Robust, Efficient, and Accurate Contact Algorithms

David Harmon

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2010

ABSTRACT

Robust, Efficient, and Accurate Contact Algorithms

David Harmon

Robust, efficient, and accurate contact response remains a challenging problem in the simulation of deformable materials. Contact models should robustly handle contact between geometry by preventing interpenetrations. This should be accomplished while respecting natural laws in order to maintain physical correctness. We simultaneously desire to achieve these criteria as efficiently as possible to minimize simulation runtimes. Many methods exist that partially achieve these properties, but none yet fully attain all three. This thesis investigates existing methodologies with respect to these attributes, and proposes a novel algorithm for the simulation of deformable materials that demonstrate them all. This new method is analyzed and optimized, paving the way for future work in this simplified but powerful manner of simulation.

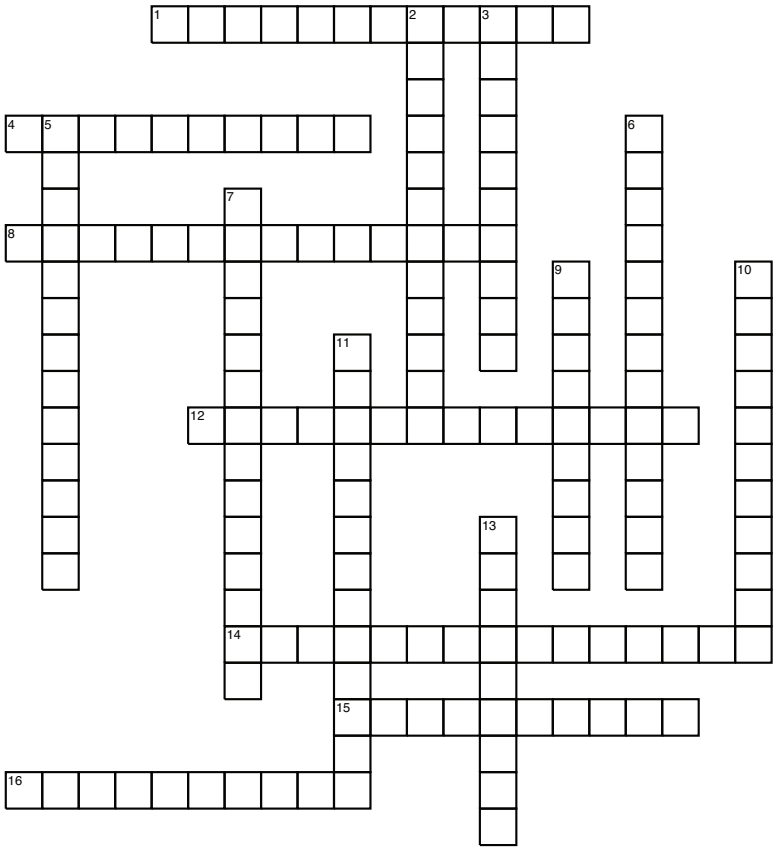
Table of Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgments



Across

1 My friend, for all the movie nights

4 My undergraduate professor, for all the advice and mentoring

8 My inside thesis committee member, for your perspective on graphics

12 My partner, for being my best friend

14 My sister, for always being there

15 My thesis chair, for asking the hard questions

16 My outside thesis committee member, for all the valuable feedback that made this a stronger thesis

Down

2 My friend and co-author, for the countless SIGGRAPH all-nighters

3 My father, for telling me I can do anything

5 My undergraduate advisor, for always being in my corner and excited about whatever I'm doing

6 My sister, for always thinking of me

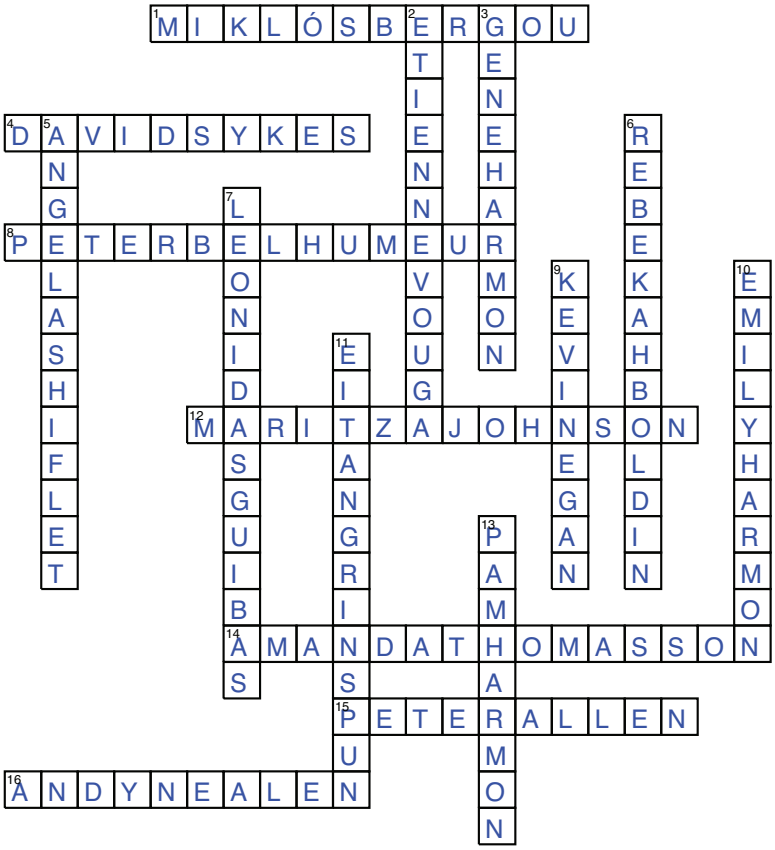
7 My outside thesis committee member, for always pushing me to dig deeper and find the core of a problem

9 My friend, for all the guitar jam sessions

10 My sister, for being my little buddy

11 My advisor, for his collaboration and support over the years

13 My mother, for always believing in me



Across

1 My friend, for all the movie nights

4 My undergraduate professor, for all the advice and mentoring

8 My inside thesis committee member, for your perspective on graphics

12 My partner, for being my best friend

14 My sister, for always being there

15 My thesis chair, for asking the hard questions

16 My outside thesis committee member, for all the valuable feedback that made this a stronger thesis

Down

2 My friend and co-author, for the countless SIGGRAPH all-nighters

3 My father, for telling me I can do anything

5 My undergraduate advisor, for always being in my corner and excited about whatever I'm doing

6 My sister, for always thinking of me

7 My outside thesis committee member, for always pushing me to dig deeper and find the core of a problem

9 My friend, for all the guitar jam sessions

10 My sister, for being my little buddy

11 My advisor, for his collaboration and support over the years

13 My mother, for always believing in me

Chapter 1

Introduction

For centuries physicists have described the behavior of galactic bodies, fluids, flexible surfaces, and many other physical materials. More recently, physical simulations have allowed us to imitate these natural processes on a computer. By modeling the laws of physics describing a system, we can recreate and study physical phenomena within a controlled environment. Such study has contributed to advancements in medicine, industrial design, engineering, and entertainment.

With these advancements comes a new set of challenges. Reproducing continuous processes on finite machines introduces additional research opportunities. This begins with the modeling process: describing the system one wishes to simulate by selecting the appropriately representative physical laws. This choice has implications when the simulation attempts to numerically solve the physical equations. A poorly chosen model often results in an unnecessarily slow simulation or one that is unable to capture the intended behavior.

One particularly difficult aspect is the modeling of collisions and contact within a simulation. A collision occurs when two points on one or more objects attempt to occupy the same location in space-time. This is impossible in nature, and thus a truly physical simulation must be equally strict. After a collision, these points often remain in contact, exerting equal and opposite forces on one another. These collision forces should not only be physically correct, *i.e.*, derived from physical laws, but they should also be adept in maintaining contact over time and keep objects well-separated (separated by at least machine precision). The simulation of physical contact models thus provides an intriguing computational

problem in addressing the efficiency of algorithms with such strict requirements.

This becomes most apparent in the simulation of deformable surfaces, such as cloth. With all deformable materials, every point on the body may collide with any other point, vastly increasing the computational effort required. Deformable surfaces are even harder, since their thickness ranges from extremely small to infinitesimal, and recovering from missed contacts can be a far more troublesome geometric problem than preventing them in the first place. Due to their particularly challenging nature, we focus on the simulation of cloth and thin shells, although the techniques and models developed apply equally to the simulation of many types of materials.

The contact model is responsible for most of the visual characteristics in a simulation, with collisions often acting as the stimuli for seemingly unrelated points of interest, such as the internal dynamics of a system. For instance, folds and wrinkles in a cloth simulation often develop in response to contact of the fabric with an object such as a character's body. Hence a high-quality model is important, even when contact is not the focus of a simulation.

This thesis is interested in models that encompass the following three properties, chosen for their ability to reliably produce animations of high-quality as well as high-fidelity:

1. The collision response model must be geometrically *robust*. Interpenetration of the mesh geometry results in a visually unsatisfying simulation, and is generally unacceptable for the strict requirements of applications such as feature film. Furthermore, geometry can become "hooked" together once it collides, causing even more problems in the remainder of the simulation. Thus, it remains important to maintain well-separated mesh geometry during the course of a simulation.

2. Additionally, the model must be physically *accurate*. In engineering applications, such as the development and testing of mechanical parts, a collision model that violates physical laws of nature is useless, no matter how robust it may be at separating mesh geometry. Thus, realistic models must strive to preserve known physical quantities conserved in nature, such as energy and momentum. This is also true when visual realism is the main goal. The preservation of physical invariants provides a quantifiable method of establishing a hard-to-identify quality: the physical behavior of

a simulation. These invariants supply quantifiable guidelines towards establishing a truly physically-behaving, and therefore looking, simulation.

3. Lastly, our contact models need to be computationally *efficient*. This becomes quite difficult in the face of the previous two qualities, further concretizing the challenges inherent in designing such models. To be considered reliable, a physically-based contact model must never halt when given a physical input, making consistent simulation progress. We further want our method to be efficient in the sense that it achieves maximum productivity with minimum wasted expense. Most of the computational effort is devoted to the collision detection, but the importance of a method that rapidly resolves impending collisions should not be underestimated.

1.1 Overview

In Chapter 2, we discuss existing physical models and evaluate them according to these criteria. We shall see that despite impressive progress over the years, there exists no method that achieves all of our requirements.

We address this concern by analyzing and extending existing methods of response (Chapter 3). In doing so, we see a fundamental limitation with the existing approach. We demonstrate that the problem must be revisited in order to construct a new method with our requirements in mind from the start.

We present a new model in Chapter 4 that satisfies all three requirements—the first to do so. We demonstrate how to implement this model, utilizing existing work in numerical methods and computational geometry in a novel manner (Chapter 5). Chapter 6 provides experimental and mathematical proofs that the proposed model retains our desirable properties.

Chapter 7 expounds upon the advantages provided by implementing the model on top of theoretically sound work from computational geometry. As a result, we are free to benefit from advancements in a separate area of research.

Chapter 8 offers analysis for evaluating and extending this method. We isolate and quantify variables in the system in order to improve efficiency, demonstrating its desirability

for applications ranging from computer graphics to mechanical engineering.

Chapter 2

Background

To begin, we evaluate existing classes of methods for dealing with collisions in a physical simulation. Collision response algorithms can be classified into three broad categories: penalty methods, constraint-based formulations, and impulse-based methods.

Each of these methods attempts to respond to collisions by modeling a physical force. During a contact scenario, the energy potential of the contact can be given as a function of separation distance between the contact points [Wriggers and Panagiotopoulos, 1999]. The key property of this function is that the potential rapidly increases to infinity as this distance approaches zero, as demonstrated in Figure 2.1. Intuitively, this illustrates the idea that penetration in nature is disallowed. Each method discussed attempts to replicate this concept.

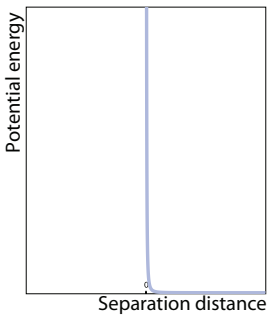


Figure 2.1: As the distance approaches zero, the potential energy escalates to infinity.

2.1 Penalty methods

The first class of collision response is penalty methods. Penalty methods operate by penalizing contact between two objects. This penalty can be given as any number of functions that attempt to replicate the physical energy potential of Figure 2.1.

Figure 2.2 offers a few choices of potentials. A penalty potential must increase monoton-

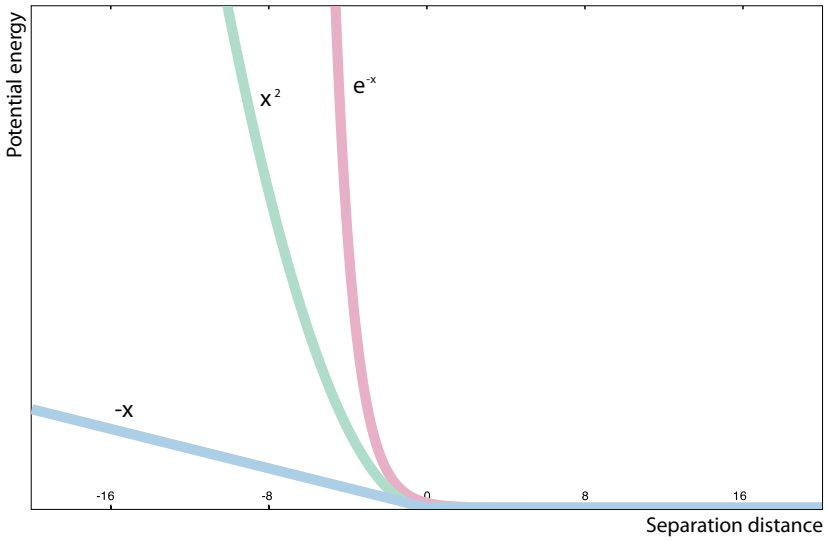


Figure 2.2: As the separation distance decreases (and the objects overlap), each of these contact potentials increases monotonically.

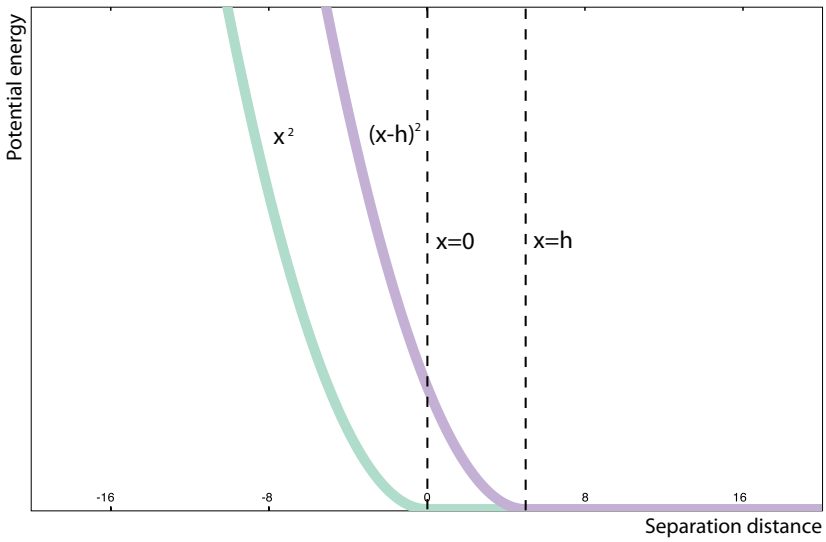
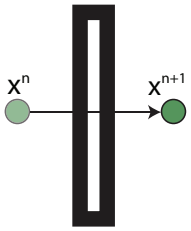


Figure 2.3: The potential is shifted and becomes non-zero when the objects are close, rather than in contact.

ically as penetration depth increases. Traditionally, the potential takes a value of zero when separation depth is non-negative and becomes active when penetration occurs. However, it is certainly possible to shift this potential (Figure 2.3), enabling contact when the objects become *close* rather than actually touching. Ideally, this allows the contact force to repel the objects before actual penetration occurs. This becomes necessary in the simulation of cloth, where overlap is disastrous, and difficult to repair [Baraff *et al.*, 2003]. A surface has no volume and lacks any notion of inside or outside, so a missed collision no longer has a reliable direction to apply response forces, leaving the cloth permanently tangled [Volino and Magnenat-Thalmann, 2006].

This reveals the first flaw with penalty methods. Before the penalty resolves the contact, some overlap may occur [Baraff, 1989]. As the overlap increases, the force from the potential grows and acts against further penetration. Even if the potential is shifted, there is no guarantee that penetration will not occur. In some cases with narrow objects (and especially surfaces), objects can completely pass through one another in a single timestep, without any force acting to stop it (see inset figure). Thus, penalty layers lack robustness against interpenetrations.



Some choices of potentials (such as e^{-x} [Terzopoulos *et al.*, 1987]) are highly non-linear, and rapidly accelerate to infinity as separation distance decreases. This is desired, as it models the physical contact energy. However, as this distance diminishes, it becomes computationally infeasible to integrate the force derived from these energies. The timestep must continuously decrease for the force to be approximated by traditional solvers [Hauth *et al.*, 2003]. Due to their highly nonlinear nature–the very property that makes them desirable–a predetermined stable timestep cannot be found. In the worst case, the force becomes unstable and "blows up." In most cases, however, the timestep is still too large to guarantee resolution of the contact, and the objects pass through one another. Implicitly integrated penalty forces are an option for improving stability, but their dynamically activated nature eliminates the possibility of a Jacobian with a fixed sparsity pattern. The matrix must then be rebuilt every timestep and cannot be prefactored. Adaptive timestepping is also an option for improving stability, although neither this nor implicit timestepping provide any

guarantee of robustness, only stability.

This is the second problem with penalty methods. An associated stiffness must be chosen pre-simulation, but this stiffness may not be adequate to resolve all collisions. Increasing the stiffness requires starting the simulation anew, and with a smaller timestep (due to stability criteria). Adjusting this stiffness "on the fly" destroys conservation and is unpredictable. In short, for any chosen stiffness, there exists an impact of sufficiently high velocity such that the force fails to stop the collision, allowing objects to *tunnel* through one another [Bridson *et al.*, 2002].

Due to the stability problems associated with non-linear penalties, simple linear forces (quadratic potentials) are often used in practice, including in the simulation of volumes [Teran *et al.*, 2005; Faure *et al.*, 2008]. These are easy to implement and simple to use, but still must withstand the two main flaws of penalty methods. The penalty method's physical basis (a conservative force derived from a contact potential) makes it a popular choice where physical correctness is important, as it is in the engineering world [Wriggers and Laursen, 2007]. In practice, shifted penalty forces can perform adequately with deformable volumes and / or sufficiently small timesteps that prevent tunneling. However, its poor robustness properties have thus far made it unsuitable as the sole method of response in graphics applications.

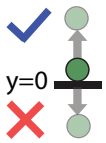
2.2 Constraints

Penalty methods are one approach to solving general constrained optimization problems, so it is natural that their shortcomings would lead to the use of other algorithms designed for handling these problems [Fletcher, 1987]. Such algorithms work by attempting to satisfy *constraint* equations, or real-valued functions of the configuration ($C(\mathbf{q}) \in \mathbb{R}$, where $\mathbf{q}$ is a point in configuration space) that represent some physical or geometric property to be preserved [Lanczos, 1986].

These constraints may be strict equality constraints, called *bilateral* or two-sided constraints ($C(\mathbf{q}) = 0$), or inequality constraints, called *unilateral* or one-sided constraints ($C(\mathbf{q}) \leq 0$).

Bilateral constraints arise when any relationship between geometry must be strictly enforced. Inextensibility constraints as well as ball and socket constraints are of this form, where drift resulting in geometric separation is undesirable [Goldenthal *et al.*, 2007; Barzel and Barr, 1988].

Unilateral constraints allow motion freely within some region of configuration space defined by the constraint function, but restrict any motion which attempts to leave this space. Contact constraints lie in this category. In the inset figure, a particle collides with a floor. Downward motion is forbidden, while motion up (or in any other direction) is permitted, making this a one-sided, unilateral constraint. In this figure, the constraint is represented by requiring that the vertical axis remain non-negative, or $C(x, y) = -y \leq 0$.



The most popular method for satisfying constraints is the method of Lagrange multipliers. Lagrange multipliers are used when minimizing or maximizing a function subject to a set of constraints [Hadley, 1964]. In a physical simulation, these functions are the equations of motion being integrated, with additional variables called *Lagrange multipliers* that act as constraint-maintaining forces:

$$M\ddot{\mathbf{q}} - F(\mathbf{q}) + \lambda^T[\nabla C_1 \nabla C_2 ... \nabla C_k]^T = 0 \tag{2.1}$$

$$\forall i \; C_i(\mathbf{q}) \leq 0 \tag{2.2}$$

$$\forall j \; C_j(\mathbf{q}) = 0. \tag{2.3}$$

$\ddot{\mathbf{q}}$ is the acceleration, F represents the forces in the system (both internal and external), and C_i are the set of k inequality contact constraints to be met. The multipliers (λ_i) are the magnitude of the constraint-maintaining force, while ∇C_i is the direction in which the force is applied. Together these represent a force that exactly counteracts the violating motion.

Equations 2.1, 2.2, and 2.3 are the continuous constrained equations of motion. In order to solve numerically, they must be discretized. Just like the equations of motion, there is a choice of how to discretize the constraint equations.

Explicit For the timestep $(\mathbf{q}^n, \mathbf{q}^{n+1})$, explicit constraints compute corrections based on violations of the positions from the *beginning* of the timestep, *i.e.*, by measuring how much

$C_i(\mathbf{q}^n)$ is violated [Witkin *et al.*, 1990; Barth *et al.*, 1995]. This makes solving the constrained system much simpler, since all dependent variables of the constraint function are known [Ryckaert *et al.*, 1977; Andersen, 1983]. However, this discretization suffers from configurations that drift from the constraint manifold, leading to a number of corrective regularization and stabilization schemes [Baumgarte, 1972; Lin and Huang, 2002; Cline and Pai, 2003]. Furthermore, the solution may very well leave the configuration in an inadmissible (penetrating) state at the end of the timestep [Hong *et al.*, 2005]. Thus, explicit constraints are generally not reliable enough for contact resolution.

Implicit An implicit discretization maintains constraints that are satisfied at the end of the timestep: $C_i(\mathbf{q}^{n+1}) \leq 0$. An implicit discretization ensures that the contact constraints are maintained at all times. Unfortunately, solving a constrained system with implicit inequality constraints is a very difficult non-linear problem, and has not yet been shown to be practical for resolving collisions. Under some circumstances, such as those involving fixed or prescribed geometry, the constraints are linear functions of position and a solution is easier to obtain [Baraff and Witkin, 1998].

First-order Another option, motivated by Baraff and Witkin [1998], is to linearize the constraint equation to obtain a first-order approximation of the fully-implicit constraint [Hong *et al.*, 2005]. This does not have the drift problem of explicit constraints, but may not always be robust enough for the demands of contact applications.

2.3 Impulses

Revisiting the original contact potential of Figure 2.1, we see that the increase in energy as a function of separation is dramatic. In fact, to the unaided observer, this potential appears to act instantaneously, as the increase can hardly be detected. Another class of methods utilizes this observation, treating contact forces as if they occur instantaneously. A force that acts instantaneously is called an *impulse* [Fowles and Cassiday, 1962]. Treating the forces in this way no longer results in changes in acceleration, but rather direct changes in momentum. When collisions are detected in the system, the troublesome velocities are

directly modified to resolve the collision [Mirtich and Canny, 1995a].

The choice of impulse may vary. However, a physically-based model is usually preferred. In this light, impulses that conserve momentum between two colliding bodies can be found. For instance, consider a collision between two bodies, A and B. The bodies approach along some direction $\hat{n}$, called the *collision normal*, which defines the tangent plane of the contact. We can then write the incoming relative velocity of the collision as

$$(v_A - v_B) \cdot \hat{n}.$$

We orient the normal to point from B to A so that a negative relative velocity indicates the bodies are approaching, although this choice is arbitrary.

Letting primes denote post-collision quantities, conservation of momentum dictates that

$$(v'_A - v'_B) \cdot \hat{n} = -e(v_A - v_B) \cdot \hat{n},$$

where e is the coefficient of restitution of the collision. $e = 1$ reflects a *perfectly elastic* collision where no kinetic energy is lost, while $e = 0$ indicates a *perfectly inelastic* collision, with maximum dissipation.

To find the impulse that resolves this collision, we write the primed velocities in terms of the pre-collision velocities with some impulse J applied:

$$v'_A = v_A - \frac{J}{m_A}\hat{n}$$
$$v'_B = v_B + \frac{J}{m_B}\hat{n}.$$

Through substitution and rearranging the momentum-conserving equation, we arrive at a value for the impulse

$$J = \frac{(1+e)(v_A - v_B) \cdot \hat{n}}{\frac{1}{m_A} + \frac{1}{m_B}}.$$

This impulse acts entirely in the normal direction, which can be separated from the tangential sliding direction [Cirak and West, 2005]. This becomes non-trivial for more than two bodies [Huh *et al.*, 2001], so impulses delivered in this way are usually performed iteratively per colliding pair [Hahn, 1988; Mirtich and Canny, 1995b]. However in doing this, it is easy to get caught in a configuration that cannot be resolved, as illustrated in Figure 2.4.

This remains the limitation of momentum-conserving impulses; there are configurations which will cause this method to halt the simulation, unable to move forward in time without violating a collision.

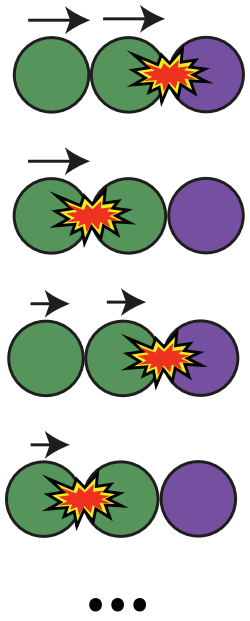


Figure 2.4: Resolving simultaneous inelastic collisions using iterative impulses. The rightmost particle is fixed, and is struck by two simulated particles moving together. Every odd iteration k results in the initial setup, with velocities a fraction ($\frac{1}{2^{k-1}}$) of the original. This geometric series never converges, and the simulator cannot move forward in time without resulting in an immediate penetration.

Chapter 3

Inelastic projection

Many of the shortcomings of existing methods, such as iterative impulses, arise due to the large number of simultaneous collisions and the method's inherently local manner of dealing with them. Satisfying constraints using the method of Lagrange multipliers is intended to deal with this issue, but can be difficult to robustly solve in practice [Wriggers, 1995]. Existing penalty methods unfortunately lack the robustness needed to simulate highly-deformable materials such as cloth in an appealing way.

This chapter builds on work in impulse-based methods, in particular work concerning simultaneous collisions. The state-of-the-art's main shortcomings are addressed, allowing straight-forward simulation of a wide range of materials.

3.1 Previous work

Since introduced by Bridson *et al.* [2002], velocity filter response schemes have grown in popularity [Bridson *et al.*, 2003; Guendelman *et al.*, 2003; Sifakis *et al.*, 2008; McAdams *et al.*, 2009]. Bridson *et al.* [2002] introduced a complete framework for responding to collisions in cloth simulations by entirely decoupling the process from the integration of forces. This approach has been widely adopted in the graphics community.

Algorithm 1 describes the velocity filter framework. The system linearizes all motion over a timestep by computing an average velocity, then filters offending velocities to attain an end-of-step configuration satisfying some criteria, in this case absence of penetrations.

This assumes that the initial configuration is free of penetrations.

