Mass Splitting for Jitter-Free Parallel Rigid Body Simulation

Richard Tonge
NVIDIA *

Feodor Benevolenski
NVIDIA

Andrey Voroshilov
NVIDIA

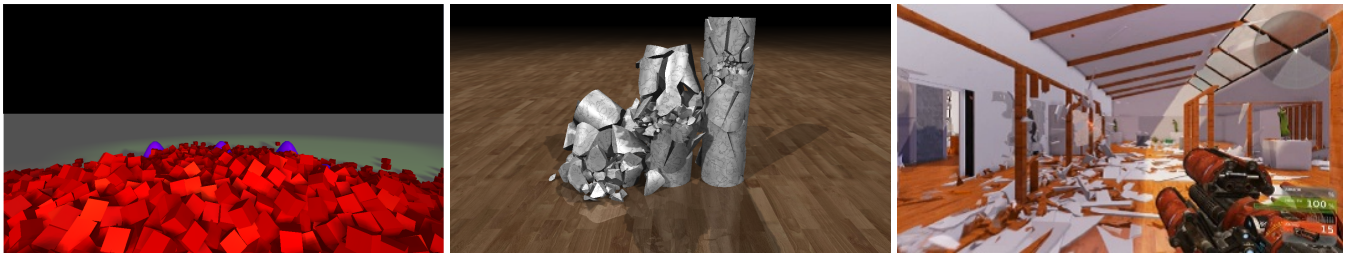

Figure 1: *Left: 5000 boxes with 40000 contacts coming to rest on a non-convex triangle mesh without jitter, simulated on an NVIDIA GTX580 at over 60 FPS. Middle: Mass splitting used in a real-time fracture simulation. The debris have irregular shapes and large mass ratios. Right: The method allows us to simulate large scale building destruction in real-time in a video game.*

Abstract

We present a parallel iterative rigid body solver that avoids common artifacts at low iteration counts. In large or real-time simulations, iteration is often terminated before convergence to maximize scene size. If the distribution of the resulting residual energy varies too much from frame to frame, then bodies close to rest can visibly jitter. Projected Gauss-Seidel (PGS) distributes the residual according to the order in which contacts are processed, and preserving the order in parallel implementations is very challenging. In contrast, Jacobi-based methods provide order independence, but have slower convergence. We accelerate projected Jacobi by dividing each body mass term in the effective mass by the number of contacts acting on the body, but use the full mass to apply impulses. We further accelerate the method by solving contacts in blocks, providing wallclock performance competitive with PGS while avoiding visible artifacts. We prove convergence to the solution of the underlying linear complementarity problem and present results for our GPU implementation, which can simulate a pile of 5000 objects with no visible jittering at over 60 FPS.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: rigid bodies, non-smooth dynamics, contact, friction

Links: DL PDF

*e-mail: rtonge@nvidia.com

ACM Reference Format
Tonge, R., Benevolenski, F., Voroshilov, A. 2012. Mass Splitting for Jitter-Free Parallel Rigid Body Simulation. *ACM Trans. Graph. 31* 4, Article 105 (July 2012), 8 pages. DOI = 10.1145/2185520.2185601 http://doi.acm.org/10.1145/2185520.2185601.

1 Introduction

Rigid body dynamics is widely used in applications ranging from movies to engineering to video games. Piles of objects are particularly common, because ultimately, gravity pulls all rigid bodies to the ground. Some of the most visually interesting simulations involve destruction, such as projectile impacts and explosions, and these can generate large piles of debris. In mechanical engineering some of the most computationally challenging problems involve simulating interaction with large resting systems of soil particles or rocks. Piles require stable simulation of static friction, dynamic friction and resting contact, which presents many challenges.

In large or real-time simulations, the computation budget can be small compared to the number of rigid body contacts. In these cases we must use iterative methods, terminating the iteration before convergence. By stopping early we introduce residual energy into the system, which can cause objects near rest to jitter. See the accompanying video for examples of these artifacts.

A commonly used iterative algorithm is projected Gauss-Seidel (PGS), which solves contacts in sequence. When we terminate the iteration, the last contact solved has no error and the other contacts have non-zero error, resulting in an uneven distribution of the residual energy. On single threaded implementations we can ensure that the distribution of this error is consistent from frame to frame by, for example, ensuring that collisions are detected in the same order each frame and that addition and deletion of bodies do not disrupt the order. Unfortunately, things are not as straightforward for parallel implementations.

Due to the widespread availability of multi-core CPUs and GPUs, parallel computing is increasingly being used to simulate rigid bodies. PGS has limited parallelism, as updates to bodies having multiple contacts must be serialized to ensure they are not lost and the connectivity between bodies can be complex, especially in piles. This serialization changes the order in which constraints are processed, often changing it dramatically from frame to frame. Operating systems and GPU schedulers can also introduce nondeterminism into the order of operations. For these reasons it is very challenging to avoid jittering with parallel PGS, and due to the serialization its performance does not scale well as more threads are added.

In real-time applications such as games, the designer does not know in advance what the player is going to do and which objects are

going to interact. This means that ad-hoc methods that require parameter tuning to converge are not suitable. Ideally, solvers should come with a convergence proof so that the designer can be sure that the simulation will be stable no matter what the player does.

1.1 Contributions

Our first contribution is a projected Jacobi-based method that has guaranteed convergence like PGS, but freedom from jitter and parallel scaling like Jacobi. Our novel idea is to divide each body mass term of the effective mass by the body's contact count, while using the full body mass to apply impulses.

We extend the method to accelerate convergence by solving blocks of contacts with PGS, and using projected Jacobi to combine the blocks. Again we modify the effective mass of the Jacobi iteration, this time dividing the mass terms of the effective mass by the number of contact blocks per body, but again using the full mass to apply impulses. The improved convergence combined with its lower cost per iteration, the method's wall-clock performance is competitive with parallel PGS. The method is especially suited to GPU execution and our GPU implementation of the second method allows real-time (>60 FPS) simulation of 5K bodies with 40K contacts.

Finally, we prove that the block method converges, which is important in open-world applications such as games. We prove that the method converges to the LCP solution, so we get the benefits of the LCP model such as correct momentum propagation and stable stacking. We first give a physical interpretation of the effective mass modification in terms of splitting each body into sub-bodies with equal position and spatial extent, one per block of contacts, with the mass divided equally among them. Running parallel PGS on this split system is operationally identical to our block method, and PGS provably converges, so our method does too. In the supplement we prove that solutions to the split system satisfy the original LCP.

