Proceedings of IMECE2008
ASME International Mechanical Engineering Congress and Exposition
November 1-6, 2008, Boston, Massachusetts, USA

IMECE2008-66766

DRAFT

A GPU-BASED IMPLEMENTATION OF A CONE CONVEX COMPLEMENTARITY APPROACH FOR SIMULATING RIGID BODY DYNAMICS WITH FRICTIONAL CONTACT

Alessandro Tasora
Dept. Mech. Engineering
University of Parma
Parma, Italy
Email: tasora@ied.unipr.it

Dan Negrut[*]
Dept. Mech. Engineering
University of Wisconsin, Madison, WI
Email: negrut@wisc.edu

Mihai Anitescu
Math. and Comp. Sci. Division
Argonne National Lab.
Argonne, IL
Email: anitescu@mcs.anl.gov

ABSTRACT

In the context of simulating the frictional contact dynamics of large systems of rigid bodies, this paper reviews a novel method for solving large cone complementarity problems by means of a fixed-point iteration algorithm. The method is an extension of the Gauss-Seidel and Gauss-Jacobi methods with overrelaxation for symmetric convex linear complementarity problems. Convergent under fairly standard assumptions, the method is implemented in a parallel framework by using a single instruction multiple data computation paradigm promoted by the Compute Unified Device Architecture library for graphical processing unit programming. The framework supports the analysis of problems with a large number of rigid bodies in contact. Simulation thus becomes a viable tool for investigating the dynamics of complex systems such as ground vehicles running on sand, powder composites, and granular material flow.

Introduction

Approximating the time evolution of a multibody system in the presence of friction and contact/impact phenomena through numerical simulation continues to be a challenging task. For instance, results reported in [1] indicate that the most widely used commercial software package for multibody dynamics simulation has significant difficulties in handling the simple problem of a collection of balls falling in a box, whenever the number of balls becomes larger than 50; in fact, the problem becomes practically intractable when the number of bodies becomes larger than 100. Presented here is an algorithm that can robustly and efficiently approximate the dynamics of *rigid* bodies undergoing frictional contact [2]. Posing challenges of its own, the case of *deformable* frictional contact is extensively discussed in [3,4] and falls outside the scope of this work.

Two approaches are most often considered when simulating the dynamics of a multibody system with frictional contact. First is the class of so-called penalty methods, where it is assumed that every time two rigid bodies come in frictional contact, the interaction can be represented by a collection of stiff springs combined with damping elements that act at the interface of the two bodies [5–8]. Implementing these regularization approaches requires little effort beyond that usually associated with developing a multibody dynamics simulation code. Furthermore, this methodology can easily accommodate complex frictional contact mechanisms, as it allows for a large number of "tuning" parameters that, in general, can be adjusted to control the dynamics of the frictional contact interaction. What has prevented the widespread use of this solution is the small step-size at which the

[*]Address all correspondence to this author.

numerical integration formula, because of stability limitations, is able to advance the simulation. This drawback is related to the stiff spring elements artificially included in the model. Most of the time, this step-size limitation is counterbalanced by the use of implicit integration formulas, a proposition that typically comes at a price as it requires the solution of a discretization nonlinear system at each integration time-step. This in turn leads to a heavy computational burden for scenarios with a large number of active frictional contact events.

A second approach, and the one pursued in this work, relies on a different mathematical framework capable of handling applications with hundreds of thousands of frictional contact events. The algorithms in this class draw on time-stepping procedures that produce weak solutions of the differential variational inequality (DVI) that describes the time evolution of rigid bodies with collision, contact, and friction. The DVI as a problem formulation was recently introduced in full generality and classified by differential index [9], though earlier numerical approaches based on DVI formulations do exist [10–12]. Recent work on time-stepping schemes has included both acceleration-force linear complementarity problem (LCP) approaches [13–15] and velocity-impulse LCP-based time-stepping methods [16–19]. The LCPs, obtained as a result of the introduction of inequalities in time-stepping schemes for DVI, coupled with a polyhedral approximation of the friction cone, must be solved at each time step in order to determine the system state configuration as well as the Lagrange multipliers associated with the frictional contact problem [11, 16]. If the simulation entails a large number of contacts and rigid bodies, as is the case of part feeders, packaging machines, and granular flows, the computational burden of classical LCP solvers can become significant. Indeed, a well-known class of numerical solutions for LCPs is based on *simplex methods*, also known as *direct* or *pivoting* methods [20]; however, these methods may exhibit exponential worst-case complexity [21]. They may be impractical even for problems involving as little as a few hundred bodies when friction is present [22,23]. Further complicating the numerical solution, since the three-dimensional Coulomb friction case leads to a nonlinear complementarity problem (NCP), the use of a polyhedral approximation to morph the NCP into an LCP introduces artificial anisotropy, which affects friction because friction cones become faceted friction pyramids [15–17]. This discrete and finite approximation of friction cones is one of the reasons for the large dimension of the problem that needs to be solved in multibody dynamics with frictional contact.

In order to circumvent the limitations imposed by the use of classical LCP solvers and the limited accuracy associated with polyhedral approximations of the friction cone, a parallel fixed-point iteration method with projection on a convex set is proposed, which can directly solve large cone complementarity problems with low computational overhead. The method is based on a time-stepping formulation that solves at every step a cone constrained optimization problem [24]. The time-stepping scheme has been proved to converge in a measure differential inclusion sense to the solution of the original continuous-time DVI. For the proposed approach, about 80% of the computational effort in simulating frictional contact dynamics is spent solving the cone complementarity problem (CCP). The goal of this work is to solve the CCP in parallel by using commodity high-performance computing hardware. Specifically, a methodology is proposed that hinges on the use of parallel computational resources available on NVIDIA's graphical processing unit (GPU) cards, which can currently handle 12,228 live computational threads simultaneously on the GeForce 8800 series. Tapping into this massively parallel computational resource has been facilitated by NVIDIA's sharing of a well-integrated application programming interface supported by the Compute Unified Device Architecture (CUDA) library [25].

Formulation of the Multibody Dynamics with Frictional Contact Problem

The equations that govern the time evolution of a multibody system can be expressed in the form (see, for instance, [26])

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{L}(\mathbf{q})\mathbf{v} \\ \mathbf{M}\dot{\mathbf{v}} &= \mathbf{f}^A(t,\mathbf{q},\mathbf{v}), \end{aligned} \tag{1}$$

where $\mathbf{q} = \left[\mathbf{r}_1^T, \varepsilon_1^T, \ldots, \mathbf{r}_{n_b}^T, \varepsilon_{n_b}^T\right]^T \in \mathbb{R}^{6n_b}$ are generalized positions, $\mathbf{v} = \left[\dot{\mathbf{r}}_1^T, \bar{\omega}_1^T, \ldots, \dot{\mathbf{r}}_{n_b}^T, \bar{\omega}_{n_b}^T\right]^T \in \mathbb{R}^{6n_b}$ are generalized velocities, and n_b represents the number of bodies in the system. The matrix $\mathbf{M}$ is the generalized mass matrix, and $\mathbf{f}^A(t,\mathbf{q},\mathbf{v})$ represents the vector of generalized applied forces. The convention used here is that any symbol in bold represents a vector or matrix quantity, and an overbar represents a vector quantity represented in the local, body-fixed reference frame associated with a body that is inferred from the context.

The formulation of the equations of motion draws on the so-called absolute, or Cartesian, representation of the attitude of each rigid body in the system. For each body j, its orientation is described by a set of three Euler angles, $\varepsilon_j \in \mathbb{R}^3$, following the 3-1-3 local rotation sequence (see, for instance, [26]). The rate at which each body changes its orientation is captured by the local angular velocity $\bar{\omega}_j \in \mathbb{R}^3$. The location of each body is uniquely determined by a position vector $\mathbf{r}_j = [x_j, y_j, z_j]^T$ that specifies where the body-fixed centroidal reference frame is located. The translational velocity of the body is simply $\dot{\mathbf{r}}_j$, where an overdot represents time differentiation. With this set of generalized coordinates, the mass matrix $\mathbf{M}$ remains constant and diagonal between any realigning of a body-fixed centroidal reference frame, which can potentially be employed to avoid Euler angles singularities. Also note that, since for each body j there is a locally nonsingular matrix $\mathbf{B}(\varepsilon_j)$ such that $\bar{\omega}_j = \mathbf{B}(\varepsilon_j)\dot{\varepsilon}_j$, the operator $\mathbf{L}(\mathbf{q})$ that relates the time derivative of the level-zero generalized coordinates to the level-one generalized coordinates is generally not the identity matrix. Note that no bilateral constraints are present in the current formulation. This case is discussed in [2,27], and a paper presenting a parallel methodology for the general case of bilateral and unilateral constraints is forthcoming.

