Simulation of Massive Multibody Systems using GPU Parallel Computation

Alessandro Tasora
University of Parma
Dept. of Industrial Engineering
V.G.Usberti 181/A
43100, Parma, Italy
tasora@ied.unipr.it

Dan Negrut
University of Wisconsin–Madison
Dept. of Mechanical Engineering
1513 University Avenue
Madison, WI 53706, US
negrut@wisc.edu

Mihai Anitescu
Argonne National Laboratories
MCS division
9700 South Cass Avenue
Argonne, IL 60439, US
anitescu@mcs.anl.gov

Hammad Mazhar
University of Wisconsin–Madison
Dept. of Mechanical Engineering
1513 University Avenue
Madison, WI 53706, US
hmazhar@wisc.edu

Toby David Heyn
University of Wisconsin–Madison
Dept. of Mechanical Engineering
1513 University Avenue
Madison, WI 53706, US
heyn@wisc.edu

ABSTRACT

We describe an efficient method for the simulation of complex scenarios with millions of frictional contacts and mechanical constraints. To this end, the GPU processors of the modern graphic boards are used to solve the differential inclusion problem that represents the most challenging part of the multi–rigid–body problem. Thank to the massive parallelism offered by GPU boards, we are able to simulate sand, granular materials, soils and other complex physical scenarios with a large speedup respect to serial CPU–based algorithms.

Keywords
Constraints, dynamics, GPU, parallel computing, collision detection

1 INTRODUCTION

The simulation of the dynamics of multi-rigid-body systems is an useful tool in many areas, such as CAD, engineering, virtual reality, videogames and in computergraphics in general (for instance, when physical simulation is used for special effects in 3D movies).

Devices composed of rigid bodies interacting through frictional contacts and mechanical joints represent a numerical challenge because of the discontinuous nature of the motion; the dynamics is nonsmooth because of the discontinuous nature of noninterpenetration, collision, and adhesion constraints. Actually, the requirement that parts must be rigid increases the difficulty of the problem respect to the case of flexible parts such as in spring-based approaches.

Even mechanisms composed of few hundreds of parts and constraints may require lot of computational efforts; indeed, more complicated scenarios such as vehicles running on pebbles and sand such as in Fig. 1, soil and rock dynamics, flow and packing of granular materials, could require too long computational times. Results reported in [Mad07a] indicate that the most widely used commercial software for multibody dynamics runs into significant difficulties when handling simple problems involving hundreds of contact events, whereas cases with thousands of contacts become intractable. The method embraced in this work can solve efficiently problems with millions of contacts on a simple scalar CPU of the Pentium family, and an improved

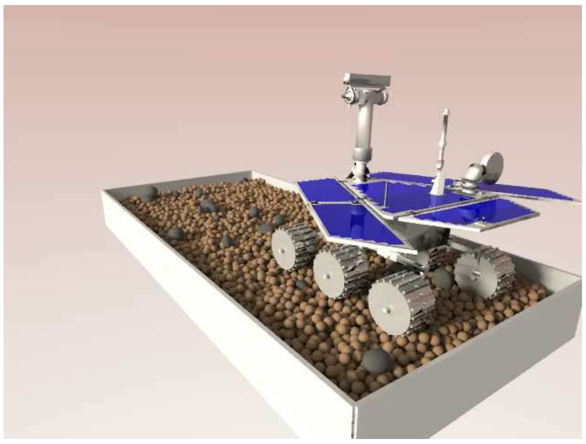
Figure 1: Simulation of a complex multi-rigid-body mechanism with contacts and joints.

performance can be obtained with the GPU version proposed herein, that can solve the contact dynamics with parallel computation.

Until recently, the massive computational power of parallel supercomputers has been available to a relatively small number of research groups, thus limiting the number of applications approached. This scenario is rapidly changing due to a trend set by general-purpose computing on the graphics processing unit (GPU). The CUDA libraries from NVIDIA allow to use the streaming microprocessors mounted in high-end graphics cards as general-purpose computing hardware. Presently, the raw computational power of these multiprocessors is measured in terms of Teraflops, that is hundreds of times the throughput of a modern scalar CPU.

Very few GPU projects are concerned with the dynamics of multibody systems and the two most significant are the Havok and the Ageia physics engines. Both are commercial and proprietary libraries used in the video-game industry. In this context, the goal of this work was to implement a general-purpose multibody solver on GPU multiprocessors backed by convergence results that guarantee the accuracy of the solution. Specifically, a parallel version was implemented of a numerical scheme presented in [Tas08a, Ani08a], which can robustly and efficiently approximate the bilaterally constrained dynamics of rigid bodies undergoing frictional contacts.

Unlike the so-called penalty or regularization methods, where the frictional interaction can be represented by a collection of stiff springs combined with damping elements that act at the interface of the two bodies, the approach embraced herein relies on a different mathematical framework. Specifically, the algorithms rely on time-stepping procedures producing weak solutions of the differential variational inequality (DVI) problem that describes the time evolution of rigid bodies with impact, contact, friction, and bilateral constraints. When compared to penalty-methods, the DVI approach has a greater algorithmic complexity, but avoids the small time steps that plague the former approach.

Early numerical methods based on DVI formulations can be traced back to [Mor83a, Lot82a, Mon93a], while the DVI formulation has been recently classified by differential index in [Pan03a]. Recent approaches based on time-stepping schemes have included both acceleration-force linear complementarity problem (LCP) approaches [Bar93a, Pan96a] and velocity-impulse LCP-based time-stepping methods [Ste96a, Ani97a, Ste00a]. Impulse-based methods, such as the one in [Ben07a], are becoming popular in the computer graphics field because of their robustness. 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 representing the reaction forces [Lot82a, Ste96a]. If the simulation entails a large number of contacts and rigid bodies, as is the case of granular materials, the computational burden of classical LCP solvers can become significant. Indeed, a well-known class of numerical methods for LCPs based on *simplex methods*, also known as *direct* or *pivoting* methods [Cot68a], may exhibit exponential worst-case complexity [Bar94a]. Moreover, the three-dimensional Coulomb friction case leads to a nonlinear complementarity problem (NCP): the use of a polyhedral approximation to transform the NCP into an LCP introduces unwanted anisotropy in friction cones [Ste96a, Ani97a].

