Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2011)
A. Bargteil and M. van de Panne (Editors)

Asynchronous Integration with Phantom Meshes

David Harmon†, Qingnan Zhou, and Denis Zorin

New York University

Abstract

Asynchronous variational integration of layered contact models provides a framework for robust collision handling, correct physical behavior, and guaranteed eventual resolution of even the most difficult contact problems. Yet, even for low-contact scenarios, this approach is significantly slower compared to its less robust alternatives—often due to handling of stiff elastic forces in an explicit framework. We propose a method that retains the guarantees, but allows for variational implicit integration of some of the forces, while maintaining asynchronous integration needed for contact handling. Our method uses phantom meshes *for calculations with stiff forces, which are then coupled to the original mesh through constraints. We use the augmented discrete Lagrangian of the constrained system to derive a variational integrator with the desired conservation properties.*

1. Introduction

Providing theoretical guarantees for physically-based animation is emerging as an important goal in computer graphics. From fluids [MCP*09] to yarns [KJM08], this approach has yielded advancements in predictability and reliability. However, such guarantees come at a cost and often result in decreased performance.

Harmon *et. al.* [HVS*09] developed a contact model and simulation framework that put provable robustness against interpenetrations and correctness with respect to physical laws on equal footing with simulation progress. While not fast, progress guarantees consistent advancement of simulation time and thus the eventual resolution of even the most difficult contact problem.

In this paper, we propose a method that regains lost simulation speed, while retaining the three guarantees of Harmon *et. al.* [HVS*09]: robustness against interpenetrations, conservation of linear and angular momentum, and guaranteed progress.

Shortcomings of asynchronous variational integrators. To guarantee correctness of contact resolution, Harmon *et. al.* [HVS*09] used a model called *discrete penalty layers* (DPL). DPL requires a multi-rate or asynchronous integrator, and they propose to use asynchronous variational integrators (AVIs), which have a number of attractive conservation properties. Namely, they conserve momentum (both linear and angular) and exhibit no energy drift, even for exponentially long simulation times. AVIs are explicit integrators, and thus small timesteps are required for stable resolution of stiff, non-linear problems. Aside from the need to compute the forces frequently, these small timesteps have a second, more severe cost. For asynchronous simulation of a conceptually infinite number of discrete penalty layers, efficient tracking of a subset of active penalty forces is needed, which is achieved by kinetic data structures (KDS). However, high-frequency maintenance of these data structures due to small time steps can be prohibitively expensive.

The standard tool for eliminating severe constraints on the time step size is to use implicit integration; while *synchronous* implicit variational integrators are well-known, in general, combining asynchronous integration with implicit time-stepping is difficult unless forces affect completely disjoint degrees of freedom: a rare occurence.

We propose a method allowing selective variational implicit integration of forces, while maintaining asynchronous integration needed for contact handling. Our method is based on using *phantom meshes*, a separate mesh with identical connectivity used for calculations with stiff forces. We couple phantom meshes to the original meshes through constraints and derive a variational integrator from the resulting augmented Lagrangian. Decoupling stiff forces from all others allows timesteps that are several orders of magnitude larger. For simulations where KDS maintenance due to non-contact force updates dominates runtime, we are able to stably simulate problems significantly faster than explicit AVIs.

† dharmon@cs.nyu.edu

Copyright © 2011 by the Association for Computing Machinery, Inc.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions Dept, ACM Inc., fax +1 (212) 869-0481 or e-mail permissions@acm.org.
SCA 2011, Vancouver, British Columbia, Canada, August 5 – 7, 2011.
© 2011 ACM 978-1-4503-0923-3/11/0008 $10.00

2. Related works

In this paper, we focus on extending the work in Harmon *et. al.* [HVS*09]. The contact algorithm literature in graphics and mechanics is vast, so we refer to the references therein for related works in that domain. For a principled treatment of contact models from the mechanics point of view, we suggest Wriggers [WL07].

Integrators that obey the discrete analogs of physical conservation laws are well-studied, and usually referred to as *geometric integrators* [HLW02]. They come in many flavors, including symplectic, symmetric (time-reversible), and variational. For our contact model we use discrete penalty layers, which, as noted in Harmon *et. al.* [HVS*09], require a multi-rate integrator. In its most general form a multi-rate integrator is completely asynchronous, so we focus on this situation, with much of our development applicable to the subset of multi-rate integrators which have integer multiple timesteps. Our formulation, then, follows that of asynchronous variational integrators (AVI) [LMOW03].

A common way to increase timesteps is to use implicit integration [BW98]. In the symplectic / variational world, specific parameters of the Newmark and Runge-Kutta integrators yield implicit update rules while remaining geometric [KMO*00, Sur90], with the accompanying large step stability. A popular single-step geometric integrator is implicit midpoint, which Volino and Magnenat-Thalmann [VMT05] demonstrated as suitable for cloth simulation. We use implicit midpoint for all our examples, albeit as part of a larger asynchronous integrator. However, any implicit integrator with the desired conservation properties could be used.

There are alternatives to implicit integration, but none that we are aware of can be easily integrated into the AVI / DPL framework. Time adaptivity is an active research topic for dealing with simulations of varying stiffness modes, and can even be structured such that symplecticity is retained [Hai97]. However, time-dilation is performed through a global function of configuration, which is not only expensive in an asynchronous setting, but adapts the time of the entire system, rather than the local stiffnesses as we would prefer. Besides, this approach aims to completely resolve the stiffnesses at a fine detail through smaller timesteps. We would prefer larger timesteps that retain the overall behavior of stiff modes.

Another option within geometric integrators is the idea of force averaging [LR01]. Force averaging substeps stiff forces and then averages the resulting force to "fake" large timesteps. This works well in a synchronous simulation, but the local force stencils of an asynchronous simulation introduced instabilities in our tests. In particular, if stiff forces are integrated at large timesteps, then the large motions of neighboring weak forces cannot be resolved by the stiff force, even with an extreme number of subseps.

Constraints are well-studied in the context of variational integrators [LMO08], and can be naturally integrated in the asynchronous variational integration framework. Gates *et. al.* [GMH08] uses constraints to couple domains with common interfaces on which different timesteps are used.