Two rigid bodies should not penetrate, and, if they are in contact, there should be friction acting at the interface. In order to enforce the nonpenetration constraint, a gap function $\Phi(\mathbf{q}) \in \mathbb{R}$ is assumed to exist and satisfy

$$\Phi(\mathbf{q}) = \begin{cases} > 0 & \text{if the bodies are separated,} \\ = 0 & \text{if the bodies touch each other,} \\ < 0 & \text{if the bodies are interpenetrating.} \end{cases} \qquad (2)$$

For such a function, the nonpenetration constraint becomes $\Phi(q) \geq 0$. An example of such a mapping is the signed distance function [28], which, when the bodies are smooth and convex, is differentiable at least up to some value of the interpenetration [29]. For most cases, even simple ones involving the relative position of two spheres, a differentiable signed distance function cannot be defined for certain configurations $\mathbf{q}$. The fact that $\Phi(\mathbf{q})$ can be differentiably defined only on a neighborhood of the set $\Phi(\mathbf{q}) \geq 0$ can be accommodated at the cost of making the analysis substantially more involved [30]. This approach will not be used here. In addition, for piecewise smooth bodies, the signed distance function, which is usually the first choice of a gap function, is nonsmooth even when the bodies are not penetrating each other [31]. For polyhedral bodies, this difficulty can be circumvented by writing the gap function as the maximum between basic contact configurations gap functions. In three dimensions, such configurations are corner-on-face and nonparallel edge-on-edge. The nonpenetration constraint can be handled, in the context of the time-stepping scheme (6–9), by the appropriate definition of the active set A to include not only active contacts but also active basic contact configuration gap functions [31]. For sufficiently small penetration, without loss of generality, one can make a *differentiability of geometrical constraint assumption:* Any contact is described by a gap function $\Phi(\mathbf{q})$ that is twice continuously differentiable. For an overwhelming majority of applications, when one deals with convex geometries and with suitably small numerical integration step-sizes, this assumption is easily verified.

The friction model used here is the Coulomb model, which leads to frictional conic constraints regarded as an extension of complementarity models discussed in [16,17]. If the configuration of the system $\mathbf{q}$ is such that a contact i is active, that is, $\Phi_i(\mathbf{q}) = 0$, then a normal force and a tangential force are going to act on each of the two bodies at the contact point. Denoting A and B the two bodies in contact, let $\mathbf{n}_i$ be the normal vector at the contact pointing toward the exterior of the first body, that is, body A. Let $\mathbf{u}_i$ and $\mathbf{w}_i$ be two vectors in the contact plane such that $\mathbf{n}_i, \mathbf{u}_i, \mathbf{w}_i \in \mathbb{R}^3$ are mutually orthonormal vectors. Although they typically depend on $\mathbf{q}$, this dependency is not explicitly indicated, in order to keep the notation simple.

The frictional contact force is impressed on the system by means of multipliers $\widehat{\gamma}_{i,n} \geq 0$, $\widehat{\gamma}_{i,u}$, and $\widehat{\gamma}_{i,w}$, which lead to the normal component of the force $\mathbf{F}_{i,N} = \widehat{\gamma}_{i,n}\mathbf{n}_i$ and the tangential component of the force $\mathbf{F}_{i,T} = \widehat{\gamma}_{i,u}\mathbf{u}_i + \widehat{\gamma}_{i,w}\mathbf{w}_i$.

The Coulomb model consists of the following constraints:

$$\begin{gathered} \widehat{\gamma}_{i,n} \geq 0, \quad \Phi_i(\mathbf{q}) \geq 0, \quad \Phi_i(\mathbf{q})\widehat{\gamma}_{i,n} = 0, \\ \mu_i\widehat{\gamma}_{i,n} \geq \sqrt{\widehat{\gamma}_{i,u}^2 + \widehat{\gamma}_{i,w}^2}\ , \quad ||\mathbf{v}_{i,T}|| \left(\mu_i\widehat{\gamma}_{i,n} - \sqrt{\widehat{\gamma}_{i,u}^2 + \widehat{\gamma}_{i,w}^2}\right) = 0, \\ \langle \mathbf{F}_{i,T}, \mathbf{v}_{i,T} \rangle = -||\mathbf{F}_{i,T}||\ ||\mathbf{v}_{i,T}|| \end{gathered}$$

where $\mathbf{v}_{i,T}$ is the relative tangential velocity at contact i. The magnitude of the friction force depends on the friction coefficient $\mu_i \in \mathbb{R}^+$, which typically has a value between 0 and 1 for most materials, and is instrumental in linking the magnitude of the tangential and normal forces through a constitutive type equation [1].

The first part of the constraint can be restated as

$$\mathbf{F}_i = \mathbf{F}_{i,N} + \mathbf{F}_{i,T} = \widehat{\gamma}_{i,n}\mathbf{n}_i + \widehat{\gamma}_{i,u}\mathbf{u}_i + \widehat{\gamma}_{i,w}\mathbf{w}_i \in C, \qquad (3)$$

where C is a cone in three dimensions, whose slope is $\tan^{-1}\mu_i$. If one defines by $\langle\ ,\ \rangle$ the inner product of two vectors, the constraint $\langle \mathbf{F}_{i,T}, \mathbf{v}_{i,T} \rangle = -||\mathbf{F}_{i,T}||\ ||\mathbf{v}_{i,T}||$ requires that the tangential force be opposite to the tangential velocity. This results in the friction force being dissipative. In fact, an equivalent convenient way of expressing this constraint is by using the maximum dissipation principle $(\widehat{\gamma}_{i,u}, \widehat{\gamma}_{i,w}) = \underset{\sqrt{\widehat{\gamma}_{i,u}^2 + \widehat{\gamma}_{i,w}^2} \leq \mu_i\widehat{\gamma}_{i,n}}{\operatorname{argmin}} \mathbf{v}_{i,T}^T \left(\widehat{\gamma}_{i,u}\mathbf{u}_i + \widehat{\gamma}_{i,w}\mathbf{w}_i\right)$ [12,32]. For this minimization problem, it is relatively straightforward to establish a connection between the first-order necessary KKT conditions [33] and the Coulomb model above. Effectively, the condition in this equation states that the friction force is such that, given a tangential velocity and a normal force, the power dissipated is maximized.

The contribution of the frictional contact forces in the equations of motion, Eq. (1), is through a set of generalized forces associated with each active contact in the model. Based on Newton's third law, each body experiences a force of the same magnitude but opposite direction at the point of contact. Therefore,

[1]Though the original Coulomb model distinguishes between static μ_s and kinetic μ_k friction coefficients, where usually the kinetic coefficient is slightly lower than its static counterpart, in this work both are considered to have the same value μ. The difference is not relevant for the discussion; it suffices to say that to correct this approach would require one to adjust the friction coefficient adaptively during the simulation depending on the slipping speed, so as to express complex nonlinearities in μ as a function of speed.

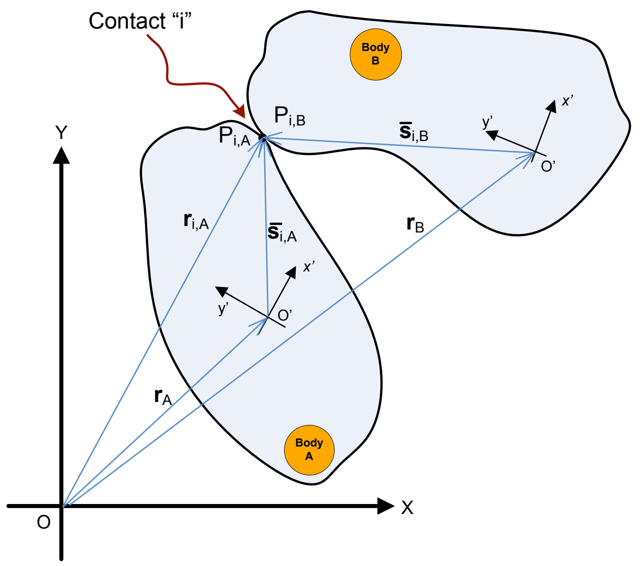

Figure 1. Contact i active between two bodies $A, B \in \{1, 2, \ldots, n_b\}$

the virtual work associated with the frictional contact force $\mathbf{F}_i$ between bodies A and B becomes $\delta W_i = \delta\mathbf{r}_{i,A}^T \mathbf{F}_i - \delta\mathbf{r}_{i,B}^T \mathbf{F}_i$. As illustrated in Fig. 1, $\mathbf{r}_{i,A} = \mathbf{r}_j + \mathbf{A}_A \bar{\mathbf{s}}_{i,j}$ gives the position, expressed in the global inertial reference frame, of the contact point $P_{i,A}$ on body A, and $\delta\mathbf{r}_{i,A} = \delta\mathbf{r}_A + \mathbf{A}_A \delta\tilde{\bar{\pi}}_A \bar{\mathbf{s}}_{i,A} = \delta\mathbf{r}_A - \mathbf{A}_A \tilde{\bar{\mathbf{s}}}_{i,A} \delta\bar{\pi}_A$ represents a virtual displacement of body A, which is due to a virtual translational displacement of the body center of mass, $\delta\mathbf{r}_A$, and a virtual rotation $\delta\bar{\pi}_A$, expressed in the local body A reference frame. Similar quantities are defined in conjunction with body B. Note that the operator $\tilde{}$ acting on a vector $\mathbf{h} = [h_1, h_2, h_3]^T$ produces a skew symmetric matrix $\tilde{\mathbf{h}} \equiv \mathbf{H} \in \mathbb{R}^{3\times3}$ with $\mathbf{H}(1,2) = -h_3$, $\mathbf{H}(1,3) = h_2$, and $\mathbf{H}(2,3) = -h_1$. From Eq. (3),

