TENSOR TRAIN ACCELERATED SOLVERS FOR NONSMOOTH RIGID BODY DYNAMICS

EDUARDO CORONA *, DAVID GORSICH †, PARAMSOTHY JAYAKUMAR † AND SHRAVAN VEERAPANENI *

Abstract. In the last two decades, increased need for high-fidelity simulations of the time evolution and propagation of forces in granular media has spurred a renewed interest in the discrete element method (DEM) modeling of frictional contact. Force penalty methods, while economic and widely accessible, introduce artificial stiffness, requiring small time steps to retain numerical stability. Optimization-based methods, which enforce contacts geometrically through complementarity constraints leading to a differential variational inequality problem (DVI), allow for the use of larger time steps at the expense of solving a nonlinear complementarity problem (NCP) each time step. We review the latest efforts to produce solvers for this NCP, focusing on its relaxation to a cone complementarity problem (CCP) and solution via an equivalent quadratic optimization problem with conic constraints. We distinguish between *first order* methods, which use only gradient information and are thus linearly convergent and *second order* methods, which rely on a Newton type step to gain quadratic convergence and are typically more robust and problem-independent. However, they require the approximate solution of large sparse linear systems, thus losing their competitive advantages in large scale problems due to computational cost.

In this work, we propose a novel acceleration for the solution of Newton step linear systems in second order methods using low-rank compression based fast direct solvers, leveraging on recent direct solver techniques for structured linear systems arising from differential and integral equations. We employ the Quantized Tensor Train (QTT) decomposition to produce efficient approximate representations of the system matrix and its inverse. This provides a versatile and robust framework to accelerate its solution using this inverse in a direct or a preconditioned iterative method. We demonstrate compressibility of the Newton step matrices in Primal Dual Interior Point (PDIP) methods as applied to the multibody dynamics problem. Using a number of numerical tests, we demonstrate that this approach displays *sublinear* scaling of precomputation costs, may be efficiently updated across Newton iterations as well as across simulation time steps, and leads to a fast, optimal complexity solution of the Newton step. This allows our method to gain an order of magnitude speedups over state-of-the-art preconditioning techniques for moderate to large-scale systems, hence mitigating the computational bottleneck of second order methods.

1. Introduction.

The discrete element method (DEM) [CS79] is one of the most widely used approaches to simulate the multi-body dynamics such as in granular materials. This method considers the granular medium as a collection of discrete particles; each responding to body forces such as gravity, inertia or drag, as well as repulsive or dissipative forces caused by contact. Materials in granular form are omnipresent in industry and understanding their dynamics is crucial for a broad range of application fields, including terramechanics, active media, additive manufacturing, nanoparticle self-assembly, avalanche dynamics, composite materials, pyroclastic flows, etc.

Distinct instances of the DEM are defined essentially by their modeling of contact forces, and thus in how collisions are handled. We distinguish two main types in the literature: force penalty methods (DEM-P) and complementarity formulations (DEM-C). Penalty methods introduce one or multiple layers of spring-like forces between objects in contact, and may introduce additional fields to represent friction. These methods are widely used due to being computationally inexpensive and easy to implement. However, in many cases they introduce high stiffness, requiring extremely small time steps to retain stability during collisions. An in-depth and informative comparison between the two can be found in [PKW+17].

DEM-C methods such as in [TA10, Ani06] enforce contacts using complementarity constraints, leading to a differential variational inequality (DVI) problem upon discretization. This allows for the use of larger time steps in its integration, as contacts are enforced geometrically. Given a time-stepping scheme for this problem, a nonlinear complementarity problem (NCP) must be solved to compute the corresponding contact forces at each time step. One common way to solve this optimization problem is via a linearization of the constraints, producing a linear complementarity problem (LCP) [ST96, AP97]. An alternative relaxation method produces an equivalent, convex cone complementarity problem (CCP). Efficient quadratic cone programming techniques such as those in [TA10, MHNT15, HATN13, PGAP09, Fan15, Kle15] have been proposed to solve the CCP.

In [MFJN17], the authors performed a comparison of these quadratic programming techniques focused on determining which class of methods performed best: second order optimization methods, i.e. those that use Hessian information (Interior Point), or first order methods, i.e. methods using only gradient information (Jacobi, Gauss-Seidel, Projected Gradient Descent). Second order methods display quadratic convergence in a neighborhood of the solution, and thus their convergence is much faster and more problem-

---
*Department of Mathematics, University of Michigan
†U.S. Army TARDEC

independent than that of linearly convergent first order methods, requiring at least 1 to 2 orders of magnitude less iterations to reach a desired accuracy across experiments in this work. However, each iteration in second order metods requires the solution of a (generally sparse) linear system, which can become costly as the dimensions of the many-body problem increase. For large problems, the added computational effort ultimately eclipses the gains obtained from the reduction in iteration counts.

The preferred solution method for large sparse systems is often a Krylov subspace iterative method. The iteration counts, and thus the performance of these methods are known to be directly affected by the eigenspectrum of the associated matrices. Preconditioning techniques can be used to cluster the eigenvalues away from zero and drastically reduce iteration counts; we refer the reader to [Ben02] for a general review. Generic sparse preconditioners are most typically based on incomplete or sparsified factorizations, such as the well-known Incomplete LU and Cholesky methods [Saa03]. In [MFJN17], a fast, parallel SaP (split and paralellize) [LSN17] preconditioner was used to accelerate the PDIP method, garnering reductions in iteration counts and execution times.

Most general-purpose preconditioners suffer a trade-off between precomputation costs and the resulting reduction in iteration counts, and their performance is often problem-dependent. Moreover, in the context of optimization problems such as the ones we are interested in, system matrices change every iteration, and it is often impossible or expensive to update the associated preconditioners. Finally, improving their scaling with the number of degrees of freedom and the level of sparsity is extremely important, as it is central to remaining competitive in large-scale problems.

In the last decades, a continued effort has been made to produce *direct* solvers for structured linear systems arising from differential and integral equations. These solvers entirely side-step the challenges related to convergence speed of iterative solvers. They can also lead to dramatic improvements in speed, in particular in situations where a large number of linear systems with coefficient matrices that stay fixed or can be updated via low rank modifications. Additionally, they also provide a methodology to produce robust preconditioners: low accuracy direct solvers may be used in conjunction with iterative refinement or the Krylov subspace method of choice; the use of an approximate low accuracy inverse requires less memory and precomputation time than a direct solver, at the expense of a slight increase in the number of iterations. These features have led to the adoption of these solvers in other areas of scientific computing and statistics.

In [CRZ15], an effective and memory-efficient solver based on the quantized tensor train decomposition (QTT) was presented. By recasting system matrices as tensors, the tensor train TT compression and inversion routines were used to produce direct solvers and robust preconditioners for integral equations in complex geometries in three dimensions. Key properties of this solver that differentiate it from other hierarchical matrix approaches feature sublinear computational costs and memory requirements with problem size N, as well as techniques to produce economic updates for a matrix and its inverse across time-steps, *even when matrix size changes.*

The main goal of this work is to employ a QTT-based approach to provide a radical speed up to second order optimization methods. We demonstrate its application to interior point methods such as the PDIP in the context of many-body dynamics problems; however, we expect our discussion to apply with little to no modification to a general class of Newton and Quasi-Newton type methods. We first study compressibilty of the system matrices and their inverses in TT format for a range of target accuracies. We then demonstrate how factorization re-use can provide significant speed-ups to precomputation costs, reducing costs by orders of magnitude. For three validation tests of common soil mechanics phenomena—sedimentation, blade drafting and direct shear experiments—we show that a TT-based preconditioner displays efficient and robust performance for problems with $> 10^4$ bodies, garnering up to an order of magnitude speed-up and greatly improved iteration counts when compared with state-of-the-art ILU-preconditioned methods. We also confirm an extremely significant gain in scaling of precomputation costs: precomputation for the TT preconditioner is *sublinear* (for all practical purposes, constant) as the number of collisions and matrix size increase.

The QTT decomposition is one of several approaches for approximate solution of linear systems based in hierarchical compression. In the context of sparse structured systems and related factorizations, work has been done for a number of hierarchical matrix formats: *Hierarchically Semi Separable (HSS)* [Gil11, AD13, XCGL09, CDG+06, CGP06, XCGL10, HY15], $\mathcal{H}$ *matrices:* [BGH03, Beb08, Bör10], and FMM [CPD17, PCD17]. Producing efficient, global factorization updates and dealing with high storage costs is an ongoing challenge in these alternate formats.

2. Mathematical Preliminaries: dynamics of rigid bodies in the presence of friction.

2.1. Problem formulation. We consider a granular material comprised of M rigid particles B_i in $\mathbb{R}^3$; the position of each body is uniquely described by the coordinates $\boldsymbol{x}_i$ for its center of mass and the rotation $\boldsymbol{Q}_i$ of a reference frame fixed to the body, represented by a unimodular quaternion requiring three extra parameters. Let $\boldsymbol{q}_i = (\boldsymbol{x}_i, \boldsymbol{Q}_i)$ then be the 6 generalized coordinates for the i-th body, and $\boldsymbol{q} = (\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_M) \in \mathbb{R}^{6M}$.