The idea of coupling degrees of freedom through constraints is widely used in different contexts, primarily for directable simulation, starting from space-time constraints of Witkin and Kass [WK88], and for control of fluids, elastic solids and thin shells (*e.g.*, [MTPS04], [CBC*05] and [BMWG07]). Node replication and coupling is also common in domain decomposition methods [SBG04]. English and Bridson [EB08] utilize a similar concept to phantom meshes for simulating conforming elements undergoing collisions. Sifakis *et. al.* [SSIF07] use "virtual" particles to embed sample points into a meshless representation.

3. Asynchronous variational integrators

The core aspect of our approach is a formulation of asynchronous variational integrators for systems with constraints, allowing for implicit integration and its specialization to the case of systems with phantom meshes. In this section, we consider a family of AVIs based on a discrete stationary-action principle, which includes explicit and implicit integrators. We discuss the conditions that have to be satisfied for implicit asynchronous integrators to be computationally tractable, and how constraints are integrated into this AVI formulation. We demonstrate how a constrained system involving a phantom mesh can be used to make implicit integration practical.

3.1. General formulation

In our exposition we mostly follow Lew *et. al.* [LMOW03], with some changes in notation and a more general form of discretization of the potential energy in the Lagrangian, similar to the one commonly used to derive implicit *synchronous* integrators.

Notation and basic concepts. Let $\mathbf{q} = [\mathbf{q}_1, \ldots, \mathbf{q}_n]^T$ be the set of n vertices defining the configuration of our discrete system. $\mathbf{v}(t) = \dot{\mathbf{q}}(t)$ is the configurational velocity, and momentum is $\mathbf{p} = \mathbf{M}\mathbf{v}$, for a diagonal mass matrix $\mathbf{M}$. For the purposes of this derivation, we assume that the Lagrangian of the system is defined by

$$L(\mathbf{q}, \dot{\mathbf{q}}) = T(\dot{\mathbf{q}}) - V(\mathbf{q}),$$

where T is kinetic energy and V is potential energy [Lan70].

The derivation of variational integrators based on the discrete form of Hamilton's principle starts with the discrete *action* over a period of time $[t_0, t_f]$ defined as an approximation of the standard action,

$$S_d(\mathbf{q}) = \sum_{j=1}^{(t_f - t_0)/h} L_d(\mathbf{q}^{j-1}, \mathbf{q}^j, h)h \approx \int_{t_0}^{t_f} L(\mathbf{q}, \dot{\mathbf{q}})dt. \quad (1)$$

The choice of quadrature rule for the discrete Lagrangian L_d determines the timestepping scheme. As an example, we will use the trapezoidal rule:

$$L_d(\mathbf{q}^{j-1}, \mathbf{q}^j, h) = \frac{1}{2}\left[L(\mathbf{q}^{j-1}, \dot{\mathbf{q}}^{j-\frac{1}{2}}) + L(\mathbf{q}^j, \dot{\mathbf{q}}^{j-\frac{1}{2}})\right],$$

where $\dot{\mathbf{q}}^{j-\frac{1}{2}} = (\mathbf{q}^j - \mathbf{q}^{j-1})/h$. We differentiate the discrete action with respect to vertex positions at different moments

in time, giving the discrete Euler-Lagrange equations of our variational integrator,

$$\frac{\partial S_d(\mathbf{q})}{\partial \mathbf{q}^j} = 0.$$

For our example action (Eqn. 1), this yields the timestepping rules for the second-order leapfrog integrator:

$$\begin{aligned} \mathbf{q}^{j+1} &= \mathbf{q}^j + h\dot{\mathbf{q}}^{j+\frac{1}{2}} \\ \dot{\mathbf{q}}^{j+\frac{1}{2}} &= \dot{\mathbf{q}}^{j-\frac{1}{2}} - h\mathbf{M}^{-1}\nabla V(\mathbf{q}^j). \end{aligned}$$

Integrators derived in this manner exactly conserve discrete linear and angular momenta and approximately conserve the energy of the system over long run-times. See West [Wes03] for full derivations of common variational integrators and error analysis.

Asynchrony. We are interested in *asynchronous* variational integrators, for which vertices $\mathbf{q}_a$ can be timestepped independently of one another, with different sequences of clock *ticks*. For the action functional discretization, clock ticks have a simple meaning: the action is approximated by a quadrature (for AVIs, separate for each term in the Lagrangian) and the ticks define how the time axis is partitioned into integration intervals for each term.

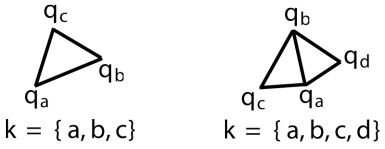


Figure 1: *Example stencils. (left) We define stretching forces across triangles, so the stencil is the* 3 *vertices of the triangle. (right) We use a hinge-based force for bending, where a stencil consists of the* 2 *edge vertices and* 2 *opposite vertices.*

The potential energy $V(\mathbf{q})$ is split based on a set of *stencils*, K, so that

$$V(\mathbf{q}) = \sum_{k \in K} V^k(\mathbf{q}_k).$$

$\mathbf{q}_k$ is the subset of $\mathbf{q}$ that completely determines a potential V^k (see Fig. 1 for example stencils). For each potential V^k we choose a timestep h^k that is small enough for stable simulation. For non-linear forces, however, we cannot know the stability criterion *a priori*, and thus we must choose a value that ensures stability for the entire simulation.

We define two types of *timelines* (sets of clock ticks): one per potential stencil,

$$T_k = \{t_k^i = ih^k \leq t_f; i \in \mathbb{N}\},$$

where t_f is the final simulation time, and one for each vertex,

$$T_a = \bigcup_{\{k \in K | a \in k\}} \mathrm{T}_k.$$

The ticks $t_a^j \in T_a$ are ordered, so that $t_a^j \leq t_a^{j+1}$. $\mathbf{q}_a^j = \mathbf{q}_a(t_a^j)$ is the position of vertex a at time t_a^j.