Algorithm 1 Velocity filter

while simulating **do**
  $\mathrm{x}^n = \mathrm{x}^{n+1}$
  $\mathrm{v}^n = \mathrm{v}^{n+1}$
  $\{\mathrm{x}^{n+1}, \mathrm{v}^{n+1}\} = dynamics(\mathrm{x}^n, \mathrm{v}^n, \Delta t)$
  $\mathrm{v}^{n+\frac{1}{2}} = \frac{\mathrm{x}^{n+1} - \mathrm{x}^n}{\Delta t}$
  $\mathrm{v}^{n+\frac{1}{2}} = filter(\mathrm{x}^n, \mathrm{v}^{n+\frac{1}{2}})$
  $\mathrm{x}^{n+1} = \mathrm{x}^n + \Delta t \mathrm{v}^{n+\frac{1}{2}}$
  $n = n + 1$
end while

The actual response presented by Bridson *et al.* [2002] inside the *filter* procedure is comprised of three stages:

1. **Penalties**

   The first pass of the algorithm applies weak penalty forces. They are not forces in the true sense, *i.e.*, they are not integrated as part of the system. Instead, the change in velocity due to a linear spring is directly computed and accumulated as an impulse:

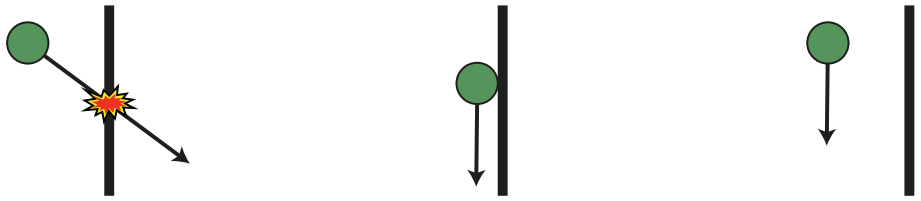
   $$J = -kd\Delta t,$$

   where Δt is the timestep size, k is the penalty stiffness, and d is the overlap distance.

   These springs are weak, in that they readily handle resting contact, but lack the robustness necessary to resolve high-speed impact. The idea is that this stage will resolve the "easy" collisions, with only the persistent making it to step two. This stage has the additional problem discussed in §2.1 of choosing the stiffness k.

2. **Iterative impulses**

   The second pass of the algorithm iteratively applies the pairwise, momentum-conserving impulses discussed in §2.3. Note that due to the velocity filter process, the response



(a) A collision occurs during a timestep.

(b) The correct solution timesteps to the collision and corrects the velocity.

(c) Velocity filters assume the collision is instantaneous and respond "early".

Figure 3.1: Velocity filters introduce errors by treating all collisions as simultaneous.

appears to be activated *before* contact, since the average velocity is directly modified. This means a collision can appear to occur without the objects actually being in contact (Figure 3.1).

This step handles the high-speed impact missed by the penalty forces. Continuous time collision detection is used to guarantee no collisions occurring during the timestep go undetected, contrasted with the static proximity tests used in stage one.

As discussed previously, these iterative impulses may not always find a collision-free solution. Furthermore, each iteration can be expensive, requiring collision detection using the updated velocities. For these reasons, the algorithm terminates the iterations after a fixed, user-defined number. This possibly leaves penetrations and necessitates a third, robust pass in the algorithm.

3. **Rigid impact zones**

Each stage in the algorithm applies increasingly aggressive response, acting as a preconditioner for those down the line. The penalty springs efficiently process low-impact situations such as resting contact, while the iterative impulses handle high-impact points of contact. The last stage must robustly resolve the remaining sets of simultaneous collisions. Bridson *et al.* [2002] use the method of *rigid impact zones* originally introduced by Provot [1997].

Rigid impact zones group collisions into disjoint sets based on shared vertices, called *impact zones* (see Figure 3.2). The rigid body motion of this zone is extracted and

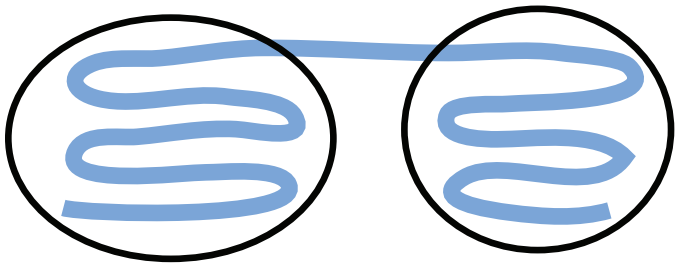


Figure 3.2: Two islands of disjoint impact zones

then assigned as the velocity of the vertices contained within. This moves the zone as a rigid body, guaranteeing no collisions within the region. Algorithm 2 describes how the velocities in an impact zone are prescribed.

However, there may be collisions with nodes outside the region, so the procedure must iterate, growing the impact zones. The algorithm's guarantee comes in the limit case, where the entire mesh is a single zone, moved as a rigid body. While this limit case is highly unlikely, it is worthwhile as a proof of guaranteed algorithm convergence.

Algorithm 2 Rigid impact zones

1: $x_{CM} = \frac{\sum_i m_i x_i^n}{\sum_i m_i}$

2: $v_{CM} = \frac{\sum_i m_i v_i^{n+\frac{1}{2}}}{\sum_i m_i}$

3: $L = \sum_i m_i (x_i^n - x_{CM}) \times (v_i^{n+\frac{1}{2}} - v_{CM})$

4: $\mathbf{I} = \sum_i m_i (|x_i^n - x_{CM}|^2 I_3 - (x_i^n - x_{CM}) \otimes (x_i^n - x_{CM}))$

5: $\omega = \mathbf{I}^{-1} L$

6: $v_i^{n+\frac{1}{2}} = v_{CM} + \omega \times (x_i^n - x_{CM})$

Shortcomings This algorithm has proved quite effective at resolving self collisions in cloth simulation. So effective, in fact, that the basic framework has been adapted to rigid body simulation [Guendelman *et al.*, 2003] and hair simulation [Selle *et al.*, 2008; McAdams *et al.*, 2009]. Furthermore, in practice this algorithm is used to resolve collisions between cloth and scripted objects in the scene, beyond its original intent of resolving only self-collisions.

The first two passes are easily adapted to use with non-simulated objects. Vertices in

scripted objects are immovable, described in the system as having infinite mass. This has no effect on the penalty forces, or on the impulse equations of §2.3. However, this proves particularly disruptive in the rigid impact zones.

$$x_{CM} = \frac{\sum_i m_i x_i^n}{\sum_i m_i} \tag{3.1}$$

Equation 3.1 computes the center of mass of the clump of vertices. When a node of infinite mass is added to the system its location determines this expression. This filters down to the rest of the rigid impact zone computation, resulting in this fixed object dominating the zone, directly dictating the behavior. This means if the object motionless, the impact zone is not allowed to move either.

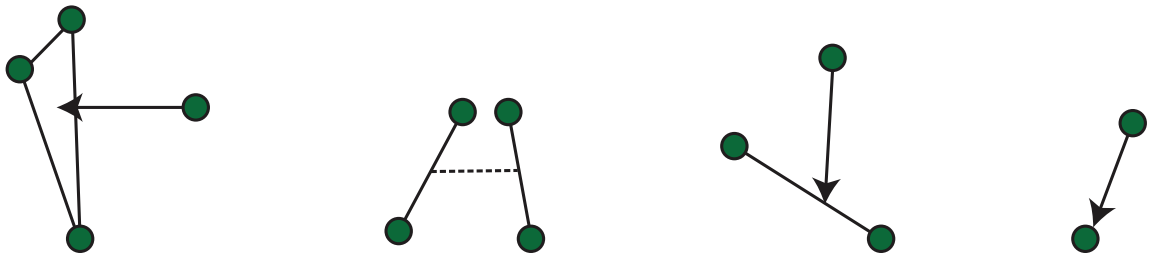
Worse, in the case of multiple scripted objects in the same impact zone, if their motions disagree then no solution can be found to move the impact zone rigidly. The only solution is to take a smaller timestep and hope collisions are resolved by the penalty springs or geometric impulses before rigid impact zones are required. However, no guarantee is provided.

3.2 Fail-safe

Algorithm 3 Fail-safe algorithm

```
1: while new collisions detected do
2:    Insert each new collision into its own impact zone (IZ)
3:    Merge all new and existing IZs that share vertices
4:    for each impact zone (IZ) do
5:       Reset IZ to pre-response velocities
6:       Apply fail-safe
7:    end for
8: end while
```

The core problem is not in the velocity filter algorithm, but rather in specifically treating the colliding mass as a rigid body. In this light, we propose to generalize the third pass of the algorithm. We accomplish this by slightly modifying the rigid impact zone method;



(a) Vertex-Triangle (b) Edge-Edge (c) Vertex-Edge (d) Vertex-Vertex

Figure 3.3: The four types of collisions in a deformable triangle mesh.

instead of rigidifying the motion, we replace it with a generic *fail-safe*, a method designed to resolve all collisions within an impact zone. Algorithm 3 gives pseudocode for this new, generalized procedure.

This fail-safe could be completely unphysical, such as zeroing all velocities or applying rigid impact zones. Instead, we revisit the idea of constraints to devise a new method that addresses the shortcomings of rigid impact zones.

3.3 Constraints

General constraints were discussed in §2.2. Here we will introduce the constraints needed to resolve contact between deformable, triangle meshes. We can then satisfy these constraints using the method of Lagrange multipliers (§3.5).

Collision types In the simulation of deformable, triangle meshes there are four types of collisions: vertex-triangle, edge-edge, vertex-edge, and vertex-vertex. Figure 3.3 illustrates each of these types.

Usually, however, two of these types are deemed redundant, by considering vertex-vertex and vertex-edge as degenerate cases of vertex-triangle and edge-edge. For instance, a vertex-vertex collision can be thought of as nothing more than a vertex-triangle collision, where the vertex collides with one of the corners of the triangle. We make use of this simplifying assumption.

Configuration space When discussing the motion and interaction of many vertices, it makes sense to speak in terms of configuration space. Rather than dealing with n vectors in $\mathbb{R}^3$, we treat the configuration of the system as a single point $\mathbf{q}$ in configuration space $\mathbb{Q} = \mathbb{R}^{3n}$, contact constraints involve many interacting degrees-of-freedom, so writing them as functions of $\mathbf{q}$ facilitates computation. As these constraints are developed, we will make note of any parallels to more familiar notions in $\mathbb{R}^3$.

Collision constraints We view each collision as the violation of a real-valued constraint function, $C(\mathbf{q})$, with $C(\mathbf{q}) > 0$ whenever $\mathbf{q}$ is an inadmissible (penetrating) configuration. This is consistent with the previous discussion of unilateral contact constraints being satisfied when $C(\mathbf{q}) \leq 0$. Through collision detection, we associate one constraint function to each impending vertex-triangle or edge-edge collision, taking care to avoid duplicates during degenerate vertex-edge and vertex-vertex contacts.

The constraint for an impending collision between the triangle $(X_1(t), X_2(t), X_3(t))$ and the vertex $X_4(t)$ (Figure 3.3(a)) is

$$C_{\text{vt}}(\mathbf{q}) = N \cdot [X_4 - (\alpha_1 X_1 + \alpha_2 X_2 + \alpha_3 X_3)] \ , \tag{3.2}$$

where $N(t)$ is the familiar $3D$ collision normal, and the three scalars $\alpha_1(t), \alpha_2(t)$, and $\alpha_3(t)$ are the barycentric coordinates of the projection of $X_4(t)$ onto the plane spanned by the triangle. Similarly, the constraint for an impending collision between the edges $(X_1(t), X_2(t))$ and $(X_3(t), X_4(t))$ (Figure 3.3(b)) is

$$C_{\text{ee}}(\mathbf{q}) = N \cdot [(\alpha_3 X_3 + \alpha_4 X_4) - (\alpha_1 X_1 + \alpha_2 X_2)] \ , \tag{3.3}$$

where $\alpha_1(t), \alpha_2(t), \alpha_3(t)$ and $\alpha_4(t)$ are the parametric values of corresponding closest points on the first and second edge, respectively. We refer to the rate of change of the constraint function, $\dot{C}$, as the *normal velocity*, a high-dimensional analogue to the usual relative velocity in the direction of the collision normal.

Unlike collision detection and response algorithms that solve these constraints as a function of time [Snyder *et al.*, 1993], we formulate them primarily to reason about and resolve collisions in configuration space. While each constraint depends on only four vertices (called the *stencil*), it is formally a scalar function of $\mathbf{q}$, and can be differentiated with

(a) A convex admissible region

(b) A concave admissible region

Figure 3.4: The importance of convexity in constraint satisfaction

respect to $\mathbf{q}$ to yield the *constraint gradient*, ∇C, a row vector in configuration space. By the chain rule, we may rewrite the normal velocity as $\dot{C} = \nabla C \dot{\mathbf{q}}$. For a vertex-triangle collision, the constraint gradient expressed in the local indices of the stencil is

$$\nabla C_{\text{vt}} = (-\alpha_1 N, -\alpha_2 N, -\alpha_3 N, N) \; ; \tag{3.4}$$

for an edge-edge collision, the gradient in local indices is

$$\nabla C_{\text{ee}} = (-\alpha_1 N, -\alpha_2 N, \alpha_3 N, \alpha_4 N) \; . \tag{3.5}$$

To elevate the constraint gradient to configuration space, we simply map the local indices to their global positions, with zeros everywhere else, in the style of finite element stiffness matrix assembly.

3.4 Convexity

When defining constraint functions, the most important problem is not whether the function is linear or non-linear, but rather whether the function is convex or not [Boyd and Vandenberghe, 2004]. A convex constraint of the form $C(\mathbf{q}) \leq 0$ defines a convex region of admissible configuration space.

Figure 3.4(a) illustrates a convex optimization problem. We are attempting to minimize the scalar function defined over the rectangular domain, but are constrained to the circular region. Solving problems of this form are relatively easy, since a local minima is also a global minima, it is harder for the algorithm to become stuck.

Contrast this with Figure 3.4(b), a concave constraint on the same domain. For this particular example, local minima may be found within convex "pockets" of admissible space, with no trivial manner of escaping.

The collision constraints of Equations 3.2 and 3.3 are non-linear functions of the configuration. As written, however, they are linear functions of the 3-dimensional collision normal $N(t)$ (which greatly simplifies evaluation of the constraint gradient). Linear functions are both convex and concave, so the convexity of these constraints depends on the choice of $N(t)$.

Vertex-triangle constraints For vertex-triangle constraints, the triangle normal provides a good choice for $N(t)$, as it is well-defined (assuming non-degenerate geometry) and results in a convex constraint.

Collision constraints are usually defined in this manner: a single contact point colliding with a plane [Baraff, 1989]. This may result in error, as discussed by Egan *et al.* [2003]; all motion in the normal direction is restricted, so the vertex may drift past the triangle as if the spanning plane were actually present.

Edge-edge constraints Choosing a well-defined direction for edge-edge constraints proves more challenging. Defining the normal as the cross product between the two colliding edges is inadequate and results in a non-convex constraint. When the two edges are parallel, the normal vanishes, driving the value of the constraint to zero, even when the edges are clearly separated.

This can be seen in Figure 3.5, which plots the value of an edge-edge constraint for a 2-dimensional subset of the 12-dimensional configuration space. While C_{ee} is a function of configuration space, it only actually uses the positions of four vertices (two for each edge), so it is a function of 12 variables (assuming three degrees-of-freedom for each vertex).

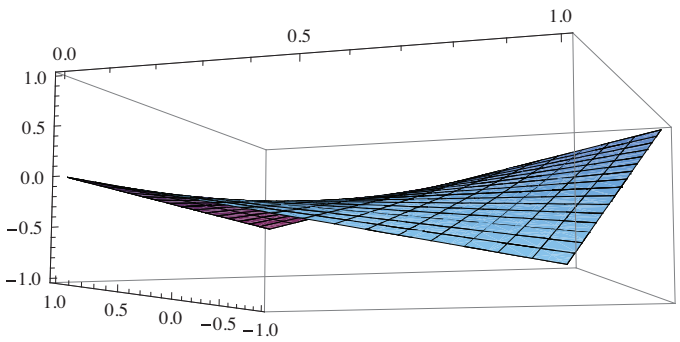


Figure 3.5: The function is non-convex, which means the admissible region is not convex.

The plot is for the function C_{ee}, where $N(t)$ is defined as the cross product between the two edges $(X_1(t), X_2(t))$ and $(X_3(t), X_4(t))$, with the values

$$X_1 = (0, 0, 0)$$

$$X_2 = (0, 1, 0)$$

$$X_3 = (x, 0, 0)$$

$$X_4 = (1, 1, y).$$

We vary x from -1 to 1 and y from 0 to 1, so that the configuration passes through a point where the edges are parallel. There is a saddle point at this configuration and the normal $N(t)$ vanishes, clearly making this constraint non-convex.

Defining a convex edge-edge constraint remains an open problem. Additionally, constraints must be developed on the α_i weights for true edge-edge and vertex-triangle contact. Otherwise, even if the primitives slip past one another, they will remain constrained as if the edges were infinite lines and the triangle an entire plane.

3.5 Linear complementarity problem (LCP)

We begin by first reproducing the collision-resolving impulses of §2.3, but in terms of the more general constraint functions.

Impulse response for a single constraint Let unprimed and primed quantities refer to the *pre-* and *post*-response states. A *valid* response satisfies two properties. First, it pushes colliding objects apart by applying a nonnegative impulse along the constraint direction, *i.e.*, $\mathbf{p} - \mathbf{p}' = \nabla C^T \lambda$ for some (unknown) nonnegative multiplier λ. Second, since the impulse should prevent colliding points from approaching further, it must lead to a nonnegative post-response normal velocity $\nabla C \dot{\mathbf{q}}' = \nabla C \mathrm{M}^{-1} \mathbf{p}'$.

Henceforth we are interested in finding an inelastic response, although extending to arbitrary coefficients of restitution is straightforward [Baraff, 1994]. For an inelastic collision, we seek the maximally dissipative response, per the definition of purely inelastic. Thus, we minimize post-response kinetic energy, $\frac{1}{2}\|\mathbf{p}'\|^2_{\mathrm{M}^{-1}} = \frac{1}{2}\|\mathbf{p} + \nabla C^T \lambda\|^2_{\mathrm{M}^{-1}}$, with respect to λ, which yields

$$\lambda = \frac{-\nabla C \dot{\mathbf{q}}}{\langle \nabla C, \nabla C \rangle_{\mathrm{M}^{-1}}} \ .$$

This is analogous to the impulses of §2.3. One property of the above inelastic response is that the normal velocity vanishes along the constraint direction: $\nabla C \dot{\mathbf{q}}' = 0$.

Impulse response for multiple constraints Cirak and West [2005] treat the above case of a single constraint in detail. We, however, are interested in the case of k simultaneous constraints, *i.e.*, k vertex-triangle and edge-edge collisions with possibly non-disjoint stencils. These collisions come from the system's collision detection subsystem, and can change between timesteps as primitives collide and separate. Now let $\nabla C = [\nabla C_1^T, \ldots, \nabla C_k^T]^T$ be a $k \times 3n$ matrix whose rows span the possible impulse directions. For any vector $\boldsymbol{\lambda} \in \mathbb{R}^k$, $\mathbf{p}' = \mathbf{p} + \nabla C^T \boldsymbol{\lambda}$ corresponds to the application of a linear combination of collision impulses to $\mathbf{p}$. As in the single-constraint case, we require that the $\lambda_1 \ldots \lambda_k$ be nonnegative, since constraint impulses can push but not pull, and that the impulse yields nonnegative post-response relative velocities, *i.e.*, that every row of $\nabla C \dot{\mathbf{q}}' = [\nabla C_1 \dot{\mathbf{q}}', \ldots, \nabla C_k \dot{\mathbf{q}}']^T$ be nonnegative.

Solving for the $\boldsymbol{\lambda}$ that minimizes kinetic energy, subject to the above constraints on $\boldsymbol{\lambda}$ and normal velocity, can be formulated as a linear complementarity problem (LCP) [Moreau, 1988], solvable using methods such as those described by Cottle, Pang, and Stone [1993]. For multiple collisions with overlapping stencils, some response impulses may be redundant, *i.e.*, responding to a subset of all the collisions may satisfy all collision constraints. The LCP approach ensures that redundant collisions do not yield pulling impulses ($\lambda_i < 0$).

LCPs are well-studied in rigid body literature [Moreau, 1988; Baraff, 1994; Stewart and Trinkle, 1996], and have even been used to resolve contact between quasi-rigid bodies [Pauly *et al.*, 2004]. However, little work has been done in applying them to the simulation of deformable bodies.

Solving an LCP proves difficult in practice. The large number of possible contact points, with strongly related constraint gradients, is stressful for solvers. We tested several examples with implementation of Dantzig's simplex algorithm [Cormen *et al.*, 2001] and Lemke's algorithm [Hillier *et al.*, 2004] for solving LCPs. Unfortunately, we found both algorithms unsatisfactory for the demands of cloth simulation, with slightly better performance from Dantzig's iterative method. In retrospect, this seems consistent with findings in rigid body research, where iterative methods such as Dantzig's are usually preferred for their additional robustness [Baraff, 1994].

More investigation is needed for exactly why these solvers fail, but work in rigid bodies has shown that LCP solvers can be exacting, both in terms of the input and computational effort. Furthermore, the velocity filter process may cause additional strain in finding a solution, as it increases the number of contacts considered simultaneous. Treating contact in order seems intractable for cloth simulation, as even more recent rigid body research has drifted from this approach [Milenkovic and Schmidl, 2001; Guendelman *et al.*, 2003; Kaufman *et al.*, 2008]. In particular, the addition of friction, a necessary parameter for modeling a range of materials, is particular troublesome for contact solutions formulated in this manner [Kaufman *et al.*, 2005].

LCPs are a form of the more general quadratic programming (QP) problem, where the minimum is unknown and no complementarity condition is present. While quadratic programming problems are more general, it remains to be seen if this class of solvers can

be more successful in addressing the problem.

3.6 Bilateral projection

We still require a method of resolving simultaneous contacts without the side effects of rigid impact zones. Thinking of the problem in terms of a fail-safe, we see that we are looking for a solution to a set of tightly clustered collisions. When the third pass is reached, the "easy" collisions have been resolved, leaving only the persistent.

In this light, we relax the unilateral constraint requirement and instead resolve collisions bilaterally. This approximates $\boldsymbol{\lambda}$ by assuming that post-response relative velocities are exactly zero: $\nabla C\dot{\mathbf{q}}' = 0$. This may result in some $\lambda_i < 0$, introducing artificial "sticking." This simplification affects only the fail-safe; since Bridson *et al.*'s framework resorts to the fail-safe only after many iterations of repulsive impulses that are designed to prevent sticking, we have not observed any significant sticking artifacts.

If we relax the conditions on a response being *valid* to allow both positive and negative entries in $\boldsymbol{\lambda}$, then $\dot{\mathbf{q}}'$ is the minimizer of

$$\|\dot{\mathbf{q}} - \dot{\mathbf{q}}'\|_M^2 \ , \quad \text{subject to} \quad \nabla C\dot{\mathbf{q}}' = \mathbf{0} \ .$$

This minimization projects the velocity onto the orthogonal complement of the span of the columns of ∇C^T. Hence we call this the *inelastic projection*. We may repose the above as an extremization of the augmented functional

$$W(\dot{\mathbf{q}}', \boldsymbol{\lambda}) = \frac{1}{2}\|\dot{\mathbf{q}}' - \dot{\mathbf{q}}\|_M^2 + (\nabla C\dot{\mathbf{q}}')^T\boldsymbol{\lambda} \ ,$$

with respect to $(\dot{\mathbf{q}}', \boldsymbol{\lambda})$, where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers. The corresponding stationary equations are

$$0 = D_1 W(\dot{\mathbf{q}}', \boldsymbol{\lambda}) = \mathbf{p}' - \mathbf{p} + \nabla C^T\boldsymbol{\lambda} \ , \tag{3.6}$$

$$0 = D_2 W(\dot{\mathbf{q}}', \boldsymbol{\lambda}) = \nabla C\dot{\mathbf{q}}' \ . \tag{3.7}$$

Here D_i denotes the derivative with respect to the i-th argument. Equation (3.6) guarantees that the response acts only along the ∇C direction, and (3.7) ensures vanishing normal

velocities. Substituting (3.6) into (3.7), it follows that we can recover the inelastic response for a set of k simultaneous collisions by solving the linear system

$$\nabla C M^{-1} \nabla C^T \boldsymbol{\lambda} = \nabla C \dot{\mathbf{q}} \tag{3.8}$$

for $\boldsymbol{\lambda}$, and then substituting it into (3.6) to obtain the unique post-inelastic-collision momentum, $\mathbf{p}'$. Solving this linear system is far simpler than satisfying the inequality constraints.

In theory, the length of the gradient vectors does not matter, but we normalize the constraint gradients to improve numerical robustness. Even so, when collision detection creates a set of nearly parallel gradients, the above linear system can be poorly conditioned or even singular. Thus we recommend the use of an iterative solver intended for near-singular systems when solving (3.8).

3.7 Friction

For the realistic simulation of a wide range of materials, a contact model requires a method of implementing friction. The most common model used is the Coulomb model, which we incorporate into inelastic projection. Coulomb's model states that static friction is proportional to the normal contact force, while dynamic (kinetic) friction is limited by the relative tangential motion of the contact.

The complex interaction between collisions requires the careful selection of the direction in which friction is applied, lest more collisions be instigated. We apply friction per impact zone by again restricting $\mathbf{q}$ to be the configuration subspace corresponding to the vertices of the impact zone. Vertices outside the impact zone remain untouched.

Building on Cirak and West [2005], we decompose the configurational velocity as

$$\dot{\mathbf{q}}' = \dot{\mathbf{q}}'_{\text{fix}} + \dot{\mathbf{q}}'_{\text{norm}} + \dot{\mathbf{q}}'_{\text{slide}} \ .$$

These three global velocity vectors are characterized by the local velocities they induce on each constraint: $\dot{\mathbf{q}}'_{\text{fix}}$ corresponds to zero relative velocity (normal and tangential) for all constraints; $\dot{\mathbf{q}}'_{\text{norm}}$ corresponds to purely normal velocity for all constraints; $\dot{\mathbf{q}}'_{\text{slide}}$ is the sliding velocity for which friction must be applied.

Following inelastic projection, $\dot{\mathbf{q}}'_{\text{norm}}$ is zero. With that, we have

$$\dot{\mathbf{q}}'_{\text{slide}} = \dot{\mathbf{q}}' - \dot{\mathbf{q}}'_{\text{fix}} \ . \tag{3.9}$$

To find $\dot{\mathbf{q}}'_{\text{fix}}$, we project out motion that induces a relative velocity for any constraint. For a single constraint, this relative velocity is

$$V_{\text{rel}} = \nabla h \dot{\mathbf{q}}' \ , \tag{3.10}$$

where $h(\mathbf{q}) = X_4 - (\alpha_1 X_1 + \alpha_2 X_2 + \alpha_3 X_3)$ for a vertex-triangle collision and $h(\mathbf{q}) = (\alpha_3 X_3 + \alpha_4 X_4) - (\alpha_1 X_1 + \alpha_2 X_2)$ for an edge-edge collision, using the same notation as in §3.3. For multiple constraints, we project out any velocity in the row space of $\nabla H = \left[\nabla h_1^T, ..., \nabla h_k^T\right]^T$, where ∇H is a $3k \times 3n$ matrix and k is the number of constraints in the impact zone, leaving us with $\dot{\mathbf{q}}'_{\text{fix}}$. To implement this projection, observe that $\dot{\mathbf{q}}'_{\text{fix}}$ is the minimizer of

$$\|\dot{\mathbf{q}}' - \dot{\mathbf{q}}'_{\text{fix}}\|_M^2 \ , \qquad \text{subject to} \quad \nabla H \dot{\mathbf{q}}'_{\text{fix}} = \mathbf{0} \ .$$

We solve the linear system $\nabla H \mathrm{M}^{-1} \nabla H^T \boldsymbol{\lambda}' = \nabla H \dot{\mathbf{q}}$ for $\boldsymbol{\lambda}' \in \mathbb{R}^{3n}$ to recover $\dot{\mathbf{q}}'_{\text{fix}} = \mathbf{q}' + \mathrm{M}^{-1} \nabla H^T \boldsymbol{\lambda}'$ and $\dot{\mathbf{q}}'_{\text{slide}}$ (see (3.9)).

From the sliding velocity $\dot{\mathbf{q}}'_{\text{slide}}$, we can apply an approximation of Coulomb friction. We would like the friction to be proportional to the normal force, which is simply the change in relative velocity, bounded by the pre-collision sliding velocity. Given a pre-projection velocity $\dot{\mathbf{q}}$, post-projection velocity $\dot{\mathbf{q}}'$, and friction coefficient μ, the change in velocity due to friction is

$$\Delta \dot{\mathbf{q}} = \min\left(\mu \|\dot{\mathbf{q}}' - \dot{\mathbf{q}}\|, \|\dot{\mathbf{q}}'_{\text{slide}}\|\right) \widehat{\dot{\mathbf{q}}'_{\text{slide}}} \ .$$

Note that the amount of friction applied is dictated by the magnitude of the normal force in configuration space. Thus a large normal force for one constraint may induce a larger friction impulse on other constraints. This operation is performed per impact zone, so in practice the colliding region is small and local enough that this approximation yields plausible results. A correct treatment could build on the work of Kaufman *et al.* [2005], who intersect with a limit curve for each constraint, preventing extraneous friction.