2 Related Work

Many authors have modelled contact dynamics as a complementarity problem [Baraff 1991; Stewart and Trinkle 1996; Tasora et al. 2008]. Early work discretized the contact model at the force-acceleration level, with different cases for colliding and resting contact. Later, time-stepping methods were introduced that unified the modelling of contact and moved to position-impulse [Stewart and Trinkle 1996] or velocity-impulse [Anitescu and Potra 1997] discretization. Systems of velocity-level non-penetration constraints can be modelled as linear complementarity problems, but allow position error, necessitating the use of constraint stabilization [Cline and Pai 2003].

Complementarity problems arising from rigid body systems can be either solved directly or iteratively. Among the iterative solvers, PGS is widely used [Murty 1988; Erleben 2004; Catto 2005; Harada 2009; Tonge et al. 2010]. Recently there has been interest in methods that have better convergence than PGS. Silcowitz et al. [2010] describe a nonsmooth nonlinear conjugate gradient method. Renouf and Alart [2005] describe projected gradient and conjugate projected gradient methods for rigid body LCPs, and Dostal and Schoberl [2005] describe a provably convergent projected conjugate gradient method that could be applied to the rigid body problem. Other methods augment PGS with a different solver to achieve a balance between speed and quality, for example, Morales et al. [2008] interleave PGS solves with exact subspace solves. The advantage of methods based on conjugate gradients is that they have excellent behavior given enough iterations, but the disadvantages are that each iteration is much more expensive than a PGS iteration and their behavior is less good at very low iteration counts.

Jacobi-based solvers are less widely used, as the basic method doesn't converge for some common rigid body systems, and when it does, it is slower than PGS. The LCP literature describes a number of ways to guarantee or improve Jacobi convergence [Cottle et al. 1992; Murty 1988], including line search, block and scaling methods.

Jitter is a common problem in rigid body physics engines [Gustafsson 2010], which particularly affect piles of objects. Many engines [Coumans 2011; NVIDIA 2011; Lengyel 2011] detect when objects should be at rest and temporarily remove them from the simulation, referred to as sleeping. Hsu and Keyser [2010] describe a sleeping method for piles, and Parker and O'Brien [2009] avoid jitter by applying only damping forces at contacts. Guendelman et al. [2003] calculate impulses from the ground up to accelerate convergence when simulating piles.

We describe a method for solving all contacts in parallel in linear time. Other approaches with similar parallel scaling (but different behavioral properties) include the penalty method [Harada 2007], approximate momentum propagation [Kaufman et al. 2005] and oriented particle shape matching [Müller and Chentanez 2011].

Rigid body contact dynamics on parallel computers has a long history, but recently there has been renewed interest due to the arrival of multi-core CPUs and GPUs. Examples of parallel GPU rigid body solvers include Tasora et al. [2008], Harada [2009], Tonge et al. [2010] and Harada [2011].

Kaufman et al. [2008] show how a non-penetration solver can be coupled loosely but robustly with a friction solver, allowing the friction model and solver to be chosen independently from the non-penetration solver. Alternatively, the solution of non-penetration and friction can be tightly coupled. Examples of friction models that have been added to PGS in this way include Tresca friction [Renouf and Alart 2005], polyhedral approximation of Coulomb friction [Stewart and Trinkle 1996], and continuous Coulomb friction [Tasora et al. 2008; Daviet et al. 2011].

3 Background

3.1 Rigid Body Contact Dynamics

The rigid body contact model and discretization that we use have been described before [Anitescu and Potra 1997], so we provide only a brief summary. We consider a scene having n rigid bodies with positions $\mathbf{x} \in \mathbb{R}^{6n}$, external forces $\mathbf{f}_e \in \mathbb{R}^{6n}$ and masses $\mathbf{M} \in \mathbb{R}^{6n \times 6n}$. Collision detection identifies m contacts between the rigid bodies, represented by constraints $\Phi(\mathbf{x}) \geq 0$ with Jacobian $\partial\Phi/\partial\mathbf{x} = \mathbf{J} \in \mathbb{R}^{m \times 6n}$. For simplicity of description we omit constraint stabilization, so the unknowns are the impulses $\mathbf{z} \in \mathbb{R}^m$ necessary to satisfy the time derivative of the constraints. We also omit friction, which we add in section 6.

Contacts must satisfy the velocity Signorini condition: impulses must not be attractive ($\mathbf{z} \geq 0$), velocities must move the system out of penetration ($\mathbf{Jv} \geq 0$), and an impulse should be applied at a contact only if that contact is not separating, written as $\mathbf{z} \geq 0 \perp \mathbf{Jv} \geq 0$. Putting this all together, the continuous model is the following differential variational inequality,

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{J}^T\mathbf{z} + \mathbf{f}_e \tag{1}$$

$$\dot{\mathbf{x}} = \mathbf{v} \tag{2}$$

$$\mathbf{z} \geq 0 \perp \mathbf{Jv} \geq 0. \tag{3}$$

3.2 Time Stepping

We discretize the model using a semi-implicit stepping scheme with time step h,

$$\begin{aligned}\mathbf{M}(\mathbf{v}_{\text{new}}-\mathbf{v}_{\text{old}}) &= \mathbf{J}^T\mathbf{z}+h\mathbf{f}_e\\ \mathbf{x}_{\text{new}}-\mathbf{x}_{\text{old}} &= h\mathbf{v}_{\text{new}}\\ \mathbf{z}\geq 0 \perp \mathbf{J}\mathbf{v}_{\text{new}} &\geq 0.\end{aligned} \tag{4}$$

The Signorini condition causes the discretized model to be an LCP rather than a linear system. Let $\mathbf{x} := \text{LCP}(\mathbf{A},\mathbf{b})$ be defined as

$$\begin{aligned}&\text{find } \mathbf{x}\in\mathbb{R}^m \text{ such that, for all } i=1...m,\\ &\mathbf{x}_i\geq 0 \text{ and } (\mathbf{A}\mathbf{x}+\mathbf{b})_i\geq 0 \text{ and}\\ &\mathbf{x}_i=0 \text{ or } (\mathbf{A}\mathbf{x}+\mathbf{b})_i=0.\end{aligned} \tag{5}$$

So we can solve the discretized model with

$$\begin{aligned}\mathbf{v}^0 &= \mathbf{v}_{\text{old}}+h\mathbf{M}^{-1}\mathbf{f}_e\\ \mathbf{q} &= \mathbf{J}\mathbf{v}^0\\ \mathbf{N} &= \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\\ \mathbf{z} &= \text{LCP}(\mathbf{N},\mathbf{q}).\end{aligned} \tag{6}$$

There are two well-known iterative methods for solving this symmetric LCP, projected Gauss-Seidel (PGS) and projected Jacobi.