Each particle's motion can thus be understood as a translation of its center, with velocity $\boldsymbol{u}_i$ and a rotation of its frame, with angular velocity $\boldsymbol{\omega}_i$. Applying Newton's second law, we relate the corresponding accelerations to the total force $\boldsymbol{F}_i$ and torque $\boldsymbol{T}_i$ applied to it. The general equations of motion are then given by

$$\dot{\boldsymbol{q}} = \mathcal{L}(\boldsymbol{q})\boldsymbol{v} \tag{1}$$

$$M(\boldsymbol{q})\dot{\boldsymbol{v}} = \boldsymbol{f}_B(\boldsymbol{q}, \boldsymbol{v}) + \boldsymbol{f}_C. \tag{2}$$

where $\mathcal{L}(\boldsymbol{q})$ is a linear operator that relates velocities to the rate of change in generalized coordinates, M is the mass matrix and $\boldsymbol{v}, \boldsymbol{f}_B, \boldsymbol{f}_C \in \mathbb{R}^{6M}$ contain each body's translational and rotational velocities, the total body forces and torques applied to them and the reaction forces and torques due to contact dynamics, respectively.

2.2. Complementarity contact model. We impose the following contact constraints: no two bodies should penetrate, and if there is contact, a normal force and a tangential frictional force act at the interface. Consider two bodies B_{i_1} and B_{i_2}, and let $\Phi_i(\boldsymbol{q})$ be an unsigned distance function (also known as a gap function) for the pair $i = (i_1, i_2)$ satisfying $\Phi_i(\boldsymbol{q}) > 0$ if the two bodies are separated, $\Phi_i(\boldsymbol{q}) = 0$ if they are touching, $\Phi_i(\boldsymbol{q}) < 0$ otherwise.

If the pair of bodies touch ($\Phi_i(\boldsymbol{q}) = 0$), let $\boldsymbol{n}, \boldsymbol{t}_1, \boldsymbol{t}_2$ be unit normal and tangential vectors at the point of contact. Contact forces $f_N = \gamma_{i,n}\boldsymbol{n}$ and $f_T = \gamma_{i,1}\boldsymbol{t}_1 + \gamma_{i,2}\boldsymbol{t}_2$ are then applied to each body in opposite directions. The complementarity constraint for the normal force (3) prevents penetration, enforcing that bodies move away from each other at contact. The Coulomb friction model ties the magnitudes of the normal and tangential forces. Using a maximum dissipation principle, the friction force is posed as the solution to an optimization problem in (4):

$$\gamma_{i,n} \geq 0 \quad \Phi_i(\boldsymbol{q}) \geq 0 \quad \Phi_i(\boldsymbol{q})\gamma_{i,n} = 0, \tag{3}$$

$$(\gamma_{i,1}, \gamma_{i,2}) = \operatorname{argmin}_{||(\beta_1, \beta_2)|| \leq \mu\gamma_{i,n}} \boldsymbol{v}^T(\beta_1 \boldsymbol{t}_1 + \beta_2 \boldsymbol{t}_2), \tag{4}$$

where μ is the static friction coefficient. The feasible set of forces $f = f_N + f_T$ for the minimization problem in (4) are known as a *friction cone* $\Upsilon_i = \{(\gamma_{i,n}, \gamma_{i,1}, \gamma_{i,2}) \mid ||(\gamma_{i,1}, \gamma_{i,2})|| \leq \mu\gamma_{i,n}\}$. The complementarity condition in (3) is typically abbreviated using the notation $0 \leq \gamma_{i,n} \perp \Phi_i(\boldsymbol{q}) \geq 0$. We note that an alternate complementarity formulation for (3) can be obtained by replacing Φ_i with the normal velocity.

Since all forces are zero in the absence of contact, we wish to incorporate contact constraints only for pairs of objects approaching collision (e.g. within the next timestep). For this purpose, we define the set $\mathcal{A}$ of pairs of bodies separated by a distance smaller than a threshold $\delta > 0$

$$\mathcal{A}(\boldsymbol{q}, \delta) = \{i | \Phi_i(\boldsymbol{q}) \leq \delta\}. \tag{5}$$

For a system with $N_c = |\mathcal{A}(q, \delta)|$ contacts, this adds a number of constraints to the equations of motion proportional to N_c. We note that for dense granular flows, N_c itself scales as $O(M)$ for M bodies.

Now, let $\boldsymbol{D}_i = [D_{i,n} \ D_{i,1} \ D_{i,2}] \in \mathbb{R}^{6M \times 3}$ mapping the multipliers $\gamma_{i,n}, \gamma_{i,1}, \gamma_{i,2}$ to the forces and torques applied to bodies B_{i_1} and B_{i_2} at contact. Then, the equations of motion result in the differential variational inequality

3

$$(6) \qquad \dot{\boldsymbol{q}} = \mathcal{L}(\boldsymbol{q})\boldsymbol{v},$$

$$(7) \qquad M(\boldsymbol{q})\dot{\boldsymbol{v}} = \boldsymbol{f}_B(\boldsymbol{q}, \boldsymbol{v}) + \sum_{i \in \mathcal{A}(\boldsymbol{q}, \delta)} D_{i,n}\gamma_{i,n} + D_{i,1}\gamma_{i,1} + D_{i,2}\gamma_{i,2},$$

$$(8) \qquad i \in \mathcal{A}(\boldsymbol{q}, \delta) : \gamma_{i,n} \geq 0 \perp \Phi_i(\boldsymbol{q}) \geq 0,$$

$$(9) \qquad (\gamma_{i,1}, \gamma_{i,2}) = \mathrm{argmin}_{||(\beta_1, \beta_2)|| \leq \mu\gamma_{i,n}} \boldsymbol{v}^T(D_{i,1}\beta_1 + D_{i,2}\beta_2).$$

A semi-implicit, first order integration scheme is then used to advance this system in time. Given position $\boldsymbol{q}^k$ and velocity $\boldsymbol{v}^k$ at a given time step t^k and step size Δt, velocity $\boldsymbol{v}^{k+1}$ and contact forces are solved via a nonlinear complementarity problem (NCP). The new velocity is used to evolve the position in time.

$$(10) \qquad \boldsymbol{q}^{k+1} = \boldsymbol{q}^k + \Delta t \mathcal{L}(\boldsymbol{q}^k)\boldsymbol{v}^{k+1},$$

$$(11) \qquad M(\boldsymbol{q}^k)(\boldsymbol{v}^{k+1} - \boldsymbol{v}^k) = \Delta t \boldsymbol{f}_B(\boldsymbol{q}^k, \boldsymbol{v}^k) + \sum_{i \in \mathcal{A}(\boldsymbol{q}^k, \delta)} D_{i,n}\gamma_{i,n} + D_{i,1}\gamma_{i,1} + D_{i,2}\gamma_{i,2},$$

$$(12) \qquad i \in \mathcal{A}(\boldsymbol{q}^k, \delta) : \gamma_{i,n} \geq 0 \perp \frac{1}{\Delta t}\Phi_i(\boldsymbol{q}^k) + D_{i,n}^T \boldsymbol{v}^{k+1} \geq 0,$$

$$(13) \qquad (\gamma_{i,1}, \gamma_{i,2}) = \mathrm{argmin}_{||(\beta_1, \beta_2)|| \leq \mu\gamma_{i,n}} (\boldsymbol{v}^{k+1})^T(D_{i,1}\beta_1 + D_{i,2}\beta_2).$$

We note that (12) is obtained from (8) via a linearization, dividing by Δt (which does not affect complementarity, but is numerically desirable). This makes the future velocity v^{k+1} the sole variable needed to enforce the complementarity condition. We also note that in this discretization, $(\gamma_{i,n}, \gamma_{i,1}, \gamma_{i,2})$ constitute contact *impulses*, i.e. force magnitudes multiplied by the step length Δt.

2.3. Solving the optimization problem. A relaxation over the complementarity constraint (12) can be introduced [Ani06], turning the problem into a convex, second-order cone complementarity problem (CCP).

$$(14) \qquad \gamma_{i,n} \geq 0 \perp \frac{1}{\Delta t}\Phi_i(\boldsymbol{q}^k) + D_{i,n}^T \boldsymbol{v}^{k+1} - \mu_i\sqrt{(D_{i,1}^T \boldsymbol{v}^{k+1})^2 + (D_{i,2}^T \boldsymbol{v}^{k+1})^2} \geq 0.$$

The solution of this relaxed problem approaches the solution of the NCP as the step-size Δt goes to zero. Additionally, the CCP is equivalent to the KKT first-order optimality conditions for a quadratic optimization problem with conic constraints. We define a contact transformation matrix $\boldsymbol{D} = [D_1 \; ; D_2 \; ; \ldots \; D_{N_c}] \in \mathbb{R}^{6M \times 3N_c}$, a matrix $\boldsymbol{N}$ and vector $\boldsymbol{r}$:

$$(15) \qquad \boldsymbol{N} = \boldsymbol{D}^T \boldsymbol{M}^{-1} \boldsymbol{D},$$

$$(16) \qquad \boldsymbol{r} = \boldsymbol{b} + \boldsymbol{D}^T \boldsymbol{M}^{-1} \boldsymbol{k},$$

where $\boldsymbol{b} = [\boldsymbol{b}_1 \; ; \boldsymbol{b}_2 \; ; \ldots \; \boldsymbol{b}_{N_c}]$, $\boldsymbol{b}_i = [\Phi_i^k/\Delta t \; ; 0 \; 0] \in \mathbb{R}^3$ and $\boldsymbol{k} = \boldsymbol{M}\boldsymbol{v}^k + \Delta t \boldsymbol{f}^k$. We note that $\boldsymbol{N}$ is a $3N_c \times 3N_c$ symmetric positive semi-definite matrix and typically sparse. The aforementioned quadratic program is then given by:

$$(17) \qquad \min q(\gamma) = \frac{1}{2}\gamma^T \boldsymbol{N}\gamma + \boldsymbol{r}^T \gamma$$

$$(18) \qquad \text{subject to} \quad ||(\gamma_{i,1}, \gamma_{i,2})|| \leq \mu_i \gamma_{i,n} \quad i = 1, 2, \ldots, N_c.$$

While this relaxed problem may introduce artifacts when step size Δt, sliding velocity or friction are large, it enables the use of a wide range of quadratic programming methods for its solution:
  • Projected Jacobi and Gauss-Seidel methods [TA10].

