Asynchronous Contact Mechanics

David Harmon
Columbia University

Etienne Vouga
Columbia University

Breannan Smith
Columbia University

Rasmus Tamstorf
Walt Disney Animation Studios

Eitan Grinspun
Columbia University

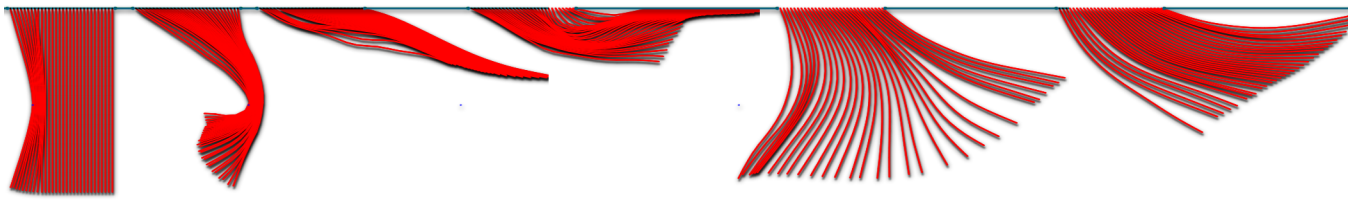


Figure 1. A prescribed particle slowly moves through a set of curtains, then impulsively shifts to a very high velocity. The slow and fast phases highlight the method's ability to handle smooth resting and sliding with deep stacking, and arbitrarily fast penetration-free movements in which collisions are treated when (as opposed to well before or after) they occur. The curtains continue to swing for a long time, even as controlled internal dissipation damps high frequencies.

Abstract

We develop a method for reliable simulation of elastica in complex contact scenarios. Our focus is on firmly establishing three parameter-independent guarantees: that simulations of well-posed problems (a) have no interpenetrations, (b) obey causality, momentum- and energy-conservation laws, and (c) complete in finite time. We achieve these guarantees through a novel synthesis of asynchronous variational integrators, kinetic data structures, and a discretization of the contact barrier potential by an infinite sum of nested quadratic potentials. In a series of two- and three-dimensional examples, we illustrate that this method more easily handles challenging problems involving complex contact geometries, sharp features, and sliding during extremely tight contact.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: variational, symplectic, contact, collision, simulation

1 Motivation

Even as computer hardware benefits from Moore's Law, our ability to program, debug, and maintain software advances at a humbler pace. This observation shapes our priorities as we develop physical simulation tools for computer graphics. While making choices that yield up-front simplicity and blazing performance is important today, we prefer that these choices do not obstruct our long-term goals of extending functionality and improving realism. Laying aside ad-hoc models in favor of physical approaches might require a deeper initial investment, but it promises to pay off handsomely in predictability, controllability, and extensibility. From this vantage point, we propose to revisit the long-studied problem of simulating deformable objects in complex contact scenarios.

Safety, correctness, progress Robust simulation of complex contact scenarios is critical to applications spanning graphics (training, virtual worlds, entertainment) and engineering (product design, safety analysis, experimental validation). Challenging scenarios involve dynamics with frequent and distributed points of contact, interaction with sharp boundaries, resting and sliding contact, and combinations thereof. Useful resolution of these scenarios requires consideration of the fundamental issues of geometric *safety*, physical *correctness*, and computational *progress*, with the respective meanings that (a) for well-posed problems the simulation does not enter an invalid (interpenetrating) state, (b) collision response obeys physical laws of causality and conservation (of mass, momentum, energy, *etc.*), and (c) the algorithm completes a simulation in finite, preferrably short, time.

An ideal algorithm offers provable guarantees of safety, correctness, and progress. A safety guarantee eliminates the need to iterate through the animation-design process because of unsightly penetration artifacts; such a guarantee should not fall on an overburdened user lapped in tunable parameters. A correctness guarantee is a prerequisite for physical behavior that is consistent under rediscretization of space and time. Respect for causality is critical to capturing chain reactions and phenomena such as waves and stacking; discrete conservation laws allow for the development of tunable dissipation that does not "cross-talk" with parasitic numerical damping. If, however, these two guarantees are not accompanied by guaranteed progress, the simulation may never complete, no matter how fast or parallel the hardware.

Shortcomings of synchrony Most time integration methods are synchronous, moving the entire configuration forward in lock-step from one instant in time to the next. Such synchrony is fundamentally at odds with safety, correctness, and progress: the first two goals are assured by attending to collisions in order of causality, which can require arbitrarily small times steps. The number of possible impact events in a single "reasonable" time step can be enormous: in their analysis of contact, Cirak and West [2005] present a counting argument and conclude that synchronous "contact simulation algorithms cannot attempt to exactly compute the sequence and timing of all impacts," as this would preclude reasonable progress.

The graphics community's prevailing emphasis on *progress* has motivated many efforts to find, *retroactively*, a physically plausible collision response to a set of collisions that occurred over a preceding time interval [Provot 1997; Bridson et al. 2002]. Such methods typically have adjustable parameters that must be carefully chosen

to balance safety and progress; other methods discard causality in favor of progress [Milenkovic and Schmidl 2001].

The principled, faithful simulation of complex collisions for deformable objects remains an open, challenging, and important problem.

Asynchrony We propose to place safety and correctness on an equal footing with progress. To overcome the fundamental opposition between these requirements, we turn to *asynchronous integration*, which integrates each geometric element at its own pace, *not* in lockstep with the entire object. Asynchrony offers compelling long-term advantages for simulations of deformable objects in complex contact—advantages that remain unexplored, in particular in terms of safety, correctness, and progress.

For scenarios involving sharp boundaries or dispersed points of contact, asynchrony renders non-interpenetration and momentum conservation tractable. Because elements advance at their own pace, those not entangled in collisions can proceed at large time steps. As shown in Figure 2, the median time step of an asynchronous method can be moderate even when tight collisions force some elements to proceed at small time steps.

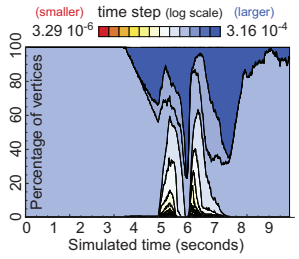


Figure 2. Asynchrony in the curtain simulation, depicted by the time-evolving distribution of vertex time step sizes, enables adaptive allocation of computational resources in spacetime.

Asynchronous integration As a point of departure we consider *asynchronous variational integrators* (AVIs) [Lew et al. 2003], which belong to a larger class of integrators that exactly conserve both momentum and symplecticity (loosely related to areas in phase space). The well-known Verlet ("leapfrog") integrator is symplectic; such integrators are highly regarded because of their provable approximate conservation of energy over long spans of simulated time. AVIs were previously demonstrated to enjoy these correctness properties while simultaneously allowing for efficient treatment of spatially non-uniform discretizations; however, a correct contact model remains unexplored.

Asynchronous collision detection To ensure safety, we require an equally principled approach to collision detection. This is a heavily studied problem; alas, the many reported successes are specific to the synchronous context, and as a group current methods can be intractably slow if naïvely applied after each local asynchronous step. This motivates our interest in *kinetic data structures* (KDSs) [Basch et al. 1999]: a KDS algorithm maintains a data structure governed by formal invariants describing some discrete attribute (such as absence of collisions), in response to the continuous movement of geometric elements. Many existing collision detection methods can be reformulated from a KDS perspective. KDSs seem destined for asynchronous applications, because their focus on fast, minimal, "output-sensitive" data-structure updates makes them ideally suited for the small, local changes effected by each AVI step. And yet, while KDSs are the perfect suitor for AVIs with their safety complementing AVIs' correctness, no such matrimony has yet been considered.

Contributions These observations motivate our interest in approaching contact mechanics for both graphics and mechanics applications from a new direction. In particular, (a) we formulate a contact model that is *safe* independent of user parameters; (b) we *correctly* discretize time, using asynchrony to preserve the model's safety and to respect causality, and using a symplectic-momentum integrator to exactly conserve momentum and approximately conserve energy over long run times. (c) We lay out the basic foundations for the union of AVIs with KDSs, making tractable the safe, correct integration of complex contact for highly deformable objects. Finally, we (d) expose a simple model of dissipation and friction that preserves symmetries of immersion and behaves consistently across changes to temporal discretization.