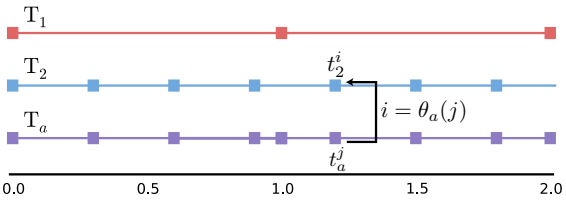


Figure 2: *Timelines T_1 and T_2 for two potentials, and the timeline T_a for a vertex contained in stencils of both. Note that while both T_1 and T_2 are evenly-spaced in time, the timeline of updates T_a for the vertex a is irregular in time.*

We define $\Delta t_a^j = t_a^{j+1} - t_a^j$. Let $\omega_a(j)$ be the unique stencil index such that $t_a^j \in T_{\omega_a(j)}$ (in other words, the tick t_a^j for a vertex a is due to the tick in the potential $V^{\omega_a(j)}$) and $\theta_a(j)$ is the index i of $t_k^i = ih^k$, for which $t_a^j = t_k^i$ (see Fig. 2).

Discrete action. To define our discrete action we assume that the motion is piecewise linear between successive ticks in $\mathrm{T_a}$ for every vertex $\mathbf{q}_a$. For the interval of time (t_a^j, t_a^{j+1}), we define the discrete velocity of a point $\mathbf{q}_a$ as the finite difference

$$\mathbf{v}_a^{j+\frac{1}{2}} \equiv (\mathbf{q}_a^{j+1} - \mathbf{q}_a^j)/\Delta t_a^j. \tag{2}$$

Following Lew *et. al.* [LMOW03], we split the *discrete action* into separate integrals of T and V. We approximate the kinetic energy *for each vertex* using a single-point quadrature over the time partition T_a, and the potential energy *for each stencil*, using a single-point quadrature for the time partition T_k:

$$S_d(\mathbf{q}, \dot{\mathbf{q}}) = \sum_a \sum_{j=0}^{|\mathrm{T}_a|-2} T_d(\dot{\mathbf{q}}) \Delta t_a^j - \sum_{k \in K} \sum_{j=0}^{|T_k|-1} V^k(\mathbf{q}) h^k.$$

For the velocities, we have a single choice for the value on the interval $[t_a^j, t_a^{j+1}]$, as we assume piecewise-linear trajectories, and velocities are piecewise-constant. For the potential energy term, we consider a family of rules based on interpolation on the endpoints of $[t_k^i, t_k^{i+1}]$, and evaluate at $t_k^{i,\alpha_k} = (1-\alpha_k)t_k^i + \alpha_k t_k^{i+1}$.

$$\begin{aligned} S_d(\mathbf{q}) = &\sum_a \sum_{j=0}^{|\mathrm{T}_a|-2} \frac{1}{2} m_a \|\mathbf{v}_a^{j+\frac{1}{2}}\|^2 \Delta t_a^j - \\ &\sum_{k \in K} \sum_{i=0}^{|\mathrm{T}_k|-1} V^k(\mathbf{q}(t_k^{i,\alpha_k})) h^k. \end{aligned} \tag{3}$$

where m_a is the mass associated with the a-th vertex. We can now write the discrete Euler-Lagrange (DEL) equations as

$$\frac{\partial S_d(\mathbf{q})}{\partial \mathbf{q}_a^j} = 0,$$

Computing these derivatives picks out exactly one poten-

tial term V^k from the sum for the potential energy, with $k = \omega_a(j)$, and yields the following time-stepping rule for momenta:

$$\mathbf{p}_a^{j+\frac{1}{2}} = \mathbf{p}_a^{j-\frac{1}{2}} - (1-\alpha_k)\frac{\partial V^k}{\partial \mathbf{q}_a^i}(\mathbf{q}(t_k^{i,\alpha_k}))h^k - \alpha_k \frac{\partial V^k}{\partial \mathbf{q}_a^i}(\mathbf{q}(t_k^{i-1,\alpha_k}))h^k$$

where $i = \theta_a(j)$. If we set α_k to 0, we obtain a purely explicit update for momenta: the right-hand side depends only on $\mathbf{q}(t_k^i) = \mathbf{q}(t_a^j)$, and the update formula for $\mathbf{q}_a$ directly follows from the discrete momentum definition (Eqn. 2).

Implicit integration. If α_k is not zero, the update rules become implicit. For synchronous integrators, Equation 3.1 immediately yields useful implicit variational integrators such as implicit midpoint ($\alpha_k = 1/2$) and the drift-kick modified Euler scheme ($\alpha_k = 1$).

The situation is far more complicated for asynchronous integrators. While formally we do get a system of equations for positions and momenta at all time ticks, it may be difficult or impossible to partition it into a computationally feasible sequential set of problems due to variable dependencies. Figure 3 shows the dependencies between different variables

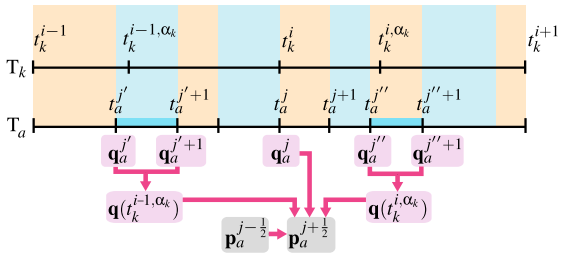


Figure 3: *Dependencies between position and momentum variables for an implicit asynchronous integrator.*

in the case of $\alpha_k \neq 0$. Suppose $k = \omega_a(j)$ is the potential triggered at t_a^j, and $t_a^j = t_k^i$. There are indices j' and j'', such that the potential evaluation points t_k^{i-1,α_k} and t_k^{i,α_k} are contained in intervals $(t_a^{j'}, t_a^{j'+1})$ and $(t_a^{j''}, t_a^{j''+1})$, respectively. Then the right-hand side of the update rule for $\mathbf{p}_a^{j+\frac{1}{2}}$ depends on $\mathbf{q}$ at the endpoints of these intervals, as shown in the figure. If there is another potential with a much smaller timestep with stencil containing a, there may be many ticks between j and j'', and $\mathbf{q}_a^{j''}$ cannot be computed without computing all intermediate values.

