Scalable Solvers for Cone Complementarity Problems in Frictional Multibody Dynamics

Saibal De
University of Michigan
Ann Arbor, MI
saibalde@umich.edu

Eduardo Corona
New York Institute of Technology
Old Westbury, NY
ecorona@nyit.edu

Paramsothy Jayakumar
US Army CCDC GVSC
Warren, MI
paramsothy.jayakumar.civ@mail.mil

Shravan Veerapaneni
University of Michigan
Ann Arbor, MI
shravan@umich.edu

***Abstract*—We present an efficient, hybrid MPI/OpenMP framework for the cone complementarity formulation of large-scale rigid body dynamics problems with frictional contact. Data is partitioned among MPI processes using a Morton encoding in order to promote data locality and minimize communication. We parallelize the state-of-the-art first and second-order solvers for the resulting cone complementarity optimization problems. Our approach is highly scalable, enabling the solution of dense, large-scale multibody problems; a sedimentation simulation involving 256 million particles ($\sim$324 million contacts on average) was resolved using 512 cores in less than half-hour per time-step.**

I. Introduction

The need for high-fidelity, scalable simulation frameworks of granular media has spurred a wave of recent developments in the efficient implementation of discrete element methods (DEM) for many-body frictional contact. The DEM approach tracks the evolution of individual particles due to external body forces and contact forces caused by non-penetration and sliding friction.

Most parallel implementations and software packages available for large-scale granular media simulations employ force penalty methods (DEM-P) [1]–[3]. For pairs of colliding objects, they introduce contact force fields which are easy to implement and inexpensive to evaluate. The fidelity and efficiency of this approach is, however, often limited by the artificial stiffness induced by the spring-like forces used to avoid penetration. A newer class complementarity methods (DEM-C) avoid this by enforcing the contact constraints geometrically [4]–[8]. For an in-depth analysis and comparison of DEM approaches, see [9]. In this work, we present an efficient, hybrid MPI/OpenMP distributed memory implementation of DEM-C methods for frictional dynamics.

For each pair of objects at contact, DEM-C methods introduce a set of complementarity constraints. Given a time-stepping scheme, this results in a nonlinear complementarity problem (NCP) that must be solved at each time step. This NCP may be relaxed into a linear complementarity problem (LCP) [4], [5]; this approach, however, introduces non-homogeneous frictional forces. An alternative relaxation method addressing the limitations of the LCP produces a cone complementarity problem (CCP) for which a wide array of quadratic cone optimization solvers have been proposed [7], [10]–[14]. From extensive comparison in multibody dynamics problems [15], [16], two families of methods have been shown to hold the greatest potential. The first-order accelerated projected gradient descent (APGD) method uses the momentum from previous iterates to greatly reduce iteration counts for the gradient descent steps. This feature makes it the method-of-choice for large-scale systems. Second-order Interior Point (IP) methods display robust, problem-independent convergence, making them clear front-runners for small to moderate-sized systems. In order to remain competitive for large-scale systems, the acceleration of the Newton step sparse linear systems involved is required, as proposed in [16].

In [17], the authors review the state-of-the-art in parallel computing for DEM multibody dynamics simulations. For moderately large granular media problems, their analysis favors a hybrid approach combining SIMD (Single Instruction, Multiple Data) parallelism in the GPU and parallel task management in the CPU via OpenMP. They have implemented the DEM-C approach in the Chrono Parallel library, producing simulation benchmarks of dense granular media for up to $\mathcal{O}(10^6)$ rigid bodies [18]. For larger granular media problems as well as for multi-physics problems involving long-range interactions, the amount of data and variety of tasks involved necessitate a distributed memory approach; the main computational bottleneck in this case is data communication. In [17], a basic Chrono MPI implementation is applied to a vehicle-terrain problem involving 2 million bodies, employing 64 computing cores.

The hybrid MPI/OpenMP framework presented here aims to address the challenges involved in efficient distributed memory implementation of collision detection and resolution via the CCP complementarity approach. We demonstrate favorable performance and parallel scaling for problems up to 256 million rigid bodies and approximately 324 million pairwise contacts employing 512 cores. We reduce the communication between MPI processes by using Morton IDs and ensure rigid bodies that are spatially close end up on the same MPI rank.

II. The Contact Model

A. Equations of Motion

Consider a granular medium consisting of n rigid bodies. We use a generalized coordinate system of dimension $6n$ to describe its dynamics (three translational and three rotational degrees of freedom per body). Let $\boldsymbol{q}$ and $\boldsymbol{v} \in \mathbb{R}^{6n}$ denote

the position and velocity of the system in these generalized coordinates. We describe the time-evolution of these two variables using Newton's equations

$$\mathbf{M}\dot{\boldsymbol{v}} = \boldsymbol{f}_{\text{ext}} + \boldsymbol{f}_{\text{col}}, \quad \dot{\boldsymbol{q}} = \mathbf{L}\boldsymbol{v}.$$

Here, $\boldsymbol{f}_{\text{ext}}$ and $\boldsymbol{f}_{\text{col}} \in \mathbb{R}^{6n}$ represent the external and contact forces, $\mathbf{M} \in \mathbb{R}^{6n\times 6n}$ is the mass matrix and $\mathbf{L} \in \mathbb{R}^{6n\times 6n}$ maps velocity $\boldsymbol{v}$ to time-derivative of position $\boldsymbol{q}$.