Method in brief Throughout our exposition, we will refer to line numbers of Algorithm 1, which summarizes the event-driven simulation loop. Rather than keeping the entire configuration synchronized in time, each vertex i stores its "most recently seen" position $\mathbf{x}_i$ and velocity $\dot{\mathbf{x}}_i$ as recorded at time t_i. Events, each embodying some simple local atomic action, are drawn and processed in causal order (see algorithm LINE 2). The state of all vertices in the *stencil* of this drawn event must be advanced to the current time (LINES 3–7). When a force event is drawn, we apply impulses to the local stencil of vertices (see LINE 9 and §3); since the impulses affect the vertices' future trajectories, we must update the continuous-time collision-detection data structures (see LINES 11–14 and §5). Some events embody data structure *certificate* updates but do not affect the trajectory (see LINES 15–18 and §5).

```
1:  loop
2:    (E,V,h,t) ← Q.pop                    // Pop event E with potential V, time step h,
                                           // and scheduled time t, from time-ordered queue Q
3:    ξ := stencil(E)                      // global indices of the local stencil
4:    for i ∈ ξ do
5:      x_i ← x_i + (t − t_i)ẋ_i           // advance vertex to current time (see §3)
6:      t_i ← t                            // update vertex's clock
7:    end for
8:    if E is a (external, internal, contact) force event then
9:      q̇_ξ ← q̇_ξ − hM_ξ^{-1} ∂V/∂q_ξ      // local impulses, local mass (see §3)
10:     Q.push(E,V,h,t+h)                  // Return the event to the queue, with new time
11:     for j ∈ ⋃_{i∈ξ} contingent(i) do
12:       s ← failureTime(E_j)             // compute new event time (see §5.1)
13:       Q.update(E_j,s)                  // reschedule the contingent event (see §5.2)
14:     end for
15:   else if E is certificate failure then
16:     update KDS certificate, reschedule in Q   // see §5.1 and §5.3
17:     (de)activate penalty forces               // see §4
18:   end if
19: end loop
```

2 Related work

Computational Contact Mechanics is a well-studied problem [Wriggers and Laursen 2007; Johnson 2008] of constraint enforcement: a physical trajectory travels only through the *admissible region*—the subspace of collision-free configurations (see Fig. 3). Framing collision response as an instance of constraint enforcement enables future generalizations of our method to other constraints (*e.g.*, inextensibility enforcement in §7).

To enforce constraints, engineers turn to *penalty* forces. As noted by Wriggers and Panagiotopoulos [1999], analysis begins with the impulsive penalty force, an infinite spike where bodies are in contact and zero elsewhere. The spike is impossible to model with a conservative force, necessitating approxima-

tion with quadratic or higher-order *penalty potentials*. In deviating from true impulses, penalty potentials permit visible penetration; stiffening the force helps, but it also induces smaller time steps. On the other hand, a low stiffeness leads to disasterous *tunnelling* through the inadmissible region (see Fig. 3). These drawbacks motivate adoption of *Lagrange multipliers* and unilateral contact laws [Pfeiffer and Glocker 2000; Eck et al. 2005], where constraint-enforcing balance constraint-violating forces. Multiple simultaneous contacts induce linear complementarity problems (LCPs) [Wriggers and Laursen 2007], with their attendant complexity and numerical pitfalls.

Graphics and robotics have embraced these developments, extending them with an eye to simplicity and efficiency. Terzopoulos et al. [1987] used penalty methods to treat contact between elastic bodies. Hahn [1988], Mirtich and Canny [1995] used impulses, viewing contact as micro-collisions, while Baraff [1989; 1994], Stewart and Trinkle [1996] presented LCP treatments for multiple simultaneous contacts with friction. Specifically targeting complex cloth collisions, Bridson et al. [2002] present a velocity filter that combines the advantages of penalty and impulsive methods, and relies on a geometric approximation for difficult impact zones [Provot 1997; Harmon et al. 2008]; geometric approaches are also instrumental in resolving pinching and other challenging configurations [Baraff et al. 2003; Volino and Magnenat-Thalmann 2006; Sifakis et al. 2008]. Recently, Guendelman et al. [2003] and Kaufman et al. [2005; 2008] treated complex stacking and friction for rigid bodies. An attempt to directly incorporate these collision algorithms into AVIs faces two challenges: many methods amortize cost by assuming temporal synchronization; a straightforward interleaving of contact-response and symplectic integration algorithms breaks the latter's good momentum and energy behavior (see §3).

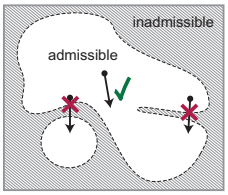


Figure 3. Trajectories (*arrows*) must remain in the admissible (*white*) region.

Several works consider asynchronous handling of contact. Lubachevsky [1991] used an event-driven priority-queue algorithm to simulate billiard balls, Celes [1998] handled contact between multiple mass-spring bodies, and such approaches extend to granular materials [Pöschel and Schwager 2005]. Mirtich [2000] enabled aggressive advancement of rigid body simulations with provably-correct partial-state rollback to fix missed collisions. Debunne et al. [2001] considered multirate time integration for simulation of visco-elastica. Dequidt et al. [2004] reframed asynchrony from an autonomous agent perspective. Thomaszewski et al. [2008] applied AVIs to cloth simulation, using a three-pass approach that aims to efficiently resolve collisions.

What sets our work apart is the focal triad of safety, correctness, and progress. Methods that prioritize progress by relaxing correctness can have downstream costs of simulation setup, feature development, and artifact resolution. For example, many popular methods for cloth simulation justifiably assume a zero coefficient of restitution (COR). These assumptions can be so deeply ingrained that allowing adjustment of CORs is impossible without a major overhaul or painstaking parameter-tuning. As another example, local ("Gauss-Seidel" or "Jacobi") iterative techniques essentially optimize for the case of light collisions, resorting to (unphysical) "failsafes" when the going gets tough.

In summary, the mechanics literature describes physical models for contact, but lacks many of the sophisticated algorithms considered by computer scientists; meanwhile, the trend in graphics has been to start with a fast but approximate solution, and then to chip away at the unphysical artifacts and the lack of scalability. By contrast, we begin with a more costly, but geometrically safe and physically conservative method, and build up efficiency using tools such as asynchrony and persistence.

3 Asynchronous variational integrators

Consider a mechanical system with a time-varying configuration $\mathbf{q}(t)$ in the space $\mathbf{Q}$ of all configurations; concretely, for a mesh with vertices $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in 3D we represent $\mathbf{Q} = \mathbb{R}^{3n}$ by a vector of all the vertices' Cartesian coordinates. We use a dot to denote differentiation in time, so that $\dot{\mathbf{q}}(t)$ is the configurational velocity. Let M be the mass matrix, so that $\mathbf{p} = M\dot{\mathbf{q}}$ is the momentum.

The Verlet integrator evolves a sequence of positions $\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2 \ldots$ and momenta $\mathbf{p}^0, \mathbf{p}^1, \mathbf{p}^2 \ldots$ via the update rules

$$\mathbf{q}_k - \mathbf{q}_{k-1} = hM^{-1}\mathbf{p}^{k-1} \,, \quad \mathbf{p}^k - \mathbf{p}^{k-1} = hF(\mathbf{q}_k) \,, \quad t_k - t_{k-1} = h \,,$$

where h is the time step and $F(\mathbf{q})$ is the force. The sub/superscripted indices allude to the method's alias, *leapfrog*, reminding us that positions and velocities are staggered in time, with t_k associated to $\mathbf{q}_k$, and (t_k, t_{k+1}) associated to $\mathbf{p}^k$. In effect, Verlet first updates the position at t_k using the constant momentum associated to the preceding interval (t_{k-1}, t_k) (Algorithm LINE 5), and then impulsively "kicks," obtaining a new momentum for the following interval (t_k, t_{k+1}) (Algorithm LINE 9), yielding a piecewise linear (p.l.) trajectory over the intervals (t_k, t_{k+1}). A *geometric integrator* [Hairer et al. 2002; Kharevych et al. 2006], Verlet tracks conservation laws (*e.g.*, mass, momentum, energy) and adiabatic invariants (*e.g.*, temperature) over long run times, and offers more consistency and qualitatively predictable behavior across a range of time step sizes.

