stabilize collision processing constraints. ARCSim's default thickness setting is $5mm$ and can be changed by users. This parameter is then associated in code with additional thickness parameters directly applied for collision projection, analysis and contact force computations. For the default $5mm$ thickness setting, even decreasing time step size down to $0.001s$, ARCSim still reports failures for collision handling and we see severe artifact in the simulation results; see Figure 5.8b. We then try with $10mm$ thickness, and here ARCSim succeeds in resolving contacts but still generates large artifacts in geometry and dynamics; see Figure 5.8c. We did not push ARCSim further

[2]We use the most recent ARCSim v0.3.1 in our testing.

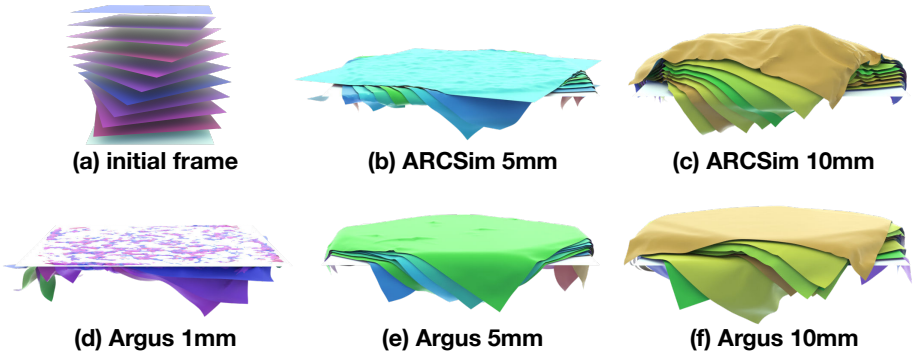


Figure 5.8: **Cloth stack comparisons.** Ten 8K-node square cloth are dropped onto a square board with friction $\mu = 0.1$ (a). Here ARCSim Narain et al. [2012] and Argus Li et al. [2018a] are tested with different cloth thickness settings at $h = 0.001s$. ARCSim fails to resolve the contact at $5mm$ (b), and at $10mm$ there is no interpenetration but the dynamics is off (c). Argus can successfully simulate cloth with $5mm$ and $10mm$ thickness (e,f), but at $1mm$, it fails to resolve contact. See Figure 5.9 1st row for our results.

with even smaller time step sizes as generating this simulation sequence with $h = 0.001s$ already makes it excessively slow for computation when compared to our method (details later).

Argus Argus Li et al. [2018a] combined ARCSim and a new contact solver Daviet et al. [2011] to provide improved contact and friction processing for shell simulations. It also applies and exposes the same thickness parameter from the underlying ArcSim code to users. Using Argus to simulate our cloth on stack example at $h = 0.001s$ with default $5mm$ thickness (Figure 5.8e) and the thicker $10mm$ (Figure 5.8f) both work, and we can observe the height difference of the pile is also captured. But as we decrease thickness towards thinner cloth materials sizes. e.g. setting thickness to $1mm$ (which is our default $\hat{d}$ for IPC cloth simulations), Argus produces severe artifacts; see Figure 5.8d. Again, for all 3 different thickness settings, Argus' timing is worse than our method (details later).

In summary what we see here is that for state-of-the-art methods, they require small enough time step size to avoid failures. With smaller thickness, the required time step size

is probably even smaller, which makes the simulation expensive and it is unclear what time step size will work for a specific thickness. This makes parameter setting challenging as not only time step size needs to be tuned to ensure success, but there are also the damping parameters to be tuned to obtain realistic dynamics.

For our method, by setting our $\hat{d}$ (the distance we start exerting contact forces) to different values, i.e. $1mm$, $5mm$, and $10mm$ here, we successfully model different thickness for the cloth on stack example (Figure 5.9 1st row) without any artifact. For this 1s animation ARCSim and Argus both took around 260 minutes to finish under $h = 0.001s$, while our method can apply $h = 0.04s$ and only took around 100 minutes for $1mm$ and $5mm$, and 156 minutes for $10mm$, which are much faster. For $\hat{d} = 10mm$, the maximum number of active contact constraints (not including friction) reaches 4.7M due to the large activation distance. Although this shows that our method is robust even with a large number of contacts, it would still be interesting to explore constraint culling strategies where uncolliding, e.g. 2-ring neighbor contact pairs, can be appropriately ignored to gain more efficiency in large $\hat{d}$ situations.

Since our contact model exert larger forces as distance gets smaller, it naturally provides a constitutive model between the distance (thickness) and the contact force in the normal direction where indentation effect can also be naturally captured if we drop a elastic ball onto our cloth stack (Figure 5.9 middle and bottom rows). In the middle row we show the frame when the ball gets the most compression, and we see that it creates a valley on the cloth stack and also generate wrinkles in the $10mm$ case. For the frame where the scene becomes nearly static (Figure 5.9 last row), we still clearly see the different height of the piles. This simulation also shows that IPC naturally couples elastic body and shells together, which is further explored later in Section 5.6.9.

Another way of modeling thickness in our method is to set an offset to the constraint definition. This is different from using different $\hat{d}$ that for a nonzero offset ξ, objects are guaranteed to be separated from each other at ξ, and contact force starts to exert when object distance is below $\xi + \hat{d}$ and goes to infinity when distance is ξ. For simulating

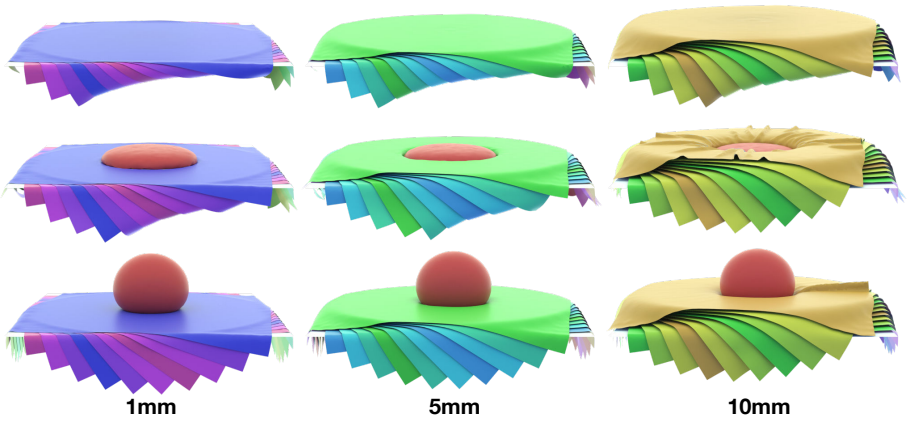


Figure 5.9: **Sphere on cloth stack.** Ten 8K-node square cloth are dropped onto a square board with friction $\mu = 0.1$ (1st row). For different IPC $\hat{d}$ per column, our method is able to provide controllable thickness modeling for cloth (the stack heights are visually distinguishable). Then a soft elastic sphere ($E = 10KPa$) is dropped onto to the cloth stack and compressed to the extreme (2nd row), where with $10mm$ thick cloth there are also intricate wrinkling behaviors. After the scenes become static (3rd row), we can see the thickness effect is still robustly maintained.

codimensional objects, these 2 mechanisms could be combined together, where larger $\hat{d}$ introduces more elastic response while a nonzero ξ provides guarantees for minimum thickness/volume even under extreme compression (e.g. Figure 5.16). For scenes where consistently modeling thickness matters (e.g. Figure 5.16, 5.19, 5.20, 5.21, 5.22, 5.23), we set ξ to the true thickness value of the object or slightly smaller, and then set $\hat{d}$ around ξ depending on how much elastic response we need.

5.6.6 Cloth Simulation Benchmark

We next, in this section, confirm our method's performance on challenging benchmark tests from prior work.

Cloth on rotating sphere Here we run a classic example in Bridson et al. [2002] where after dropping a square cloth onto a sphere on the ground, the sphere starts rotating. With friction on the sphere and ground at $\mu = 0.4$ and between cloth at $\mu = 0.01$, we see that the cloth follows the sphere to rotate together with friction accurately resolved, and fine wrinkling details are captured without any locking issue or stretchy artifacts (Figure 5.10) thanks to our robust and accurate strain limiting model. With higher resolution (246K nodes) and smaller bending stiffness, the wrinkles become more intricate with higher frequency.

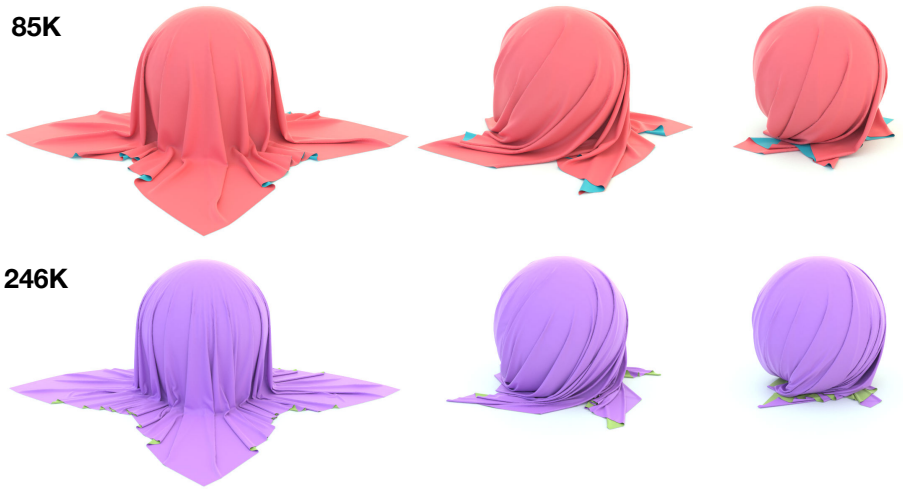


Figure 5.10: **Cloth on rotating sphere.** A square cloth (85K-node 1st row, 246K-node 2nd row) is dropped onto a sphere and a ground with friction ($\mu = 0.4$ for both). Then the sphere starts to rotate, and the cloth follows. $8 \times 10^4 Pa$ and $8 \times 10^3 Pa$ bending Young's modulus are used for the $85K$- and $246K$-node cloth respectively to produce intricate wrinkling behaviors with different frequency. Here from left to right we show the 25th (right before rotation), 50th, and 75th frame.

Funnel Inspired by Tang et al. [2018] and Harmon et al. [2008], we run a funnel example where multiple square cloth are dropped onto a funnel with friction, and then pushed through by a scripted collision object (Figure 5.11). Here we set $\mu = 0.4$ to hold the cloth in earlier frames and to make the problem more challenging, we use a 4-node tetrahedron,

with sharp corners and a very different resolution, as the scripted collision object. Our method robustly simulate this challenging scene with multiple layers of cloth and the sharp moving obstacle without jittering nor intersection artifacts.

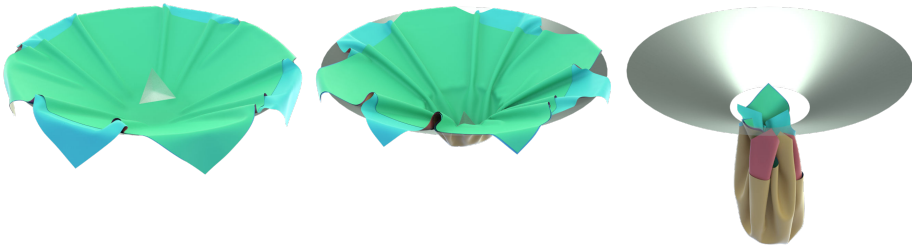


Figure 5.11: **Funnel.** Three 26K-node square cloth are dropped onto a funnel with friction ($\mu = 0.4$). Then a tetrahedron moving collision object pushes the cloth through the funnel.

Ribbon knot Here we look at cloth contact under large stress from a classic example in Harmon et al. [2009]. 2 ribbons are initially placed intertwind with each other and then stretched to form a knot (Figure 5.12). In our simulation the cloth contains 100K nodes ($10\times$ that of the original example) and is nearly stretched to its strain limit, resulting in extreme stress in the center and form a much smaller knot than in Harmon et al. [2009]. We also applied a $\mu = 0.02$ friction here to realistically capture the intricate dynamics. Our simulation has no interpenetration or jittering issue.

Garments Garment is an important application of cloth simulation where 2 main challenges lie in multilayered garment and garment simulation with moving characters. To test the robustness of our method we first simulate the yellow dress (15K nodes) produced by FoldSketch Li et al. [2018c] with knife pleats where cloth are folded and then placed on top of each other and stitched, which is very challenging to simulate. We start the simulation from flat patterns staged around the 13K-node mannequin (Figure 5.13a), and by efficiently time stepping at $h = 0.04s$ with the spring forces to stitch seams together, we quickly obtain the static equilibrium of the draped garment with all knife pleats simulated with correct layering free from any intersection (Figure 5.13b).

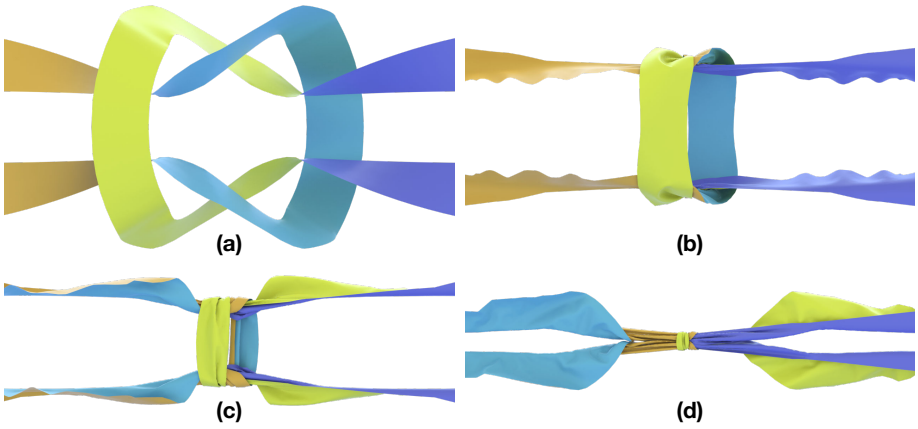


Figure 5.12: **Ribbon knot.** A classic example in ACM Harmon et al. [2009] with 2 ribbons dragged apart to form a knot. Here our 2 ribbons contain 100K nodes in total, and we drag them starting from rest shape (a) to nearly reach the strain limit (b-d), where the friction coefficient is $\mu = 0.02$. Our results have no interpenetration or jittering artifacts.

Then we create a multilayer skirt (30K nodes) using Sensitive Couture Umetani et al. [2011], and obtain a realistic drape in the same way (Figure 5.13c). Next we will use this looser garment to simulate with a character animation sequence that has large movement. To setup the simulation we download a skeleton sequence from Adobe Mixamo[3], generate skinning animation with our mannequin, and then use the animated mannequin sequence as kinetic object to simulate with garment. In Figure 5.13(d,e,f) we show 3 frames of the simulation featuring the kicking motion. With these large movement, our method stays stable and efficient while capturing intricate garment details.

The multilayer garment simulation with the input character animation is efficiently simulated at $2min$ per $h = 0.04s$ time step, where it would be interesting to explore whether replacing the stiff stitch springs with change of variable strategy will further accelerate our convergence.