$$\begin{aligned}
\delta W_i &= (\delta\mathbf{r}_A^T + \delta\bar{\pi}_A^T \tilde{\bar{\mathbf{s}}}_{i,A} \mathbf{A}_A^T)(\widehat{\gamma}_{i,n}\mathbf{n}_i + \widehat{\gamma}_{i,u}\mathbf{u}_i + \widehat{\gamma}_{i,w}\mathbf{w}_i) \\
&- (\delta\mathbf{r}_B^T + \delta\bar{\pi}_B{}^T \tilde{\bar{\mathbf{s}}}_{i,B} \mathbf{A}_B^T)(\widehat{\gamma}_{i,n}\mathbf{n}_i + \widehat{\gamma}_{i,u}\mathbf{u}_i + \widehat{\gamma}_{i,w}\mathbf{w}_i) \\
&= \delta\mathbf{q}^T \ \mathbf{D}_i^* \ (\widehat{\gamma}_{i,n}\mathbf{n}_i + \widehat{\gamma}_{i,u}\mathbf{u}_i + \widehat{\gamma}_{i,w}\mathbf{w}_i) \\
&= \delta\mathbf{q}^T \ (\widehat{\gamma}_{i,n}\mathbf{D}_{i,n} + \widehat{\gamma}_{i,u}\mathbf{D}_{i,u} + \widehat{\gamma}_{i,w}\mathbf{D}_{i,w}),
\end{aligned}$$

where, with $\mathbf{I}_3$ the 3×3 identity matrix, the projection matrix $\mathbf{D}_i^* \in \mathbb{R}^{6n_b\times3}$ is defined for contact i as

$$\mathbf{D}_i^{*T} = \left[\mathbf{0} \ldots \mathbf{I}_3 \ (\tilde{\bar{\mathbf{s}}}_{i,A}\mathbf{A}_A^T)^T \ \ \mathbf{0} \ldots \mathbf{0} \ \ -\mathbf{I}_3 \ -(\tilde{\bar{\mathbf{s}}}_{i,B}\mathbf{A}_B^T)^T \ \ldots \mathbf{0}\right],$$

and $\mathbf{D}_{i,n} \equiv \mathbf{D}_i^*\mathbf{n}_i$, $\mathbf{D}_{i,u} \equiv \mathbf{D}_i^*\mathbf{u}_i$, and $\mathbf{D}_{i,w} \equiv \mathbf{D}_i^*\mathbf{w}_i$.

These three vectors can be grouped in a matrix $\mathbf{D}_i = [\mathbf{D}_{i,n}, \mathbf{D}_{i,u}, \mathbf{D}_{i,w}] \in \mathbb{R}^{6n_b\times3}$. Denoting $\mathbf{A}_{i,p} = [\mathbf{n}_i, \mathbf{u}_i, \mathbf{w}_i]$ the $\mathbb{R}^{3\times3}$ matrix of the local coordinates of the ith contact, one can express $\mathbf{D}_i$ also as

$$\mathbf{D}_i^T = \left[\mathbf{0} \ldots -\mathbf{A}_{i,p}^T \ \mathbf{A}_{i,p}^T\mathbf{A}_A\tilde{\bar{\mathbf{s}}}_{i,A} \ \ \mathbf{0} \ldots \mathbf{0} \ \ \mathbf{A}_{i,p}^T \ -\mathbf{A}_{i,p}^T\mathbf{A}_B\tilde{\bar{\mathbf{s}}}_{i,B} \ \ldots \mathbf{0}\right]. \tag{4}$$

Note that the velocity at the point of contact can also be expressed in terms of $\mathbf{D}_{i,u}$ and $\mathbf{D}_{i,w}$. To this end, the velocity in local contact coordinates can be expressed as $\mathbf{v}_{i,T}^T = \mathbf{v}^T \mathbf{D}_i$, and therefore the power dissipated can be equivalently expressed as $\mathbf{v}_{i,T}^T (\widehat{\gamma}_{i,u}\mathbf{u}_i + \widehat{\gamma}_{i,w}\mathbf{w}_i) = \mathbf{v}^T \mathbf{D}_i (\widehat{\gamma}_{i,u}\mathbf{u}_i + \widehat{\gamma}_{i,w}\mathbf{w}_i) = \mathbf{v}^T (\widehat{\gamma}_{i,u}\mathbf{D}_{i,u} + \widehat{\gamma}_{i,w}\mathbf{D}_{i,w})$.

When one revisits Eq. (1) and assumes a set of p active constraints at time t, a more specific expression can be provided for the differential equations governing the time evolution of the multibody system by singling out the contribution of the frictional contact force. Drawing on the Coulomb model discussed, the following differential variational inequality is associated with the time evolution of the multibody system [34]:

$$\begin{aligned}
\dot{\mathbf{q}} &= \mathbf{L}(\mathbf{q})\mathbf{v} \\
\mathbf{M}\dot{\mathbf{v}} &= \mathbf{f}(t,\mathbf{q},\mathbf{v}) + \sum_{i=1}^{p} (\widehat{\gamma}_{i,n}\mathbf{D}_{i,n} + \widehat{\gamma}_{i,u}\mathbf{D}_{i,u} + \widehat{\gamma}_{i,w}\mathbf{D}_{i,w}) \\
1 \le i \le p \ &: \ \widehat{\gamma}_{i,n} \ge 0 \perp \Phi_i(\mathbf{q}) \ge 0, \qquad \text{and} \\
(\widehat{\gamma}_{i,u}, \widehat{\gamma}_{i,w}) &= \underset{\mu_i\widehat{\gamma}_{i,n} \ge \sqrt{(\widehat{\gamma}_{i,u})^2 + (\widehat{\gamma}_{i,w})^2}}{\operatorname{argmin}} \mathbf{v}^T (\widehat{\gamma}_{i,u}\mathbf{D}_{i,u} + \widehat{\gamma}_{i,w}\mathbf{D}_{i,w}).
\end{aligned} \tag{5}$$

Herein $u \perp v$ denotes that $u^T v = 0$. The Coulomb model used in this work is the predominant model used in the engineering literature to describe dry friction. Unfortunately, the model may be inconsistent: there exist configurations for which the model does not have a solution [13, 19]. This situation has led to the need to explore weaker formulations where the forces are measures and Newton's law is satisfied in a measure differential inclusion sense [19]. It has been shown that solutions in that sense do exist and can be found by time-stepping schemes [35]. Note that, for the type of application targeted — pebble-bed nuclear reactor simulation, granular flow dynamics, and so forth, — all collisions that appear during the simulation can be assumed of the inelastic type.

Proposed Solution

Starting from a time-step $t^{(l)}$ with position $\mathbf{q}^{(l)}$ and velocity $\mathbf{v}^{(l)}$, one finds the numerical solution at the new time-step $t^{(l+1)} = t^{(l)} + h$ by solving the following optimization problem with equilibrium constraints:

$$\mathbf{M}(\mathbf{v}^{(l+1)} - \mathbf{v}^{l}) = h\mathbf{f}(t^{(l)}, \mathbf{q}^{(l)}, \mathbf{v}^{(l)}) + \sum_{i \in A(q^{(l)},\delta)} (\gamma_{i,n}\mathbf{D}_{i,n} + \gamma_{i,u}\mathbf{D}_{i,u} + \gamma_{i,w}\mathbf{D}_{i,w}) \quad (6)$$

$$i \in A(q^{(l)},\delta) \ : \ 0 \le \frac{1}{h}\Phi_i(\mathbf{q}^{(l)}) + \mathbf{D}_{i,n}^{T}\mathbf{v}^{(l+1)} \perp \gamma_n^i \ge 0, \text{ and} \quad (7)$$

$$(\gamma_{i,u}, \gamma_{i,w}) = \underset{\mu_i\gamma_{i,n} \ge \sqrt{\gamma_{i,u}^2 + \gamma_{i,w}^2}}{\operatorname{argmin}} \ \mathbf{v}^{T}(\gamma_{i,u}\mathbf{D}_{i,u} + \gamma_{i,w}\mathbf{D}_{i,w}), \quad (8)$$

$$\mathbf{q}^{(l+1)} = \mathbf{q}^{(l)} + h\mathbf{L}(\mathbf{q}^{(l)})\mathbf{v}^{(l+1)}. \quad (9)$$