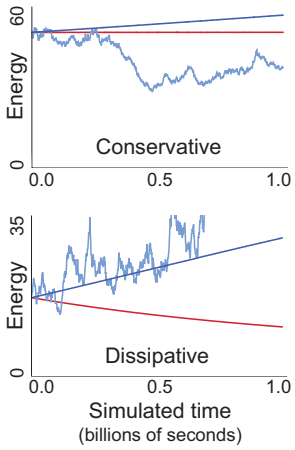
AVIs naturally extend Verlet. Each force receives an independent, regular (fixed-rate) clock, fixed a priori by stability requirements. While impulses of a force are regularly spaced in time, the superposition of forces yields events irregular in time. As with Verlet, the trajectory is p.l., interrupted by "kicks." When their clocks are nested—as quarter notes are nested in half notes—AVIs reduce to an instance of multistepping methods [Hairer et al. 2002]. Our developments apply to this family of methods.

For example, Lew et al. [2003] assign an elastic potential to each mesh element. Irregular meshes have spatially-varying element shapes and corresponding time step stability restrictions; with AVIs each element advances at its own pace. Since an elemental potential depends only on a local mesh nieghborhood, each integration event is *local*, affecting the position and velocity of a small number of *stencil* vertices. Correspondingly, Algorithm LINE 9 uses the local forces and mass matrix.

To schedule the interrupts to the p.l. trajectory, AVIs use a priority queue, conceptually populated with all event times until eternity. In practice it suffices to schedule only the next tick for each clock, since that event can schedule the subsequent tock (LINE 10).

Ensuring correctness A more complete analysis leading to the geometric and conservation properties of AVIs invokes ideas from discrete mechanics and variational integration [Marsden et al. 1998; Lew et al. 2003]. Here we stress a key outcome: Lew et al. conjecture that AVIs' remarkable properties are due to its *multisymplecticity* (a property we further develop in a technical report [Vouga et al. 2009]); the derivation requires each force to have a regular (constant-rate, ever-ticking) clock. Playing with this clock—accelerating or pausing—is strictly forbidden. Interrupting the p.l. trajectory with other mechanisms (*e.g.*, interleaving a velocity filter) breaks multisymplecticity.

We demonstrate the perils of tampering with the clock. A free spring of unit stiffness, rest length, and endpoint masses is placed one unit above a ground plane, in a vertical "pogo stick" orientation, and allowed to bounce repeatedly on the ground under gravity. We simulate thrice: with ordinary AVIs for gravity, the spring, and contact penalty (red horizontal line); as before, but restarting the contact penalty clock at the instant of contact (dark blue sloped line); and with AVIs for gravity and the spring, but resolving collisions with a reflective impulse at the contact instant (light blue jagged curve). The time-evolution of total energy reveals that first approach has no evident energy drift, whereas the second systematically injects energy and the third takes a random walk. Good energy behavior is equally important for dissipative systems. We add a small dashpot and repeat the experiment. Only the regularly clocked penalty force yields the expected, controllable energy dissipation.



In large-scale simulations, we observe that tampering with the clock leads to instabilities and inconsistent behavior across mesh resolutions. Supporting the observed difficulties, Zhong and Marsden [1988] prove that symplectic-momentum-energy preserving methods of regular time step do not exist (except for certain integrable systems); one cannot hope to interleave an energy-momentum collision integration with a symplectic-momentum force integration and retain either set of properties.

AVIs and contact To the best of our knowledge, the problem of extending AVIs to handle contact mechanics remains open. The conservation properties of AVIs rely on preservation of the multisymplectic form [Marsden et al. 1998; Marsden et al. 2001], and are easily broken by naïvely incorporating existing contact-resolution methods. A principled treatment must consider a multisymplectic formulation of contact mechanics, and an asynchronous computation of collision detection and response.

4 Discrete penalty layers

Consider a simple penalty method that penalizes proximity between bodies. For a given surface thickness η, the gap function

$$g_\eta(\mathbf{q}) = \|\mathbf{x}_b - \mathbf{x}_a\| - \eta$$

tracks signed proximity between moving points $\mathbf{x}_a$ and $\mathbf{x}_b$. When $g < 0$, the points are said to be *proximate*. We can express the contact (or "interaction") potential and force in terms of g

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

respectively, 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 (see Fig. 3).

The *barrier method* replaces the above contact potential by a function that grows unbounded as the configuration nears the boundary $g(\mathbf{q}) = 0$, eliminating the possibility of tunneling. 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].

To alleviate these concerns, we propose a construction consisting of an infinite family of *nested potentials*

$$V^{r(l)}_{\eta(l)}, \qquad l = 1, 2, \dots\,,$$

where $\eta(l)$ is a monotonically decreasing proximity (or "thickness") for the l-th potential, and $r(l)$ is a monotonically increasing penalty stiffness. For these nested potentials to be a barrier, the cumulative energy of these potentials must diverge as the distance between two primitives vanishes:

$$\sum_l r(l)\eta(l)^2 \to \infty.$$

We use $r(l) = r(1)l^3$ and $\eta(l) = \eta(1)l^{-1/4}$ [Vouga et al. 2009], where $r(1)$ and $\eta(1)$ are a simulation-dependent base stiffness and thickness for the outermost layer.

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 Fig. 4). The nested potentials' respective maximal stable time steps form a decaying sequence, and therefore this construction *requires* an adaptive or asynchronous time stepping algorithm. Each interaction potential has its own integration clock, and has the opportunity to apply an impulse when its clock ticks. The question is how to time step such an infinite sequence.

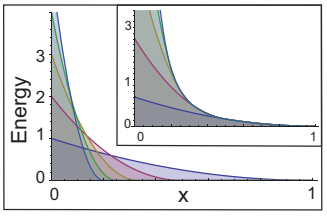


Figure 4. *Discrete penalty layers.* Potential energy of layer n plotted against proximity; *Inset:* total potential energy contributed by all layers $\le n$. The potential energy diverges as x_a approaches x_b, guaranteeing that constraint enforcement is robust.

As we are about to see, the above construction transforms a seemingly intractable problem in Computational Mechanics—establishing a multisymplectic treatment of contact mechanics with *guaranteed* absence of tunneling—into a challenging but addressable problem in Computer Science: efficient bookkeeping on a conceptually infinite set of interaction potentials.

Central observation. During any time interval, while conceptually the (infinite number of) clocks continue to tick, and the totality of the clock ticks is dense in time, only a *finite, sparse* set of clock ticks apply (non-zero) impulses. In particular, the index of the discrete penalty layer indicates the number of *active* potentials; the rest, while conceptually present, do not influence the trajectory, and can be culled without approximation. What is needed is efficient bookkeeping to track which interaction potentials are active; each status change corresponds to a transition between penalty layers—a *discrete* change in state due to motion along a *continuous* trajectory. This is a problem that KDSs were born to solve.

5 Kinetic Data Structures for AVIs

Guibas [1998] gives an overview of kinetic data structures. Our culling of inactive forces uses an implementation of kinetic separating slabs, closely related to those used by Guibas et al. [2001b] in the context of rigid polytopes.

5.1 Kinetic separating slabs

As an illustrative example, consider a single particle falling toward a fixed floor. Conceptually, the clock for the first penalty layer is always ticking; however, it is active (exerting a nonzero impulse) only when the particle drops below height $\eta(1)$, say at time t. We must "activate the clock," placing it on the priority queue for explicit consideration, no later than time t. Activating too late introduces error (misses impulses), while activating too early is correct, albeit overly conservative (some null events are not culled).