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 has been proposed, developed, and tested in [Ani08a]. The method is based on a time-stepping formulation that solves at every step a cone constrained optimization problem [Ani04a]. The time-stepping scheme has been proved to converge in a measure differential inclusion sense to the solution of the original continuous-time DVI. This paper illustrates how this problem can be solved in parallel by exploiting the parallel computational resources available on NVIDIA's GPU cards.

2 FORMULATION OF MULTIBODY DYNAMICS

The formulation of the equations of motion, that is the equations that govern the time evolution of a multibody system, is based on the so-called absolute, or Cartesian, representation of the attitude of each rigid body in the system.

The state of the system is denoted by the generalized positions $\mathbf{q} = \left[\mathbf{r}_1^T, \varepsilon_1^T, \dots, \mathbf{r}_{n_b}^T, \varepsilon_{n_b}^T\right]^T \in \mathbb{R}^{7n_b}$ and their time derivatives $\dot{\mathbf{q}} = \left[\dot{\mathbf{r}}_1^T, \dot{\varepsilon}_1^T, \dots, \dot{\mathbf{r}}_{n_b}^T, \dot{\varepsilon}_{n_b}^T\right]^T \in \mathbb{R}^{7n_b}$, where n_b is the number of bodies, $\mathbf{r}_j$ is the absolute position of the center of mass of the j-th body and the quaternions ε_j are used to represent rotation and to avoid singularities. Instead of using quaternion derivatives in $\dot{\mathbf{q}}$, it is more advantageous to work with angular velocities: the method described will use the vector of generalized velocities $\mathbf{v} = \left[\dot{\mathbf{r}}_1^T, \bar{\omega}_1^T, \dots, \dot{\mathbf{r}}_{n_b}^T, \bar{\omega}_{n_b}^T\right]^T \in \mathbb{R}^{6n_b}$. Note that the generalized velocity can be easily obtained as $\dot{\mathbf{q}} = \mathbf{L}(\mathbf{q})\mathbf{v}$, where $\mathbf{L}$ is a linear mapping that transforms each $\bar{\omega}_i$ into the corresponding quaternion derivative $\dot{\varepsilon}_i$ by means of the linear algebra formula $\dot{\varepsilon}_i = \frac{1}{2}\mathbf{G}^T(\mathbf{q})\bar{\omega}_i$, with 3x4 matrix $\mathbf{G}(\mathbf{q})$ as defined in [Sha05a].

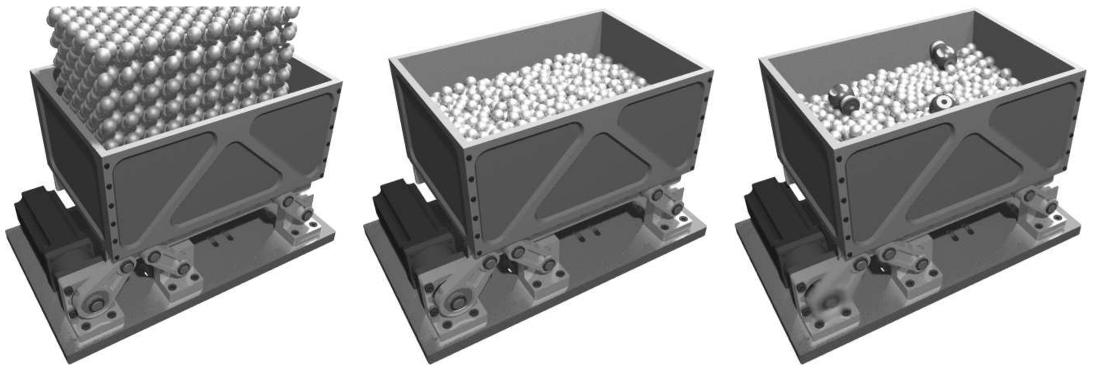
Figure 2: The proposed method can simulate the dynamics of devices with motors, joints and contacts, such as in the case of this size segregation machine that shakes thousands of steel spheres.

We denote by $\mathbf{f}(t, \mathbf{q}, \mathbf{v})$ the set of applied, or external, generalized forces.

Bilateral constraints

Bilateral constraints represent kinematic pairs, for example spherical, prismatic or revolute joints, and can be expressed as algebraic equations constraining the relative position of two bodies. Assuming a set $\mathcal{B}$ of constraints is present in the system, they lead to the scalar equations $\Psi_i(\mathbf{q}, t) = 0, \quad i \in \mathcal{B}$. Assuming smoothness of constraint manifold, $\Psi_i(\mathbf{q}, t)$ can be differentiated to obtain the Jacobian $\nabla_q \Psi_i = [\partial \Psi_i / \partial \mathbf{q}]^T$.

Constraints are consistent at velocity-level provided that $\nabla \Psi_i^T \mathbf{v} + \frac{\partial \Psi_i}{\partial t} = 0$, where $\nabla \Psi_i^T = \nabla_q \Psi_i^T \mathbf{L}(\mathbf{q})$.

Contacts with friction

Given a large number of rigid bodies with different shapes, modern collision detection algorithms are able to find efficiently a set of contact points, that is points where a *gap function* $\Phi(\mathbf{q})$ can be defined for each pair of near-enough shape features. Where defined, such a gap function must satisfy the non-penetration condition $\Phi(\mathbf{q}) \geq 0$ for all contact points.

Note that a signed distance function, differentiable at least up to some value of the interpenetration, can be easily defined if bodies are smooth and convex [Gue03a]. However, this is not always possible, for instance when dealing with concave or faceted shapes often used to represent parts of mechanical devices.