Here, for a conveniently chosen small value of $\delta > 0$,

$$A(\mathbf{q},\delta) = \{ i \,|\, i \in \{1,2,\ldots,p\}, \ \Phi_i(\mathbf{q}) \le \delta \},$$

and γ_s represents the constraint impulse of a contact constraint, that is, $\gamma_s = h\widehat{\gamma}_s$, for $s = n,u,w$. The $\frac{1}{h}\Phi_i(\mathbf{q}^{(l)})$ term achieves constraint stabilization. Its effect is discussed in detail in [30]. As the step size $h \to 0$, the scheme converges to the solution of a measure differential inclusion [24]. Numerical solutions for the case when the nonlinear frictional contact constraint is approximated by a piecewise linear cone can be found by Lemke's algorithm [17]. Nonetheless, as the number of constraints in the problem increases, the computational cost of Lemke's method increases far faster than linearly with the size of the problem [22]. Alternatively, the problem is cast as a monotone optimization problem by introducing a relaxation over the complementarity constraints; that is, the time-stepping scheme is modified by replacing Eq. (7) with

$$i \in A(q^{(l)},\delta) : 0 \le \frac{1}{h}\Phi_i(\mathbf{q}^{(l)}) + \mathbf{D}_{i,n}^{T}\mathbf{v}^{(l+1)} - \mu_i\sqrt{(\mathbf{v}^{T}\mathbf{D}_{i,u})^2 + (\mathbf{v}^{T}\mathbf{D}_{i,w})^2} \perp \gamma_n^i \ge 0 \, . \quad (10)$$

As $h \to 0$, the solution of the modified time-stepping scheme will approach the solution of the same measure differential inclusion as the original scheme [24]. It can be immediately verified that, for one step, the solution of the scheme using (10) approaches the one of the scheme using (7) when $\mu_i\gamma_{i,n}\sqrt{(\mathbf{v}^{T}\mathbf{D}_{i,u})^2 + (\mathbf{v}^{T}\mathbf{D}_{i,w})^2} \ll 1$ [22]. This assumption is satisfied both in the regime in which pebble-bed nuclear reactors operate [36] and for granular flow applications, two classes of applications targeted by the proposed approach. Generally, the above assumption will be satisfied whenever there is little friction between the two bodies in contact (low μ) or when the relative velocity at the contact point is small. However, the sequence produced by the scheme using the relaxation (10) approaches the one produced by the scheme using (7) in far more general circumstances. Provided that the time step is small compared to the characteristic time scale of the tangential velocity, the dynamics of the relaxed scheme will approach the dynamics of the original scheme [24].

The KKT optimality conditions for the equilibrium constraint in Eq. (8) state that, for any $i \in A(q^{(l)},\delta)$, there exists a Lagrange multiplier λ_i such that

$$\begin{aligned} \lambda_i\gamma_{i,u} &= -\mathbf{v}^{T}\mathbf{D}_{i,u}\,, \ \ \lambda_i\gamma_{i,w} = -\mathbf{v}^{T}\mathbf{D}_{i,w}, \\ \lambda_i \ge 0 \ &\perp \ \mu_i\gamma_{i,n} - \sqrt{\gamma_{i,u}^2 + \gamma_{i,w}^2} \ge 0. \end{aligned} \quad (11)$$

If for $i \in A(\mathbf{q}^{(l)},\delta)$, $\mathbf{c}_i^T \equiv [\gamma_{i,n}, \gamma_{i,u}, \gamma_{i,w}]$, and $\mathbf{g}_i^T \equiv \left[\frac{1}{h}\Phi_i(\mathbf{q}^{(l)}) + \mathbf{D}_{i,n}^{T}\mathbf{v}^{(l+1)}, \mathbf{D}_{i,u}^{T}\mathbf{v}^{(l+1)}, \mathbf{D}_{i,w}^{T}\mathbf{v}^{(l+1)}\right]$, then drawing on Eqs. (10) and (11), one can show that $\mathbf{g}_i^T\mathbf{c}_i = 0$, and thus $\mathbf{g}_i \perp \mathbf{c}_i$. If one defines the cones

$$\Lambda_i = \left\{ \mathbf{g} = [g_1, g_2, g_3]^T \in \mathbb{R}^3 \ | \ g_1 \ge \mu_i\sqrt{g_2^2 + g_3^2} \right\},$$
$$C_i = \left\{ \mathbf{c} = [c_1, c_2, c_3]^T \in \mathbb{R}^3 \ | \ \mu_i c_1 \ge \sqrt{c_2^2 + c_3^2} \right\},$$

then Λ_i is the negative polar cone of C_i; that is, $\mathbf{g} \in \Lambda_i$ and $\mathbf{c} \in C_i$ implies that $\mathbf{g}^T\mathbf{c} \ge 0$. Here, C°, the polar cone of a given cone $C \subset \mathbb{R}^m$, is defined as $C^\circ = \{ \mathbf{x} \in \mathbb{R}^m | \langle \mathbf{x}, \mathbf{y} \rangle \le 0, \ \forall \mathbf{y} \in C \}$. Then, based on Eqs. (8) and (10), the following set of cone complementarity constraints holds:

$$-\mathbf{g}_i \in C_i^\circ \perp \mathbf{c}_i \in C_i, \quad \forall i \in A(\mathbf{q}^{(l)},\delta). \quad (12)$$

In what follows, the focus shifts back to reformulating the optimization problem with equilibrium constraints of Eqs. (6) through (9) to account for the above cone complementarity constraints. To this end, the notation $\mathbf{Q} \equiv \mathbf{M}\mathbf{v}^{(l)} + h\mathbf{f}(t^{(l)}, \mathbf{q}^{(l)}, \mathbf{v}^{(l)})$ is introduced. Then, Eq. (6) is reformulated as $\mathbf{M}\mathbf{v}^{(l+1)} = \mathbf{Q} + \mathbf{D}\gamma$, where with n_A being the cardinality of $A(q^{(l)},\delta)$, $\mathbf{D} \equiv [\mathbf{D}_1 \ldots \mathbf{D}_{n_A}] \in \mathbb{R}^{6n_b \times 3n_A}$, $\gamma = [\gamma_1^T \ldots \gamma_{n_A}^T]^T \in \mathbb{R}^{3n_A}$, and, for $i \in A(q^{(l)},\delta)$, $\mathbf{D}_i \equiv [\mathbf{D}_{i,n} \ \mathbf{D}_{i,u} \ \mathbf{D}_{i,w}] \in \mathbb{R}^{6n_b \times 3}$ and $\gamma_i^T \equiv [\gamma_{i,n} \ \gamma_{i,u} \ \gamma_{i,w}]$. Next, the vector $\mathbf{b} \in \mathbb{R}^{3n_A}$ is defined as $\mathbf{b}^T \equiv [\frac{1}{h}\Phi_1(\mathbf{q}^{(l)}) \ 0 \ 0 \ldots \frac{1}{h}\Phi_{n_A}(\mathbf{q}^{(l)}) \ 0 \ 0]$. Here, for an arbitrary vector $\mathbf{h} \in \mathbb{R}^{3n_A}$, the notation $\mathbf{h}_i$, $1 \le i \le n_A$, is used to represent the entries in rows $3(i-1)+1, 3(i-1)+2, 3i$, that is, the entries associated with contact i. Finally, with the notation $\mathbf{N} \equiv \mathbf{D}^T\mathbf{M}^{-1}\mathbf{D}$, and $\mathbf{d} \equiv \mathbf{b} + \mathbf{D}^T\mathbf{M}^{-1}\mathbf{Q}$, the configuration of the multibody system at $t^{(l+1)}$ is obtained as the solution of the following:

$$-\left(\mathbf{N}\gamma^{(l+1)} + \mathbf{d}\right)_i \in C_i^\circ \perp \gamma_i^{(l+1)} \in C_i\,, \ \ 1 \le i \le n_A, \quad (13)$$

$$\mathbf{M}\mathbf{v}^{(l+1)} = \mathbf{Q} + \mathbf{D}\gamma, \quad (14)$$

$$\mathbf{q}^{(l+1)} = \mathbf{q}^{(l)} + h\mathbf{L}(\mathbf{q}^{(l)})\mathbf{v}^{(l+1)}. \quad (15)$$