Suppose that calculating t is expensive. A conservative optimization uses an $\eta(1)$-slab—a line extruded to thickness $\eta(1)$—separating the particle from the floor. The separating $\eta(1)$-slab serves as a proof, or *certificate*, that the the particle and floor are at least $\eta(1)$ apart. This guarantee remains valid until either the floor or the particle enters the slab, at which point the certificate *fails*: we can try to find a new slab, or if doing so is costly or impossible, activate the first penalty force.

Concretely, simulation begins with identifying an $\eta(1)$-slab (see Fig. 5); from the initial vertex state, and assuming a straight line trajectory, we compute the time t when the particle enters the slab (LINE 12), and schedule this certificate *failure event* on the priority queue (LINE 13). If t is hard to compute, any earlier time $t_1^{\text{cf}} < t$ is correct but conservative.

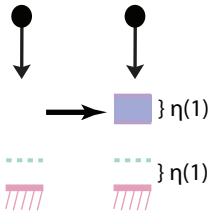


Figure 5. A separating slab KDS is created as proof of no contact.

At time t_1^{cf}, the failure event pops off of the queue (LINE 2). We check the separation distance; suppose it exceeds $\eta(1)$. We identify a new $\eta(1)$-slab (LINE 16), and schedule a new failure event, say at time t_2^{cf} (see Fig. 6).

Suppose that the next event, at time $t^{\text{g}} < t_2^{\text{cf}}$, corresponds to integration of gravity. We integrate the particle position, based on its last-known state and the elapsed time (LINE 5); we integrate the particle velocity based on the gravitational force (LINE 9). Failure time t_2^{cf} was computed assuming a constant velocity, an assumption now broken; t_2^{cf} might no longer be conservative, so the failure event must be rescheduled to guarantee safety (LINES 11–14).

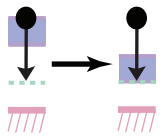


Figure 6. A new separating slab is scheduled.

The simulation continues in this manner. As the particle approaches the floor, the benefits of culling clock ticks of penalty layer one are eventually outweighed by the increasing frequency of $\eta(1)$-slab events. Our implementation considers the trade-off to occur when the separation distance is below $\frac{11}{10}\eta(1)$; the decision of how to flag this trade-off affects performance but not safety or correctness.

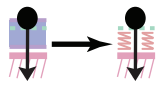


Figure 7. No efficient slab exists, so a penalty force is activated.

When the use of $\eta(1)$-slabs is no longer considered profitable, we activate the layer-one penalty clock (LINE 17) and forgo $\eta(1)$-slabs (see Fig. 7). We cull clock ticks of only deeper layers. We certify inactivity by identifying an $\eta(2)$-slab, computing and processing failure, reconstruction, and rescheduling as described above.

With the layer-one clock active, we soon encounter a layer-one penalty force integration event. This event is treated in the same manner as the gravity event or any force event. Furthermore, this event serves as an opportunity to check whether the particle is transitioning to a shallower penalty layer: if (a) the penalty impulse is null, *i.e.*, separation distance exceeds $\eta(1)$, and (b) the relative velocity is separating rather than approaching, then we de-activate the penalty force, transitioning to the next-shallower layer, and adjusting the certificates accordingly. This lazy approach to deactivation is safe by clause (a) alone; clause (b) aids in efficiency, avoiding rapid toggling of penalty layers.

Event	Supporting vertices	Stencil vertices
Gravity		Entire mesh
Stretching force [Lew et al. 2003]		Triangle
Bending force [Grinspun et al. 2003]		Hinge
Penalty force (§4)		Pair of primitives
Separation slab (§5.1)	Pair of primitives	
k-DOP overlap (§5.5)	Those in k-DOP	
Render frame		

Table 1. Events and their associated supports and stencils.

5.2 Stencils, supports, and scheduling dependencies

With a basic depiction of a KDS in place, we proceed to discuss efficiency and optimization, after laying out the requisite terminology. Consider the execution of an event at its scheduled time. The set of vertices whose velocities are altered is the *stencil* of the event. The set of vertices whose trajectory was used to schedule this time is the *support* of that event Building on the notions of stencil and support, an event *depends*, or is *contingent*, on another if the support of the former overlaps the stencil of the latter; vice versa, an event *supports* another if the stencil of the former overlaps the support of the latter. Table 1 shows the support and stencils for a set of typical events.

To our knowledge, KDSs were previously applied only to synchronous simulations, where the velocities of all primitives are updated at the same instant, *i.e.*, the stencil of *the* force-integration event contains the set of *all* vertices. By contrast, in an AVI simulation, force-integration events typically bear small stencils.

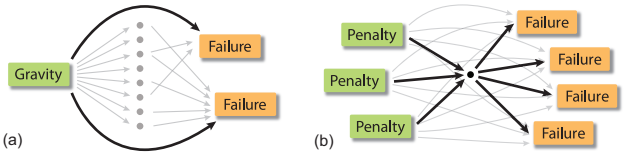


Figure 8. Directed graphs depicting events (boxes), vertices (dots), and dependencies (directed edges). Integration events (left green boxes) alter vertex trajectories, forcing rescheduling of dependent events (right orange boxes). (a) If an integration event has a large stencil, we store event-event dependencies. (b) If a vertex belongs to multiple stencil and support relations, we store event-vertex-event dependencies.

Having executed a supporting event, we must reschedule all dependent events before proceeding (Algorithm LINES 11–14). This is a problem of executing partially ordered instructions with dependencies, and it is thoroughly studied in the computer systems literature [Korneev and Kiselev 2004].

Our implementation maintains a directed graph, where edges from events to vertices and vice versa denote stencil and support relations, respectively. When an event executes, the two-neighborhood of outgoing edges yields the set of events to reschedule. The graph abstraction reveals that events with large stencils, such as gravity, should cache a list of contingent events, while events with small stencils should construct the list of contingent events on-the-fly; refer to Figs. 8a and 8b, respectively.

5.3 Thinning out certificate rescheduling

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. 2001a; 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. Because we are using KDSs specifically in the context of contact mechanics, we can bring into play physical insights that would otherwise not be at our disposal. As an illustrative example, consider Newton's apple, which after being tossed into the air follows a parabolic trajectory before hitting the ground. We now split the apple and connect the two halves with a stiff spring. When we 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, we consider trajectories with bounded uncertainty. In place of precise linear trajectories, we consider "tubes" wide enough to encompass the noisy oscillations. On the one hand, this requires us to compute certificate expiration times that are conservative in the sense that they are valid for any precise trajectory that fits in the tube. 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 tube. If the predicted tube is not too thick, and if the actual trajectory remains inside the predicted tube for sufficient time, we could potentially reap a (safe, correct) dramatic reduction in rescheduling.

We pursue a simple implementation motivated by this idea. Recall our scheduling approach for the simple separating slab KDS. After creating a new certificate (say at time $t = t_0$), we scheduled a certificate failure time 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, we weaken the assumption to allow for a time-varying velocity. We therefore let the velocity of the particle $\dot{\mathbf{x}}(t) = \dot{\mathbf{x}}(t_0) + \mathbf{u}(t)$, where $\mathbf{u}(t)$ is a time-varying vector of bounded length $\|\mathbf{u}(t)\| \leq \varepsilon$. 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 $\|\dot{\mathbf{x}}(t) - \dot{\mathbf{x}}(t_0)\| \leq \varepsilon$. Indeed, for $\varepsilon < |\dot{\mathbf{x}}(t_0)|$, 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.*, we must compute the earliest possible failure time. For the separating slab, the trajectory producing the earliest failure "worst case" failure is the one maintaining $\|\mathbf{u}(t)\| = \varepsilon$ 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.

5.4 Broad phase

Our implementation begins with the simple separating slab KDS described above, modified so that slabs have constant (rather than zero) normal velocity. We consider this the "narrow phase."