Observation: *For the implicit integrator (3.1) with $\alpha_k \neq 0$, the velocities at a time t_a^j may depend on positions $\mathbf{q}_a^{j''}$ for $j'' > j+1$. The difference $|j''-j|$ is determined by the ratio of sizes of the timesteps for the potential V^k activated at t_a^j, and the smallest time step for other potentials with stencils containing a.*

This observation means that in general (with no additional conditions on the timesteps for potentials with overlapping stencils) the implicit solve at a particular timestep may require solving simultaneously for interdependent positions and momenta at an arbitrarily large number of time steps, unless α_k is restricted so that $j'' \leq j+1$. In general, this can only be guaranteed if $\alpha_k = 0$.

A simple situation when an implicit integrator (3.1) is practical, is when stencils can be separated into sets K_i with no stencil overlaps for potentials from different sets, and for each set a single synchronous time step h^i is defined; then every vertex in the set K_i has the same time step. This situation, however, is of little interest—we want to have different clock rates for penalty forces, whose stencils inevitably overlap stencils for elastic forces. We can, however, reduce an important class of problems to this situation artificially, by replicating degrees of freedom and using constraints to maintain a coherent motion.

Integrators for problems with constraints. If a problem has m constraints $\phi(\mathbf{q}) = 0$, for some $\phi : \mathbb{R}^{3n} \to \mathbb{R}^m$, then the standard way of deriving the Lagrangian equations of motion is by replacing the action integral with the augmented action integral

$$\tilde{S} = S + \int_{t_0}^{t_f} \lambda^T \phi(\mathbf{q}) dt.$$

We observe that the new problem (finding a stationary point for the augmented action integral with respect to $\mathbf{q}(t)$ and $\lambda(t)$) is exactly the same mathematically as the original problem, with additional potential terms $\lambda^T\phi(\mathbf{q})$. For this reason, we can use *exactly the same* approach to discretization: we treat the Lagrange multiplier terms as an extra set of potentials which can be assigned a separate clock.

3.2. Implicit integration with phantom meshes

We specialize the integrator (3.1) to a simple case that allows for easy implicit integration of stiff forces. As we have observed, the main problem with implicit integration is due to the fact that the timeline T_a for a vertex is an overlay of timelines of different stencils, with potentially highly non-uniform timesteps.

To separate the stencils of forces with different timesteps, we replicate degrees of freedom to create a *phantom mesh*, and couple the original mesh (which we call *primary*) with the phantom mesh through constraints with relatively large time steps. More specifically, we classify all potentials into two sets: *implicitly integrated* K_{imp} and *explicitly integrated* K_{exp}; for brevity, we call the former implicit and the latter explicit.

The phantom mesh is an extra vector of degrees of freedom $\tilde{\mathbf{q}}$—one new vertex for each original vertex. Our choice of time steps and stencils is defined by the following set of conditions:

1. All *implicit* potentials are assigned stencils in the phantom mesh, and *the same* time step h^{imp}.
2. All *explicit* potentials are assigned stencils in the primary

mesh; timesteps can be chosen arbitrarily (within stability requirements).
3. The kinetic energy is split equally between the phantom and primary meshes.
4. We define constraints $\mathbf{q} - \tilde{\mathbf{q}} = 0$; the time step for the constraints is h^{imp}.

In the discrete action, we use $\alpha_k = 0$ for the explicit potentials, and $\alpha_k = 1/2$ for implicit, resulting in a midpoint implicit style integrator. Note that as all implicit potentials have the same time step, we can assume that their stencils are the whole phantom mesh, and combine these forces into a single potential $V^{\text{imp}}(\tilde{\mathbf{q}})$, with associated timeline $\mathrm{T}_{\text{imp}} = \{ih^{\text{imp}} \leq t_f : i \in \mathbb{N}\}$.

As constraint terms also share the same timestep, we consider them as a single potential $\lambda^T(\mathbf{q} - \tilde{\mathbf{q}})$, with λ of dimension $3n$, with stencil consisting of all vertices of both meshes. Because constraints share the timestep with implicit forces, the discrete Euler-Lagrange equation for the phantom mesh degrees of freedom has terms for both implicit potentials and constraints. We also note that the implicit timesteps T_{imp} is contained in all primary mesh vertex timelines T_a.

In summary, we use the following expression for the discrete action:

$$\begin{aligned} S_d(\mathbf{q}, \tilde{\mathbf{q}}) = & \\ & \sum_a \sum_{j=0}^{|\mathrm{T}_a|-2} \frac{1}{4} m_a \|\mathbf{v}_a^{j+\frac{1}{2}}\|^2 \Delta t_a^j - \sum_{k \in K_{\text{exp}}} \sum_{i=0}^{|\mathrm{T}_k|-2} V^k(\mathbf{q}(t_k^i)) h^k \\ & + \sum_a \sum_{j=0}^{|\mathrm{T}_{\text{imp}}|-2} \frac{1}{4} m_a \|\tilde{\mathbf{v}}_a^{j+\frac{1}{2}}\|^2 h^{\text{imp}} - \sum_{i=0}^{|\mathrm{T}_{\text{imp}}|-2} V^{\text{imp}}(\tilde{\mathbf{q}}(t_{\text{imp}}^{i,\alpha_k})) \\ & + \sum_{i=0}^{|\mathrm{T}_{\text{imp}}|-2} \sum_a \lambda_a(t_{\text{imp}}^i)^T (\mathbf{q}_a(t_{\text{imp}}^i) - \tilde{\mathbf{q}}_a(t_{\text{imp}}^i)) h^{\text{imp}} \end{aligned} \tag{4}$$

where the three lines correspond to explicit, implicit and constraint constituents (We use indices i for quantities related to implicit time steps, and j for aynchronous explicit.)