The frictional contact forces $\gamma^{(l+1)}$ are obtained by solving the cone complementarity problem in Eqs. (13) and (14), where, at each iteration, the velocity is evaluated as $\mathbf{v}^{(l+1)} = \mathbf{M}^{-1}\left(\mathbf{Q}+\mathbf{D}\gamma^{(l+1)}\right)$. The attitude of each body in the system is obtained by using Eq. (15). Note that $\mathbf{N}$ is positive semidefinite, while $\mathbf{M}$ is diagonal, constant, and positive definite. Equation (13) in fact represents the optimality condition of the following cone complementarity problem:

$$\min f(\gamma) = \frac{1}{2}\gamma^T\mathbf{N}\gamma + \mathbf{d}^T\gamma \quad \text{s.t.} \quad \gamma_i \in C_i\,, \quad \forall\ 1 \le i \le n_A,$$

whose solution is found by using an iterative algorithm that, starting with an arbitrarily chosen $\gamma^{(0)}$, computes the iteration $r+1$, $r \ge 0$,

$$\gamma_i^{r+1} = \rho\,\Pi_{C_i}\left[\gamma_i^r - \omega\eta_i\,(\mathbf{N}\gamma+\mathbf{d})_i + \sum_{m\in D(i,r)} \mathbf{K}_{i,m}^r\left(\gamma_m^{r+1}-\gamma_m^r\right)\right] + (1-\rho)\,\gamma_i^r, \tag{16}$$

where η_i will be defined in Algorithm 1 below, $0 < \rho \le 1$ and $\omega > 0$ are two parameters, and, for each r, $\mathbf{K}_{i,m}^r$ is a coefficient matrix that indicates how the frictional contact force associated with contact m gets reflected in the computation of the frictional contact force associated with contact i. Here $D(i,r)$ represents the set of contacts considered when updating the frictional contact forces associated with contact i. Note that for a Gauss-Jacobi-type iteration $D(i,r) = \emptyset$ and for a Gauss-Seidel type iteration $D(i,r) = \{1,2,\dots,i-1\}$, but other update strategies, which might depend on the iteration index r, can and will be pursued. Finally, the operator Π_{C_i} is the cone i projection operator; see, for instance, [27].

The iterative scheme in Eq. (16) was proved to converge under mild assumptions that can be met by a suitable choice of relaxation parameter ω [2]. Therein, it was also pointed out that a Gauss-Seidel update sequence in the iterative process led to a robust algorithm. Although convenient for the convergence analysis, Eq. (16) is not the form that is considered for software implementation. Rather, an inner loop iteration algorithm, that also updates the speed $\mathbf{v}^{(l+1)}$, is provided in the following pseudocode, Algorithm 1:

Algorithm 1: Inner Iteration Loop

```
1. For $1 \le i \le n_A$, evaluate $\eta_i = 3/\text{Trace}(\mathbf{D}_i^T\mathbf{M}^{-1}\mathbf{D}_i)$.
2. If warm start with initial guess $\gamma^*$, then set $\gamma^0 = \gamma^*$, otherwise
   $\gamma^0 = \mathbf{0}$.
3. Initialize speeds: $\mathbf{v}^{(l+1)} = \sum_{i=1}^{n_A}\mathbf{M}^{-1}\mathbf{D}_i\gamma^0 + \mathbf{M}^{-1}\mathbf{Q}$.
4. For $i = 1,\dots n_A$, perform the updates:
       $\gamma_i^{prelim} = \gamma_i^r - \omega\eta_i\left(\mathbf{D}_i^T\mathbf{v}^{(l+1)} + \mathbf{b}_i\right)$;
       $\gamma_i^{r+1} = \rho\,\Pi_{C_i}\left(\gamma_i^{prelim}\right) + (1-\rho)\gamma_i^r$;
       $\Delta\gamma_i^{r+1} = \gamma_i^{r+1} - \gamma_i^r$;
       $\mathbf{v}^{(l+1)} := \mathbf{v}^{(l+1)} + \mathbf{M}^{-1}\mathbf{D}_i\Delta\gamma_i^{r+1}$.
5. Repeat step 4 by looping on the list of contacts in reverse
   order, if symmetric updates are desired.
6. $r := r+1$. Repeat from 4 until convergence or until $r > r_{max}$.
```

The stopping criterion is based on the value of the velocity update. The overall algorithm that provides an approximation to the solution of Eqs. (13) through (15) relies on Algorithm 1 and requires the following steps, Algorithm 2

Algorithm 2: Outer, Time-Stepping, Loop

```
1. Set $t = 0$, step counter $l = 0$, and provide initial values for
   $\mathbf{q}^{(l)}$ and $\mathbf{v}^{(l)}$.
2. Perform collision detection between bodies, obtaining $n_A$
   possible contact points within a distance $\delta$. For each contact $i$,
   compute $\mathbf{D}_{i,n}$, $\mathbf{D}_{i,u}$, $\mathbf{D}_{i,w}$, and residual $\Phi_i(\mathbf{q})$, which
   also provides $\mathbf{b}_i$.
3. For each body, compute forces $\mathbf{f}(t^{(l)},\mathbf{q}^{(l)},\mathbf{v}^{(l)})$ and then $\mathbf{Q}$.
4. Use Algorithm 1 to solve the cone complementarity problem
   and obtain unknown impulse $\gamma$ and velocity $\mathbf{v}^{(l+1)}$.
5. Update positions using $\mathbf{q}^{(l+1)} = \mathbf{q}^{(l)} + h\mathbf{L}(\mathbf{q}^{(l)})\mathbf{v}^{(l+1)}$.
6. Increment $t := t+h$, $l := l+1$, and repeat from step 2 until
   $t > t_{\text{end}}$.
```

Note that choosing a proper value δ for the collision envelope is not trivial. On the one hand, if a very small or zero value is used, contacts will enter the CCP solver only when it is too late, and some amount of interpenetration will be unavoidable, which in turn adversely impacts the stability of the method. On the other hand, if too large a value is used, the collision detection algorithm will return too many potential contacts that waste computational resources and could occasionally create trouble with convex shapes, which decreases the efficiency of the method. If $\mathbf{v}$ is known, a simple yet efficient heuristic is to choose δ as a rough approximation of the maximum distance that can be traveled within a step size by an arbitrary point of a moving body.

Parallel Implementation Details

The parallel implementation for the multibody dynamics frictional contact problem depends heavily on the underlying hardware used to run the simulation. Specifically, the algorithm proposed is well suited for running on parallel platforms that support the single instruction multiple data computational paradigm, which is ideally suited for handling problems with contacts in excess of hundreds of thousands. NVIDIA's GeForce 80 family of GPUs has been adopted as the implementation platform. Priced at $460 per card, the GeForce 8800 GTX has been clocked at 320 GFlop, about seven times faster than an Intel Core 2 Duo running at 3 GHz, and this gap is bound to grow in the immediate future[2]. A description of the hardware behind the GeForce 8800 GTX card used is beyond the scope of this paper. It suffices to indicate that for the frictional contact problem discussed here, contacts are processed by each of the 16 streaming multiprocessors in warps of 16 contacts per streaming multiprocessor. As far as memory allocation is concerned, each streaming multiprocessor has 8,192 registers that each can hold a float or an integer, and each has 16 KB of shared memory that can be shared by all live threads associated with the respective streaming multiprocessor. In this context, the maximum number of live threads is, at 768, much larger than the batch of 16 threads that are executed at each time, since some of threads are active but parked while waiting for global memory fetches, a strategy that hides the memory latency and improves the overall efficiency. The recommended strategy for GPU computing on GeForce 8800 GTX is to try to maximize the number of parallel threads active at any given time, which in an ideal situation would be at 12,228 threads. Since register and shared memory resources are fixed, however, the more threads active, the fewer the resources per thread. In the end, an optimal point is reached where, while the number of threads per streaming processor is still large, the memory allocated to each thread is enough to allow it to run a batch of commands associated with Algorithm 1. In the current parallel implementation, Algorithm 1 runs with 6,144 active threads at any given time; in other words, the GPU is working with 6,144 contacts at each time. The number of active threads can be further increased but at the price of an increased number of global memory accesses, where "global" here represents the 768 MB GDDR3 on-chip GPU memory. While global memory fetches are still fast because of low latency and high memory bandwidth (86.4 GB/sec), they are still two orders of magnitude slower than register or shared-memory access and should be avoided whenever possible.

In the context of this work, the most relevant consequence of using a parallel execution approach is that this execution model can lead to a random velocity update sequence. In a *sequential* execution mode, the loop over the active contact set in Algorithm 1 is carried out in an orderly fashion. If symmetric updates are desired, then the order is reversed, but it is still predefined and deterministic. When implementing a *parallel* version of Algorithm 1, two approaches can be followed. The first falls back on a Gauss-Jacobi approach and, referring back to Eq. (16), corresponds to $D(i,r) = \emptyset$. Numerical experiments suggest that the convergence rate decreases and, for certain models, leads to a large number of iterations r or even lack of convergence. The second approach investigated in this work allows for a random update sequence, to the extent that in general $D(i,r) \neq D(i,p)$, for $r \neq p$. In this context, for the Gauss-Seidel there is no dependency in Algorithm 1 of $D(i,r) = \{1,2,\ldots,i-1\}$ on r, and in fact each iteration follows the same update sequence. Enforcing a similar update sequence in a parallel execution scenario would unacceptably compromise performance, as this has to rely on a system of semaphores similar to the `mutex` support in the POSIX standard threading library API, which in fact is not supported by CUDA library support [25]. Note that there is a remote chance that during some iteration r a certain update produced by contact i would fail altogether because of a race condition and lack of `mutex`-type support. This situation is not expected to impact the convergence of the algorithm, however, particularly for large frictional contact problems with hundreds of thousands of contacts, where the probability of having 2 out of 256 contacts associated with the same body and of updating its velocity at the same time is small, but not zero. Note also that the 256 value corresponds on GeForce 8800 GTX to the number of streaming multiprocessors (16), each processing simultaneously a warp of 16 threads (frictional contacts).