While formally correct, the simple KDS used on its own will not scale efficiently to large scenes. Various sophisticated KDSs track proximity, offer better "broad-phase" scaling, and could be easily adapted to the bookkeeping of the DPL index [Basch et al. 1997; Erickson et al. 1999; Guibas et al. 2001a; Agarwal et al. 2002; Gao et al. 2003; Agarwal et al. 2004; Gao et al. 2005]. We have not implemented all the available methods, thus rather than advocating for one candidate, we dedicate our exposition (recall §5.2–5.3 and see §5.6 below) to those concepts particular to the synthesis of AVIs with KDSs, independent of the chosen KDS. For completeness we briefly describe our implementation of a broad-phase KDS, then return to cross-cutting concepts.

5.5 Kinetic k-DOP hierarchy

A *k-discrete oriented polytope* (k-DOP) is a bounding volume (BV) described by $k/2$ real intervals $S_i = [\alpha_i, \beta_i], 1 \leq i \leq k/2$, each describing an object's extent (or "support") along some predetermined supporting axis $\mathbf{d}_i$ [Konečný and Zikan 1997]. For $k = 6$ and orthogonal axes, k-DOPs reduce to axis-aligned bounding boxes (AABBs). For $k \to \infty$, k-DOPs approximate convex hulls.

Like most BVs, k-DOPs work best in a hierarchy whose leaves bound primitives and progressively coarser levels bound aggregates [Klosowski et al. 1998]. In synchronous simulations, a collision step updates ("rebuilds") and traverses ("broad phase detection") the entire hierarchy; the cost is amortized over the consideration of all pairwise collisions in the scene. This economy of scale does not immediately carry over to AVIs, where each integration step updates only a handful of primitives.

The KDS amortizes not over space but over time: as the position of primitives continuously evolve, we identify discrete transitions in execution flow of the broad-phase traversal algorithm. Such a *kinetization* of the hierarchy traversal is described in detail by Weller and Zachmann [2006] in the context of AABBs and synchronous simulations, but the general idea of incrementally updating a collision-detection tree traversal is known well beyond the KDS literature [Ericson 2004]). In retrospect, the approach seems to fit most naturally as a component of an asynchronous simulation, yet we are not aware of prior work harnessesing the natural affinity of KDSs and AVIs.

For implementation details, we refer the reader to Weller and Zachmann's exposition [2006]. Here we describe only the k-DOP and AVI-specific concepts that the former work did not explore.

Since a hierarchical BV algorithm decides whether to recurse by testing the overlap of two k-DOPs, the associated kinetic proof uses overlap and non-overlap certificates; the failure times thus correspond to instants at which two k-DOPs become (non-)overlapping.

Consider a certificate guaranteeing non-overlap of k-DOPs a and b. The simplest proof identifies a single axis $\mathbf{d}_i$ with disjoint extents. This proof is valid during time intervals where $S_i^a \cap S_i^b = \emptyset$, *i.e.*, $\alpha_i^b - \beta_i^a > 0$ or $\alpha_i^a - \beta_i^b > 0$. Following our didactic example (§5.1), assume that the vertex trajectories are linear in time. Since a k-DOP contains multiple vertices of differing positions and velocities, the upper extent $\beta_i^a(t)$ is convex piecewise linear in time,

where kinks correspond to a new "leader" overtaking the extremal vertex (see Fig. 9); likewise, $\alpha_i^a(t)$, $\alpha_i^b(t)$, $\beta_i^b(t)$ are convex p.l.. Therefore, computing the failure time reduces to finding the first instant at which a p.l. function becomes negative.

A more ambitious certificate uses the k-DOP's *multiple* axes, and the observation that the failure of one axis need not bring down the whole certificate. Each axis $\mathbf{d}_i$ yields a set of time intervals where the a and b are separated; the certificate is valid over the union of all positive intervals associated to all $k/2$ axes, *i.e.*, it fails at the instant where all $k/2$ axes have negative p.l. functions.

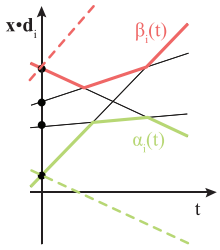


Figure 9. One axis of a k-DOP bounding 4 vertices over time. The bounds $\alpha_i(t)$ and $\beta_i(t)$ are linear functions. The dashed lines show the linear envelope which is a conservative bound for all $t > 0$.

5.6 Fast certificate scheduling

The Achilles' heel of KDSs is the frequency and cost of rescheduling. We reduced rescheduling frequency using vague trajectories; now we explore reducing rescheduling cost.

Solving for a certificate failure time requires fast root finding techniques [Guibas et al. 2004]. Even when vertex trajectories are linear in time, the algebraic function represented by the certificate can have non-trivial algebraic complexity. We avoid these numerical issues by using only certificates whose failure times are roots of a univariate p.l. polynomial, requiring identification of the ("piecewise") segment followed by a subtraction and a division for the ("linear") solve.

The effort of computing a certificate failure time goes in vain when a supporting event executes in the interlude. The more distant the failure time, the more likely the wasted effort. Can we avoid precise scheduling without compromising safety or correctness?

The general answer is to quickly compute a *safe approximate time* guaranteed not to exceed the actual certificate failure time [Guibas et al. 2001a]. The associated *reconfirmation* event no longer implies a certificate failure; instead it reconfirms the current proof by attempting to find a future reconfirmation time. When a reconfirmation event fails to find a safe future time, we can either schedule a true failure time, or treat the failed reconfirmation as a (premature) certificate failure. Either approach is safe, but the latter (our choice) fully eliminates the typically more complex implementation of precise failure time computation. We implemented two conservative approximations:

Linear envelope on k-DOP extent: A p.l. function $f(t)$ over $t \geq 0$ can be bounded from above by a linear function $\hat{f}(t) = f(0) + mt$, with slope m the maximum over the slopes of the pieces of f (see Fig. 9); a bound from below follows similarly. We use this fact to find conservative (non-)overlap times for k-DOP extents.

Adaptively short-circuit to the most useful k-DOP axes: Not all k-DOP axes are created equal. Depending on the configuration, some excel while others fail in establishing separation. Can we 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: we first assume that the k axes are already (nearly) sorted from most- to least-useful, and we progressively improve our bound by incorporating an additional axis, until an axis fails to improve the bound; in the second step, we improve the sorting (for next time) by attempting to incorporate one random unused axis, and promoting this axis to the front of the list if it did improve the 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. 2005]; we dynamically update the coreset constituency as the system evolves.

6 Dissipative Forces

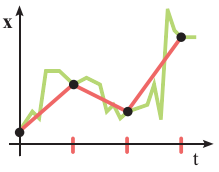
In using a geometric integrator, our approach exhibits energy near-conservation for long run times. Controlled dissipation, in the form of friction, impact coefficient of restitution, or viscous damping of high-frequency modes, is often desired in practical simulations. Our emphasis here is not on advanced models of dissipation, rather on basic ideas for incorporating *controlled* dissipation into our framework without compromising safety, correctness, and progress.

We ask that the limiting behavior as two events are brought to coincide should be *continuous*, *i.e.*, unique and independent of how this limit is approached. Simultaneous events must yield an order-independent outcome. If a perturbation to the problem setup (*e.g.*, time step size, initial vertex position) or a numerical error perturbs the order of nearly contemporaneous events, *reordering continuity* keeps the trajectory predictable.

Sufficient condition. If forces depend on positions, past and present, but *not* on momenta, then the trajectory is independent of the processing order of simultaneous events. *Proof:* an event outcome affects only future positions; a computation based on past and present positions is unaffected by outcomes of simultaneous events.

Conservative forces trivially satisfy this condition. However, since the most straightforward implementation of dissipation computes momenta-dependent quantities, it fails to meet our sufficient conditions, and generally leads to order-dependent outcomes.