We model $\boldsymbol{f}_{\text{col}}$ using Coulomb's model of friction coupled with complementarity model of contact [7]. Suppose there are m pairs of bodies that are in contact. Consider the i-th such pair; we decompose the contact force acting on this pair along three directions: one normal to the contact plane and the other two mutually orthogonal spanning the plane. Let $\hat{\gamma}_{i,n}$, $\hat{\gamma}_{i,1}$ and $\hat{\gamma}_{i,2}$ be the magnitudes of these components; suppose $\boldsymbol{d}_{i,n}$, $\boldsymbol{d}_{i,1}$ and $\boldsymbol{d}_{i,2} \in \mathbb{R}^{6n}$ represent these directions in our generalized coordinate system. We can assume $\hat{\gamma}_{i,n} \geq 0$ without loss of generality. Then,

$$\boldsymbol{f}_{\text{col}} = \sum_{i=1}^{m} \hat{\gamma}_{i,n}\boldsymbol{d}_{i,n} + \hat{\gamma}_{i,1}\boldsymbol{d}_{i,1} + \hat{\gamma}_{i,2}\boldsymbol{d}_{i,2} = \sum_{i=1}^{m} \mathbf{D}_i\hat{\boldsymbol{\gamma}}_i = \mathbf{D}\hat{\boldsymbol{\gamma}}$$

is the total force due to all the contacts. Here

$$\hat{\boldsymbol{\gamma}} = (\hat{\boldsymbol{\gamma}}_1, \ldots, \hat{\boldsymbol{\gamma}}_m), \quad \hat{\boldsymbol{\gamma}}_i = (\hat{\gamma}_{i,n}, \hat{\gamma}_{i,1}, \hat{\gamma}_{i,2})$$

is the vector of pairwise contact forces, and

$$\mathbf{D} = \begin{bmatrix}\mathbf{D}_1 & \cdots & \mathbf{D}_m\end{bmatrix}, \quad \mathbf{D}_i = \begin{bmatrix}\boldsymbol{d}_{i,n} & \boldsymbol{d}_{i,1} & \boldsymbol{d}_{i,2}\end{bmatrix}$$

is the so called contact transformation matrix.

In the Coulomb model of friction, each contact force lies in a convex cone defined by the coefficient of friction μ_i,

$$\mathcal{C}_i = \left\{\hat{\boldsymbol{\gamma}}_i \in \mathbb{R}^3 : \sqrt{\hat{\gamma}_{i,1}^2 + \hat{\gamma}_{i,2}^2} \leq \mu_i\hat{\gamma}_{i,n}\right\}.$$

The frictional components satisfy a maximum dissipation condition

$$(\hat{\gamma}_{i,1}, \hat{\gamma}_{i,2}) = \text{argmin}_{\hat{\boldsymbol{\gamma}}_i\in\mathcal{C}_i}(\hat{\gamma}_{i,1}\boldsymbol{d}_{i,1} + \hat{\gamma}_{i,2}\boldsymbol{d}_{i,2})^\top\boldsymbol{v}.$$

The complementarity condition implies contact force $\hat{\boldsymbol{\gamma}}_i$ is inactive unless the i-th pair of bodies comes into contact. Let $\phi_i(\boldsymbol{q})$ be the distance between these bodies in configuration $\boldsymbol{q}$. Then, $\hat{\gamma}_{i,n} \geq 0$, $\phi_i(\boldsymbol{q}) \geq 0$ and $\hat{\gamma}_{i,n}\phi_i(\boldsymbol{q}) = 0$. These three conditions are denoted together by

$$0 \leq \hat{\gamma}_{i,n} \perp \phi_i(\boldsymbol{q}) \geq 0.$$

The full model for our system's dynamics is the differential variational inequality (DVI) problem:

$$\mathbf{M}\dot{\boldsymbol{v}} = \boldsymbol{f}_{\text{ext}} + \mathbf{D}\hat{\boldsymbol{\gamma}} \tag{1a}$$
$$0 \leq \hat{\gamma}_{i,n} \perp \phi_i(\boldsymbol{q}) \geq 0 \tag{1b}$$
$$(\hat{\gamma}_{i,1}, \hat{\gamma}_{i,2}) = \text{argmin}_{\hat{\boldsymbol{\gamma}}_i\in\mathcal{C}_i}(\hat{\gamma}_{i,1}\boldsymbol{d}_{i,1} + \hat{\gamma}_{i,2}\boldsymbol{d}_{i,2})^\top\boldsymbol{v} \tag{1c}$$
$$\dot{\boldsymbol{q}} = \mathbf{L}\boldsymbol{v} \tag{1d}$$

B. Discretization and Cone Complementarity

We use a semi-implicit Euler time-stepping scheme to discretize the DVI (1). Given position $\boldsymbol{q}^k$ and velocity $\boldsymbol{v}^k$ at the k-th time-step with step-size h, we obtain $\boldsymbol{q}^{k+1}$ and $\boldsymbol{v}^{k+1}$ by solving a nonlinear complementarity problem (NCP)

$$\boldsymbol{v}^{k+1} = \boldsymbol{v}^k + \mathbf{M}^{-1}(h\boldsymbol{f}_{\text{ext}} + \mathbf{D}\boldsymbol{\gamma}) \tag{2a}$$
$$0 \leq \gamma_{i,n} \perp \phi_i^k/h + \boldsymbol{d}_{i,n}^\top\boldsymbol{v}^{k+1} \geq 0 \tag{2b}$$
$$(\gamma_{i,1}, \gamma_{i,2}) = \text{argmin}_{\boldsymbol{\gamma}_i\in\mathcal{C}_i}(\gamma_{i,1}\boldsymbol{d}_{i,1} + \gamma_{i,2}\boldsymbol{d}_{i,2})^\top\boldsymbol{v}^{k+1} \tag{2c}$$
$$\boldsymbol{q}^{k+1} = \boldsymbol{q}^k + h\mathbf{L}\boldsymbol{v}^{k+1} \tag{2d}$$

In (2a), we use $\boldsymbol{\gamma} = h\hat{\boldsymbol{\gamma}}$ as the impulse vector. The right hand side of (2b) is obtained by discretizing ϕ_i^{k+1} using a forward Euler scheme and dividing it by h for better numerical stability (this avoids small floating point numbers).

