Fast Solvers and Simulation Data Compression Algorithms for Granular Media and Complex Fluid Flows

by

Saibal De

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Applied and Interdisciplinary Mathematics and Scientific Computing)
in The University of Michigan
2021

Saibal De

saibalde@umich.edu

ORCID iD: 0000-0003-4691-189X

ACKNOWLEDGEMENTS

I would like to extend my deepest gratitude to my advisors, Professor Shravan Veerapaneni and Professor Xun Huan, for their help and guidance during my studies at the University of Michigan. I am extremely grateful to Professor Silas Alben, Professor Robert Krasny, and Dr. Pramsothy Jayakumar for agreeing to serve on my dissertation committee and for their detailed and valuable feedback. I wish to thank my collaborators, Professor Eduardo Corona, Professor Alex Gorodetsky, Dr. Wen Yan, Dr. Dhairya Malhotra, and Dr. Hadi Salehi; writing this thesis would not have been possible without their help. I am grateful to Professor Hiroyuki Sugiyama for kindly providing me with the simulation datasets for my numerical experiments. Many thanks to my colleagues Dr. Bowei Wu and Dr. Hai Zhu, and my roommate Dr. Mitul Islam for the thought-provoking discussions over the years.

Several funding sources have supported my research, including the National Science Foundation (NSF) under grant DMS-1454010 and the Automotive Research Center (ARC) in accordance with Cooperative Agreement W56HZV-14-2-0001 with U.S. Army Ground Vehicle Systems Center.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Granular and particulate flows are common forms of materials used in various physical and industrial applications. For instance, we model the soil as a collection of rigid particles with frictional contact in soil-vehicle simulations, and we simulate bacterial colonies as active rigid particles immersed in a viscous fluid. Due to the complex interactions in-between the particles and/or the particles and the fluid, numerical simulations are often the only way to study these systems apart from typically expensive physical experiments.

A standard method for simulating these systems is to apply simple physical laws to each of the particles using the discrete element method (DEM) and evolve the resulting multi-body system in time. However, due to the sheer number of particles in even a moderate-scale real-world system, it quickly becomes expensive to timestep these systems unless we exploit fast algorithms and high-performance computing techniques. For instance, a big challenge in granular media simulations is resolving contact between the constituent particles. We use a cone-complementarity formulation of frictional contact to resolve collisions; this approach leads to a quadratic optimization problem whose solution gives us the contact forces between particles at each timestep. In this thesis, we introduce strategies for solving these optimization problems on distributed memory machines. In particular, by imposing a locality-preserving partitioning of the rigid bodies among the computing nodes, we minimize the communication cost and construct a scalable framework for collision detecting and resolution that can be easily scaled to handle hundreds of millions of particles.

For robust and efficient simulation of axisymmetric particles in viscous fluids, we introduce a fast method for solving Stokes boundary integral equations (BIEs) on surfaces of revolution. By first transforming the Stokes integral kernels into a rotationally invariant form and then decoupling the transformed kernels using the Fourier series, we reduce the dimensionality of the problem. This approach improves the time complexity of the BIE solvers by an order of magnitude; additionally we can use high-order one-dimensional singular quadrature schemes to construct highly accurate solvers. Finally, coupling our solver framework with the fast multipole method, we construct a fast solver for simulating Stokes flow past a system of axisymmetric bodies. Combining this with our complementarity collision resolution framework, we have the potential to simulate dense particulate suspensions.

Physics-based simulations similar to those described above generate large amounts of output data, often in the hundreds of gigabytes range. We introduce data compression techniques based on the tensor-train decomposition for DEM simulation outputs and demonstrate the high compressibility of these large datasets. This allows us to keep a reduced representation of simulated data for post-processing or use in learning tasks.

Finally, due to the high cost of physics-based models and limited computational budget, we can typically run only a limited number of simulations when exploring a high-dimensional parameter space. Formally, this can be posed as a matrix/tensor completion problem, and Bayesian inference coupled with a linear factorization model is often used in this setup. We use Markov chain Monte Carlo (MCMC) methods to sample from the unnormalized posteriors in these inference problems. In this thesis, we explore the properties of the posterior in a simple low-rank matrix factorization setup and develop strategies to break its symmetries. This leads to better quality MCMC samples and lowers the reconstruction errors with various synthetic and real-world datasets.