Fortunately, a simple solution is at hand. Where a force formulation calls for momentum, we use the Verlet identity $\mathbf{p}^{k-1} = M(\mathbf{q}_k - \mathbf{q}_{k-1})/h$, a finite difference of past and present positions. To ensure a well-behaved, regular temporal discretization, we consider *only the positions associated to the dissipative force's clock.* This corresponds to the average momentum in the interval between dissipative events, which, because of the interruptions induced by other asynchronous clocks, will generally not correspond to the momentum immediately preceding the dissipative event. All of this is illustrated in the adjacent figure where the red trajectory shows what is being used for the computation of the dissipative force, while the green trajectory includes updates from all events. We invoke this concept in discretizing several dissipative forces.



6.1 Viscous damping

Consider an elastic spring connecting two vertices i and j. Viscous damping acts to slow the *rate* at which the spring changes length. We create a new clock (we could also ride on the elastic clock, stability permitting) and compute a viscous force

$$\mathbf{F}_{k,i} = b(l_k - l_{k-1})\mathbf{e}_k/l_k = -\mathbf{F}_{k,j}\,, \quad \mathbf{e}_k = \mathbf{q}_{k,j} - \mathbf{q}_{k,i}\,, \quad l_k = \|\mathbf{e}_k\|\,,$$

where b is the damping coefficient, $\mathbf{q}_{k,i}$ is the position of the i-th vertex, and the vector $\mathbf{e}_k$ and length l_k are local; a subscript (i,j) is implied. We cache l_{k-1} so that we can use its value at time t_k.

In the case of a single spring, the approach reduces to explicit Verlet integration of the viscous force, with its attendant time step restriction. Bridson et al. [2003] advocate a semi-implicit integration of

the viscous force. Such an approach might also be adopted in place of the one presented above, trading order independent processing of simultaneous events for larger viscous force time steps.

Most of our simulations incorporate some viscous internal (stretching and bending) damping. The curtains (see Fig. 1 and video) illustrate the benefit of starting from a conservative foundation. Using only internal damping, high-frequency vibrations introduced by the prescribed particle are quickly damped out, while the curtains' graceful swinging continues; using a non-geometric integrator such as backward Euler or BDF2 [Ascher and Petzold 1998], the swinging motion would also be damped.

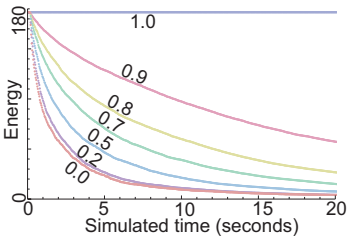
6.2 Coefficient of restitution

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]

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

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 $g(\mathbf{q})$.

While simple, our appraoch has a drawback in the inelastic limit $e_{\text{COR}} = 0$: the penalty impulses can leave as residue a small separating relative velocity. 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.



The long-term good energy behavior accompanying our use of a symplectic-momentum integrator translates into predictable, controllable energy dissipation when a non-unit coefficient of restitution is used in a simulation. To test the energy behavior for a variety of coefficients of restitution, we simulated a box of 900 particles with random initial velocities. The incident figure 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. Fig. 10 illustrates that our method reproduces this clustering when the above experiment is run with $e_{\text{COR}} = 0$.

6.3 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.

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 halt high-speed tangential motion.

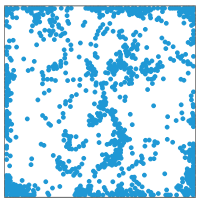


Figure 10. Dissipative collisions form characteristic clusters.

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.

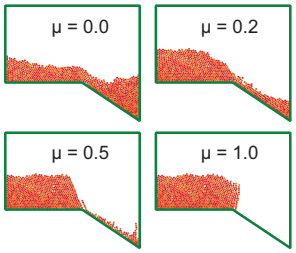


Figure 11. Friction alters the flow of sludge.

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. Fig. 11 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 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.

7 Generalization: Inextensibility constraint

As foreshadowed in §2, nested penalty layers can enforce a broader class of unilateral constraints. Consider edge inextensibility constraints in 2D. For two vertices i and j delimiting an edge of rest length ℓ, we constrain this edge to not compress or extend by $\pm s\ell$, for some $s \ll 1$. We introduce two nested sequences of penalty layers, their gap functions replaced by constraint functions

$$g_i(\mathbf{q}) = (1+s)\ell - \|\mathbf{x}_j - \mathbf{x}_i\|$$
$$g_j(\mathbf{q}) = \|\mathbf{x}_j - \mathbf{x}_i\| - (1-s)\ell\ .$$

The penalty layers associated with g_i and g_j prevent excess stretch and compression, respectively.

We simulate a cloth curtain hit by a fast-moving projectile (refer to supplemental video), comparing implementations based on constraints and elastic springs. For any chosen spring stiffness, a sufficiently energetic projectile stretches the curtain by arbitrary amounts, resulting in a "rubbery" curtain. On the other hand, enforcing inextensibility using nested penalty layers avoids stretching

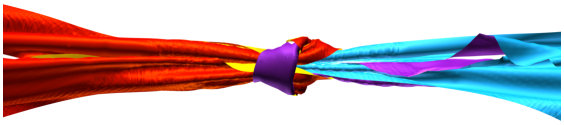


Figure 12. Simulated tying of ribbons into a reef knot.

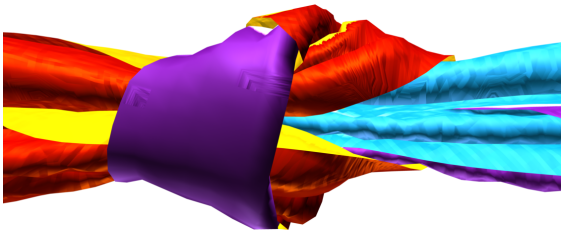


Figure 13. A closeup of the reef knot.

no matter the projectile's velocity. Implementation of the inextensibility constraint for $3D$ triangle meshes would require a constraint formulation that does not lock bending modes, such as that proposed by English and Bridson [2008].

8 Results

In §3 and §6, we described simple experiments and empirical measurements supporting the guaranteed safety and good energy behavior of the proposed contact algorithm, for both conservative and dissipative contact. 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 Figs. 12 and 14, 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.

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 reagent, *i.e.*, a tighter knot cannot be tied without splitting

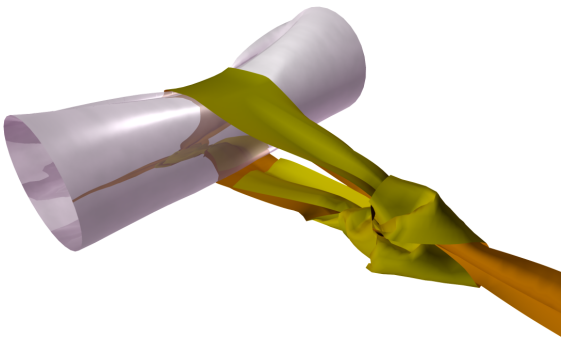


Figure 14. Simulated tying of a ribbon into a bowline knot.

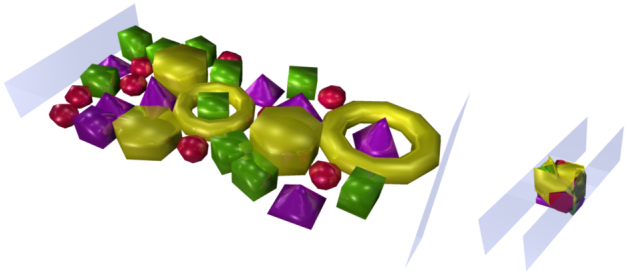


Figure 15. Virtual trash compactor and assorted virtual trash.

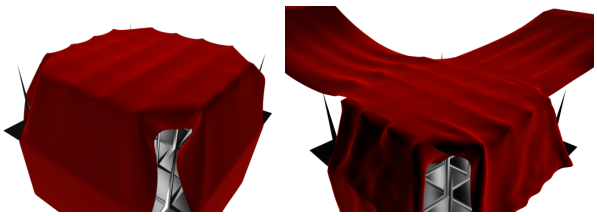
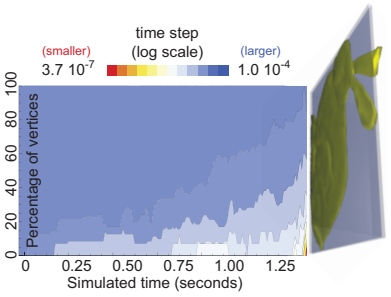


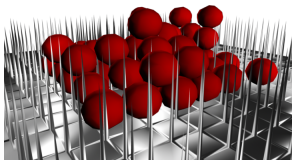
Figure 16. Experiments with a bed of nails highlight the method's ability to deal with sharp boundaries, isolated points of contact, sliver triangles, and localized points of high pressure between two nearly incident surfaces.