[3] https://www.mixamo.com/

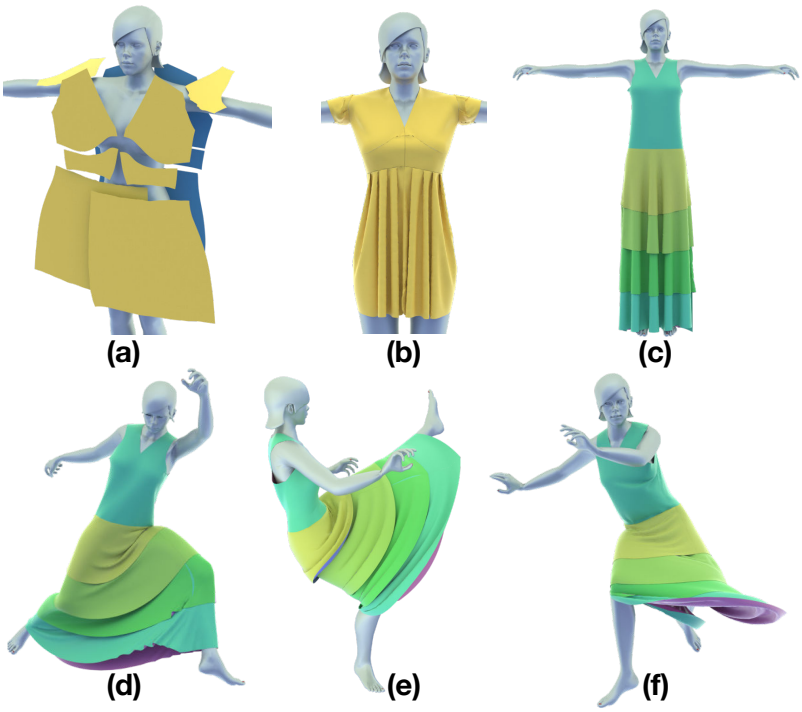


Figure 5.13: **Garments.** A yellow dress with knife pleats produced by FoldSketch Li et al. [2018c] is staged (a) and then draped on a mannequin to static equilibrium (b) with our method that stitches the seams together. A multilayer skirt is draped in a same way (c), and then used to simulate with an input character animation sequence with large motion (d,e,f). As the mannequin kicking with her leg, intricate garment details are captured by our method.

5.6.7 Cloth Simulation Stress test

Now we perform stress tests on our method with challenging examples that require a high level of robustness and accuracy. To our knowledge we are the first to achieve these simulations without any artifacts and algorithmic parameter tuning.

Pulling cloth on needles When simulating cloth with sharp contacts, snagging artifact is a notorious issue for existing methods. Now we show that our method is snagging-free with sharp contact. We form a needle bed directly using codimensional segments and drop

a square cloth onto it (Figure 5.14). With a large time step size at $h = 0.02s$, we see that our cloth is safely becoming static on the bed of needles without any jittering. Then we pull the left side of the cloth with $1m/s$ speed moving Dirichlet boundary condition, and we see that the cloth can be pulled through on top of the bed of needles without snagging. In this example as we have extreme stretching with sharp contact, our adaptive strain limiting stiffness adjustment strategy is effective to make sure the simulation stays numerically feasible and efficient.

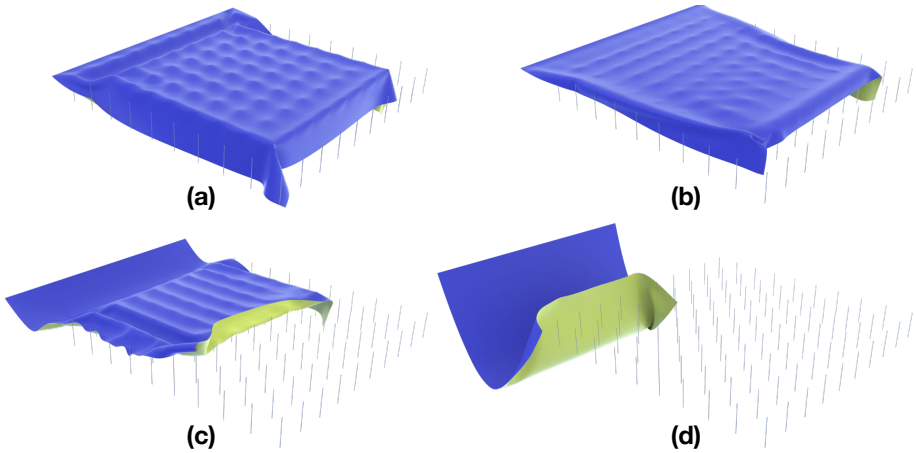


Figure 5.14: **Pulling cloth on needles.** A 26K-node square cloth is first dropped onto a bed of needles formed directly by codimensional segments (a) and then pulled to the left at $1m/s$ without snagging issue (b, c, d). Here from (a) to (d) we show the 50th (right before pulling), 55th, 65th, and 85th frame.

Table cloth pull Pulling a table cloth out from under a set of objects with the objects staying on the table while not following the cloth is a classic magic trick[4]. To simulate it, requires an accurate friction model and a method that can robustly handle the large stresses and normal forces from pulling, as well as the contact with sharp edges. Here, we successfully simulate this trick with both heavy and stiff objects on table by pulling the cloth at high ($4m/s$) speed; Figure 5.15d. Similarly, we correctly capture the trick failing if we pull at too small a speed where, due to stiction, some objects follow the table cloth

[4] https://www.juggle.org/table-cloth-pull-trick/

pull and so fall off; and finnally confirm that with an even slower pull even more objects fall; see Figure 5.15b and c. Here IPC naturally resolves stiff elastic bodies with cloth contact by coupling all directly in an accuarte solve. In Section 5.6.9 we demonstrate more coupling simulations. This is also a nice example that demonstrates the robustness of our method on handling very different tessellations and stiffnesses (the cube and tetrahedron only have 8 and 4 nodes), and extreme boundary conditions that pulls the cloth at a high speed under large friction forces given by the heavy objects.

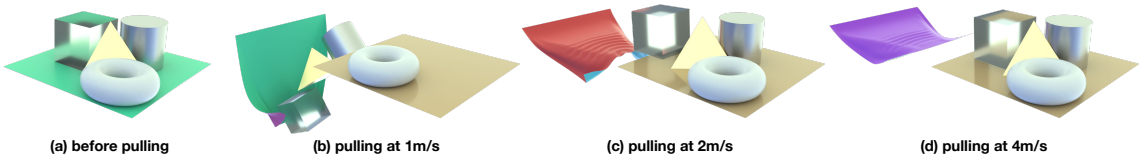


Figure 5.15: **Table cloth pull** A 26K-node square cloth is put on a table board with a cube, a tetrahedron, a cylinder and a torus solid ($\rho = 2000kg/m^3$, $E = 0.1GPa$) on it (a). The capability of our method on accurately resolving controllable friction behaviors is shown here by pulling the cloth leftwards with different speed (b-c). In (b) the pull is at $1m/s$ and nearly all objects follow the cloth due to static friction. As the pull gets faster at $2m/s$ in (c), only the cube is falling and the other objects stay on the table after some slight move. Finally in (d) pulling the cloth at $4m/s$ left all objects on the table because of the small duration of the action of dynamic friction. In (c) and (d) there are also elastic waves on the cloth right after it is detached from the objects and board.

Cloth cylinder twist Now we test cloth simulation under extreme stress. We made a $1m$-wide cylinder cloth with 88K nodes, and twist and move the two sides together at $72°/s$ and $5mm/s$ respectively without gravity and strain limiting (Figure 5.16). Right after twisting we get interesting folds patterns (Figure 5.16b), but due to the extreme stress and the codimensional shape representation the center part quickly becomes a tiny knot. Therefore, we add an offset of $1.5mm$ to the IPC to model an inelastic thickness for the cloth, and then the same frame looks more realistic with thickness supporting the structure

(Figure 5.16c). This is a nice example demonstrating the need for thickness modeling when simulating with codimensional shape representations, which will be further analyzed in Section 5.6.5. After $32.96s$ twisting, our cylinder cloth experience lots of buckling (Figure 5.16e) and our method stays robust with thickness behaviors consistentely modeled.

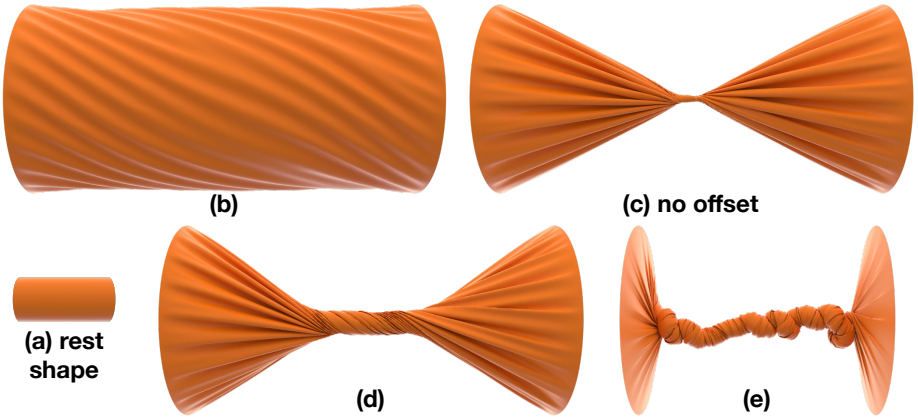


Figure 5.16: **Twisting cylinder.** A $1m$-wide 88K-node cylinder cloth is twisted and moved closer at $72°/s$ and $5mm/s$ respectively at both ends from rest shape without gravity (a). Right after twisting, intricate folds start to form (b). Here strain limiting is turned off to test simulation under extreme stress, which however makes the center part a tiny knot (c). By setting a $1.5mm$ IPC offset to model some inelastic thickness, the cloth looks more realistic at the same frame (d). After $32.96s$ twisting with the offset setting, we get lots of buckling behaviors (e) and our method stays robust with thickness behaviors consistentely modeled.

5.6.8 CCD Benchmark

Up to this point all our examples can be nicely simulated with the conservative CCD strategy in IPC Li et al. [2020b]. But when we start exploring more complex examples in the next Section 5.6.9 with much more degenerate contact pairs and also higher demand on CCD accuracy for inelastic thickness (constraint offset), we found that the CCD can simply return 0 toi, completely stalling our optimization incorrectly. Multiple alternative CCD methods also turned out failing. This triggered our innovation of ACCD, which stays robust

and accurate even for our complex examples where all alternative methods fail. Then by running all our examples in this chapter using ACCD we found that for easier cases ACCD still works nicely and can provide comparable or even faster timing performance. Here we provide some of our comparison statistics in Figure 5.17.

	EV's	MinSep	Root Parity	BSC	T-BSC	ACCD (ours)
Sprinkles	Tiny toi	Tiny toi	N/A	N/A	N/A	72.8
Noodles	Tiny toi	Tiny toi	N/A	N/A	N/A	98
Aleka	Tiny toi	Tiny toi	N/A	N/A	N/A	855.5
Braids	1012.8	Tiny toi	N/A	N/A	N/A	965.4
Card Shuffle	Tiny toi	Tiny toi	N/A	N/A	N/A	165.2
Needle bed	55.6	Tiny toi	Intersection	Runtime error	Infinite loop	43.4
Garment	262.3	Tiny toi	Intersection	Runtime error	Tiny toi	229.6
Cloth stack (1mm)	498.3	Tiny toi	448.5	Runtime error	Infinite loop	441.5
Cloth stack (5mm)	236.8	Tiny toi	191.9	Runtime error	Infinite loop	189.6
Cloth stack (10mm)	135	Tiny toi	124.4	Runtime error	Infinite loop	123.3

Figure 5.17: **CCD comparison statistics.** We provide the per Newton iteration total CCD querying time (excluding spatial hash) in milliseconds for different simulation examples when we use different CCD methods (EV's Li et al. [2020b], MinSep Harmon et al. [2011], Lu et al. [2019], Root Parity Brochu et al. [2012], BSC Tang et al. [2014], T-BSC Wang et al. [2015], and our ACCD). The first 5 examples are with inelastic thickness, which is not supported by Root Parity, BSC, and T-BSC methods. "Infinite Loop" means the method never reports "no collision" even when querying a configuration with tiny displacement that is interpenetration-free.

We tested minimal separation (MinSep) CCD Harmon et al. [2011], Lu et al. [2019] by wrapping it with the same conservative CCD strategy in IPC Li et al. [2020b] that the minimal separation is set to sd_{cur} where s is the slackness and d_{cur} is the current distance, and if the returned toi is smaller than 10^{-6} the query will be performed again without any

slackness ($s = 0$) to make it easier. For Root Parity Brochu et al. [2012], BSC Tang et al. [2014], and T-BSC Wang et al. [2015] that only return "has collision" or "no collision" without computing toi, we apply a monotonic bisection for each query until finding a step size without collision and return it after multiplying by $1 - s$. Since the bisection still cannot directly extend the methods to compute the time or step size that *first* brings the distance of a contact pair to ξ even if we combine it with distance computation, we do not test these 3 methods on our examples with inelastic thickness (the first 5 examples in Figure 5.17).

For IPC's conservative strategy with EV's CCD, except for the complex scenes, it works well with slightly slower timing than our ACCD. One interesting thing to note is that EV's CCD handles degenerate cases like parallel edge-edge via querying extra point-edge and point-point pairs, while our robust ACCD only needs to query the general pairs (e.g. point-triangle and edge-edge pairs for surface only scenes). On the other hand, MinSep always return tiny values, sometimes even 0, for all the listed examples. For examples without inelastic thickness (constraint offset), Root Parity performs well with nice efficiency but there are missed collisions leading to interpenetration in the needle bed and garment example; BSC always report runtime error with an internal message "Inflection points are not handled exactly in BSC"; and T-BSC returns tiny value for the garment example, and trapped in the loop for finding a step size without collision in all the other four examples.

5.6.9 General Shells, Rods, Particles, and Coupling

In this section we demonstrate IPC simulation across all codomains. IPC simulation is thus extended to includes the full spectrum from volumetric materials to codimensional shells, rods and even, if desired, particles. Because these domains then interact via IPC barriers we also are able to seamlessly and directly couple all such mesh-based models in the same common simulation. Here we then demonstrate the application of this coupling and further highlight the advantages and suitability of our consistent and controllable thickness modeling for this range of thin material interactions.

Card shuffle First, let's looks at stiff shells by modeling playing cards. Traditional card simulation examples mainly focus on accurately resolving the friction. Here we not only model friction, but also show that we can accurately capture the bending and thickness effects of the card by simulating card shuffling. We uniformly separate 54 poker cards into 2 piles and apply Dirichlet boundary condition on their sides to bend them in preparation for shuffling (Figure 5.18a). Then we release the Dirichlet boundary conditions from bottom up card-by-card alternatingly from the left and right pile (Figure 5.18b). The released cards quickly retain their rest shape, falling onto the ground, and form another shuffled pile without intersection throughout. After all cards are released we use 2 handles to make the pile align better (Figure 5.18c). By setting the IPC offset and $\hat{d}$ to $0.1mm$ and $0.2mm$ respectively to accurately resolving the thickness of every card, we ensure that the height of the final pile is visually the same with reality (Figure 5.18d).

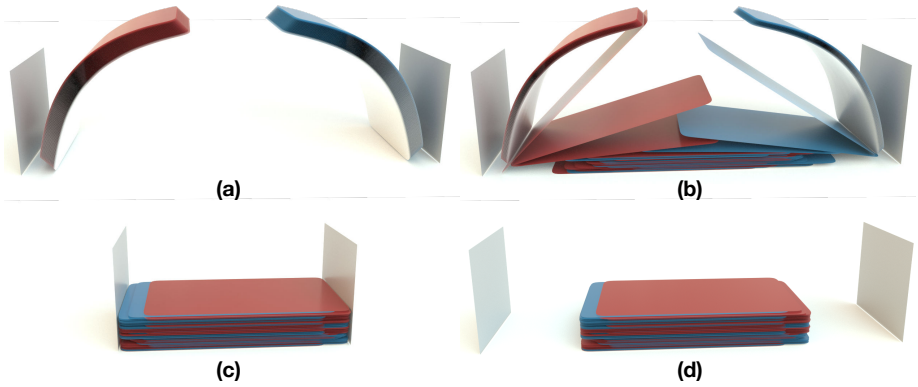


Figure 5.18: **Cards shuffling.** 54 poker cards are uniformly separated into 2 piles initially and bent (a), and then one-by-one released in turns from bottom to up to get shuffled (b). After all cards are released 2 handles are squashed to make the pile align better (c), and finally we obtain a nicely aligned shuffled cards pile.