In general, NCPs are very difficult to solve numerically. However, relaxing the complementarity condition (2b)

$$0 \leq \gamma_{i,n} \perp \phi_i^k/h + \boldsymbol{d}_{i,n}^\top\boldsymbol{v}^{k+1} - \mu_i\sqrt{(\boldsymbol{d}_{i,1}^\top\boldsymbol{v}^{k+1})^2 + (\boldsymbol{d}_{i,2}^\top\boldsymbol{v}^{k+1})^2} \geq 0$$

leads to a cone complementarity problem (CCP) [7]

$$\mathcal{C} \ni \boldsymbol{\gamma} \perp \boldsymbol{g} = \mathbf{A}\boldsymbol{\gamma} + \boldsymbol{b} \in \mathcal{C}^* \tag{3}$$

where $\mathcal{C} = \mathcal{C}_1 \oplus \cdots \oplus \mathcal{C}_m$, $\mathcal{C}^* = \mathcal{C}_1^* \oplus \cdots \oplus \mathcal{C}_m^*$,

$$\mathbf{A} = [\mathbf{A}_1; \ldots; \mathbf{A}_m] \qquad \boldsymbol{b} = [\boldsymbol{b}_1; \ldots; \boldsymbol{b}_m]$$
$$\mathbf{A}_i = \mathbf{D}_i^\top\mathbf{M}^{-1}\mathbf{D} \qquad \boldsymbol{b}_i = \boldsymbol{\Phi}_i/h + \mathbf{D}_i^\top\hat{\boldsymbol{v}}$$
$$\hat{\boldsymbol{v}} = \boldsymbol{v}^k + h\mathbf{M}^{-1}\boldsymbol{f}_{\text{ext}} \qquad \boldsymbol{\Phi}_i = (\phi_i^k, 0, 0)$$

and $\mathcal{C}_i^* = \{\boldsymbol{g}_i \in \mathbb{R}^3 : \boldsymbol{\gamma}_i^\top\boldsymbol{g}_i \geq 0\}$ is the dual cone of $\mathcal{C}_i$. We denote, with $\boldsymbol{g}_i = \mathbf{A}_i\boldsymbol{\gamma} + \boldsymbol{b}_i$,

$$\boldsymbol{\gamma} \perp \boldsymbol{g} \iff \boldsymbol{\gamma}_i^\top\boldsymbol{g}_i = 0 \text{ for all } i = 1:m.$$

It can be shown that as time-step $h \to 0$, the solution of the CCP approaches that of the NCP.

III. Solution of the Complementarity Problem

In [15], the authors compare performance of several solvers for the CCP (3). They conclude that accelerated projected gradient descent (APGD) and symmetric cone interior point (SCIP) methods are best among the first order and second order solvers, respectively. In this section, we briefly outline these methods.

A. Accelerated Projected Gradient Descent

It was shown in [7] that (3) represents the Karush-Kuhn-Tucker (KKT) optimality conditions for a cone constrained quadratic optimization problem:

$$\text{minimize } f_0(\boldsymbol{\gamma}) = \frac{1}{2}\boldsymbol{\gamma}^\top\mathbf{A}\boldsymbol{\gamma} + \boldsymbol{b}^\top\boldsymbol{\gamma} \text{ subject to } \boldsymbol{\gamma} \in \mathcal{C}. \tag{4}$$

Gradient descent algorithms are perhaps the simplest family of iterative solvers for this convex optimization problem. At each iteration step, one simply moves along the steepest descent direction (opposite to the current gradient).

We implement the scheme proposed by Nesterov [19]. It accelerates the slow convergence of the ordinary gradient descent scheme by utilizing the concept of momentum. Essentially, at each step, the gradient information from previous iterations is used to modify the step direction: starting with $\theta^0 = 1$ and $\boldsymbol{y}^0 = \boldsymbol{\gamma}^0$, we repeat

$$\begin{aligned}\boldsymbol{\gamma}^{k+1} &= \boldsymbol{y}^k - \alpha^k \nabla f_0(\boldsymbol{y}^k)\\ \theta^{k+1} &= \tfrac{\theta^k\sqrt{(\theta^k)^2+4}-(\theta^k)^2}{2}\\ \beta^{k+1} &= \tfrac{\theta^k(1-\theta^k)}{(\theta^k)^2+\theta^{k+1}}\\ \boldsymbol{y}^{k+1} &= \boldsymbol{\gamma}^{k+1} + \beta^{k+1}(\boldsymbol{\gamma}^{k+1} - \boldsymbol{\gamma}^k)\end{aligned}$$

We choose the step-size parameter α^k based on local properties of f_0. Let L be the local Lipschitz constant satisfying

$$f_0(\boldsymbol{y}) \le f_0(\boldsymbol{y}^k) + \nabla f_0(\boldsymbol{y}^k)^\top(\boldsymbol{y} - \boldsymbol{y}^k) + \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{y}^k\|_2^2.$$

Then, a choice of $\alpha^k < 1/L$ ensures that Nesterov's algorithm achieves optimal convergence rate among first order methods. We estimate L at the beginning of each APGD iteration using a standard line search.

For constrained convex optimization problems, we can extend this accelerated gradient descent algorithm. We simply project the iterates $\boldsymbol{\gamma}^k$ onto the feasible set:

$$\boldsymbol{\gamma}^{k+1} = \Pi_{\mathcal{C}}\big(\boldsymbol{y}^k - \alpha^k \nabla f_0(\boldsymbol{y}^k)\big).$$

Here $\Pi_{\mathcal{C}}$ is the orthogonal projection operator onto $\mathcal{C}$. This modified algorithm is the accelerated projected gradient descent (APGD) method.