Data Structures

At each simulation step, the CPU, that is, the "host", feeds data into the GPU memory, launches one or more *kernels* (functions to be performed simultaneously on many parallel GPU threads), and gathers the results of the computations by copying select portions of the GPU memory back into the host RAM. Special care should be paid to minimize the memory overhead caused by repeated transfers of large data structures. Moreover, data structures should be organized to exploit fast GPU coalesced memory access to fetch data for all parallel threads in a warp. Provided that bytes are contiguous and that the $k-\mathrm{th}$ thread accesses the $k-\mathrm{th}$ element in the data structure, up to 512 bytes can be fetched in one operation by a warp of threads. Failing to perform coalesced memory access may slow the kernel significantly. For the algorithm developed, the data structure for the contacts has been mapped into columns of four floats, as shown in Fig. 2.

There is no need to store the entire $\mathbf{D}_i$ matrix for the $i-\mathrm{th}$ contact because it has zero entries for most of its part, except for the two 12x3 blocks corresponding to the coordinates of the two bodies in contact. Hence, the product $\mathbf{D}_i^T \mathbf{v}^{(l+1)}$ in the fourth step of Algorithm 1 can be performed as

$$\mathbf{D}_i^T \mathbf{v}^{(l+1)} = \mathbf{D}_{i,v_A}^T \dot{\mathbf{r}}_\mathbf{A} + \mathbf{D}_{i,\omega_A}^T \omega_\mathbf{A} + \mathbf{D}_{i,v_B}^T \dot{\mathbf{r}}_\mathbf{B} + \mathbf{D}_{i,\omega_B}^T \omega_\mathbf{B} \qquad (17)$$

[2]Note that through a new product line called Larrabee, Intel is integrating the GPU and CPU on the same chip and is expected to support in early 2010 the same parallel computation model promoted by NVIDIA, only with larger shared memory and L1 cache.

with the adoption of the following 3x3 matrices.

$$\mathbf{D}_{i,v_A}^T = -\mathbf{A}_{i,p}^T \tag{18a}$$
$$\mathbf{D}_{i,\omega_A}^T = \mathbf{A}_{i,p}^T \mathbf{A}_A \tilde{\bar{\mathbf{s}}}_{i,A} \tag{18b}$$
$$\mathbf{D}_{i,v_B}^T = \mathbf{A}_{i,p}^T \tag{18c}$$
$$\mathbf{D}_{i,\omega_B}^T = -\mathbf{A}_{i,p}^T \mathbf{A}_B \tilde{\bar{\mathbf{s}}}_{i,B} \tag{18d}$$

Similarly, the update $\mathbf{v}^{(l+1)} := \mathbf{v}^{(l+1)} + \mathbf{M}^{-1}\mathbf{D}_i \Delta\gamma_i^{r+1}$ can be computed explicitly as the following sparse update to the speeds of two bodies only.

$$\dot{\mathbf{r}}_{\mathbf{A}}^{(l+1)} := \dot{\mathbf{r}}_{\mathbf{A}}^{(l)} - m_A^{-1}\mathbf{D}_{i,v_A}\Delta\gamma_i^{r+1} \tag{19a}$$
$$\omega_{\mathbf{A}}^{(l+1)} := \omega_{\mathbf{A}}^{(l)} + \mathbf{J}_A^{-1}\mathbf{D}_{i,\omega_A}\Delta\gamma_i^{r+1} \tag{19b}$$
$$\dot{\mathbf{r}}_{\mathbf{B}}^{(l+1)} := \dot{\mathbf{r}}_{\mathbf{B}}^{(l)} + m_B^{-1}\mathbf{D}_{i,v_B}\Delta\gamma_i^{r+1} \tag{19c}$$
$$\omega_{\mathbf{B}}^{(l+1)} := \omega_{\mathbf{B}}^{(l)} - \mathbf{J}_B^{-1}\mathbf{D}_{i,\omega_B}\Delta\gamma_i^{r+1} \tag{19d}$$

Since $\mathbf{D}_{i,v_A}^T = -\mathbf{D}_{i,v_B}^T$, there is no need to store both matrices, so in each contact data structure only a matrix $\mathbf{D}_{i,v_{AB}}^T$ is stored, which is then used with opposite signs for each of the two bodies.

Figure 3 shows that for each body there is a data structure containing the state (speed and position), the mass moments of inertia and mass values, and the external applied force $\mathbf{F}_j$ and torque $\mathbf{C}_j$. Forces and torques, if any, are used to compute the third step of Algorithm 1. Note that, in order to speed the iteration, it is better to store the inverse of the mass and inertias rather than their original values, because the operation $\mathbf{M}^{-1}\mathbf{D}_i\Delta\gamma_i^{r+1}$ must be performed multiple times. Each contact will reference its two touching bodies through the two pointers B_A and B_B, in the fourth and seventh rows of the contact data structure.

Numerical experiments show that for high memory throughput, it is better to pad the data into a four-float width structure even at the cost of wasting memory space when several entries end up not being used. Also, the variables in the data structures are organized in a way that minimizes the number of fetch and store operations. This approach maximizes the arithmetic intensity of the kernel code, as recommended by the CUDA development guidelines.

Another software design decision that improved the overall performance regards the delegation of contact preprocessing step to the GPU. Specifically, instead of computing the data structures of the contacts on the host, only the contact normals and contact points were copied into the GPU memory. Then, a GPU kernel computed $\mathbf{D}_{i,v_{AB}}^T$, $\mathbf{D}_{i,\omega_A}^T$, $\mathbf{D}_{i,\omega_B}^T$, η_i, $b_{i,n}$, as shown in Fig. 4. This strategy leads to faster code, not only because the preprocessing kernel runs in parallel on the GPU, but also because it avoids the memory overhead if copying the full contact structures from host to GPU. Note that $b_{i,v}$ and $b_{i,w}$ are always zero and that the data structures for both bodies and contacts on the GPU are processed in thread blocks, which are organized in block grids.

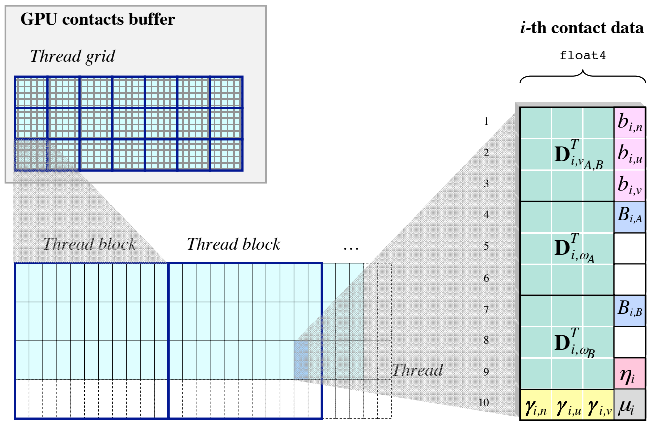

Figure 2. Grid of data structures for frictional contacs, in GPU memory.

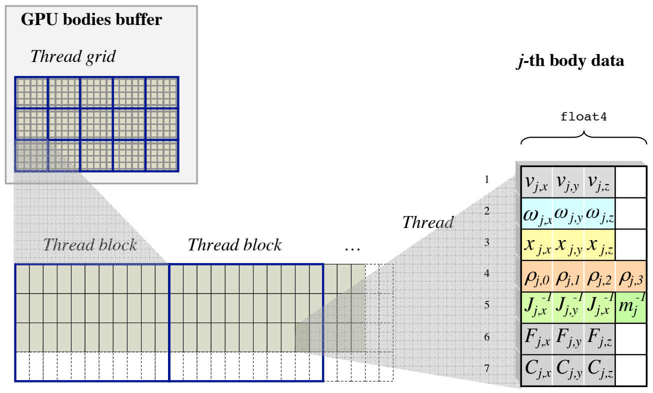
GPU bodies buffer
Thread grid
Thread block
Thread block
...
Thread
j-th body data
float4
1
2
3
4
5
6
7
$v_{j,x}$ $v_{j,y}$ $v_{j,z}$
$\omega_{j,x}$ $\omega_{j,y}$ $\omega_{j,z}$
$x_{j,x}$ $x_{j,y}$ $x_{j,z}$
$\rho_{j,0}$ $\rho_{j,1}$ $\rho_{j,2}$ $\rho_{j,3}$
$J_{j,x}^{-1}$ $J_{j,y}^{-1}$ $J_{j,x}^{-1}$ m_j^{-1}
$F_{j,x}$ $F_{j,y}$ $F_{j,z}$
$C_{j,x}$ $C_{j,y}$ $C_{j,z}$

Figure 3. Grid of data structures for rigid bodies, in GPU memory.