3.3 Projected Jacobi

Projected Jacobi solves each constraint in isolation and then merges the results, allowing efficient parallel implementation. Abstractly, it generates the following iteration,

$$\mathbf{z}^{r+1} = (\mathbf{z}^r-\mathbf{D}^{-1}(\mathbf{N}\mathbf{z}^r+\mathbf{q}))^+, \tag{7}$$

where $\mathbf{D}=\text{diag}(\mathbf{N})$ and $(\mathbf{x}^+)_i=\max(0,\mathbf{x}_i)$. In the rigid body model, $\mathbf{D}_i^{-1}$ can be interpreted as being the effective mass along the contact normal, that is, the mapping between constraint space velocity and constraint space impulse at contact i,

$$\mathbf{E}_{ii}=\mathbf{D}_{ii}^{-1}=\left(\textstyle\sum_{j=1}^n \mathbf{J}_{ij}\mathbf{M}_j^{-1}\mathbf{J}_{ij}^T\right)^{-1}. \tag{8}$$

Rather than implementing Equation 7 directly, applications typically update velocities at each iteration in order to reduce storage and computation costs,

$$\begin{aligned}\mathbf{z}^{r+1} &= (\mathbf{z}^r-\mathbf{E}(\mathbf{q}+\mathbf{J}\mathbf{v}^r))^+\\ \mathbf{v}^{r+1} &= \mathbf{v}^r+\mathbf{M}^{-1}\mathbf{J}^T(\mathbf{z}^{r+1}-\mathbf{z}^r).\end{aligned} \tag{9}$$

Convergence is guaranteed if $2\mathbf{D}-\mathbf{N}$ is positive definite [Murty 1988], but many rigid body systems don't converge with Jacobi.

3.3.1 Modified Jacobi Methods

There are a number of ways to modify Jacobi to guarantee convergence. One possibility [Cottle et al. 1992] is to add a line search step,

$$\begin{aligned}\mathbf{z}^{r+0.5} &= (\mathbf{z}^r-\mathbf{D}^{-1}(\mathbf{N}\mathbf{z}^r+\mathbf{q}))^+\\ \mathbf{d}^r &= \mathbf{z}^{r+0.5}-\mathbf{z}^r\\ \alpha &= \underset{\substack{0\leq\alpha\\ 0\leq\mathbf{z}+\alpha\mathbf{d}^r}}{\text{argmin}}\ f(\mathbf{z}+\alpha\mathbf{d}^r)\\ f(\mathbf{z}) &= \tfrac{1}{2}\mathbf{z}^T\mathbf{N}\mathbf{z}+\mathbf{z}^T\mathbf{q}\\ \mathbf{z}^{r+1} &= \mathbf{z}^r+\alpha\mathbf{d}^r.\end{aligned} \tag{10}$$

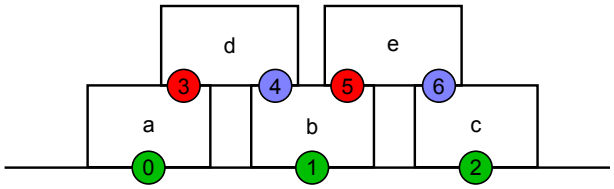

Figure 2: *Coloring for parallel projected Gauss-Seidel. Three colors are needed to avoid parallel threads writing to the same bodies, first constraints {0,1,2} are processed, then {3,5} and finally {4,6}.*

Another is to scale $\mathbf{D}^{-1}$ using $\rho(\mathbf{N})$, the spectral radius of $\mathbf{N}$

$$\begin{aligned}\mathbf{z}^{r+1} &= (\mathbf{z}^r-\omega\mathbf{D}^{-1}(\mathbf{N}\mathbf{z}^r+\mathbf{q}))^+\\ \omega &< \tfrac{2}{\rho(N)}\end{aligned} \tag{11}$$

Bridson et al. [2002] proposed a Jacobi modification for cloth simulation that can be adapted to the rigid body iteration (Equation 9): Let n_{b_β} be the number of bodies contacting body β and divide the impulse applied to body β by n_{b_β},

$$\begin{aligned}\mathbf{z}_{r+1} &= (\mathbf{z}_r-\mathbf{E}(\mathbf{q}+\mathbf{J}\mathbf{v}^r))^+\\ \mathbf{v}^{r+1} &= \mathbf{v}^r-\mathbf{M}^{-1}\text{diag}(n_1^{-1},\cdots,n_n^{-1})(\mathbf{z}_{r+1}-\mathbf{z}_r).\end{aligned} \tag{12}$$

However, Bridson doesn't specify the convergence conditions for the method or give a proof.

3.4 Projected Gauss-Seidel

PGS solves each constraint individually in sequence. Let $\mathbf{L}$ be the (strict) lower triangle of $\mathbf{N}$. Abstractly, PGS generates the following iteration,

$$\mathbf{z}^{r+1} = (\mathbf{z}^r-\mathbf{D}^{-1}(\mathbf{q}+\mathbf{L}\mathbf{z}^{r+1}+(\mathbf{N}-\mathbf{L})\mathbf{z}^r))^+. \tag{13}$$

Again, applications generally use an equivalent iteration [Erleben 2004; Tonge et al. 2004] that updates the velocity after processing each constraint,

$$\begin{aligned}\mathbf{z}_i^{r+1} &= (\mathbf{z}_i^r-\mathbf{E}_{ii}(\mathbf{q}_i+\mathbf{J}_i\mathbf{v}^{rm+i-1}))^+\\ \mathbf{v}^{rm+i} &= \mathbf{v}^{rm+i-1}+\mathbf{M}^{-1}\mathbf{J}_i^T(\mathbf{z}_i^{r+1}-\mathbf{z}_i^r).\end{aligned} \tag{14}$$

The size of $\mathbf{z}$ is m (the number of contacts), and the indexing of $\mathbf{v}$ is arranged so that the velocity used to calculate the first element of each iteration ($i=1$) is the velocity calculated after the last element ($i=m$) of the previous iteration.

3.4.1 Parallel Projected Gauss-Seidel

Ideally we'd like to process each contact in parallel on a separate thread. However, each body can be in contact with more than one other body, so if PGS were parallelized naively then multiple threads could try to update the same body simultaneously, causing velocity updates to be lost and convergence to be affected. A common solution is contact coloring [Hege and Stüben 1991; Harada 2011]. Contact coloring partitions (or colors) the contacts so that each body is referenced at most once in each partition. The PGS solver can then process each color sequentially, processing the contacts within each color in parallel. See Figure 2 for an example of coloring.