CHAPTER 1

Introduction

In recent decades, we have seen tremendous improvements in computing technologies. With the emergence and widespread adoption of multicore CPUs, accelerators and graphical processing units (GPUs), our computing nodes can provide very high throughputs in terms of the number of floating-point operations per second (FLOP/s). These powerful nodes can then be linked using high-speed interconnects to further increase the computing power. Software development also has kept up the pace with these hardware improvements. The message-passing interface (MPI) has become the de-facto standard for modeling communication between computing nodes. We also have many high-quality open-source numerical libraries for various applications, such as linear algebra [52, 13, 44] and ordinary differential equation integrators [47].

With these advancements, physics-based high-fidelity simulations are becoming an increasingly common tool in modeling many natural and industrial phenomena. These models make minimal assumptions on the dynamics of a system and therefore produce robust and accurate predictions. This increased accuracy, however, comes at the cost of a high computational burden. Developing fast algorithms and designing scalable computing platforms is therefore an active area of current research in many disciplines.

In this thesis, we explore and develop fast algorithms for simulating granular media and complex fluid flows. We also address the issue of the large amounts of output data these high-fidelity simulations can generate, and explore how tensor decomposition techniques can be used to retain a compressed representation of the data for post-processing. Finally, we present a Bayesian framework for robust matrix completion from sparse observations, which can be used as a reduced order model for high-fidelity simulations when we have a limited computational budget to run only a few simulations.

The following is a brief overview of the different chapters in this thesis.

1.1 Scalable Framework for Rigid Body Contact

One of the key ingredients in developing a simulation scheme for dense particulate flows and granular media is rigid body contact resolution. The discrete element method (DEM), where each particle is tracked individually using simple first-principle physical laws such as Newton's equations of motion, has proven to be very successful in making robust and accurate predictions in this setup [28]. This method can also be easily coupled in a straightforward manner with multi-physics simulation models [111].

There are two primary approaches to rigid-body collision resolution in DEM models in the literature [88]:

- In the penalty-based approach (DEM-P), spring-type forces are applied to prevent particles from interpenetrating. However, the resulting system of ordinary differential equations is often very stiff, requiring us to take very small timesteps ($\sim 10^{-5}$ s). However, evaluation of all the body forces in a multi-physics setup at each timestep can quickly become very expensive.

- More recently, a complementarity-based approach (DEM-C) has been proposed that solves an optimization problem to compute the contact forces. This approach does not suffer from stiffness issues, and we can take much larger timesteps ($\sim 10^{-3}$ s).

Various recent software packages have implemented the DEM-C framework for collision resolution between rigid bodies, e.g. Chrono [74, 106]. However, they only implement shared memory parallelism (OpenMP and GPU). Consequently, the systems they can simulate are constrained in terms of the number of particles. In this thesis, we introduce a MPI-based distributed memory framework for DEM-C collision resolution that can overcome this barrier. By ensuring locality during partitioning the particles among MPI processes, we can minimize the communication and develop a demonstrably scalable implementation.

1.2 Viscous Flow Past Axisymmetric Bodies

In a fluid flow at the microscopic level, the viscous and pressure forces far outweigh any inertial and advective forces, reducing the Navier-Stokes equations to a system of linear, constant coefficient system of elliptic partial differential equations: the Stokes equations [90]. Despite the simple description of such flows, they often exhibit unusual behaviors when confined by a complex and/or moving geometry, e.g. in particulate flows consisting of rigid and/or deformable particles. Besides laboratory experiments, numerical simulations are typically the only way of studying the behavior of these systems.

Standard numerical methods, such as the finite element method (FEM), are not suitable for simulating these systems; for instance, a moving geometry would require constant remeshing of the fluid domain. In contrast, in the boundary integral equation (BIE) framework, all the unknowns reside on the boundary of the particles, eliminating the remeshing phase and also reducing the dimensionality of the simulation space by one. Moreover, state-of-the-art fast numerical methods, such as iterative solvers and fast multipole method (FMM), can be used to construct fast and robust solvers.