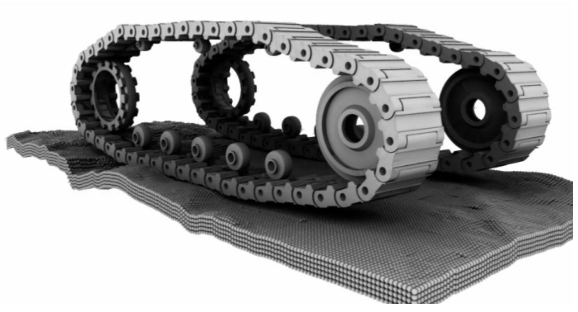
Figure 3: Simulation of a tracked vehicle on a granular soil: we used the GPU for both dynamics and collision detection between tracks, sprockets and pebbles.

When a contact i is active, that is $\Phi_i(\mathbf{q}) = 0$, a normal force and a tangential friction force act on each of the two bodies at the contact point. We use the classical Coulomb friction model to define these forces [Ani97a]. If the contact is not active, that is $\Phi_i(\mathbf{q}) > 0$, no contact forces can exist, and viceversa: this is the Signorini condition $\Phi_i(\mathbf{q}) \geq 0$, $\widehat{\gamma}_{i,n} \geq 0$, $\Phi_i(\mathbf{q})\widehat{\gamma}_{i,n} = 0$ that can be expressed using the complementarity notation [Ste96a]: $\Phi_i(\mathbf{q}) \geq 0 \perp \widehat{\gamma}_{i,n} \geq 0$.

Given two bodies in contact A and B, let $\mathbf{n}_i$ be the normal at the contact pointing toward the exterior of the body of lower index, which by convention is considered to be 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.

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 imposes the following nonlinear constraints:

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

where $\mathbf{v}_{i,T}$ is the relative tangential velocity.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. Note that the friction force depends on the friction coefficient $\mu_i \in \mathbb{R}^+$.

An equivalent convenient way of expressing this constraint is by using the maximum dissipation principle:

$$(\widehat{\gamma}_{i,u}, \widehat{\gamma}_{i,w}) = \underset{\sqrt{\widehat{\gamma}_{i,u}^2 + \widehat{\gamma}_{i,w}^2} \leq \mu_i \widehat{\gamma}_{i,n}}{\operatorname{argmin}} \mathbf{v}_{i,T}^T \left(\widehat{\gamma}_{i,u}\mathbf{u}_i + \widehat{\gamma}_{i,w}\mathbf{w}_i \right). \quad (1)$$

In fact, the the first-order necessary Karush-Kuhn-Tucker conditions for the minimization problem

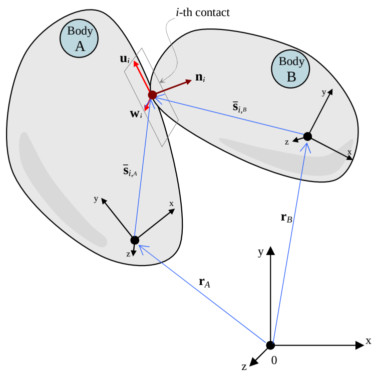

Figure 4: Contact i between two bodies $A,B \in \{1,2,\dots,n_b\}$

(1) correspond to the Coulomb model above [Mor88a, Mon93a].

The complete model

Considering the effects of both the set $\mathbb{A}$ of frictional contacts and the set $\mathbb{B}$ of bilateral constraints, the time evolution of the dynamical system is governed by the following differential variational inequality (a differential problem with set-valued functions and complementarity constraints):

$$
\begin{array}{rcl}
\dot{\mathbf{q}} & = & \mathbf{L}(\mathbf{q})\mathbf{v} \\
\mathbf{M}\dot{\mathbf{v}} & = & \mathbf{f}(t,\mathbf{q},\mathbf{v}) + \sum\limits_{i\in\mathbb{B}} \widehat{\gamma}_{i,b}\nabla\Psi_i + \\
 & & + \sum\limits_{i\in\mathbb{A}} \left(\widehat{\gamma}_{i,n}\mathbf{D}_{i,n} + \widehat{\gamma}_{i,u}\mathbf{D}_{i,u} + \widehat{\gamma}_{i,w}\mathbf{D}_{i,w}\right) \\
i\in\mathbb{B} & : & \Psi_i(\mathbf{q},t) = 0 \\
i\in\mathbb{A} & : & \widehat{\gamma}_{i,n} \geq 0 \perp \Phi_i(\mathbf{q}) \geq 0, \qquad \text{and} \\
(\widehat{\gamma}_{i,u},\widehat{\gamma}_{i,w}) & = & \underset{\mu_i\widehat{\gamma}_{i,n}\geq\sqrt{\widehat{\gamma}_{i,u}^2+\widehat{\gamma}_{i,w}^2}}{\operatorname{argmin}} \mathbf{v}^T\left(\widehat{\gamma}_{i,u}\mathbf{D}_{i,u} + \widehat{\gamma}_{i,w}\mathbf{D}_{i,w}\right)
\end{array}
\tag{2}
$$

The tangent space generators $\mathbf{D}_i = [\mathbf{D}_{i,n}, \mathbf{D}_{i,u}, \mathbf{D}_{i,w}] \in \mathbb{R}^{6n_b\times 3}$ are sparse and are defined given a pair of contacting bodies A and B as:

$$
\mathbf{D}_i^T = \begin{bmatrix} \mathbf{0} & \dots & -\mathbf{A}_{i,p}^T & \mathbf{A}_{i,p}^T\mathbf{A}_A\tilde{\mathbf{s}}_{i,A} & \mathbf{0} & \dots \\ \mathbf{0} & \dots & \mathbf{A}_{i,p}^T & -\mathbf{A}_{i,p}^T\mathbf{A}_B\tilde{\mathbf{s}}_{i,B} & \mathbf{0} & \dots \end{bmatrix} \tag{3}
$$

where we use $\mathbf{A}_{i,p} = [\mathbf{n}_i, \mathbf{u}_i, \mathbf{w}_i]$ as the $\mathbb{R}^{3\times 3}$ matrix of the local coordinates of the ith contact, and introduce the vectors $\bar{\mathbf{s}}_{i,A}$ and $\bar{\mathbf{s}}_{i,B}$ as contact point positions in body coordinates, see Fig. (4), with skew matrices $\tilde{\mathbf{s}}_{i,A}$ and $\tilde{\mathbf{s}}_{i,B}$.