The number of colors required depends on Δ, the maximum number of contacts per body over all the bodies. From Vizing's theorem

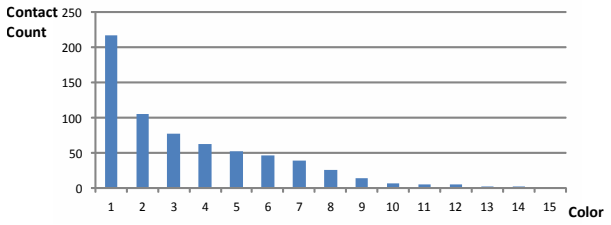

Figure 3: *Histogram of number of constraints processed in parallel per color for the frame shown in Figure 1, middle. This example has 277 bodies and 665 contacts and the greedy coloring algorithm produces 15 colors.*

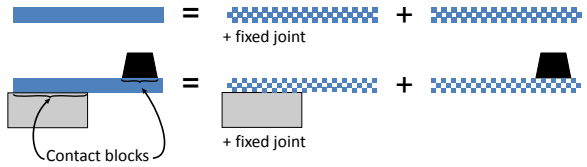

Figure 4: ***Mass splitting.*** *Top: The original body is split into two sub-bodies, each with the same spatial extent but with half the mass. A fixed joint holds them together in the same place. Bottom: A system with two contact blocks, each assigned to a separate sub-body so that the blocks can be solved in parallel.*

[1964] the minimum number of colors required is either Δ or $\Delta+1$. Finding the optimal coloring is NP-hard so we use a parallel greedy algorithm, which gives at most $2\Delta-1$ colors. In piles of objects, Δ can be quite high and generally, the first few colors contain a large proportion of the contacts, but the following colors contain fewer and fewer contacts, see Figure 3. As the colors must be processed sequentially, the time taken by each iteration is multiplied by the number of colors, reducing the speedup of parallel PGS versus serial PGS. Also, on architectures (such as GPUs) with hundreds of cores, most of them are idle for most of the iteration. In our method we harness this unused processing power to reduce jitter.

4 Mass Splitting Solver

Our novel idea is to take the Jacobi algorithm and divide each body mass term in the effective mass (Equation 8) by the number of contacts affecting it, but use the full mass to apply the impulses. In Section 5 we show why this is an appropriate choice. Let n_i be the number of contacts involving body i, then the mass splitting algorithm is

$$\begin{aligned}\mathbf{z}_{r+1} &= \left(\mathbf{z}_r - \mathbf{E}^S(\mathbf{q}+\mathbf{J}\mathbf{v}^r)\right)^+ \\ \mathbf{v}^{r+1} &= \mathbf{v}^r + \mathbf{M}^{-1}(\mathbf{z}_{r+1}-\mathbf{z}_r) \\ \mathbf{E}^S_{ii} &= \left(\textstyle\sum_{j=1}^n n_j \mathbf{J}_{ij}\mathbf{M}_j^{-1}\mathbf{J}_{ij}^T\right)^{-1}. \end{aligned} \tag{15}$$

This is different from Bridson's method (Equation 12) where the impulse application is scaled, but the effective mass uses the full mass. We prove that our method converges in the supplement.

4.1 Block Mass Splitting Solver

A standard method for parallelizing solvers is to partition a system into blocks, solving the blocks on separate threads using one method and then combining the results in an outer iteration, possibly using a different method. Such block splitting methods for LCP are described in Cottle et al. [1992]. We improve upon the convergence of our basic method by solving contacts in blocks, using PGS to solve the contacts within the blocks, and projected Jacobi to combine the blocks. Solving all the contacts in a block on a single thread allows us to use serial PGS, which converges faster than Jacobi. The collision detection system typically generates more than one (point) contact to represent the area of contact between each body pair, so a natural choice is to partition the contacts by body pair. See Figure 5 for an example. Again we modify the effective mass to accelerate convergence, but now we divide the mass of each body β by n_{b_β}, the number of body pairs it belongs to,

$$\begin{aligned}\mathbf{z}_\alpha^{r+1} &= \left(\mathbf{z}_\alpha^r - \hat{\mathbf{E}}_\alpha\left(\mathbf{q}_\alpha + \mathbf{L}_\alpha\mathbf{z}_\alpha^{r+1} + (\mathbf{N}_\alpha-\mathbf{L}_\alpha)\mathbf{z}_\alpha^r + \sum_{\gamma\neq\alpha}\mathbf{N}_{\alpha\gamma}\mathbf{z}_\gamma^r\right)\right)^+ \\ [\hat{\mathbf{E}}_\alpha]_{ii} &= \left(\textstyle\sum_{\beta=1}^n n_{b_\beta}[\mathbf{J}_{\alpha\beta}]_i\mathbf{M}_\beta^{-1}[\mathbf{J}_{\alpha\beta}]_i^T\right)^{-1}. \end{aligned} \tag{16}$$

Here, α and γ are used to index contact blocks and β is used to index bodies. $[M_{\alpha\beta}]_{ij}$ represents element i,j of block α,β of matrix M. In particular, $[\mathbf{J}_{\alpha\beta}]_i$ represents the effect of contact i from contact block α on body β. We describe how we derived this solver in the next section, see Algorithm 1 for the final pseudocode.

5 Derivation and Convergence

Unlike the scalar algorithm, we do not have a direct convergence proof for the block solver. Instead, we transform the original system into a system of split masses and fixed joints. We show that solving this system with PGS interleaved with an exact fixed joint solver converges and is operationally equivalent to running the block solver on the original system. Given that this interleaved solver provably converges, then our block algorithm must also converge, and in the supplement we prove that solutions of the split system are also solutions of the original system. This also demonstrates why scaling the terms of the effective mass, but not the applied impulses, was the right choice in section 4.

We construct the split system by splitting each original body into sub-bodies, one for each contacting body. Each sub-body has the same spatial extent as the original body, but the mass of the original body is divided equally between the sub-bodies. We then add fixed joints to ensure that the position and orientation of all the sub-bodies are equal. See Figure 4 for an example.

The key insight is that fixed joints have a closed form solution: the final velocity of every sub-body should be the average of all the sub-body velocities.

5.1 Splitting the Bodies

We now describe in detail how to construct the split system from the original system. Let n_{b_β} be the number of bodies contacting original body β. We split body β into n_{b_β} sub-bodies, giving a total of

$$n_s = \sum_{\beta=1}^{n} n_{b_\beta}. \tag{17}$$

We split the mass and inertia of body β evenly between its sub-bodies, so the inverse mass matrix of the sub-bodies of body β is

$$\mathbf{W}_\beta = n_{b_\beta}\,\mathrm{diag}\left(\mathbf{M}_\beta^{-1},\cdots,\mathbf{M}_\beta^{-1}\right) \in \mathbb{R}^{6n_{b_\beta}\times 6n_{b_\beta}}, \tag{18}$$

and the inverse mass matrix of the whole split system is

$$\mathbf{W} = \mathrm{diag}\left(\mathbf{W}_1,\cdots,\mathbf{W}_n\right). \tag{19}$$