Figure 3.6: Shoving cloth through a narrow funnel yields free-flowing motion, despite the complex network of contact regions.

3.8 Results

We tested our method on several challenging scenarios that exercise two possible fail-safes—the original rigid impact zones and inelastic projection.

Figure 3.6 shows frames from our simulation of a square cloth being plunged into a small funnel by a scripted ball. This example highlights the importance of allowing free-flowing tangential motion, since otherwise the cloth rigidifies and halts inside the chute. Unlike the inelastic projection, the rigid impact zones fail-safe fails to give a plausible response. We also ran this example using two passes of collisions response, where the first pass treats only cloth-ball and cloth-funnel collisions, and the second pass solely responds to cloth-cloth collisions (where rigid impact zones behave as expected). This ensures that the simulation does not fail due to an invalid input into the rigid impact zone computation. Again rigid impact zones fail, this time by allowing the cloth to penetrate the funnel, even with an object thickness 100 times larger than the cloth thickness, and a timestep of 10^{-4} seconds.

Figure 3.7 shows a scenario where tangential sliding is key and rigid impact zones have a fundamental limitation. In this example, we drop a sequence of small (2-triangle) squares onto an incline. The projection method allows the squares to smoothly slide despite the long chain of simultaneous, interacting collisions. We repeat this example with friction, to reinforce that free-flowing tangential sliding still allows control of friction. As an example where rigid impact zones have an advantage over inelastic projection, we drop a large number of cloth squares in a stack on a flat plane; here the rigid response is physically correct, and linear projection is wasted work.

Figure 3.8 shows a thin ribbon falling through a trough under varying coefficients of

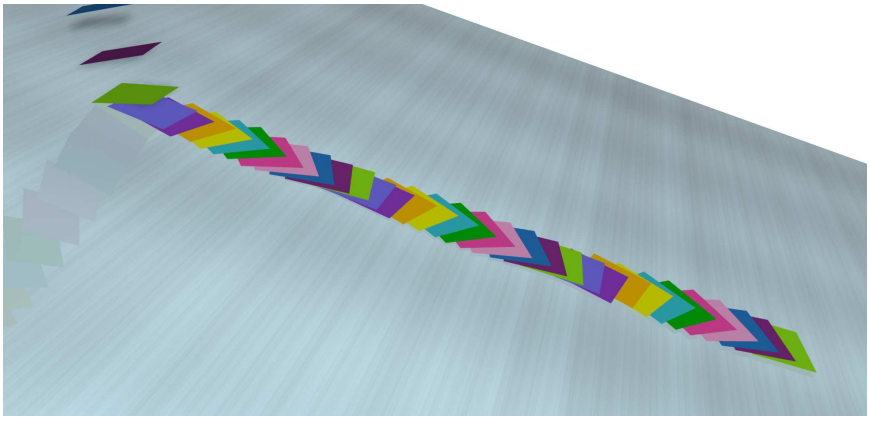


Figure 3.7: Our projection method easily handles a multitude of simultaneous collisions with overlapping stencils, allowing each piece of stacked fabric to freely flow down the inclined plane.

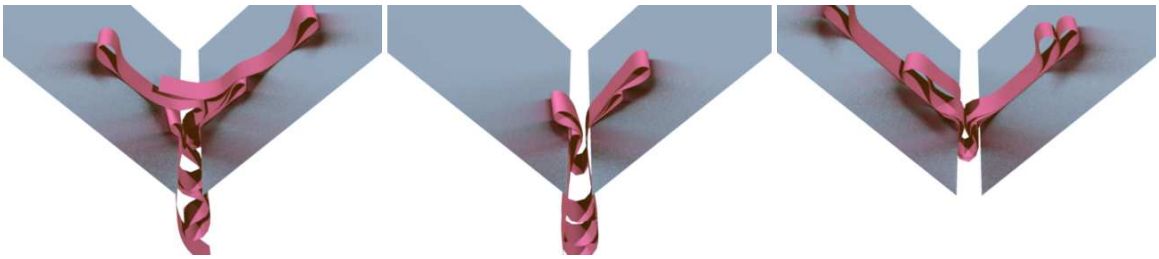


Figure 3.8: Simulated ribbons with varying coefficients of friction

Example	DOFs	Time Step	Frames	Total	Collision	Mean
Ribbon	7416	1/480	1000	7627.33	6647.97	6.65
				5029.29	4049.09	4.05
Funnel	16569	1/480	1000	23711.75	18814.60	18.81
				13124.61	9613.79	9.61
Funnel (low-res)	7695	1/480	1000	18763.28	15933.07	15.93
				13714.11	10899.00	10.90
Crushed Cylinder	3099	1/384	1000	2833.33	1911.35	1.91
				2846.09	1923.50	1.92
Squares on Incline	732	1/24	2500	1578.13	1540.98	0.62
				853.67	816.08	0.33
Squares on Plane	732	1/24	2500	1224.55	1178.21	0.47
				1388.75	1341.87	0.54

Table 3.1: Timing (in seconds) for examples executed on a single thread of a 2.66 Ghz Intel Core 2 Duo with 4GB RAM, comparing RIZ (shaded rows) and inelastic projection (white rows).

friction. Notice that the linear projection fail-safe gives markedly different behaviors as friction is increased, whereas with rigid impact zones even the frictionless ribbon sticks.

Timing Table 1 lists timings for many of the above examples. Notice that simulations run with the inelastic projection fail-safe have run times comparable to the rigid impact zones fail-safe, with linear projection at worst 15% slower than rigid impact zones. Indeed, since most simulation time is spent performing collision detection, and not collision response, the additional linear solve in the proposed fail-safe does not substantially decrease the speed of simulation. In fact, by allowing heavily colliding regions to smoothly slide past one another, contact clumps tend to dissipate quicker, reducing overall runtime.

3.9 Discussion

Limitations Our method suffers from all the usual limitations of impulse-based collision response. Large timestep sizes tend to be disagreeable, as internal forces are disrupted by the discontinuities introduced by velocity filters. Also, large timesteps render the repulsive penalty forces useless, and instead cause collision response to rely heavily on the geometric impulses, which do not always behave in a physically plausible manner within the velocity filter framework (see Figure 3.1).

Additionally, treating all collisions occurring over a timestep as simultaneous may in-

troduce artificial dissipation: if collisions were handled instead in causal order, responding to earlier collisions could prevent subsequent ones. This approach to collisions also violates conservation of energy and momentum. Thus, this work and all other that continues in its direction is clearly delineated by the fact that it is only useful in the context of purely graphical applications, *i.e.*, situations where only the visual plausibility of the material matters, and not true physical accuracy.

The linear system solved during inelastic projection can be ill-conditioned or singular; since it is not acceptable for the fail-safe itself to fail, a numerical method suited for ill-conditioned matrices (such as SVD or GMRES) is strongly recommended. We also tested QLP factorization [Stewart, 1999], but unfortunately it did not approximate the singular values well enough for our needs.

Our method has two friction-related limitations. The magnitude of friction is governed by the magnitude of the collision response impulse in configuration space. This magnitude dictates the amount of friction for an entire impact zone, potentially resulting in large collision impulses increasing the applied friction for another part of the impact zone. Our method also supports only one coefficient of friction per impact zone.

Conclusion Since inelastic projection requires a straightforward linear solve, we hope that it can be quickly incorporated into existing frameworks based on the velocity filter of Bridson *et al.* [2002]. Our method is compatible with an approximation of Coulomb friction that does not create additional collisions; this allows for the efficient simulation of a wide range of materials, including cloth and thin shells.

The possibility of incorporating LCP as a fail-safe is intriguing. However, preliminary tests have not been promising. Nonetheless, the ability to use such a method to resolve collisions would eliminate the need for the second pass of iterative impulses. We believe that while the penalty forces would not technically be necessary, they would alleviate much of the burden from the LCP solver, both by removing the "easy" collisions from consideration, and in aiding the numerics by keeping the geometry well-separated.

The shortcomings that this method resolves leaves a new set of research opportunities. In particular, current deformable simulations ignore the causality of collisions, treating

any that occur within a timestep as simultaneous. While rigid body simulations can treat collisions in order by stepping to the collision time, resolving it, and starting with new initial conditions [Witkin and Baraff, 2001], the large number of degrees of freedom in cloth simulation have thus far prohibited such methods. This motivates our ongoing investigation of asynchronous methods for resolving collisions in causal order.

Chapter 4

Discrete penalty layers

Chapter 3 reviewed and presented the state-of-the-art in collision response for deformable materials; the work of Bridson *et al.* [2002] with the novel extension of inelastic projection. The state-of-the-art can guarantee absence of penetrations, up to some pre-determined epsilon amount. The difficulty is in determining this epsilon, which can vary between simulations based on scale, system stiffnesses, and other factors. Once found, the user enjoys robust simulations for that particular setup. However, any small, unpredictable change could disturb the delicate balance and require a re-tweaking of these parameters.

Using velocity filters requires a tradeoff: exchanging physical correctness for speed. By treating all collisions over a timestep as simultaneous, simulation times are substantially reduced, but any hope of maintaining conservative properties is lost. One consequence of this is that any advancements made in this area are useful only to those purely interested in the visuals of a simulation, ignoring any application to engineers. Furthermore, even in graphics applications the objective is blurred, as users attempt to replicate physical behavior with very non-physical parameters. With no reliable intuition, the meaning of parameters changes with any modification to the underlying method, confusing and frustrating the users.

We wish to retain all these properties: robustness, efficiency, and physical accuracy. In Chapter 2, we covered the broad classes of methods used for simulation. The velocity filter approach is a type of impulse, since it modifies velocities. For it to regain some semblance of the physical accuracy we desire it must treat collisions in order, integrating up to collision

times, resolving offending velocities, then resuming the simulation. Unfortunately, this counteracts the advantages that make velocity filters desirable, and while straightforward to implement, the enormous number of possible impact events during a timestep make this approach unfeasible [Cirak and West, 2005].

Constraint-based formulations seem promising, but would require continued work on simulating fully deformable materials. Some work has been conducted in applying this class of methods for more robust simulations, but has mostly been restricted to rigid body simulations, with some work addressing deformable materials, but only in the quasi-rigid or quasi-deformable regime [Pauly *et al.*, 2004; Kaufman *et al.*, 2008]. However, these formulations, and their respective solvers, have not been demonstrated to have the necessary numerical robustness for the simulation of cloth and thin shells.

This leaves penalty-based methods. Penalty methods are the only class that has no problems with correctness or efficiency in the simulation of deformable materials, while much has been attempted to regain these lost characteristics for the other classes. We revisit penalty-based schemes in an attempt to make them robust against interpenetrations.

4.1 Previous work

General penalty methods were discussed in §2.1. They are conservative forces, derived from an energy potential. This quality makes them very popular in mechanical engineering applications, where they are referred to as constitutive equations [Wriggers and Laursen, 2007]. In this domain, physical correctness is so important users are willing to tolerate the robustness deficiencies of the penalty method, namely that for any chosen stiffness there exists a relative momentum large enough to cause interpenetration. This mindset of placing physical correctness on equal footing with robustness motivates our work in this area.

Sections 2.1 and 3.1 covered previous work in the graphics literature. Here we briefly revisit the contact mechanics literature.

Most formulations express a contact in terms of a gap function [Stein and Wriggers, 1982; Pires and Oden, 1983; Endo *et al.*, 1984; Bathe and Chaudhary, 1985; Chaudhary and Bathe, 1986; Rabier *et al.*, 1986; Wriggers and Van, 1990; Peric and Owen, 1992;

Papadopoulos and Taylor, 1993; Taylor and Papadopoulos, 1993; Wriggers and Imhof, 1993; Wriggers and Scherf, 1995; Wriggers and Zavarise, 1997],

$$g = X_a - X_b,$$

where X_a and X_b are the projections of the closest points between two bodies a and b onto the normal direction.

This gap functions has the property that it is positive when the contact surfaces are separated, zero when touching, and negative when penetrating. The Hertz-Signorini-Moreau law of unilateral contact can then be expressed in terms of the gap g and some contact force f, determined by the response method [Wriggers and Panagiotopoulos, 1999]:

$$g > 0$$

$$f > 0$$

$$gf = 0.$$

The preceding conditions state that either the separation distance between two contact points is zero or the force is zero, yet only one is true. If two contact points are not touching, then $g > 0$ and f must be zero for the third condition to hold. Likewise, if some force is active, $f > 0$, then impenetrability is being enforced, and the contact gap must therefore be zero. Requiring that the force be nonnegative ensures the absence of *adhesion*, or any force that pulls separating bodies back together. The inelastic projection of Chapter 3 relaxed this requirement for computational speed. The third condition is called a complementarity condition.

As discussed in §2.1, one method to satisfy these conditions is the penalty method, and we gave a few choices of contact energies, written in terms of the contact gap g (Figure 2.2).

The commonly seen collision response penalty forces in graphics can be thought of as a single iteration of penalty methods in optimization, also known as *barrier methods* [Auslender, 1999]. In simulation, the functions being optimized are the equations of motion, which are, naturally, dynamic in nature. Barrier methods could be thought of as a quasi-static solution, treating a configuration as a stationary point until a penetration-free solution is found. This solution is found iteratively, with subsequently stiffer penalties applied until convergence (see Algorithm 4 for one barrier method). However, such a function must

also have unbounded second derivative, ruling out stable fixed-step time integration for *any* choice of step size [Hairer *et al.*, 2002]. Additionally, as with the penalty method, ill-conditioning can occur. Most solutions also require the algorithm to find a feasible starting point which satisfies all constraints (Algorithm 4). Due to these drawbacks, barrier methods are not often used in computational contact mechanics [Wriggers and Nettingsmeier, 2008].

Algorithm 4 Barrier method for function f augmented with penalty forces

Require: $\epsilon > 0, \eta > 1$, and initial penalty stiffness c_0

1: Choose x^0 that violates at least one constraint
2: $k = 1$
3: **repeat**
4: Minimize $x^k = f(c_{k-1}, x^{k-1})$
5: $c^k = \eta c^{k-1}$
6: $k = k + 1$
7: **until** $\|x^{k-1} - x^k\| < \epsilon$

The deficiencies of both penalty methods and Lagrangian methods have led to the development of *Augmented Lagrangian* methods, which combine the strengths of traditional penalty and Lagrangian methods [Wriggers, 1995]. As the penalty method becomes ill-conditioned, its contribution is then used to update a Lagrange multiplier to maintain the contact. This method has proved popular and successful for many applications. We, however, are drawn to the simplicity of penalty methods and focus our efforts on making them provably robust against contact violations.

4.2 Penalty layers

Consider a simple penalty method that penalizes proximity between bodies. To begin, we modify the gap function to take into account a given surface thickness η

$$g_\eta(\mathbf{q}) = X_b - X_a - \eta,$$

which tracks signed proximity between moving points X_a and X_b, rather than distance. When $g < 0$, the points are said to be *proximate*. In engineering literature, this proximity distance η is often called "bond", "film", or the "interface." We can express a half-quadratic contact potential and force in terms of g

$$V_\eta^r(g(\mathbf{q})) = \begin{cases} \frac{1}{2} r g^2 & \text{if } g \le 0 \\ 0 & \text{if } g > 0 \ , \end{cases} \qquad \mathbf{F} = \begin{cases} -r g \nabla g & \text{if } g \le 0 \\ 0 & \text{if } g > 0 \ , \end{cases}$$

where r is the contact *stiffness*. Choosing a penalty stiffness is the most criticized problem of the penalty method [Baraff, 1989]. For any fixed stiffness r, there exists a sufficiently large approach velocity such that the contact potential will be overcome by the momentum, allowing the configuration to tunnel illegally through an inadmissible region (§2.1).

We propose a construction consisting of an infinite family of *nested potentials*

$$V_{\eta(l)}^{r(l)}, \qquad l = 1, 2, \dots \ , \tag{4.1}$$

where $\eta(l)$ is a monotonically decreasing proximity (or "thickness") for the l-th potential, and $r(l)$ is a monotonically increasing penalty stiffness. These functions have the property that as $l \to \infty$, $\eta(l) \to 0$ and $r(l) \to \infty$. We call the region $\eta(n+1) \le g(\mathbf{q}) \le \eta(n)$, where exactly n of the potentials are nonzero, the n-th *discrete penalty layer* (see Figure 4.2). Given a distance d between X_a and X_b, we can compute the active penalty layer l as

$$l = \eta^{-1}(d).$$

The potential energy of the contact is then the sum of all active penalty potentials

$$V = \sum_{i=1}^{l} V_{\eta(i)}^{r(i)}.$$

Layer strength and distribution The strength of successive penalty layers is determined by the functions $r(l)$, while the distribution is given by $\eta(l)$. The choice of functions has practical considerations, covered in §6.1. We use $r(l) = r(1)l^3$ and $\eta(l) = \eta(1)l^{-1/4}$, where $r(1)$ and $\eta(1)$ are a simulation-dependent base stiffness and proximity thickness for

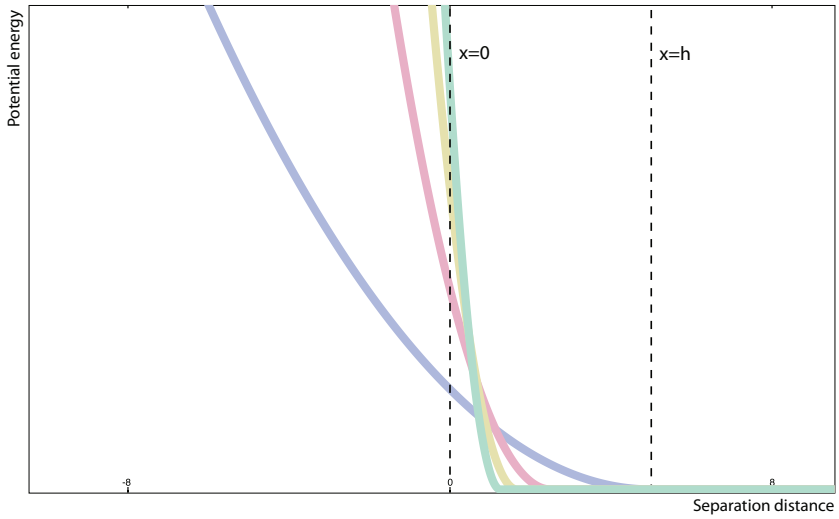
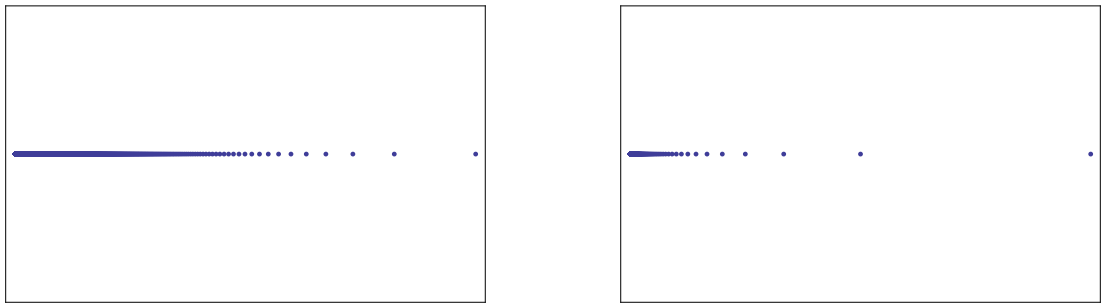


Figure 4.1: As separation distance decreases, increasingly stronger penalty layers are activated.



(a) $\eta(l) = l^{-\frac{1}{4}}$ (b) $\eta(l) = l^{-1}$

Figure 4.2: Plots showing the distribution of layers for two functions.

the outermost layer. These choices provide a sufficiently "front-heavy" distribution of layers and growth of layer stiffness. Other choices, such as $\eta(l) = \eta(1)l^{-1}$, gives a sparse distribution, with layers densely packed near a separation distance of zero, where numerical precision rapidly decreases (see Figure 4.2).

4.3 Dissipation

Energy near-conservation is imperative for the reliability of the simulator. Controlled dissipation, however, in the form of friction and coefficients of restitution, is often desired in practical simulations. Thus we extend the discrete penalty layers model to incorporate these effects.

Coefficient of restitution The coefficient of restitution e_{COR} is a "melting pot" approximation, accounting for various unresolved micro-level phenomena [Brilliantov and Pöschel, 2004; Schwager and Pöschel, 2007] including viscosity and plasticity. To model plastic work, we replace the nested penalty potentials with biphasic potentials [Choi and Ko, 2005]

$$V_\eta^r(g(\mathbf{q})) = \begin{cases} \frac{1}{2} rcg(\mathbf{q})^2 & g \leq 0 \\ 0 & g \geq 0, \end{cases}$$

where c is e_{COR} if the primitives are separating, 1 otherwise. The penalty layers exert their full force during compression, then weaken according to the coefficient of restitution during decompression. We could (but did not) further extend this model to account for viscous damping during impact, measuring strain rate by (some monotonic function of) the change in the gap function $g(\mathbf{q})$.

Friction The Coulomb friction model serves as a simple approximation of an extremely complicated physical interaction. Consider the Coulomb friction force $F_f = \mu F_n$, where μ is the coefficient of friction and F_n is the normal force. The force opposes relative tangential motion between points in contact (§3.7).

We apply friction along with each penalty force, separately for each penalty layer. Just as increasingly stiff penalty forces are applied for contact forces, friction forces are increasingly applied (bounded by each F_n) to correctly impede high-speed tangential motion.

4.4 Discussion

The nested penalty layers form a barrier potential, by construction. For every possible impact there is a high enough layer such that the cumulative potential is strong enough to counteract the collision, *before* penetration occurs, making them robust against interpenetrations. Chapter 6 formalizes this argument.

However, this is a continuous construction, and we encounter difficulties when implementing in a discrete setting. Forces of increasing stiffness cannot always be integrated stably at the same timestep. The smallest stable timestep is required, but in the context of penalty layers, this may not always be known. The next chapter addresses these difficulties.

Chapter 5

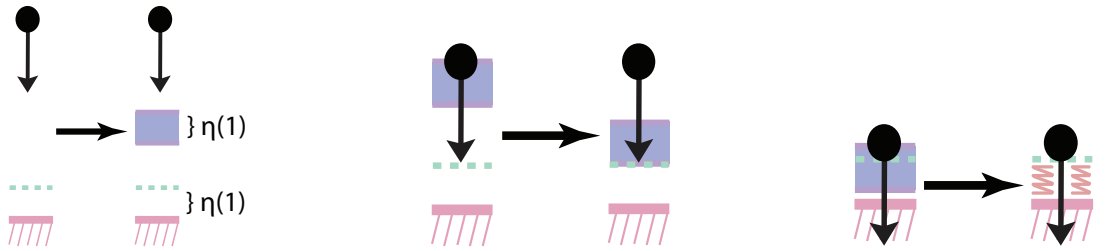
Asynchronous contact mechanics

The Discrete Penalty Layers (DPL) presented in Chapter 4 guarantee, by construction, the absence of penetration in a simulation. In doing so, however, we introduce two additional computational difficulties in the simulation:

1. Each successive penalty layer has an increasing force stiffness, and thus, a correspondingly decreasing stable timestep. Timestepping the entire simulation at the smallest timestep is wasteful, especially when considering the localized nature of heavy contact regions (high penalty layers). Thus, penalty layers necessitate the use of an adaptive or multi-rate integrator that can integrate forces with different stability requirements [Neal and Belytschko, 1989].

2. It cannot be known, *a priori*, the deepest penalty layer that will be activated (have a non-zero energy) in a simulation. Hence, even if one desired to simulate the entire mesh with the smallest timestep, it cannot be known at startup time. Deciding on a maximum penalty layer beforehand eliminates any robustness guarantees.

   However, recall the half-quadratic design of the penalty potentials. When separation distance is greater than $\eta(l)$ for a given layer l, that force evaluates to exactly 0, and thus may be safely ignored from the force computation. Furthermore, any layers below it can also be ignored. Based on this observation we introduce a method of continuously culling these inactive forces, and dynamically adding successive penalty layers as they are needed.



(a) A separating slab KDS is created as proof of no contact.

(b) A new separating slab is scheduled.

(c) No efficient slab exists, so a penalty force is activated.

Figure 5.1: A few snapshots of a simple simulation.

5.1 Walkthrough

To introduce the method of asynchronous contact mechanics, we walk through a simplified example of the entire algorithm, which operates around the notion of distinct *events*, sorted in time order on a priority queue. We encourage the reader to refer back to this section after each concept is discussed.

Consider a single particle free-falling towards a fixed floor in two dimensions. At some time t the particle will reach a height of $\eta(1)$ above the floor, and enter the first penalty layer. The algorithm must activate the penalty force for this layer no later than at time t—activating the force earlier is conservative. The queue is initialized with two events: a gravity force event and a separation slab event (§5.3) between the particle and the floor that represents the moment they will be in proximity.

We assume there exists an $\eta(1)$-slab–a line extruded to thickness $\eta(1)$–that separates the particle from the floor. The existence of such a slab provides a provable guarantee, or *certificate*, that the distance between the particle and the floor is at least $\eta(1)$. This guarantee remains valid until either the floor or the particle hits the slab, at which point the certificate *fails*, and the algorithm can either try to find a new separating slab, or if doing so is too costly, activate the penalty force.

More concretely, begin by confirming the existence of such a slab and place it between the particle and the floor (Figure 5.1(a)). To schedule this as an event on the queue, a certificate failure time must be computed. Compute the *failure time* as the time at which

either primitive (the particle or the floor) comes into contact with the slab. This time, t_{cf}^1, is a conservative time for when the particle will enter the first penalty layer.

During simulation, at time t_{cf}^1, the event representing the certificate failure is popped off the queue. We check the separation distance and see that it is greater than $\eta(1)$. A new slab is found, and the event is rescheduled using the failure time of the new slab, t_{cf}^2 (Figure 5.1(b)).

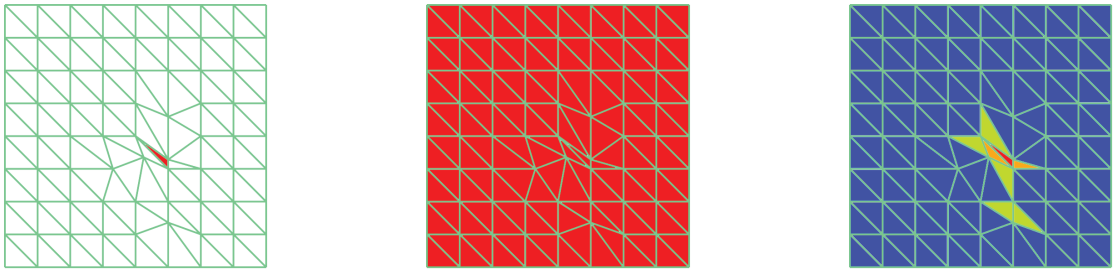
As the simulation proceeds, an event at time $t_g < t_{cf}^2$, representing a gravity force event, is popped off the queue. This event modifies the velocity of the particle. However, the separating slab event time was computed using this particle's original velocity. The computed failure time t_{cf}^2 for the slab may no longer be conservative, so it must be rescheduled to guarantee no penetrations occur.

Simulation continues in this manner, until the frequency of separation slab creation / destruction is high enough that it becomes computationally cheaper to activate the outermost layer's penalty force and stop creating slabs (Figure 5.1(c)). The choice for what is considered "high enough" is a simulation parameter that affects performance, but not correctness.

When the outermost penalty force is activated, the $\eta(1)$-slab is no longer required, and instead becomes an $\eta(2)$-slab, providing a proof for the culling of the layer-2 penalty force. If the layer-1 force ever evaluates to 0 (separation distance is greater than $\eta(1)$) and the particle is separating from the floor, the force is de-activated and the slab is elevated back to a thickness of $\eta(1)$. This "lazy" approach to force de-activation is safe.