3 THE TIME-STEPPING SCHEME

We formulate the dynamical problem in terms of measure differential inclusions [Ste00a], whose numerical solution can be obtained using the following time-stepping scheme based on the solution of a complementarity problem at each time step.

Given a position $\mathbf{q}^{(l)}$ and velocity $\mathbf{v}^{(l)}$ at the time-step $t^{(l)}$, the numerical solution is found at the new time-step $t^{(l+1)} = t^{(l)} + h$ by solving the following optimization problem with equilibrium constraints [Tas08a]:

$$
\begin{array}{rcl}
\mathbf{M}(\mathbf{v}^{(l+1)} & -\mathbf{v}^{(l)}) = & h\mathbf{f}(t^{(l)},\mathbf{q}^{(l)},\mathbf{v}^{(l)}) + \sum\limits_{i\in\mathbb{B}} \gamma_{i,b}\nabla\Psi_i + \\
 & & + \sum_{i\in\mathbb{A}} \left(\gamma_{i,n}\mathbf{D}_{i,n} + \gamma_{i,u}\mathbf{D}_{i,u} + \gamma_{i,w}\mathbf{D}_{i,w}\right), \quad (4) \\
i\in\mathbb{B} & : & \frac{1}{h}\Psi_i(\mathbf{q}^{(l)},t) + \nabla\Psi_i^T\mathbf{v}^{(l+1)} + \frac{\partial\Psi_i}{\partial t} = 0 \quad (5) \\
i\in\mathbb{A} & : & 0 \leq \frac{1}{h}\Phi_i(\mathbf{q}^{(l)}) + \mathbf{D}_{i,n}^T\mathbf{v}^{(l+1)} \perp \gamma_n^i \geq 0, \quad (6) \\
(\gamma_{i,u},\gamma_{i,w}) & = & \underset{\mu_i\gamma_{i,n}\geq\sqrt{\gamma_{i,u}^2+\gamma_{i,w}^2}}{\operatorname{argmin}} \mathbf{v}^T\left(\gamma_{i,u}\mathbf{D}_{i,u} + \gamma_{i,w}\mathbf{D}_{i,w}\right) \quad (7) \\
\mathbf{q}^{(l+1)} & = & \mathbf{q}^{(l)} + h\mathbf{L}(\mathbf{q}^{(l)})\mathbf{v}^{(l+1)}. \quad (8)
\end{array}
$$

Here, 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, and its effect is discussed in [Ani03a]. Similarly, the term $\frac{1}{h}\Phi_i(\mathbf{q}^{(l)})$ achieves stabilization for bilateral constraints. The scheme converges to the solution of a measure differential inclusion [Ani04a] when the step size $h \rightarrow 0$.

Several numerical methods can be used to solve (4)-(7) [Buc98a]. Our approach casts the problem as a monotone optimization problem by introducing a relaxation over the complementarity constraints, replacing Eq. (6) with $i \in \mathbb{A} : 0 \leq \frac{1}{h}\Phi_i(\mathbf{q}^{(l)}) + \mathbf{D}_{i,n}^T\mathbf{v}^{(l+1)} - \mu_i\sqrt{(\mathbf{v}^T\mathbf{D}_{i,u})^2 + (\mathbf{v}^T\mathbf{D}_{i,w})^2} \perp \gamma_n^i \geq 0$. The solution of the modified time-stepping scheme will approach the solution of the same measure differential inclusion for $h \rightarrow 0$ as the original scheme [Ani04a].

Previous work [Ani08a] showed that the modified scheme is a Cone Complementarity Problem (CCP), which can be solved efficiently by an iterative numerical method that rely on projected contractive maps. Omitting for brevity some of the details discussed in [Ani08a], introducing $\gamma_i = \{\gamma_{i,n}, \gamma_{i,u}, \gamma_{i,w}\}^T, i \in \mathbb{A}$, the algorithm makes use of the following vectors:

$$
\begin{array}{rcll}
\tilde{\mathbf{k}} & \equiv & \mathbf{M}\mathbf{v}^{(l)} + h\mathbf{f}(t^{(l)},\mathbf{q}^{(l)},\mathbf{v}^{(l)}) & (9) \\
\mathbf{b}_i & \equiv & \left\{\frac{1}{h}\Phi_i(\mathbf{q}^{(l)}), 0, 0\right\}^T \quad i \in \mathbb{A}, & (10) \\
b_i & \equiv & \frac{1}{h}\Psi_i(\mathbf{q}^{(l)},t) + \frac{\partial\Psi_i}{\partial t}, \quad i \in \mathbb{B} & (11)
\end{array}
$$

The solution, in terms of dual variables of the CCP (the multipliers), is obtained by iterating the following contraction maps until convergence:

$$
\begin{array}{rcll}
\forall i\in\mathbb{A} & : & \gamma_i^{r+1} = \Pi_{\Upsilon_i}\left[\gamma_i^r - \omega\eta_i\left(\mathbf{D}_i^T\mathbf{v}^r + \mathbf{b}_i\right)\right] & (12) \\
\forall i\in\mathbb{B} & : & \gamma_i^{r+1} = \gamma_i^r - \omega\eta_i\left(\nabla\Psi_i^T\mathbf{v}^r + b_i\right) & (13)
\end{array}
$$

At each iteration r, before repeating (12) and (13), also the primal variables (the velocities) are updated as:

$$
\mathbf{v}^{r+1} = \mathbf{M}^{-1}\left(\sum_{z\in\mathbb{A}} \mathbf{D}_z\gamma_z^{r+1} + \sum_{z\in\mathbb{B}} \nabla\Psi_z\gamma_z^{r+1} + \tilde{\mathbf{k}}\right) \tag{14}
$$

Note that the superscript $(l+1)$ was omitted for brevity.