- Projected gradient descent methods like Accelerated Projected Gradient Descent [MHNT15], Barzi-lai - Borwein [BB88] and the Kucera and Preconditioned spectral projected gradient with fallback (P-SPG-FB) methods in [HATN13].
- Krylov subspace methods: Gradient projected minimum residual (GPMINRES) in [HATN13].
- Primal-Dual Interior Point (PDIP) methods [ADLV11, PGAP09, Fan15].
- Symmetric Cone Interior Point (SCIP) methods [Kle15].

Projected Jacobi and Gauss-Seidel methods, while requiring only fast matrix applies of $\boldsymbol{N}$, have slow, linear convergence, often requiring thousands of iterations to obtain a significant reduction for the objective function. The projected gradient descent and Krylov subspace methods have since been proposed, providing considerable iteration count reductions while retaining cost-efficiency per time step. However, they remain linearly convergent, and they require an increasing number of iterations as problem size increases for a variety of problems of interest.

Interior point methods are often based on a modified Newton or Quasi-Newton step, displaying quadratic convergence near minima and iteration counts which are less dependent on problem size. However, they require the approximate solution of large linear systems in order to produce the Newton step, thus losing this competitive advantage due to computational cost.

2.4. Overview of interior point methods. Interior point methods, also known as barrier methods, are a class of algorithms tailored to solve constrained convex optimization problems [NW06]. That is,

$$\min f_0(\boldsymbol{\gamma}) \tag{19}$$

$$\text{subject to } f_i(\boldsymbol{\gamma}) \leq 0, \ i = 1, \ldots, m \tag{20}$$

for $f_i \in C^2(\mathbb{R}^n)$ and convex. They proceed by transforming this problem into an unconstrained minimization problem, encoding the feasible set defined by the constraints using a barrier function, e.g., the logarithmic barrier $B_t(z) = -(1/t) \log(-z)$. They then pose the unconstrained convex problem:

$$\min \ f_0(\boldsymbol{\gamma}) + \sum_{i=1}^{m} B_t(f_i(\boldsymbol{\gamma})). \tag{21}$$

The parameter t controls the strength of the barrier, and as $t \to \infty$, $B(z) \to I(z)$ with $I(z)$ the indicator function over $(-\infty, 0]$, and the problem in Eq 21 becomes equivalent to the original constrained program defined by Eqs 19 and 20. Given the set of problems defined by Eq 21, interior point methods proceed by following along the corresponding *central path* $\{\boldsymbol{\gamma}^*(t) : t > 0\}$ of optima, which are located inside of the feasible set, and converge to the solution of the original problem as $t \to \infty$.

Primal-Dual Interior Point methods. Primal-dual methods proceed by defining a path of solutions $(\boldsymbol{\gamma}^*, \boldsymbol{\lambda}^*)$. These can be obtained by considering the KKT conditions of Eq 21,

$$f_i(\boldsymbol{\gamma}) < 0, \quad i = 1, \ldots, m \tag{22}$$

$$\nabla f_0(\boldsymbol{\gamma}) + \sum_{i=1}^{m} \frac{-1}{t f_i(\boldsymbol{\gamma})} \nabla f_i(\boldsymbol{\gamma}) = 0 \tag{23}$$

We then define the lagrange multipliers $\boldsymbol{\lambda}_i = \frac{-1}{t f_i(\boldsymbol{\gamma})}$, setting up the following conditions:

$$f_i(\boldsymbol{\gamma}) < 0, \quad i = 1, \ldots, m \tag{24}$$

$$\boldsymbol{\lambda}_i < 0, \quad i = 1, \ldots, m \tag{25}$$

$$-\boldsymbol{\lambda}_i f_i(\boldsymbol{\gamma}) = \frac{1}{t}, \quad i = 1, \ldots, m \tag{26}$$

$$\nabla f_0(\boldsymbol{\gamma}) + \sum_{i=1}^{m} \boldsymbol{\lambda}_i \nabla f_i(\boldsymbol{\gamma}) = 0. \tag{27}$$

We note that, as $t \to \infty$, Eq 26 becomes a strict complementarity condition.
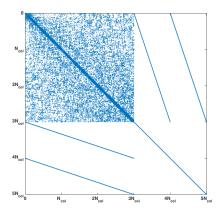
FIG. 1. *Sparsity structure of PDIP Newton system matrix for a general multibody dynamics problem*

The PDIP step. The PDIP step is then obtained by applying Newton's algorithm to solve these KKT conditions. Defining the residual of the system given by Eqs (26) and (27) $\boldsymbol{r}_t(\boldsymbol{\gamma}, \boldsymbol{\lambda})$ as:

$$(28) \qquad \boldsymbol{r}_t(\boldsymbol{\gamma}, \boldsymbol{\lambda}) = \begin{bmatrix} \nabla f_0(\boldsymbol{\gamma}) + \nabla \boldsymbol{f}(\boldsymbol{\gamma})^T \boldsymbol{\lambda} \\ -diag(\boldsymbol{\lambda}) \boldsymbol{f}(\boldsymbol{\gamma}) - \frac{1}{t} \mathbf{1} \end{bmatrix} = 0$$

the Newton step is then defined by the solution of the linear system given by:

$$(29) \qquad \begin{bmatrix} \nabla^2 f_0(\boldsymbol{\gamma}) + \sum_{i=1}^{m} \boldsymbol{\lambda}_i \nabla^2 \boldsymbol{f}_i(\boldsymbol{\gamma}) & \nabla \boldsymbol{f}(\boldsymbol{\gamma})^T \\ -diag(\boldsymbol{\lambda}) \nabla \boldsymbol{f}(\boldsymbol{\gamma}) & -diag(\boldsymbol{f}(\boldsymbol{\gamma})) \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{\gamma} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = -\boldsymbol{r}_t(\boldsymbol{\gamma}, \boldsymbol{\lambda})$$

this is typically coupled with a backtracking line search, and a strategy to increase t until sufficient convergence to the solution of the original problem is satisfied.

Application to the CCP. For the equivalent quadratic program with conic constraints in Eqs (17) - (18), we define $f_0(\boldsymbol{\gamma}) = q(\boldsymbol{\gamma}) = \frac{1}{2} \boldsymbol{\gamma}^T \boldsymbol{N} \boldsymbol{\gamma} + \boldsymbol{r}^T \boldsymbol{\gamma}$ and $2N_c$ inequality constraints (Eq 18) given by $f_i(\boldsymbol{\gamma})$ as:

$$(30) \qquad \boldsymbol{f}_i(\boldsymbol{\gamma}) = \begin{cases} \frac{1}{2}(\gamma_{i,1}^2 + \gamma_{i,2}^2 - \mu_i^2 \gamma_{i,n}^2) & i = 1, \dots, N_c \\ -\gamma_{i-N_c, n} & i = N_c + 1, \dots, 2N_c \end{cases}$$

the Newton equations that define the PDIP step thus require the solution of a linear system of the form:

$$(31) \qquad \begin{bmatrix} \boldsymbol{N} + \hat{\boldsymbol{M}} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{E} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{\gamma} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} r_{\boldsymbol{\gamma}} \\ r_{\boldsymbol{\lambda}} \end{bmatrix}$$

where $\hat{\boldsymbol{M}}$ is a diagonal matrix defined by $\hat{\boldsymbol{M}} = \sum_{i=1}^{2N_c} \boldsymbol{\lambda}_i \nabla^2 \boldsymbol{f}_i(\boldsymbol{\gamma}) = diag(\hat{\boldsymbol{m}})$, with $\hat{\boldsymbol{m}} = [\mu_1^2 \lambda_1, \lambda_1, \lambda_1, \dots, \mu_1^2 \lambda_{N_c}, \lambda_{N_c}, \lambda_{N_c}]$. $\boldsymbol{B}, \boldsymbol{C}$ are banded rectangular matrices, and $\boldsymbol{E}$ is diagonal. A typical sparsity pattern for a multibody dynamics problem is shown in Fig. 1. In order to find the Newton step, we must then solve this sparse linear system. We may either proceed directly, or by eliminating $\Delta \boldsymbol{\lambda}$, a reduced, symmetric positive definite Schur complement matrix of size $3N_c \times 3N_c$ for $\Delta \boldsymbol{\gamma}$ can be obtained.

3. The Tensor Train solver. In the context of multibody dynamics, we know the system matrices for the Newton step in (31) to be sparse and highly structured, and dependent on the iterates $(\boldsymbol{\gamma}, \boldsymbol{\lambda})$ as the PDIP iteration proceeds to the solution of problem (17). Further, from one time step to the next, we expect the matrix required to obtain the Newton step to change in size as the active set of constraints (corresponding to pairs of objects in contact) evolves. It is because of these changes both within and between timesteps that producing an efficient and robust solution technique for the Newton system remains challenging.
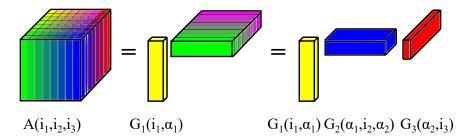
6

$$A(i_1,i_2,i_3) \qquad G_1(i_1,\alpha_1) \qquad G_1(i_1,\alpha_1)\, G_2(\alpha_1,i_2,\alpha_2)\, G_3(\alpha_2,i_3)$$

FIG. 2. **TT decomposition example:** *compression of 3 tensor from samples of $f(x,y,z) = sin(x+y+z)$ through a chain of two rank 2 matrix decompositions corresponding to angle addition formulas in (32) and (33). TT cores $\mathcal{G}_i$ are obtained through a chain of low rank matrix decompositions of unfolding matrices.*

Amongst currently available hierarchical compression techniques, the TT decomposition features compression and inversion algorithms that are applicable to a large set of structured matrices and that lend themselves to inexpensive global updates. In addition, they have shown to achieve sublinear precomputation times. We thus propose to use it as a framework for direct solution and preconditioning of iterative solvers for the linear systems in each PDIP iteration.