B. Symmetric Cone Interior Point Method

The symmetric cone interior point (SCIP) method solves the CCP (3) by utilizing the Jordan algebraic structure on $\mathbb{R}^3$ [14]. We define the Jordan product

$$\boldsymbol{x}_i \circ \boldsymbol{y}_i = \tfrac{1}{\sqrt{2}}(\boldsymbol{x}_i^\top \boldsymbol{y}_i, x_{i,n}y_{i,1} + x_{i,1}y_{i,n}, x_{i,n}y_{i,2} + x_{i,2}y_{i,n})$$

for $\boldsymbol{x}_i = (x_{i,n}, x_{i,1}, x_{i,2})$, $\boldsymbol{y}_i = (y_{i,n}, y_{i,1}, y_{i,2}) \in \mathbb{R}^3$. The unit element is $\boldsymbol{e}_i = (\sqrt{2}, 0, 0)$. We define the symmetric cone

$$\mathcal{K}_i = \{\boldsymbol{x}_i \circ \boldsymbol{x}_i : \boldsymbol{x}_i \in \mathbb{R}^3\} = \{\boldsymbol{x}_i : x_{i,n} \ge (x_{i,1}^2 + x_{i,2}^2)^{1/2}\}.$$

Let $\mathcal{K} = \mathcal{K}_1 \oplus \cdots \oplus \mathcal{K}_m$. Define the maps

$$\mathbf{T}_x = \operatorname{diag}(\ldots, \mu_i, 1, 1, \ldots), \mathbf{T}_y = \operatorname{diag}(\ldots, 1, \mu_i, \mu_i, \ldots).$$

Denote $\bar{\mathbf{A}} = \mathbf{T}_y\mathbf{A}\mathbf{T}_x^{-1}$ and $\bar{\boldsymbol{b}} = \mathbf{T}_y\boldsymbol{b}$. Then the CCP (3) is equivalent to

$$\mathcal{K} \ni \boldsymbol{x} \perp \boldsymbol{y} = \bar{\mathbf{A}}\boldsymbol{x} + \bar{\boldsymbol{b}} \in \mathcal{K} \text{ with } \boldsymbol{x} = \mathbf{T}_x\boldsymbol{\gamma}, \boldsymbol{y} = \mathbf{T}_y\boldsymbol{g}.$$

The corresponding optimization problem is given by

$$\text{minimize } \boldsymbol{x}^\top\boldsymbol{y} \text{ subject to } \boldsymbol{x}, \boldsymbol{y} \in \mathcal{K}.$$

As per [14], we define the barrier function for the double cone $\mathcal{K} \cup (-\mathcal{K})$, which gives rise to the potential function

$$\begin{aligned}f(\boldsymbol{x}, \boldsymbol{y}) &= \rho \log(\boldsymbol{x}^\top\boldsymbol{y}) + f_{\text{cen}}(\boldsymbol{x}, \boldsymbol{y}),\\ f_{\text{cen}}(\boldsymbol{x}, \boldsymbol{y}) &= 2m \log \frac{\boldsymbol{x}^\top\boldsymbol{y}/m}{2\prod_{i=1}^m [\det(\boldsymbol{x}_i)\det(\boldsymbol{y}_i)]^{1/2m}}.\end{aligned}$$

Here $\det(\boldsymbol{x}_i) = \frac{1}{2}(x_{i,n}^2 - x_{i,1}^2 - x_{i,2}^2)$ and $\rho > 0$ is the barrier parameter. The logarithmic barrier f_{cen} penalizes values close to the boundary.

We construct a sequence $(\boldsymbol{x}^k, \boldsymbol{y}^k)$ that approach the optimal value strictly from the interior point of the feasible set. We enforce $f_{\text{cen}}(\boldsymbol{x}^k, \boldsymbol{y}^k) = 0$; then the cost function decreases as rapidly as the sequence approaches the boundary. These points lie on the central path, where $\boldsymbol{h}(\boldsymbol{x}^k, \boldsymbol{y}^k) = \boldsymbol{x}^k \circ \boldsymbol{y}^k - \alpha\boldsymbol{e} = 0$ for some $\alpha > 0$ (the Jordan product and the unit element is extended per-contact). We apply Newton step to the function $\boldsymbol{h}$ with decreasing α; the search direction is obtained by solving

$$\begin{bmatrix}\nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x}^k, \boldsymbol{y}^k) & \nabla_{\boldsymbol{y}}\boldsymbol{h}(\boldsymbol{x}^k, \boldsymbol{y}^k)\\ \bar{\mathbf{A}} & -\mathbf{I}\end{bmatrix}\begin{bmatrix}\Delta\boldsymbol{x}\\ \Delta\boldsymbol{y}\end{bmatrix} = \begin{bmatrix}\alpha\boldsymbol{e} - \boldsymbol{x}^k \circ \boldsymbol{y}^k\\ \mathbf{0}\end{bmatrix}.$$

To start the iteration, we require a strictly feasible pair $(\boldsymbol{x}^0, \boldsymbol{y}^0)$. A well-known procedure to achieve this involves adding one artificial variable, augmenting the $3m$-variable complementarity problem to a $3m+1$ variable problem with a straight-forward solution [14].

As the iterates approach the boundary of the feasible set, the linear system becomes increasingly ill-conditioned. This issue can be resolved by applying Nesterov-Todd scaling [20], which rescales the space in which the symmetric cone lies. This leads to a linear system of the form

$$[\bar{\mathbf{A}} + \mathbf{P}(\boldsymbol{w})]\Delta\boldsymbol{x} = \boldsymbol{r}$$

where $\mathbf{P}(\boldsymbol{w})$ is 3×3 block-diagonal and $\boldsymbol{r} \in \mathbb{R}^{3m}$ is a vector obtained from reducing the Newton system using Schur's complement [14]. The i-th block of $\mathbf{P}$ is given by

$$\mathbf{P}_i(\boldsymbol{w}_i) = \boldsymbol{w}_i\boldsymbol{w}_i^\top - \det(\boldsymbol{w}_i)\mathbf{J}$$

with $\mathbf{J} = \operatorname{diag}(1, -1, -1)$ and

$$\boldsymbol{w}_i = \frac{\boldsymbol{y}_i^k + \lambda_i\mathbf{J}\boldsymbol{x}_i^k}{\sqrt{(\boldsymbol{x}_i^k)^\top\boldsymbol{y}_i^k + 2\sqrt{\det(\boldsymbol{x}_i^k)\det(\boldsymbol{y}_i^k)}}}, \lambda_i = \sqrt{\frac{\det(\boldsymbol{y}_i^k)}{\det(\boldsymbol{x}_i^k)}}.$$