The iterative process uses the metric projector $\Pi_{\Upsilon_i}(\cdot)$ [Tas08a], which is a non-expansive map $\Pi_{\Upsilon_i}: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ acting on the triplet of multipliers associated with the i-th contact. Thus, if the multipliers fall into the friction cone, they are not modified; if they are in the polar cone, they are set to zero; in the remaining cases they are projected orthogonally onto the surface of the friction cone. The overrelaxation factor ω and η_i parameters are adjusted to control the convergence. Interested readers are referred to [Ani08a] for a proof of the convergence of this method.

The previous algorithm has been implemented on serial computing architectures and proved to be reliable and efficient. In the following the time-consuming part of the methodology, that is the CCP iteration, will be reformulated to take advantage of the parallel computing resources available on GPU boards.

4 PARALLEL SOLVER ON THE GPU

Modern GPU processors can execute thousands of threads in parallel, providing computating power in terms of Teraflops. These processors, usually devoted to the execution of pixel shading fragments for three dimensional visualization, can be also exploited for scientific computation thank to development environments such as CUDA from NVIDIA, that provides C++ functions to easily manage GPU data buffers and *kernels*, that is operations to executed in parallel on the data. The proposed algorithm fits well into the GPU multithreaded model because the computation can be split in multiple threads each acting on a single contact, or kinematic constraint, or rigid body depending on the stage of the computation.

Buffers for data structures

In the proposed approach, the data structures on the GPU are implemented as large arrays (*buffers*) to match the execution model associated with NVIDIA's CUDA. Specifically, threads are grouped in rectangular thread blocks, and thread blocks are arranged in rectangular grids. Four main buffers are used: the contacts buffer, the constraints buffer, the reduction buffer, and the bodies buffer.

Special care should be paid to minimize the memory overhead caused by repeated transfers of large data structures: we organized data structures in a way that minimizes the number of fetch and store operations and maximizes the arithmetic intensity of the kernel code, as recommended by the CUDA development guidelines.

The data structure for the contacts has been mapped into columns of four floats as shown in Fig. 5. 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.

There is no need to store the entire $\mathbf{D}_i$ matrix for the i^{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. In fact, once the velocities of the two bodies $\dot{\mathbf{r}}_{A_i}$, ω_{A_i}, $\dot{\mathbf{r}}_{B_i}$ and ω_{B_i} have been fetched, the product $\mathbf{D}_i^T \mathbf{v}^r$ in Eq.(12) can be performed as

$$\mathbf{D}_i^T \mathbf{v}^r = \mathbf{D}_{i,v_A}^T \dot{\mathbf{r}}_{A_i} + \mathbf{D}_{i,\omega_A}^T \omega_{A_i} + \mathbf{D}_{i,v_B}^T \dot{\mathbf{r}}_{B_i} + \mathbf{D}_{i,\omega_B}^T \omega_{B_i} \tag{15}$$

with the adoption of the following 3x3 matrices

$$\begin{array}{rclrcl} \mathbf{D}_{i,v_A}^T & = & -\mathbf{A}_{i,p}^T, & \mathbf{D}_{i,\omega_A}^T & = & \mathbf{A}_{i,p}^T \mathbf{A}_A \tilde{\mathbf{s}}_{i,A} \\ \mathbf{D}_{i,v_B}^T & = & \mathbf{A}_{i,p}^T, & \mathbf{D}_{i,\omega_B}^T & = & -\mathbf{A}_{i,p}^T \mathbf{A}_B \tilde{\mathbf{s}}_{i,B} \end{array} \tag{16}$$

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.

Also the velocity update vector $\Delta\mathbf{v}_i$, needed for the sum in Eq.(14) is sparse: it can be decomposed in small subvectors. Specifically, given the masses and the inertia tensors of the two bodies m_{A_i}, m_{B_i}, $\mathbf{J}_{A_i}$ and $\mathbf{J}_{B_i}$, the term $\Delta\mathbf{v}_i$ will be computed and stored in four parts as follows:

$$\begin{array}{ll} \Delta\dot{\mathbf{r}}_{A_i} = m_{A_i}^{-1} \mathbf{D}_{i,v_A} \Delta\gamma_i^{r+1}, & \Delta\omega_{A_i} = \mathbf{J}_{A_i}^{-1} \mathbf{D}_{i,\omega_A} \Delta\gamma_i^{r+1} \\ \Delta\dot{\mathbf{r}}_{B_i} = m_{B_i}^{-1} \mathbf{D}_{i,v_B} \Delta\gamma_i^{r+1}, & \Delta\omega_{B_i} = \mathbf{J}_{B_i}^{-1} \mathbf{D}_{i,\omega_B} \Delta\gamma_i^{r+1} \end{array} \tag{17}$$

Note that those four parts of the $\Delta\mathbf{v}_i$ terms are not stored in the i-th contact data structure or in the data structure of the two referenced bodies (because multiple contacts may refer the same body, hence they would overwrite the same memory position). These velocity updates are instead stored in the reduction buffer, which will be used to efficiently perform the summation in Eq.(14). This will be discussed shortly.

The constraints buffer, shown in Fig. 6, is based on a similar concept. Jacobians $\nabla\Psi_i$ of all scalar constraints are stored in a sparse format, each corresponding to four rows $\nabla\Psi_{i,v_A}$, $\nabla\Psi_{i,\omega_A}$, $\nabla\Psi_{i,v_B}$, $\nabla\Psi_{i,\omega_B}$. Therefore the product $\nabla\Psi_i^T \mathbf{v}^r$ in Eq.(13) can be performed as the scalar value $\nabla\Psi_i^T \mathbf{v}^r = \nabla\Psi_{i,v_A}^T \dot{\mathbf{r}}_{A_i} + \nabla\Psi_{i,\omega_A}^T \omega_{A_i} + \nabla\Psi_{i,v_B}^T \dot{\mathbf{r}}_{B_i} + \nabla\Psi_{i,\omega_B}^T \omega_{B_i}$. Also, the four parts of the sparse vector $\Delta\mathbf{v}_i$ can be computed and stored as

$$\begin{array}{ll} \Delta\dot{\mathbf{r}}_{A_i} = m_{A_i}^{-1} \nabla\Psi_{i,v_A} \Delta\gamma_i^{r+1}, & \Delta\omega_{A_i} = \mathbf{J}_{A_i}^{-1} \nabla\Psi_{i,\omega_A} \Delta\gamma_i^{r+1} \\ \Delta\dot{\mathbf{r}}_{B_i} = m_{B_i}^{-1} \nabla\Psi_{i,v_B} \Delta\gamma_i^{r+1}, & \Delta\omega_{B_i} = \mathbf{J}_{B_i}^{-1} \nabla\Psi_{i,\omega_B} \Delta\gamma_i^{r+1} \end{array} \tag{18}$$