Hairs Next we consider rod models. Here we test our method on two hair simulation examples. First, we show twisting of two hair clusters that forms braids and then the bottom handle is released so that the braids untie (Figure 5.19a-e). Then we test a typical hair simulation example in McAdams et al. [2009] where one end of a hair cluster is fixed, and

the other end falls onto another hair cluster with two ends fixed. Here we also added a sphere below the falling hair cluster to make the dynamics more interesting. Our method robustly simulate both scenes without any intersection at 2 to 3 minutes per $h = 0.01s$ time step, and accurately capture the intricate hair dynamics. Although hair thickness is nearly negligible, we still set an offset of $0.08mm$ (the thickness of a hair strand) and $\hat{d} = 0.1mm$ for IPC to make sure that thickness behaviors are always consistently captured even with large stress as in the braids example.

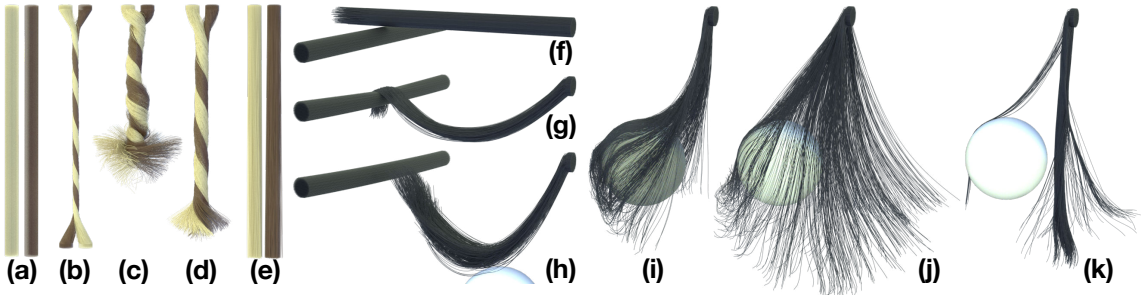


Figure 5.19: **Hairs.** Left: two hair clusters (each with 650 60-segment rods) are twisted to form braids and then the bottom end is released. Right: a typical hair simulation example in McAdams [2009] where a hair cluster with one end fixed is falling onto another hair cluster with both ends fixed (900 50-segment rods for each cluster). A sphere collision object is also added below to add more interesting dynamics.

Noodles Beyond hairs which are extremely thin, our contact-driven thickness modeling naturally allows us to also capture the contacting geometries of even thicker materials directly with discrete rods. Here we first show a simulation of noodles falling into a bowl (Figure 5.20). We setup the scene by arranging 625 vertically placed $40cm$-long noodles (each represented with a 200-segment discrete rod) into a xz-plane matrix separating each other by $3.3mm$ and place them on top of a $13.85cm$-wide fixed bowl (Figure 5.20a). We set IPC's offset and $\hat{d}$ to $1mm$ and $0.5mm$ respectively to model the noodles' thickness at $1.5mm$, which let us reconstruct the surface mesh with $1.5mm$-thick cylinders along

the rods for rendering. After dropping the noodles, they quickly fill up the bowl (Figure 5.20bc) thanks to our controllable and high-quality modeling of thickness. Our method robustly simulate this scene until static at around 23min per $h = 0.02s$ time step without any jittering artifacts. If we do the calculation, the total volume of the noodles is about $4.42 \times 10^{-4} m^3$, which is right below the hemisphere bowl's volume $6.96 \times 10^{-4} m^3$. These numbers further verified that our static equilibrium is consistent with reality.

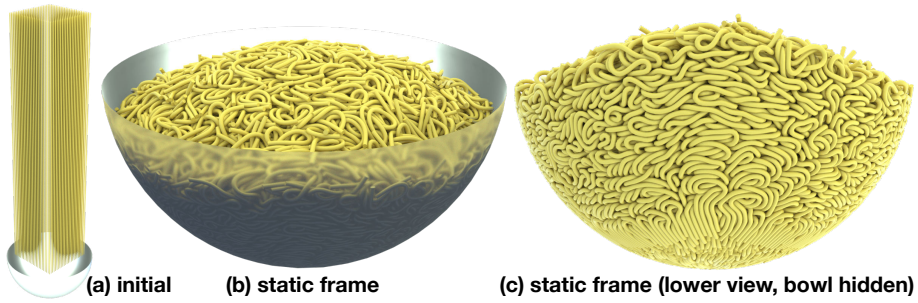


Figure 5.20: **Noodles.** 625 $40cm$-long noodles (each represented with a 200-segment discrete rod, separating each other by $3.3mm$ initially) are dropped into a $13.85cm$-wide bowl (fixed). Here we set IPC's offset and $\hat{d}$ to $1mm$ and $0.5mm$ respectively to model the noodles' thickness at $1.5mm$, which let us reconstruct the surface mesh with $1.5mm$-thick cylinders along the rods for rendering. The controllable and high-quality modeling of thickness here is also essential for the noodles to pile up and fill the bowl. Our method robustly simulate this scene until static at around 23min per $h = 0.02s$ time step without any jittering artifacts.

Sprinkles Now let's do something creative by representing each sprinkle using a single segment with its thickness modeled by IPC offset and $\hat{d}$. Here we place $25 \times 25 \times 40$ $6mm$-long and $2mm$-thick sprinkle separating each other by $3.3mm$ above a $13.8cm$-wide fixed bowl (Figure 5.21a). We set IPC's offset and $\hat{d}$ to $1.5mm$ and $0.5mm$ respectively to model the sprinkle's thickness, which let us reconstruct the surface mesh with 2mm-thick cylinders along the rods for rendering. Since sprinkle often does not deform, here we make offset $3\times$ large as $\hat{d}$ so that thickness behaviors are more inelastic. Then we drop

the sprinkles, and they quickly fill up the bowl (Figure 5.21bc). Our method robustly simulate this scene until static at around $6.5min$ per $h = 0.02s$ time step without any jittering artifacts. If we calculate the total volume of the sprinkles by summing up each capsule volume $25000(l\pi r^2 + \frac{4}{3}\pi r^3) \approx 5.76 \times 10^{-4}m^3$, and calculate the hemisphere bowl's volume $\frac{2}{3}\pi r_b^3 = 6.88 \times 10^{-4}m^3$, the close match of these numbers again nicely verified our simulation results.

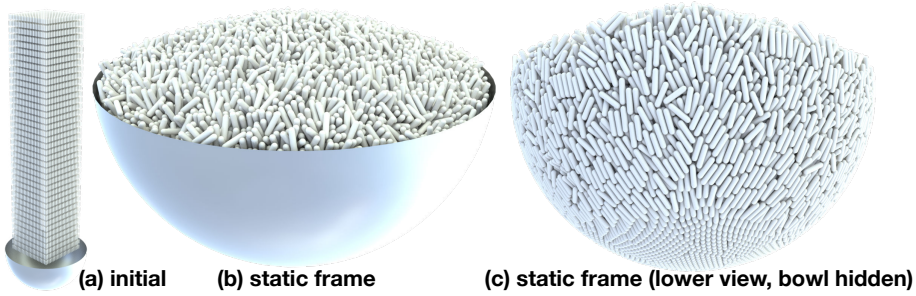


Figure 5.21: **Sprinkles.** 25K $6mm$-long and $2mm$-thick sprinkles (each represented with a single-segment discrete rod, separating each other by 3.3mm initially) are dropped into a 13.8cm-wide bowl (fixed). Here we set IPC's offset and $\hat{d}$ to 1.5mm and 0.5mm respectively to model the sprinkle's thickness, which let us reconstruct the surface mesh with 2mm-thick cylinders along the rods for rendering. Our method robustly simulate this scene until static at around 6.5min per $h = 0.02s$ time step without any jittering artifacts.

Sand on cloth Here we show that our method can even handle contacts between single particles, which has codimension 3. We arrange single particles into a $10 \times 10 \times 500$ cuboid separating each other by $4mm$ and let the high sand column fall onto a cloth with its four corners fixed (Figure 5.22). We compute the mass of each sand using $\rho = 1600kg/m^3$ and $V = 4/3\pi r^3 \approx 4.2 \times 10^{-9}m^3$. With IPC offset and $\hat{d}$ set to $2mm$ and $1mm$ respectively, the volume of each sand is consistently modeled during contact, which is essential to connecting every sand together through interaction and let them move as a flow. Here we rendered each sand as a $2mm$-wide sphere. This simulation highly resembles the Discrete Element Method (DEM) Cundall and Strack [1979], Yue et al. [2018] where near-rigid

bodies, or particles, are treated as perfectly rigid, and an interaction zone (analogously $\xi < d < \xi + \hat{d}$ in our situation) is coated outside each of them for contact forces to be calculated based on the depth, area, or volume of the intersections among these zones Jiang et al. [2020]. DEM is popular in geomechanics communities since its reasonable simplification ignores the deformation for the particles while accurately simulates the momentum and contact forces through contact laws. What is different here for our method is that we relate our contact forces directly to distances with a barrier function, which guarantees interpenetration-free while for DEM the geometric contact accuracy would need to be controled by carefully choosing time step size and contact model stiffness.

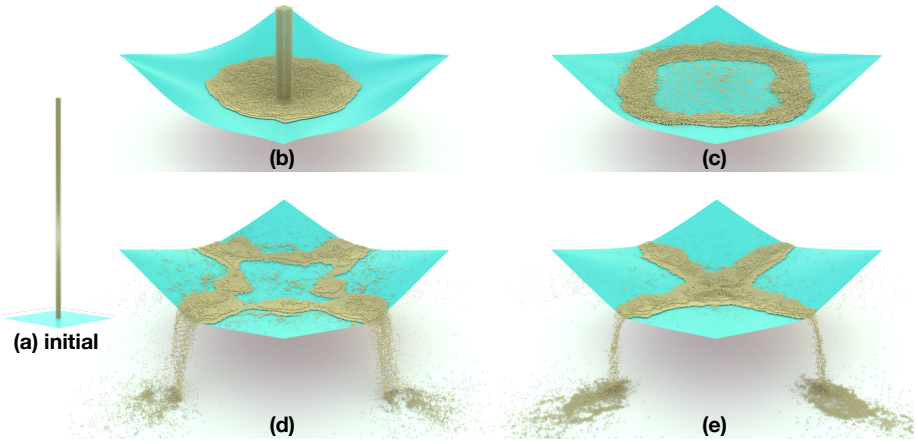


Figure 5.22: **Sand on cloth.** A $10 \times 10 \times 500$ sand column is falling onto a square cloth with its four corners fixed. Here each sand is represented with a single isolated FEM node, with IPC offset and $\hat{d}$ set to $2mm$ and $1mm$ respectively, the sand-sand contacts are accurately captured which is essential for the volumetric effects in this scene.

All in Finally, we show a scene that couples objects of every codimension together. Here an Armadillo volume, a square cloth, and a sand column formed by isolated particles are falling onto a rod net (Figure 5.23a), where we release them in sequence at $0s$, $2s$, and $4s$. The rod net is constructed by alternating the y-coordinates of the rods in x direction up and down by $1mm$, so that adjacent x-direction rods are on the different sides of the y-direction rods. We can see that this net structure provides interesting interactions with

the Armadillo (Figure 5.23b). Then after the cloth falls and covers the Armadillo on the net, we see cloth-rod contacts are also accurately captured (Figure 5.23c). When sand colume hit the cloth, the cloth slightly deforms and the sand particles flow into the 2 valleys of cloth supported by the Armadillo and the rod net (Figure 5.23d,e). Here IPC offset and $\hat{d}$ are $0.5mm$ and $1mm$ everywhere.

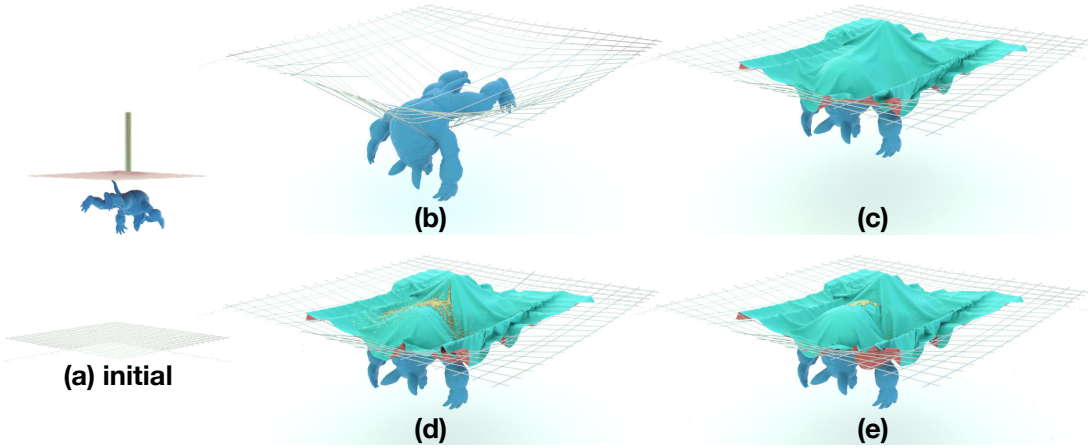


Figure 5.23: **All-in.** We simulate objects of all codimensions coupled together in a scene with an Armadillo volume, a square cloth, and a sand column formed by isolated particles released in sequence to fall onto a rod net.

5.7 Summary

In summary, this chapter extends the incremental potential contact (IPC) framework from volumetric deformables Li et al. [2020b] to thickness and strain limit-aware modeling of codimensional/mixed-dimensional structures. Independent from the elasticity model or the time step size, the resulting framework guarantees out-of-the-box intersection-free output with a strict satisfaction of strain limits (down to 0.1%) and an accurate capture of geometrically meaningful thickness. Furthermore, the proposed additive CCD (ACCD) approach shows extremely stable behaviours for challenging first-time-of-impact tasks. It can also be used as an extremely easy-to-implement and efficient alternative CCD algorithm

for other graphics applications.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this dissertaion, we at the first time proposed a robust and accurate self-contact handling framework with friction and fully implicit nonlinear elastodynamics that can guarantee interpenetration-free trajectories, supporting a full spectrum of codimension-0,1,2,3 objects modeled with geometric thickness and fully coupled exact strain limiting. Specifically, this includes

- a decomposed optimization time integrator for large-step elastodynamics,
- a rigorous formulation on a globally consistent contact constraint definition between triangulated surfaces,
- a scalable barrier method for inequality constrained optimization,
- a mollified friction model with controllable accuracy and consistent transition between static and dynamic modes,
- a semi-implicit friction discretization with a potential energy that can be seamlessly incorporated into optimization time integration framework,

- a constitutive strain limiting method that guarantees constraint satsifaction and can be solved simultaneously with accurate contact and elastodynamics,
- a consistent thickness model for simulation with codimensional shape representation, and
- a simple but robust and efficient additive continuous collision detection approach that outperforms all existing alternatives.

Based on solid theoretical foundations, our methods provide controllable trade-off between efficiency and accuracy for different application scenarios. We also performed extensive experiments and analyses to thoroughly test and verify all the proposed features. Our framework frees designers and engineers from extensive algorithmic parameter tuning as when traditional methods are used, enables brand new phenomena to be easily simulated, and can be applied in a wide range of applications from other communities like mechanical engineering, robotics design, etc.

6.2 Future Work

We consider four main aspects that futher extend our work and spread its impact into more research and industry communities.

More Physical Models With our explorations in elastoplasticity and fracture simulation Fang et al. [2019], Wolper et al. [2019, 2020], and solid-fluid coupling Fang et al. [2020], it would be meaningful to incorporate all these different physics into our IPC based optimization time integration framework. Thus multiphsics phenomena like a bullet fired towards a glass window, origami structrue design for quadroptor wings, a robot swims under water, etc, could also be accurately simulated with robostness and predictiveness to further benefits more complex application scenarios. The challenges would lie in how to couple fracture dynamics, fluid dynamics, and plasticity, into our optimization time

integration framework, where most of these effects have their own evolving rules and do not have a well-defined potential form. The idea of our semi-implicit temporal discretization of friction is certainly promising to be considered, and a recently advanced Hyperplasticity theory Houlsby and Puzrin [2007] is also of interest.