We can usually exploit any symmetries the particles may exhibit. For instance, with spherical particles the Stokes boundary integral operators diagonalize in vectorized spherical harmonics basis [26], leading to very accurate evaluations of the layer potentials even when the particles are very close. When the particles are axisymmetric, the BIEs for the Laplace, Helmholtz and Maxwell equations can be decoupled using the Fourier series [87, 43, 33], further reducing the computational complexity. However, the Stokes BIE kernels do not satisfy the necessary invariance property for the Fourier decoupling out-of-the-box. In this thesis, we develop a transformation that puts the kernels in the required form, and thereby develop the fast solvers for Stokes BIEs on axisymmetric geometries.

These fast Stokes solvers, coupled with a close evaluation scheme and the complementarity rigid contact resolution can be used as a simulation framework for the motion of axisymmetric particles in viscous fluids. This will provide an extension to the recent work in [111], which simulates spherical rotors in a Stokes suspension.

1.3 Tensor Methods for Data Compression

Large scale physics-based models (e.g. DEM simulation of soil as a granular media with millions of particles) typically generate a huge quantity of simulation output, often ranging in hundreds of gigabytes per simulation. This poses a new challenge for the widespread adoption of high-fidelity models in terms of storing simulation data. In current state-of-the-art applications, output data is typically purged once the quantity of interest is extracted. However throwing away a majority of the data potentially limits the usefulness of high-fidelity simulations since we cannot post-process the data, or train data-driven surrogate models that can in turn guide learning tasks.

The output of high-fidelity simulations can usually be realized as high-dimensional tensors. For instance, a simulation can generate data on a 3D spatial grid for several timesteps and corresponding to multiple simulation parameters, each of which constitutes a new dimension in the data tensor. From this viewpoint tensor decomposition methods, which are designed to mitigate the "curse of dimensionality" in working with high-dimensional tensors, can be used to compress high-fidelity simulation data. In this thesis, we explore using the tensor-train (TT) decomposition [86] to compress scientific data.

1.4 Bayesian Matrix Completion

Low-rank matrix completion is a classical problem with a variety of applications such as recommender system design [104], drug-target interaction prediction [110, 116], image inpainting [42, 63], social network topology recovery [70] and sensor localization [109]. It can also serve as data-driven reduced order models for high-fidelity simulations: when the computational budget is limited, we can infer model outputs and the associated uncertainty at new parameter values from sparse observations in the parametric space.

Optimization-based approaches for this problem is widely studied in the literature [18, 19, 93]. These optimization-based approaches also serve as a base for probabilistic approaches [66, 92, 76, 95], which has also been generalized to tensor completion applications [91, 114, 115]. Bayesian matrix completion problem setups in the literature typically use zero-mean Gaussian priors; consequently, the posteriors we obtain using these priors generally have symmetries. This leads to poor performance of Markov-chain Monte Carlo (MCMC) sampling algorithms. In this thesis, we introduce a simple modification of these priors which can provably break the posterior symmetries, leading to improved sampling performance and reduced reconstruction errors.

CHAPTER 2

Scalable Solvers for Cone Complementarity Problems in Frictional Multibody Dynamics

Preamble. In this chapter, we develop an efficient, hybrid MPI/OpenMP framework for the cone complementarity formulation of large scale rigid body dynamics problems with frictional contact. Using a spatially coherent ordering to partition the rigid bodies among MPI processes, we are able to minimize inter-node communication. Our approach is highly scalable, enabling the solution of dense, large scale multibody problems; a sedimentation simulation involving 256 million particles (~ 324 million contacts on average) was resolved using 512 cores in less than half-hour per time-step. This is joint work with Eduardo Corona, Paramsothy Jayakumar and Shravan Veerapaneni, and is published in [30].