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's safety guarantee induces Zeno's Paradox.

Trash compactor We place triangle meshes of varying complexity into a virtual trash compactor consisting of a floor and four walls, and then prescribe the inward motion of opposing walls (see Fig. 15 and incident image). The method is able to simulate the approach of the walls without ever allowing for seen or unseen penetrations. As with the knots, the overall rate of progress decays as the simulation approaches a limiting configuration.



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 Fig. 16), 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.



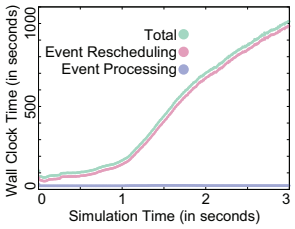
We extend the bed of nails into a landing pad for various coarsely-meshed projectiles. Variably-sized to barely fit or not fit between

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

Timing We list computation time for the various examples, as executed on a single thread of a 3.06Ghz Intel Xeon with 4GB RAM. The bulk is allocated to the maintenance of the kinetic data structures used for collision detection. We measured the effect of introducing vague trajectories for the reef knot and bunny compactor, and observed at approximately 30% improvement in overall performance.

Examples	Vertices	Simulation Seconds	Event Processing (hours)	KDS Event Rescheduling (hours)	Total (hours)
Reef Knot	10642	2.00	1.5	16.7	18.5
Bowline Knot	3995	5.00	3.0	141.1	144.5
Trash Compactor	714	3.08	0.5	53.0	53.6
Two Sheets Draped	15982	3.95	4.5	260.8	265.5
Two Sheets Pulled	15982	3.83	13.6	310.5	325.6

As a more detailed study, consider that the reef knot simulation required 4.8% of total simulation time for integration of elastic forces and gravity, 0.09% for integration of penalty forces, 0.9% for processing and 1.0% for rescheduling of separating plane events, respectively, 5.2% and 23.0% for processing and rescheduling of separation list events, respectively. All other time was spent performing vague trajectory checks and queue maintenance. The incident figure demonstrates how per frame runtime increases as the stress on the ribbons elevates.



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.

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	-	-	-

9 Discussion

Parameters and the triad of safety, correctness, and progress One of our driving goals is to investigate methods that ensure safety, correctenss, and progress regardless of the choice of parameters. The method proposed here does expose some parameters to the user, such as the proximity η and the trajectory vagueness ε. These parameters affect performance, not the triad of guarantees. Our experience in running the problem scenarios, therefore, were qualitatively different than when using other methods, in that we did not need to search for parameters to ensure a successful modeling of contact. On the other hand, our method does not address the spatial discretization of elasticity (stretching and bending models), which can also require user tuning.

Although in theory the nested penalty barrier has infinitely many penalty layers at its disposal, it is impractical to activate penalty layers whose stable time steps are too small, *e.g.*, below the floating point epsilon. Simulations with thicknesses $\eta(1)$ too small, or velocities or masses too high, can thus fail to make progress (but remain safe). This limitation can be worked around by choosing a slow-shrinking layer distribution function, which is why we recommend $\eta(l) = \eta(1)l^{-1/4}$. For more on this we refer to the accompanying technical report.

Multistepping methods such as AVIs are known to have *resonance instabilities* [Hairer et al. 2002; Fong et al. 2008], particularly if the simulation contains adjacent mesh elements of very different size. However, we have not observed any such instabilities or artifacts that we can attribute to such instabilities in our use of the method.

Broader exploration In this paper we were concerned with building the most robust contact implementation we could; therefore, we tied the knots as tight as possible, until each triangle was packed as tightly as possible into its neighbors. In the tightest configurations the spatial discretization becomes evident. It would therefore be interesting to introduce spatial adaptation, refining the mesh where curvature is high. Another alternative would be to improve the smoothness at render time, using for example the collision-aware subdivision of Bridson et al. [2002].

Dissipation and friction are important, complex topics deserving full publications of their own [Kaufman et al. 2005; Kaufman et al. 2008], and certainly more than the space allocated here. Our goal in this area was to provide some initial models that fit the method, and to demonstrate the controllability arising from a conservative foundation. Future work might explore efficient algorithms to handle stacking and static friction while still fitting the multisymplectic treatment.

Immediate and future impact In considering this method for immediate industrial use, we anticipate two important hurdles.

From the standpoint of incorporation into animation systems the first hurdle is the method's insistance on safety even at the cost of artistic freedom. This effectively disallows all pinching [Baraff et al. 2003; Volino and Magnenat-Thalmann 2006], as well as commencing from invalid configurations. We believe that the method can be extended to permit shallow ("skimming") pinching, but handling extremely unphysical boundary conditions within this framework seems at least initially at odds with the basic premise, and it will require further research.

Second, the proposed method is not competitive in performance compared to existing methods, which do not attempt to make strong safety and correctness guarantees; if an artist is willing to search for parameters that provide non-penetrating good-looking results, they may become impatient with the method proposed here.

From the standpoint of long-term, curiosity-driven research, however, this method is appealing not just in its formalism but also in terms of performance, since it lays out a formal asynchronous framework from which one can investigate parallelization, optimization, and even approximation techniques that preserve guarantees of safety, correctness, and progress. To aid such future investigation, source code for our intial C++ implementation, along with data files needed to generate the examples shown in this paper, are available online.

Acknowledgements We thank David Mooy for modeling the knots, and Igor Boshoer, Matt Kushner, Kori Valz for lighting and rendering. We are grateful for the valuable feedback provided by Miklós Bergou, Rony Goldenthal, Bernhard Thomaszewski, Max Wardetzky, and the anonymous reviewers. This work was supported in part by the NSF (MSPA Award No. IIS-05-28402, CSR Award No. CNS-06-14770, CAREER Award No. CCF-06-43268) and the Amazon Elastic Compute Cloud. The Columbia authors are supported in part by generous gifts from Adobe, ATI, Autodesk, mental images, NVIDIA, the Walt Disney Company, and Weta Digital.

References

AGARWAL, P., BASCH, J., GUIBAS, L. J., HERSHBERGER, J., AND ZHANG, L. 2002. Deformable free space tilings for kinetic collision detection. *Intl. J. Robotics Research 21*, 179–197.

AGARWAL, P., GUIBAS, L., NGUYEN, A., RUSSEL, D., AND ZHANG, L. 2004. Collision detection for deforming necklaces. *Computational Geometry: Theory and Applications 28*, 137–163.

AGARWAL, P. K., HAR-PELED, S., AND VARADARAJAN, K. R. 2005. Geometric approximation via coresets. In *Combinatorial and Computational Geometry*, J. E. Goodman, J. Pach, and E. Welzl, Eds. Cambridge University Press, New York, 1–30.

ASCHER, U. M., AND PETZOLD, L. R. 1998. *Computer methods for ordinary differential equations and differential-algebraic equations*. Society for Industrial and Applied Mathematics, Philadelphia.

BARAFF, D., WITKIN, A., AND KASS, M. 2003. Untangling cloth. *ACM Trans. Graph. 22*, 3, 862–870.

BARAFF, D. 1989. 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*, ACM, New York, NY, USA, 223–232.

BARAFF, D. 1994. Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH '94*, 23–34.

BASCH, J., GUIBAS, L. J., AND ZHANG, L. 1997. Proximity problems on moving points. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, 344–351.

BASCH, J., GUIBAS, L. J., AND HERSHBERGER, J. 1999. Data structures for mobile data. *Journal of Algorithms 31*, 1–28.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. In *SIGGRAPH '02*, 594–603.