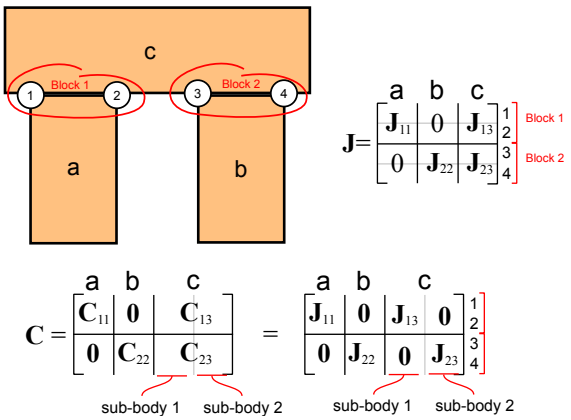

Figure 5: ***Block Mass Splitting****. Top Left: System of three bodies with four contacts. The contacts are grouped into two blocks of two. In the split system, body c is split into two subbodies, one contacting body a, and one contacting body b. Top Right: Contact Jacobian for the original system. Bottom: Contact Jacobian for the split system.*

Each sub-body is just a portion of the mass of the original body, so the intial velocity of each sub-body $\mathbf{V}_\beta$ is set to the initial velocity of the original body β, giving whole system velocity $\mathbf{v}_s$,

$$\mathbf{V}_\beta = \begin{bmatrix} \mathbf{v}_\beta \\ \vdots \\ \mathbf{v}_\beta \end{bmatrix} \in \mathbb{R}^{6n_i}, \mathbf{v}_s = \begin{bmatrix} \mathbf{V}_1 \\ \vdots \\ \mathbf{V}_n \end{bmatrix}. \tag{20}$$

The unconstrained velocity, $\mathbf{v}_{s0}$ is constructed from $\mathbf{v}_s$ in the same way.

5.2 Assigning the Contacts to the Split Bodies

Let the total number of contact blocks (body pairs) be p and the number of contacts in block i be m_i. We partition the original Jacobian into $p \times n$ blocks, such that $\mathbf{J}_{\alpha\beta} \in \mathbb{R}^{m_i \times 6}$ represents the effect of the contacts in contact block α on body β and

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \cdots & \mathbf{J}_{1n} \\ \vdots & & \vdots \\ \mathbf{J}_{p1} & \cdots & \mathbf{J}_{pn} \end{bmatrix}. \tag{21}$$

In each column block β of $\mathbf{J}$, there are n_{b_β} non-zero blocks, corresponding to the contact blocks that affect body β. Let $r_{j,\beta}$ be the index of the jth non-zero block in column block β of $\mathbf{J}$. We now build the contact Jacobian of the split system, $\mathbf{C}$, in such a way that each sub-body has exactly one contact block,

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{11} & \cdots & \mathbf{C}_{1n} \\ \vdots & & \vdots \\ \mathbf{C}_{p1} & \cdots & \mathbf{C}_{pn} \end{bmatrix}. \tag{22}$$

Each block $\mathbf{C}_{\alpha\beta}$ has n_{b_β} sub-blocks, one for each sub-body of β, and $[\mathbf{C}_{\alpha\beta}]_{1,j}$ represents the effect of contact block α on sub-body j of original body β. We assign the jth non-zero contact block of body β in $\mathbf{J}$ to sub-body j of body β in $\mathbf{C}$,

$$[\mathbf{C}_{\alpha\beta}]_{1,j} = \begin{cases} \mathbf{J}_{\alpha\beta} & \text{if } \alpha = r_{j,\beta} \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

Let $b_{\alpha,1}$ and $b_{\alpha,2}$ be the bodies constrainted by contact block α and $s_{\alpha,1}$ be the rank of contact block α in its first body's constraint list, and $s_{\alpha,2}$ be the rank in its second, so that

$$\mathbf{J}_{\alpha\beta} = \begin{cases} [\mathbf{C}_{\alpha\beta}]_{1,s_{\alpha,1}} & \text{if } b_{\alpha,1} = \beta \\ [\mathbf{C}_{\alpha\beta}]_{1,s_{\alpha,2}} & \text{if } b_{\alpha,2} = \beta. \end{cases} \tag{24}$$

For an example of how to construct $\mathbf{C}$ from $\mathbf{J}$, see Figure 5.

5.3 Joining the Bodies Back Together

A fixed joint forces the position and orientation of two bodies to be the same,

$$\Phi^{FJ}\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \mathbf{x}_1 - \mathbf{x}_2 = 0, \tag{25}$$

and has Jacobian

$$\mathbf{J}^{FJ} = \frac{\partial \Phi^{FJ}}{\partial \mathbf{x}} = \begin{bmatrix} \mathbf{I} & -\mathbf{I} \end{bmatrix}. \tag{26}$$

For each body β we introduce $n_{b_\beta} - 1$ fixed joints to join the sub-bodies together, giving a total of

$$m_f = \sum_{\beta=1}^{n} (n_{b_\beta} - 1). \tag{27}$$

The Jacobian of the fixed joints of body β is

$$\mathbf{F}_\beta = \begin{bmatrix} \mathbf{I}_6 & -\mathbf{I}_6 & 0 & \cdots & 0 \\ \mathbf{I}_6 & 0 & -\mathbf{I}_6 & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ \mathbf{I}_6 & 0 & \cdots & 0 & -\mathbf{I}_6 \end{bmatrix} \in \mathbb{R}^{6n_{b_\beta} \times 6(n_{b_\beta}-1)}, \tag{28}$$

and the Jacobian of all the fixed joints in the split system is

$$\mathbf{F} = \text{diag}\left(\mathbf{F}_1, \cdots, \mathbf{F}_{m_f}\right). \tag{29}$$

5.4 The Split System

We concatenate the split contact constraints and the fixed joints to form system Jacobian $[\mathbf{C}^T \mathbf{F}^T]^T$, and define

$$\begin{bmatrix} \mathbf{N}_{cc} & \mathbf{N}_{cf} \\ \mathbf{N}_{fc} & \mathbf{N}_{ff} \end{bmatrix} = \begin{bmatrix} \mathbf{C} \\ \mathbf{F} \end{bmatrix} \mathbf{W} \begin{bmatrix} \mathbf{C} \\ \mathbf{F} \end{bmatrix}^T \tag{30}$$

$$\mathbf{D}_{cc} = \text{diag}(\mathbf{N}_{cc}). \tag{31}$$

We can find impulses $[\mathbf{z}_c^T \mathbf{z}_f^T]^T$ that satisfy the contacts and fixed joints simultaneously by solving the following mixed linear complementarity problem (MLCP),

$$\begin{aligned} \mathbf{q} + \mathbf{N}_{cc}^T \mathbf{z}_c + \mathbf{N}_{cf}^T \mathbf{z}_f \geq 0 \quad \perp \quad \mathbf{z}_c \geq 0 \\ \mathbf{N}_{fc}^T \mathbf{z}_c + \mathbf{N}_{ff}^T \mathbf{z}_f = 0. \end{aligned} \tag{32}$$