The Parallel Algorithm

The pseudocode in Algorithm 3 outlines how Algorithm 1 and Algorithm 2 can be combined and turned into a sequence of computational phases, for the most part executed as parallel kernels on the GPU. In terms of resource allocation, the computation kernels followed a one-thread-per-body or a one-thread-per-contact philosophy, depending on the phase of the algorithm.

Algorithm 3: Complete Time Stepping, when GPU is Available.

1. (*Host, serial*) Perform collision detection between bodies,

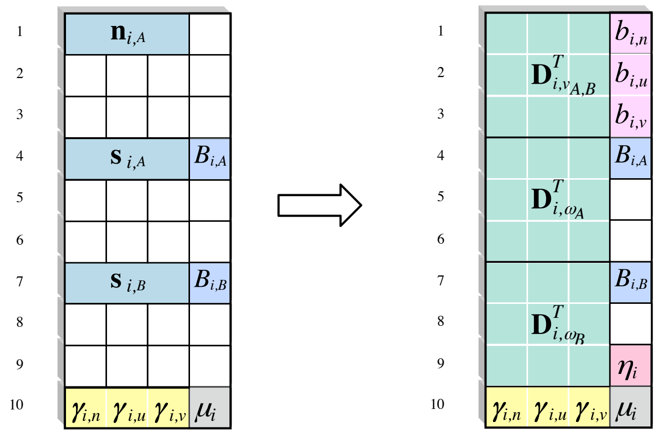
1
2
3
4
5
6
7
8
9
10
$\mathbf{n}_{i,A}$
$\mathbf{s}_{i,A}$
$B_{i,A}$
$\mathbf{s}_{i,B}$
$B_{i,B}$
$\gamma_{i,n}$ $\gamma_{i,u}$ $\gamma_{i,v}$ μ_i
$\mathbf{D}^T_{i,v_{A,B}}$
$b_{i,n}$
$b_{i,u}$
$b_{i,v}$
$\mathbf{D}^T_{i,\omega_A}$
$B_{i,A}$
$\mathbf{D}^T_{i,\omega_B}$
$B_{i,B}$
η_i
$\gamma_{i,n}$ $\gamma_{i,u}$ $\gamma_{i,v}$ μ_i

Figure 4. Contact data structure, before and after the preprocessing kernel.

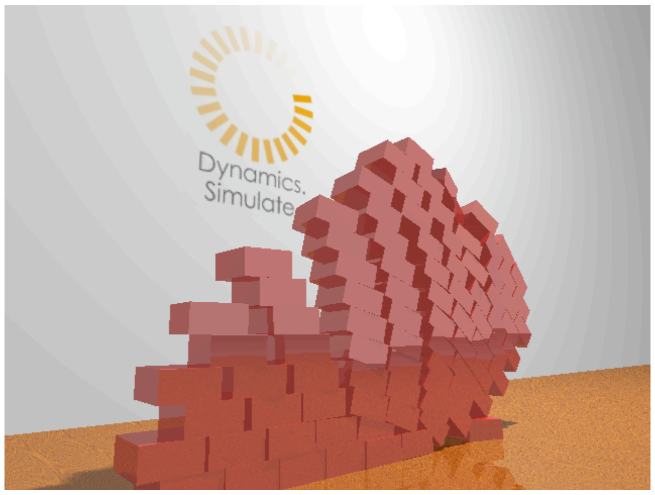

Figure 5. Frame from the simulation of a brick wall. Frictional contact is present between the bricks in the wall and between the bricks and ground.

obtaining n_A possible contact points within a distance δ, as contact positions $s_{i,A}$, $s_{i,B}$ on the two touching surfaces, and normals $\mathbf{n}_i$. If warm start is used, then fetch the last reactions in contact point γ_i^* (obtained in previous frame, if the contact is persistent) and set $\gamma_i = \gamma_i^*$, otherwise $\gamma_i = \mathbf{0}$.

2. (*Host, serial*) Copy the contact and body data structures from host memory to GPU memory.
3. (*GPU, body-parallel*) **Force kernel**. For each body, compute forces $\mathbf{f}(t^{(l)}, \mathbf{q}^{(l)}, \mathbf{v}^{(l)})$, if any. Store these forces and torques into F_j and C_j. For example, apply the gravitational and gyroscopic forces.
4. (*GPU, contact-parallel*) **Contact preprocessing kernel**. For each contact, given contact normal and position, compute in-place the matrices $\mathbf{D}^T_{i,v_{AB}}$, $\mathbf{D}^T_{i,\omega_A}$ and $\mathbf{D}^T_{i,\omega_B}$, then compute η_i and the contact residual $b_{i,n} = \frac{1}{h}\Phi_i(\mathbf{q})$. Set $b_{i,u}$ and $b_{i,w}$ as zero.
5. (*GPU, body-parallel*) **CCP force kernel**. For each body j, initialize body speeds: $\dot{\mathbf{r}}_j^{(l+1)} = h\, m_j^{-1}\mathbf{F}_j$ and $\omega_j^{(l+1)} = h\,\mathbf{J}_j^{-1}\mathbf{C}_j$.
6. (*GPU, contact-parallel*) **CCP initialization kernel**. For each contact i, update initial speeds of bodies A and B using Eq. (19), with the initial values of γ_i. This set can be skipped if no warm start is used. Project to friction cones when necessary.
7. (*GPU, contact-parallel*) **CCP iteration kernel**. For each contact i, do $\gamma_i^{prelim} = \gamma_i^r - \omega\eta_i \left(\mathbf{D}_i^T \mathbf{v}^{(l+1)} + \mathbf{b}_i\right)$. Note that $\mathbf{D}_i^T \mathbf{v}^{(l+1)}$ must be done with sparse data, using Eq. (18). Also do $\gamma_i^{r+1} = \rho\, \Pi_{C_i}\left(\gamma_i^{prelim}\right) + (1-\rho)\gamma_i^r$, by projecting multipliers onto the ith friction cone. After computing $\Delta\gamma_i^{r+1} = \gamma_i^{r+1} - \gamma_i^r$, update the speeds of the two connected bodies A and B as in Eq. (19).
8. Repeat the previous kernel until convergence or until number of CCP steps reached $r > r_{max}$.
9. (*GPU, body-parallel*) **Time integration kernel**. For each j body, perform time integration as $\mathbf{q}_j^{(l+1)} = \mathbf{q}_j^{(l)} + h\mathbf{L}(\mathbf{q}_j^{(l)})\mathbf{v}_j^{(l+1)}$
10. (*Host, serial*) Copy body data structures from GPU memory to host memory. Copy contact multipliers from GPU memory to host memory.

Numerical Results

A set of two numerical experiments was carried out with a benchmark problem to assess the performance of the frictional contact method discussed when executed in a sequential computational framework and to compare the sequential to the parallel solution approach proposed in the paper. For each numerical experiment three scenarios were considered, differentiated by the number of brick elements: 1000, 2000, and 8000 bricks, respectively. Initial conditions were such that the wall slowly collapsed. Figure 5 presents a snapshot of the dynamics of the process. The wall in the picture has a small number of bricks yet qualitatively captures the essence of the dynamics for the larger models considered here. The friction coefficient between bricks, and between bricks and ground, was set to 0.6.

The simulation was carried out with a numerical integration step-size $h = 0.01$s, which is three orders of magnitude larger than typically required by the DEM method in [36]. Perhaps less relevant are absolute timing results, since they depend on issues such as cache management that can vastly change with different compilers and optimization levels. With respect to the sequential simulation results, the scaling of the algorithm with the num-

Table 1. Average simulation times (in sec.) required to capture one second of the dynamics of the falling wall.

Bricks	Sequential Version	GPU Coprocessing Version
1000	43	6
2000	87	10
8000	319	42

ber of bodies is of interest. These and other results obtained for large-scale simulation of models with frictional contact suggest that the algorithm discussed here displays linear complexity [2]. In other words, the simulation time increases linearly with the number of bodies in the model. Comparing the sequential and parallel simulation results, one can see a speedup of a factor of 7 when relying on the GPU in coprocessor mode. In spite of using the NVIDIA GTX8800 GPU model with 128 parallel threads, the speedup is limited to 7x because the collision detection draws on an open source package that does not yet leverage GPU parallelism [37].