5.2 Asynchronous variational integrators

Asynchronous variational integrators (AVIs) [Lew *et al.*, 2003], were developed in mechanics for the integration of non-uniform meshes. Take, for instance, the mesh in Figure 5.2; element sizes across the mesh vary greatly. In a traditional simulation setting, the smallest element would determine a timestep for the entire simulation to guarantee stability (Figure 5.2(a)). This greatly affects the speed of simulations, as the large elements, along with their larger stability requirement, are forced to be timestepped at a much smaller step than



(a) Near degenerate geometry requires small timesteps

(b) In synchronous simulation the timestep is limited

(c) Asynchronous integration timesteps independently

Figure 5.2: In synchronous simulations, near-degenerate geometry determines the stable timestep for the entire simulation. Asynchronous simulation allows integration of each element with its own stability requirement.

required (Figure 5.2(b)).

AVIs alleviate this difficulty. They allow the smooth integration of a non-uniform mesh with each element running at its own stable timestep (Figure 5.2(c)). AVIs are a subclass of a larger class of integrators called *variational integrators*. To derive these integrators, and understand their importance, we revisit ideas from continuous as well as discrete Lagrangian dynamics. For a detailed treatment, see Marsden and West [2001] or Kharevych *et al.* [2006].

Continuous dynamics Given a configuration $\mathbf{q} \in \mathbb{Q}$ and corresponding configurational velocity $\dot{\mathbf{q}}$, we can write the Lagrangian of the system as

$$L(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T M \dot{\mathbf{q}} - V(\mathbf{q}),$$

where M is a symmetric mass matrix and V is the potential energy of the system.

In Lagrangian mechanics, the action functional is written by integrating the Lagrangian along a curve $\mathbf{q}(t)$:

$$S(\mathbf{q}) = \int_0^T L(\mathbf{q}(t), \dot{\mathbf{q}}(t))dt.$$

We then take a first-order variation of this action and apply integration by parts:

$$\begin{aligned}\delta S(\mathbf{q}) &= \delta \int_0^T L(\mathbf{q}(t), \dot{\mathbf{q}}(t))dt \\ &= \int_0^T \left[\frac{\partial L}{\partial \mathbf{q}} \cdot \delta\mathbf{q} + \frac{\partial L}{\partial \dot{\mathbf{q}}} \cdot \delta\dot{\mathbf{q}}\right] dt \\ &= \int_0^T \left[\frac{\partial L}{\partial \mathbf{q}} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\mathbf{q}}}\right)\right] \cdot \delta\mathbf{q}dt + \left[\frac{\partial L}{\partial \dot{\mathbf{q}}} \cdot \delta\mathbf{q}\right]_0^T.\end{aligned}$$

We assume the variation at the endpoints is zero, so $\delta\mathbf{q}(T) = \delta\mathbf{q}(0) = 0$, eliminating the last term. Applying Hamilton's principle, which states that the variation of the action must be zero at all times [Goldstein *et al.*, 2002], results in the *Euler-Lagrange equations*

$$\frac{\partial L}{\partial \mathbf{q}}(\mathbf{q}, \dot{\mathbf{q}}) - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\mathbf{q}}}(\mathbf{q}, \dot{\mathbf{q}})\right) = 0.$$

This is simply a form of Newton's second law of motion.

Discrete dynamics We will now perform an analogous derivation for a discrete trajectory. Let $\mathbf{q}_0$ and $\mathbf{q}_1$ be two points on the parameterized discrete trajectory, separated by a time step $h \in \mathbb{R}$.

Following the above, we begin by writing a discrete approximate to the Lagrangian L_d, which takes the time step as an additional parameter,

$$L_d(\mathbf{q}_0, \mathbf{q}_1, h) = h\left[\frac{1}{2}\left(\frac{\mathbf{q}_1 - \mathbf{q}_0}{h}\right)^T M\left(\frac{\mathbf{q}_1 - \mathbf{q}_0}{h}\right) - V(\mathbf{q}_0)\right].$$

This choice of Lagrangian uses a simple first-order approximation to velocity, but other choices are available. We shall see that different approximations for the discrete Lagrangian lead to different timestep rules for the integrator.

The action integral becomes a sum in the discrete setting, on which we can compute

variations. The action is a function of the discrete trajectory of the system, the set $\{\mathbf{q}_k\}_{k=0}^N$.

$$\begin{aligned}\delta S_d(\mathbf{q}_k) &= \delta \sum_{k=0}^{N-1} L_d(\mathbf{q}_k, \mathbf{q}_{k+1}, h) \\ &= \sum_{k=0}^{N-1} \left[D_1 L_d(\mathbf{q}_k, \mathbf{q}_{k+1}, h) \cdot \delta\mathbf{q}_k + D_2 L_d(\mathbf{q}_k, \mathbf{q}_{k+1}, h) \cdot \delta\mathbf{q}_{k+1}\right] \\ &= \sum_{k=0}^{N-1} \left[D_2 L_d(\mathbf{q}_{k-1}, \mathbf{q}_k, h) + D_1 L_d(\mathbf{q}_k, \mathbf{q}_{k+1}, h)\right] \cdot \delta\mathbf{q}_k + \\ &\qquad D_1 L_d(\mathbf{q}_0, \mathbf{q}_1, h) \cdot \delta\mathbf{q}_0 + D_2 L_d(\mathbf{q}_{N-1}, \mathbf{q}_N, h) \cdot \delta\mathbf{q}_N.\end{aligned}$$

Here D_i denotes the derivative with respect to the i-th argument. Recall again, as in the continuous case, that we require the action to be zero at the end points $\delta\mathbf{q}_0 = \delta\mathbf{q}_N = 0$. This gives the *discrete Euler-Lagrange equations*

$$D_2 L_d(\mathbf{q}_{k-1}, \mathbf{q}_k, h) + D_1 L_d(\mathbf{q}_k, \mathbf{q}_{k+1}, h) = 0.$$

Our choice of L_d now takes effect, where we compute

$$\begin{aligned}D_2 L_d(\mathbf{q}_{k-1}, \mathbf{q}_k, h) &= M\left(\frac{q_k - q_{k-1}}{h}\right) \\ D_1 L_d(\mathbf{q}_k, \mathbf{q}_{k+1}, h) &= -\left[M\left(\frac{q_k - q_{k-1}}{h}\right) + h\nabla V(\mathbf{q}_k)\right].\end{aligned}$$

We can then concretely rewrite the discrete Euler-Lagrange equations as

$$M\left(\frac{\mathbf{q}_{k+1} - 2\mathbf{q}_k + \mathbf{q}_{k-1}}{h^2}\right) = -\nabla V(\mathbf{q}_k),$$

a discretization of Newton's equations which uses finite differences to approximate the derivative.

Specifying initial configurations $\mathbf{q}_0$ and $\mathbf{q}_1$ allows us to give a recursive definition for each successive $\mathbf{q}_i$ for $i > 1$, or, in other words, a timestepping scheme. In this case, we arrive at

$$\begin{aligned}\mathbf{q}_{k+1} &= \mathbf{q}_k + h\dot{\mathbf{q}}_{k+1} \\ \dot{\mathbf{q}}_{k+1} &= \dot{\mathbf{q}}_k - hM^{-1}\nabla V,\end{aligned}$$

or the *symplectic Euler* timestepping rules.

Variational integrators An integrator which is the discrete Euler-Lagrange equation for some discrete Lagrangian is called a *variational integrator*. Many common integrators, such as symplectic Euler and Newmark, are variational integrators [Suris, 1990; Kane *et al.*, 2000]. The preceding was a brief treatment, but this class of integrators is well-studied throughout the literature [Kane *et al.*, 2000; Marsden *et al.*, 2001; Marsden and West, 2001; West, 2003].

Variational integrators satisfy a discrete form of Noether's Theorem, which leads to momentum conservation laws [West, 2003]. Furthermore, variational integrators are automatically *symplectic*, guaranteeing that the energy of the system does not drift over exponentially long runtimes, which can be verified both computationally [Wendlandt and Marsden, 1997] and mathematically [Marsden *et al.*, 1998]. In our context, we wish to obey known conservation laws, including energy and momentum. Thus, we are drawn to variational integrators for their provably correct manner of preserving these quantities [Hairer *et al.*, 2002]. However, the discrete Lagrangian used here takes a single, constant time step parameter h. We wish to allow variable timestep sizes based on varying stiffnesses in the system.

Asynchronous dynamics To allow for asynchronous time integration, we follow Lew *et al.* [2003] and split up the energy potential into discrete elements,

$$L(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T M \dot{\mathbf{q}} - \sum_{i \in T} V_i(\mathbf{q}),$$

where V_i is the potential for the i-th mesh element. In the simple case of strain energy on a triangle mesh, V_i is a function of the degrees of freedom in $\mathbf{q}$ that comprise the i-th triangle.

For discretization, instead of choosing a single timestep h we choose a timestep h^i per energy potential, derived from stability requirements. The i-th element is then updated at times $t_i^k = kh^i$, for $k \in \mathbb{N}$. Note that the timesteps h^i and h^j for two elements i and j are not necessarily integer multiples.

The discrete action sum that results from an asynchronous Lagrangian is

$$S_d = \sum_a \sum_{i=1}^{N_a - 1} \frac{1}{2} m_a (t_a^{i+1} - t_i^i) \| \frac{x_a^{i+1} - x_x^i}{t_a^{i+1} - t_a^i} \|^2 - \sum_k \sum_{j=0}^{N_k - 1} (t_k^{j+1} - t_k^j) V_k(x_k^{j+1}),$$

where x_k^j is the position in $\mathbb{R}^3$ of the k-th node during its j-th timestep. N_a is the number of nodal positions for node a, m_a is the lumped mass at node a, and N_k is the number of evaluations of the k-th energy potential.

Setting this sum to zero and taking the variations gives the discrete Euler-Lagrange equation

$$p_a^{i+\frac{1}{2}} - p_a^{i-\frac{1}{2}} = I_a^i,$$

where

$$p_a^{i-\frac{1}{2}} \equiv m_a \frac{x_a^i - x_a^{i-1}}{t_a^i - t_a^{i-1}} \equiv m_a v_a^{i-\frac{1}{2}}$$

and

$$I_k^j \equiv -(t_k^j - t_k^{j-1}) \frac{\partial}{\partial x_k^j} V_k(x_k^j).$$

For the full derivation, see Lew *et al.* [2003].

Lew *et al.* [2003] applies AVIs to problems in elasticity, where elements consist of triangles or tetrahedra undergoing strain. We are still interested in integrating the internal dynamics of a system, so our simulations include these elements, but we extend AVIs to include elements representing contact between regions of a mesh. West [2003] and Lew *et al.* [2003] both allude to this possibility, but never fully explore it.

Asynchronous timestepping The above formulation gives rise to an explicit asynchronous variational integrator. Algorithm 5 reproduces the resulting asynchronous timestepping algorithm.

This algorithm is simple to implement, with every force computation represented by an event on a time-sorted priority queue. While a force computation repeats infinitely through time, it is sufficient to only have it represented by a single event at any instant. When that event is processed and trajectories updated, it is pushed back onto the queue at time $t + h^i$, where h^i is that force element's timestep.

Penalty layers We apply the asynchronous algorithm to accurately and efficiently integrate the discrete penalty layers of Chapter 4. In fact, it is a perfect match; just as different size mesh elements have different stability requirements, so do the penalty layers. Events

Algorithm 5 AVI timestepping algorithm

{Let Q be a priority queue of events, sorted by event times $E.t.$}
$q \leftarrow q_0$; $\dot{q} \leftarrow \dot{q}_0$ {Set up initial conditions}
for all V_i **do**
  Compute h^i {Stable timestep for the i-th potential}
  $E_i \leftarrow (V_i, h^i)$ {Add all potentials to the queue as events}
  Q.push(E_i)
end for
loop
  $(E, t) \leftarrow Q$.**pop**
  for $i \in E$ **do**
    $X_i \leftarrow X_i + (t - t_i)\dot{X}_i$ {advance vertex to current time}
    $\dot{\mathbf{q}}_\xi \leftarrow \dot{\mathbf{q}}_\xi - hM_\xi^{-1}\partial V/\partial \mathbf{q}_\xi$ {local impulses, local mass}
    Q.**push**$(E, V, t + h^i)$ {Return the event to the queue, with new time}
    $t_i \leftarrow t$ {update vertex's clock}
  end for
end loop

on the priority queue are extended to include penalty force events for each layer, in addition to the events representing elastic forces.

The problem of which penalty layers to integrate remains. A "brute-force" solution, putting every possible force event on the queue, is not tractable for a single penalty layer (there are theoretically a quadratic number of potentials, since every primitive could possible collide with every other) and impossible for the theoretically infinite number of penalty layers. We must be able to dynamically add penalty force events to the queue.

5.3 Kinetic data structures

One last difficulty remains: at what layer must the simulation be integrated? The answer is simply, the deepest layer that the simulation needs. The difficulty here arises from the fact that the deepest layer needed is not known at the beginning, or at any other point in the simulation. If it were, simple penalty forces with an appropriate stiffness would suffice. Rather, we only know the deepest layer required *right now*, with a given configuration.

Recall the potential of Chapter §4:

$$V_\eta^r(g(\mathbf{q})) = \begin{cases} \frac{1}{2}rg^2 & \text{if } g \leq 0 \\ 0 & \text{if } g > 0 \ , \end{cases} \qquad \mathbf{F} = \begin{cases} -rg\nabla g & \text{if } g \leq 0 \\ 0 & \text{if } g > 0. \end{cases}$$

When separation distance exceeds a threshold $\eta(l)$, the potential energy (and hence force) for that layer evaluates to zero. This holds for the vast majority of the layers (for a contact in layer i, the set of potentials for layers $[i+1, \infty)$ are all zero). Thus, they may be safely pruned from integration, even though their clock with timestep h^i is still ticking. It is only a tick where the potential is non-zero that cannot, indeed, *must* not, be missed. For this, we must continuously monitor proximities to safely detect when separation distance passes a threshold, and activate the appropriate layer.

This is the problem statement for kinetic data structures. A *kinetic data structure* (KDS) is a data structure maintained under continuous motion of the underlying data. The data structure is usually a combinatorial structure of mesh geometry. For instance, a kinetic data structure could maintain an axis-aligned bounding box for a set of points $\{x_i\}$ with velocities $\{v_i\}$ (Figure 5.3). See Guibas [2004a] for a recent survey.

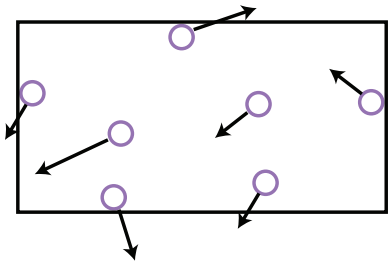


Figure 5.3: An axis-aligned bounding box around seven points.

Certificates KDSs maintain the data structure through a set of certificates. A *certificate* is an assertion that some property of the data structure currently holds. Together, these individual certificates guarantee that the data structure is valid. However, the geometry is undergoing motion, and thus the data structure may change over time.

For a bounding box, a certificate asserts that a particular point forms the boundary of the box. Therefore, for a 2-dimensional axis-aligned bounding box, four certificates are created.

Additional, "helper" certificates may be created that do not necessarily define the data structure, but assist in maintaining the combinatorial structure. These certificates are called *internal*, because if the data structure were a black box, their presence would be unknown. Similarly, certificates that maintain visible properties of the data structure are called *external*.

Events Using the known trajectories of the underlying data, the times at which these certificates are no longer valid are computed. These certificate failure times are called *events*. The certificates must be repaired either by revalidating the existing certificate or updating the data structure and creating new certificates. In this way, the data structure is valid at all instances in time.

In the case of a bounding box, an event occurs when a point on the boundary of the box is overtaken by another point within (Figure 5.4). In repairing this certificate, a new certificate must be created, establishing the new vertex as the extent of the boundary.

With the introduction of events, we begin to see the asynchronous nature of KDSs. Certificate failures can happen at any moment in time, not necessarily synchronously with

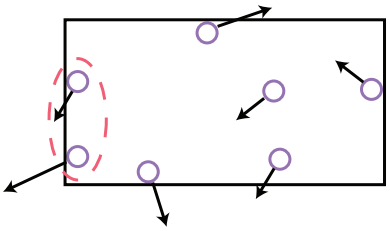


Figure 5.4: When the faster point overtakes the boundary point, a new certificate is created asserting that it now forms the left boundary.

motion of the data. When certificate failure times are computed, they are inserted into a time-sorted priority queue. Events are then processed in order through time. If the trajectories are changed at any time, all certificates which depend on the altered trajectories must have their failure time recomputed.

Evaluation Four measures have been proposed to evaluate the quality of a particular KDS [Guibas, 1998; Basch *et al.*, 1999; Guibas, 2004b]. They are:

1. **Responsiveness:** A KDS is *responsive* if the cost of repairing the set of certificates and updating attributes is small. "Small" is defined as polylogarithmic in the problem size.

2. **Efficiency:** A KDS is *efficient* if the total number of certificate failures that must be processed is comparable to the number of combinatorial attribute changes. In other words, the ratio must be small.

3. **Compactness:** A KDS is *compact* if the number of certificates required is close to linear to the number of degrees of freedom in the dynamic system.

4. **Locality:** A KDS is *local* if no single object participates in too many certificates.

The existence of these properties can be a theoretical question for many classes of problems [Guibas, 2004a].

Kinetic separation slabs For narrow-phase (primitive level) collision detection, we use a new construct we call *kinetic separation slabs*. Separation slabs certify the separation of

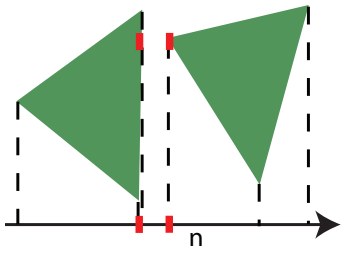


Figure 5.5: The projection of the closest points onto the normal vector.

geometric primitives by a given proximity thickness. They are inspired by the rigid motion separation planes of Guibas *et al.* [2001].

A separation slab certificate certifies that the geometry is separated by at least $\eta(l)$. A valid certificate for layer l guarantees the existence of a valid certificate for layers $l+1, l+2, ...$, so their explicit construction is not necessary.

When a valid separation slab can no longer be constructed, we activate the penalty force for layer l, and then create a separation slab certificate for layer $l+1$. This guarantees that the activation of any penalty force is never missed, and enforces an ordering on penalty layer activation.

To compute the failure time of the separation slab, we calculate the closest points between the geometry. This vector defines an axis of separation. We then project all vertices of the geometry onto this axis, and take the maximum / minimum velocities to conservatively compute the time when they will be proximate (Figure 5.5).

Given the closest points X_a and X_b, with respective velocities V_a and V_b, it is then straight-forward to compute the certificate failure time as

$$t = \frac{X_a - X_b}{V_b - V_a}.$$

Because the velocities used may not belong to the closest points, this time may not actually represent the exact time to activate a penalty force. When an event fires we check the distance, and if it is less than some arbitrary threshold (we use $\frac{11}{10}\eta(l)$), we go ahead and activate the penalty force. If it is not, we recompute the failure time and reschedule the certificate. This approach is conservative.

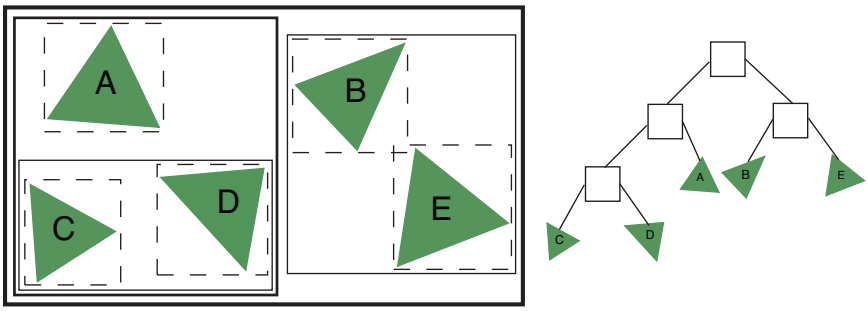


Figure 5.6: A sample scene (left) with its corresponding BVH (right).

Bounding volume hierarchies A *bounding volume* (BV) of a set of objects is a closed volume that completely contains the union of the objects in the set. A *bounding volume hierarchy* (BVH) is a tree of BVs such that a parent BV completely contains all the objects contained by its children BVs.

This structure permits logarithmic time collision queries by recursively testing for BV overlap against a BV's children until no overlap is found and the query ends, or a leaf node is reached and low-level collision detection takes over. Since BVs completely enclose the objects they represent, if two BVs do not overlap their respective objects cannot possibly be in collision.

BVs are usually convex, due to the simplicity of construction and overlap tests provided by such geometry. The choice of BV is a balance of construction speed, as some volumes are more expensive to compute, query speed, as complicated volumes take longer to test for overlap, and the amount of culling provided, as tighter fitting BVs provide better culling for objects in close proximity, saving expensive low-level tests. The optimal choice is often dependent on the scene, geometry, and type of simulation.

For more information about choices of BVHs, their construction, fitting, querying, and more, see Ericson [2004].

Frontier In a synchronous setting, to detect all collisions, a BVH representing the entire scene is queried against itself. The root nodes (which are equal) trivially overlap, and the queries recurse to testing each child node against itself and all others. At some point this query ends, either by arriving at leaf nodes or successfully culling sub-branches of the tree

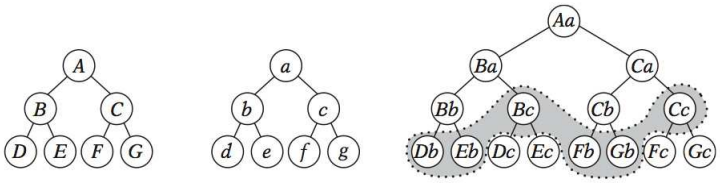


Figure 5.7: Two intersected BVHs, and the resulting frontier

from further consideration (Algorithm 6).

Algorithm 6 BVH Traversal Algorithm

```
1: traverse(A, B)
2: if A and B do not overlap then
3:    return
4: end if
5: if A and B are leaves then
6:    flag A and B as potentially-colliding
7: else
8:    for each child A_i of A and B_j of B do
9:       traverse(A_i, B_j)
10:   end for
11: end if
```

The pairs of nodes reached during a hierarchy traversal is called the *frontier*, because it describes the extent to which BV overlap tests traversed. Not surprisingly, the temporal coherency of physical simulations results in very little movement of the frontier between successive queries. Notwithstanding, this information is lost and the full traversals are usually repeated, only to arrive at or very near the same frontier.

For this reason, the technique of *front tracking* was developed [Li and Chen, 1998]. Front tracking "remembers" the frontier and continues traversal from this set during successive queries (Figure 5.7). The frontier can be stored as a list of BV pairs that represent the traversal's termination point. This list is called a *separation list*.

This notion of using temporal coherency to incrementally update geometric data struc-

tures leads very naturally to a KDS for tracking the frontier of a BVH.

Kinetic separation list The broad-phase algorithm we use is a slightly modified version of the separation lists KDS described by Weller and Zachmann [2006]. Separation lists use the concept of a frontier to maintain a list describing the last-known separating nodes in a BVH. This list is incrementally updated using the dynamics of the simulation.

In the modified structure, there is only a single certificate for each separation pair in the list:

- *The two BVs in the pair do not overlap.*

This is a modification because leaf nodes are treated differently than by Weller and Zachmann [2006]. When leaf BVs (representing triangles in 3D) overlap, control is handed over to the separation slab KDS and is no longer represented by a separation list.

This additionally simplifies the events corresponding to a certificate failure:

- *The two BVs in the pair overlap.* Clearly at the time that the BVs overlap no certificate exists and the structure must be modified by creating new certificates, one for each child-child pair of the two BVs.

- *The two parents of the two BVs in the pair no longer overlap.* By definition of frontier, a separation pair only exists if the parents in the tree overlap, otherwise the frontier would be one level higher. While the previous event corresponds to the frontier "moving down", this event represents the frontier "moving up" as separation between objects occurs. When this event occurs the separation pair is removed from the list and replaced by a separation pair for the two parent nodes.

During each rescheduling, the times of the preceding two events is computed with the earlier one scheduled on the queue. Of course, this event may never come to fruition, as a modification of flightplans (via a force event) may occur sooner, altering the scheduled time.

Since we are concerned with convex BVs (as is most work in collision detection), the problem of rescheduling simplifies to finding the interval in time during which the BVs are

overlapping. For two parent BVs, the end of this interval is the time they no longer overlap, and for two separation pair BVs, the start of this interval is the beginning of overlap.

No mention so far has been mode of the choice of BV. In Weller and Zachmann [2006], Axis Aligned Bounding Boxes (AABBs) were used, and all analysis is presented based on these. For our implementation, we use k-DOPs [Klosowski *et al.*, 1998; Zachmann, 1998] as our bounding volume, with $k = 18$. The use of alternative BVs is investigated in Chapter 7.

5.4 Algorithm

As mentioned in §5.2, AVIs are implemented using a priority queue. This priority queue consists of events representing elemental forces that occur throughout time. When that event is reached, the new simulation time is the time of that event, and positions and velocities are updated appropriately.

In §5.3, we saw that KDSs are also implemented using a priority queue consisting of time-sorted events. These events, however, represent changes in some geometrical structure (external events), or "helper" events that support the maintenance of this structure (internal events). The times of these events are computed based on the current trajectories, and thus the events are subject to a recomputation of these times whenever those trajectories are altered.

Here we present a unified algorithm that integrates forces, including contact using the discrete penalty layers of Chapter 4, using asynchronous variational integrators, and dynamically adds in contact force events as they are needed using kinetic data structures. The full construction is given in Algorithm 7.

We see in this algorithm the core elements of Algorithm 5, upon which our method is based. The whole procedure operates around the notion of *events*, which have distinct meanings for both AVIs and KDSs. We unify this into a single concept.

Events Quite simply, we view an event as an occurrence that can be represented by a distinct moment in time. This time allows insertion of the event into the time-sorted priority queue.

Algorithm 7 The full asynchronous contact mechanics algorithm

loop
  $(E, t) \leftarrow Q$.**pop**
  for $i \in$ **stencil**$(E) \bigcup$ **support**(E) **do**
    $X_i \leftarrow X_i + (t - t_i)\dot{X}_i$ {advance vertex to current time}
    $t_i \leftarrow t$ {update vertex's clock}
  end for
  if E is a force event **then**
    $\dot{\mathbf{q}}_\xi \leftarrow \dot{\mathbf{q}}_\xi - hM_\xi^{-1}\partial V/\partial \mathbf{q}_\xi$ {local impulses, local mass}
    Q.**push**$(E, V, h, t + h)$ {Return the event to the queue, with new time}
    for $j \in \bigcup_{i \in \xi}$ **contingent**(i) **do**
      $s \leftarrow$ **failureTime**(E_j) {compute new event time}
      Q.**update**(E_j, s) {reschedule the contingent event}
    end for
  else if E is certificate failure **then**
    update KDS certificate, reschedule in Q
    (de)activate penalty forces
  end if
end loop

	Event	Support	Stencil
Force	Gravity	$\emptyset$	Entire mesh
	Stretching	$\emptyset$	Triangle
	Bending	$\emptyset$	Hinge
	Penalty	$\emptyset$	Pair of primitives
Contingent	Separation slab	Pair of primitives	$\emptyset$
	k-DOP separation pair	Those in k-DOP	$\emptyset$
Benign	Snapshot	$\emptyset$	$\emptyset$

Table 5.1: Events and their associated supports and stencils.

We can partition events into three disjoint sets:

1. **Force events** are distinguished by the fact that they alter the trajectory of elements in the mesh by applying impulses to velocities. This category includes events representing gravity, internal strain, and contact forces.

   We call the set of vertices whose trajectories are altered by this event the event's *stencil*. This is useful in determining which events need to be rescheduled on the queue.

2. **Contingent events** are those events which do not modify any trajectories, but whose scheduled times are dependent on trajectories of some subset of the mesh. We call the vertices on which an event depends the *support* of the event.

   Whenever the trajectory of any vertex in an event's support is altered, the event's scheduled time may be invalid, and thus recomputation of this time is necessary.

   In our implementation, contingent events are entirely represented by KDS certificate failure events.