2.1 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) [27, 16, 74]. 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 [102, 10, 9, 11, 105]. For an in-depth analysis and comparison of DEM approaches, see [88]. 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) [102, 10]; 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 [11, 73, 45, 89, 34, 56]. From extensive comparison in multibody dynamics problems [75, 24], 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 [24].

In [79], 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 [80]. 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 [79], 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.

2.2 The Contact Model

2.2.1 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 q and $v \in \mathbb{R}^{6n}$ 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

$$M\dot{v} = f_{\text{ext}}(q, v) + f_{\text{col}}, \quad \dot{q} = Lv. \tag{2.1}$$

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

We model f_{col} using Coulomb's model of friction coupled with complementarity model of contact [11]. 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 $d_{i,n}$, $d_{i,1}$ and $d_{i,2} \in \mathbb{R}^{6n}$ represent these directions in our generalized coordinate system. We can assume $\hat{\gamma}_{i,n} \geq 0$ without loss of generality. Then,

$$f_{\text{col}} = \sum_{i=1}^{m} \hat{\gamma}_{i,n} d_{i,n} + \hat{\gamma}_{i,1} d_{i,1} + \hat{\gamma}_{i,2} d_{i,2} = \sum_{i=1}^{m} D_i \hat{\gamma}_i = D\hat{\gamma} \tag{2.2}$$

is the total force due to all the contacts. Here

$$\hat{\gamma} = (\hat{\gamma}_1, \ldots, \hat{\gamma}_m), \quad \hat{\gamma}_i = (\hat{\gamma}_{i,n}, \hat{\gamma}_{i,1}, \hat{\gamma}_{i,2}) \tag{2.3}$$

is the vector of pairwise contact forces, and

$$D = \begin{bmatrix} D_1 & \cdots & D_m \end{bmatrix}, \quad D_i = \begin{bmatrix} d_{i,n} & d_{i,1} & d_{i,2} \end{bmatrix} \tag{2.4}$$

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,

$$\mathcal{C}_i = \left\{ \hat{\gamma}_i \in \mathbb{R}^3 : \sqrt{\hat{\gamma}_{i,1}^2 + \hat{\gamma}_{i,2}^2} \leq \mu_i \hat{\gamma}_{i,n} \right\}. \tag{2.5}$$

The frictional components satisfy a maximum dissipation condition

$$(\hat{\gamma}_{i,1}, \hat{\gamma}_{i,2}) = \operatorname{argmin}_{\hat{\gamma}_i \in \mathcal{C}_i} (\hat{\gamma}_{i,1} d_{i,1} + \hat{\gamma}_{i,2} d_{i,2})^\top v. \tag{2.6}$$

This maximizes the energy loss by ensuring the local frictional force points opposite to the relative velocity of the particles.

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

$$0 \leq \hat{\gamma}_{i,n} \perp \phi_i(q) \geq 0. \tag{2.7}$$

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

$$M\dot{v} = f_{\text{ext}} + D\hat{\gamma} \tag{2.8a}$$

$$0 \leq \hat{\gamma}_{i,n} \perp \phi_i(q) \geq 0 \tag{2.8b}$$

$$(\hat{\gamma}_{i,1}, \hat{\gamma}_{i,2}) = \text{argmin}_{\hat{\gamma}_i \in \mathcal{C}_i} (\hat{\gamma}_{i,1} d_{i,1} + \hat{\gamma}_{i,2} d_{i,2})^\top v \tag{2.8c}$$

$$\dot{q} = Lv \tag{2.8d}$$

2.2.2 Discretization and Cone Complementarity

We use a semi-implicit Euler time-stepping scheme to discretize the DVI (2.8). Given position q^k and velocity v^k at the k-th time-step with step-size h, we obtain q^{k+1} and v^{k+1} by solving a nonlinear complementarity problem (NCP)

$$v^{k+1} = v^k + M^{-1}(hf_{\text{ext}} + D\gamma) \tag{2.9a}$$

$$0 \leq \gamma_{i,n} \perp \phi_i^k/h + d_{i,n}^\top v^{k+1} \geq 0 \tag{2.9b}$$