Conclusions

The paper proposes a theoretically rigorous approach to simulating multibody dynamics problems with frictional contact. The algorithm proposed is backed by convergence results for measure differential inclusions [24] and by a rigorous convergence analysis [2]. The methodology leverages commodity high-performance parallel computing on the GPU. Preliminary results obtained with the proposed parallel algorithm demonstrate that for very large problems the computational bottleneck associated with the sequential algorithm — that is, the solution of the cone complementarity problem — has been eliminated. The computation is now dominated by the collision detection stage, which at this time runs sequentially. Four issues remain to be addressed. First, a rigorous convergence analysis for the case of random velocity updates is needed. Although preliminary results show that this update strategy works in conjunction with large frictional contact models, it is important to understand and possibly address some of the limitations associated with this approach. Second, the methodology should be expanded to include the case of bilateral and unilateral constraints present in a multitude of mechanical system models. It is expected that the latter will positively impact the RATTLE and SHAKE algorithms in molecular dynamics simulation. Third, the cone complementarity problem approach needs to be extended to deformable multibody dynamics problems. Fourth, a parallel collision detection engine would allow for an entirely parallel approach to multibody dynamics with frictional contact that, given the recent advances in commodity high-performance parallel computing hardware, opens up new fields of simulation-based engineering in materials science, the pharmaceutical industry, and granular flow dynamics. The first three open issues will be addressed in future work; the fourth one is essentially a challenging computer science problem that is currently being addressed by that community.

Acknowledgment

Financial support for the second author is provided in part by National Science Foundation under Grant No. CMMI-0700191. A. Tasora and D. Negrut thank the NVIDIA corporation for sponsoring their research programs in the area of high-performance multibody dynamics simulation. M. Anitescu was supported by Contract DE-AC02-06CH11357 of the U.S. Department of Energy.

REFERENCES

[1] J. Madsen, N. Pechdimaljian, and D. Negrut. Penalty versus complementarity-based frictional contact of rigid bodies: A CPU time comparison. Technical Report TR-2007-05, Simulation-Based Engineering Lab, University of Wisconsin, Madison, 2007.

[2] M. Anitescu and A. Tasora. An iterative approach for cone complementarity problems for nonsmooth dynamics. *Computational Optimization and Applications*, 2008, in press.

[3] N. Kikuchi and J.T. Oden. *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods*. SIAM, Philadelphia, PA, 1998.

[4] Peter Wriggers. *Computational Contact Mechanics*. Springer-Verlag, Berlin, Heidelberg, second edition, 2006.

[5] Bruce R. Donald and Dinesh K. Pai. On the motion of compliantly connected rigid bodies in contact: A system for analyzing designs for assembly. In *Proceedings of the Conf. on Robotics and Automation*, pages 1756–1762. IEEE, 1990.

[6] Peng Song, P. Kraus, Vijay Kumar, and P. Dupont. Analysis of rigid-body dynamic models for simulation of systems with frictional contacts. *Journal of Applied Mechanics*, 68(1):118–128, 2001.

[7] Peng Song, Jong-Shi Pang, and Vijay Kumar. A semi-implicit time-stepping model for frictional compliant contact problems. *International Journal of Numerical Methods in Engineering*, 60(13):267–279, 2004.

[8] Jong-Shi Pang, Vijay Kumar, and Peng Song. Convergence of time-stepping method for initial and boundary-value frictional compliant contact problems. *SIAM Journal of Numerical Analysis*, 43(5):2200–2226, 2005.

[9] Jong-Shi Pang and David Stewart. Differential variational inequalities. *Mathematical Programming*, 113(2):345–424, 2008.

[10] Jean J. Moreau. Standard inelastic shocks and the dynamics of unilateral constraints. In G. Del Piero and F. Macieri, editors, *Unilateral Problems in Structural Analysis*, pages 173–221, New York, 1983. CISM Courses and Lectures no. 288, Springer–Verlag.

[11] P. Lotstedt. Mechanical systems of rigid bodies subject to unilateral constraints. *SIAM Journal of Applied Mathematics*, 42(2):281–296, 1982.

[12] M. D. P.Monteiro Marques. *Differential Inclusions in Nonsmooth Mechanical Problems: Shocks and Dry Friction*, volume 9 of *Progress in Nonlinear Differential Equations and Their Applications*. Birkhäuser Verlag, Basel, 1993.

[13] David Baraff. Issues in computing contact forces for nonpenetrating rigid bodies. *Algorithmica*, 10:292–352, 1993.

[14] Jong-Shi Pang and Jeffrey C. Trinkle. Complementarity formulations and existence of solutions of dynamic multi-rigid-body contact problems with Coulomb friction. *Mathematical Programming*, 73(2):199–226, 1996.

[15] Jeffrey Trinkle, Jong-Shi Pang, Sandra Sudarsky, and Grace Lo. On dynamic multi-rigid-body contact problems with Coulomb friction. *Zeitschrift fur angewandte Mathematik und Mechanik*, 77:267–279, 1997.

[16] David E. Stewart and Jeffrey C. Trinkle. An implicit time-stepping scheme for rigid-body dynamics with inelastic collisions and Coulomb friction. *International Journal for Numerical Methods in Engineering*, 39:2673–2691, 1996.

[17] Mihai Anitescu and Florian A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14:231–247, 1997.

[18] Mihai Anitescu, Florian A. Potra, and David Stewart. Time-stepping for three-dimensional rigid-body dynamics. *Computer Methods in Applied Mechanics and Engineering*, 177:183–197, 1999.

[19] David E. Stewart. Rigid-body dynamics with friction and impact. *SIAM Review*, 42(1):3–39, 2000.

[20] Richard W. Cottle and George B. Dantzig. Complementary pivot theory of mathematical programming. *Linear Algebra and Its Applications*, 1:103–125, 1968.

[21] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Computer Graphics (Proceedings of SIGGRAPH)*, pages 23–34, 1994.

[22] Mihai Anitescu and Gary D. Hart. A fixed-point iteration approach for multibody dynamics with contact and friction. *Mathematical Programming, Series B*, 101(1):3–32, 2004.

[23] Alessandro Tasora, E. Manconi, and M. Silvestri. Un nuovo metodo del simplesso per il problema di complementarit lineare mista in sistemi multibody con vincoli unilateri. In *Proceedings of AIMETA 05*, Firenze, Italy, 2005.

[24] Mihai Anitescu. Optimization-based simulation of nonsmooth rigid multibody dynamics. *Mathematical Programming*, 105(1):113–143, 2006.

[25] NVIDIA. CUDA Programming Guide. Available online at http://developer.download.nvidia.com/compute/cuda/1_1/NVIDIA_CUDA_Programming_Guide_1.1.pdf, 2007.

[26] A. A. Shabana. *Dynamics of Multibody Systems*. Cambridge University Press, third edition, 2005.

[27] A. Tasora. High performance complementarity solver for non-smooth dynamics. In C. L. Bottasso, P. Masarati, and L. Trainelli, editors, *Proceedings of the ECCOMAS Multibody Dynamics Conference*, Milan, Italy, 2007.

[28] Young J. Kim, Ming C. Lin, and Dinesh Manocha. Deep: Dual-space expansion for estimating penetration depth between convex polytopes. In *Proceedings of the 2002 International Conference on Robotics and Automation*, volume 1, pages 921–926. Institute for Electrical and Electronics Engineering, 2002.

[29] Mihai Anitescu, James F. Cremer, and Florian A. Potra. Formulating 3D contact dynamics problems. *Mechanics of Structures and Machines*, 24(4):405–437, 1996.

[30] Mihai Anitescu and Gary D. Hart. A constraint-stabilized time-stepping approach for rigid multibody dynamics with joints, contact and friction. *International Journal for Numerical Methods in Engineering*, 60(14):2335–2371, 2004.

[31] Gary D. Hart. *A constraint-stabilized time-stepping approach for piecewise smooth multibody dynamics*. Ph.D. in mathematics, University of Pittsburgh, 2007.

[32] J. J. Moreau. Unilateral contact and dry friction in finite freedom dynamics. In J. J. Moreau and P. D. Panagiotopoulos, editors, *Nonsmooth Mechanics and Applications*, pages 1–82, Berlin, 1988. Springer-Verlag.

[33] Dimitri P Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

[34] J. S. Pang and D. E. Stewart. Differential variational inequalities. *Mathematical Programming*, 113:1–80, 2008.

[35] David E. Stewart. Convergence of a time-stepping scheme for rigid body dynamics and resolution of Painleve's problems. *Archive Rational Mechanics and Analysis*, 145(3):215–260, 1998.

[36] C. H. Rycroft, G. S. Grest, J. W. Landry, and M. Z. Bazant. Analysis of granular flow in a pebble-bed nuclear reactor. *Physical Review E*, 74, 021306, 2006.

[37] Physics Simulation Forum. Bullet Physics Library. Available online at http://www.bulletphysics.com/Bullet/wordpress/bullet, 2008.