Easier Calibration to Reality In Li et al. [2020d] we found that matching simulation results to real experiments requires several iterations of calibration on the physical parameters of the system. Although we matched every physical setting of the quadroptor that can be refered to like dimensions and material stiffness, the damping coefficient of the material is still hard to match. This issue will be magnified for even more complex systems where there will be more physical factors that are hard to quantify. Given that our methods are differentiable, it would be great if a physical parameter calibration system could be designed to combine computer vision techniques to directly extract information from video footage of real experiments, and then apply some optimization or searching algorithm to automatically figure out the best physical parameters that matches reality. This will be very meaningful for simulation-in-the-loop material or robotics design applications.

Better Efficiency Then it would be critical to improve the timing of our methods. Currently we still require several minutes per frame to obtain high quality results with 50K to 100K DoFs. Although this timing matches or is even faster than existing accuracy targetted methods, and it is acceptable for engineering and robotics applications, it still needs much faster speed to scale up to even higher resolutions, and to be applied in a wider range of scenarios like gaming and virtual reality (VR) where real-time or interactive-rate responses are required. Therefore, it would be meaningful to explore implementation improvements on advanced computing architectures like what we explored on GPU for MPM Wang et al. [2020b]. Devising new mathematical or numerical approaches like higher-order basis with a smaller number of DoFs, and hierarchical optimization schemes Wang et al. [2020a] would also be helpful. In addition, applying machine learning techniques to accelerate certain steps of the simulation by assuming a specific application target would also be

interesting.

Differentiable Simulation Framework Finally, it would be impactful to design a differentiable simulation framework based on our optimization time integrators. Although our methods are differentiable by construction, computing sensitivity information in an efficient and robust way is still challenging especially for large scale dynamic problems. In addition, traditional simulation-in-the-loop design optimizations often only rely on 1st-order sensitivity information, which can make the convergence slow and would require algorithmic parameter tuning to obtain reasonable results. It would be interesting to explore 2nd-order optimization methods for design optimizations with consistent and scalable approximation to the 2nd-order sensitivity information. Providing this differentiable simulation framework that are directly applicable to various design tasks would greatly benefit many research and industry fields.

Appendix A: Smoothing

Let $f(x)$ be a function we wish to smooth. It is C^1 continuous everywhere except at $x = a$ where it is only C^0 continuous. Applying a function $g(x)$ that is C^1 continuous everywhere with $g(a) = 0$, we have a smoothed function $f(x)g(x)$ that is C^1 continuous everywhere. For $x \neq a$ we have $(f(x)g(x))' = f'(x)g(x) + f(x)g'(x)$ is C^0 continuous everywhere. At $x = a$, the left and right derivatives of $f(x)g(x)$ are then

$$\begin{aligned} \lim_{x \to a^-} (f(x)g(x))' &= \lim_{x \to a^-} f'(x)g(a) + f(a)g'(a), \\ \lim_{x \to a^+} (f(x)g(x))' &= \lim_{x \to a^+} f'(x)g(a) + f(a)g'(a). \end{aligned} \tag{1}$$

As $f(x)$ is C^0 continuous at $x = a$, $\lim_{x \to a^-} f'(x)$ and $\lim_{x \to a^+} f'(x)$ are both bounded. Then with $g(a) = 0$ we then have left and right derivatives of $f(x)g(x)$ both equal $f(a)g'(a)$ at $x = a$,

$$\lim_{x \to a^-} (f(x)g(x))' = \lim_{x \to a^+} (f(x)g(x))' = f(a)g'(a) \tag{2}$$

Thus $(f(x)g(x))'$ is likewise C^0 continuous at $x = a$ and $f(x)g(x)$ is correspondingly C^1 continuous.

Appendix B: Barrier Continuity and Testing

The continuity of our C^2 barrier function is confirmed when $d < \hat{d}$, as $\frac{\partial b}{\partial d} = (\hat{d} - d)(2 \ln \frac{d}{\hat{d}} - \frac{\hat{d}}{d} + 1)$ and $\frac{\partial^2 b}{\partial d^2} = -2 \ln \frac{d}{\hat{d}} + (\hat{d} - d)\frac{\hat{d}+3d}{d^2}$ both vanish as $d \to \hat{d}$. Thus the left and right derivatives of b at $d = \hat{d}$ are both equal at zero up to 2nd order.

Our motivation for applying a C^2 clamped barrier rather than a less nonlinear, but still smooth, C^1 barrier is to provide 2nd-order derivatives suitable for our Newton-type solver. Thus it is genralluy better to have a continuous Hessian for improved convergence Nocedal and Wright [2006]. Nevertheless, here we also provide an ablation study applying all both C^0 ($b = -\ln(d/\hat{d})$) and C^1 ($b = (d - \hat{d}) \ln(d/\hat{d})$), along with our final choice of our C^2 ($b = -(d - \hat{d})^2 \ln(d/\hat{d})$) barrier in IPC on a set of examples in Figure 1.

Examples	C0 # iters, t (s) / time	C1 # iters, t (s) / time	C2 (IPC) # iters, t (s) / time
mat on knives	5.51, 2.16	5.59, 1.43	5.47, 1.40
2 mats40x40 fall	NC	26.83, 10.64	26.17, 10.48
octocat on knives	NC	9.29, 4.21	7.73, 3.76
sphere1K roller	NC	47.64, 9.41	45.22, 9.02
mat40x40 twist (10s)	7.74, 1.93	8.04, 2.07	7.56, 1.89
sphere1K pin-cushion	12.82, 1.27	9.22, 0.75	8.93, 0.69
rods630 twist (10s)	6.65, 1.05	3.05, 0.38	3.07, 0.40

Figure 1: **Ablation study on barrier functions with different continuity.** NC means not converging after 10000 PN iterations when solving a certain time step.

From the results we see that for the C^0 barrier, optimization can be non-convergent. Here this can be a result if the local minima is right inside the clamped region of the barrier, where the gradient does not change smoothly – here intermediate values may not be found to balance terms so decrease the total gradient. While, in comparison to C^1, our C^2 barrier is generally 5%–10% faster due to the continuity of the Hessian – this is reflected in less iteration counts.

Appendix C: CFL-inspired Culling of CCD

As IPC requires performing CCD for every Newton iteration, CCD clearly becomes a bottleneck. Therefore we propose a novel CFL-inspired culling strategy to accelerate CCD.

Recall that our culled constraint set contains all surface primitive pairs with distances smaller than $\hat{d}$. Thus all remaining primitives outside the culled constraint set have farther

distances lower-bounded by $\hat{d}$. We place all vertices participating in these primitives in a set $\mathscr{F}$. At each iteration i with a search direction p^i, we find the index ℓ of the simulation node with the largest component in p (i.e., its effective search-direction "velocity") among all remaining node pairs, $\ell = \text{argmax}_{\kappa \in \mathscr{F}} ||p^i_k||$. We then compute a conservative bound on the largest-feasible step size for surface primitives not in $\hat{\mathscr{C}}$ as

$$\alpha_{\mathscr{F}} = \frac{\hat{d}}{2||p^i_\ell||}. \tag{3}$$

We then apply conservative CCD (see below) to the remaining primitive pairs in $\hat{\mathscr{C}}(x^i)$, obtaining a large feasible step size, $\alpha_{\hat{\mathscr{C}}}$, for that set. Then $\alpha_0 = \min(\alpha_{\mathscr{F}}, \alpha_{\hat{\mathscr{C}}})$ is our maximum *feasible* step size for iteration i. We then apply α_0 as starting step size to bootstrap backtracking for Newton iteration i and so ensure decrease with feasibility.

Computing large, feasible step sizes in this way effectively reduces CCD cost by two-orders of magnitude. However, for scenes that contain high speed motions, $\alpha_{\mathscr{F}}$ can be overly conservative (smaller than needed) which then would increase iteration counts and so cause energy related computations to increase overall cost unnecessarily.

Thus we adapt by balancing between applying full CCD for all candidate pairs provided by spatial hash and applying our CFL-like strategy. Here we will designate the exact feasible bound computed from applying CCD to all primitive pairs in the spatial hash as $\alpha_{\mathscr{S}}$.

In each step we first compute $\alpha_{\mathscr{F}}$ and $\alpha_{\hat{\mathscr{C}}}$ – they are both efficient and inexpensive to find. Next we observe that the culled bound $\alpha_{\hat{\mathscr{C}}}$ is often very close to the exact bound $\alpha_{\mathscr{S}}$, while computing this exact bound is generally one-third of the total timing cost in a single iteration. We thus proceed by computing $\alpha_{\mathscr{S}}$ if $\alpha_{\mathscr{F}} < \frac{1}{2}\alpha_{\hat{\mathscr{C}}}$. Otherwise we apply our CFL-type bound and apply $\alpha_0 = \min(\alpha_{\mathscr{F}}, \alpha_{\hat{\mathscr{C}}})$.

We thus avoid overly restrictive step sizes when $\alpha_{\mathscr{F}}$ and $\alpha_{\hat{\mathscr{C}}}$ are already quite close – meaning their minimum should also be quite close to $\alpha_{\mathscr{S}}$. In practice we observe that our CFL-like assisted CCD culling strategy provides a 50% speed-up for all CCD related costs with nearly the same iteration counts. Ovxerall this results in an average 10% speedup for IPC; see Figure 2.

Examples	full CCD # iters, t (s) / time step	CFL combined (IPC) # iters, t (s) / time step	full CCD CCD related Timing (s)	CFL combined (IPC) CCD related Timing (s)
mat on knives	5.34, 1.66	5.47, 1.40	97.50	47.25
2 mats40x40 fall	23.22, 10.21	26.17, 10.48	273.38	162.10
octocat on knives	6.99, 3.92	7.73, 3.76	206.61	129.93
sphere1K roller	49.38, 9.89	45.22, 9.02	482.13	394.01
mat40x40 twist (10s)	7.37, 2.13	7.56, 1.89	147.60	62.91
sphere1K pin-cushion	8.35, 0.77	8.93, 0.69	37.25	19.61
rods630 twist (10s)	3.04, 0.47	3.07, 0.40	23.89	9.13

Figure 2: **CCD strategies ablation.**

Appendix D: Conservative CCD

CCD is generally applied to compute a time of impact corresponding to a step size that would bring distances between primitives to 0. In our barrier setting this "largest feasible step size" needs to be made conservative by backing away from an exact zero distance. A simple strategy would be a conservative rescaling with a factor $c \in (0, 1)$; e.g., by starting the line search at 0.5 or 0.9 of the total step along the descent direction. However, for the CCD computations rounding error can be severe for the tiny contacting distances we allow and so even small naive scaling factors (e.g., 0.1) can allow unacceptable intersections in such cases while in others is a much too conservative bound unnecessarily slowing convergence.

Rather than directly finding and then conservatively rescaling a CCD-computed step length that takes us to intersection we directly compute via CCD a step size along p^i that will bring primitives to a distance of $(1 - c)d_c > 0$. Here d_c is the current distance between the primitives. Standard CCD libraries [1] directly provide this option, e.g., exposed as an η parameter. In turn this modifies coefficients of the polynomial equations solved

[1] we use https://github.com/evouga/collisiondetection

during CCD, and effectively reduces numerical rounding errors that cause issues for direct scaling. In our implementation we apply $c = 0.8$ between mesh primitives, $c = 0.9$ for mesh-to-plane, and similarly $c = 0.8$ for computing the large feasible step size to avoid element inversion when barrier elasticity energies, e.g., neo-Hookean, are applied.

Finally, to ensure we remain intersection- and inversion- free at each step. We apply a post-step check whenever nodal positions are displaced (e.g. line search and initial movements of boundary conditions and collision objects. These include the edge-triangle intersection check filtered by our spatial hash and a volume check for every tetrahedral element. In exceedingly rare cases when an edge-triangle intersection or negative volume are detected, we half the final step size bound.

Appendix E: Equality Constraints for Moving Collision Objects and Time-Varying Boundary Conditions

In many scenarios, e.g., scripting animations, kinematic objects, and engineering tasks, scripted kinematic collision objects (CO) and/or moving positional (Dirichlet-like) boundary conditions (BC) are required. Contact algorithms generally handle these functionalities by either directly prescribing and updating nodal positions of CO/BC nodes at start of each time step, or interpolating them in substeps across a step. Remainder of simulation DOF are then solved w.r.t. the prescribed nodes being fixed at "current" positions. However, such strategies are extremely limiting, with simulations generally restricted to small time step sizes, speeds, and/or deformations as the BO and/or CO become faster and more challenging. For example, directly prescribing CO or BC nodes can often generate tunneling artifacts, and for moving BC, simulations can often fail simply by inverting elements when barrier energies like neo-Hookean energy are applied. To address these issues we formulate scripted dynamic BC and CO as equality constraints in IPC. This simultaneously ensures that intersections (and inversions) are avoided even while scripted motions are applied at large time step.

We start by computing the prescribed nodal positions at the beginning of each time step, and then apply CCD and element inversion detection to find a large feasible step size towards the prescribed position from the current one (see above).

If it is safe to apply them fully without causing intersection or inversion, we simply update prescribed nodal positions, solve the remainder of simulation DOF and are done. However, if our feasible step size is smaller than taking a full step (we use a criterion of 0.999), we first move the prescribed nodes as far as we can conservatively (see Section 6.2) and then add new *equality constraints* for each of these prescribed nodes provided by either CO or BC scripting.

As in our treatment of intersection-free *inequality* constraints, we build an unconstrained form for each by applying an augmented Lagrangian. For every such prescribed node, we add a Lagrangian and a penalty potential. We initialize each Lagrange multiplier to 0 and each penalty stiffness to 10^6. Concretely, we add an energy

$$-\sqrt{m_k}\lambda_{\mathrm{A},k}^T(x_k-\hat{x}_k)+\frac{\kappa_{\mathrm{A}}}{2}m_k||x_k-\hat{x}_k||^2 \tag{4}$$

to our barrier-form incremental potential for each prescribed node k with corresponding destination $\hat{x}_k$ in the current time step, if *any* of the prescribed nodes could not reach its destination during our start-of-time-step test.

We measure satisfaction of BC/CO node constraints at each Newton iteration i by calculating their total in-time-step progress as

$$\eta_{\mathrm{A}}=1-\sqrt{\frac{\sum_k||\hat{x}_k^{t+1}-x_k^i||^2}{\sum_k||\hat{x}_k^{t+1}-x_k^t||^2}}, \tag{5}$$

where $\hat{x}^{t+1}$ is the prescribed BC/CO positions for time step $t+1$. Then, whenever a current iterate is both close to satisfying stationarity, via the stopping criteria, and progress, with $\eta_{\mathrm{A}}<0.999$, we either increase the BC/CO penalty stiffness or else update the Lagrange multipliers, via the first-order update rule, see Algorithm 1 below. Finally, whenever $\eta_{\mathrm{A}}\geq 0.999$, we fix all prescribed nodes at current position and solve for remaining DOF in order to avoid unnecessary slow down of convergence due to added stiffness.

Algorithm 5 Augmented Lagrangian Update Rule

for each PN iteration i **do**
    ...
    if $\frac{1}{h}||p^i||_\infty < \max(10^{-2}l, \epsilon_d)$ **and** $\eta_{\text{AL}} < 0.999$ **then**
        if $\eta_{\text{AL}} < 0.99$ **and** $\kappa_{\text{AL}} < 10^8$ **then**
            $\kappa_{\text{AL}} \leftarrow 2\kappa_{\text{AL}}$
        else
            for each constrained node k **do**
                $\lambda_{\text{AL},k} \leftarrow \lambda_{\text{AL},k} - \kappa_{\text{AL}}\sqrt{m_k}(x_k^i - \hat{x}_k^{t+1})$

For codimensional surface and segment collision objects their nodal mass is computed by estimating their nodal volume as the half sphere with diameter being the average length of the incident edges. For point collision objects we set their mass to be the average nodal mass of the simulated objects. Alternatively, simply setting all codimensional nodal masses to be the average of the simulated objects should also be fine. These estimated masses are only used for moving codimensional collision objects and they do not affect any physical accuracy.

Appendix F: Adaptive Barrier Stiffness

Stiffness, and so difficulty in solution of the barrier comes from two sources: $\hat{d}$ and likewise κ. As we can improve accuracy by directly decreasing $\hat{d}$, this frees κ to adaptively condition our solves for improved convergence.