BRIDSON, R., MARINO, S., AND FEDKIW, R. 2003. Simulation of clothing with folds and wrinkles. In *SCA '03*, 28–36.

BRILLIANTOV, N. V., AND PÖSCHEL, T. 2004. *Kinetic Theory of Granular Gases*. Oxford University Press, USA.

CELES, W. 1998. Efficient asynchronous evolution of physical simulations. In *SIBGRAPI '98: Proceedings of the International Symposium on Computer Graphics, Image Processing, and Vision*, IEEE Computer Society, Washington, DC, USA, 224.

CHOI, K.-J., AND KO, H.-S. 2005. Stable but responsive cloth. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, ACM, New York, NY, USA, 1.

CIRAK, F., AND WEST, M. 2005. Decomposition-based contact response (DCR) for explicit finite element dynamics. *Int'l Journal for Numerical Methods in Engineering 64*, 8, 1078–1110.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 31–36.

DEQUIDT, J., GRISONI, L., AND CHAILLOU, C. 2004. Asynchronous interactive physical simulation. Tech. Rep. RR-5338, INRIA.

ECK, C., JANUŠEK, J., AND KRBEC, M. 2005. *Unilateral contact problems: variational methods and existence theorems*. Chapman and Hall/CRC Press, Boca Raton.

ENGLISH, E., AND BRIDSON, R. 2008. Animating developable surfaces using nonconforming elements. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, ACM, New York, NY, USA, 1–5.

ERICKSON, J., GUIBAS, L. J., STOLFI, J., AND ZHANG, L. 1999. Separation-sensitive collision detection for convex objects. In *Proc. 10th ACM-SIAM Symp. Discrete Algorithms*, 102–111.

ERICSON, C. 2004. *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann, December.

FONG, W., DARVE, E., AND LEW, A. 2008. Stability of asynchronous variational integrators. *J. Comput. Phys. 227*, 18, 8367–8394.

GAO, J., GUIBAS, L., HERSHBERGER, J., ZHANG, L., AND ZHU, A. 2003. Discrete mobile centers. *Discrete and Computational Geometry 30*, 1, 45–65.

GAO, J., GUIBAS, L. J., AND NGUYEN, A. 2005. Distributed proximity maintenance in ad hoc mobile network. In *IEEE International Conference on Distributed Computing in Sensor System (DCOSS'05)*, 4–19.

GRINSPUN, E., HIRANI, A., DESBRUN, M., AND SCHRÖDER, P. 2003. Discrete Shells. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 62–67.

GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. 2003. Nonconvex rigid bodies with stacking. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, ACM, New York, NY, USA, 871–878.

GUIBAS, L., XIE, F., AND ZHANG, L. 2001. Kinetic collision detection: Algorithms and experiments. In *Proceedings of the International Conference on Robotics and Automation*, 2903–2910.

GUIBAS, L. J., XIE, F., AND ZHANG, L. 2001. Kinetic collision detection: Algorithms and experiments. In *ICRA*, 2903–2910.

GUIBAS, L., KARAVELES, M., AND RUSSEL, D. 2004. A computational framework for handling motion. In *Proceedings of teh Sixth Workshop on Algorithm Engineering and Experiments*, 129–141.

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

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

HAIRER, E., LUBICH, C., AND WANNER, G. 2002. *Geometric Numerical Integration: Structure-preserving Algorithms for Ordinary Differential Equations*. Springer.

HARMON, D., VOUGA, E., TAMSTORF, R., AND GRINSPUN, E. 2008. Robust Treatment of Simultaneous Collisions. *SIGGRAPH (ACM Transactions on Graphics) 27*, 3, 1–4.

JOHNSON, K. L. 2008. *Contact mechanics*. Cambridge University Press.

KAUFMAN, D. M., EDMUNDS, T., AND PAI, D. K. 2005. Fast frictional dynamics for rigid bodies. In *SIGGRAPH '05*, 946–956.

KAUFMAN, D. M., SUEDA, S., JAMES, D. L., AND PAI, D. K. 2008. Staggered projections for frictional contact in multibody systems. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers*, ACM, New York, NY, USA, 1–11.

KHAREVYCH, L., YANG, W., TONG, Y., KANSO, E., MARSDEN, J. E., SCHRÖDER, P., AND DESBRUN, M. 2006. Geometric, variational integrators for computer animation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 43–51.

KLOSOWSKI, J. T., HELD, M., MITCHELL, J. S. B., SOWIZRAL, H., AND ZIKAN, K. 1998. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics 4*, 1, 21–36.

KONEČNÝ, P., AND ZIKAN, K. 1997. Lower Bound of Distance in 3D. In *Proceedings of WSCG 1997*, vol. 3, 640–649.

KORNEEV, V., AND KISELEV, A. 2004. *Modern Microprocessors*. Charles River Media.

LEW, A., MARSDEN, J. E., ORTIZ, M., AND WEST, M. 2003. Asynchronous variational integrators. *Archive for Rational Mechanics And Analysis 167*, 85–146.

LUBACHEVSKY, B. 1991. How to simulate billiards and similar systems. *Journal of Computational Physics 94*, 2 (June), 255–283.

MARSDEN, J., PATRICK, G., AND SHKOLLER, S. 1998. Multisymplectic Geometry, Variational Integrators, and Nonlinear PDEs. *Communications in Mathematical Physics 199*, 2, 351–395.

MARSDEN, J., PEKARSKY, S., SHKOLLER, S., AND WEST, M. 2001. Variational methods, multisymplectic geometry and contiunuum mechanics. *Journal of Geometry and Physics 38*, 3–4 (June), 253–284.

MILENKOVIC, V. J., AND SCHMIDL, H. 2001. Optimization-based animation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 37–46.

MIRTICH, B., AND CANNY, J. 1995. Impulse-based dynamic simulation. In *WAFR: Proceedings of the workshop on Algorithmic foundations of robotics*, A. K. Peters, Ltd., Natick, MA, USA, 407–418.

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

PFEIFFER, F., AND GLOCKER, C., Eds. 2000. *Multibody Dynamics With Unilateral Contacts*. SpringerWienNewYork, ch. 2, 69–146.

PÖSCHEL, T., AND SCHWAGER, T. 2005. *Computational Granular Dynamics: Models and Algorithms*. Springer.

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

SCHWAGER, T., AND PÖSCHEL, T. 2007. Coefficient of restitution and linear dashpot model revisited. *Granular Matter 9*, 6 (November), 465–469.

SIFAKIS, E., MARINO, S., AND TERAN, J. 2008. Globally coupled collision handling using volume preserving impulses. In *2008 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 147–154.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 205–214.

THOMASZEWSKI, B., PABST, S., AND STRASSER, W. 2008. Asynchronous cloth simulation. In *Computer Graphics International*.

TRINKLE, D. S. J. 1996. 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.

VOLINO, P., AND MAGNENAT-THALMANN, N. 2006. Resolving surface collisions through intersection contour minimization. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, ACM, New York, NY, USA, 1154–1159.

VOUGA, E., HARMON, D., TAMSTORF, R., AND GRINSPUN, E. 2009. Discrete penalty layers admit multisymplectic integration. Tech. rep., Columbia University.

WELLER, R., AND ZACHMANN, G. 2006. Kinetic separation lists for continuous collision detection of deformable objects. In *Third Workshop in Virtual Reality Interactions and Physical Simulation (Vriphys)*.

WRIGGERS, P., AND LAURSEN, T. A. 2007. *Computational contact mechanics*, vol. no. 498 of *CISM courses and lectures*. Springer, Wien.

WRIGGERS, P., AND PANAGIOTOPOULOS, P., Eds. 1999. *New Developments in Contact Problems*. SpringerWienNewYork, ch. 1, 1–54.

ZHONG, G., AND MARSDEN, J. E. 1988. Lie-Poisson Hamilton-Jacobi theory and Lie-Poisson integrators. *Physics Letters A 133* (Nov.), 134–139.