This MLCP can be solved by interleaving PGS iterations for the contacts with direct (exact) solves for the joints [Lacoursière 2007],

$$\begin{aligned} \mathbf{z}_c^{r+1} &= \left(\mathbf{z}_c^r - \mathbf{D}_{cc}^{-1}(\mathbf{q} + \mathbf{N}_{cc}\mathbf{z}_c + \mathbf{N}_{cf}\mathbf{z}_f)\right)^+ \\ \mathbf{z}_f^{r+1} &= \text{linsolve}(\mathbf{N}_{ff}, \mathbf{N}_{fc}\mathbf{z}_c). \end{aligned} \tag{33}$$

See the supplementary document for a convergence proof. We transform the iteration to a velocity iteration as follows,

$$\mathbf{v}_s^{0,1} = \mathbf{v}_{s0} \tag{34}$$

$$\mathbf{z}_{c_i}^{r+1} = (\mathbf{z}_{c_i}^r - \mathbf{D}_{cc_{ii}}^{-1}(\mathbf{q}_i + \mathbf{C}_i\mathbf{v}_s^{r,i}))^+ \tag{35}$$

$$\mathbf{v}_s^{r,i+1} = \mathbf{v}_s^{r,i} + \mathbf{W}\mathbf{C}_i^T(\mathbf{z}_{c_i}^{r+1} - \mathbf{z}_{c_i}^r) \tag{36}$$

$$\mathbf{v}_s^{r+1,1} = \mathbf{v}_s^{r,m+1} + \mathbf{W}\mathbf{F}^T \text{linsolve}(\mathbf{N}_{ff}, \mathbf{F}\mathbf{v}_s^{r,m+1}). \tag{37}$$

Equations 35 and 36 apply a single PGS iteration to all the contacts at each iteration. Each sub-body has exactly one constraint, so no coloring is required and the contacts can be safely computed concurrently on m threads.

5.5 Closed Form Solution for Fixed Joints

We could solve Equation 37 using a direct solver, but there is a low-cost analytical solution,

$$\left[(\mathbf{v}_s)_\beta\right]_i^{r+1,1} = \text{average}\left([(\mathbf{v}_s)_\beta]_1^{r,m+1} \cdots [(\mathbf{v}_s)_\beta]_{n_{b_\beta}}^{r,m+1}\right). \tag{38}$$

In other words, the fixed joints can be enforced simply by averaging the sub-body velocities of each body at each iteration (see proof in the supplementary document).

Algorithm 1 Block Mass Splitting Solver

1: **for all** contact blocks, α **do**
2: **for all** contacts in block, i **do**
3: $[\mathbf{p}\alpha]_i = 0$
4: **for all** $\beta \in \{b_{\alpha,1}, b_{\alpha,2}\}$ **do**
5: $[\mathbf{p}\alpha]_i = [\mathbf{p}\alpha]_i + [\mathbf{J}_{\alpha\beta}]_i(n_{b_\beta}\mathbf{M}_\beta^{-1})[\mathbf{J}_{\beta\alpha}]_i^T$
6: **end for**
7: $\mathbf{p}_\alpha = \mathbf{p}_\alpha^{-1}$
8: **end for**
9: **end for**
10: **for all** iterations **do**
11: **for all** contact blocks, α **do** {in parallel}
12: $\beta_1, \beta_2 = b_{\alpha 1}, b_{\alpha 2}$
13: $\mathbf{D}_{\alpha 1}, \mathbf{D}_{\alpha 2} = \mathbf{v}_{\beta_1}, \mathbf{v}_{\beta_2}$
14: **for all** contacts in block, i **do** {sequentially on thread α}
15: $t = [\mathbf{z}_{C\alpha}]_i$
16: $[\mathbf{z}_{C\alpha}]_i = \max\left(0, [\mathbf{z}_{C\alpha}]_i - [\mathbf{p}\alpha]_i(\mathbf{J}_{\alpha,\beta_1}\mathbf{D}_{\alpha_1} + \mathbf{J}_{\alpha,\beta_2}\mathbf{D}_{\alpha_2})\right)$
17: **for all** $j \in \{1,2\}$ **do**
18: $\mathbf{D}_{\alpha j} = \mathbf{D}_{\alpha j} + \mathbf{M}_{\beta_j}^{-1}\mathbf{J}_{\alpha,\beta_j}^T(\mathbf{z}_{C\alpha} - t)$
19: **end for**
20: **end for**
21: **end for**
22: $\mathbf{a} = \mathbf{0}$
23: **for all** $j \in \{1,2\}$ **do**
24: **for all** α such that $b_{\alpha j} = \beta$ **do**
25: $\mathbf{a} = \mathbf{a} + \mathbf{D}_{\alpha j}$
26: **end for**
27: **end for**
28: **end for**

5.6 Implementation and Convergence Proof

The sub-body velocities of each body are equal at the end of each iteration, so we can just store one velocity for each body. Also, we don't need to store the masses of the sub-bodies, since we can just scale the corresponding body mass before it is used. Applying these optimizations we get Equation 16 and making use of the definitions of $\mathbf{C}$, $\mathbf{v}_s$ and $\mathbf{W}$, we get Algorithm 1. The whole algorithm is parallelizable, as lines 22 to 27 can be implemented using a parallel segmented reduction. The algorithm that we've derived is just PGS applied to the split system, so it unconditionally converges, yet it is operationally identical to our block splitting method, Equation 16. Therefore the block splitting method converges. See the supplement for a proof that the block splitting method converges to the solution of the original LCP.

		PGS		Block Mass Splitting	
Test	Bodies	Frame time (ms)	FPS	Frame time (ms)	FPS
Boxes	5160	17.7	57	15.7	64
Chain	25	12.4	81	5.2	193
Chess	1000	24.9	40	25.5	39
Card house	58	4.8	206	3.5	282
Fracture	277	7.8	129	7.0	143

Table 1: *Frame time.*

6 Friction

We use a pyramid approximation to the coulomb friction cone. To add this friction approximation to the discretized system, we need to represent boxed LCPs (BLCPs), which have both upper and lower limits. Let $x := \text{BLCP}(\mathbf{A}, \mathbf{b}, \mathbf{l}, \mathbf{h})$ be defined as

$$\begin{aligned} \text{find } \mathbf{x} \in \mathbb{R}^m \text{ such that, for all } i = 1..m, \\ \mathbf{x}_i = \mathbf{l}_i \text{ and } (\mathbf{A}\mathbf{x}+\mathbf{b})_i \geq 0 \text{ or} \\ \mathbf{x}_i = \mathbf{h}_i \text{ and } (\mathbf{A}\mathbf{x}+\mathbf{b})_i \leq 0 \text{ or} \\ \mathbf{l}_i < \mathbf{x}_i < \mathbf{h}_i \text{ and } (\mathbf{A}\mathbf{x}+\mathbf{b})_i = 0. \end{aligned} \tag{39}$$

All of the LCP algorithms in this paper can be turned into BLCP algorithms by changing $\max(0, \mathbf{x}_i)$ to $\max(\mathbf{l}_i, \min(\mathbf{x}_i, \mathbf{h}_i))$.