A feature of our barrier framework is that the estimation of Lagrange multipliers are efficiently self-adjusted by the constraint values, in our case, the d_k's. However, if κ is not set appropriately, the contact primitives would either need to get extremely close to get enough repulsion (barrier gradient) when κ is too small, or need to get a distance right below $\hat{d}$ for a small repulsion if κ is set too large, both resulting in slow convergence because of ill-conditioning and nonsmoothness. Thus adaptively setting and/or adjusting κ

is essential.

Intuitively, the smaller d_k is, the better the complimentarity condition is satisfied. This is certainly true from optimization theory, but is troublesome in numerical computation. In numerical optimization, since double precision floating point number is used, when d_k gets tiny, the rounding errors will significantly slow down convergence and even result in incorrect intermediate computations. Therefore, we must also prevent d_k from being too tiny.

If we view $\hat{d}$ as a control on the upper bound acting distance of our contact forces, κ can be seen as an indirect control on the lower bound of the acting distance as larger κ can provide the same amount of "repulsion" at a larger distance, avoiding the need to push distances to become too tiny. Tiny distances not only make CCD less robust and make the optimization less efficient in our numerical simulation, but also are not physically reasonable in science. Generally speaking, the space between the nucleus of two bonded atoms is around 10^{-10} meters. There is no way for two macroscopic touching objects to get to that close. On the other hand, recall that the energy gradient of our optimization is

$$g(x, \kappa, \hat{d}) = \nabla E(x) + \kappa \sum_k \frac{\partial b}{\partial d_k} \nabla d_k(x, \hat{d}), \tag{6}$$

where at subproblem convergence, the IP gradient $\nabla E(x)$ balances with the barrier gradient $\kappa \sum_k \frac{\partial b}{\partial d_k} \nabla d_k(x, \hat{d})$. In addition, the barrier stiffness κ also influences the condition of the energy Hessian. Thus we adapt barrier stiffness strategy based on balancing the two gradients, iteratively increasing κ when needed, and applying lower and upper bounds obtained from conditioning analysis on the Hessian. Here d_k can then avoid being too small or too close to $\hat{d}$ to provide improved convergence regardless of $\hat{d}$, material, h, and our other input settings change.

The idea of balancing gradients can be traced back in optimization literature Nocedal and Wright [2006] for estimating appropriate initial stiffnesses of barriers. In our case, we solve $\text{argmin}_\kappa ||g(x^j, \kappa, \hat{d})||^2$ which gives us $-g_c \cdot \nabla E(x)/||g_c||^2$ (where $g_c = \sum_k \frac{\partial b}{\partial d_k} \nabla d_k(x, \hat{d})$) at start of each time step to obtain an estimate of κ that seeks to

balance the two gradients at x^j. However, we observe that the effectiveness of this balancing term is highly dependent on the x^j applied. It can be quote far from the configuration at solution and so potentially can leads to poorly scaled or even negative values for κ. Thus, we extend our analysis from the balancing gradients to additionally include conditioning of the Hessian. This, in turn, obtains an effective estimate to provide a lower bound of κ in support of the gradient balancing strategy.

Our analysis seeks to keep the scaling of the diagonal entries of the Hessian, at small distances, close to the mass. Specifically, a scaling characterization of the Hessian diagonal in $\nabla^2 b$ at $d = 10^{-8}l$ is easily estimated by taking its first term $\nabla d^T \frac{\partial^2 b}{\partial d^2} \nabla d$ in point-point formula as

$$c_{\nabla^2 b} = (||\nabla d^T||^2 \frac{\partial^2 b}{\partial d^2})|_{d=10^{-8}l} = 4 \times 10^{-16} l^2 \frac{\partial^2 b}{\partial d^2}(10^{-8}l).$$

We then set the lower bound of κ to provide at least 10^{11} times of the average lumped nodal mass $\bar{m}$ on the diagonal entries when $d = 10^{-8}l$ and so enable production of sufficient repulsion at larger distances, that is

$$\kappa_{\text{min}} = 10^{11} \bar{m} / c_{\nabla^2 b}$$

Note that our κ lower bound will be different for different $\hat{d}$, effectly capturing the curvature change of the barrier. With this lower bound, we now can safely use the gradient scheme with bounded κ value.

Distance can still become small if resultant stress or applied compression (e.g, BCs) in the scene are extreme. We thus add an additional, final κ adjustment that doubles κ, when needed, in between Newton iterations. After every Newton iteration, if we detect that there are contact pairs having a characteristic distance smaller than minimum $\hat{d}_\epsilon = 10^{-9}l$ both before and after this iteration, and the distance is decreasing in this Newton iteration, we double the κ value. Although we do not observe divergence of the adapted κ values we apply a fixed upper bound of $\kappa_{\text{max}} = 100\kappa_{\text{min}}$.

To summarize, our adaptive barrier stiffness strategy is:

1. At start of each time step, compute κ_g giving smallest gradient, and set $\kappa \leftarrow$

$\min(\kappa_{\max}, \max(\kappa_{\min}, \kappa_g))$.

2. After each Newton iteration, if any contact pair has distance smaller than $\hat{d}_\epsilon$ both before and after this iteration, and the distance is decreasing, set $\kappa \leftarrow \min(\kappa_{\max}, 2\kappa)$.

Appendix G: Distance Computation Implementation

Point-point and point-edge constraint duplications

As we discuss in Chapter 4 many point-triangle and edge-edge distances can and will reduce to point-point or point-edge distance in computation. Thus there can be multiples of exactly the same point-point and/or point-edge stencils in our constraint set. While it is tempting to simply either ignore or remove these duplicates, neither strategy is effective. Ignoring duplication in code can lead to significant redundant computation of the same force and Hessian. On the other hand removing them introduces inconsistency into our objective energy, leading to poor convergence or even divergence over iterations. Instead, we track duplicate stencils, computing their energy, gradient, and Hessian evaluations only once for each distinct stencil and then multiply their entries appropriately so that all terms are correctly applied but still avoiding redundant and expensive computation.

Nearly parallel edge-edge distance

When computing distances between two nearly parallel edges using the edge-edge plane distance formula (4.23), numerical rounding errors will generate huge gradient and Hessian values, and even results in wrong distances (Figure 3) because of the ill-defined normal, making our optimization intractable with double precision floating point numbers. Therefore, we check the angle between edges, forcing each case to reduce to the most appropriate point-point or point-edge constraints if the sine value is smaller than 10^{-10}. This clearly makes the distance function C^0 continuous again at the threshold. However, the nonsmoothness is nearly negligible (Figure 3), while our multiplying energy smoother $e_{k,l}(x)$ is also

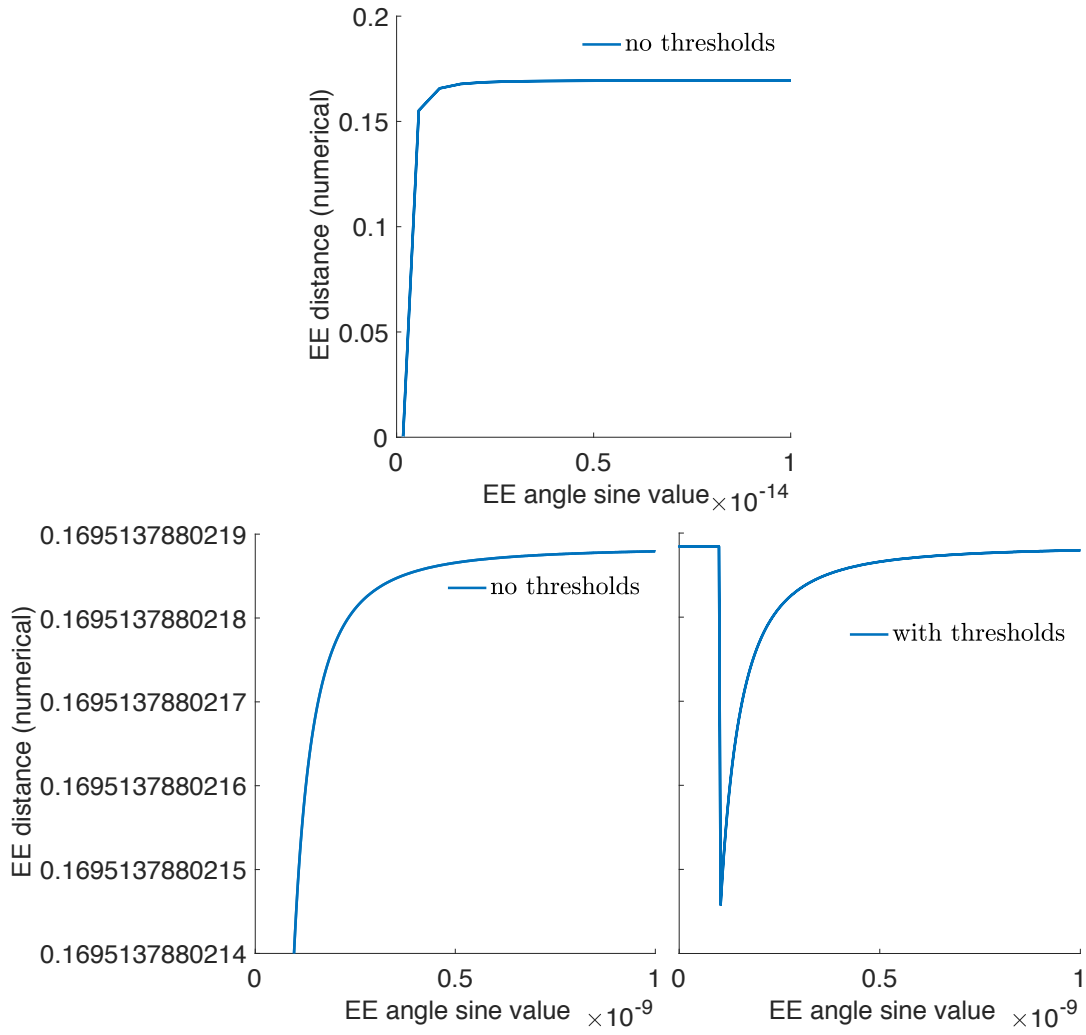


Figure 3: **Parallel-edge degeneracy handling.** Left: Due to numerical rounding errors for distances of edge-edge pairs decreases to 0 when the edges get more and more parallel (see Figure 4.6 for an example); Middle: a zoom-in view show clearly that the distance really starts decreasing when their angle's sine value is at around 10^{-9}; Right: we here set a threshold to force the use of point-point or point-edge formulas to compute the distance of edge-edge pairs when their angle's sine value is below 10^{-10}, which introduces a negligible (for optimization) nonsmoothness.

extremely small when edges are nearly parallel. In practice we find this robustly avoids numerical issues and converges well for all benchmark tests; see Section 4.7.

Appendix H: Tangent and Sliding Modes

After reducing the general point-triangle, and edge-edge distances to one of the closed form formulas, see Section 4.6, we can directly compute the sliding basis operators for each of the four types of contact pairs required for computing friction.

We start by defining the basis, $P_k(x) \in R^{3\times 2}$, formed by the two orthogonal 3D unit-length column vectors spanning the tangent space of the contact pair k, and a selection matrix $\mathit{\Gamma}_k \in R^{3\times 3n}$ which computes relative velocity $v_k = \mathit{\Gamma}_k v$ of each contact pair k. Then we can define the sliding basis $T_k(x) = \mathit{\Gamma}_k^T P_k(x)$ that maps tangent space relative velocity or displacement to the stacked global vector. Here we then list the construction for P and $\mathit{\Gamma}$ for each contact distance type:

Point (x_0) – Triangle ($x_1x_2x_3$)

$$P_k(x) = \left[\frac{x_2-x_1}{||x_2-x_1||}, n \times \frac{x_2-x_1}{||x_2-x_1||}\right] \tag{7}$$

where $n = \frac{(x_2-x_1)\times(x_3-x_1)}{||(x_2-x_1)\times(x_3-x_1)||}$. Each row of $\mathit{\Gamma}_k$ is

$$[..., 1, ..., (-1+\beta_1+\beta_2), ..., -\beta_1, ..., -\beta_2, ...] \tag{8}$$

where β's are those defined in $\mathscr{D}^{P-T}$ (4.18).

Edge (x_0x_1) – Edge (x_2x_3)

$$P_k(x) = \left[\frac{x_1-x_0}{||x_1-x_0||}, n \times \frac{x_1-x_0}{||x_1-x_0||}\right] \tag{9}$$

where $n = \frac{(x_1-x_0)\times(x_3-x_2)}{||(x_1-x_0)\times(x_3-x_2)||}$. Each row of $\mathit{\Gamma}_k$ is

$$[..., 1-\gamma_1, ..., \gamma_1, ..., \gamma_2-1, ..., -\gamma_2, ...] \tag{10}$$

where γ's are those defined in $\mathscr{D}^{E-E}$ (4.19).

Point (x_0) – Edge (x_1x_2)

$$P_k(x) = \left[\frac{x_2-x_1}{||x_2-x_1||}, \frac{(x_2-x_1)\times(x_0-x_1)}{||(x_2-x_1)\times(x_0-x_1)||}\right] \tag{11}$$

Each row of Γ_k is

$$[..., 1, ..., \eta - 1, ..., -\eta, ...] \tag{12}$$

where $(1-\eta)x_1 + \eta x_2$ is the closest point to x_0 on edge x_1x_2.

Point x_0 – Point x_1

$$P_k(x) = \left[t, \frac{x_1-x_0}{||x_1-x_0||} \times t\right] \tag{13}$$

where $t = \frac{e\times(x_1-x_0)}{||e\times(x_1-x_0)||}$ and e is $(1,0,0)$ if $(x_1 - x_0)$ is not colinear with $(1,0,0)$, or e is $(0,1,0)$. Each row of Γ_k is $[..., 1, ..., -1, ...]$.

Appendix I: Friction Implementation

Since we lag the sliding basis in friction computations to $T^n = T(x^n)$ and normal forces to λ^n in (see Section 4.5) from either the last time step or the last friction update iteration n, all other terms are integrable. The lagged friction is then

$$F_k(x, \lambda^n, T^n, \mu) = -\mu\lambda^n T_k^n f_1(||u_k||)\frac{u_k}{||u_k||} \tag{14}$$

and gives us a simple and compact friction potential

$$D_k(x) = \mu\lambda_k^n f_0(||u_k||). \tag{15}$$

Here f_0 is given by $f_0' = f_1$ and $f_0(\epsilon_v h) = \epsilon_v h$ so that $F_k(x) = -\nabla D_k(x)$. In turn this likewise provides a simple-to-compute Hessian contribution

$$\begin{aligned}\nabla^2 D_k(x) = \ & \mu\lambda_k^n T_k^n \Big(\frac{f_1'(||u_k||)||u_k|| - f_1(||u_k||)}{||u_k||^3} u_k u_k^T \\ & + \frac{f_1(||u_k||)}{||u_k||} I_2\Big) T_k^{nT}.\end{aligned} \tag{16}$$

where $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Projecting this Hessian to PSD then simply requires projecting the 2×2 matrix

$$\frac{f_1'(||u_k||)||u_k|| - f_1(||u_k||)}{||u_k||^3} u_k u_k^T + \frac{f_1(||u_k||)}{||u_k||} I_2 \tag{17}$$

to SPD as T_k^n is symmetrically multiplied on both of its two sides.

The above model is general so that f_0 and f_1' are both easy to define for a range of f_1 choices:

1. C^0 F_f: $f_0(x) = \frac{x^2}{2\epsilon_v h} + \frac{\epsilon_v h}{2}$, $f_1(x) = \frac{x}{\epsilon_v h}$, and $f_1'(x) = \frac{1}{\epsilon_v h}$;
2. C^1 F_f: $f_0(x) = -\frac{x^3}{3\epsilon_v^2 h^2} + \frac{x^2}{\epsilon_v h} + \frac{\epsilon_v h}{3}$, $f_1(x) = -\frac{x^2}{\epsilon_v^2 h^2} + \frac{2x}{\epsilon_v h}$, and $f_1'(x) = -\frac{2x}{\epsilon_v^2 h^2} + \frac{2}{\epsilon_v h}$;
3. C^2 F_f: $f_0(x) = \frac{x^4}{4\epsilon_v^3 h^3} - \frac{x^3}{\epsilon_v^2 h^2} + \frac{3x^2}{2\epsilon_v h} + \frac{\epsilon_v h}{4}$, $f_1(x) = \frac{x^3}{\epsilon_v^3 h^3} - \frac{3x^2}{\epsilon_v^2 h^2} + \frac{3x}{\epsilon_v h}$, and $f_1'(x) = \frac{3x^2}{\epsilon_v^3 h^3} - \frac{6x}{\epsilon_v^2 h^2} + \frac{3}{\epsilon_v h}$.