Once we solve for the search direction $\Delta\boldsymbol{x}$ and $\Delta\boldsymbol{y}$, we pick a step-size $\theta \in (0, 1]$ such that $\boldsymbol{x}^k + \theta\Delta\boldsymbol{x},\ \boldsymbol{y}^k + \theta\Delta\boldsymbol{y} \in \operatorname{int}(\mathcal{K})$; we use a standard backtracking line search. Finally, we update the iterates:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \theta\Delta\boldsymbol{x}, \quad \boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \theta\Delta\boldsymbol{y}.$$

C. Convergence Criteria

We choose our convergence criteria based on the original CCP (3) following [14], [15]. Given a primal-dual pair $(\boldsymbol{\gamma}, \boldsymbol{g})$ with $\boldsymbol{g} = \mathbf{A}\boldsymbol{\gamma} + \boldsymbol{b}$, we compute three residuals

- $\boldsymbol{r}_p \in \mathbb{R}^m$ measures violation of the primal constraint:
$$r_{p,i} = \max\{0, (\gamma_{i,1}^2 + \gamma_{i,2}^2)^{1/2} - \mu_i\gamma_{i,n}\}$$
- $\boldsymbol{r}_d \in \mathbb{R}^m$ measures violation of the dual constraint:
$$r_{d,i} = \max\{0, (g_{i,1}^2 + g_{i,2}^2)^{1/2} - g_{i,n}/\mu_i\}$$
- $r_c = |\boldsymbol{\gamma}^\top\boldsymbol{g}|/m$ measures violation of the complementarity condition.

We stop the iterative solvers when the residual reaches some prescribed tolerance: $\max\left\{\|\boldsymbol{r}_p\|_\infty, \|\boldsymbol{r}_d\|_\infty, r_c\right\} < \tau_{\text{res}}$.

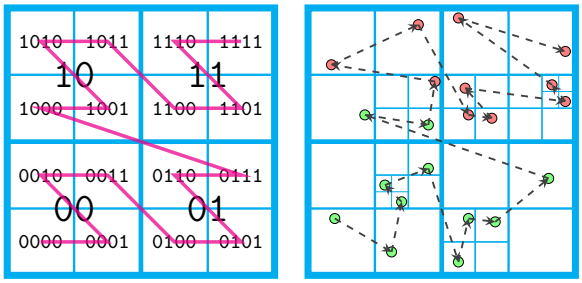


Fig. 1. Example of Morton ordering in two dimensions. (Left) Construction of Morton IDs by interlacing binary expansion of coordinates, and the Z-order on the second level discretization. (Right) A point cloud with octree bounding boxes and imposed Morton ordering on the points. We show 'equi'-partition of the body list into two parts using different colors.

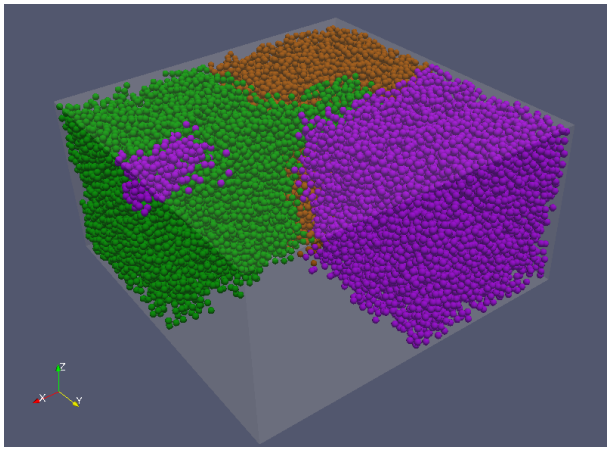


Fig. 2. Distribution of 40,000 spheres with unit radius in a $100 \times 100 \times 50$ box (volume fraction $\simeq 33.5\%$), among 4 MPI processes, using Morton coding. The spheres belonging to the last rank are suppressed in the figure to emphasize interface structure.

IV. Parallel Implementation

A. Spatial Partitioning

Simulation of large-scale particulate systems is ultimately limited by memory, as storing all bodies in the same processor becomes intractable. The message passing interface (MPI) library provides a well-known framework to overcome this; data associated to the set of bodies must be partitioned among the available MPI processes in a way that minimizes communication between them, e.g. when determining particle pairs likely to collide in the next time-step. Minimizing this communication overhead is crucial in designing fast rigid body simulations.

One way to ensure minimal communication is by ensuring that spatially close bodies end up on the same MPI process. One can achieve this very naturally by using tree based, hierarchical spatial partitioning schemes, e.g. octree, k-D tree [21]. In our implementation, we use a Morton-code based octree partitioning scheme [22]; it achieves good spatial locality with very low computational cost.

Morton ranking induces linear order on a multi-dimensional particle cloud. As an example, let us consider a point $(x_1, x_2, x_3) \in [0,1)^3$ in three dimensions. Given the k-length binary expansions of the coordinates $x_i = 0.b_{i1} \dots b_{ik}$, we compute the $3k$-bit Morton index by interleaving the bits: $I = b_{31}b_{21}b_{11} \dots b_{3k}b_{2k}b_{1k}$. Sorting the particle cloud according to these indices creates a zigzag ordering of the points (see Figure 1) and ensures that points that are close in Morton order are also spatially nearby.

Once the list of bodies is sorted according to their Morton ranks, we partition this list equally, and assign each part to one MPI rank. Figure 2 illustrates this partitioning for 40,000 unit spheres, contained in a $100 \times 100 \times 50$ box, among 4 MPI processes.

B. Collision Detection

Once bodies are assigned to MPI processes, we construct the collision pairs. This is done in two phases. In the broad phase, the list of potential contacts within each MPI rank is pruned by re-using the octree structure from the partitioning phase, and eliminating pairs of bodies that are far away. In the narrow phase, we test for contact between the true rigid bodies. This brings down the $\mathcal{O}(n^2)$ complexity of the naive algorithm to $\mathcal{O}(n \log n)$, where n is the number of bodies in a MPI process. We further accelerate this by using OpenMP task-based parallelism. For inter-process collisions, we assign the collision pair $i = (i_1, i_2)$, $i_1 < i_2$ to the MPI process containing the i_1-th body.