Let $\mathbf{D}$ be the Jacobian matrix of the discretized friction pyramid and let $\mathbf{N}_{\text{fric}} = \mathbf{D}\mathbf{M}^{-1}\mathbf{D}^T$. Now we can write the discretized model with friction as a pair of coupled complementarity problems,

$$\mathbf{z} = \text{LCP}\left(\mathbf{N}, \mathbf{J}(\mathbf{v}^0 + \mathbf{M}^{-1}\mathbf{D}^T\mathbf{z}_{\text{fric}})\right) \tag{40}$$

$$\mathbf{z}_{\text{fric}} = \text{BLCP}\left(\mathbf{N}_{\text{fric}}, \mathbf{D}(\mathbf{v}^0 + \mathbf{M}^{-1}\mathbf{J}^T\mathbf{z}), -\text{diag}(\mu)\mathbf{z}, \text{diag}(\mu)\mathbf{z}\right). \tag{41}$$

This is analagous to the pair of coupled projections in the staggered projections model of Kaufman et al. [2008]. In their solver, Kaufman et al. interleave exact solves of the non-penetration subsystem with exact solves of the friction subsytem using the updated normal forces to update the friction force limits between them. In our setting, this would correspond to repeatedly running our mass splitting solver to convergence on Equation 40, followed by running it to convergence on Equation 41. Our goal is to make a solver for the coupled system that executes in as few iterations as possible, so we don't do this. Instead we repeatedly execute a single iteration of the mass splitting solver on the non-penetration LCP, followed by another on the friction BLCP.

7 Results

In this section we present measurements of jitter, convergence and overall performance. We detect jitter by simulating the following scenes, which should come to rest in a short amount of time. As the bodies settle we check how close the total kinetic energy gets to zero. The scenes include benchmarks from previous papers, but note that we are comparing mass splitting against PGS and Jacobi

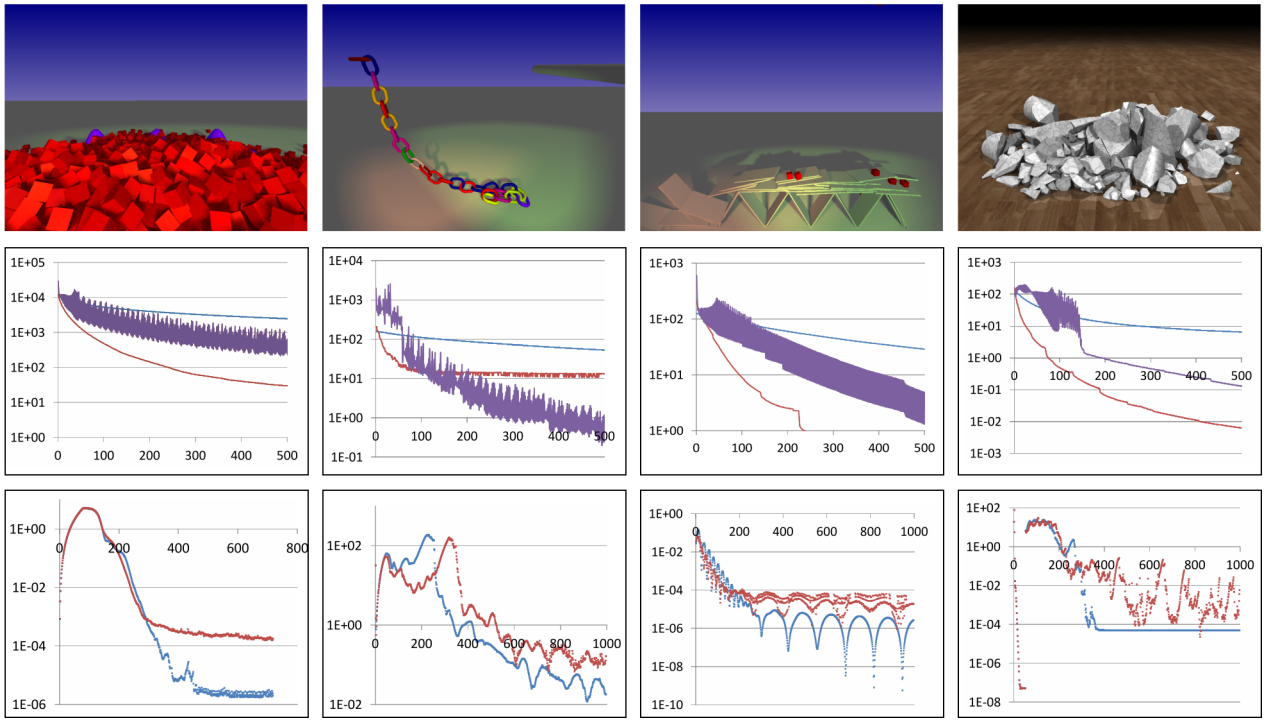

Figure 6: ***Test cases (top row, left to right):*** *Boxes, Chain, Card house and Fracture.* ***Convergence graphs (middle row):*** *average constraint violation versus iteration count.* ***Jitter graphs (bottom row):*** *average kinetic energy vs frame.* ***Color key:*** *Blue, Mass Splitting. Purple, Jacobi Line Search. Red, PGS.*

line search (all with staggered friction), not the solvers introduced in those papers.

Boxes: We drop 5160 boxes onto a non-convex mesh containing 8192 triangles. The kinetic energy should decrease after the objects start hitting the ground, eventually reaching zero.

Chain: We simulate a chain of 25 links fixed at one end. Eventually the links in the free end of the chain should form a stable pile on the ground.

Card house: We simulate a pyramid of 48 cards arranged into five levels, inspired by Kaufman et al. [2008]. This is challenging because the friction and non-penetration forces have to be in perfect balance for the house to stay up. We throw boxes at the card house to check that it can regain equilibrium if only part of it is demolished.

Fracture: We fracture three marble columns into the Voronoi regions of a set of particles placed around a set of impact points. This generates a wide range of body sizes and masses, a configuration that is especially prone to jitter.