3. **Benign events** are those events which do not depend on any vertices and do not alter any vertices. Both the stencil and the subset of a benign event are empty.

   Our simulator contains only one benign event: a snapshot event that outputs recorded frames of the configuration at fixed points in time.

As expected, force events are fixed throughout time, since they do not depend on any particular configuration of the mesh. Technically, this includes contact force events. However in practice, it is far too inefficient to always process penalty layer events. Furthermore, one can never know *a priori* the deepest penalty layer to activate. As a result, penalty layer events can be added and removed from the queue. They are added when a KDS event determines that a penalty force between two primitives is active (non-null).

We remove penalty force events in a lazy manner. When the force event fires, we check if the force evaluates to zero. If it is null, and we see the primitives are separating, we remove the event from the queue.

Note that although penalty forces are added and removed dynamically, their force computations are fixed throughout time. When we determine at a time t that a penalty force with timestep h_i is to be activated, the penalty force event is *not* scheduled for time t, but rather for time nh_i, where $n \in \mathbb{N}$ is the smallest positive integer for which $nh_i > t$. This ensures the integrator's conservative properties are maintained.

Rescheduling An important consideration in our algorithm is the rescheduling of events. Rescheduling of force events is straightforward. All force events have a constant timestep associated with them, and thus the next occurrence of the event is trivial to determine.

Rescheduling of contingent events is more problematic. Whenever a force event alters trajectories, we must recompute the event time for all contingent events whose support contains one or more of the altered vertices. This can be quite expensive, especially when force events occur frequently, such as during periods of heavy contact.

The new event time is found in the same manner as finding certificate failure times, only with the new trajectories used as input.

5.5 Discussion

This chapter addressed the two main problems in implementing the discrete penalty layer method of Chapter 4.

First, we augment the method of asynchronous variational integrators to include integration of penalty forces, rather than forces associated with fixed mesh elements. This

allows us to freely integrate any number of penalty layers independently.

Second, we modify our integration algorithm to include kinetic data structure event handling. This efficiently tells us precisely when to add new penalty layers to the collection of force events. With this addition, we never need to know at what depth penalty layer a simulation must be running; the algorithm will automatically adapt to whatever layer is needed at that time. This includes the continuous movement to shallower layers (or no layers at all) when mesh primitives separate.

AVIs are one method of independently timestepping elements. There is an existing body of work in multistepping algorithms, which substep stiffer forces [Smolinski *et al.*, 1996; Daniel, 1997; Daniel, 1998; Smolinski and Wu, 1998]. These methods are not purely asynchronous, but rather result in the integration of weak forces at integer multiples of the stiff forces in the system [Neal and Belytschko, 1989]. We have not investigated the use of these methods to implement discrete penalty layers, but it would be an interesting experiment to compare performance between AVIs and this class of methods. Kinetic data structures would not be as cleanly integrated into the system, as enabled by the AVI priority queue. Traditional KDSs or other means of predictive proximity detection would be required to enable penalty layers.

Chapter 6

Results

This chapter presents results obtained from our implementation of discrete penalty layers using the technologies presented in Chapter 5.

6.1 Guarantees

The integration of the preceding construction provides four guarantees:

Safety Safety represents a method's robustness in preventing collisions and maintaining contact. More than just experimental evidence of safety, a proof of safety is strongly desired.

Here we prove, given functions $r(l)$ and $\eta(l)$, the conditions for which safety is guaranteed, with the sequence of discrete penalty layer potentials forming a barrier at $g = 0$.

Theorem 6.1.1. *The nested potentials of Equation 4.1 form a barrier.*

Proof. For these nested potentials to be a barrier, the cumulative energy of these potentials must diverge as the distance between two primitives vanishes. Let $d = \|X_b - X_a\|$ be the separation distance between two objects a and b. Then $l = \lfloor \eta^{-1}(d) \rfloor$ is the currently active penalty layer. We take the limit as the separation distance approaches zero:

$$\lim_{d \to 0} \sum_{i=1}^{l=\lfloor \eta^{-1}(d) \rfloor} \frac{1}{2} r(i) g_i(d)^2 = \lim_{d \to 0} \sum_{i=1}^{l=\lfloor \eta^{-1}(d) \rfloor} \frac{1}{2} r(i) \left(d - \eta(i)\right)^2$$

As the distance approaches zero, the penalty layer goes to infinity.

$$\lim_{d\to 0} \sum_{i=1}^{l=\lfloor \eta^{-1}(d) \rfloor} \frac{1}{2} r(i) \left(d - \eta(i)\right)^2 = \frac{1}{2} \sum_{i=1}^{\infty} r(i)\eta(i)^2$$

Thus, as long as $r(i)\eta(i)^2$ is a divergent series, a barrier is formed at $d = 0$. □

Looking at our common choice of $r(l) = r(1)l^3$ and $\eta(l) = \eta(1)l^{-\frac{1}{4}}$, we see that the summation above diverges. A poor choice of functions would be $\eta(l) = \eta(1)l^{-2}$ and $r(l) = r(1)l^2$, since

$$\begin{aligned}
\frac{1}{2} \sum_{i=1}^{\infty} r(i)\eta(i)^2 &= \frac{1}{2} \sum_{i=1}^{\infty} r(1)l^2(\eta(1)l^{-2})^2 \\
&= \frac{1}{2} r(1)\eta(1)^2 \sum_{i=1}^{\infty} l^2 l^{-4} \\
&= \frac{1}{2} r(1)\eta(1)^2 \sum_{i=1}^{\infty} l^{-2},
\end{aligned}$$

which is a bounded series.

Progress Progress conveys a solution's ability to reliably move forward in simulation time in reference to real time, or wall-clock time. When evaluated in an absolute sense, a method either guarantees progress is made or has the potential of finding itself in situations it is unable to solve.

Our construction guarantees progress for a *well-posed* problem with bounded energy.

Theorem 6.1.2. *Given a well-posed problem with finite energy, a simulation that uses the nested potentials of Equation 4.1 is guaranteed to progress.*

Proof. For a given separation distance d at a contact point, the active penalty layer is $l = \lfloor \eta^{-1}(d) \rfloor$. By construction, $\eta^{-1}(d) \to \infty$ as $d \to 0$, and $\eta^{-l}(d) = \infty$ only if $d = 0$, so for $d \neq 0$, a condition guaranteed by the barrier proof, l is finite. A finite l implies a finite $r(l)$, the stiffness of the active penalty layer. Every finite stiffness of a linear force (quadratic

$\eta(l)$	$r(l)$	Kinetic Energy	Cloth Vertex ($\frac{cm}{s}$)	Sheet Metal Vertex ($\frac{cm}{s}$)
$\frac{1}{l}$	$1000l^3$	6.93375×10^5	5.8880×10^3	3.0406×10^2
$l^{-\frac{1}{4}}$	$1000l^3$	9.17877×10^8	2.14228×10^5	1.10627×10^4
$l^{-\frac{1}{4}}$	$1000l^4$	5.81644×10^{11}	5.39279×10^6	2.78482×10^5

Table 6.1: Practical bounds for choices of layer distribution and stiffness functions. We give the bounds on kinetic energy and the velocity a cloth vertex (density $0.02\frac{g}{cm^2}$) and sheet metal vertex (density $7.5\frac{g}{cm^2}$) would have to attain to successfully tunnel through the barrier. We see these bounds are far beyond the needs of most simulations.

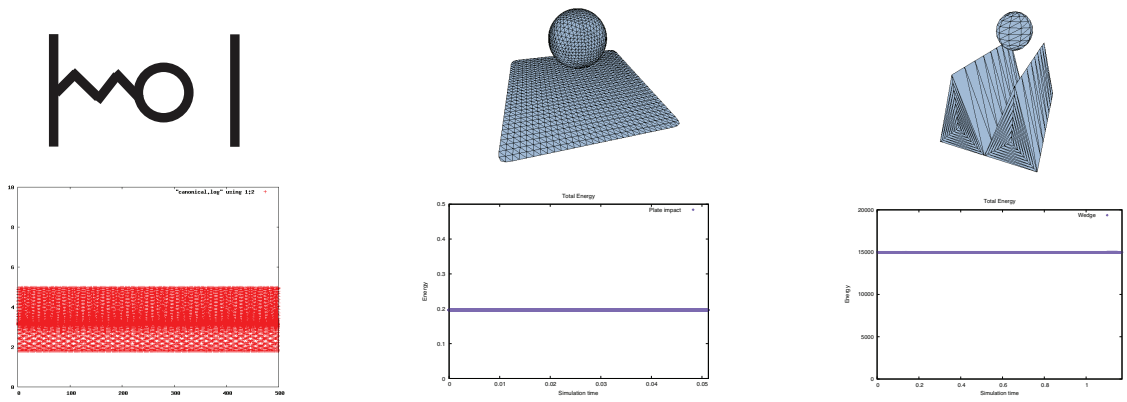
potential) has a non-zero stable timestep, and hence the simulation always moves forward in time. □

This theoretical proof has one implicit assumption: that every real number is representable by a computer. We know this assumption to be false; computers have finite precision. Thus, when a layer's stable timestep drops below the machine epsilon (the number ϵ for which $1 + \epsilon = 1$), the integrator is unable to move forward in time.

In terms of penalty layers, this means that there exists a finite amount of energy that will be able to halt progress. Note that safety is not violated, the simulator does not step past the point where $d = 0$, but rather is unable to move forward.

Appendix A contains *Mathematica* code that computes this amount of energy given functions $r(l)$ and $\eta(l)$. In practice this bound is large, far beyond the traditional needs of the physical configurations we are interested in simulating. Table 6.1 gives these bounds on kinetic energy for different choices of distribution and stiffness functions, along with the corresponding velocities for a typical cloth and sheet metal vertex.

Correctness Correctness represents the degree to which a proposed solution is physically consistent, in particular its ability to satisfy known physical laws, including conservation of energy and momentum. While there has been some research into human perception of physicality, all simulators essentially have are known physical laws to gauge correctness and provide a quantitative measure for realistic simulations.



(a) Canonical 1D example (b) Ball striking a plate (c) Ball striking a wedge

Figure 6.1: Three energy plots

The construction of discrete penalty layers asserts correctness; indeed, each layer is constructed from a potential energy, leading to conservative forces. Respect of physical laws past this construction relies on the method of integration, a problem discussed in §5.2.

Figure 6.1 shows total energy (kinetic and potential) profiles for a few illustrative examples. Figure 6.1(a) is a canonical 1-dimensional construction [Wriggers and Panagiotopoulos, 1999]. A particle is attached to a wall by a spring, then placed close to another wall. An initial load is placed on the spring, causing the particle to collide repeatedly. We see the oscillations in energy expected from variational integrators, but energy neither consistently increases nor decreases, instead oscillating about the conserved amount.

Figure 6.1(b) repeats an example from Cirak and West [2005]. In this setup, a ball strikes a stiff plate, sending vibrations through the materials. We again see reliable energy behavior throughout.

The last setup (Figure 6.1(c)) is from Pandolfi *et al.* [2002]. In this example, a stiff ball impacts between two wedges, causing the ball to repeatedly bounce as it falls into the crevice. Our algorithm efficiently handles this configuration while delivering the desired correctness.

In addition to conservation of energy and momentum, correctness ideally includes the notion of *causality*. Causality dictates a strict ordering on events in nature. Fortunately the penalty layers treat collisions strictly in causal order. No velocity filter "grouping" is

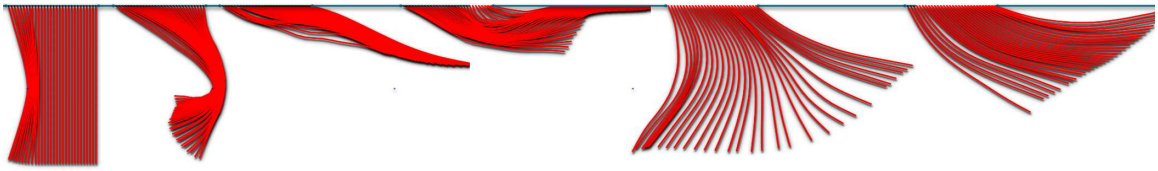


Figure 6.2: The importance of causality is illustrated by this example, where a scripted ball pushes a dense stack of curtains into one another.

required to guarantee simulation progress. Figure 6.2 demonstrates the importance (and satisfaction) of causality. A high-speed ball pushes through a sequence of densely-packed curtains. Very clearly, the ball pushes curtains into each other in a well-defined order.

Stability Penalty layers are quadratic forces, yielding constant second derivatives. This results in a timestep that is unconditionally stable for each penalty layer. Asynchronous variational integrators allow integration of each layer at its own stable timestep. The cumulative energy approximates the non-linear potential of Chapter 2 without the stability issues.

6.2 Parameters

Another benefit of asynchronous contact mechanics is a vast reduction in the number of user-defined parameters. Existing methods require adjusting a large number of parameters that are essentially arbitrary, *i.e.*, have no physical basis. As a result, the success of the simulation depends heavily on these "magic numbers."

Material parameters are the minimal set of acceptable system parameters, excluding contact parameters. They encompass those parameters that define the behavior of a material in a simulation.

For our simulations of deformable surfaces, this includes gravity, mass damping, bending stiffness, bending damping, stretching stiffness, and stretching damping. This list is identical for a synchronous simulation of the same material.

We differentiate between these parameters and others in that there exists some physical intuition, or a physical analogue guiding the selection of material parameters. For instance, if a material appears too stretchy, it is natural to assume an increase in stretching stiffness would alleviate the problem.

Contact parameters are those defining the behavior of contacts. For ACM simulations one must define the stiffness function $r(l)$ and the distribution function $\eta(l)$. §6.1 outlines the requirements for these two functions. In particular, the divergence property *must* be met in order for the enumerated guarantees to hold. Fortunately, this property is easy to verify mathematically.

Given choices for $\eta(l)$ and $r(l)$ that satisfy this property, behavior can still vary between identical simulations. These parameters control how quickly contacts are resolved and how many layers are used. Figure 6.3 shows the trajectory for a particle thrown towards a fixed floor in two dimensions, for two different choices of $\eta(l)$. For high-impact situations it is advantageous to enter deeper layers earlier, while this could be unnecessarily slow for low-impact velocities and resting contact.

Additionally, coefficient of restitution and friction coefficients for each simulated material must be specified to fully determine the behavior of contacts. These are similar to material parameters, where intuition and desired output guide parameter selection.

One key characteristic of ACM simulations is the lack of *arbitrary parameters*. Arbitrary parameters are those that have no physical guidance or means of non-experimental verification. In the model described in Chapter 3, these included the number of iterations taken in the second pass as well as different tolerances and thresholds that are simulation dependent. The only means of selection for these parameters is trial and error, yet the robust resolution of contact relies on finding optimal values. This non-intuitive parameter manipulation can be frustrating for users.

ACM simulations have no such parameters. Stiffness and distribution functions can be verified mathematically, and therefore are not arbitrary. All prior guarantees hold regardless of choices for material and contact parameters (as long as the combined stiffness and distribution functions diverge). This allows the user to spend time designing a simulation,

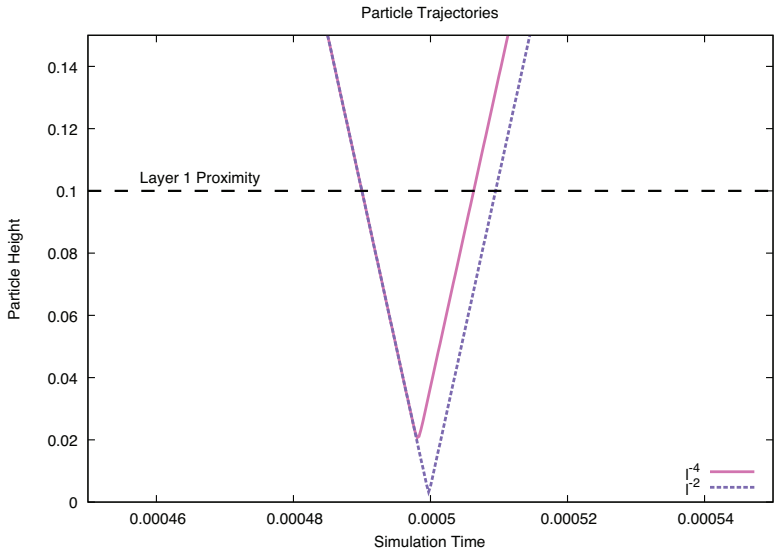


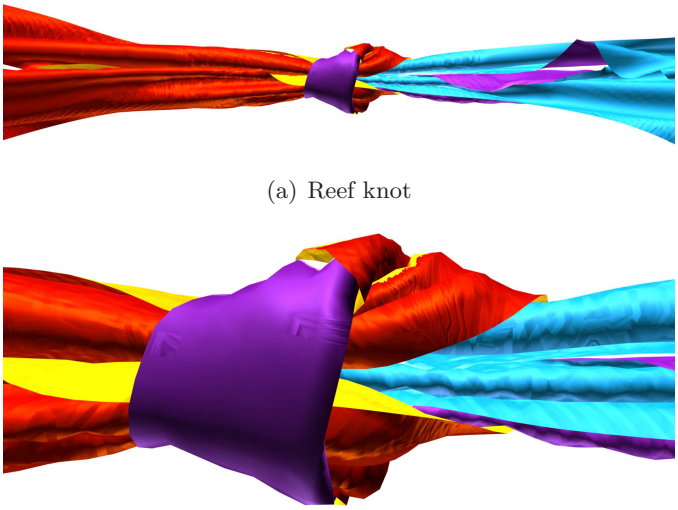
Figure 6.3: The height of a 2D particle plotted as a function of time with two different distributions. The particle has an initial velocity of $(2, -10000)$. The first distribution, $\eta(l) = \eta(1)l^{-4}$, resolves the contact sooner, leaving a discrepancy between the two trajectories. This quicker resolution comes at the cost of smaller timesteps earlier in the contact.

rather than manipulating arbitrary parameter choices.

6.3 Experiments

We described simple experiments and empirical measurements supporting the guaranteed safety and good energy behavior of the proposed contact algorithm. We turn our attention to challenging problems involving complex contact geometries, sharp features, and sliding during extremely tight contact.

Knots We simulate the tying of ribbons into reef and bowline knots (see Figures 6.4 and 6.5, respectively). The ribbons are modeled as a loose knot, assigned a material with stiff stretching and weak bending, and their ends are pulled by a prescribed force; the bowline knot requires also the prescription of fixed vertices behind the cylinder where a finger normally holds the material in place. The final configuration is faithful to the shape of actual "boyscout manual" knots.



(a) Reef knot

(b) Reef knot closeup

Figure 6.4: Simulated tying of ribbons into a reef knot.

This example demonstrates the strength of asynchrony in allocating resources to loci of tight contact. As the knot tightens, progressively finer time steps are used for the tightest areas of contact. If instead of prescribing reasonable forces we directly prescribe an outward motion of the two ends of the ribbon, the simulations execute to the point where the mesh resolution becomes the limiting factor, *i.e.*, a tighter knot cannot be tied without splitting triangles; past this point, the computation slows as penalty interactions burrow to deeper layers and the mean time step decays. This highlights both a feature and a potential artistic objection to the method: when presented with an impossible or nearly-impossible situation (non-stretchy ribbon with prescribed diametrically opposing displacements at its ends) the method halts as the bound on kinetic energy is surpassed.

Bed of nails We crafted a problem to test the handling of isolated point contacts and sharp boundaries. Four sliver triangles are assembled into a nail, and many such nails are placed point-up on a flat bed. We drape two stacked fabrics over the bed of nails (see Figure 6.6), and observe that the simulated trajectory is both realistic and free of penetrations, oscillations, or any other artifacts typically associated to contact discontinuities. Next, we prescribe the motion of one end of the fabric, tugging on the draped configuration to demonstrate sliding over sharp features.

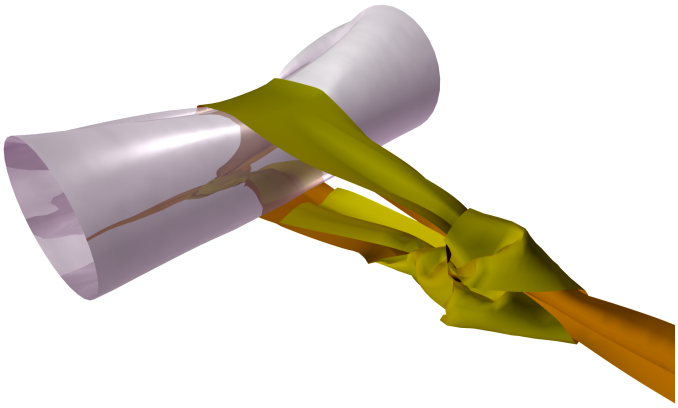
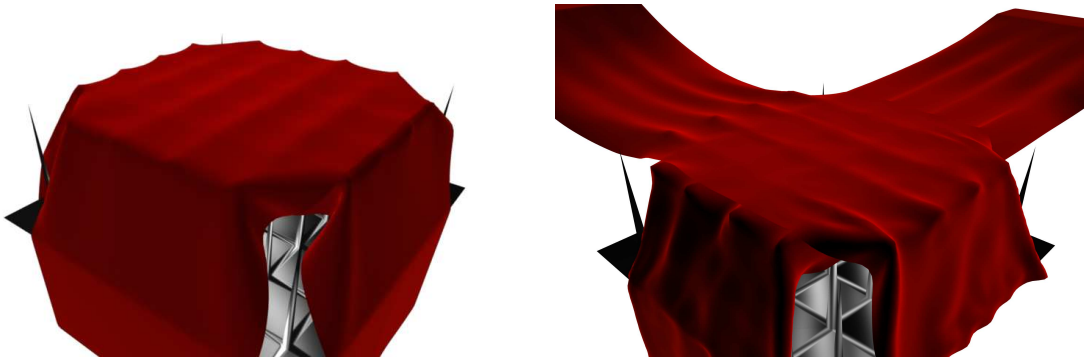


Figure 6.5: Simulated tying of a ribbon into a bowline knot.



(a) (b)

Figure 6.6: Experiments with a bed of nails highlight the method's ability to deal with sharp boundaries, isolated points of contact, sliver triangles, and localized points of high pressure between two nearly incident surfaces.

We extend the bed of nails into a landing pad for various coarsely-meshed projectiles. Variably-sized to barely fit or not fit between the nails, and thrown with different initial velocities and angles, the projectiles exhibit a wide array of behaviors, including bouncing, rolling, simple stacking, ricochetting at high frequencies (this requires resolving each collision when it occurs, as resolving collisions over a fixed collision step size can cause aliasing that prevents the ricochet); sliding and getting stuck between nails (the sliding requires a deformable model and friction, since a perfectly rigid object would be constrained to a sudden stop by the distance).

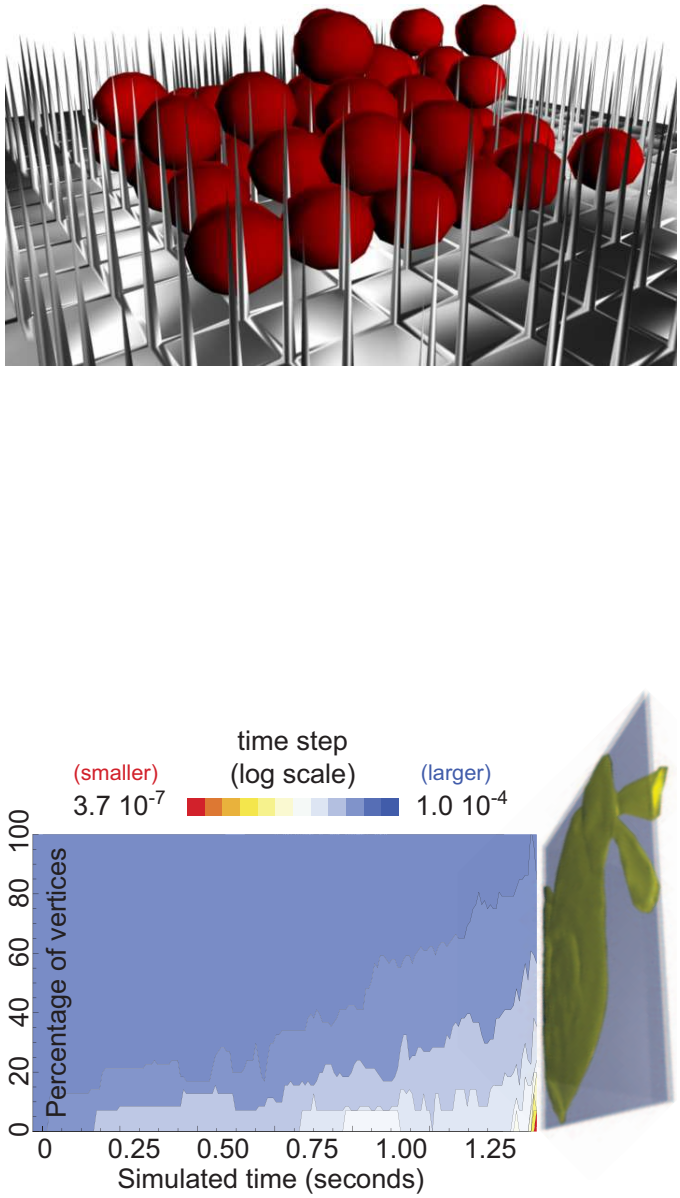
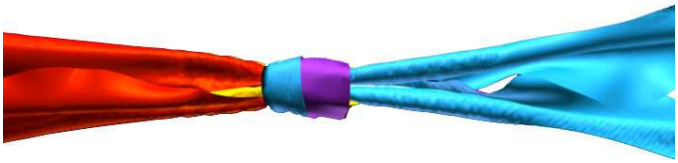
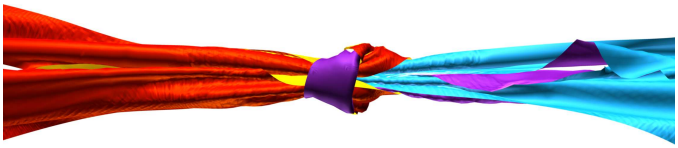


Figure 6.7: As the bunny is compressed by the scripted walls, a larger percentage of vertices participate in deeper penalty layers.



(a) Synchronous reef knot



(b) ACM reef knot

Figure 6.8: Synchronous simulation takes over an order magnitude longer to achieve comparable results, without any guarantees.

Comparison with synchronous We ran the reef knot tying example with the synchronous simulation framework of Chapter 3. Despite giving the method over 12 times as much time to run, the synchronous simulation fails to simulate as far as ACM without allowing penetrations. Figure 6.8 shows the last, penetration-free frame obtained from this simulation, side-by-side with the final frame of the ACM simulation.

Timing We list computation time for the various examples, as executed on a single thread of a 2.33Ghz Intel Xeon with 8GB RAM (Table 6.2). Over 99% of runtime is allocated to the maintenance of the kinetic data structures used for collision detection.

Parameters We list parameters for the various examples. Bending and stretching stiffness refers to the Discrete Shells [Grinspun *et al.*, 2003] and common edge spring models.

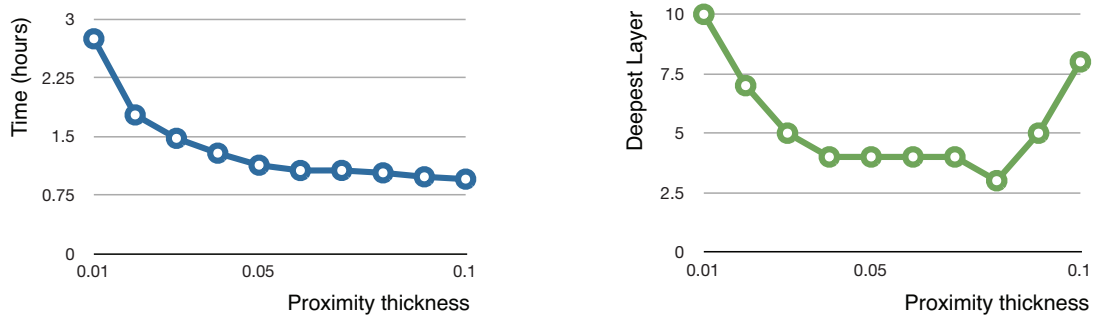
Examples	Vertices	Simulation seconds	Total time (hours)
Bunny Compactor	1768	2.0	1.6775
Bowline Knot	3995	5.0	86.5775
Reef Knot	10642	5.4	35.2347
Two Sheet Drape	15982	2.5	77.6616

Table 6.2: Timings (in hours) for examples executed on a single thread of a 2.33Ghz Intel Xeon with 8GB RAM.