Figure 7 shows that each body is represented by a data structure containing the state (velocity and position), the mass moments of inertia and mass values, and the external applied force $\mathbf{F}_j$ and torque $\mathbf{C}_j$. Note that to speed the iteration, it is advantageous 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.

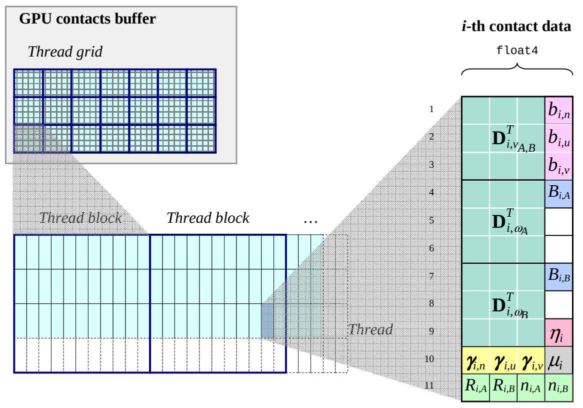

Figure 5: Grid of data structures for frictional contacts, in GPU memory.

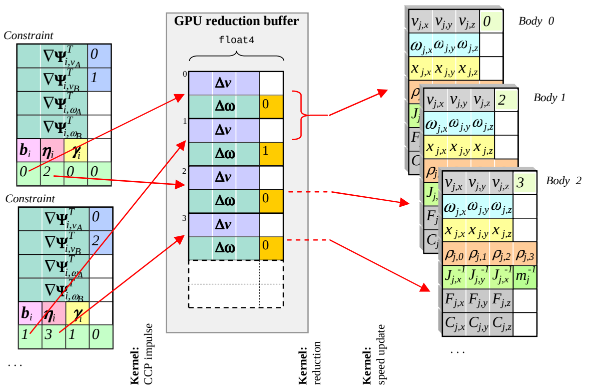

Figure 8: The reduction buffer avoids race conditions in parallel updates of the same body state.

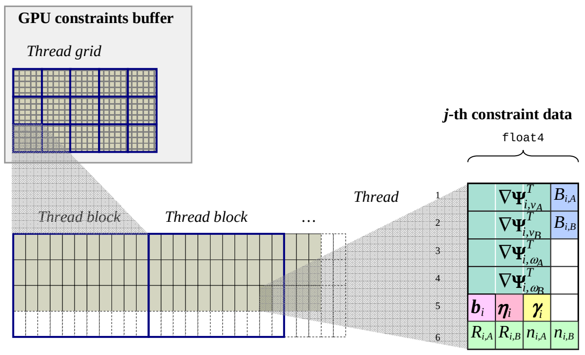

Figure 6: Grid of data structures for scalar constraints, in GPU memory.

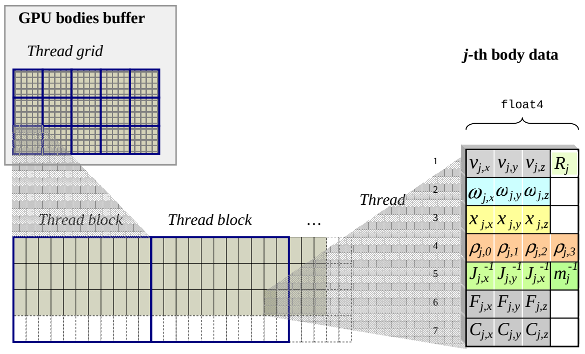

Figure 7: Grid of data structures for rigid bodies, in GPU memory.

The Parallel Algorithm

A parallelization of computations in Eq.(12) and Eq.(13) is easily implemented, by simply assigning one contact per thread (and, similarly, one constraint per thread). In fact the results of these computations would not overlap in memory, and it will never happen that two parallel threads need to write in the same memory location at the same time. These are the two most numerically-intensive steps of the CCP solver, called the **CCP contact iteration kernel** and the **CCP constraint iteration kernel**.

However, the sums in Eq.(14) cannot be performed with embarrassingly-parallel implementations: it may happen that two or more contacts need to add their velocity updates to the same rigid body. A possible approach to overcome this problem is presented in [Har07a], for a similar problem. We adopted an alternative method, with higher generality, based on the *parallel segmented scan* algorithm [Sen07a] that operates on an intermediate reduction buffer (Fig.8); this method sums the values in the buffer using a binary-tree approach that keeps the computational load well balanced among the many thread processors. In the example of Fig.8, the first constraint refers to bodies 0 and 1, the second to bodies 0 and 2; multiple updates to body 0 are then accumulated with parallel segmented reduction.

Since collision detection is the biggest computational overhead after the CCP solution, we also developed a GPU-based parallel code for collision detection, obtaining a 20x speedup factor when compared to the serial code of the Bullet library. The GPU collision code requires the use of multiple kernels and complex data structures that we cannot describe here because of limited space; details are available in [Maz09a].

The following pseudocode shows the sequence of main computational phases at each time step, for the most part executed as parallel kernels on the GPU.

Algorithm 1: Time Stepping using GPU.