$$(\gamma_{i,1}, \gamma_{i,2}) = \text{argmin}_{\gamma_i \in \mathcal{C}_i} (\gamma_{i,1} d_{i,1} + \gamma_{i,2} d_{i,2})^\top v^{k+1} \tag{2.9c}$$

$$q^{k+1} = q^k + hLv^{k+1} \tag{2.9d}$$

In (2.9a), we use $\gamma = h\hat{\gamma}$ as the impulse vector. The right hand side of (2.9b) 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, [11] introduces a relaxation

$$0 \leq \gamma_{i,n} \perp \phi_i^k/h + d_{i,n}^\top v^{k+1} - \mu_i \sqrt{(d_{i,1}^\top v^{k+1})^2 + (d_{i,2}^\top v^{k+1})^2} \geq 0 \tag{2.10}$$

of the complementarity condition (2.9b) that leads to a cone complementarity problem (CCP)

$$\mathcal{C} \ni \gamma \perp g = A\gamma + b \in \mathcal{C}^* \tag{2.11}$$

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

$$A = [A_1; \ldots ; A_m] \qquad b = [b_1; \ldots ; b_m] \tag{2.12}$$

$$A_i = D_i^\top M^{-1} D \qquad b_i = \Phi_i/h + D_i^\top \hat{v} \tag{2.13}$$

$$\hat{v} = v^k + hM^{-1} f_{\text{ext}} \qquad \Phi_i = (\phi_i^k, 0, 0) \tag{2.14}$$

and $\mathcal{C}_i^* = \{g_i \in \mathbb{R}^3 : \gamma_i^\top g_i \geq 0\}$ is the dual cone of $\mathcal{C}_i$. A detailed derivation can be found in [11]. We denote, with $g_i = A_i\gamma + b_i$,

$$\gamma \perp g \iff \gamma_i^\top g_i = 0 \text{ for all } i = 1 : m. \tag{2.15}$$

It can be shown that as time-step $h \to 0$, the solution of the CCP approaches that of the NCP [9, 11].

2.3 Solution of the Complementarity Problem

In [75], the authors compare performance of several solvers for the CCP (2.11). 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.

2.3.1 Accelerated Projected Gradient Descent

It was shown in [11] that (2.11) represents the Karush-Kuhn-Tucker (KKT) optimality conditions for a cone constrained quadratic optimization problem:

$$\text{minimize } f_0(\gamma) = \frac{1}{2}\gamma^\top A\gamma + b^\top\gamma \text{ subject to } \gamma \in \mathcal{C}. \tag{2.16}$$

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 [82]. 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 $y^0 = \gamma^0$, we repeat

$$\gamma^{k+1} = y^k - \alpha^k \nabla f_0(y^k) \tag{2.17}$$

$$\theta^{k+1} = \tfrac{\theta^k\sqrt{(\theta^k)^2+4}-(\theta^k)^2}{2} \tag{2.18}$$

$$\beta^{k+1} = \tfrac{\theta^k(1-\theta^k)}{(\theta^k)^2+\theta^{k+1}} \tag{2.19}$$

$$y^{k+1} = \gamma^{k+1} + \beta^{k+1}(\gamma^{k+1} - \gamma^k) \tag{2.20}$$

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

Lipschitz constant satisfying

$$f_0(y) \leq f_0(y^k) + \nabla f_0(y^k)^\top (y - y^k) + \frac{L}{2}\|y - y^k\|_2^2. \tag{2.21}$$

Then, a choice of $\alpha^k \leq 1/L$ 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: starting from an initial choice of L, it is repeatedly doubled until (2.21) is satisfied for $y = \gamma^{k+1}$.