Example	Density	COR	r(1)	η(1)	Stretching Stiffness	Stretching Damping	Bending Stiffness
Reef Knot	0.1	0.0	1000.0	0.1	750.0	0.1	0.01
Bowline Knot	0.01	0.0	1000.0	0.1	100.0	0.1	0.01
Bunny Compactor	0.01	0.01	10000.0	0.05	1000.0	0.0	1000.0
Trash Compactor	0.001	0.01	1000.0	0.05	1000.0	15.0	10.0
Two Sheets Draped	0.001	0.0	1000.0	0.1	1000.0	1.0	0.1
Reef Knot Untied	0.1	0.0	1000.0	0.1	1000.0	0.1	0.01
Two Sheets Pulled	0.001	0.0	1000.0	0.1	1000.0	1.0	0.1
Balls on Nails	0.016	0.3	10000.0	0.1	50000.0	1.0	100000.0
2D Sludge	-	0.0	1000.0	0.1	-	-	-

Additionally, we examine the effect proximity thickness of the first layer has on the runtime of a simulation.

In this setup, we simulate a thin-shell bunny model between two walls scripted to crush the bunny. We vary the proximity thickness of the first penalty layer and measure computation time for a portion of the simulation (The first 1.15 simulated seconds). The latter segment of the simulation, where the bunny is entirely crushed, is not representative of normal contact situations, so we exclude it from our measures.



(a) Runtimes plotted as a function of $\eta(1)$.

(b) The deepest penalty layer required as a function of $\eta(1)$.

Figure 6.9: Higher proximity thickness for the first layer allows for more room between successive layers, giving each layer more distance to prevent the collision. As thickness increases, elements are identified as being in contact sooner, and reach deeper layers quickly. However, as these contacts are initially resolved, the system maintains contact within shallow layers and runs quickly.

Figure 6.9 plots the results of this experiment. Increasing proximity thickness decreases overall runtime. The increased distance between penalty layers gives each layer more time to act and prevent further contact before entering the next layer. At the beginning of the simulation, close elements quickly enter higher penalty layers for larger proximity thicknesses. After this initial contact is resolved, the simulation returns to low penalty layers.

Based on this data, we recommend users construct $\eta(l)$ with $\eta(1)$ as high as possible without compromising visual quality in order to achieve faster simulation runtimes.

Dissipation To test the energy behavior for a variety of coefficients of restitution, we simulated a box of 900 particles with random initial velocities. Figure 6.10 shows the energy of the system as a function of time for multiple values of e_{COR}; in all cases energy decays smoothly and predictably.

Pöschel and Schwager [2005] describe experiments with granular media. They observe that large numbers of particles participating in frequent, dissipative collisions form *clusters*, or groups of proximate particles with very little relative velocity, over time. Figure 6.11 illustrates that our method reproduces this clustering when the above experiment is run

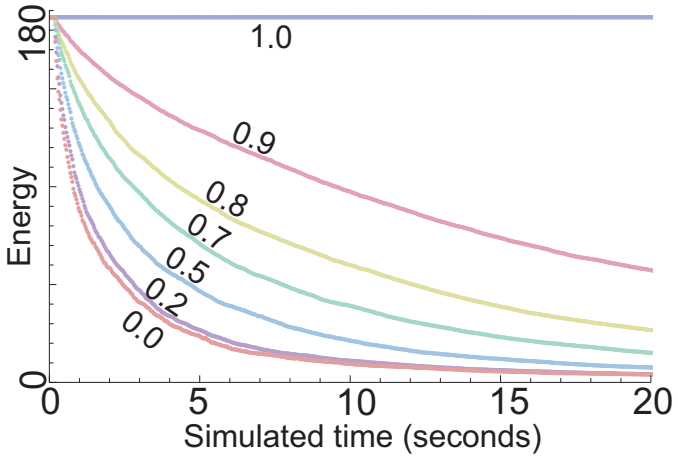


Figure 6.10: Energy over time of a closed box of particles, for different coefficients of restitution.

with $e_{\mathrm{COR}} = 0$.

Impulse-based collision response methods cap the magnitude of the Coulomb friction force, so that a large normal impulse does not cause relative tangential motion to reverse direction. Our implementation does not cap, because we have not identified a capping strategy that is compatible with order-independence of simultaneous events. For a pair of primitives in contact, friction is applied piecemeal, at the ticks of the penalty layer clocks, instead of as a single impulse. This serves as a reasonable discretization of kinetic friction, but it is certainly a crude approximation of static friction. In particular, it is possible for a friction update to reverse relative tangential motion; the magnitude of this reverse motion is bounded by $\mu r(l)\eta(l)h$, so it can be limited by choosing sufficiently small stiffness function r or time step h. Structures whose stability depends on static friction, such as the house of cards simulated by Kaufman *et al.* [2008], would benefit from future work developing a more complete treatment of friction.

As a test of our friction model, we applied gravity to the box of particles described above, and allowed the particles to come to rest on the floor of the box. We then removed the right side of box and replaced it with a downwards slope. Figure 6.12 shows the configuration of the balls 2.5s after removal of the wall; the result varies with the coefficient of friction. When no friction is applied, the particles flow freely down the slope. As friction

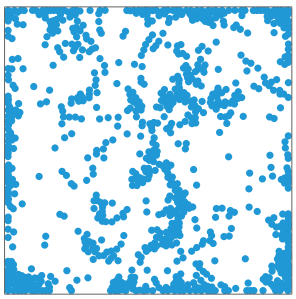


Figure 6.11: Dissipative collisions form characteristic clusters.

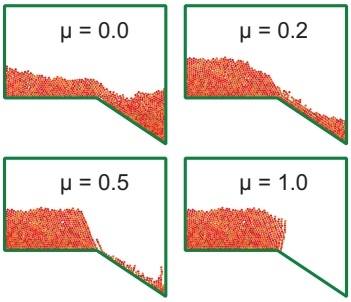


Figure 6.12: Friction alters the flow of sludge down an incline.

is increased, the rate of flow decreases. Note that a simulation of granular materials should store as a state variable the angular momentum of each grain [Pöschel and Schwager, 2005]; our implementation neglects this, evidence a small vertical stack of grains that slides down the inclined plane without tipping.

Source code The full source code implementing everything described here can be found at the project page, `http://www.cs.columbia.edu/cg/ACM/`.

6.4 Discussion

Discrete penalty layers satisfy the criteria set forth in Chapter 1. Their fundamental nature in quadratic penalty potentials firmly establishes its physical roots, and the non-linear nature of the sum of the potentials solidifies robustness. The fixed clock timesteps of linear forces gives the guarantee of progress through time.

Limitations When properly implemented, discrete penalty layers will never move into a configuration that violates unilateral contact laws. However, there is a practical limit to the energy that they can tolerate, after which the method will halt in the name of safety. This limit depends on the growth functions of the potential energy, and there exists easy choices that guarantee reliable progress with "normal" simulations (non-extreme momentum).

We further equipped this construction with a dissipation model capturing both coefficient of restitution and a Coulomb model for friction. This allows the model to represent

the widest possible variety of simulations. While simple, our approach has a drawback in the inelastic limit $e_{\mathrm{COR}} = 0$: the penalty impulses can leave as residue a small separating relative velocity; a side effect of the penalty method approximation and the imprecise effect of coefficients of restitution. The magnitude of this velocity is at most $r(l)\eta(l)h$, where h is the layer's time step, so it can be limited by choosing a small enough $r(l)$ or h. It would be interesting to critically damp the linear half-spring so that this residual velocity is eliminated. Experimental evidence has suggested that coefficients of restitution are functions of relative velocity [Hunt and Crossley, 1975; Marhefka and Orin, 1996], giving such an implementation physical merit.

We experience further difficulties when handling static friction. This is a known problem within penalty-based methods. It is difficult to determine the exact limit on the amount of friction applied, and thus the tangential direction could end up reversed. This reversal is a function of the timestep size—an undesirable result. We alleviate this problem using the notion of virtual particles [Lee and Herrmann, 1993]. When a contact is activated, we record the point of contact and insert a 0-length tangential spring between those contact points. In this way, the total displacement since contact is used to compute the magnitude of the tangential force [Bell *et al.*, 2005].

Conclusion The asynchronous contact mechanics (ACM) algorithm allows the simulation of materials and setups that are impossible with today's state of the art. On the surface, however, they seem to perform at speeds much slower than what is considered acceptable today. Side-by-side speed comparisons of simple animations such as a cloth draping put ACM at a significant disadvantage. On the other hand, for an existing method to even begin to run one of the simulations presented here, such as the ribbon knots, ACM not only out-performs in speed, but manages to progress in the simulation past the point where other methods fail.

An additional advantage to our method is the lack of artificial parameters. The velocity filter model requires adjusting collision "epsilons", which are simulation-dependent. We guarantee resolution up to floating point epsilon, the smallest number representable on a finite machine. This introduces a new paradigm to simulation, where the user no longer

worries about modifying arbitrary numbers, and instead may focus on adjusting material properties to get the desired results. The physics "just work."

The separation lists were one choice of KDS to act as a broad-phase proximity detector. One advantage of this AVI and KDS integration is the plug-and-play nature of KDSs; we can cleanly exchange separation lists for any other kinetic data structure that serves the same purpose. Chapter 7 provides some preliminary investigation in making such a substitution.

A new approach to simulation opens opportunities for improvements. In Chapter 8, we offer improvements to ACM to enable faster simulation times. In the process, we begin to analyze the variables affecting ACM simulations, as a gateway to continuous improvements in asynchronous simulation.

Chapter 7

Broad-phase kinetic data structures

Chapter 5 presented a complete algorithm for the asynchronous simulation of deformable materials. One key to this algorithm is the use of kinetic data structures (KDS). KDSs provide the foresight to activate penalty forces as necessary: a necessity to claim the guarantees of Chapter 6.

We proposed the use of separation lists for broad-phase culling of potential collisions. Separation lists are built around a bounding volume hierarchy; our initial implementation used a k-DOP.

In this chapter, we explore alternative bounding volumes for broad-phase collision detection. In addition to separation lists, we also investigate the use of a kinetic spatial partitioning scheme; no prior work exists in this area. These grid-based methods prove quite efficient at pruning collisions in a traditional framework, so their use in a kinetic setting is a natural direction of inquiry.

7.1 Related work

Kinetic data structures were developed in computational geometry to maintain combinatorial structures of data in motion by exploiting the coherence that often exists in such motion [Guibas, 1998; Basch *et al.*, 1999; Guibas, 2004b]. Each KDS maintains some ge-

ometric attribute of the entire system. For instance, this could be a minimum spanning tree in a mobile ad hoc network [Agarwal *et al.*, 1998b] or the closest pair of objects in an interactive simulation [Basch *et al.*, 1997].

Good KDSs have been developed for a variety of problems pertinent to collision detection, including spatial proximity [Basch *et al.*, 1997; Erickson *et al.*, 1999; Agarwal *et al.*, 2002a; Gao *et al.*, 2003; Agarwal *et al.*, 2004; Gao *et al.*, 2005] (*e.g.*, collision detection, closest pair, clustering), extent or partitioning [Agarwal *et al.*, 1997a; Basch *et al.*, 1999; Agarwal *et al.*, 2002b; Agarwal *et al.*, 2005b] (*e.g.*, diameter, convex hull, *k*-*d* trees), visibility [Agarwal *et al.*, 1997b; Agarwal *et al.*, 1998a] (*e.g.*, binary space partitions, occlusions), and connectivity [Agarwal *et al.*, 1998b; Guibas *et al.*, 2000; Karavelas and Guibas, 2001; Gao *et al.*, 2006] (*e.g.*, minimum spanning trees, sparse spanners). In particular, generalizations of the spanner KDS may be relevant to our work.

Collision detection often uses structures that provide high-level pruning capabilities. Detection of this type is called *broad-phase*, since it allows for quick rejection of large batches of potential collisions. Due to the nature of these structures, it is natural that they would be kineticized. Our initial implementation of asynchronous contact mechanics used the kinetic data structure presented by Weller and Zachmann [2006], called separation lists, which extended the kinetic bounding volume hierarchy structures of Zachmann and Weller [2006]. We slightly modify the method as presented to include support for our novel separation slabs. See §5.3 for those details.

We restrict our investigation here to bounding volume hierarchies, with the exception of spatial partitioning. Due to the plug and play nature of KDSs in our framework, opportunity remains for the exploration of existing and future KDSs appropriate for the activation of penalty layers.

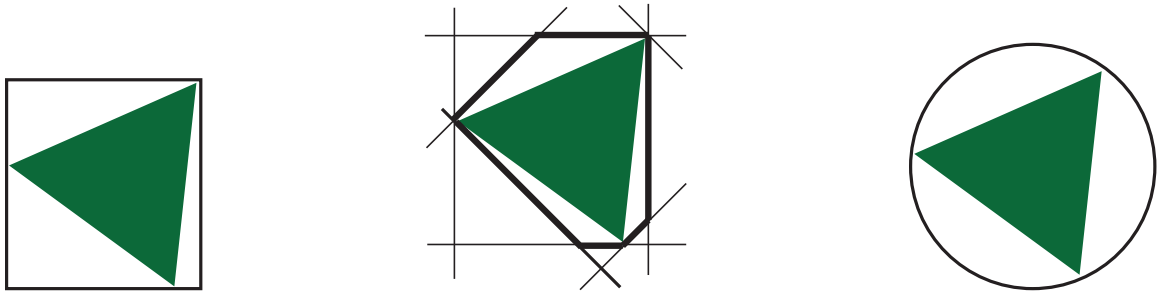
7.2 Bounding volume hierarchies

Section 5.3 described bounding volume hierarchies, frontiers, and separation lists. In short, the data structure "remembers" where the last traversal ended and computes the times when this structure will change.

For every frontier node, we put two events on the queue: one scheduled at the time when two bounding volumes will overlap and the other when the two parent bounding volumes will no longer overlap. We did this for simplicity of implementation, although performance could be improved by combining these two events into a single event.

We initialize by simply pushing one event on the queue. This event represents the time the root node for the entire scene will overlap with itself. This time is trivially zero, so new child events will instantly (and automatically) be scheduled. This continues until the representative frontier is on the queue and simulation time moves past zero.

In the following, we describe the choices of Bounding Volumes (BV) investigated, along with a discussion of the potential effect each may have on the KDS quality. We then detail how certificate failure times are computed for each. We do this in terms of time intervals where the BVs either overlap or not.

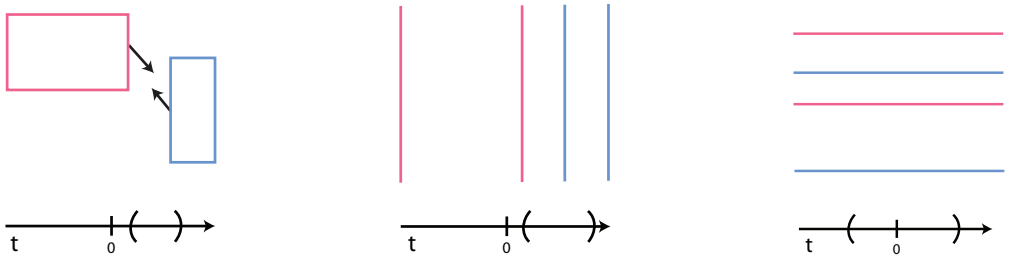


(a) AABB (b) k-DOP (for $k = 8$) (c) Sphere

Figure 7.1: Three choices of bounding volumes

Axis-aligned Bounding Boxes (AABBs) AABBs [Larsson and Akenine-Möller, 2001; Van Den Bergen, 2005] are one of the most common bounding volumes. In 3D, they are rectangular six-sided boxes (four-sided in 2D), oriented such that the sides are aligned to the coordinate axes. For this reason they can be examined quickly by just looking at individual coordinate values.

Certificate failure The certificate fails in two cases: either the BVs overlap or the two parent volumes in the hierarchy are no longer overlapping. Either way, we need to compute the time interval when two BVs overlap, since the latter is simply the



(a) The time interval the two AABBs overlap.

(b) The time interval the x-axes overlap.

(c) The time interval the y-axes overlap.

Figure 7.2: The time interval two AABBs overlap is taken as the intersection of the time intervals when the x bounds and the y bounds overlap.

complement of the former.

Two AABBs overlap when the spatial interval defined by the AABB's bounds along each axis overlap. Consider these bounds along one axis for two objects A and B:

$$(x^A_{min}, x^A_{max}) \text{ and } (x^B_{min}, x^B_{min}).$$

These intervals overlap when $x^A_{max} > x^B_{min}$ and $x^A_{min} < x^B_{max}$.

Each bound x^A_{min} and x^A_{max} has a corresponding velocity v^A_{min} and v^A_{max}, so we can compute when the above inequalities are true as a polynomial function of time. In fact, these functions are linear in the positions. Furthermore, we can modify it to detect when they are distance h, rather than actually touching,

$$f(A, B) = \frac{x^B_{min} - x^A_{max} - h}{v^A_{max} - v^B_{min}}.$$

We are interested in the interval where $f < 0$.

This only accounts for one set of bounds, to account for the other, we take the intersection of the resulting intervals. The start of the resulting interval is the time when the two original intervals along one axis overlap. If the interval is empty, then they never overlap.

We repeat this process for each of the three axes, taking the intersection of intervals. If at any point the resulting interval is empty, we can safely short-circuit the test. If

testing for the time when two BVs will *not* overlap, simply take the end of the final interval. Figure 7.2 illustrates this for a simple 2D AABB.

k-Discrete-orientation Polytopes (k-DOPs) k-DOPs [Klosowski *et al.*, 1998; Zachmann, 1998] can be thought of as generalizations of AABBs. Whereas AABBs are aligned so that their six bounding faces have normals in the three coordinate axis directions, k-DOPs have their k bounding faces oriented along $\frac{k}{2}$ fixed directions. This allows for potentially tighter fitting BVs at the expense of more axes to iterate over, both during fitting and overlap tests.

Certificate failure The process to find the times of overlap is identical to AABBs, whereas instead of three axes, we repeat the interval intersections for all $\frac{k}{2}$ directions.

Spheres Sphere trees [Hubbard, 1996; Palmer and Grimsdale, 1995] provide an interesting case study: intersections between them only require looking at two rates of change (the radii of each sphere and the motion of the sphere center). This one comparison is quadratic, compared to the linear equations of k-DOPs and AABBs.

Certificate failure The overlap times for spheres is somewhat simpler than for k-DOPs; instead of k bounds there is only one direction. The condition for two spheres to be a distance h apart is

$$\|x_A - x_B\| - r_A - r_B - h < 0,$$

where x_A and x_B are the centers of spheres A and B, with r_A and r_B their respective radii.

With spheres in motion, the centers have a velocity and the radii have a rate of change. We can then write this condition as a function of time

$$f(t) = \|(x_A - x_B) + t(v_A - v_B)\| - (r_A + r_B + h) - t(r_A^v + r_B^v) < 0.$$

This is a non-linear function, due to the distance calculation, but we can square both

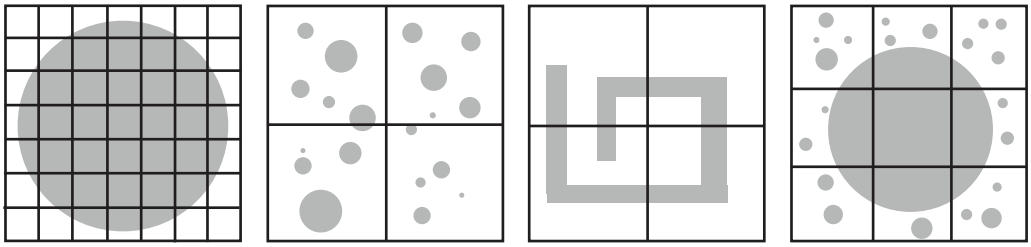


Figure 7.3: Left: A grid that is too fine. Left-center: A grid that is too coarse (with respect to object size). Right-center: A grid that is too coarse (with respect to object complexity). Right: A grid that is both too fine and too coarse.

sides to rewrite as a polynomial function of time

$$\begin{aligned} f(t) = &\left[(x_A - x_B) + t(v_A - v_B)\right] \cdot \left[(x_A - x_B) + t(v_A - v_B)\right] + \\ &2\left[(x_A - x_B) + t(v_A - v_B)\right]\left[(r_A + r_B + h) - t(r_A^v + r_B^v)\right] + \\ &\left[(r_A + r_B + h) - t(r_A^v + r_B^v)\right]^2. \end{aligned}$$

This polynomial is quadratic in time.

7.3 Spatial partitioning

The collision detection discussed thus far works by tracking the motion of simulated primitives using convex hulls. This is a *Lagrangian* view of tracking collisions, and we now discuss the complementary *Eulerian* view of collision detection.

Instead of identifying elements and explicitly following their locations in space, space can be discretized into regions, or cells, and the flow of primitives in and out can be monitored. This class of methods are referred to as *Eulerian grids*, or *spatial partitioning* [Ericson, 2004].

By tracking individual regions, the objects contained in any given cell are known at any time. An object being co-located with another object in a cell is a necessary, but not sufficient condition for collision. This provides the pruning necessary for efficient collisions.

All objects contained in a cell must be tested pairwise for collision, so reducing the number of these potential pairs is a key consideration in designing grids. Use cells that

are too large in relation to the size of primitives and a large number of pairwise tests are necessary, reducing to brute force $O(n^2)$ in the worst case. Conversely, extremely small cells require far too many cells updates as objects move through space [Cohen *et al.*, 1995].

We investigate the use of fixed size grids within ACM simulations. Fixed size grids use a pre-determined cell size, dx for 1D simulations, (dx, dy) for 2D simulations, and (dx, dy, dz) for 3D simulations. Note that this does not assume a uniform cell size (*i.e.*, cells may not form cubes). To be used by ACM, Eulerian grids must be *kineticized*, or adapted to work within the KDS framework. No prior work exists in this direction.

Kinetic Eulerian grids Every pair of primitives that share a cell must have a separation slab event to guarantee safety. Therefore a kinetic grid must track the time at which objects share cells, to create the appropriate slab. It must also track the time objects no longer share cells so that separation slab events can be removed from the queue. As an alternative, this can be done in a lazy manner, by simply removing events when they are noticed to be in separate cells, rather than actively tracking when this occurs.

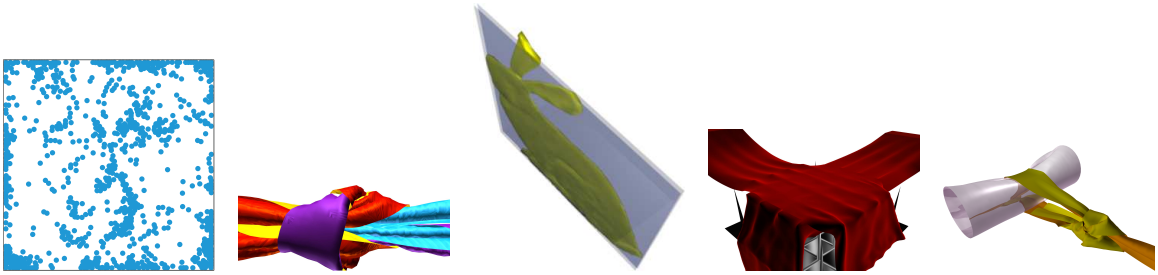
Both of these events can be combined into the maintenance of a single certificate per primitive (assuming an ordering on the cells in the grid):

- *Primitive i is contained by cell j.*

At any moment, a primitive can be contained by more than one cell, but we simplify by combining all these certificates into a single event.

Certificate failure We want to know when the primitive enters or leaves a cell, so we rasterize all vertices to the grid and compute the earliest time one passes through a neighboring cell. We break this process down further, by computing the times when individual vertices enter / leave cells.

For each vertex, we look at its velocity in each coordinate direction. If the velocity is zero, then the vertex is not moving and will thus never pass into a neighboring cell; the event time is ∞. If the velocity is positive in the direction of the coordinate axis, then the vertex is moving into the next cell. We can compute the boundary of that cell using the grid's corner and the cell width, from which we can compute the time it passes through the



(a) 2D Box of balls (b) Reef knot (c) Bunny compactor (d) Spike drape (e) Bowline knot

boundary. Similarly, if the velocity is negative, the vertex is moving towards the previous cell. The minimum of all these times is the time the vertex will pass into a neighboring cell.

The minimum of all vertices' times is the scheduled event time. When this event is processed, it may not result in the primitive passing in or out of a cell, but it is conservative and will catch all the instances in which this does occur.

7.4 Results

All broad-phase algorithms are tested and analyzed for quality in a variety of situations. The following set of examples will serve as a core testing suite to verify the efficiency of each method:

(a) **2D Box of balls** This example tests the method's ability to handle large numbers of disparate objects. Contact between the balls comes and goes quickly and should be handled efficiently.

(b) **Reef knot** In this example a reef knot is tied out of ribbons by forces acting on each of the four ends. The knot gets extremely tight, stressing the robustness of the method. As the contacts grow and penalty layers deepen, any advantages provided by optimizations will be made visible.

(c) **Bunny compactor** This example purposely puts ACM in an unphysical situation. As the walls get closer, the penalty layers grow, eventually reaching its practical limits as the walls eventually touch, leaving the simulated bunny in between with nowhere to go.

	Box of Balls	Bunny Crusher	Reef Knot	Bowline	Two Cloth Drape
AABB	96.062	2.180	54.202	120.641	82.107
8-DOP	66.430	-	-	-	-
18-DOP	-	1.825	44.678	86.578	77.689
Sphere	-	-	-	-	-
Grid	-	0.516	-	-	-

Table 7.1: Timings (in hours) for examples executed on a single thread of a 2.33Ghz Intel Xeon with 8GB RAM.

(d) **Cloth drape on spikes** Two long cloth sheets are draped on a bed of spikes, with the bottom one pulled off. This examples tests devious geometry, as each spike is composed of only four extremely narrow triangles.

(e) **Bowline knot** Similar to the reef knot example, this simulation forms a tight knot out of ribbon. However, it has the additional constraint of being tied around a simulated cylinder. This scene tests asynchrony's ability to handle separate, yet competing forces. The force pulling on the end of the ribbon, the internal cylinder forces, the contact forces of the ribbon with the cylinder, and the internal contact forces maintaining the knot all struggle to maintain equilibrium.

Timings Table 7.1 gives timing results. All timings were run in a single thread on a 2.33 Ghz Intel Xeon processor with 8GB RAM.

Timings for sphere trees are not specified, due to the extremely poor performance (on the order of weeks). Two factors contributed to this: First, sphere trees provide very poor bounds for most geometry. This caused the frontier to be much lower in the tree than a tighter fitting volume. As a direct result, the number of events on the queue was over twice that of the other bounding volume choices.

Sphere trees are also slow to fit. We used an implementation of the Miniball sphere fitting algorithm [Gärtner, 1999], which is recognized for its speed. While there exists many other methods for achieving tight-fitting spheres, it seems unlikely that they will have a significant effect on the overall runtime, due to the large number of events.

However, that the polynomials associated to sphere certificate failure times were quadratic rather than linear did not appear to deteriorate performance. This is positive, since it means linear polynomials are not necessary for efficient KDS simulations. Russel [2007] investigates this trade-off for higher-degree problems.

In the remaining bounding volumes, we see a noticeable improvement in using k-DOPs over AABBs. The extra bounds provide a tighter fit and additional culling, reducing queue size by a small, but noticeable amount. The overhead of fitting additional bounding planes is worth the additional cost in computation. For the example in 2D, we used an 8-DOP, while all 3D examples were simulated using 18-DOPs.

Our timing information for the Eulerian grids is incomplete as well. The kinetic data structure performed beautifully for the Bunny Crusher example, vastly outperforming the k-DOPs. However, they did not scale well. We had much difficulty in sizing the grid appropriately for the other examples. If the cells were too large, far too many separation slab events were on the queue from the start. Making the cells small enough to keep the queue size appropriately small required too much overhead to store the grid in memory.

7.5 Discussion

We have seen that our initial choice for bounding volume, the k-DOP, was a particularly good one. The pruning provided by the extra bounding planes is easily worth the additional computation. Furthermore, they lend themselves well to many forms of optimization, as we discover in Chapter 8.