1. (*GPU or host*) Perform collision detection between bodies, obtaining $n_{\mathcal{A}}$ possible contact points within a distance δ, as contact positions $s_{i,A}$, $s_{i,B}$ on the two touching surfaces, and normals $\mathbf{n}_i$.

2. (*Host, serial*) If needed, copy contact and body data structures from host memory to GPU buffers. Copy also constraint data (residuals b_i and jacobians) into the constraint buffer.

3. (*GPU, body-parallel*) **Force kernel**. For each body, compute forces $\mathbf{f}(t^{(l)},\mathbf{q}^{(l)},\mathbf{v}^{(l)})$, if any (example, gravity). Store these forces and torques into F_j and C_j.

4. (*GPU, contact-parallel*) **Contact preprocessing kernel**. For each contact, given contact normal and position, compute in-place the matrices $\mathbf{D}_{i,v_A}^T$, $\mathbf{D}_{i,\omega_A}^T$ and $\mathbf{D}_{i,\omega_B}^T$, then compute η_i and the contact residual $\mathbf{b}_i = \{\frac{1}{h}\Phi_i(\mathbf{q}),0,0\}^T$.

5. (*GPU, body-parallel*) **CCP force kernel**. For each body j, initialize body velocities: $\dot{\mathbf{r}}_j^{(l+1)} = h\, m_j^{-1}\mathbf{F}_j$ and $\omega_j^{(l+1)} = h\,\mathbf{J}_j^{-1}\mathbf{C}_j$.

6. (*GPU, contact-parallel*) **CCP contact iteration kernel**. For each contact i, do $\gamma_i^{r+1} = \lambda\ \Pi_{\Upsilon_i}\left(\gamma_i^r - \omega\eta_i\left(\mathbf{D}_i^T\mathbf{v}^r+\mathbf{b}_i\right)\right) + (1-\lambda)\gamma_i^r$. Note that $\mathbf{D}_i^T\mathbf{v}^r$ is evaluated with sparse data, using Eq. (15). Store $\Delta\gamma_i^{r+1} = \gamma_i^{r+1} - \gamma_i^r$ in contact buffer. Compute sparse updates to the velocities of the two connected bodies A and B, and store them in the $R_{i,A}$ and $R_{i,B}$ slots of the reduction buffer.

7. (*GPU, constraint-parallel*) **CCP constraint iteration kernel**. For each constraint i, do $\gamma_i^{r+1} = \lambda\ \left(\gamma_i^r - \omega\eta_i\left(\nabla\Psi_i^T\mathbf{v}^r+b_i\right)\right) + (1-\lambda)\gamma_i^r$. Store $\Delta\gamma_i^{r+1} = \gamma_i^{r+1} - \gamma_i^r$ in contact buffer. Compute sparse updates to the velocities of the two connected bodies A and B, and store them in the $R_{i,A}$ and $R_{i,B}$ slots of the reduction buffer.

8. (*GPU, reduction-slot-parallel*) **Segmented reduction kernel**. Sum all the $\Delta\dot{\mathbf{r}}_i$, $\Delta\omega_i$ terms belonging to the same body, in the reduction buffer.

9. (*GPU, body-parallel*) **Body velocity updates kernel**. For each j body, add the cumulative velocity updates which can be fetched from the reduction buffer, using the index R_j.

10. Repeat from step 6 until convergence or until number of CCP steps reached $r > r_{max}$.

11. (*GPU, body-parallel*) **Time integration kernel**. For each j body, perform time integration as $\mathbf{q}_j^{(l+1)} = \mathbf{q}_j^{(l)} + h\mathbf{L}(\mathbf{q}_j^{(l)})\mathbf{v}_j^{(l+1)}$

12. (*Host, serial*) If needed, copy body, contact and constraint data structures from GPU to host memory.

---

5 IMPLEMENTATION AND RESULTS

The GPU iterative solver and the GPU collision detection have been embedded in our C++ simulation software Chrono::Engine. We tested the GPU-based parallel method with benchmark problems and compared it, in terms of computing time, with the serial method.

Number of bodies	CPU CCP [s]	GPU CCP [s]	Speedup CCP	Speedup CD
16000	7.11	0.57	12.59	4.67
32000	16.01	1.00	16.07	6.14
64000	34.60	1.97	17.58	10.35
128000	76.82	4.55	16.90	21.71

Table 1: Stress test of the GPU CCD solver and GPU collision detection.

For the results of Tab.1, we simulated densely packed spheres that flow from a silo. The CPU is an Intel Xeon 2.66 GHz, the GPU is an NVIDIA Tesla C1060. The simulation time increases linearly with the number of bodies in the model. The GPU algorithm is at least one order of magnitude faster than the serial algorithm.

Other stress tests were performed with even larger amounts of spheres, such as in the benchmark of Fig.10. Similarly, the test of Fig.9 simulates one million of rigid bodies inside a tank being shaken horizontally (the amount of available RAM on a single GPU board limited us to go beyond that limit).

Using the proposed GPU method we are already able to simulate granular soil (pebbles, sand) under the tracks of a vehicle, see Fig.3, in fact our GPU collision detection code is able to handle nonconvex shapes by performing spherical decomposition. To simulate larger scenarios, with smaller grains of sand, future efforts will address the possibility of using domain decomposition, with clusters of multiple GPU boards on multiple host.

6 CONCLUSIONS

A parallel numerical method has been proposed for the simulation of multibody mechanical systems with frictional contacts and bilateral constraints. The parallel method is based on an iterative approach that falls within the mathematical framework of measure differential inclusions and is backed by a rigorous convergence analysis.

Results obtained with the proposed method demonstrate that the GPU version of the dynamics solver is about 20x faster than the CPU version. A similar speedup has been obtained for the collision detection.