Importantly this also emphasizes that there are never any divisions by $||u_k||$ in all of our energy, gradient, and Hessian computations for friction as they are always cancelled out. This ensures that so that the implemented computation can be robust and accurate.

Our friction model applies our C^1 – (2) in the above.. This design choice again provides a continuous Hessian for better convergence in our Newton-type method. Here we also provide a comparison of behavior of the C^1 model w.r.t. to the different orders of smoothed friction model on our arch and ball roller example (Figure 4).

Examples	C0 # iters, t (s) / time step	C1 (IPC) # iters, t (s) / time step	C2 # iters, t (s) / time step
sphere1K roller	1.37, 0.01	1.24, 0.01	1.26, 0.02
1m-height arch (static)	53.60, 12.21	45.22, 9.79	52.83, 11.42

Figure 4: **Ablation study on smoothed static friction.**

We observe that C^1 friction model provides a "sweet-spot": improvement over C^0 due to the C^1 model's continuous hessian, while it also improves over the C^2 model with less additional nonlinearity.

To compute friction potential, f_0 is well defined without any division by 0.

For friction gradient/force, a robust implementation requires algebraically deriving $\frac{f_1(||u_k||)}{||u_k||}$, which does not contain division by 0 by construction of f_1.

For friction Hessian, the inner 2×2 matrix is

$$\frac{f_1'(||u_k||)||u_k|| - f_1(||u_k||)}{||u_k||^3} u_k u_k^T + \frac{f_1(||u_k||)}{||u_k||} I_2$$

, one needs to first algebraically derive $\frac{f_1'(||u_k||)||u_k|| - f_1(||u_k||)}{||u_k||^2}$ which does not contain division by 0 by construction, and note that $\frac{u_k u_k^T}{||u_k||}$ is always bounded and it equals 0 when $||u_k|| = 0$, in this case since $\frac{f_1(||u_k||)}{||u_k||} > 0$ is always true there's no need for SPD projection.

Let $u_k = (x, y)$,

$$\begin{aligned}
&\frac{f_1'(||u_k||)||u_k|| - f_1(||u_k||)}{||u_k||^3} u_k u_k^T + \frac{f_1(||u_k||)}{||u_k||} I_2 \\
=&\frac{f_1'(||u_k||)}{||u_k||^2} u_k u_k^T + \frac{-f_1(||u_k||)}{||u_k||^3} \begin{bmatrix} x^2 & xy \\ xy & y^2 \end{bmatrix} \\
&+ \frac{f_1(||u_k||)}{||u_k||^3} \begin{bmatrix} x^2 + y^2 & 0 \\ 0 & x^2 + y^2 \end{bmatrix} \\
=&\frac{f_1'(||u_k||)}{||u_k||^2} u_k u_k^T + \frac{f_1(||u_k||)}{||u_k||^3} \begin{bmatrix} y^2 & -xy \\ -xy & x^2 \end{bmatrix} \\
=&\frac{f_1'(||u_k||)}{||u_k||^2} u_k u_k^T + \frac{f_1(||u_k||)}{||u_k||^3} \bar{u}_k \bar{u}_k^T
\end{aligned} \tag{18}$$

where $\bar{u}_k = (-y, x)$. So for $||u_k|| \geq \epsilon_v h$ when we have $f_1'(\cdot) = 0$ and $f_1(\cdot) = 1$, the matrix is always SPD (must compute using $\bar{u}_k$, the original formula can still be not SPD due to numerical error of subtraction!); otherwise we project the 2x2 matrix.

Appendix J: Squared Terms

In our implementation, we apply squared distances in our evaluations to avoid numerical errors and inefficiencies that can be introduced by taking squared roots – especially in gradient and Hessian computations. Concretely, our barrier terms are applied as $b(d^2, \hat{d}^2)$ throughout our implementation. This manipulation leaves our problem formulation with unsigned distances unchanged as have $d > 0 \Leftrightarrow d^2 > 0$. However, we must be careful with units in order to preserve appropriate scaling of dimensions – especially in terms of frictio. Fortunately, we can sort this out directly via the chain rule.

To ensure that units of normal force remain correct we now observe that directly plugging in d^2 and $\hat{d}^2$ into Equation (4.9) in our Chapter 4 no longer computes the contact force magnitudes λ_k defined there. Here we now have $-\frac{\kappa}{h^2}\frac{\partial b}{\partial (d_k^2)}(d_k^2, \hat{d}_k^2)$. Applying the squared formulation we rewrite the stationarity of the barrier as

$$M(\frac{x - \hat{x}}{h^2}) = -\nabla \Psi(x) - \frac{\kappa}{h^2} \sum_{k \in \mathscr{C}} \frac{\partial b}{\partial (d_k^2)} \frac{\partial (d_k^2)}{\partial (d_k)} \nabla d_k(x) \tag{19}$$

which is $M(\frac{x-\hat{x}}{h^2}) = -\nabla \Psi(x) - \frac{\kappa}{h^2} \sum_{k \in \mathscr{C}} \frac{\partial b}{\partial (d_k^2)} 2 d_k \nabla d_k(x)$. In turn this allows us to extract the correct contact force magnitudes (when using squared distances) as $\lambda_k = -\frac{\kappa}{h^2}\frac{\partial b}{\partial (d_k^2)} 2 d_k$.

Appendix K: Isotropic Strain Limiting Derivative Derivations

Will need derivatives of singular values:

$\partial S_{ii}/\partial F = u_i v_i^T$

$\frac{\partial^2 S_{ii}}{(\partial F)^2} = \frac{\partial (u_i v_i^T)}{\partial F} = \frac{\partial u_i}{\partial F} v_i^T + u_i \frac{\partial v_i}{\partial F}^T$

where $\frac{\partial u_i}{\partial F}$ and $\frac{\partial v_i}{\partial F}$ can be found in Xu et al. [2015].

Then if we call the barrier $b_{t,i}$ for triangle t, i-th singular value, we have

$\frac{\partial b_{t,i}}{\partial x} = \frac{\partial b_{t,i}}{\partial S_{t,ii}} \frac{\partial S_{t,ii}}{\partial F_t} \frac{\partial F_t}{\partial x}$

$\frac{\partial^2 b_{t,i}}{\partial x^2} = \frac{\partial S_{t,ii}}{\partial x}^T \frac{\partial^2 b_{t,i}}{\partial S_{t,ii}^2} \frac{\partial S_{t,ii}}{\partial x} + \frac{\partial b_{t,i}}{\partial S_{t,ii}} \frac{\partial^2 S_{t,ii}}{\partial x^2}$

where

- $\frac{\partial F_t}{\partial x}$ can be found in anisotropic strain limiting implementation detail section.
- $\frac{\partial b_{t,i}}{\partial S_{t,ii}} = \frac{\kappa_s}{(s-\hat{s})^2}(2(\hat{s}-S_{t,ii})\ln(\frac{s-S_{t,ii}}{s-\hat{s}}) + \frac{(\hat{s}-S_{t,ii})^2}{s-S_{t,ii}})$
- $\frac{\partial^2 b_{t,i}}{\partial S_{t,ii}^2} = \frac{\kappa_s}{(s-\hat{s})^2}(-2\ln(\frac{s-S_{t,ii}}{s-\hat{s}}) - 4\frac{\hat{s}-S_{t,ii}}{s-S_{t,ii}} + \frac{\hat{s}-S_{t,ii}}{s-S_{t,ii}}^2)$
- $\frac{\partial S_{t,ii}}{\partial x} = \frac{\partial S_{t,ii}}{\partial F_t}\frac{\partial F_t}{\partial x}$
- $\frac{\partial^2 S_{t,ii}}{\partial x^2} = \frac{\partial F_t}{\partial x}^T \frac{\partial^2 S_{t,ii}}{\partial F_t^2}\frac{\partial F_t}{\partial x}$

For efficiency we can compute

$\frac{\partial b_t}{\partial x} = \sum_i (\frac{\partial b_{t,i}}{\partial S_{t,ii}}\frac{\partial S_{t,ii}}{\partial F_t})\frac{\partial F_t}{\partial x}$

$\frac{\partial^2 b_{t,i}}{\partial x^2} = \frac{\partial F_t}{\partial x}^T \sum_i (\frac{\partial^2 b_{t,i}}{\partial F_t^2})\frac{\partial F_t}{\partial x}$

- where $\frac{\partial^2 b_{t,i}}{\partial F_t^2} = \frac{\partial S_{t,ii}}{\partial F_t}^T \frac{\partial^2 b_{t,i}}{\partial S_{t,ii}^2}\frac{\partial S_{t,ii}}{\partial F_t} + \frac{\partial b_{t,i}}{\partial S_{t,ii}}\frac{\partial^2 S_{t,ii}}{\partial F_t^2}$
- for SPD projection we can process the 6x6 matrix $\sum_i (\frac{\partial^2 b_{t,i}}{\partial F_t^2})$

Appendix L: Anisotropic Strain Limiting Derivations

Quadratically Approximating Data-Driven Model

- 1st order derivatives

$\partial\psi/\partial\tilde{E}_{11} = a_{11}\eta_1'(\tilde{E}_{11}^2)\tilde{E}_{11} + a_{12}\eta_2'(\tilde{E}_{11}\tilde{E}_{22})\tilde{E}_{22}$

$\partial\psi/\partial\tilde{E}_{22} = a_{22}\eta_3'(\tilde{E}_{22}^2)\tilde{E}_{22} + a_{12}\eta_2'(\tilde{E}_{11}\tilde{E}_{22})\tilde{E}_{11}$

$\partial\psi/\partial\tilde{E}_{12} = 2G_{12}\eta_4'(\tilde{E}_{12}^2)\tilde{E}_{12}$

when $\tilde{E} = 0$, element is at rest shape, all gradients are equal to 0.

- 2nd order derivatives

$\partial^2\psi/\partial\tilde{E}_{11}^2 = 2a_{11}\eta_1''(\tilde{E}_{11}^2)\tilde{E}_{11}^2 + a_{11}\eta_1'(\tilde{E}_{11}^2) + a_{12}\eta_2''(\tilde{E}_{11}\tilde{E}_{22})\tilde{E}_{22}^2$

$\partial^2\psi/\partial\tilde{E}_{11}\partial\tilde{E}_{22} = \partial^2\psi/\partial\tilde{E}_{22}\partial\tilde{E}_{11} = a_{12}\eta_2''(\tilde{E}_{11}\tilde{E}_{22})\tilde{E}_{22}\tilde{E}_{11} + a_{12}\eta_2'(\tilde{E}_{11}\tilde{E}_{22})$

$\partial^2\psi/\partial\tilde{E}_{22}^2 = 2a_{22}\eta_3''(\tilde{E}_{22}^2)\tilde{E}_{22}^2 + a_{22}\eta_3'(\tilde{E}_{22}^2) + a_{12}\eta_2''(\tilde{E}_{11}\tilde{E}_{22})\tilde{E}_{11}^2$

$\partial^2\psi/\partial\tilde{E}_{12}^2 = 4G_{12}\eta_4''(\tilde{E}_{12}^2)\tilde{E}_{12}^2 + 2G_{12}\eta_4'(\tilde{E}_{12}^2)$

other terms are 0

when $\tilde{E} = 0$, $\partial^2\psi/\partial\tilde{E}_{11}^2 = a_{11}$ $\partial^2\psi/\partial\tilde{E}_{11}\partial\tilde{E}_{22} = \partial^2\psi/\partial\tilde{E}_{22}\partial\tilde{E}_{11} = a_{12}$ $\partial^2\psi/\partial\tilde{E}_{22}^2 = a_{22}$ $\partial^2\psi/\partial\tilde{E}_{12}^2 = 2G_{12}$ so to quadratically approximate the data-driven model at $\tilde{E} = 0$

using our barrier model, we just need to use the same stiffnesses. This is very simple, but also suggest that our approximation might not work very well to match intricate behaviors as the data-driven model.

Implementation Details

For simplicity, if we align the cloth weft and warp directions to coordinate axis, we can easily compute $\tilde{E}$ from 3x2 deformation gradient $F = [x_2 - x_1, x_3 - x_1][X_2' - X_1', X_3' - X_1']^{-1}$: $\tilde{E}_{ij} = \frac{1}{2}(F_{mi}F_{mj} - \delta_{ij})$ so $\partial \tilde{E}_{ij}/\partial F_{kl} = \frac{1}{2}(\delta_{il}F_{kj} + F_{ki}\delta_{jl})$ $\partial^2 \tilde{E}_{ij}/\partial F_{kl}\partial F_{mn} = \frac{1}{2}\delta_{mk}(\delta_{il}\delta_{nj} + \delta_{ni}\delta_{jl})$

Then together with $\partial\psi$ above and $dF/dx = \{ \begin{bmatrix} -I \\ -I \end{bmatrix} \begin{bmatrix} I \end{bmatrix} \begin{bmatrix} \\ I \end{bmatrix} \} : B$ where $B = [X_2' - X_1', X_3' - X_1']^{-1}$ so

$$
\begin{aligned}
& dF/dx \\
&= \begin{bmatrix} -(B_{11}+B_{21}) & & \\ & * & \\ & & * \\ -(B_{12}+B_{22}) & & \\ & * & \\ & & * \end{bmatrix}, \begin{bmatrix} B_{11} & & \\ & * & \\ & & * \\ B_{12} & & \\ & * & \\ & & * \end{bmatrix}, \begin{bmatrix} B_{21} & & \\ & * & \\ & & * \\ B_{22} & & \\ & * & \\ & & * \end{bmatrix}
\end{aligned} \tag{20}
$$

we are able to implement the constitutive model: $\partial\psi/\partial x = \partial\psi/\partial\tilde{E} * \partial\tilde{E}/\partial F * \partial F/\partial x$

$\partial^2\psi/\partial x^2 = (\partial F/\partial x)^T * \partial^2\psi/\partial F^2 * \partial F/\partial x$ where $\partial^2\psi/\partial F^2 = (\partial\tilde{E}/\partial F)^T * \partial^2\psi/\partial\tilde{E}^2 * \partial\tilde{E}/\partial F + \partial\psi/\partial\tilde{E} * \partial^2\tilde{E}/\partial F^2$

Appendix M: Rod Bending Modulus

In discrete rods model Bergou et al. [2008], the bending energy is directly integrated over the length of the rod as

$$E_{\text{bend}}(x) = \sum_{i=1}^{n} \frac{\alpha(\kappa b_i)^2}{\bar{l}_i} \tag{21}$$

Here κb_i is the curvature binormal at a vertex, $\bar{l}_i = |\bar{e}^{i-1}| + |\bar{e}^i|$ where $|\bar{e}^i|$ is the rest length of edge i, and α is an adjustable parameter to control the stiffness of bending. As we know when a rod becomes thicker, it is harder to bend, which means we need to manually increase α accordingly and this makes seting up examples inconvenient.

According to Kirchoff rod theory Dill [1992], the bending energy of rod at a unit length is defined as

$$\frac{1}{2} E r^4 \frac{\pi}{4} \kappa^2$$

where E is the bending modulus in Pa unit, r is the radius of the rod, and $\kappa = \frac{\kappa b}{\bar{l}/2}$ is the point-wise curvature Bergou et al. [2008]. Integrating this quantity over rod length then gives us

$$\begin{aligned} E_{\text{bend}}(x) &= \int_{\Omega} \frac{1}{2} E r^4 \frac{\pi}{4} \kappa^2 dl \\ &\approx \sum_i \frac{\bar{l}_i}{2} \frac{1}{2} E r^4 \frac{\pi}{4} (\frac{\kappa b_i}{\bar{l}_i/2})^2 \\ &= \sum_i E r^4 \frac{\pi}{4} \frac{(\kappa b_i)^2}{\bar{l}_i} \end{aligned} \tag{22}$$

Combining with Equation 21 we finally know that the α in Bergou et al. [2008] is associated with the rod's radius as

$$\alpha = E r^4 \frac{\pi}{4}$$

so now the change of rod thickness under a certain bending modulus E can be automatically reflected in the bending stiffness. This enables us to easily setup rod examples starting from using real material thickness and Young's modulus of e.g. hairs, iron, etc, without having to tune the unintuitive bending stiffness α.