Sphere trees were surprisingly disappointing. Besides being slow to fit, the poor culling behavior resulted in a larger number of events on the queue, along with more refitting and even slower performance.

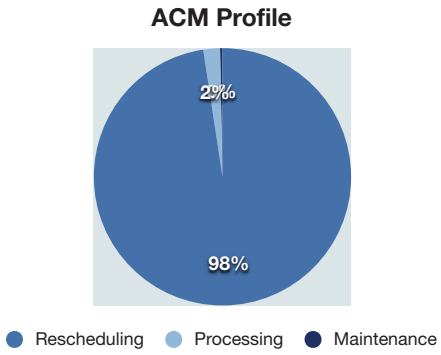
The most criticized problem of Eulerian grids, how to appropriately size them, is magnified in the kinetic setting. We believe the performance offered justifies future research into a kinetic grid. A good starting point is in hierarchical grids, such as octrees, which can give performance comparable to grids without the difficulty in choosing a cell size [Ericson, 2004]. Adaptive grids may also show significant advantages here.

There are still many opportunities for KDS exploration within the ACM framework. In particular, deformable spanners are of interest, even though they currently only apply to point sets [Gao *et al.*, 2006]. We are investigating their extension to represent arbitrary convex shapes. This could be particularly advantageous, since it would eliminate the need for kinetic separation slabs and have a single data structure providing broad-phase and narrow-phase detection.

Chapter 8

Analysis and optimizations

Most runtime in an ACM simulation is spent maintaining KDS certificates. When a force event fires, including penalty forces, some subset of certificate failures are invalidated and require a recomputation of the scheduled event time. This rescheduling time quickly accumulates as heavy contact regions form and penalty force events fire at a rapid rate.



We wish to understand this process in order to improve it. Event rescheduling dominates the simulation runtime. The total cost of rescheduling can be broken down into three variables:

1. **Rescheduling cost**

   Clearly, a slow rescheduling time contributes to a slow simulation. In this area, we wish to investigate why particular events are more expensive to schedule than others, and how we can reduce this cost safely.

2. **Rescheduling frequency**

   Stiff forces require small timesteps. These small timesteps mean more event rescheduling. However, as the timestep decreases, the actual change in trajectories also decreases, yet we are still required to reschedule all affected events. This is highly

wasteful, since most events never actually come to fruition. We examine a few alternatives to reduce this unnecessary rescheduling.

3. **Number of contingent events**

   A large number of contingent events not only means more rescheduling, but more overhead throughout the simulation. Reducing this number may mean slower rescheduling for a fewer number of events, so we explore whether this tradeoff results in an overall faster simulation.

We analyze these three problems and offer optimizations for each. Section 8.4 presents the results of these optimizations.

8.1 Rescheduling cost

There are two types of non-trivially rescheduled events in our simulations: separation list events and separation slab events.

Separation slab events are generally fast. They have very small supports of only four vertices. Their failure time is a linear polynomial. We instead focus on bounding volume events. These are relatively expensive, becoming moreso the higher the BV is in the hierarchy. This is because the support size grows larger for these bounding volumes, requiring many vertices to be visited to refit along each direction.

We also found that these events based higher on the hierarchy are less likely to fire during the course of a simulation. This can be seen as top-level events having more inertia and requiring the overall motion of a large number of vertices to affect change.

We present two optimizations intended to reduce bounding volume event computation time by reducing the fitting of unnecessary bounds.

Short-circuited scheduling In order for two k-DOPs to overlap, all of their bounding planes must overlap. This requires iterating over all $\frac{k}{2}$ directions and finding the time of overlap. Since the overlap time of the bounding volumes is the intersection of each axis' overlap time, the earliest time two BVs can overlap is the latest time any individual axes planes overlap.

Combining this observation with the observation that there will almost always be some force event, usually gravity, that affects all vertices with a steady clock, rescheduling can be short-circuited. If any bounding plane overlap time is encountered that is past when the next gravity even is scheduled, it is assured that this event will have to be revisited at that time (since gravity affects all events), and thus further refinement of the event time is unnecessary. This myopic view of event scheduling is valuable, and is an important consideration in the development of future KDS optimizations.

Bounding plane ordering Not all k-DOP axes are created equal. Depending on the configuration, some excel while others fail in establishing separation. Can one process only the the useful axes, taking the intersection of their bounds, and thus reducing by a constant factor the $O(nd)$ computation of extremal velocities and positions? We achieve this in two steps: first assume that the k axes are already (nearly) sorted from most- to least-useful, and progressively improve the bound by incorporating an additional axis, until an axis fails to improve the bound; in the second step, improve the sorting (for next time) by promoting the axis to the front of the list that provided the most useful bound. For surface meshes, where k-DOPs have high aspect ratios described by a couple of axes, this approach is very effective. This idea can be understood in the language of *coresets* [Agarwal *et al.*, 2005a]; dynamically updating the coreset constituency as the system evolves.

8.2 Rescheduling frequency

Rescheduling frequency is determined by the frequency of force events. AVIs are explicit integrators, which usually necessitate small timesteps to maintain stability. For instance, in the simulation of the reef knot, a timestep of 10^{-5} is required for bending forces. This results in over 600 force events (and resulting reschedules) for every frame. As penalty forces are activated this number grows.

Even so, the average impulse for a non-gravity force event is less than $\frac{1}{10}$ (cm / s). With only a very small number of contingent events (less than $\frac{1}{100000}$) in a time of interest (before the next snapshot), very few of these reschedules are critical to maintain program correctness.

We offer two optimizations which alleviate this in different ways. The first enables events to "ignore" rescheduling when flightplans of its support are altered. The second reduces the number of distinct flightplan adjustments.

Fuzzy trajectories Every velocity update requires the rescheduling of dependent events. This rescheduling tends to be too costly and so frequent that it becomes intractable; these drawbacks are recognized in the KDS literature [Guibas *et al.*, 2001; Guibas *et al.*, 2004]. We introduce the notion of vague trajectories to safely reduce the frequency of rescheduling.

Certificates are rescheduled when a supporting trajectory is altered. Using KDSs specifically in the context of contact mechanics brings into play physical insights that would otherwise not be available. As an illustrative example, consider Newton's apple, which after being tossed into the air follows a parabolic trajectory before hitting the ground. Now split the apple and connect the two halves with a stiff spring. Toss the apple once more, what happens? Since the two halves quickly oscillate against each other, the trajectory of each half has many wiggles—changes in velocity. Even so, the trajectory of the center of mass is exactly parabolic and, ignoring the high-frequency wiggles, the trajectory of each half is "overall" parabolic. Most importantly, unless the half-apple is very close to the floor, the parabola serves as an excellent predictor of the collision time with the floor, while the velocity associated to the rapid oscillations is noisy. This noise is twice detrimental: it impoverishes the collision time estimate, and, worse, it causes frequent rescheduling.

To harness this insight, consider trajectories with bounded uncertainty. In place of precise linear trajectories, imagine "cones" wide enough to encompass the noisy oscillations. On the one hand, this requires computing certificate expiration times that are conservative in the sense that they are valid for any precise trajectory that fits in the cone. On the other hand, the certificate will remain valid, despite noisy changes to the future trajectory, or *flightplan*, so long as the current trajectory remains inside the cone. If the predicted cone is not too thick, and if the actual trajectory remains inside the predicted cone for sufficient time, it could potentially reap a (safe, correct) dramatic reduction in rescheduling.

This thesis pursues a simple implementation motivated by this idea. Recall the scheduling approach for the simple separating slab KDS. After creating a new certificate (say at

time $t = t_0$), a certificate failure time was scheduled by solving for the time at which the particle enters the slab *assuming a constant velocity*. Because of this restrictive assumption, even a small impulse necessitated event rescheduling.

To introduce vagueness, weaken the assumption to allow for a time-varying velocity. Therefore let the velocity of the particle $\dot{X}(t) = \dot{X}(t_0) + \mathbf{u}(t)$, where $\mathbf{u}(t)$ is a time-varying vector of bounded length $\|\mathbf{u}(t)\| \leq \epsilon$. The relaxed assumption has two implications. First, it is now possible for many impulse events to affect the particle without necessitating a certificate rescheduling, so long as each impulse keeps $\|\dot{X}(t) - \dot{X}(t_0)\| \leq \epsilon$. Indeed, for $\epsilon < |\dot{X}(t_0)|$, there is a *cone* of trajectories that avoid rescheduling. Second, the computation of the failure time must be conservative over all future trajectories satisfying the relaxed assumption, *i.e.*, it must compute the earliest possible failure time. For the separating slab, the trajectory producing the earliest failure "worst case" failure is the one maintaining $\|\mathbf{u}(t)\| = \epsilon$ with $\mathbf{u}(t)$ in the direction of the slab.

Increasing ϵ reduces rescheduling frequency, since it widens the cone of covered trajectories; unfortunately, it also increases the frequency of certificate failures, since the worst-case trajectory reaches the slab sooner; these two considerations must be balanced. Fortunately, any choice of ϵ keeps the system safe—the choice of ϵ cannot alter the actual simulated trajectory.

We control the choice of ϵ through a *quality loss* parameter, so-named because it controls the quality, or accuracy, of scheduled times. This parameter varies between 0 and 1, with a value of 0 corresponding to zero quality loss, *i.e.*, the actual scheduled time is used. A value of 1 should not be used, for it corresponds to 100% quality loss and no simulation progress would be made. We smoothly interpolate between the current simulation time $t_{current}$ and the event time t_{event} using this parameter QL, so that the scheduled time is $t_{scheduled} = QL\ t_{current} + (1 - QL)\ t_{event}$.

With this time, we can work backwards to compute the permitted ϵ per vertex. If two objects x_A and x_B with velocities v_A and v_B are approaching one another, the time they will collide is $\frac{x_B - x_A}{v_A - v_B}$. Instead of velocities v_A and v_B we will use velocities with an epsilon

built-in, $v_A + \epsilon$ and $v_B - \epsilon$. We also know the schedule time, so the formula becomes

$$t_{scheduled} = \frac{x_B - x_A}{v_A - v_B + 2\epsilon}.$$

We rearrange and solve for ϵ,

$$\epsilon = \frac{1}{2}\left(\frac{x_B - x_A}{t_{scheduled}} + v_B - v_A\right).$$

When an event is scheduled, we store the velocity v_i and ϵ_i for each vertex in the support. Then, during rescheduling we check each vertex using the current velocity $v_i^{current}$. If $\|v_i^{current} - v_i\| < \epsilon_i$ for all vertices in the support, the event does not need to be rescheduled.

Super-elements AVIs, as presented, have one force event for every force stencil, such as the four vertices of a hinge for a bending force, or a triangle for a constant strain force. Given the stiffness of these forces, the timestep required for stability is automatically chosen based on the size of the stencil. These forces are then integrated asynchronously as presented in §5.2.

As demonstrated by Lew *et al.* [2003], this can be quite computationally efficient given meshes with large dicrepancies in the size of the elements. However, it is just as common to have a regular, evenly-sampled mesh. In this case, all force stencils tend to run at essentially the same size and any computational advantage is lost.

For this reason, the concept of *super-elements* was developed concurrently by Huang *et al.* [2007]. We are mainly interested in asynchrony for its advantages in handling contact, so events running with roughly the same clock are combined and integrated together, with the union of their stencils rescheduled only after all combined events have been processed.

This can be enhanced even further by applying the concept specifically to ACM. Each penalty force in the discrete penalty layers of Chapter 4 has a unique stiffness and thus timestep. These timesteps are unique across layers, but *not* across different colliding stencils, *i.e.*, penalty layer 2 for one collision and penalty layer 2 for another have the same penalty stiffness and timestep. Following the reasoning of super-elements, these penalty forces can be combined into penalty layer super-events, which integrate all penalty forces at that layer and then reschedule the union of their respective stencils.

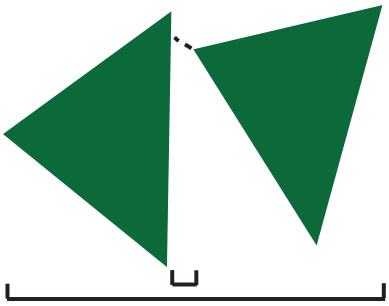


Figure 8.1: Distance between farthest features is much higher than closest features.

This optimization vastly reduces the number of force events that require rescheduling, at the cost of larger stencils. For a cloth simulation, the number of force events is reduced from $O(T+E)$, where T is the number of triangles (for constant strain triangle "stretching" events) and E is the number of edges (for hinge-based bending events) down to a small constant. For a uniform mesh this would be 3 (gravity, grouped stretching, and grouped bending) plus the number of the deepest penalty layer.

8.3 Number of rescheduled events

Fewer contingent events means less rescheduling. However, as contact regions form, the number of such events necessarily increases. We investigate two methods of reducing this cost.

Triangle-triangle slabs The separation slabs in Chapter 5 estimate the time of collision for vertex-triangle and edge-edge pairs. These 15 pairs are created when a broad-phase triangle-triangle separation pair longer exists. Regardless, many of these 15 pairs are still relatively far apart and are in no danger of colliding (Figure 8.1). Thus it seems largely wasteful that all 15 events should be scheduled (and more severely, rescheduled).

As a second layer of defense, we implemented triangle-triangle separation slabs. These events will describe the separation of triangles, squeezing out more performance before the triangles are in collision and lower-level events must be created.

Any separation slab event could be created between convex sets, yet it becomes more expensive as the set grows because the closest point between the sets must be found. This

is trivial for vertex-triangle and edge-edge pairs and only moderately more intensive for triangle pairs, yet becomes prohobitively expensive for larger sets. Hence it seems unlikely that higher-phase slabs would provide any benefit over existing broad-phase methods.

Triangle-triangle penalty forces As touched upon in the preceding section, computational effort grows quickly when triangle-triangle pairs are broken down into their 15 constituent feature pairs.

This cost comes in the form of event scheduling as well as bookkeeping to keep track of which triangle-triangle pair leads to which primitive pairs, since there are multiple triangle pairs that lead to the same vertex-triangle and edge-edge primitives.

Bookkeeping can be messy and bug-prone. Simple code usually leads to fewer bugs and faster runtimes. For this reason, we modify the ACM framework to keep all events at the triangle-triangle level. Instead of creating 15 primitive pair children events when two triangles are within proximity, a single penalty force will be created for the triangle pair, while the triangle-triangle slab will be incremented to check the next penalty layer.

Within the penalty force event we check the 15 primitive pairs, with penalty forces applied where necessary. This may seem to be wasted computation, but the triangle slabs can remember the closest feature sets, as discussed in the previous section. Furthermore, the Lin-Canny algorithm [Lin and Canny, 1991], given a pair of closest features, gives the sequence of features to check to find those that are *next* closest. Following this ordering, distance computations are short-circuited when a feature separation distance greater than the penalty layer proximity is reached.

It is true that the same vertex-triangle and edge-edge pairs will still show up in the penalty force computations. However, this time, the system *knows* that they will show up repetitively. This way, the penalty force for a vertex-triangle collision can be scaled by $\frac{1}{k}$, where k is the valence of the vertex, and is precisely the number of triangle-triangle penalty forces this same vertex-triangle pair will show up inside. The sum of the penalty forces will therefore be the same as if bookkeeping existed that eliminated the pair from the other force events.

	Box of Balls	Bunny Crusher	Reef Knot	Bowline	Two Cloth Drape
Baseline	66.430	1.825	44.678	86.578	104.216
k-DOP Plane Reordering	59.690	1.656	35.235	79.458	71.928
k-DOP Short-circuiting	61.571	1.460	32.129	75.551	66.759
Triangle Separation Slabs	-	1.535	66.991	159.134	78.421
Triangle Penalty Forces	-	1.498	29.631	75.382	69.317
Fuzzy Trajectories	-	0.705	14.947	44.793	41.492

Table 8.1: Timings (in hours) for examples executed on a single thread of a 2.33Ghz Intel Xeon with 8GB RAM.

8.4 Results

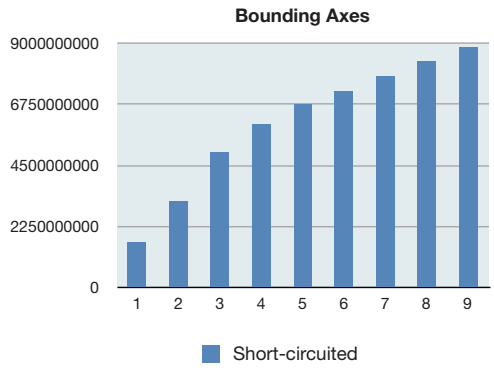
Table 8.1 shows the timing results for all of our optimizations. All timings were collected as a single process on a 2.33 Ghz Intel Xeon processor with 8GB of RAM. We run the same set of experiments as in Chapter 7.

The super-elements optimization is not listed. Without it, simulations were prohobitively slow, and thus we included them as part of the initial implementation.

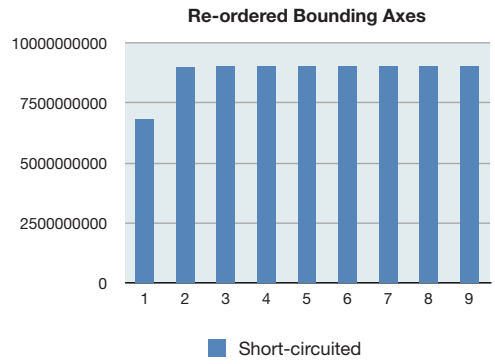
Optimizations were implemented sequentially, so each number is a cummulative effect of all optimizations up to and including that row. Incomplete data is signified by a dash, for instance where optimizations do not apply.

Unsuprisingly, we see a consistent benefit with each successive optimization, as each one reduces total rescheduling costs. The one exception is triangle-based separation slabs, which performed poorly. We believe this happens for two reason. First, the separation slabs may not provide any additional culling, and very quickly descend to its 15 children events. Second, because separation slabs separate in a lazy manner, *i.e.*, time of separation is not explicitly computed, the hierarchy is unlikely to ever move back up to the bounding volume level when primitives eventually separate. This keeps the queue much larger for much longer.

Figure 8.2 shows the additional pruning provided by adaptively reordering k-DOP planes. With this optimization, over 99% of reschedules are short-circuited before the third plane is evaluated.



(a) Each axis provides equal additional pruning.



(b) Shortcircuited detection by reordering axes.

Figure 8.2: By adaptively reordering the axes tested, 99% of rescheduled events can be shortcircuited by the 2nd axis.

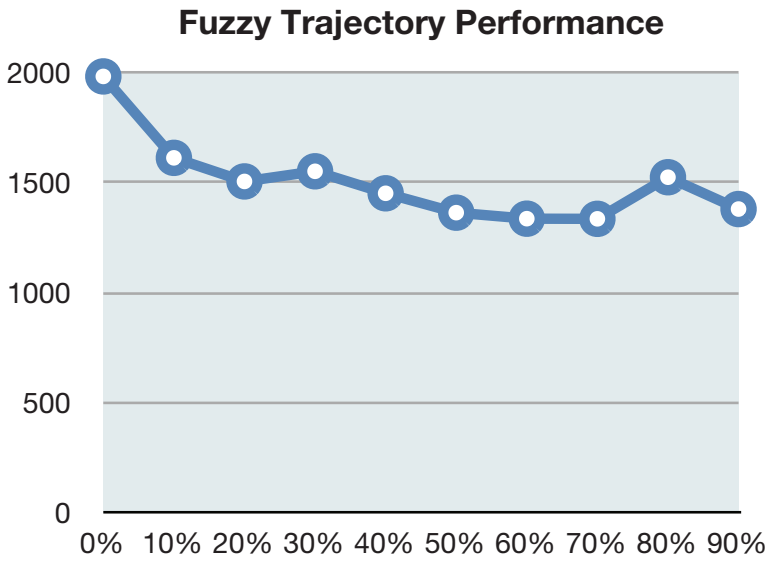


Figure 8.3: Running time varies as epsilon bounds aggresion is varied.

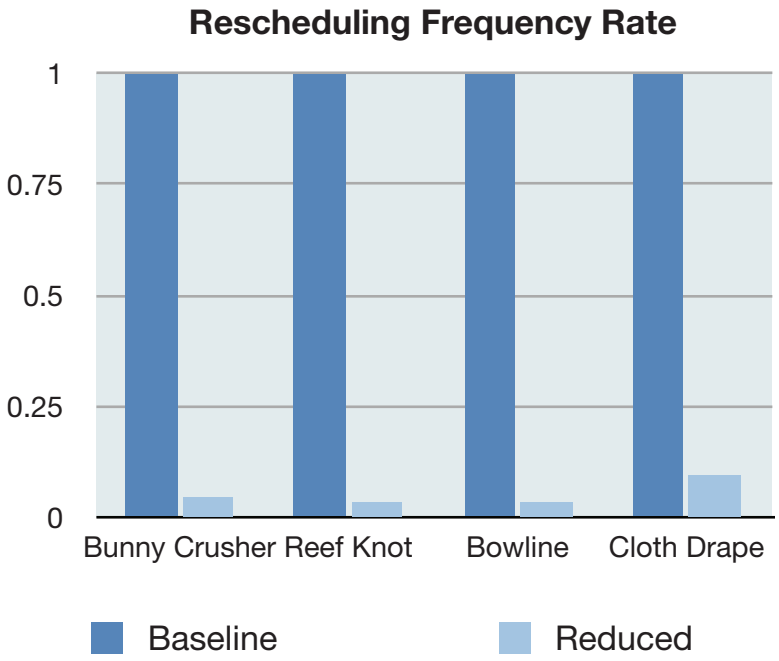
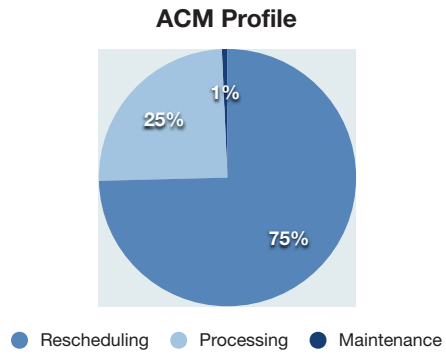


Figure 8.4: Overall rescheduling is reduced to $5 - 12\%$ of baseline amounts.

We tested the Bunny Compactor with a range of quality loss parameters to find the optimal value. Recall that a higher quality loss means less rescheduling but pushes the event closer to the front of the queue, while a low quality loss maintains event times close to the actual time, but with smaller epsilon bounds. For this example we see an optimal value around $QL = 0.7$, as illustrated by Figure 8.3.



We additionally measure the reduction in rescheduling obtained through these optimizations. Figure 8.4 gives a bar graph representing currently rescheduled events as a fraction of the initial implementation, where all events must be rescheduled.

Figure 8.5 plots the number of events during the course of the cloth draping simulation, both before and after optimization. The 15% decrease in total number of events on the queue is reflected by the faster runtimes. Notice that the overall event creation / deletion patterns remain the same.

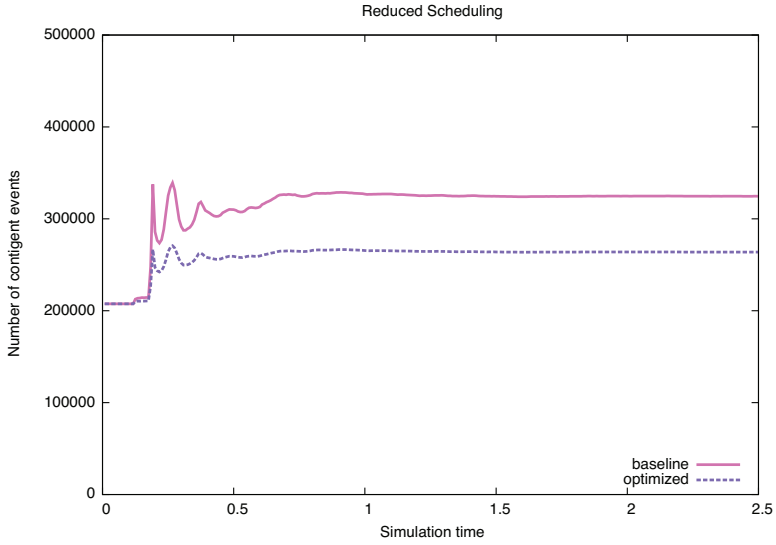


Figure 8.5: Switching to triangle-based events yields a 15% reduction in queued events.

Cummulatively, these optimizations result in rescheduling taking about 75% of total runtime, averaged across all the simulations in our test suite. This is a noticeable improvement, but leaves opportunity for continued research in this area.

8.5 Discussion

We understand that simulation costs are dominated by three factors: rescheduling an event, the number of events that need to be rescheduled, and the frequency rescheduling is required. Reducing any of these factors has a noticeable effect on runtimes.

We offered a set of optimizations based on these three observations, yet many opportunities remain. We still require a more fundamental approach to resolving this rescheduling problem.

The vast majority of events never come to fruition, yet they occupy more than their share of memory and computation time. They must always be present, but only as safeguards in an unlikely event. An elegant solution would be aware of their existance, but not focus unnecessary energy on maintaining them.

One direction is to focus rescheduling effort based on the underlying physics. For instance, each penalty layer acts as a linear half-spring. During a contact there is compression,

yet we know they will never penetration because of our layer construction. Nevertheless, nodes in a shared event separated by a surface are constantly rescheduled, despite our high-level knowledge that they cannot collide. The fuzzy trajectories begin to exploit this idea, but are physics agnostic.

Fuzzy trajectories are currently controlled through an arbitrary "quality loss" parameter. This parameter gives a new scheduled time, and the method computes the allowable change in velocity based on that new time. We have begun initial experiments with reversing this setup: we tell the scheduler how much change in velocity we would like to have, and it schedules with that variability built in. In this way, if no vertex is modified by more than the built-in amount, absolutely no events need be rescheduled. On the other hand, if it does exceed that amount, all events must be rescheduled. This all or nothing approach has shown promising so far.

Advancement in area of scheduling could not only benefit asynchronous simulations, but all applications of kinetic data structures. Therefore, while solutions utilizing the underlying physics or other special knowledge of the problem domain, an ideal solution would be general and indifferent to the nature of the underlying trajectories.

Chapter 9

Conclusion

This thesis covered the advantages and challenges of designing contact models that robustly prevent interpenetrations, accurately portray the physics of real-world contact, and efficiently move forward in simulation time.

The previous state-of-the-art makes a conscious exchange of accuracy for speed in the development of the velocity filter model. Chapter 3 augmented this approach through the novel inelastic projection. In doing so, we examined the viability of regaining physical accuracy in this model. We concluded that to do so is currently an intractable problem, and would require forgoing any of the advantages originally offered by the model.

This conclusion required revisiting the problem from the ground up to design a model with all criteria in mind. This resulted in the discrete penalty layers model (Chapter 4), the first to offer guarantees of safe, penetration-free simulations, physical correctness, and steady progress of the simulator (Chapter 6).

This model requires a new simulation framework to ensure the guarantees remain valid. We offered one such solution that combines the power of asynchronous variational integrators with the foresight of kinetic data structures to efficiently integrate discrete penalty layers (and all other forces in the system) while preserving both geometric (penetration-free) and physical (conservation laws) properties (Chapter 5). Alternative kinetic data structures are investigated in Chapter 7.

We then presented measurements and analysis to extend this asynchronous contact mechanics framework (Chapter 8). This becomes advantageous in guiding optimizations of

the simulator. Overall, we improve the performance of the simulator by a factor of 2, while preserving all of its desired properties.

In the process, we gained numerous insights into how asynchronous simulations work, which will prove valuable in extending the asynchronous simulation framework.

9.1 Future work

This thesis focused on the contact mechanics of asynchronous simulation. There is still much to research in this area, as well as other aspects where asynchronous simulation may demonstrate potential.

1. Considering how much time is spent maintaining kinetic data structures, it is natural to continue to develop alternative data structures that can specifically be applied to asynchronous contact mechanics.

   Maintaining a sorted list is a well-studied KDS problem, so a kinetic sweep and prune style algorithm could be efficient. Here, three sorted lists would be maintained, one for each spatial dimension.

   Deformable spanners, as mentioned in Chapter 7, could make for an useful broad-phase collision algorithm, even eliminating the need for separation slabs. As presented, they work only with point sets, but could be extended to deal with arbitrary convex objects. In our application, triangles would be of particular interest.