For constrained convex optimization problems, we can extend this accelerated gradient descent algorithm. We simply project the iterates γ^k onto the feasible set:

$$\gamma^{k+1} = \Pi_{\mathcal{C}}\big(y^k - \alpha^k \nabla f_0(y^k)\big). \tag{2.22}$$

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

2.3.2 Symmetric Cone Interior Point Method

The symmetric cone interior point (SCIP) method solves the CCP (2.11) by utilizing the Jordan algebraic structure on $\mathbb{R}^3$ [56]. We define the Jordan product

$$x_i \circ y_i = \tfrac{1}{\sqrt{2}}(x_i^\top y_i, x_{i,n}y_{i,1} + x_{i,1}y_{i,n}, x_{i,n}y_{i,2} + x_{i,2}y_{i,n}) \tag{2.23}$$

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

$$\mathcal{K}_i = \{x_i \circ x_i : x_i \in \mathbb{R}^3\} = \{x_i : x_{i,n} \geq (x_{i,1}^2 + x_{i,2}^2)^{1/2}\}. \tag{2.24}$$

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

$$T_x = \operatorname{diag}(\ldots, \mu_i, 1, 1, \ldots), T_y = \operatorname{diag}(\ldots, 1, \mu_i, \mu_i, \ldots). \tag{2.25}$$

Denote $\bar{A} = T_y A T_x^{-1}$ and $\bar{b} = T_y b$. Then the CCP (2.11) is equivalent to

$$\mathcal{K} \ni x \perp y = \bar{A}x + \bar{b} \in \mathcal{K} \text{ with } x = T_x\gamma, y = T_y g. \tag{2.26}$$

The corresponding optimization problem is given by

$$\text{minimize } x^\top y \text{ subject to } x, y \in \mathcal{K}. \tag{2.27}$$

As per [56], we define the barrier function for the double cone $\mathcal{K} \cup (-\mathcal{K})$, which gives rise to the potential function

$$f(x, y) = \rho \log (x^\top y) + f_{\text{cen}}(x, y), \tag{2.28}$$

$$f_{\text{cen}}(x, y) = 2m \log \frac{x^\top y/m}{2\prod_{i=1}^m [\det(x_i)\det(y_i)]^{1/2m}}. \tag{2.29}$$

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

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

$$\begin{bmatrix} \nabla_x h(x^k, y^k) & \nabla_y h(x^k, y^k) \\ \bar{A} & -I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \alpha e - x^k \circ y^k \\ \vec{0} \end{bmatrix}. \tag{2.30}$$

To start the iteration, we require a strictly feasible pair (x^0, 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 [56].

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 [81], which rescales the space in which the symmetric cone lies. This leads to a linear system of the form

$$[\bar{A} + P(w)]\Delta x = r \tag{2.31}$$

where $P(w)$ is 3×3 block-diagonal and $r \in \mathbb{R}^{3m}$ is a vector obtained from reducing the Newton system using Schur's complement [56]. The i-th block of P is given by

$$P_i(w_i) = w_i w_i^\top - \det(w_i) J \tag{2.32}$$

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

$$w_i = \frac{y_i^k + \lambda_i J x_i^k}{\sqrt{(x_i^k)^\top y_i^k + 2\sqrt{\det(x_i^k)\det(y_i^k)}}}, \lambda_i = \sqrt{\frac{\det(y_i^k)}{\det(x_i^k)}}. \tag{2.33}$$

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

$$x^{k+1} = x^k + \theta\Delta x, \quad y^{k+1} = y^k + \theta\Delta y. \tag{2.34}$$

2.3.3 Convergence Criteria

We choose our convergence criteria based on the original CCP (2.11) following [56, 75]. Given a primal-dual pair (γ, g) with $g = A\gamma + b$, we compute three residuals

- $r_p \in \mathbb{R}^m$ measures violation of the primal constraint:

$$r_{p,i} = \max\{0, (\gamma_{i,1}^2 + \gamma_{i,2}^2)^{1/2} - \mu_i \gamma_{i,n}\} \tag{2.35}$$

- $r_d \in \mathbb{R}^m$ measures violation of the dual constraint:

$$r_{d,i} = \max\{0, (g_{i,1}^2 + g_{i,2}^2)^{1/2} - g_{i,n}/\mu_i\} \tag{2.36}$$

- $r_c = |\gamma^\top g|/m$ measures violation of the complementarity condition.