In this section, we give a cursory description of the quantized tensor train (QTT) decomposition as a method to efficiently compress, invert and perform fast arithmetic with approximate representations of structured matrices. We then present a general discussion of its application to solving the linear systems associated with the PDIP for the CCP. This constitutes, to our knowledge, the first application of hierarchical compression solvers to the acceleration of second order optimization methods. Although it is beyond the scope of this work, we expect the techniques laid out in this section to be readily applicable to a more general class of interior point and other Newton and Quasi Newton based methods for smooth convex problems.

3.1. The tensor train decomposition . The tensor train (TT) decomposition provides a powerful tensor compression technique [OT10] via low rank representation akin to that of a generalized SVD. We will focus on its application in the approximation of *tensorized* vectors and matrices given a hierarchical subdivision of their indices, known as quantized tensor train (QTT). Matrices in this setting are further interpreted as tensorized operators acting on such tensorized vectors. We outline how this interpretation allows us to effectively employ the TT as a tool for hierarchical compression and inversion of structured matrices.

Since this approach involves considering reshaping, tensorization and vectorization of arrays, we introduce the following index notation: For the tensor multi-index $(i_1, \dots, i_d)$, the one-dimensional index obtained by lexicographic ordering of multi-indices will be denoted by placing a bar on top, removing commas between indices: $i = \overline{i_1 i_2 \cdots i_d}$. This mapping from multi-indices to one-dimensional index corresponds to the conversion of a multidimensional array to a vector which we denote $\mathsf{b} = \mathrm{vec}(\mathcal{b})$, with $\mathsf{b}(\overline{i_1 i_2 \cdots i_d}) = \mathcal{b}(i_1, i_2, \dots, i_d)$.

A motivating example. Suppose we wish to compress the 3-tensor $\mathcal{A}(i_1, i_2, i_3)$ obtained from sampling the function $f(x,y,z) = sin(x+y+z)$ on a uniform grid with n^3 points $(x_{i_1}, y_{i_2}, z_{i_3})$ in $[0,1]^3$. We can use addition formulas to decompose $f(x,y,z)$ as a sum of separable functions:

$$\tag{32} sin(x+y+z) = \begin{bmatrix} sin(x) & cos(x) \end{bmatrix} \begin{bmatrix} cos(y+z) \\ sin(y+z) \end{bmatrix}$$

$$\tag{33} = \begin{bmatrix} sin(x) & cos(x) \end{bmatrix} \begin{bmatrix} cos(y) & -sin(y) \\ sin(y) & cos(y) \end{bmatrix} \begin{bmatrix} cos(z) \\ sin(z) \end{bmatrix}$$

Evaluation of (32) and (33) on the tensor $\mathcal{A}$ is depicted in Fig 2. The first step (32) produces a rank 2 decomposition $\mathcal{A}(i_1, i_2, i_3) = \mathcal{G}_1(i_1, \alpha_1)\mathcal{V}_1(\alpha_1, i_2, i_3)$. In (33) we then decompose $\mathcal{V}_1$ to separate dependency of y and z; each of the two terms results in a rank 2 decomposition $\mathcal{V}_1(\alpha_1, i_2, i_3) = \mathcal{G}_2(\alpha_1, i_2, \alpha_2)\mathcal{G}_3(\alpha_3, i_3)$. We note that each term G_k (depicted in solid yellow, blue and red in Fig 2) depends on one dimension i_k of the original tensor $\mathcal{A}$, and storage has been reduced from n^3 to $2n + 4n + 2n = 8n$.

What we have presented in this example is an exact tensor train decomposition of $\mathcal{A}$. We present the analogous definition for the TT decomposition of a d dimensional tensor.

DEFINITION 1. *For a d-dimensional tensor $\mathcal{A}(i_1, i_2, \ldots, i_d), i_k \leq n_k$, sampled at $N = \prod_{k=1}^{d} n_k$ points, a TT decomposition is of the form*

$$(34) \qquad \mathsf{A}(i_1, i_2, \ldots, i_d) := \sum_{\alpha_1, \ldots, \alpha_{d-1}} \mathcal{G}_1(i_1, \alpha_1) \mathcal{G}_2(\alpha_1, i_2, \alpha_2) \ldots \mathcal{G}_d(\alpha_{d-1}, i_d),$$

where, each two- or three-dimensional $\mathcal{G}_k$ is known as a tensor core. The ranges of auxiliary indices $\alpha_k = 1, \ldots, r_k$ determine the number of terms in the decomposition. We refer to r_k as the kth TT-rank, analogous to matrix numerical rank.

In order to understand the chain of "low rank" decompositions in the tensor train format in terms of matrix low rank decompositions, we introduce auxiliary objects known as unfolding matrices.

DEFINITION 2. *For a tensor of dimension d, the kth unfolding matrix is defined as*

$$(35) \qquad \mathsf{A}^k(p_k, q_k) = \mathsf{A}^k(\overline{i_1 i_2 \cdots i_k}, \overline{i_{k+1} \cdots i_d}) = \mathcal{A}(i_1, i_2, \cdots, i_d) \quad for \quad k = 1, \ldots, d,$$

where $p_k = \overline{i_1 \cdots i_k}$ and $q_k = \overline{i_{k+1} \cdots i_d}$ are two flattened indices. Using Matlab's notation

$$(36) \qquad \mathsf{A}^k = \mathtt{reshape}\left(\mathcal{A}, \prod_{\ell=1}^{k} n_\ell, \prod_{\ell=k+1}^{d} n_\ell\right).$$

The ranks of the TT decomposition are thus the ranks of unfolding matrices. For instance, if we consider the first unfolding matrix A^1 of tensor $\mathcal{A}$ in our example, its rows will depend on x_{i_1}, and its columns on y_{i_2}, z_{i_3}. Eq (32) clearly implies that A^1 is exactly of column rank 2, and the resulting matrix decomposition

$$(37) \qquad \mathsf{A}^1 = \mathcal{G}_1(i_1, \alpha_1) \mathcal{V}_1(\alpha_1, \overline{i_2 i_3})$$

is interchangeable with the first rank 2 decomposition of $\mathcal{A}$, requiring only to merge indices i_2, i_3 in $\mathcal{V}_1$. We may similarly show that A^2 is of rank 2 (by applying addition formulas for $(x+y)$ and z); however, the second rank 2 approximation in the chain is obtained by decomposing an unfolding matrix of $\mathcal{V}_1$:

$$(38) \qquad \mathsf{V}_1^2(\overline{\alpha_1 i_2}, i_3) = \mathcal{G}_2(\overline{\alpha_1 i_2}, \alpha_2) \mathcal{G}_3(\alpha_2, i_3)$$

As is the case with low rank matrix decompositions, most often a tensor $\mathcal{A}$ of interest will be approximately of low rank, in the sense that given a target accuracy ε, a TT decomposition with low TT ranks A may be found such that $||\mathcal{A} - \mathsf{A}||_F < \varepsilon$. This decomposition can be obtained by a sequence of low-rank approximations to A^k. A generic algorithm proceeds as in Algorithm 1.

Algorithm 1 TT decomposition

Require: Tensor $\mathcal{A}$, and target accuracy ε

1: $\mathsf{M}_1 = \mathsf{A}^1$ `// First unfolding matrix`

2: $r_0 = 1$

3: **for** $k = 1$ to $d - 1$ **do**

4:      $[\mathsf{U}_k, \mathsf{V}_k] = \mathtt{lowrank\_approximation}(\mathsf{M}_k, \varepsilon/\sqrt{d-1})$

5:      $r_k = \mathtt{size}(\mathsf{U}_k, 2)$ `// kth TT rank`

6:      $\mathcal{G}_k = \mathtt{reshape}(\mathsf{U}_k, [r_{k-1}, n_k, r_k])$ `// kth TT core`

7:      $\mathsf{M}_{k+1} = \mathtt{reshape}\left(\mathsf{V}_k, [r_k n_{k+1}, \prod_{\ell=k+2}^{d} n_\ell]\right)$ `// M_{k+1} corresponds to the (k+1)th unfolding matrix`
                                                                            `of A`

8: **end for**

9: $\mathcal{G}_d = \mathtt{reshape}(\mathsf{M}_d, [r_{d-1}, n_d, 1])$ `// Set last core to the right factor in the low rank`
                                                                           `decomposition`

10: **return** A

Due to the cost of successive low rank factorizations, implementing Algorithm 1 will predictably lead to relatively high computational cost, $O(N)$ or higher, which is exponential in the tensor dimension d. Instead, we employ TT rank revealing strategies based on the multi-pass AMEN (Alternating Minimal Energy) Cross algorithm of [OT10], in which a low-TT-rank approximation is initially computed with fixed ranks and is improved upon by a series of passes through all cores.

TT compression update. We note that, since these algorithms obtain the TT approximation based on an iterative, greedy rank detection procedure, they allow for inexpensive updates to a TT compressed representation of a tensor $\mathcal{A}$. If we wish to produce the TT representation of $\tilde{\mathcal{A}} = \mathcal{A} + \mathcal{E}$, the AMENCross algorithm may be started using the TT representation of $\mathcal{A}$ as an initial guess. If $\mathcal{E}$ has small TT ranks, this provides a significant speed-up for this compression algorithm, which often converges in a few iterations to the updated representation.