7 ACKNOWLEDGEMENTS

We thank the NVIDIA corporation for sponsoring our research programs. Financial support for A.Tasora is provided in part by the Italian Ministry of University under the PRIN grant 2007Z7K4ZB. Mihai Anitescu was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

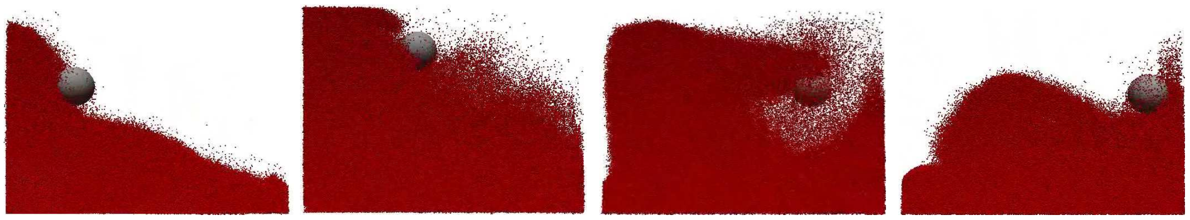
Figure 9: Benchmark with one million of rigid bodies with friction.

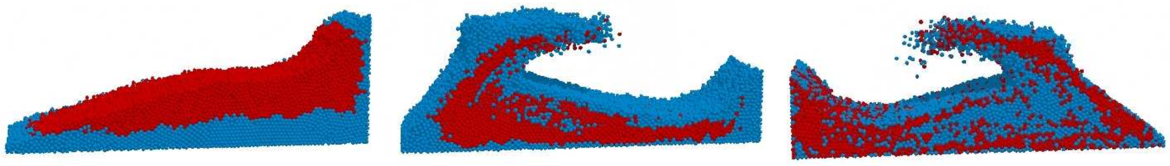
Figure 10: Benchmark: mixing of two granular materials.

REFERENCES

[Ani97a] Anitescu, M. and Potra, F.A. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14:231–247, 1997.

[Ani03a] Anitescu, M. and Hart, G.D. 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.

[Ani04a] Anitescu, M. Optimization-based simulation of non-smooth rigid multibody dynamics. *Math. Program.*, 105(1):113–143, 2006.

[Ani08a] Anitescu, M. and Tasora, A. An iterative approach for cone complementarity problems for nonsmooth dynamics. *Computational Optimization and Applications*, 2008, in press.

[Bar93a] Baraff, D. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10:292–352, 1993.

[Bar94a] Baraff, D. Fast contact force computation for nonpenetrating rigid bodies. In *Computer Graphics (Proceedings of SIGGRAPH)*, pages 23–34, 1994.

[Ben07a] Bender, J. Impulse-based dynamic simulation in linear time. *Journal of Computer Animation and Virtual Worlds*, 18(4,5):225–233, 2007

[Buc98a] Buck, M. and Schömer, E. Interactive Rigid Body Manipulation with Obstacle Contacts. In *6th WSCG Conf. in Central Europe on Computer Graphics and Visualization*, Plzen, 1998.

[Cot68a] Cottle, R.W. and Dantzig, G.B. Complementary pivot theory of mathematical programming. *Linear Algebra and Its Applications*, 1:103–125, 1968.

[Gil88a] Gilbert, E.G., Johnson, D.W. and Keerthi, S.S. A fast procedure for computing the distance between complex objectsin three-dimensional space. *Robotics and Automation, IEEE Journal of [see also IEEE Transactions on Robotics and Automation]*, 4(2):193–203, 1988.

[Gue03a] Guendelman, E., Bridson, R., and Fedkiw, R. Nonconvex rigid bodies with stacking. *ACM Trans. Graph.*, 22(3):871–878, 2003.

[Har07a] Harada, T. Real-time rigid body simulation on gpus. In Hubert Nguyen, editor, *GPU Gems 3*, chapter 23. Addison-Wesley, 2007.

[Lot82a] Lotstedt, P. Mechanical systems of rigid bodies subject to unilateral constraints. *SIAM Journal of Applied Mathematics*, 42(2):281–296, 1982.

[Mad07a] Madsen, J., Pechdimaljian, N. and Negrut, D. Penalty versus complementarity-based frictional contact of rigid bodies: A CPU time comparison. Technical Report TR-2007-05, SBEL, University of Wisconsin, Madison, 2007.

[Maz09a] Mazhar, H. GPU Collision Detection Using Spatial Subdivision With Applications In Contact Dynamics *Proceedings of ASME IDETC09*, San Diego, 2009.

[Mon93a] Monteiro Marques, M.D.P. *Differential Inclusions in Nonsmooth Mechanical Problems: Shocks and Dry Friction*, volume 9 of *Progress in Nonlinear Differential Equations and Their Applications*. Birkhäuser Verlag, Basel, Boston, Berlin, 1993.

[Mor83a] Moreau, Jean J. 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, CISM Courses and Lectures no. 288, Springer–Verlag, 1983.

[Mor88a] Moreau, Jean J. 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, Springer-Verlag, 1988.

[Pan96a] Pang, J.S. and Trinkle, J.C. Complementarity formulations and existence of solutions of dynamic multi-rigid-body contact problems with Coulomb friction. *Math. Program.*, 73(2):199–226, 1996.

[Pan03a] Pang, J.S. and Stewart, D.E. Differential variational inequalities. *Mathematical Programming*, 113(2):345–424, 2008.

[Sha05a] Shabana, A.A. *Dynamics of Multibody Systems*. Cambridge University Press, third edition, 2005.

[Ste96a] Stewart, D.E. and Trinkle, J.C. 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.

[Ste98a] Stewart, D.E. Convergence of a time-stepping scheme for rigid body dynamics and resolution of Painleve's problems. *Arch. Rat. Mech. Anal.*, 145(3):215–260, 1998.

[Ste00a] Stewart, D.E. Rigid-body dynamics with friction and impact. *SIAM Review*, 42(1):3–39, 2000.

[Sen07a] Sengupta, S., Harris, M., Zhang, Y. and Owens, J.D. Scan Primitives for GPU Computing. *ACM Graphics Hardware*, 97–106, 2007.

[Tas08a] Tasora, A. A Fast NCP Solver for Large Rigid-Body Problems with Contacts, Friction, and Joints. In C. L. Bottasso Ed., *Computational Methods in Applied Sciences*, Springer, 12:45–55, 2008.