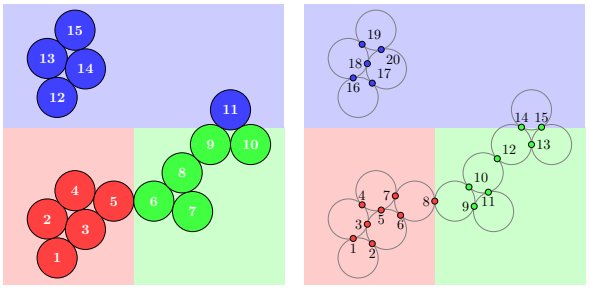
C. Collision Resolution

For each collision pair i, we store distance ϕ_i^k, unknown impulse $\boldsymbol{\gamma}_i$ and contact transformation matrix $\mathbf{D}_i$ on the same MPI rank that contains the collision pair. We use a distributed memory linear algebra library to manage the different parts of the distributed vectors $\boldsymbol{\Phi}$ and $\boldsymbol{\gamma}$, and the distributed matrix $\mathbf{D}$.

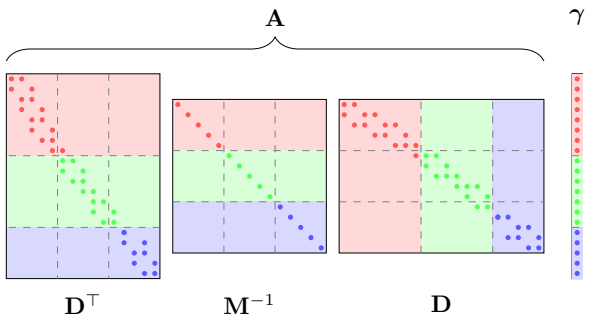
In APGD, the main bottleneck is matrix-vector multiplications (mat-vec) with the collision matrix $\mathbf{A} = \mathbf{D}^\top \mathbf{M}^{-1} \mathbf{D}$. We need one mat-vec per iteration to compute the gradient $\boldsymbol{g} = \mathbf{A}\boldsymbol{\gamma} + \boldsymbol{b}$. Additionally, in every iteration, we use a backtracking search to estimate the Lipschitz constant; this requires one mat-vec per backtracking step.

We exploit the sparsity structure of the three factor matrices to create an efficient mat-vec implementation. For granular media simulations, the mass matrix $\mathbf{M}$ is 6×6 block diagonal, and the columns of $\mathbf{D}$ contain at most 12 non-zero entries (6 per body in the corresponding collision pair). This allows us to store $\mathbf{M}^{-1}$ and $\mathbf{D}^\top$ in compressed row storage (CRS) format. The partitioning of the bodies and collision pairs among the MPI ranks also induce a natural partitioning of the rows of these matrices among the MPI processes. As long as $\boldsymbol{\gamma}$ is partitioned using the same schemes, matrix vector multiplication will be fast (see Figure 3).

In SCIP, the bottleneck is solving a linear system at each iteration. We currently use a direct sparse LU factorization of matrix $\mathbf{A}$; for this purpose, we must build it explicitly.



(a) Distribution of 15 bodies (left) with 20 collision pairs (right) among 3 MPI ranks. The colors indicate which rank owns the bodies/collision pairs.



(b) Sparsity structure of the corresponding collision matrix (solid dots indicate non-zero blocks). The inverse mass matrix $\mathbf{M}^{-1}$ is 6×6 block diagonal. The columns of the contact transformation matrix $\mathbf{D}$ encode information about collision pairs. E.g. the 8^{th} collision occurs between bodies 5 and 6; this corresponds to non-zero 6×3 blocks in the 8^{th} 'column'.

Fig. 3. Example of distributed construction of the collision matrix

The sparsity structure of $\mathbf{A}$ is dependent on contact structure; the 3×3 non-zero blocks correspond to contacts that share a body. In [16], a tensor train preconditioner exploiting this structure was proposed as an acceleration for IP methods. We aim to implement this in our hybrid MPI/OpenMP framework in future work.

V. Numerical Experiments

In this section, we describe the results from the numerical experiments we conducted to investigate the performance and scalability characteristics of our implementation of cone-complementarity collision solver.

A. Architecture and Implementation

We ran our simulations on the Flux and Great Lakes clusters at University of Michigan. Each compute node in Flux is equipped with two 12-core 2.5 GHz Intel Xeon E5-2680 v3 processors and 128 GB RAM. Compute nodes in Great Lakes are equipped with two 18-core 3.0 GHz Intel Xeon Gold 6154 processors and 192 GB RAM.

The code is written in C++ and is built on top of Trilinos [23], Msgpack [24], Eigen [25] and TRNG [26] libraries. The Trilinos library provides a large number of data structures to manage distributed vectors and sparse matrices and implements efficient sparse mat-vecs. It also provides an interface to the SuperLU direct solver package [27]. Msgpack is a binary serialization library; we use it to facilitate interchange of rigid bodies between MPI processes. We use the three dimensional vectors and quaternion classes from the Eigen library to capture the motion of the rigid bodies. Finally, we use TRNG to set up the random initial configurations (e.g. radius and location of the spheres).

TABLE I
Number of iterations required to reach prescribed residual tolerance for APGD and SCIP solvers

	Number of Spheres			
	25,000		100,000	
Residual	APGD	SCIP	APGD	SCIP
$\tau_{\text{res}} = 10^{-1}$	409	45	435	47
$\tau_{\text{res}} = 10^{-2}$	433	50	451	52
$\tau_{\text{res}} = 10^{-3}$	470	54	506	56
$\tau_{\text{res}} = 10^{-4}$	509	57	521	58

B. Simulation Setup

In our experiments, we simulate sedimentation of rigid bodies under gravity. Our setup is very simple: we place a large number of spheres (radius = 0.01 m) inside a 3D rectangular box, and release them from rest. The spheres experience constant acceleration due to gravity ($g = 9.81$ m/s^2). In the course of the simulation, the spheres collide with each other and with the interior walls of the box.