Computational complexity and memory requirements. Because AMEN Cross and related TT rank revealing approaches proceed by enriching low-TT-rank approximations, all computations are performed on matrices of size $r_{k-1} n_k \times r_k$ or less. In [CRZ15], it is shown that the complexity for this algorithm is thus bounded by $O(r^3 d)$ or equivalently $O(r^3 \log N)$, where $r = \max(r_k)$ is the maximal TT-rank that may be a function of sample size N and accuracy ε. For a large number of structured matrices as well as their inverses, r typically stays constant or grows logarithmically with N [KK12, Ose10, OTZ11, CRZ15]. The overall complexity of computations is then *sublinear* in N.

FIG. 3. ***TT decomposition for a matrix with*** $d = 3$. *Each level of refinement for the matrix block hierarchy corresponds to one tensor dimension and TT core. Columns of the corresponding unfolding matrix are obtained by vectorizing matrix blocks (in red if non-zero, white otherwise). The example illustrated above uses an interpolatory low rank decomposition, producing a uniform subsampling of tree nodes. The TT decomposition process thus consists of finding a hierarchical basis of matrix block entries. We show the main steps in algorithm 1: (i) computing a low rank decomposition* $\mathsf{U}_k \mathsf{V}_k$ *for the corresponding unfolding matrix* M_k; *(ii) taking* U_k *(in green) as the* k*th TT core* $\mathcal{G}_k$; *and (iii) interpreting the right factor* V_k *as a matrix in level* $k+1$ *to form* M_{k+1}.

3.2. TT for hierarchically structured matrices. Consider a block-sparse, structured matrix A. For simplicity of presentation we assume matrix rows and columns to be of size $N = n_1 2^{d-1}$. We then recursively bisect them, representing the resulting hierarchy with a binary tree $\mathcal{T}$ (number of children $n_k = 2$) with n_1 points in each leaf node. Let $\mathcal{T}_\Pi$ then denote the product tree $\mathcal{T} \times \mathcal{T}$; nodes on this tree correspond to pairs of source and target nodes. $\mathcal{T}_\Pi$ can be understood as a hierarchy of matrix-blocks, or equivalently, of all interaction operators between subsets B_i and B_j at a given level ℓ of $\mathcal{T}$. Each node on this tree can thus be indexed by integer row and column index coordinate pairs (i_k, j_k) with $k \leq \ell$. Equivalently, we can consider block integer coordinates $b_k \in \{1, \cdots, n_k^2\}$ for $\mathcal{T}$ such that $b_k = \overline{i_k j_k}$. We then apply the TT decomposition to the corresponding tensorized form of A, $\mathcal{A}_\mathcal{T}$, a d-dimensional tensor with entries defined as

(39) $$\mathcal{A}_\mathcal{T}(b_1, b_2, \ldots, b_d) = \mathcal{A}_\mathcal{T}(\overline{i_1 j_1}, \overline{i_2 j_2}, \ldots, \overline{i_d j_d}) = \mathsf{A}(\overline{i_1 i_2 \cdots i_d}, \overline{j_1 j_2 \cdots j_d}),$$

and obtain an approximate TT factorization A. Each core of A, $\mathcal{G}_k(\alpha_{k-1}, \overline{i_k j_k}, \alpha_k)$ depends only on the pair of source and target tree indices at the corresponding level of the hierarchy. When performing matrix arithmetic, such as matrix-vector product or inversion, TT cores are often reshaped as $n_k \times n_k$ matrices parametrized by α_{k-1} and α_k.

In Fig. 3, we demonstrate the TT decomposition algorithm applied to a matrix A for binary source and target trees with depth $d = 3$ and three points in the leaf nodes $n_1 = m_1 = 3$, implying $N = 12$. In this example, the tree $\mathcal{T}_\Pi$ is a matrix-block quadtree. At each level of the hierarchy, every column of the corresponding unfolding matrix A^k is a vectorized form of the block A_{ij}. We note that, for a block-sparse matrix A, the majority of these blocks, and hence columns of A^k at intermediate levels k of the tree are identically zero. This, along with any additional block structure in A ensures that A^k is approximately low rank.

Fast arithmetic and TT direct solvers. Fast methods for TT matrix arithmetic are available for a number of operations, including inversion and matrix-vector and matrix-matrix products. The TT solvers in this work find an approximate TT structure for the inverse and employ the corresponding TT matrix-vector product $\gamma = \mathsf{A}^{-1} b$. This mat-vec algorithm proceeds by contracting one dimension of the tensorized matrix A^{-1} at a time, applying the kth TT core. Its complexity can be shown to be $O(r^2 N \log N)$.

TT inversion methods compute an approximate TT decomposition of A^{-1} given the TT decomposition of A. In these algorithms, matrix equations for each tensor core of the inverse is solved iteratively. Following an Alternating Least Squares (ALS) algorithm, given an initial guess for the inverse in TT form, it proceeds by iteratively cycling through the cores (freezing all cores but one) and solving a linear system to update the kth core of A^{-1}. TT ranks of the inverse are not known a priori (and are distinct to those for A), and so strategies to increase core ranks are needed to ensure convergence of the ALS procedure to an accurate inverse representation. Further details about variants of this approach can be found in [OD12, DS13a, DS13b]. The complexity of this algorithm for a maximum TT rank r for both A and A^{-1} is bound by $O(r^4 \log N)$, as shown in [CRZ15].

Inverse compression update. We note that, since they share the same iterative, greedy rank detection structure with the AMENCross compression algorithm, TT direct solver routines may also be significantly sped up using an approximate initial solution. In the context of the Newton step in interior point methods, we expect both forward and inverse operators to be obtainable via low TT rank updates.

3.3. TT Newton system solver. We now discuss how to adapt the tensor train decomposition framework to accelerate computation of the Newton steps within the PDIP method applied to the CCP. We first show how at a given timestep, the tensor train provides an easy-to-update approximate inverse for the Newton system's Schur complement matrix. We then describe a procedure to hot-start the TT compression and inversion algorithms re-using information from the previous timestep, even when the corresponding set of contacts (and thus, matrix size and structure) change.

PDIP iteration. At a given timestep t, the kth PDIP iteration involves the solution of the Newton system in 31, with a $5N_c \times 5N_c$ system matrix of the form

(40) $$\boldsymbol{A}_k = \begin{bmatrix} \boldsymbol{N} + \hat{\boldsymbol{M}}_k & \boldsymbol{B}_k \\ \boldsymbol{C}_k & \boldsymbol{E}_k \end{bmatrix}$$

where $\boldsymbol{N}$ is fixed and the rest of the matrix blocks involved depend on the current iterate $\boldsymbol{\gamma}_k, \boldsymbol{\lambda}_k$. We may, alternatively, choose to solve the Schur-complement system for $\Delta\boldsymbol{\gamma}$, resulting in the $3N_c \times 3N_c$ matrix:

(41) $$\boldsymbol{S}_k = \boldsymbol{N} + \hat{\boldsymbol{M}}_k - \boldsymbol{B}_k \boldsymbol{E}_k^{-1} \boldsymbol{C}_k$$

We note that working with the Schur complement $\boldsymbol{S}_k$ is generally preferable, as it is symmetric positive definite and smaller in size. We observe that its sparsity pattern across iterations is always that of matrix $\boldsymbol{N}$. That is due to the fact that $\hat{\boldsymbol{M}}_k$ is diagonal and $\boldsymbol{B}_k \boldsymbol{E}_k^{-1} \boldsymbol{C}_k$ is block-diagonal (3×3 blocks).

We recall that matrix N, the Hessian of the quadratic $q(\gamma)$, is of the form $D^T M^{-1} D$ (Eq (15)), with $D \in \mathbb{R}^{6M \times 3N_c}$ the contact transformation matrix and M the diagonal mass matrix. In order to understand the sparsity pattern of N, we partition it into 3×3 blocks corresponding to each contact. The (ℓ, i)th block of D is non-zero if the ith contact involves body B_ℓ. As a result, the (i, j)th block of N is non-zero if the ith and jth contacts share a body in common. In Fig. 4, we show a simple example with three spherical bodies lying on a flat surface. Given the network of bodies at contact in (b), the sparsity pattern in N corresponds to the edge-adjacency graph in (c); two two edge nodes (indexed by body pairs) are connected if they share a body in common.

FIG. 4. **Collision graphs example:** *(a) configuration of three spherical bodies lying on a flat surface. Centers of mass are depicted as red circles, collision points as white squares. (b) Graph connecting bodies that are in contact (c) Graph connecting contacts sharing a body in common. This graph reveals the block-sparsity pattern for N and S.*

A hierarchy for contact pairs may be generally constructed from the corresponding adjacency graph using graph partition techniques such as the nested dissection method. For the cases of interest in this work, this hierarchy can be readily obtained from a spatial octree hierarchy of the positions of contact pairs in $3d$ space, as long as care is put to separate those associated with large contact geometries (e.g. those associated with walls or containers such as the flat surface in Fig 4).

As indicated in Section 3.2, given a hierarchical partition of matrix indices, at each timestep we tensorize the Schur complement S_k, and for a given target accuracy ε we construct approximate TT decompositions S_k and S_k^{-1}. We then have the option to use S_k^{-1} as a direct solver, or couple it with an iterative procedure if a more accurate solve is needed (e.g. iterative refinement, or a preconditioned Krylov subspace method). We then consider S_{k+1} as a perturbation

$$S_{k+1} = S_k + L_k, \tag{42}$$

where L_k is block-diagonal. While L_k is generally of matrix rank $O(N_c)$, across all experiments in Section 4 we observe it to be of approximate low TT rank, and the same is found for S_{k+1}^{-1} as a perturbation of S_k^{-1}. This fact allows us to use the TT decompositions $\mathsf{S}_k, \mathsf{S}_k^{-1}$ to hot-start the corresponding compression and inversion algorithms, reducing precomputation times considerably.