This yields the following update equations for momenta:

$$\mathbf{p}_a^{j+\frac{1}{2}} = \mathbf{p}_a^{j-\frac{1}{2}} - \frac{\partial V^{\omega_a(j)}}{\partial \mathbf{q}_a^j}(\mathbf{q}(t_a^j)) h^{\omega_a(j)} \tag{5}$$

$$\mathbf{p}_a^{j+\frac{1}{2}} = \mathbf{p}_a^{j-\frac{1}{2}} + \lambda_a^i h^{\text{imp}} \tag{6}$$

$$\begin{aligned} \tilde{\mathbf{p}}_a^{i+\frac{1}{2}} = & \ \tilde{\mathbf{p}}_a^{i-\frac{1}{2}} + \lambda_a^i h^{\text{imp}} - \\ & (1-\alpha) \frac{\partial V^{\text{imp}}}{\partial \mathbf{q}_a^i}(\tilde{\mathbf{q}}(t_{\text{imp}}^{i,\alpha})) h^{\text{imp}} - \alpha \frac{\partial V^{\text{imp}}}{\partial \mathbf{q}_a^i}(\tilde{\mathbf{q}}(t_{\text{imp}}^{i-1,\alpha})) h^{\text{imp}} \end{aligned} \tag{7}$$

and where $\lambda_a^i = \lambda_a((t_{\text{imp}}^i))$. The expression (5) is used for primary mesh vertices for non-constraint ticks, (6) is used for constraint ticks (assuming $t_a^j = t_{\text{imp}}^i$) and (7) for the phantom mesh vertices.

Handling constraints. The updates above require Lagrange multiplier values for constraints (in other words, virtual impulses $\lambda_a^i h^{\text{imp}}$). Enforcing constraints exactly would require solving for values of λ_a^i so that the constraint $\tilde{\mathbf{q}}_a(t_{\text{imp}}^i) = \mathbf{q}_a(t_{\text{imp}}^i)$ is satisfied for all i. Instead of solving for exact values of λ_a^i, we compute them approximately (similar, *e.g.*, to SHAKE [BKLS95]). We approximate the trajectory of a point $\mathbf{q}_a$, from $t_{\text{imp}}^i = t_a^j$ to t_{imp}^{i+1} by a linear trajectory with velocity $\mathbf{v}^{j+\frac{1}{2}}$, and determine λ_i^a from the condition

$$\tilde{\mathbf{q}}_a(t_{\text{imp}}^{i+1}) \approx \mathbf{q}_a^j + \mathbf{q}_a^{j+\frac{1}{2}} h^{\text{imp}} = \tilde{\mathbf{q}}_a^{i+1} = \tilde{\mathbf{q}}_a^i + \dot{\tilde{\mathbf{q}}}_a^{i+\frac{1}{2}} h^{\text{imp}} \tag{8}$$

The result of this approximation is that the phantom and primary meshes may not match exactly at any step; the more complicated the primary mesh trajectories between t_{imp}^i and t_{imp}^{i+1}, the greater the mismatch. However, this mismatch is continuously corrected, and as soon as penalty forces desist, the constraint will converge exactly (Eqn. 8 will no longer be an approximation, but exact). In the meantime, we get the desired behavior of the primary mesh behaving overall like an implicitly integrated stiff mesh.

Conservation. We observe that the proof of Lew *et. al.* [LMOW03] of conservation of linear and angular momenta applies *without changes* to our setting: the main property it relies on, aside fundamental translational and rotational symmetries of the discrete action clearly retained in (4), is that the variational integrator is derived from the discrete Euler-Lagrange equations, which is still the case in our setup. We emphasize that this fact is independent of the manner in which the Lagrange multipliers are computed: the approximation does not affect the conservation properties of the whole system, since the constraint impulses are momentum-conserving and behave essentially as another oscillating force in the system.

Figures 4 and 5 show experimental verification of the exact momentum conservation and energy behavior for our integrator.

4. Asynchronous integration algorithm

We implement the variational integrator of Section 3.2 in the kinetic data structure (KDS) framework of Harmon *et. al.* [HVS*09], extending the event priority queue framework described in Lew *et. al.* [LMOW03]. We summarize the algorithm of [HVS*09], and highlight our modifications.

Instead of advancing simulation one global time step at a time, we advance it by processing the events on the queue, in the order of times associated with these events. There are two main classes of events. *Force events* (including stretching, bending, gravity, and penalty forces for collisions) result in changes in velocities. The main purpose of the *certificate failure events* is to provide a mechanism for managing the conceptually infinite set of penalty potentials: they do not change velocities, but create and destroy penalty force events based on changes in vertex proximity.

Force events have associated stencils (vertices they update velocities for) and proximity certificate events have *supports*, consisting of vertices whose velocities determine the time of the event; when a velocity in the support changes, certificate event times need to be recomputed.

Force events. Each force event corresponds to a velocity update in the integrator. We integrate elastic forces *implicitly* and collision penalty forces *explicitly*. Although penalty forces can be very stiff, it is preferable to handle them explicitly for two reasons: (a) the stiffer the penalty force, the smaller its range and the shorter the time over which it needs to be integrated, and (b) correctly resolving collisions requires small time steps for these forces.

We use layered penalty potentials of Harmon *et. al.* [HVS*09] to prevent intersections. For any pair of primitives a and b, we define a sequence of increasingly stiff potentials $V^{r(\ell)}_{\eta(\ell)}$, $\ell = 1,2,\ldots$ with stiffnesses $r(\ell) = r(1)l^3$ which act at decreasing distances $\eta(\ell) = \eta(1)\ell^{-1/4}$:

$$V^r_\eta = \begin{cases} \frac{1}{2} r g(\mathbf{y}_a, \mathbf{y}_b) \text{ if } g(\mathbf{y}_a, \mathbf{y}_b) < 0 \\ 0, \text{ otherwise,} \end{cases}$$

where $\mathbf{y}_a$ and $\mathbf{y}_b$ are closest points on the primitives, and $g(\mathbf{y}_a, \mathbf{y}_b) = \|\mathbf{y}_a - \mathbf{y}_b\| - \eta$ is the gap function. Only a finite number of penalty potentials are active (*i.e.*, have corresponding events on the queue) at any given time. These events are automatically generated by slab events described below.

Compared to Harmon *et. al.* [HVS*09], where all updates are explicit, we introduce two new types of force events: implicit and constraint events, which are distinguished by the type of update rule used.