C. Comparison of APGD and SCIP Solvers

We compared our APGD and SCIP implementations on simulations with 25,000 and 100,000 rigid bodies. We keep the number of collisions proportional to the number of bodies; we fix the box height at 0.5 m and assign 0.04 m^2 base area per thousand spheres. The spheres occupy approximately 20.94% of the box volume. We allow a maximum of 10,000 solver iterations per timestep, and choose a timestep of $h = 10^{-3}$ s. We run these simulations on a single node of the Flux cluster using 16 cores.

Table I records the number of iterations required to reach a prescribed tolerance in one timestep. As we can see, SCIP converges using fewer iterations compared to APGD, which is consistent with the findings in [15]. However, since each SCIP iteration requires a linear solve, APGD converges faster w.r.t. runtime.

D. Scalability of APGD Solver

We tested the scalability of the collision detection and resolution phases of our algorithm, using the APGD iterative solver. We used a timestep size $h = 2.5 \times 10^{-3}$ s, residual tolerance $\tau_{\text{res}} = 10^{-2}$ and allowed a maximum 100,000 solver iterations. In these simulations, the box height was 0.6 m and 0.036 m^2 base area was assigned per thousand spheres (approximately 19.39% volume fraction). Since our primary goal is to test performance of the distributed memory aspect, we only use 4 cores per node in these simulations. We ran these on the Great Lakes cluster.

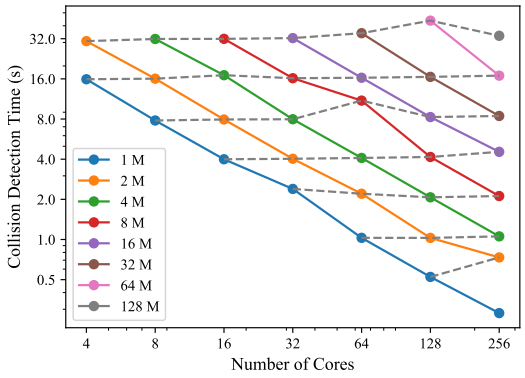


Fig. 4. Scalability of collision detection algorithm. The solid lines connect simulations with same problem size (strong scaling), and the dashed lines connect simulations with same problem size per core (weak scaling).

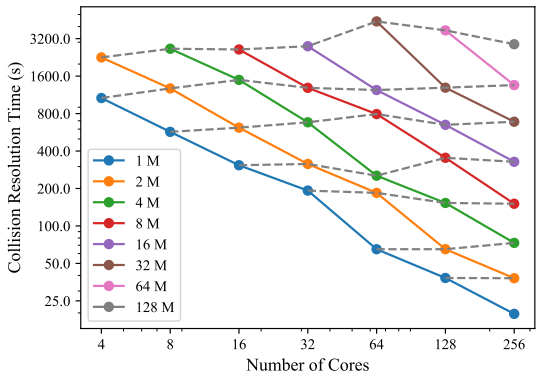


Fig. 5. Scalability of collision resolution algorithm with APGD optimization solver. The solid lines connect simulations with same problem size (strong scaling), and the dashed lines connect simulations with same problem size per core (weak scaling).

We ran an array of simulations using 1 to 128 million particles and 4 to 256 cores. The scaling results for the collision detection and collision resolution phases are shown in Figure 4 and Figure 5, respectively. Solid lines in the figures connect simulations with the same problem size (strong scaling). We observe that time required to solve a fixed size problem drops as fast as the number of processors is increased.

The dashed lines in these two figures connect simulations with the same number of bodies per core (weak scaling). We see that the collision detection time remains almost constant as we increase the number of cores (and the problem size), demonstrating near-perfect scalability. On the other hand, collision resolution times increase slowly with the number of cores. The main reason for this is: as we increase the problem size, the number of APGD iterations required for convergence also increases (see Table II). This results in longer collision resolution time. Nonetheless, we note that the rate of increase in collision resolution time is considerably slower than the increase is problem size (less than 2-fold increase in runtime compared to 64-fold increase in problem size).

Our largest simulation was a sedimentation test with 256 million bodies. We ran it on the Flux cluster with 64 nodes, each node using 8 cores (a total of 512 cores), with timestep $h = 2.5 \times 10^{-3}$ s and tolerance $\tau_{\text{res}} = 5 \times 10^{-2}$. On average, 324 million contacts were detected in 2 minutes, and collisions were resolved in 24 minutes per time step.

TABLE II
AVERAGE ITERATION COUNT OF APGD IN WEAK SCALABILITY TESTS

# Cores	4	8	16	32	64	128	256
# Spheres ($\times 10^6$)	2	4	8	16	32	64	128
# Collisions ($\times 10^6$)	5	10	19	39	77	155	309
# Iterations	713	763	748	788	836	803	832

VI. CONCLUSIONS

Simulating the dynamics of a system of rigid bodies with the CCP formulation of frictional contact involves efficient collision detection and the solution of a second order cone-constrained quadratic optimization problem. In this article, we proposed a framework for this problem featuring a hybrid distributed/shared memory computing model in both stages.

We used the Morton ordering to partition the rigid bodies among MPI ranks. This imposes data locality, that is, bodies that are located close to each other end up on the same rank. In turn, this limits the communication overhead in the collision detection phase significantly. Our experiments show that this phase scales almost perfectly with the number of processors.

Using a very simple strategy to divide the collision pairs among MPI ranks, we ensure proper load balancing. We implemented distributed memory versions of the accelerated projected gradient descent (APGD) and symmetric-cone-interior point (SCIP) solvers for the optimization problems exploiting the sparsity structure of the collision matrix. Additionally, we built in shared memory parallelism within each MPI rank using OpenMP, providing additional acceleration for these solvers.