Re-using information across timesteps. Employing information from the solution of the CCP at a timestep t to hot-start the PDIP iteration at the next timestep $t + \Delta t$ is notoriously hard; even if it can be used to produce a feasible point that is close to the optimum (which is non-trivial due to changes in the set of contacts), efforts by the PDIP algorithm to preserve centrality might cause it to take small steps and waste time "returning" to the central path.

For this reason, we initialize each PDIP iteration by making the tangential force impulses $\gamma_{i,1}, \gamma_{i,2}$ equal to zero, and the normal force $\gamma_{i,n}$ equal to either a constant preset value (e.g. 1) or to the value computed in the previous timestep if the corresponding contact persists across timesteps. This ensures that our initial value is feasible and lies safely inside the friction cone.

Since pairs of bodies may phase in and out of contact, the set of contacts considered at each timestep changes. However, as long as Δt and relative velocities are sufficiently small, it is likely that the set of persistent contacts from one timestep to the next will be large. In all validation experiments in Section 4, in fact, well over 90% of contacts persist once objects have sedimented. We may then re-use the TT

decompositions for the *initial* Schur complement matrix. Care must be taken to make this decomposition compatible, introducing new contacts into the hierarchy and "deleting" contacts that have ceased to exist. A simple example of this is depicted in Fig 5.
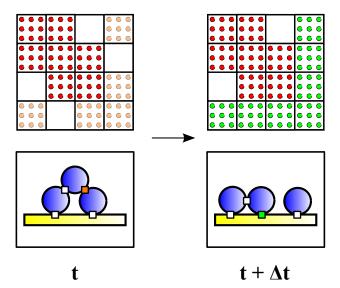

FIG. 5. ***Change in contact set and matrix entries:*** *From time t to $t + \Delta t$, we depict a change in the configuration of the three bodies in Fig 4 as the system evolves. Above each configuration, we depict the corresponding block-sparsity pattern for the initial Schur complement matrices. Persistent matrix entries are depicted in red, entries being removed in orange and entries being introduced in green.*

In order to use the TT-factorization at timestep t as shown in Fig 3 to form an initial approximation for compression and inversion at the next timestep, we need to reconfigure the TT cores so that they remain an approximation for the submatrix for persistent contacts (in red). For instance, if they correspond to a hierarchical interpolation of matrix entries, we could keep the first $d - 1$ cores (containing interpolation weights) and update the entries in $\mathcal{G}_d$ corresponding to new contacts (in green).

We note that assigning tensor indices to "new body pairs" and eliminating "old" ones requires us to track and modify the spatial hierarchy that is used to encode our matrix as a tensorized array. This requires a certain degree of flexibility and adaptivity, if representations for t and $t + \Delta t$ are to be compatible. In this work, we resolve this issue by considering our hierarchy as an adaptive tree which is itself a subset of a uniform tree with $\tilde{N}_c = 2^L$ elements. Any elements outside our current hierarchy are dummy variables, and the corresponding matrix for this augmented set of degrees of freedom is a permutation of a matrix with two diagonal blocks, A and I the identity. We detail a small one-dimensional example of this in Fig 6.

This setup allows us to produce a compatible initial guess for $\mathtt{A}$ and $\mathtt{A}^{-1}$ across timesteps as long as the adaptive tree can be properly updated and still fits within the regular tree with $\tilde{N}_c$ leaf nodes. New nodes take the place of dummy variables, and persistent nodes are potentially re-indexed if they move too far. Otherwise, we reset this structure and compute the TT factorizations from scratch.

4. Numerical Results. We now demonstrate the performance of the TT-based solver when accelerating the solution of the Newton step system, and thus of second order methods such as PDIP, in the context of dense, multiple rigid-body dynamics. As mentioned in Sections 1 and 3, we know that given a desired target accuracy ε, if the associated TT ranks r_k are bounded or slowly growing as a function of problem size N, factorization costs and storage for the TT solver are *sublinear* in N, and that applying this matrix to a given right-hand-side is $O(N \log N)$. We wish to study if Newton system matrices are TT compressible in this sense, and to test their performance and scaling in ther solution as problem size grows (determined by the number of collisions N_c). By harnessing the ability of the TT to produce economic updates both from one Newton iteration to the next, as well as across timesteps, we demonstrate significant improvement for the solution of the complementarity problem using second order methods.

The complementarity method for frictional contact, as well as the solution methods for its CCP relaxation have been thoroughly validated and contrasted with experimental data [MJN16, MFJN17]. We
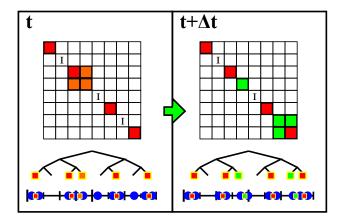
FIG. 6. **TT tensor index hierarchy evolution:** *We depict the evolution of a spatial hierarchy used to index pairs of bodies at collision from a simple one-dimensional example as it evolves from timestep t to $t + \Delta t$. Above each configuration, we show the corresponding binary tree and sparse matrix structure for the Schur complement matrix. Persistent collision pairs and corresponding matrix entries are depicted in red, pairs being removed in orange and pairs being introduced in green. In this example, one of 5 nodes is removed and two new nodes are introduced.*

set up three experiments based on standard phenomena in terramechanics, focusing on how TT-based linear solvers perform in the PDIP iteration. Models and contact dynamics simulations are performed using open-source library Project Chrono [TSM+15], PDIP solver comparisons are performed with a serial Matlab implementation and all TT methods are based on the Matlab TT-Toolbox [Ose12]. All experiments are run on the serial queue of University of Michigan's Flux computing cluster.

4.1. TT compressibility and information re-use. We first carry out an assessment to determine how the general behavior of TT ranks for the Schur matrix and its inverse vary with target accuracy ε and maximum allowed TT ranks, varying these parameter from 10^{-2} to 10^{-4} and $r \leq 10, 100, 1000$, respectively. This gives us a general idea of the compressibility of these matrices upon re-ordering their nodes according to a spatial hierarchy. Since algorithmic constants for TT compression, inversion and matrix-vector apply all depend on rank, this also informs out choice of ε. We present average results for 1000 PDIP iterations for a sedimentation experiment with $M = 35939$ rigid bodies in Section 4.2; we note these are largely replicated across experiments, and that ranks and compression times grow very slowly with the number of degrees of freedom.

	$r \leq 10$	$r \leq 100$	$r \leq 1000$
$\varepsilon = 10^{-2}$	4.6/4.4	10.5/6.7	10.5/6.7
$\varepsilon = 10^{-3}$	8.7/8.5	59.3/45.2	65.2/46.4
$\varepsilon = 10^{-4}$	9.1/9.0	68.6/58.1	113.3/91.2

TABLE 1

Average TT ranks of Schur matrix: Setting target accuracy ε and maximum rank r, we record average ranks for 1000 timesteps of a sedimentation simulation of $M = 35939$ rigid bodies.

	$r \leq 10$	$r \leq 100$	$r \leq 1000$
$\varepsilon = 10^{-2}$	10	25	25
$\varepsilon = 10^{-3}$	16	1244	1317
$\varepsilon = 10^{-4}$	16	2740	8157

TABLE 2

Average TT compression times for Schur matrix: Setting target accuracy ε and maximum rank r, we record average compression times (in seconds) for the entire PDIP solver ($\sim 50 - 100$ iterations per timestep) for 1000 timesteps of a sedimentation simulation of $M = 35939$ rigid bodies.

In Table 1 we observe that the Schur matrices and their inverses are indeed highly compressible, and

13

that as we increase the maximum allowed rank, individual and average TT ranks converge. This is replicated in all our experiments, regardless of number of particles and number of collisions. In Table 2 we record the combined matrix compression and inversion times; as indicated in Section 3.2, performance of these algorithms is bounded by terms of the form $r^k \log N_c$. Given that rank growth results in a substantial increase in precomputation times, in our experiments we find that a TT preconditioner approach with $\varepsilon = 10^{-2}$ is most effective in reducing overall computation for the PDIP iterations. We note, however, that this parameter selection is generally problem and implementation dependent.

We then wish to quantify the impact of the strategies described in Section 3.3 to re-use information in TT factorizations for the matrix and its inverse. For this purpose, we run two versions of the TT preconditioned PDIP iteration with and without factorization re-use for 100 timesteps for the same sedimentation problem described above. We set target accuracy to $\varepsilon = 10^{-2}$ and bound maximum ranks at $r \leq 10$.