Our focus is measuring performance and quality at very low iteration counts. A typical PGS iteration count for real-time applications is five [NVIDIA 2011]. On the boxes test, the largest, parallel PGS with five iterations takes at most 4 mS per frame. We found that 30 iterations of mass splitting can be performed in the same amount of time, largely because mass splitting does not have to color and serialize constraints. So in our jitter tests (Figure 6, bottom), we compare five iterations of PGS against 30 iterations of mass splitting. In all the tests, mass splitting has less jitter than PGS. The kinetic energy does not reach zero with our method, whether we use double or single precision math, but from the video you can see that the level of jitter is below perceptible levels. Table 1 shows the performance of highly optimized CUDA implementations of PGS and mass splitting, run on a GTX580 with 512 cores.

Limitations Although not popular in rigid body animation, the Jacobi line search method [Cottle et al. 1992] does enjoy order independent residual distribution and guaranteed convergence. Figure 6 (middle) compares the convergence of Jacobi line search against PGS and mass splitting. Although our method has better convergence for real-time iteration counts (< 50), Jacobi line search has better convergence for larger iteration counts. The convergence of our method is worse than PGS in all the tests, but this manifests itself as increased contact compliance rather than jitter. This is particularly noticeable in the card house, which visibly bounces before it comes to rest.

8 Conclusion

We have described a new iterative method for solving large rigid body systems that avoids jitter at low iteration counts. This enables the method to be used in large off-line or real-time simulations without having to tune the system to prevent jittering. Our results show that the performance of the solver is comparable with parallel PGS and that a GPU physics engine implemented using the solver can simulate a pile of 5000 bodies with 40000 contacts without jittering at over 60 FPS.

9 Acknowledgements

The authors would like to thank the NVIDIA PhysX team, especially Adam Moravanszky and Pierre Terdiman for the original CPU engine, Kevin Newkirk for video creation and Dilip Sequeira for assistance with editing.The fracture demo was developed by Nuttapong Chentanez and Matthias Müller-Fischer.

References

ANITESCU, M., AND POTRA, F. A. 1997. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics 14*, 231–247.

BARAFF, D. 1991. Coping with friction for non-penetrating rigid body simulation. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '91, 31–41.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *Proceedings of ACM Siggraph*, 594–603.

CATTO, E. 2005. Iterative dynamics with temporal coherence. Presented at the Game Developers Conference.

CLINE, M. B., AND PAI, D. K. 2003. Post-stabilization for rigid body simulation with contact and constraints. In *ICRA*, IEEE, 3744–3751.

COTTLE, R., PANG, J.-S., AND STONE, R. 1992. *The Linear Complementarity Problem*. Academic Press.

COUMANS, E. 2011. Game physics artefacts. Presented at the Game Developers Conference.

DAVIET, G., BERTAILS-DESCOUBES, F., AND BOISSIEUX, L. 2011. A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, ACM, New York, NY, USA, SA '11, 139:1–139:12.

DOSTAL, Z., AND SCHOBERL, J. 2005. Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination. *Computational Optimization and Applications 30*, 23–43.

ERLEBEN, K. 2004. *Stable , Robust , and Versatile Multibody Dynamics Animation*. PhD thesis, University of Copenhagen Copenhagen.

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

GUSTAFSSON, D. 2010. Understanding game physics artefacts. In *Game Physics Pearls*. A K Peters, ch. 2.

HARADA, T. 2007. Real-time rigid body simulation on GPUs. In *GPU gems 3*. Addison-Wesley Professional, ch. 29.

HARADA, T. 2009. Parallelizing the physics pipeline: Physics simulations on the GPU. Presented at the Game Developers Conference.

HARADA, T. 2011. A parallel constraint solver for a rigid body simulation. Presented at SIGGRAPH Asia.

HEGE, H.-C., AND STÜBEN, H. 1991. Vectorization and parallelization of irregular problems via graph coloring. In *ICS*, 47–56.

HSU, S.-W., AND KEYSER, J. 2010. Piles of objects. In *ACM SIGGRAPH Asia 2010 papers*, ACM, New York, NY, USA, SIGGRAPH ASIA '10, 155:1–155:6.

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

KAUFMAN, D. M., SUEDA, S., JAMES, D. L., AND PAI, D. K. 2008. Staggered projections for frictional contact in multibody systems. In *ACM SIGGRAPH Asia 2008 papers*, ACM, New York, NY, USA, SIGGRAPH Asia '08, 164:1–164:11.

LACOURSIÈRE, C. 2007. A parallel block iterative method for interactive contacting rigid multibody simulations on multicore pcs. In *Proceedings of the 8th international conference on Applied parallel computing: state of the art in scientific computing*, Springer-Verlag, Berlin, Heidelberg, PARA'06, 956–965.

LENGYEL, E. 2011. A jitter-tolerant rigid body sleep condition. In *Game Engine Gems 2*. A K Peters, ch. 23.

MORALES, J., NOCEDAL, J., AND SMELYANSKIY, M. 2008. An algorithm for the fast solution of symmetric linear complementarity problems. *Numerische Mathematik 111*, 251–266.

MÜLLER, M., AND CHENTANEZ, N. 2011. Solid simulation with oriented particles. In *ACM SIGGRAPH 2011 papers*, ACM, New York, NY, USA, SIGGRAPH '11, 92:1–92:10.

MURTY, K. G. 1988. Iterative methods for LCPs. In *Linear Complementarity, Linear and Non-linear Programming*. Helderman Verlag, ch. 9.

NVIDIA. 2011. *NVIDIA PhysX 3.2 user guide*.

PARKER, E. G., AND O'BRIEN, J. F. 2009. Real-time deformation and fracture in a game environment. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, New York, NY, USA, SCA '09, 165–175.

RENOUF, M., AND ALART, P. 2005. Conjugate gradient type algorithms for frictional multi-contact problems: applications to granular materials. *Computer Methods in Applied Mechanics and Engineering 194*, 18-20, 2019 – 2041.

SILCOWITZ, M., NIEBE, S., AND ERLEBEN, K. 2010. A nonsmooth nonlinear conjugate gradient method for interactive contact force problems. *The Visual Computer 26*, 893–901.

STEWART, D. E., AND TRINKLE, J. C. 1996. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering 39*, 15, 2673–2691.

TASORA, A., NEGRUT, D., AND ANITESCU, M. 2008. A GPU-based implementation of a cone convex complementarity approach for simulating rigid body dynamics with frictional contact. *ASME Conference Proceedings 2008*, 48722, 107–118.

TONGE, R., ZHANG, L., AND SEQUEIRA, D. 2004. Method and program solving LCPs for rigid body dynamics. United States Patent 7079145.

TONGE, R., WYATT, B., AND NICHOLSON, B. 2010. PhysX GPU rigid bodies in Batman: Arkham Asylum. In *Game Programming Gems 8*, A. Lake, Ed. Cengage Learning, ch. 7.2, 590–601.

VIZING, V. G. 1964. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz. 3*, 25–30.