The first order APGD solver relies on matrix-vector products for the collision matrix, with performance reliant on the efficiency of Lipschitz constant estimation and growth of iteration counts with problem size. Overall, this solver shows extremely good scaling and performance.

The second order SCIP solver relies on the solution of a sparse linear system per iteration, resulting in costly solution for large numbers of collisions. Consequently, our implementation, while being more robust compared to the first order APGD solver, cannot compete in terms of performance. In future work, we plan to incorporate a tensor-train preconditioned iterative linear solve in our hybrid computing framework.

ACKNOWLEDGEMENTS

This work was supported by the Automotive Research Center (ARC) in accordance with Cooperative Agreement W56HZV-19-2-0001 with U.S. Army CCDC Ground Vehicle Systems Center and the National Science Foundation under grant DMS-1454010. This research was also supported in part through computational resources and services provided by the Advanced Research Computing Center at the University of

Michigan. We thank Dhairya Malhotra and Wen Yan for many helpful discussions.

REFERENCES

[1] E. Coumans, "Bullet physics library," https://bulletphysics.org.

[2] R. Berger, C. Kloss, A. Kohlmeyer, and S. Pirker, "Hybrid parallelization of the LIGGGHTS open-source DEM code," *Powder technology*, vol. 278, pp. 234–247, 2015.

[3] H. Mazhar, T. Heyn, A. Pazouki, D. Melanz, A. Seidl, A. Bartholomew, A. Tasora, and D. Negrut, "Chrono: a parallel multi-physics library for rigid-body, flexible-body, and fluid dynamics," *Mechanical Sciences*, vol. 4, no. 1, pp. 49–64, 2013.

[4] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[5] M. Anitescu and F. A. Potra, "Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems," *Nonlinear Dynamics*, vol. 14, no. 3, pp. 231–247, 1997.

[6] M. Anitescu, "Optimization-based simulation of nonsmooth rigid multibody dynamics," *Mathematical Programming*, vol. 105, no. 1, pp. 113–143, 2006.

[7] M. Anitescu and A. Tasora, "An iterative approach for cone complementarity problems for nonsmooth dynamics," *Computational Optimization and Applications*, vol. 47, no. 2, 2010.

[8] A. Tasora and M. Anitescu, "A convex complementarity approach for simulating large granular flows," *Journal of Computational and Nonlinear Dynamics*, vol. 5, no. 3, p. 031004, 2010.

[9] A. Pazouki, M. Kwarta, K. Williams, W. Likos, R. Serban, P. Jayakumar, and D. Negrut, "Compliant contact versus rigid contact: A comparison in the context of granular dynamics," *Physical Review E*, vol. 96, no. 4, p. 042905, 2017.

[10] H. Mazhar, T. Heyn, D. Negrut, and A. Tasora, "Using Nesterov's method to accelerate multibody dynamics with friction and contact," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 3, p. 32, 2015.

[11] T. Heyn, M. Anitescu, A. Tasora, and D. Negrut, "Using Krylov subspace and spectral methods for solving complementarity problems in many-body contact dynamics simulation," *International Journal for Numerical Methods in Engineering*, vol. 95, no. 7, pp. 541–561, 2013.

[12] C. Petra, B. Gavrea, M. Anitescu, and F. Potra, "A computational study of the use of an optimization-based method for simulating large multibody systems," *Optimization Methods & Software*, vol. 24, no. 6, pp. 871–894, 2009.

[13] L. Fang, "A primal-dual interior point method for solving multibody dynamics problems with frictional contact," Ph.D. dissertation, University of Wisconsin–Madison, 2015.

[14] J. Kleinert, *Simulating granular material using nonsmooth time-stepping and a matrix-free interior point method*. Fraunhofer Verlag, 2015.

[15] D. Melanz, L. Fang, P. Jayakumar, and D. Negrut, "A comparison of numerical methods for solving multibody dynamics problems with frictional contact modeled via differential variational inequalities," *Computer Methods in Applied Mechanics and Engineering*, 2017.

[16] E. Corona, D. Gorsich, P. Jayakumar, and S. Veerapaneni, "Tensor train accelerated solvers for nonsmooth rigid body dynamics," *Applied Mechanics Reviews*, 2019.

[17] D. Negrut, R. Serban, H. Mazhar, and T. Heyn, "Parallel computing in multibody system dynamics: why, when, and how," *Journal of Computational and Nonlinear Dynamics*, vol. 9, no. 4, p. 041007, 2014.

[18] D. Negrut, A. Tasora, M. Anitescu, H. Mazhar, T. Heyn, and A. Pazouki, "Solving large multibody dynamics problems on the gpu," in *GPU Computing Gems Jade Edition*. Elsevier, 2012, pp. 269–280.

[19] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate o(1/kˆ2)," *Sov. Math. Dokl.*, vol. 27, no. 2, pp. 372–376, 1983.

[20] Y. E. Nesterov and M. J. Todd, "Self-scaled barriers and interior-point methods for convex programming," *Mathematics of Operations Research*, vol. 22, no. 1, 1997.

[21] J. Schauer and A. Nüchter, "Collision detection between point clouds using an efficient kd tree implementation," *Advanced Engineering Informatics*, vol. 29, no. 3, pp. 440–458, 2015.

[22] T. M. Chan, "Closest-point problems simplified on the RAM," in *13th ACM-SIAM Symposium on Discrete Algorithms*, 2002.

[23] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, "An overview of the trilinos project," *ACM Transactions on Mathematical Software*, vol. 31, no. 3, 2005.

[24] S. Furuhashi *et al.*, "Msgpack," https://msgpack.org.

[25] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org.

[26] H. Bauke and S. Mertens, "Random numbers for large scale distributed monte carlo simulations," *Physical Review E*, vol. 75, no. 6, 2007.

[27] X. S. Li, "An overview of SuperLU: Algorithms, implementation, and user interface," *ACM Transactions on Mathematical Software*, vol. 31, no. 3, 2005.