Bibliography

A. Abdelfattah, A. Haidar, S. Tomov, and J. Dongarra. Fast cholesky factorization on gpus for batch and native modes in magma. *J of Comp Sci*, 20, 2017. 55

Pierre Alart and Alain Curnier. A mixed formulation for frictional contact problems prone to Newton like solution methods. *CMAME*, 92(3), November 1991. 68

ANSYS Group. ANSYS, 2020. URL https://www.ansys.com/. 107

U. M Ascher. *Numerical methods for evolutionary differential equations*. 2008. 29, 34

John M Ball. Global invertibility of sobolev functions and the interpenetration of matter. *Proc. of the Royal Society of Edinburgh: Section A Math.*, 88(3-4), 1981. 65

David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, 1998. 114, 115

James R Barber. *Elasticity*. Springer, 2002. 20

Ted Belytschko, Wing Kam Liu, and Brian Moran. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons, Ltd, 2000. 63, 64, 66

Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. Discrete elastic rods. In *ACM SIGGRAPH 2008 papers*, pages 1–12. 2008. 1, 22, 23, 121, 122, 189

Florence Bertails-Descoubes, Florent Cadoux, Gilles Daviet, and Vincent Acary. A non-smooth newton solver for capturing exact coulomb friction in fiber assemblies. *ACM Trans. on Graph.*, 30(1), February 2011. ISSN 0730-0301. 68

Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2016. 67

Kiran S Bhat, Christopher D Twigg, Jessica K Hodgins, Pradeep Khosla, Zoran Popovic, and Steven M Seitz. Estimating cloth simulation parameters from video. 2003. 115

S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans Graph*, 33(4), 2014. 32, 33

S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1), 2011. 16, 33, 42

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, 2004. ISBN 0521833787. 72

Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. on Graph.*, 21, 05 2002. 64, 68, 104, 114, 115, 119, 125, 151

Tyson Brochu, Essex Edwards, and Robert Bridson. Efficient geometrically exact continuous collision detection. *ACM Transactions on Graphics (TOG)*, 31(4):1–7, 2012. xxii, 119, 158, 159

Bernard Brogliato. *Nonsmooth Mechanics*. Springer-Verlag, 1999. 63

J. Brown and P. Brune. Low-rank quasi-newton updates for robust jacobian lagging in newton-type methods. In *Int Conf Math Comp Meth App Nucl Sci Eng*, 2013. 35, 36, 38

J. C. Butcher. *Numerical methods for ordinary differential equations*. 2016. 29

Rui PR Cardoso, Jeong Whan Yoon, Made Mahardika, S Choudhry, RJ Alves de Sousa, and RA Fontes Valente. Enhanced assumed strain (eas) and assumed natural strain (ans) methods for one-point quadrature solid-shell elements. *International Journal for Numerical Methods in Engineering*, 75(2):156–187, 2008. 118

Juan J Casafranca, Gabriel Cirio, Alejandro Rodríguez, Eder Miguel, and Miguel A Otaduy. Mixing yarns and triangles in cloth simulation. In *Computer Graphics Forum*, volume 39, pages 101–110. Wiley Online Library, 2020. 115

Jumyung Chang, Fang Da, Eitan Grinspun, and Christopher Batty. A unified simplicial model for mixed-dimensional and non-manifold deformable elastic objects. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2(2):1–18, 2019. 119

I. Chao, U. Pinkall, P. Sanan, and P. Schröder. A simple geometric model for elastic deformations. *ACM Trans Graph (SIGGRAPH)*, 29(4), 2010. 29

Hsiao-Yu Chen, Arnav Sastry, Wim M van Rees, and Etienne Vouga. Physical simulation of environmentally induced thin shell deformation. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018. 121

Hsiao-yu Chen, Paul Kry, and Etienne Vouga. Locking-free simulation of isometric thin plates. *arXiv preprint arXiv:1911.05204*, 2019. 20, 117, 140, 141

Ming Chen and Kai Tang. A fully geometric approach for developable cloth deformation simulation. *The visual computer*, 26(6-8):853–863, 2010. xvii, 116, 117

Yanqing Chen, Timothy A Davis, William W Hager, and Sivasankaran Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. on Math. Software (TOMS)*, 35(3), 2008. 12, 45, 88, 89, 136

Byoungwon Choe, Min Gyu Choi, and Hyeong-Seok Ko. Simulating complex hair with robust collision handling. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 153–160, 2005. 114

David Clyde. *Numerical Subdivision Surfaces for Simulation and Data Driven Modeling of Woven Cloth*. PhD thesis, UCLA, 2017. 128, 141

David Clyde, Joseph Teran, and Rasmus Tamstorf. Modeling and data-driven parameter estimation for woven fabrics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–11, 2017. 115, 121, 128

COMSOL Group. COMSOL Multiphysics, 2020. URL https://www.comsol.com/. 107

Richard Courant, Kurt Friedrichs, and Hans Lewy. On the partial difference equations of mathematical physics. *IBM J. of Research and Development*, 11(2), 1967. 77

Peter A Cundall and Otto DL Strack. A discrete numerical model for granular assemblies. *geotechnique*, 29(1):47–65, 1979. 163

Gilles Daviet, Florence Bertails-Descoubes, and Laurence Boissieux. A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics. *ACM Trans. on Graph.*, 30, 12 2011. 64, 68, 108, 148

Denis Demidov. Amgcl: An efficient, flexible, and extensible algebraic multigrid implementation. *Lobachevskii J. of Math.*, 40(5), May 2019. ISSN 1818-9962. 89, 101

P. Deuflhard. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*. 2011. 34, 35

Ellis Harold Dill. Kirchhoff's theory of rods. *Archive for History of Exact Sciences*, 44(1): 1–23, 1992. 189

V. Dolean, P. Jolivet, and F. Nataf. *An introduction to domain decomposition methods: algorithms, theory, and parallel implementation*. 2015. 33

David Doyen, Alexandre Ern, and Serge Piperno. Time-integration schemes for the finite element dynamic signorini problem. *SIAM J. on Sci. Comp.*, 33, 01 2011. 66

Elliot English and Robert Bridson. Animating developable surfaces using nonconforming elements. In *ACM SIGGRAPH 2008 papers*, pages 1–5. 2008. 116, 117

Kenny Erleben. Methodology for assessing mesh-based contact point methods. *ACM Trans. on Graph. (TOG)*, 37(3), 2018. xv, 64, 66, 91

Ye Fan, Joshua Litven, David IW Levin, and Dinesh K Pai. Eulerian-on-lagrangian simulation. *ACM Transactions on Graphics (TOG)*, 32(3):1–9, 2013. 15

Yu Fang, Minchen Li, Ming Gao, and Chenfanfu Jiang. Silly rubber: an implicit material point method for simulating non-equilibrated viscoelastic and elastoplastic solids. *ACM Trans. on Graph. (TOG)*, 38(4), 2019. 2, 168

Yu Fang, Ziyin Qu, Minchen Li, Xinxin Zhang, Yixin Zhu, Mridul Aanjaneya, and Chenfanfu Jiang. Iq-mpm: an interface quadrature material point method for non-sticky strongly two-way coupled nonlinear solids and fluids. *ACM Transactions on Graphics (TOG)*, 39(4):51–1, 2020. 2, 168

François Faure, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, et al. Sofa: A multi-model framework for interactive physical simulation. In *Soft tissue biomechanical modeling for computer assisted surgery*. Springer, 2012. 106

T. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J. M Teran. Optimization integrator for large time steps. *IEEE Trans Vis Comp Graph*, 21(10), 2015. 1, 32, 120

Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. Efficient simulation of inextensible cloth. In *ACM SIGGRAPH 2007 papers*, pages 49–es. 2007. 115, 116, 117

Suresh Goyal, Andy Ruina, and Jim Papadopoulos. Planar sliding with dry friction, Part 2. Dynamics of motion. *Wear*, 143, 1991. 67, 78

Eitan Grinspun, Anil N Hirani, Mathieu Desbrun, and Peter Schröder. Discrete shells. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 62–67. Citeseer, 2003. 1, 19, 21, 121

Gaël Guennebaud, Benoît Jacob, et al. Eigen v3, 2010. 12, 88, 136

Qi Guo, Xuchen Han, Chuyuan Fu, Theodore Gast, Rasmus Tamstorf, and Joseph Teran. A material point method for thin shells with frictional contact. *ACM Transactions on Graphics (TOG)*, 37(4):1–15, 2018. 115

LLC Gurobi Optimization. Gurobi optimizer reference manual, 2019. URL http://www.gurobi.com. 108

E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II*. 1996. 29

E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration*. 2006. 29

E. Hairer, S. P Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I*. 2008. 29

David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. Robust treatment of simultaneous collisions. *SIGGRAPH (ACM Trans. on Graph.)*, 27(3), 2008. 64, 86, 108, 109, 115, 147, 151

David Harmon, Etienne Vouga, Breannan Smith, Rasmus Tamstorf, and Eitan Grinspun. Asynchronous contact mnumpageechanics. In *ACM Trans. on Graph. (TOG)*, volume 28. ACM, 2009. xxi, 64, 69, 94, 114, 118, 152, 153

David Harmon, Daniele Panozzo, Olga Sorkine, and Denis Zorin. Interference-aware geometric modeling. *ACM Transactions on Graphics (TOG)*, 30(6):1–10, 2011. xxii, 119, 158

R Hauptmann and K Schweizerhof. A systematic development of 'solid-shell'element formulations for linear and non-linear analyses employing only displacement degrees of

freedom. *International Journal for Numerical Methods in Engineering*, 42(1):49–69, 1998. 117

F. Hecht, Y. J. Lee, J. R. Shewchuk, and J. F. O'Brien. Updated sparse cholesky factors for corotational elastodynamics. *ACM Trans Graph*, 31(5), 2012. 35

F Hernandez, G Cirio, AG Perez, and MA Otaduy. Anisotropic strain limiting. In *Proc. of Congreso Español de Informática Gráfica*, volume 2, 2013. 117

Guy T Houlsby and Alexander M Puzrin. *Principles of hyperplasticity: an approach to plasticity theory based on thermodynamic principles*. Springer Science & Business Media, 2007. 169

J. Huang, X. Liu, H. Bao, B. Guo, and H. Shum. An efficient large deformation method using domain decomposition. *Comp & Graph*, 30(6), 2006. 33

Michel Jean and Jean Jacques Moreau. Unilaterality and dry friction in the dynamics of rigid body collections. In *Proc. of Contact Mech. Int. Symp.*, volume 1, October 1992. 68, 108

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, pages 1–52. 2016. 18

Chenfanfu Jiang, Theodore Gast, and Joseph Teran. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017a. 15, 115, 119

Yupeng Jiang, Minchen Li, Chenfanfu Jiang, and Fernando Alonso-Marroquin. A hybrid material-point spheropolygon-element method for solid and granular material interaction. *International Journal for Numerical Methods in Engineering*, 2020. 2, 7, 164

Zhongshi Jiang, Scott Schaefer, and Daniele Panozzo. Simplicial complex augmentation framework for bijective maps. *ACM Trans. on Graph.*, 36(6), November 2017b. ISSN 0730-0301. 64, 65

Ning Jin, Wenlong Lu, Zhenglin Geng, and Ronald P Fedkiw. Inequality cloth. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–10, 2017. 117, 140

Mark W. Jones, J. Andreas Baerentzen, and Milos Sramek. 3d distance fields: A survey of techniques and applications. *IEEE Trans. on Vis. and Comp. Graph.*, 12(4), July 2006. ISSN 1077-2626. 65

C. Kane, J. E Marsden, M. Ortiz, and M. West. Variational integrators and the newmark algorithm for conservative and dissipative mechanical systems. *Int J for Numer Meth in Eng*, 49(10), 2000. 29, 61, 81, 120

Couro Kane, Eduardo A Repetto, Michael Ortiz, and Jerrold E Marsden. Finite element analysis of nonsmooth contact. *CMAME*, 180(1-2), 1999. 1, 24, 61, 64, 66, 67, 108, 109

G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J on Sci Comp*, 20, 1998. 39, 45

Danny M Kaufman and Dinesh K Pai. Geometric numerical integration of inequality constrained, nonsmooth Hamiltonian systems. *SIAM J. on Sci. Comp.*, 34(5), 2012. 61, 66, 120, 121

Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. Staggered projections for frictional contact in multibody systems. *ACM Trans. on Graph. (SIGGRAPH Asia 2008)*, 27(5), 2008. 1, 24, 67, 68, 98, 108

Danny M. Kaufman, Rasmus Tamstorf, Breannan Smith, Jean-Marie Aubry, and Eitan Grinspun. Adaptive nonlinearity for collisions in complex rod assemblies. *ACM Trans. on Graph.*, 33(4), July 2014. ISSN 0730-0301. 68, 108

L. Kharevych, W. Yang, Y. Tong, E. Kanso, J. E Marsden, P. Schröder, and M. Desbrun. Geometric, variational integrators for computer animation. In *Symp Comp Anim*, 2006. 29

Noboru Kikuchi and John Tinsley Oden. *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods*, volume 8 of *SIAM Studies in App. and Numer. Math.* Society for Industrial and Applied Mathematics, 1988. 63

T. Kim and D. L James. Physics-based character skinning using multidomain subspace deformations. *IEEE Trans on visualization and Comp Graph*, 18(8), 2012. 33

Dan Koschier, Crispin Deul, Magnus Brand, and Jan Bender. An hp-adaptive discretization algorithm for signed distance field generation. *IEEE Trans. on Vis. and Comp. Graph.*, 23(10), 2017. 65

Rolf. Krause and Patrick. Zulian. A parallel approach to the variational transfer of discrete fields between arbitrarily distributed unstructured finite element meshes. *SIAM J. on Sci. Comp.*, 38(3), 2016. 107

Chen Li, Min Tang, Ruofeng Tong, Ming Cai, Jieyi Zhao, and Dinesh Manocha. P-cloth: Interactive complex cloth simulation on multi-gpu systems using dynamic matrix assembly and pipelined implicit integrators. *arXiv preprint arXiv:2008.00409*, 2020a. 114, 115

Jie Li, Gilles Daviet, Rahul Narain, Florence Bertails-Descoubes, Matthew Overby, George E Brown, and Laurence Boissieux. An implicit frictional contact solver for adaptive cloth simulation. *ACM Transactions on Graphics (TOG)*, 37(4):1–15, 2018a. xx, 115, 118, 147, 148

Minchen Li, Danny M. Kaufman, Vladimir G. Kim, Justin Solomon, and Alla Sheffer. Optcuts: Joint optimization of surface cuts and parameterization. *ACM Trans. on Graph.*, 37(6), December 2018b. ISSN 0730-0301. 65

Minchen Li, Alla Sheffer, Eitan Grinspun, and Nicholas Vining. Foldsketch: Enriching garments with physically reproducible folds. *ACM Transaction on Graphics*, 37(4), 2018c. doi: http://dx.doi.org/10.1145/3197517.3201310. xxi, 152, 154

Minchen Li, Ming Gao, Timothy Langlois, Chenfanfu Jiang, and Danny M Kaufman. Decomposed optimization time integrator for large-step elastodynamics. *ACM Transactions on Graphics (TOG)*, 38(4):1–10, 2019. 2, 12, 14, 15, 47, 49, 50, 53, 120

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. Incremental potential contact: Intersection- and inversion-free large deformation dynamics. *ACM Transactions on Graphics*, 39(4), 2020b. xiv, xvii, xxii, 2, 15, 23, 83, 89, 92, 95, 97, 98, 102, 106, 107, 111, 118, 119, 120, 121, 122, 123, 124, 126, 129, 130, 131, 132, 157, 158, 165