In Fig 7, we plot precomputation times for the initial Newton steps and for the entire PDIP iteration, in order to measure how effective our information re-use strategies for the TT are in bringing down compression and inversion costs. For the initial PDIP iteration, we see that except for the first timestep (for which both methods have no prior information), information re-use always provides a speedup, between 5 and 15x for most timesteps shown here. The accumulated effect of both re-use strategies can be seen in the second plot on the right; precomputation is improved for all timesteps, resulting in a 10 to 15x overall speedup. Across all experiments, we observe both an improvement by about an order of magnitude and a drastic reduction in the variance of precomputation times, making the TT approach faster and more robust. We also note that predictably, this speedup is larger for higher target accuracies, as the iterations through all TT cores become more computationally burdensome.
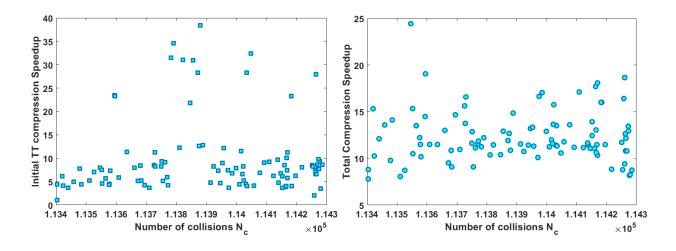

FIG. 7. *TT compression speedups due to information re-use:* Left: *we plot the ratio of compression times without and with information re-use for the initial Newton step for each timestep against number of collisions N_c;* Right: *we plot the same ratio for total compression times for the PDIP iterations.*

4.2. Performance comparison experiments for PDIP iteration.

Sedimentation on box with rotating mixer. A randomly pertubed cubic lattice of $(2n+1)^3$ rigid particles of spherical shape of radius 0.1 and friction coefficient $\mu = 0.25$ are dropped and sediment under gravity into a fixed box with a rotating mixer, as shown in Fig. 8. We then run simulations for $n = 8, 16, 32$, with a total of rigid bodies $M = 4915, 35939$ and 274627 (including the box and mixer), scaling up the box size to keep particle density roughly constant. We set a timestep size $\Delta t = 0.025$ and target accuracy $1e\text{-}4$ for the PDIP solver, until such time as all objects are deposited in the box and undergoing mixing.
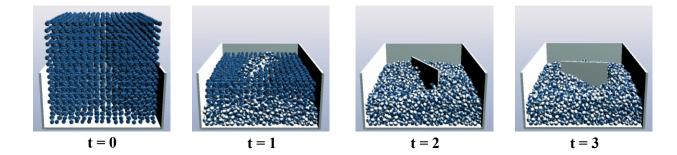
14

| t = 0 | t = 1 | t = 2 | t = 3 |

FIG. 8. **Sedimentation with rotating mixer:** *snapshots of simulation for sedimentation of* 4913 *spheres on a box shaped container with a rotating mixer with constant angular velocity.*

We compute the Newton step through the solution of the corresponding Schur complement system. In order to test performance of the TT-based preconditioner, we compare precomputation and solution times for a preconditioned biconjugate gradient stabilized (BICGSTAB) method against Incomplete LU and unpreconditioned versions of this iterative solver. We note that while use of the conjugate gradient (CG) method may be generally preferrable for these systems, we observe its performance may degrade due to ill-conditioning as iterates approach the feasible set boundary, and so for simplicity of presentation we exclude it from our comparison.

As discussed in Section 3.3, the Tensor Train approach allows us to produce approximate direct and preconditioned iterative solvers by varying target accuracy and maximum TT rank in the compression and inversion processes. Following preliminary parametric studies, we find that setting target accuracy for TT inversion to $1e$-2 and capping maximum TT ranks at $r \leq 10$ provides the best performance in terms of the trade-offs involved in precomputation and TT preconditioner apply costs for our experimental setup.

We note that for practical purposes, the maximum number of iterations for the unpreconditioned solve was set at 1000; this limit was often reached reducing the overall accuracy of the resulting linear system solution and thus the quality of the PDIP iteration. Through further testing, we consistently observe average BICG iteration counts of 3 - 5 $\times 10^3$ and a 5-6 fold increase in computational cost when removing this constraint. These estimates should be considered whenever comparing either of the preconditioned methods against unpreconditioned BICG.
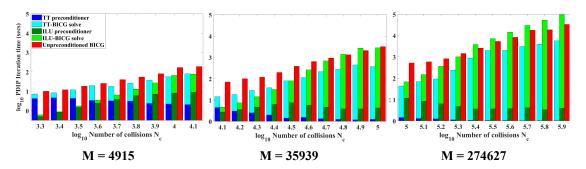

| M = 4915 | M = 35939 | M = 274627 |

FIG. 9. **Log-log plot solver comparison:** *for each experiment, we bin according to $\log_{10} N_c$ and plot average PDIP iteration times; Unpreconditioned BICGSTAB in red, ILU-BICGSTAB in green and TT-BICGSTAB in blue. For preconditioned solvers, we display the proportion spent in precomputation in a darker shade, solve times in a lighter one.*

In Fig. 9 we can observe how performance for each of these solvers scales with the number of collisions N_c. Two factors are contributing to increase problem complexity in this experiment: as objects sediment in the box, the number of collisions tends to increase and the Schur complement matrix becomes less sparse. From this plot, we can readily observe that the TT preconditioned solver generally shows superior scaling, outperforming the ILU preconditioner at about $N_c \sim 20000$, and gaining an order of magnitude speed-up against the ILU preconditioner by the end of experiments with $M = 35939, 274627$. We can observe that while the fraction of time spent in precomputation tends to a constant for ILU (indicating similar asymptotic scaling for precomputation and solve times, experimentally $O(N_c^3)$), it quickly goes down to zero for the TT.

15

In Fig. 10, we take a closer look at average precomputation times for each PDIP iteration for TT and ILU. We note that this involves computing roughly 50-100 factorizations, one per Newton iteration. In all experiments, we confirm that compression and inversion times for the Tensor Train approach grow extremely slowly with N_c, staying on a range from 5 to 15 seconds.
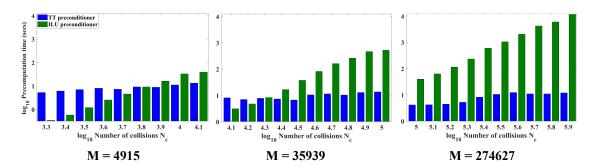

M = 4915 **M = 35939** **M = 274627**

FIG. 10. ***Log-log plot precomputation comparison:*** *for each experiment, we bin according to $\log_{10} N_c$ and plot average preocmputation times; ILU-BICGSTAB in dark green and TT-BICGSTAB in dark blue.*

Finally, we compare BICGSTAB average iteration counts in Fig. 11. We note that while iteration counts generally increase at the beginning of the experiment, those for the TT grow slower and settle sooner as particles sediment; for $M = 35939, 274627$, the number of iterations for the ILU becomes up to 8 times larger. For the unpreconditioned case, the maximum iteration count of 1000 is reached for a significant number of linear system solves, limiting the accuracy of the resulting Newton steps.

M = 4915 **M = 35939** **M = 274627**

FIG. 11. ***Average iteration count comparison:*** *for each experiment, we bin according to $\log_{10} N_c$ and compare average BICGSTAB iteration counts; Unpreconditioned BICGSTAB in red, ILU-BICGSTAB in green and TT-BICGSTAB in blue.*

Drafting test with rectangular blade. We follow the validation experiment in [MFJN17], we set up a drafting test involving a rectangular blade of width 0.1 moving through a container filled with spherical rigid particles of radius 0.1 and friction coefficient $\mu = 0.25$. This test may be used to compute the force that the blade experiences as it moves through the granular flow, which eventually reaches a steady state. As in the sedimentation test above, we set up a perturbed lattice of $(2n + 1)^3$ spherical particles inside the box, and perform experiments for $n = 8, 16$ for a total number of $M = 4915, 35939$ rigid bodies. The blade moves from one end of the box to the other with a prescribed sinusoidal velocity with period $4s$.
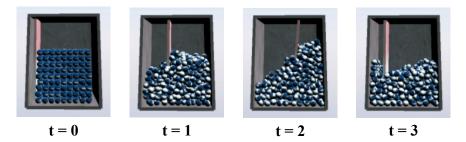
FIG. 12. **Blade drafting test:** *snapshots of simulation for rectangular blade drafting test with* 729 *spheres inside a closed box container. The blade moves on the x direction with prescribed sinusoidal velocity.*

Direct shear experiment. Finally, we include a direct shear test commonly used to measure the shear strength properties of granular soil, used in [MJN16] to validate the DEM CCP approach for frictional contact. In this test, a soil sample is placed inside of a box and subjected to a normal load force (exterted by a cell weighing down on the material). The top half of the shear box is clamped while the lower is displaced in a controlled fashion from left to right, shearing the soil sample.This test can then be utilized to measure the shear stress and other relevant properties as a function of the shear displacement in the box. For our experiments, a perturbed box-shaped lattice with $(3n+1)(2n+1)^2$ spherical particles is set up inside the box, experiencing shear from the lower half of the box and a load from a ceiling press 10 times denser than the granular material. We perform experiments for $n = 8, 16$, for a total number of $M = 7228, 53364$ rigid bodies. The movement of the lower half is again controlled prescribing a sinusoidal velocity.
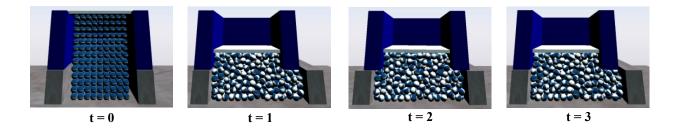

FIG. 13. Direct shear box experiment: *snapshots of simulation for direct shear test with* 1053 *spheres inside a closed box container. The bottom half of the box is displaced in the x direction with a prescribed sinusoidal velocity, and the granular fluid is loaded on the top by a rectangular press.*

In 14, we once again show a comparison of performance for each of the three solvers as it scales with number of collisions N_c. For all four experiments we observe essentially the same scaling for solution and precomputation times for the TT preconditioned solver, with TT precomputation times again staying roughly around 10 seconds per timestep. The overall speedups attained are slightly smaller (about 5-10x), due likely to the increase in problem complexity and the force and velocity magnitudes involved compared to the sedimentation case. However, it remains the case that the TT provides a significantly more robust and better acceleration to the linear system, with better scaling precomputation times and reduced iteration counts than the ILU sparse preconditioner. As is the case for the sedimentation tests, due to the limit of the maximum number of iterations for the unpreconditioned solve, average iteration counts and timings can be estimated to be about 5 times higher than presented in these plots for full accuracy.
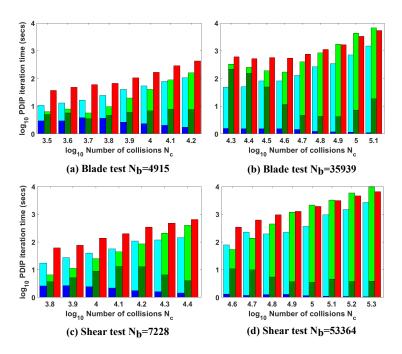
FIG. 14. ***Log-log plot solver comparison for blade drafting (a,b) and direct shear (c,d) experiments:*** *for each experiment, we bin according to $log_{10} N_c$ and plot average PDIP iteration times; Unpreconditioned BICGSTAB in red, ILU-BICGSTAB in green and TT-BICGSTAB in blue. For preconditioned solvers, we display the proportion spent in precomputation in a darker shade, solve times in a lighter one.*

5. Conclusions. In this work we have presented a robust and highly efficient acceleration technique for the solution of Newton step linear systems in second order methods based on approximate hierarchical compression and inversion in the Tensor Train format. In multiple experiments for common terramechanics phenomena modeled with frictional contact for dense, multiple rigid body systems, we have successfully applied a TT preconditioner to accelerate their solution, providing speed-ups of up to an order of magnitude against state-of-the-art sparse solvers for systems with $N_c \gtrsim 20000$. Across all our experiments, we observe that the Tensor Train preconditioner provides lower and more reliable iteration count reductions, as well as practically constant precomputation costs and storage requirements, which are improved significantly by our proposed TT factorization re-use techniques.