   Not all broad-phase algorithms should simply be kineticized versions of existing algorithms. A more difficult, but equally more interesting, problem is to develop a broad-phase KDS specifically to address the known constraints within the asynchronous simulation framework.

2. We use discrete penalty layers to resolve unilateral contact constraints. They could easily be extended to maintain bilateral constraints. This would introduce new challenges in the kinetic data structures. For example, as-is it would double the number of separation slab events, exacerbating the difficulties with scheduling.

3. Augmented Lagrangian formulations have become a popular solution in contact mechanics, combining the benefits of penalty methods with Lagrangian methods. An augmented Lagrangian method that uses AVIs could greatly benefit simulation speed. As the penalty layers grow, their accumulated force contribution could be "handed over" to a Lagrange multiplier in the system, with the penalty layers continuing to keep accelerating impact in check. This could potentially keep simulations in shallower penalty layers, improving overall performance.

   Furthermore, momentum-conserving impulses applied during initial contact may take pressure off of the penalty layers and accelerate contact resolution.

4. Our implementation only simulates cloth and thin shells, but it is simple to extend to include the dynamics of rigid bodies, volumes, and even particle-based fluids. Events on the queue know how to handle themselves, and so would know how to integrate the appropriate material it represents.

   Such an implementation, which includes discrete penalty layers, would allow the seamless coupling of many simulated materials. This kind of coupling is often achieved through a variety of awkward interleavings of integration steps. AVIs provide the timestepping rules to do so in a single simulator, while still maintaining good energy and momentum properties.

5. The bottleneck in simulations continues to be the required rescheduling of kinetic data structure certificates. When examined, however, we see that most of these events never come to fruition. This means all the effort in rescheduling them is essentially wasted, disrupting our claims of true simulation efficiency.

   A smarter rescheduling algorithm would take advantage of the underlying data to intelligently focus effort where it is needed; this is the strategy guiding the rest of the simulator, and so should be extended to this domain.

   How to accomplish this remains a challenging open problem.

6. As with most algorithms, parallelization is an interesting problem. AVIs have been parallelized, but without considering contact [Kedar G. Kale, 2007]. Contact intro-

duces considerable difficulties, since coupling is essentially unpredictable. Rolling back simulations would be expensive, as it requires maintaining states throughout the simulation, although there has been progress in this direction for rigid body simulations [Mirtich, 2000].

Appendix A

Finite penalty layers energy

On a finite precision machine, the infinitely sequenced penalty layers are eventually truncated; any deeper layer would have a stable timestep that is computationally zero. Adding this timestep to the current simulation time returns the same time and hence time will not progress.

Our guarantee of progress assumed that it would take an infinite amount of kinetic energy to overpower an infinite number of penalty layers. We In practice only a finite number of layers is representable, thus a finite amount of kinetic energy is able to power through them. Therefore it is useful to investigate the practical bounds on kinetic energy for a given construction.

The following is Mathematica source code, which given a defined penalty layer distribution function, a stiffness growth function, and a machine's floating point epsilon (the smallest number representable by that machine) returns the amount of energy required to halt the simulation.

```
(* r defines layer stiffness growth *)
r[l_] := 1000*l^3;

(* eta gives the distribution of the penalty layers *)
eta[l_] := 0.1*l^(-(1/4));
```

```
(* Floating point epsilon, as defined by your architecture *)
FloatEps = 1.11*10^-16;

(* MaxLayer is the deepest layer allowed, based on the timestep *)
MaxLayer = Power[1/(1000*(10/(2 \[Pi])*FloatEps)^2), (9)^-1];

V = 0.0;
(* Accumulate potential contribution from each layer *)
For[l = 1, l <= MaxLayer, l = l + 1,
    (* The linear half-spring cannot be compressed past MaxLayer,
   so subtract off that distance *)
    V = V + 1/2 r[l] (eta[l] - eta[MaxLayer])^2;
   ];

Print[V]
```

Bibliography

[Agarwal *et al.*, 1997a] P. K. Agarwal, L. J. Guibas, J. Hershberger, and E. Veach. Maintaining the extent of a moving set of points. In *5-th workshop on algorithms & data structures (WADS)*, pages 31–44, 1997.

[Agarwal *et al.*, 1997b] P. K. Agarwal, L. J. Guibas, T. Murali, and J. Vitter. Cylindrical static and kinetic binary space partitions. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 39–48, 1997.

[Agarwal *et al.*, 1998a] P. K. Agarwal, J. Erickson, and L. J. Guibas. Kinetic binary space partitions for intersecting segments and disjoint triangles. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, pages 169–178, 1998.

[Agarwal *et al.*, 1998b] Pankaj K. Agarwal, D. Eppstein, L. J. Guibas, and M. Henzinger. Parametric and kinetic minimum spanning trees. In *Proc. 39th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 596–605, 1998.

[Agarwal *et al.*, 2002a] P. Agarwal, J. Basch, L. J. Guibas, J. Hershberger, and L. Zhang. Deformable free space tilings for kinetic collision detection. *Intl. J. Robotics Research*, 21:179–197, 2002.

[Agarwal *et al.*, 2002b] P. K. Agarwal, J. Gao, and L. J. Guibas. Kinetic medians and *kd*-trees. In *Proc. of the 10th Annual European Symposium on Algorithms (ESA'02)*, pages 5–16, 2002.

[Agarwal *et al.*, 2004] Pankaj Agarwal, Leonidas Guibas, An Nguyen, Daniel Russel, and Li Zhang. Collision detection for deforming necklaces. *Computational Geometry: Theory and Applications*, 28:137–163, 2004.

[Agarwal *et al.*, 2005a] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. In J. E. Goodman, J. Pach, and E. Welzl, editors, *Combinatorial and Computational Geometry*, pages 1–30. Cambridge University Press, New York, 2005.

[Agarwal *et al.*, 2005b] Pankaj K. Agarwal, Mark de Berg, Jie Gao, Leonidas J. Guibas, and Sariel Har-Peled. Staying in the middle: Exact and approximate medians in r^1 and r^2 for moving points. In *Proc. of the 17th Canadian Conference on Computational Geometry (CCCG05)*, pages 42–45, August 2005.

[Andersen, 1983] H.C. Andersen. Rattle: a velocity version of the shake algorithm for molecular dynamics calculations. *Journal of Computational Physics*, 52(1):24–34, 1983.

[Auslender, 1999] A. Auslender. Penalty and barrier methods: A unified framework. *SIAM J. on Optimization*, 10(1):211–230, 1999.

[Baraff and Witkin, 1998] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *SIGGRAPH '98*, pages 43–54, New York, NY, USA, 1998.

[Baraff *et al.*, 2003] David Baraff, Andrew Witkin, and Michael Kass. Untangling cloth. *ACM Trans. Graph.*, 22(3):862–870, 2003.

[Baraff, 1989] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 223–232, New York, NY, USA, 1989. ACM.

[Baraff, 1994] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH '94*, pages 23–34, 1994.

[Barth *et al.*, 1995] E. Barth, K. Kuczera, B. Leimkuhler, and R.D. Skeel. Algorithms for constrained molecular dynamics. *Journal of Computational Chemistry*, 16(10):1192–1209, 1995.

[Barzel and Barr, 1988] Ronen Barzel and Alan H. Barr. A modeling system based on dynamic constraints. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 179–188, New York, NY, USA, 1988. ACM.

[Basch *et al.*, 1997] J. Basch, L. J. Guibas, and L. Zhang. Proximity problems on moving points. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 344–351, 1997.

[Basch *et al.*, 1999] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. *Journal of Algorithms*, 31:1–28, 1999.

[Bathe and Chaudhary, 1985] K.J. Bathe and A. Chaudhary. A solution method for planar and axisymmetric contact problems. *International Journal for Numerical Methods in Engineering*, 21(1):65–88, 1985.

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

[Bell *et al.*, 2005] N. Bell, Y. Yu, and P.J. Mucha. Particle-based simulation of granular materials. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, page 86. ACM, 2005.

[Boyd and Vandenberghe, 2004] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[Bridson *et al.*, 2002] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *SIGGRAPH '02*, pages 594–603, 2002.

[Bridson *et al.*, 2003] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *SCA '03*, pages 28–36, 2003.

[Brilliantov and Pöschel, 2004] Nikolai V. Brilliantov and Thorsten Pöschel. *Kinetic Theory of Granular Gases*. Oxford University Press, USA, 2004.

[Chaudhary and Bathe, 1986] AB Chaudhary and K.J. Bathe. A solution method for static and dynamic analysis of three-dimensional contact problems with friction. *Comp. Struct.*, 24(6):855–873, 1986.

[Choi and Ko, 2005] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 1, New York, NY, USA, 2005. ACM.

[Cirak and West, 2005] F. Cirak and M. West. Decomposition-based contact response (DCR) for explicit finite element dynamics. *Int'l Journal for Numerical Methods in Engineering*, 64(8):1078–1110, 2005.

[Cline and Pai, 2003] M. Cline and D. Pai. Post-stabilization for rigid body simulation with contact and constraints. In *Proc. 2003 IEEE Int'l Conf. Robotics and Automation (ICRA 2003)*, pages 3744–3751. IEEE, 2003.

[Cohen *et al.*, 1995] J.D. Cohen, M.C. Lin, D. Manocha, and M. Ponamgi. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In *Proceedings of the 1995 symposium on Interactive 3D graphics*. ACM, 1995.

[Cormen *et al.*, 2001] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd revised edition edition, September 2001.

[Cottle *et al.*, 1993] R.W. Cottle, J.S. Pang, and R.B. Stone. *The Linear Complementarity Problem*. Academic Press, New York, NY, 1993.

[Daniel, 1997] WJT Daniel. The subcycled Newmark algorithm. *Computational Mechanics*, 20(3):272–281, 1997.

[Daniel, 1998] W.J.T. Daniel. A study of the stability of subcycling algorithms in structural dynamics. *Computer methods in applied mechanics and engineering*, 156(1-4):1–13, 1998.

[Egan *et al.*, 2003] Kevin Egan, Stephen Berard, and J.C. Trinkle. Modeling nonconvex constraints using linear complementarity. Technical report, Rensselaer Polytechnic Institute, 2003.

[Endo *et al.*, 1984] T. Endo, JT Oden, EB Becker, and T. Miller. A numerical analysis of contact and limit-point behavior in a class of problems of finite elastic deformation. *Computers & structures*, 18(5):899–910, 1984.

[Erickson *et al.*, 1999] J. Erickson, L. J. Guibas, J. Stolfi, and L. Zhang. Separation-sensitive collision detection for convex objects. In *Proc. 10th ACM-SIAM Symp. Discrete Algorithms*, pages 102–111, 1999.

[Ericson, 2004] Christer Ericson. *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann, December 2004.

[Faure *et al.*, 2008] François Faure, Sébastien Barbier, Jérémie Allard, and Florent Falipou. Image-based collision detection and response between arbitrary volumetric objects. In *ACM Siggraph/Eurographics Symposium on Computer Animation, SCA 2008, July, 2008*, Dublin, Irlande, July 2008.

[Fletcher, 1987] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons Ltd., Hoboken, NJ, USA, 1987.

[Fowles and Cassiday, 1962] G.R. Fowles and G.L. Cassiday. *Analytical mechanics*. Holt, Rinehart and Winston New York, 1962.

[Gao *et al.*, 2003] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. *Discrete and Computational Geometry*, 30(1):45–65, 2003.

[Gao *et al.*, 2005] J. Gao, L. J. Guibas, and A. Nguyen. Distributed proximity maintenance in ad hoc mobile network. In *IEEE International Conference on Distributed Computing in Sensor System (DCOSS'05)*, pages 4–19, June 2005.

[Gao *et al.*, 2006] Jie Gao, Leonidas Guibas, and An Nguyen. Deformable spanners and their applications. *Computational Geometry: Theory and Applications*, 35(1-2):2–19, 2006.

[Gärtner, 1999] B. Gärtner. Fast and robust smallest enclosing balls. *Lecture notes in computer science*, pages 325–338, 1999.

[Goldenthal *et al.*, 2007] Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. Efficient Simulation of Inextensible Cloth. *SIGGRAPH (ACM Transactions on Graphics)*, 26(3), 2007.

[Goldstein *et al.*, 2002] Herbert Goldstein, Charles P. Poole, and John Safko. Classical mechanics, 2002.

[Grinspun *et al.*, 2003] Eitan Grinspun, Anil Hirani, Mathieu Desbrun, and Peter Schröder. Discrete Shells. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 62–67, Aug 2003.

[Guendelman *et al.*, 2003] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex rigid bodies with stacking. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 871–878, New York, NY, USA, 2003. ACM.

[Guibas *et al.*, 2000] L. J. Guibas, J. Hershberger, S. Suri, and L. Zhang. Kinetic connectivity of unit disks. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 331–340, 2000.

[Guibas *et al.*, 2001] Leonidas J. Guibas, Feng Xie, and Li Zhang. Kinetic collision detection: Algorithms and experiments. In *ICRA*, pages 2903–2910, 2001.

[Guibas *et al.*, 2004] Leonidas Guibas, Menelaos Karaveles, and Daniel Russel. A computational framework for handling motion. In *Proceedings of teh Sixth Workshop on Algorithm Engineering and Experiments*, pages 129–141, 2004.

[Guibas, 1998] L. J. Guibas. Kinetic data structures — a state of the art report. In *Proc. 3rd Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 191–209, 1998.

[Guibas, 2004a] L. Guibas. Kinetic data structures. In D. Mehta and S. Sahni, editors, *Handbook of Data Structures and Applications*, pages 23–1–23–18. Chapman and Hall/CRC, 2004.

[Guibas, 2004b] L. Guibas. Modeling motion. In J. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 1117–1134. Chapman and Hall/CRC, 2nd edition, 2004.

[Hadley, 1964] G. Hadley. *Nonlinear and dynamic programming.* Addison-Wesley Reading, MA, 1964.

[Hahn, 1988] James K. Hahn. Realistic animation of rigid bodies. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 299–308, New York, NY, USA, 1988. ACM.

[Hairer *et al.*, 2002] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-preserving Algorithms for Ordinary Differential Equations.* Springer, 2002.

[Hauth *et al.*, 2003] Michael Hauth, Olaf Etzmuß, and Wolfgang Straßer. Analysis of numerical methods for the simulation of deformable models. *The Visual Computer*, 19(7-8):581–600, 2003.

[Hillier *et al.*, 2004] Frederick S. Hillier, Gerald J. Lieberman, Frederick Hillier, and Gerald Lieberman. *Introduction to Operations Research.* McGraw-Hill Science/Engineering/Math, July 2004.

[Hong *et al.*, 2005] Min Hong, Min-Hyung Choi, Sunhwa Jung, Samuel Welch, and John Trapp. Effective constrained dynamic simulation using implicit constraint enforcement. In *International Conference on Robotics and Automation*, pages 4520–4525, April 2005.

[Huang *et al.*, 2007] J Huang, X Jiao, R Fujimoto, and H Zha. Dag-guided parallel asynchronous variational integrators with super-elements. *Proceedings of the 2007 summer computer simulation ...*, Jan 2007.

[Hubbard, 1996] P.M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics (TOG)*, 15(3):210, 1996.

[Huh *et al.*, 2001] Suejung Huh, Dimitris Metaxas, and Norman Badler. Collision resolutions in cloth simulation. In *Proceedings of Computer Animation*, pages 122–127, 2001.

[Hunt and Crossley, 1975] KH Hunt and FRE Crossley. Coefficient of restitution interpreted as damping in vibroimpact. *ASME Journal of Applied Mechanics*, 42(2):440–445, 1975.

[Kane *et al.*, 2000] C. Kane, J. E. Marsden, M. Ortiz, , and M. West. Variational integrators and the newmark algorithm for conservative and dissipative mechanical systems. *Int. J. Num. Math. Eng.*, 49:1295–1325, 2000.

[Karavelas and Guibas, 2001] Menelaos Karavelas and Leonidas Guibas. Static and kinetic geometric spanners with applications. In *Proceedings 12-th ACM-SIAM Symposium on Discrete Algorithms*, pages 168–176, 2001.

[Kaufman *et al.*, 2005] Danny M. Kaufman, Timothy Edmunds, and Dinesh K. Pai. Fast frictional dynamics for rigid bodies. In *SIGGRAPH '05*, pages 946–956, 2005.

[Kaufman *et al.*, 2008] Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. Staggered projections for frictional contact in multibody systems. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, pages 1–11, New York, NY, USA, 2008. ACM.

[Kedar G. Kale, 2007] Adrian J. Lew Kedar G. Kale. Parallel asynchronous variational integrators. *International Journal for Numerical Methods in Engineering*, 70(3):291–321, 2007.

[Kharevych *et al.*, 2006] L. Kharevych, Weiwei Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun. Geometric, variational integrators for computer animation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 43–51, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[Klosowski *et al.*, 1998] J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[Lanczos, 1986] Cornelius Lanczos. *The Variational Principles of Mechanics*. Dover, 4th edition, 1986.

[Larsson and Akenine-Möller, 2001] T. Larsson and T. Akenine-Möller. Collision detection for continuously deforming bodies. *Eurographics 2001*, pages 325–333, 2001.

[Lee and Herrmann, 1993] J. Lee and HJ Herrmann. Angle of repose and angle of marginal stability: molecular dynamics of granular particles. *Journal of Physics A: Mathematical and General*, 26:373–383, 1993.

[Lew *et al.*, 2003] Adrian Lew, Jerrold E. Marsden, Michael Ortiz, and Matthew West. Asynchronous variational integrators. *Archive for Rational Mechanics And Analysis*, 167:85–146, 2003.

[Li and Chen, 1998] Tsai-Yen Li and Jin-Shin Chen. Incremental 3d collision detection with hierarchical data structures. In *VRST '98: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 139–144, New York, NY, USA, 1998. ACM.

[Lin and Canny, 1991] Ming Lin and John Canny. A fast algorithm for incremental distance calculations. *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 1008–1014, April 1991.

[Lin and Huang, 2002] Shih-Tin Lin and Jiann-Nan Huang. Stabilization of baumgarte's method using the runge-kutta approach. *Journal of Mechanical Design*, 124(4):633–641, 2002.

[Marhefka and Orin, 1996] DW Marhefka and DE Orin. Simulation of contact using a nonlinear damping model. In *1996 IEEE International Conference on Robotics and Automation, 1996. Proceedings.*, volume 2, 1996.

[Marsden and West, 2001] J. E. Marsden and M. West. Discrete mechanics and variational integrators. *Acta Numerica*, 10:357–514, 2001.

[Marsden *et al.*, 1998] Jerrold Marsden, George Patrick, and Steve Shkoller. Multisymplectic Geometry, Variational Integrators, and Nonlinear PDEs. *Communications in Mathematical Physics*, 199(2):351–395, 1998.

[Marsden *et al.*, 2001] J. Marsden, S. Pekarsky, S. Shkoller, and M. West. Variational methods, multisymplectic geometry and contiunuum mechanics. *Journal of Geometry and Physics*, 38(3–4):253–284, June 2001.

[McAdams *et al.*, 2009] Aleka McAdams, Andrew Selle, Kelly Ward, Eftychios Sifakis, and Joseph Teran. Detail preserving continuum simulation of straight hair. *ACM Trans. Graph.*, 28(3):1–6, 2009.

[Milenkovic and Schmidl, 2001] Victor J. Milenkovic and Harald Schmidl. Optimization-based animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 37–46, New York, NY, USA, 2001. ACM.

[Mirtich and Canny, 1995a] Brian Mirtich and John Canny. Impulse-based dynamic simulation. In *WAFR: Proceedings of the workshop on Algorithmic foundations of robotics*, pages 407–418, Natick, MA, USA, 1995. A. K. Peters, Ltd.

[Mirtich and Canny, 1995b] Brian Mirtich and John Canny. Impulse-based simulation of rigid bodies. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 181–ff., New York, NY, USA, 1995. ACM.

[Mirtich, 2000] Brian Mirtich. Timewarp rigid body simulation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 193–200, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[Moreau, 1988] J.J. Moreau. *Non Smooth Mechanics and Applications*, volume 302, pages 1–82. Springer, 1988.

[Neal and Belytschko, 1989] MO Neal and T. Belytschko. Explicit-explicit subcycling with non-integer time step ratios for structural dynamic systems. *COMP. STRUCT.*, 31(6):871–880, 1989.

[Palmer and Grimsdale, 1995] IJ Palmer and RL Grimsdale. Collision detection for animation using sphere-trees. In *Computer Graphics Forum*, volume 14, pages 105–116. Blackwell, 1995.

[Pandolfi *et al.*, 2002] A. Pandolfi, C. Kane, JE Marsden, and M. Ortiz. Time-discretized variational formulation of non-smooth frictional contact. *Int. J. Numer. Meth. Eng*, 53(8):1801–1829, 2002.

[Papadopoulos and Taylor, 1993] P. Papadopoulos and RL Taylor. A simple algorithm for three-dimensional finite element analysis of contact problems. *Computers & Structures*, 46(6):1107–1118, 1993.

[Pauly *et al.*, 2004] Mark Pauly, Dinesh K. Pai, and Leonidas J. Guibas. Quasi-rigid objects in contact. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 109–119, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[Peric and Owen, 1992] D. Peric and DRJ Owen. Computational model for 3-D contact problems with friction based on the penalty method. *International Journal for Numerical Methods in Engineering*, 35(6):1289–1309, 1992.

[Pires and Oden, 1983] EB Pires and JT Oden. Analysis of contact problems with friction under oscillating loads. *Computer Methods in Applied Mechanics and Engineering*, 39:337–362, 1983.

[Pöschel and Schwager, 2005] Thorsten Pöschel and T. Schwager. *Computational Granular Dynamics: Models and Algorithms*. Springer, 2005.

[Provot, 1997] Xavier Provot. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation '97*, pages 177–189, Wien, 1997. Springer Verlag.

[Rabier *et al.*, 1986] P. Rabier, JAC Martins, JT Oden, and L. Campos. Existence and local uniqueness of solutions to contact problems in elasticity with nonlinear friction laws. *International Journal of Engineering Science*, 24(11):1755–1768, 1986.

[Russel, 2007] D. Russel. *Kinetic data structures in practice*. PhD thesis, Citeseer, 2007.

[Ryckaert *et al.*, 1977] J.P. Ryckaert, G. Ciccotti, and H.J.C. Berendsen. Numerical integration of the Cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *Journal of Computational Physics*, 23(3):327–341, 1977.

[Schwager and Pöschel, 2007] Thomas Schwager and Thorsten Pöschel. Coefficient of restitution and linear dashpot model revisited. *Granular Matter*, 9(6):465–469, November 2007.

[Selle *et al.*, 2008] A. Selle, M. Lentine, and R. Fedkiw. A mass spring model for hair simulation. *ACM Transactions on Graphics-TOG*, 27(3):64–64, 2008.

[Sifakis *et al.*, 2008] E. Sifakis, S. Marino, and J. Teran. Globally coupled impulse-based collision handling for cloth simulation. In *SCA '08*, 2008.

[Smolinski and Wu, 1998] P. Smolinski and Y.S. Wu. An implicit multi-time step integration method for structural dynamics problems. *Computational Mechanics*, 22(4):337–343, 1998.

[Smolinski *et al.*, 1996] P. Smolinski, S. Sleith, and T. Belytschko. Stability of an explicit multi-time step integration algorithm for linear structural dynamics equations. *Computational Mechanics*, 18(3):236–244, 1996.

[Snyder *et al.*, 1993] John M. Snyder, Adam R. Woodbury, Kurt Fleischer, Bena Currin, and Alan H. Barr. Interval methods for multi-point collisions between time-dependent curved surfaces. In *SIGGRAPH '93*, pages 321–334, 1993.

[Stein and Wriggers, 1982] E. Stein and P. Wriggers. Calculation of impact contact problems of thin elastic shells taking into account geometrical nonlinearities within the contact region. *Computer Methods in Applied Mechanics and Engineering*, 34(1-3):861–880, 1982.

[Stewart and Trinkle, 1996] D. Stewart and J.C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *Intl. Journal for Numerical Methods in Engineering*, 39:2673–2691, 1996.

[Stewart, 1999] G. W. Stewart. The QLP approximation to the singular value decomposition. *SIAM J. Sci. Comput.*, 20(4):1336–1348, 1999.

[Suris, 1990] Y.B. Suris. Hamiltonian methods of Runge–Kutta type and their variational interpretation. *Mat. Model.*, 2:78–87, 1990.

[Taylor and Papadopoulos, 1993] R.L. Taylor and P. Papadopoulos. On a finite element method for dynamic contact/impact problems. *International Journal for Numerical Methods in Engineering*, 36:2123–2123, 1993.

[Teran *et al.*, 2005] Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. Robust quasistatic finite elements and flesh simulation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 181–190, New York, NY, USA, 2005. ACM.

[Terzopoulos *et al.*, 1987] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, New York, NY, USA, 1987. ACM.

[Van Den Bergen, 2005] G. Van Den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Graphics Tools: The Jgt Editors' Choice*, page 131, 2005.

[Volino and Magnenat-Thalmann, 2006] Pascal Volino and Nadia Magnenat-Thalmann. Resolving surface collisions through intersection contour minimization. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1154–1159, New York, NY, USA, 2006. ACM.

[Weller and Zachmann, 2006] René Weller and Gabriel Zachmann. Kinetic separation lists for continuous collision detection of deformable objects. In *Third Workshop in Virtual Reality Interactions and Physical Simulation (Vriphys)*, Madrid, Spain, 6–7 November 2006.

[Wendlandt and Marsden, 1997] J. Wendlandt and J.E. Marsden. Mechanical integrators derived from a discrete variational principle. *Physica D*, 106:223–246, 1997.

[West, 2003] Matthew West. *Variational Integrators*. PhD thesis, California Institute of Technology, 2003.

[Witkin and Baraff, 2001] Andrew Witkin and David Baraff. Physically based modeling: Course notes. In *SIGGRAPH '01: ACM SIGGRAPH 2001 Courses*, New York, NY, USA, 2001. ACM.

[Witkin *et al.*, 1990] Andrew Witkin, Michael Gleicher, and William Welch. Interactive dynamics. *SIGGRAPH Comput. Graph.*, 24(2):11–21, 1990.

[Wriggers and Imhof, 1993] P. Wriggers and M. Imhof. On the treatment of nonlinear unilateral contact problems. *Archive of Applied Mechanics (Ingenieur Archiv)*, 63(2):116–129, 1993.

[Wriggers and Laursen, 2007] P Wriggers and Tod A Laursen. *Computational contact mechanics*, volume no. 498 of *CISM courses and lectures*. Springer, Wien, 2007.

[Wriggers and Nettingsmeier, 2008] P. Wriggers and J. Nettingsmeier. Homogenization and multi-scale approaches for contact problems. In Peter Wriggers and Tod A. Laursen, editors, *Computational Contact Mechanics*, volume 498 of *CISM International Centre for Mechanical Sciences*. Springer Vienna, 2008.

[Wriggers and Panagiotopoulos, 1999] Peter Wriggers and Panagiotis Panagiotopoulos, editors. *New Developments in Contact Problems*, chapter 1, pages 1–54. SpringerWienNewYork, 1999.

[Wriggers and Scherf, 1995] P. Wriggers and O. Scherf. An adaptive finite element algorithm for contact problems in plasticity. *Computational Mechanics*, 17(1):88–97, 1995.

[Wriggers and Van, 1990] P. Wriggers and V. Van. Finite element formulation of large deformation impact-contact problems with friction. *Comp. Struct.*, 37(3):319–331, 1990.

[Wriggers and Zavarise, 1997] P. Wriggers and G. Zavarise. On contact between three-dimensional beams undergoing large deflections. *Communications in numerical methods in engineering*, 13(6):429–438, 1997.

[Wriggers, 1995] P. Wriggers. Finite element algorithms for contact problems. *Archives of Computational Methods in Engineering*, 2(4):1–49, 1995.

[Zachmann and Weller, 2006] Gabriel Zachmann and Rene Weller. Kinetic bounding volume hierarchies for deformable objects. In *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pages 189–196, New York, NY, USA, 2006. ACM.

[Zachmann, 1998] G. Zachmann. Rapid collision detection by dynamically aligned DOP-trees. In *Proc. of IEEE Virtual Reality Annual International Symposium; VRAIS98*, pages 90–97, 1998.