Minchen Li, Danny M. Kaufman, and Chenfanfu Jiang. Contact-aware constitutive strain limiting for thin structures, 2020c. 2

Xuan Li, Jessica McWilliams, Minchen Li, Cynthia Sung, and Chenfanfu Jiang. Soft hybrid aerial vehicle via bistable mechanism, 2020d. 2, 7, 169

Yue Li, Xuan Li, Minchen Li, Yixin Zhu, Bo Zhu, and Chenfanfu Jiang. A hybrid lagrangian-eulerian method for topology optimization, 2020e. 2, 7

Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. In *Advances in Neural Information Processing Systems*, pages 772–781, 2019. 115

H. Liu, N. Mitchell, M. Aanjaneya, and E. Sifakis. A scalable schur-complement fluids solver for heterogeneous compute platforms. *ACM Trans Graph*, 35(6), 2016. 33

T. Liu, A. W. Bargteil, J. F. O'Brien, and L. Kavan. Fast simulation of mass-spring systems. *ACM Trans Graph*, 32(6), 2013. Proc of ACM SIGGRAPH Asia. 32

T. Liu, S. Bouaziz, and L. Kavan. Quasi-Newton methods for Real-Time simulation of hyperelastic materials. *ACM Trans Graph*, 36(4), 2017. 1, 29, 32, 33, 34, 35, 36, 38, 45, 120

Libin Lu, Matthew J Morse, Abtin Rahimian, Georg Stadler, and Denis Zorin. Scalable simulation of realistic volume fraction red blood cell flows through vascular networks. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–30, 2019. xxii, 119, 158

M. Macklin, M. Müller, and N. Chentanez. Xpbd: position-based simulation of compliant constrained dynamics. In *Proc of the 9th Int Conf on Motion in Games*, 2016. 33

Miles Macklin, Kenny Erleben, Matthias Müller, Nuttapong Chentanez, Stefan Jeschke, and Viktor Makoviychuk. Non-smooth newton methods for deformable multi-body dynamics. 07 2019. 68, 108

S. Martin, B. Thomaszewski, E. Grinspun, and M. Gross. Example-based elastic materials. *ACM Trans Graph (SIGGRAPH)*, 30(4), 2011. 29

Sebastian Martin, Peter Kaufmann, Mario Botsch, Eitan Grinspun, and Markus Gross. Unified simulation of elastic rods, shells, and solids. *ACM Transactions on Graphics (TOG)*, 29(4):1–10, 2010. 119

A. McAdams, A. Selle, R. Tamstorf, J. Teran, and E. Sifakis. Computing the singular value decomposition of $3\times$ 3 matrices with minimal branching and elementary floating point operations. *University of Wisconsin Madison*, 2011. 45

Aleka McAdams, Andrew Selle, Kelly Ward, Eftychios Sifakis, and Joseph Teran. Detail preserving continuum simulation of straight hair. *ACM Transactions on Graphics (TOG)*, 28(3):1–6, 2009. xxiii, 114, 160, 161

Eder Miguel, Derek Bradley, Bernhard Thomaszewski, Bernd Bickel, Wojciech Matusik, Miguel A Otaduy, and Steve Marschner. Data-driven estimation of cloth simulation

models. In *Computer Graphics Forum*, volume 31, pages 519–528. Wiley Online Library, 2012. 115

Marek Krzysztof Misztal and Jakob Andreas Bærentzen. Topology-adaptive interface tracking using the deformable simplicial complex. *ACM Trans. on Graph.*, 31(3), June 2012. ISSN 0730-0301. 64

Jean Jacques Moreau. On unilateral constraints, friction and plasticity. *New Variational Tech. in Math. Phys.*, 1973. 27, 61, 67, 78

M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position based dynamics. *J. Vis. Commun. Imag Represent.*, 18(2), 2007. 33

Matthias Müller, Tae-Yong Kim, and Nuttapong Chentanez. Fast simulation of inextensible hair and fur. *VRIPHYS*, 12:39–44, 2012. 114

Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. Air meshes for robust collision handling. *ACM Trans. on Graph.*, 34(4), July 2015. ISSN 0730-0301. 65

R. Narain, M. Overby, and G. E Brown. Admm $\supseteq$ projective dynamics: fast simulation of general constitutive models. In *Symp on Comp Anim*, 2016. 32, 34

Rahul Narain, Armin Samii, and James F O'brien. Adaptive anisotropic remeshing for cloth simulation. *ACM transactions on graphics (TOG)*, 31(6):1–10, 2012. xx, 114, 115, 117, 118, 138, 144, 147, 148

JW Neuberger. Steepest descent and differential equations. *J of the Mathematical Society of Japan*, 37(2), 1985. 34, 38

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006. 35, 37, 38, 44, 66, 67, 69, 75, 171, 178

M. Ortiz and L. Stainier. The variational formulation of viscoplastic constitutive updates. *Comp Meth in App Mech and Eng*, 171(3-4), 1999. 29, 61

Miguel Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus Gross. Implicit contact handling for deformable objects. *Comp. Graph. Forum*, 28, 04 2009. 64, 67, 68, 108, 109, 115

M. Overby, G. E Brown, J. Li, and R. Narain. Admm $\supseteq$ projective dynamics: Fast simulation of hyperelastic models with dynamic constraints. *IEEE Trans Vis Comp Graph*, 23(10), 2017. 16, 29, 32, 34, 35, 45, 120

Stéphane Pagano and Pierre Alart. Self-contact and fictitious domain using a difference convex approach. *Int. J. for Numer. Meth. in Eng.*, 75, 07 2008. 65

Anna Pandolfi, Couro Kane, Jerrold E Marsden, and Michael Ortiz. Time-discretized variational formulation of non-smooth frictional contact. *Int. J. for Numer. Meth. in Eng.*, 53(8), 2002. 81

N. Parikh and S. Boyd. Block splitting for distributed optimization, 2012. 33

Željko Penava, Diana Šimić-Penava, and Ž Knezic. Determination of the elastic constants of plain woven fabrics by a tensile test in various directions. *Fibres & Textiles in Eastern Europe*, 2014. 137, 138

Xavier Provot. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation'97*, pages 177–189. Springer, 1997. 118

Xavier Provot et al. Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Graphics interface*, pages 147–147. Canadian Information Processing Society, 1995. 125

A. Quarteroni, A. Valli, and P.M.A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. 1999. 33, 53

Stefanie Reese. A large deformation solid-shell concept based on reduced integration with hourglass stabilization. *International Journal for Numerical Methods in Engineering*, 69 (8):1671–1716, 2007. 118

Nikolas Schmitt, Martin Knuth, Jan Bender, and Arjan Kuijper. Multilevel cloth simulation using gpu surface sampling. *VRIPHYS*, 13:1–10, 2013. 115

T. Schneider, Y. Hu, J. Dumas, X. Gao, D. Panozzo, and D. Zorin. Decoupling simulation accuracy from mesh quality. *ACM Trans Graph*, 2018. 55

Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. Locally injective mappings. *Comp. Graph. Forum (Proc. of EUROGRAPHICS/ACM SIGGRAPH Symp. on Geom. Proc.)*, 32(5), 2013. 69

S. Sellán, H. Y. Cheng, Y. Ma, M. Dembowski, and A. Jacobson. Solid geometry processing on deconstructed domains. *CoRR*, 2018. 33

Andrew Selle, Michael Lentine, and Ronald Fedkiw. A mass spring model for hair simulation. In *ACM SIGGRAPH 2008 papers*, pages 1–11. 2008. 114

VE Shamanskii. A modification of newton's method. *Ukrainian Mathematical J*, 19(1), 1967. 35

A. Shtengel, R. Poranne, O. Sorkine-Hornung, S. Z. Kovalsky, and Y. Lipman. Geometric optimization via composite majorization. *ACM Trans Graph*, 36(4), 2017. 35

SideFX. Houdini, 2020. URL https://www.sidefx.com/products/houdini/. 106

Eftychios Sifakis, Sebastian Marino, and Joseph Teran. Globally coupled collision handling using volume preserving impulses. In *Proc. of the 2008 ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, SCA 2008, 2008. 64

Juan C. Simó and Thomas J. R. Hughes. *Computational Inelasticity*. Springer, 1998. 61

B. Smith, F. De Goes, and T. Kim. Stable neo-hookean flesh simulation. *ACM Trans Graph*, 37(2), 2018. 47

Jason Smith and Scott Schaefer. Bijective parameterization with free boundaries. *ACM Trans. on Graph. (TOG)*, 34(4), 2015. 18, 69, 76

Georg Sperl, Rahul Narain, and Chris Wojtan. Homogenized yarn-level cloth. *ACM Transactions on Graphics (TOG)*, 39(4):48–1, 2020. 115

David E Stewart. Finite-dimensional contact mechanics. *Phil. Trans. R. Soc. Lond. A*, 359, 2001. 63

Alexey Stomakhin, Russell Howes, Craig Schroeder, and Joseph M Teran. Energetically consistent invertible elasticity. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.* Eurographics Association, 2012. 47, 121

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)*, 32 (4):1–10, 2013. 13, 15

Andrew Stuart and Anthony R Humphries. *Dynamical Systems and Numerical Analysis*. Cambridge Univ. Press, 1996. 63

Rasmus Tamstorf and Eitan Grinspun. Discrete bending forces and their jacobians. *Graphical models*, 75(6):362–370, 2013. 21, 121

Rasmus Tamstorf, Toby Jones, and Stephen F McCormick. Smoothed aggregation multigrid for cloth simulation. *ACM Transactions on Graphics (TOG)*, 34(6):1–13, 2015. 115

Min Tang, Dinesh Manocha, Sung-Eui Yoon, Peng Du, Jae-Pil Heo, and Ruo-Feng Tong. Volccd: Fast continuous collision culling between deforming volume meshes. *ACM Transactions on Graphics (TOG)*, 30(5):1–15, 2011. 118

Min Tang, Ruofeng Tong, Rahul Narain, Chang Meng, and Dinesh Manocha. A gpu-based streaming algorithm for high-resolution cloth simulation. In *Computer Graphics Forum*, volume 32, pages 21–30. Wiley Online Library, 2013. 115

Min Tang, Ruofeng Tong, Zhendong Wang, and Dinesh Manocha. Fast and exact continuous collision detection with bernstein sign classification. *ACM Trans. on Graph. (TOG)*, 33 (6), 2014. xxii, 104, 119, 158, 159

Min Tang, Huamin Wang, Le Tang, Ruofeng Tong, and Dinesh Manocha. Cama: Contact-aware matrix assembly with unified collision handling for gpu-based cloth simulation. In *Computer Graphics Forum*, volume 35, pages 511–521. Wiley Online Library, 2016. 115

Min Tang, Tongtong Wang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. I-cloth: incremental collision handling for gpu-based interactive cloth simulation. *ACM Transactions on Graphics (TOG)*, 37(6):1–10, 2018. 115, 151

Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. Robust quasistatic finite elements and flesh simulation. In *Proc. of the 2005 ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.* ACM, 2005. 1, 35, 73

Bernhard Thomaszewski, Simon Pabst, and Wolfgang Strasser. Continuum-based strain limiting. In *Computer Graphics Forum*, volume 28, pages 569–576. Wiley Online Library, 2009. 116, 117

Nobuyuki Umetani, Danny M Kaufman, Takeo Igarashi, and Eitan Grinspun. Sensitive couture for interactive garment modeling and editing. *ACM Trans. Graph.*, 30(4):90, 2011. 153

Mickeal Verschoor and Andrei C Jalba. Efficient and accurate collision response for elastically deformable models. *ACM Trans. on Graph. (TOG)*, 38(2), 2019. xv, 1, 24, 64, 67, 68, 94, 96, 105, 108, 109

Pascal Volino and N Magnenat Thalmann. Implementing fast cloth simulation with collision response. In *Proceedings Computer Graphics International 2000*, pages 257–266. IEEE, 2000. 114

Huamin Wang, James O'Brien, and Ravi Ramamoorthi. Multi-resolution isotropic strain limiting. *ACM Transactions on Graphics (TOG)*, 29(6):1–10, 2010. 117

Huamin Wang, James F O'Brien, and Ravi Ramamoorthi. Data-driven elastic models for cloth: modeling and measurement. *ACM transactions on graphics (TOG)*, 30(4):1–12, 2011. 138

Xinlei Wang, Minchen Li, Yu Fang, Xinxin Zhang, Ming Gao, Min Tang, Danny M Kaufman, and Chenfanfu Jiang. Hierarchical optimization time integration for cfl-rate mpm stepping. *ACM Transactions on Graphics (TOG)*, 39(3):1–16, 2020a. 2, 13, 15, 120, 169

Xinlei Wang, Yuxing Qiu, Stuart R Slattery, Yu Fang, Minchen Li, Song-Chun Zhu, Yixin Zhu, Min Tang, Dinesh Manocha, and Chenfanfu Jiang. A massively parallel and scalable multi-cpu material point method. *ACM Transactions on Graphics (TOG)*, 39(4): 30–1, 2020b. 2, 169

Zhendong Wang, Min Tang, Ruofeng Tong, and Dinesh Manocha. Tightccd: Efficient and robust continuous collision detection using tight error bounds. In *Computer Graphics Forum*, volume 34, pages 289–298. Wiley Online Library, 2015. xxii, 119, 158, 159

Zhendong Wang, Tongtong Wang, Min Tang, and Ruofeng Tong. Efficient and robust strain limiting and treatment of simultaneous collisions with semidefinite programming. *Computational Visual Media*, 2(2):119–130, 2016. 117

Kelly Ward and Ming C Lin. Adaptive grouping and subdivision for simulating hair dynamics. In *11th Pacific Conference onComputer Graphics and Applications, 2003. Proceedings.*, pages 234–243. IEEE, 2003. 114

Nicholas J Weidner, Kyle Piddington, David IW Levin, and Shinjiro Sueda. Eulerian-on-lagrangian cloth simulation. *ACM Transactions on Graphics (TOG)*, 37(4):1–11, 2018. 15, 115

Joshuah Wolper, Yu Fang, Minchen Li, Jiecong Lu, Ming Gao, and Chenfanfu Jiang. Cd-mpm: Continuum damage material point methods for dynamic fracture animation. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019. 2, 15, 168

Joshuah Wolper, Yunuo Chen, Minchen Li, Yu Fang, Ziyin Qu, Jiecong Lu, Meggie Cheng, and Chenfanfu Jiang. Anisompm: Animating anisotropic damage mechanics. *ACM Trans. Graph.*, 39(4), 2020. 2, 168

Peter Wriggers. Finite element algorithms for contact problems. *Archives of Comp. Meth. in Eng.*, 2, 12 1995. 63, 64, 67

C. Xiao-Chuan and D. Maksymilian. Domain decomposition methods for monotone nonlinear elliptic problems. In *Contemporary Math*, 1994. 33

Hongyi Xu, Funshing Sin, Yufeng Zhu, and Jernej Barbič. Nonlinear material design using principal stretches. *ACM Transactions on Graphics (TOG)*, 34(4):1–11, 2015. 186

Yonghao Yue, Breannan Smith, Peter Yichen Chen, Maytee Chantharayukhonthorn, Ken Kamrin, and Eitan Grinspun. Hybrid grains: adaptive coupling of discrete and continuum simulations of granular media. *ACM Transactions on Graphics (TOG)*, 37(6):1–19, 2018. 163

Eugene Zhang, Konstantin Mischaikow, and Greg Turk. Feature-based surface parameterization and texture mapping. *ACM Trans. on Graph.*, 24(1), January 2005. ISSN 0730-0301. 65

Changxi Zheng and Doug L. James. Energy-based self-collision culling for arbitrary mesh deformations. *ACM Trans. on Graph. (Proc. of SIGGRAPH 2012)*, 31(4), August 2012. 101

Y. Zhu, R. Bridson, and D. M. Kaufman. Blended cured quasi-newton for distortion optimization. *ACM Trans. on Graph (SIGGRAPH)*, 2018. 15, 31, 36, 38, 48, 55