As discussed in Section 1, the benefits of rapid, problem-independent convergence of second order optimization solvers are often negated by expensive large sparse matrix solves. The application of sparse and structured linear algebra techniques has been thus far limited by unfavorably scaling precomputation costs, excessive memory storage and communication requirements and the absence of efficient global factorization updates. We have demonstrated that the Tensor Train approach can successfully address these issues in DEM complementarity simulations of granular media. Based on its versatility and exploitation of a wide class of hierarchical low rank structure, we expect this to be true for a wide array of large scale optimization problems. We also anticipate that the low precomputation cost and storage requirements provided by the TT will be most impactful in high performance computing implementations; our ongoing work features a distributed memory implementation of the frictional contact CCP and the TT accelerated PDIP solver.

REFERENCES

[AD13] Sivaram Ambikasaran and Eric Darve. An $o(n \log n)$ fast direct solver for partial hierarchically semi-separable matrices. *Journal of Scientific Computing*, 57(3):477–501, 2013.

[ADLV11] Martin Andersen, Joachim Dahl, Zhang Liu, and Lieven Vandenberghe. Interior-point methods for large-scale cone programming. *Optimization for machine learning*, pages 55–83, 2011.

[Ani06] Mihai Anitescu. Optimization-based simulation of nonsmooth rigid multibody dynamics. *Mathematical Programming*, 105(1):113–143, 2006.

[AP97] Mihai Anitescu and Florian A Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14(3):231–247, 1997.

[BB88] Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988.

[Beb08] M. Bebendorf. *Hierarchical matrices*, volume 63 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin, 2008. A means to efficiently solve elliptic boundary value problems.

[Ben02] Michele Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of computational Physics*, 182(2):418–477, 2002.

[BGH03] S. Börm, L. Grasedyck, and W. Hackbusch. Hierarchical matrices. *Lecture notes*, 21, 2003.

[Bör10] S. Börm. *Efficient numerical methods for non-local operators*, volume 14 of *EMS Tracts in Mathematics*. European Mathematical Society (EMS), Zürich, 2010. $\mathcal{H}^2$-matrix compression, algorithms and analysis.

[CDG+06] S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, and T. Pals. A fast solver for HSS representations via sparse matrices. *SIAM Journal on Matrix Analysis and Applications*, 29(1):67–81, 2006.

[CGP06] S. Chandrasekaran, M. Gu, and T. Pals. A fast ULV decomposition solver for hierarchically semiseparable representations. *SIAM Journal on Matrix Analysis and Applications*, 28(3):603–622, 2006.

[CPD17] Pieter Coulier, Hadi Pouransari, and Eric Darve. The inverse fast multipole method: using a fast approximate direct solver as a preconditioner for dense linear systems. *SIAM Journal on Scientific Computing*, 39(3):A761–A796, 2017.

[CRZ15] E. Corona, A. Rahimian, and D. Zorin. A tensor-train accelerated solver for integral equations in complex geometries. *arXiv preprint arXiv:1511.06029*, 2015.

[CS79] Peter A Cundall and Otto DL Strack. A discrete numerical model for granular assemblies. *Geotechnique*, 29(1):47–65, 1979.

[DS13a] S. Dolgov and D.V. Savostyanov. Alternating minimal energy methods for linear systems in higher dimensions. *Part I: SPD systems, arXiv preprint*, 1301, 2013.

[DS13b] S. Dolgov and D.V. Savostyanov. Alternating minimal energy methods for linear systems in higher dimensions. part II: Faster algorithm and application to nonsymmetric systems. *arXiv preprint arXiv:1304.1222*, 2013.

[Fan15] Luning Fang. *A Primal-Dual Interior Point Method for Solving Multibody Dynamics Problems with Frictional Contact*. PhD thesis, University of Wisconsin–Madison, 2015.

[Gil11] A. Gillman. *Fast direct solvers for elliptic partial differential equations*. PhD thesis, University of Colorado, 2011.

[HATN13] Toby Heyn, Mihai Anitescu, Alessandro Tasora, and Dan Negrut. Using krylov subspace and spectral methods for solving complementarity problems in many-body contact dynamics simulation. *International Journal for Numerical Methods in Engineering*, 95(7):541–561, 2013.

[HY15] Kenneth L Ho and Lexing Ying. Hierarchical interpolative factorization for elliptic operators: differential equations. *Communications on Pure and Applied Mathematics*, 2015.

[KK12] V.A. Kazeev and B. Khoromskij. Low-rank explicit qtt representation of the laplace operator and its inverse. *SIAM Journal on Matrix Analysis and Applications*, 33(3):742–758, 2012.

[Kle15] Jan Kleinert. *Simulating granular material using nonsmooth time-stepping and a matrix-free interior point method*. Fraunhofer Verlag, 2015.

[LSN17] Ang Li, Radu Serban, and Dan Negrut. Analysis of a splitting approach for the parallel solution of linear systems on gpu cards. *SIAM Journal on Scientific Computing*, 39(3):C215–C237, 2017.

[MFJN17] Daniel Melanz, Luning Fang, Paramsothy Jayakumar, and Dan 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.

[MHNT15] Hammad Mazhar, Toby Heyn, Dan Negrut, and Alessandro Tasora. Using nesterov's method to accelerate multibody dynamics with friction and contact. *ACM Transactions on Graphics (TOG)*, 34(3):32, 2015.

[MJN16] Daniel Melanz, Paramsothy Jayakumar, and Dan Negrut. Experimental validation of a differential variational inequality-based approach for handling friction and contact in vehicle/granular-terrain interaction. *Journal of Terramechanics*, 65:1–13, 2016.

[NW06] Jorge Nocedal and Stephen J Wright. *Nonlinear Equations*. Springer, 2006.

[OD12] I.V. Oseledets and S.V. Dolgov. Solution of linear systems and matrix inversion in the TT-format. *SIAM Journal on Scientific Computing*, 34(5):A2718–A2739, 2012.

[Ose10] I.V. Oseledets. Approximation of $2^d \times 2^d$ matrices using tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2130–2145, 2010.

[Ose12] I.V. Oseledets. TT-Toolbox 2.2, 2012.

[OT10] I.V. Oseledets and E. Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.

[OTZ11]   I.V. Oseledets, E. Tyrtyshnikov, and N. Zamarashkin. Tensor-train ranks for matrices and their inverses. *Comput. Methods Appl. Math.*, 11(3):394–403, 2011.

[PCD17]   Hadi Pouransari, Pieter Coulier, and Eric Darve. Fast hierarchical solvers for sparse matrices using extended sparsification and low-rank approximation. *SIAM Journal on Scientific Computing*, 39(3):A797–A830, 2017.

[PGAP09]   Cosmin Petra, Bogdan Gavrea, Mihai Anitescu, and Florian Potra. A computational study of the use of an optimization-based method for simulating large multibody systems? *Optimization Methods & Software*, 24(6):871–894, 2009.

[PKW⁺17]   Arman Pazouki, Michał Kwarta, Kyle Williams, William Likos, Radu Serban, Paramsothy Jayakumar, and Dan Negrut. Compliant contact versus rigid contact: A comparison in the context of granular dynamics. *Physical Review E*, 96(4):042905, 2017.

[Saa03]   Yousef Saad. *Iterative methods for sparse linear systems*, volume 82. siam, 2003.

[ST96]   David E Stewart and Jeffrey C Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering*, 39(15):2673–2691, 1996.

[TA10]   Alessandro Tasora and Mihai Anitescu. A convex complementarity approach for simulating large granular flows. *Journal of Computational and Nonlinear Dynamics*, 5(3):031004, 2010.

[TSM⁺15]   Alessandro Tasora, Radu Serban, Hammad Mazhar, Arman Pazouki, Daniel Melanz, Jonathan Fleischmann, Michael Taylor, Hiroyuki Sugiyama, and Dan Negrut. Chrono: An open source multi-physics dynamics engine. In *International Conference on High Performance Computing in Science and Engineering*, pages 19–49. Springer, 2015.

[XCGL09]   J. Xia, S. Chandrasekaran, M. Gu, and X.S. Li. Superfast multifrontal method for large structured linear systems of equations. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1382–1411, 2009.

[XCGL10]   J. Xia, S. Chandrasekaran, M. Gu, and X.S. Li. Fast algorithms for hierarchically semiseparable matrices. *Numerical Linear Algebra with Applications*, 17(6):953–976, 2010.