Separation slab certificate events. The simplest type of certificate failure event is a *slab event*. For a slab event at proximity level ℓ, the support is the six vertices of two triangles, and the time of the event is the time when the two triangles at current velocities of their vertices come too close (within $\eta(\ell+1)$).In this case, a new deeper penalty layer needs to be activated, which means creating a new force event for that layer and replacing the current slab event with a new slab event for the deeper layer.

Using only slab proximity events would be prohibitively expensive; in Harmon *et. al.* [HVS*09], k-Discrete Oriented Polytope (k-DOP) hierarchies are used with associated events corresponding to bounding volumes in the hierarchy separating or coming into proximity, with no slab events generated for primitives from separated volumes.

Hash grid kinetic data structure. The k-DOP data structure of Harmon *et. al.* [HVS*09] tends to produce an excessive number of certificate events for self-collisions (as observed by Barbič and James [BJ10]); we use a kinetic grid structure instead. Following the synchronous work of Teschner *et. al.* [THM*03], we do not store the whole grid; rather, we use a hash to store the set of active grid cells.

To kineticize this data structure, we create certificates to track the cells which contain triangle primitives. Precisely, each certificate declares *triangle T is in cell* (i,j,k). We must only compute its failure conservatively, not necessarily exactly. To this end, we simply compute the times each vertex of the triangle will cross the next cell boundary and schedule the event for the minimum of these. When a failure is popped off the queue and processed, we check if the grid data structure needs to be updated.

When two triangles share a cell, we create a corresponding separation slab. When two triangles no longer share a cell, we remove the slab event from the queue. In practice, this kinetic data structure is about 2-4x faster than the k-DOPs. Therefore, we use it for all examples in §5.

As with traditional, synchronous grids, choosing the optimal cell size is difficult, and poor choices can significantly affect performance. We achieve consistent performance using two times the average edge length in the mesh, although optimal cell size computation or a grid hierarchy style solution (*e.g.*, octrees) could prove beneficial.

Event interaction. Changes in velocities resulting from force events affect the times of certificate events; certificate events create new events and destroy existing ones. The interaction between events is determined by overlaps of their stencils and supports. We say that a certificate event is *contingent* on a force event if its support overlaps the stencil of the force event.

Initialization. For all potentials other than penalty potentials, events are created in the beginning with time $t = 0$; in addition, the triangles are placed into the hash grid and initial hash grid events, also with $t = 0$, are created for all (triangle, cell) pairs (T,C), such that T overlaps C. These initial hash grid events are responsible for creating slab events, which, in turn, create penalty force events.

```
1:  loop
2:    (E,t) ← Q.pop {pop event E from time-ordered queue Q}
3:    if E is an (explicit,implicit,constraint) force event with potential V^k then
4:      for a ∈ k do
5:        q_a ← q_a + (t − t_a)v_a {position update}
6:        t_a ← t {update vertex's clock}
7:      end for
8:      update v, using (5), (6), or (7)
9:      Q.push(E, t + h^k) {back to the queue, with new time}
10:     for j ∈ {contingent(i)|i ∈ k} do
11:       s ← failureTime(E_j) {compute new event times based on updated velocities }
12:       Q.update(E_j, s) {reschedule the contingent events}
13:     end for
14:   else if E is a slab event with depth ℓ then
15:     create penalty force event for depth ℓ
16:     create slab event for depth ℓ + 1
17:   else if E is a hash grid enter event for triangle T and cell C then
18:     create all missing slab depth 1 events for T and triangles of C
19:     schedule the next hash grid event for T
20:   else if E is a hash grid leave event for triangle T and cell C then
21:     destroy all slab depth 1 events for T and triangles of C, not needed by other cells
22:     schedule the next hash grid event for T
23:   end if
24: end loop
```

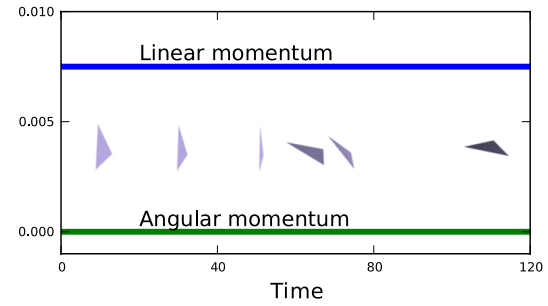


Figure 4: *We set up a sequence of triangles and give one an initial velocity so that they domino into one another. Because our derivation is completely variational, our timestepping respects conservation of linear and angular momentum.*

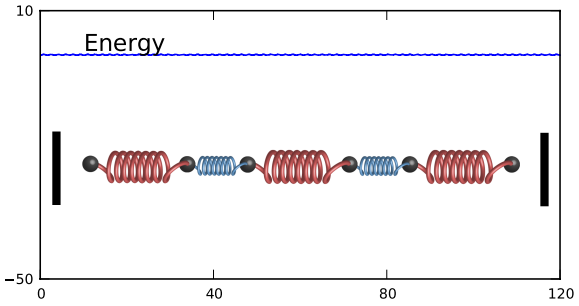


Figure 5: *Our method nearly conserves total energy for a long run-time of the adapted Fermi-Pasta-Ulam problem. Near energy conservation is an observed property of variational integrators.*

Implicit solver. We use non-linear conjugate gradient (CG) to solve for $\tilde{\mathbf{p}}^{i+\frac{1}{2}}$ at each step. For a preconditioner, we compute and factorize (Cholesky decomposition) the Hessian matrix only once per 50 timesteps for all examples, although this number is user-controlled. For prescribed vertices, we use the modified CG algorithm of Baraff and Witkin [BW98], adapted to the non-linear CG algorithm.

In-plane stretching forces are computed using Constant Strain Triangle [ZTZ05], and out-of-plane bending follows the Discrete Shells formulation [GHDS03]. We also combine gravity and wind forces into our implicit solve to improve convergence, rather than let them be transferred through constraint forces.

5. Results

We demonstrate our method on a variety of results, beginning with two didactic examples that illustrate important conservation properties of our method.

Figure 6: *Stiff thin shells are also challenging for explicit integrators. As this sequence progresses and collisions increase, contact forces (and rescheduling) dominate runtime rather than internal forces.*

5.1. Didactic

We line up a sequence of parallel triangles and one triangle perpendicular. The horizontal triangle has one vertex given an initial velocity of $(0,0,3/2)$. It quickly strikes the leading triangle, spurring a chain reaction of colliding triangles, transferring momentum down the line. Figure 4 shows the total angular and linear momentum. Throughout the sequence we are able to exactly preserve momentum, thanks to our structure-preserving geometric integrator.

We also analyze a variation of the common Fermi-Pasta-Ulam problem [HLW02]. The setup is a sequence of springs, alternating between stiff and weak coefficients. Additionally, we create an initial gap between each endpoint and a fixed wall, and we give several vertices an initial velocity to instigate motion. This example tests coupling of weak and stiff forces as well as interaction with the stiff sequence of discrete penalty layers. The key data we are after is the total energy of the system, plotted in Figure 5. Our method nearly preserves energy for the entire simulation, hovering around the initial energy $T+V$. This near-preservation is an experimentally observed property of variational integrators. While not exact, energy oscillates around the initial value for arbitrarily long runtimes. While our method has increased complexity with the implicit integration and the constraint enforcement, each step respects the derivation of a variational integrator, thus reaping their demonstrated benefits.

5.2. Examples

For practical scenarios we focused on situations where internal forces, rather than contact, dominate the runtime, yet still have an interesting collision environment. For the explicit runs, the largest stable timestep was used that gave the desired behavior, *e.g.*, stiff thin shells. For the implicit code, timesteps are limited by what is visually acceptable rather than stability, and that was our criteria for selecting timesteps. Table 1 lists a variety of pertinent data for each example.

Hat cascade. We drop a cascade of 9 hats onto a rigid pole to demonstrate stiff thin shells in contact. The hats are staggered to encourage tilting off the pole. Timesteps of 10^{-6}

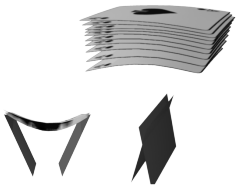
Scene	Vertices	Certificates	Explicit (%)	Penalties (%)	Stepsize (sec)	Frame (sec)	Implicit (%)	Penalties (%)	Frame (sec)	Speedup
Hats	973	4520	91.91	4.53	2.72e-6	123.24	77.89	20.81	38.19	3.22x
Cards	788	16439	51.39	48.97	2.22e-7	689.52	36.94	56.69	19.53	35.31x
Cape	5854	46384	83.57	15.89	4.41e-6	318.61	32.67	64.81	46.36	6.87x
Reef	10642	142928	14.17	83.10	4.61e-5	98.15	17.59	80.25	117.45	1.07x

Table 1: *For every scene we include the number of vertices, the average number of KDS certificates, percentage of time due to explicit forces, percentage due to penalty forces in an explicit simulation, the explicit simulation timestep, explicit simulation time per frame, percentage of time due to implicit forces, percentage due to penalty forces during implicit simulation, simulation time for implicit timestepping per frame, and speedup obtained by our method. For the implicit runs, an implicit timestep of* 0.001 *was used for all examples. All examples were run on a single core of a 3.6GHz i5.*

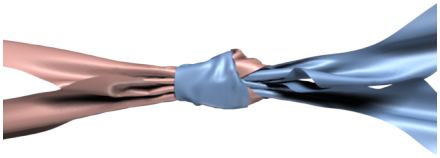
Figure 7: *The ogre character wears a cape in a strong wind. Resistance to excess strain is difficult for explicit integrators, however proves much easier for our implicit formulation.*

were required to sufficiently resolve the stiff forces in the explicit integrator, while our implicit method could take steps 10^{-3}. As the simulation continued, contact forces began to dominate the runtime, eating into the speedup offered by implicit integration. In contrast, up to the point where only two hats had landed, the overall speedup was over 16x.

Card riffle. We riffle a stack of extremely stiff cards. Our explicit integrator had a very difficult time with this scene, in sharp contract to the implicit method we propose. One particular effect we noticed was the relative insensitivity of the implicit solver to parameter changes. Explicit solvers, on the other hand, require careful parameter tuning to achieve stability.



Cape in wind. This example illustrates a problem that plagues explicit integrators: inextensible cloth. Because of the slow wave propagation, triangles can significantly stretch before forces have a chance to catch up, even just in the presence of gravity. Our character's cape is experiencing a strong wind. Explicit integrators require extremely small timesteps to maintain the cloth's integrity, or one of a variety of post-timestep inextensibility methods [GHF*07]. This remains an unexplored area of asynchronous integration, so we take the small timesteps required to enforce strain below 5%. The implicit solver is not limited in this way, and the speedup obtained reflects this.



Reef knot. We run the reef knot simulation from Harmon *et. al.* [HVS*09]. This simulation demonstrates the robustness of discrete penalty layers even in a seemingly impossible situation, and thus penalty force events, along with certificate reschedules, dominate the runtime for most of the simulation. Nevertheless, our implicit code manages to perform comparably. This is a far from ideal scene for our method, since internal force computations, whether explicit or implicit, are negligible, and thus there is little room for an overall speedup.

6. Discussion

The simple modifications of AVI for discrete penalty layers that we have presented already result in significant performance improvements. These improvements come from reducing the frequency of force integrations and, in particular, the reduction in KDS maintenance that follows. Therefore, potential runtime gains are limited by the total time spent in maintaining KDSs due to non-penalty force integration—simulations dominated by contact forces will not see as substantial an improvement (*e.g.*, the Reef Knot example).

On average we have increased the size of timesteps $2-3$ orders of magnitude. While we could take larger steps, simulations are still limited by the laws of physics. Large steps that involve contact can induce larger strains before the constraint impulses can correct them. This can result in a lag before stiff cloth behavior is seen. Thus, this expected behavior limits our timestep size, not stability.

Since implicit solvers for cloth simulation are an active research area, there are a variety of modifications left to be

explored, including different implicit discretizations, nonlinear solvers, preconditioners, and more. We chose a particular setup for all simulations and note that all research advancements in these domains can directly benefit our framework as well.

We observe that the general idea of first decoupling forces by introducing additional degrees of freedom, and then adding coupling back through constraints can be applied in more complicated scenarios to obtain asynchronous integrators: it is not strictly necessary to have a single clock and a global stencil for all implicit forces. At the same time, introducing more than one phantom mesh, even localized, is likely to have a performance penalty of its own. Exploring different ways of introducing implicit time stepping and adaptivity in an asynchronous variational integration context, while retaining desirable properties of these integrators is a promising direction for future work.

Acknowledgements

We thank Miklós Bergou for his valuable feedback and Rony Goldenthal for useful discussions about implicit integration with constraints. This work was supported in part by the NSF (awards DMS-0602235 and IIS-0905502) and Adobe Research. The first author is supported by a CRA Computing Innovation Fellowship.

References

[BJ10] BARBIČ J., JAMES D. L.: Subspace self-collision culling. *ACM Trans. Graph. 29* (July 2010), 81:1–81:9. 6

[BKLS95] BARTH E., KUCZERA K., LEIMKUHLER B., SKEEL R.: Algorithms for constrained molecular dynamics. *Journal of Computational Chemistry 16*, 10 (1995), 1192–1209. 5

[BMWG07] BERGOU M., MATHUR S., WARDETZKY M., GRINSPUN E.: TRACKS: Toward Directable Thin Shells. *SIGGRAPH (ACM Transactions on Graphics) 26*, 3 (Jul 2007), 50. 2

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *SIGGRAPH '98* (New York, NY, USA, 1998), pp. 43–54. 2, 7

[CBC*05] CAPELL S., BURKHART M., CURLESS B., DUCHAMP T., POPOVIĆ Z.: Physically based rigging for deformable characters. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), ACM, pp. 301–310. 2

[EB08] ENGLISH E., BRIDSON R.: Animating developable surfaces using nonconforming elements. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–5. 2

[GHDS03] GRINSPUN E., HIRANI A., DESBRUN M., SCHRÖDER P.: Discrete Shells. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Aug 2003), pp. 62–67. 7

[GHF*07] GOLDENTHAL R., HARMON D., FATTAL R., BERCOVIER M., GRINSPUN E.: Efficient Simulation of Inextensible Cloth. *SIGGRAPH (ACM Transactions on Graphics) 26*, 3 (2007). 8

[GMH08] GATES M., MATOUŠ K., HEATH M.: Asynchronous multi-domain variational integrators for non-linear problems. *International Journal for Numerical Methods in Engineering 76*, 9 (2008), 1353–1378. 2

[Hai97] HAIRER E.: Variable time step integration with symplectic methods. *Appl. Numer. Math. 25* (November 1997), 219–227. 2

[HLW02] HAIRER E., LUBICH C., WANNER G.: *Geometric Numerical Integration: Structure-preserving Algorithms for Ordinary Differential Equations*. Springer, 2002. 2, 7

[HVS*09] HARMON D., VOUGA E., SMITH B., TAMSTORF R., GRINSPUN E.: Asynchronous contact mechanics. *ACM Trans. Graph. 28* (2009), 87:1–87:12. 1, 2, 5, 6, 8

[KJM08] KALDOR J. M., JAMES D. L., MARSCHNER S.: Simulating knitted cloth at the yarn level. *ACM Trans. Graph. 27* (August 2008), 65:1–65:9. 1

[KMO*00] KANE C., MARSDEN J. E., ORTIZ M., , WEST M.: Variational integrators and the newmark algorithm for conservative and dissipative mechanical systems. *Int. J. Num. Math. Eng. 49* (2000), 1295–1325. 2

[Lan70] LANCZOS C.: *The variational principles of mechanics*, 4th ed. Dover Publications, New York, 1970. 2

[LMO08] LEYENDECKER S., MARSDEN J., ORTIZ M.: Variational integrators for constrained dynamical systems. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik 88*, 9 (2008), 677–708. 2

[LMOW03] LEW A., MARSDEN J. E., ORTIZ M., WEST M.: Asynchronous variational integrators. *Archive for Rational Mechanics And Analysis 167* (2003), 85–146. 2, 3, 5

[LR01] LEIMKUHLER B., REICH S.: A reversible averaging integrator for multiple time-scale dynamics. *J. Comput. Phys. 171* (July 2001), 95–114. 2

[MCP*09] MULLEN P., CRANE K., PAVLOV D., TONG Y., DESBRUN M.: Energy-preserving integrators for fluid animation. *ACM Trans. Graph. 28* (July 2009), 38:1–38:8. 1

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. In *ACM Transactions on Graphics (TOG)* (2004), vol. 23, ACM, pp. 449–456. 2

[SBG04] SMITH B., BJØRSTAD P., GROPP W.: *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. Cambridge Univ Pr, 2004. 2

[SSIF07] SIFAKIS E., SHINAR T., IRVING G., FEDKIW R.: Hybrid simulation of deformable solids. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 81–90. 2

[Sur90] SURIS Y.: Hamiltonian methods of Runge–Kutta type and their variational interpretation. *Mat. Model. 2* (1990), 78–87. 2

[THM*03] TESCHNER M., HEIDELBERGER B., MÜLLER M., POMERANETS D., GROSS M.: Optimized spatial hashing for collision detection of deformable objects. In *Proc. VMV* (2003), pp. 47–54. 6

[VMT05] VOLINO P., MAGNENAT-THALMANN N.: Implicit midpoint integration and adaptive damping for efficient cloth simulation. *Computer Animation and Virtual Worlds 16*, 3-4 (2005), 163–175. 2

[Wes03] WEST M.: *Variational Integrators*. PhD thesis, California Institute of Technology, 2003. 3

[WK88] WITKIN A., KASS M.: Spacetime constraints. In *ACM Siggraph Computer Graphics* (1988), vol. 22, ACM, pp. 159–168. 2

[WL07] WRIGGERS P., LAURSEN T. A.: *Computational contact mechanics*, vol. 498 of *CISM courses and lectures*. Springer, 2007. 2

[ZTZ05] ZIENKIEWICZ O., TAYLOR R., ZHU J.: *The Finite Element Method–Its Basis and Fundamentals, volume 1*. Butterworth-Heinemann,, 2005. 7