We stop the iterative solvers when the residual reaches some prescribed tolerance:

$$\max\left\{\|r_p\|_\infty, \|r_d\|_\infty, r_c\right\} < \tau_{\text{res}} \tag{2.37}$$

2.4 Parallel Implementation

2.4.1 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 [96]. In our implementation, we use a Morton-code based octree partitioning scheme [21]; 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 2.1) 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 2.2 illustrates this partitioning for 40,000 unit spheres, contained in a $100 \times 100 \times 50$ box, among 4 MPI processes.

2.4.2 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.

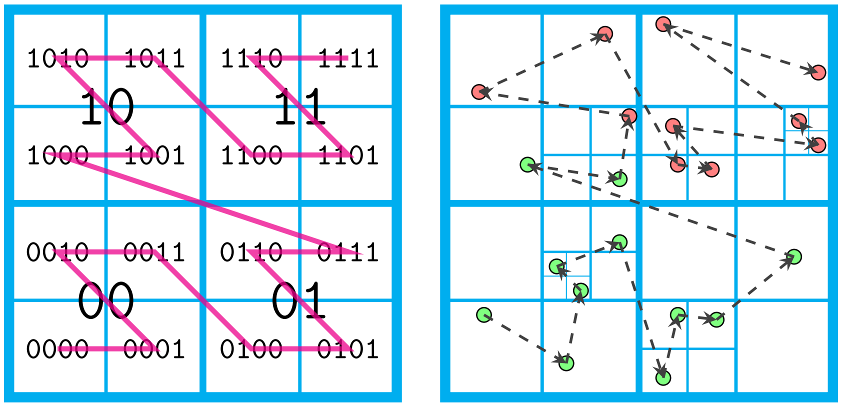


Figure 2.1: Example of Morton ordering in two dimensions. (Left) Construction of Morton IDs by interlacing binary expansion of coordinates, and the Z-order on the second level discretization. (Right) A point cloud with octree bounding boxes and imposed Morton ordering on the points. We show 'equi'-partition of the body list into two parts using different colors.

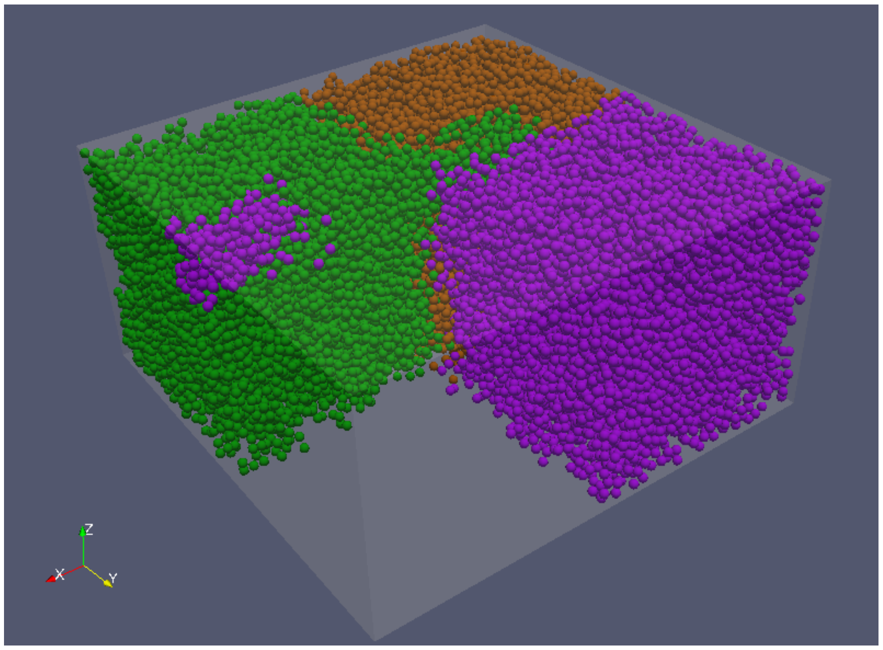


Figure 2.2: Distribution of 40,000 spheres with unit radius in a $100 \times 100 \times 50$ box (volume fraction $\simeq 33.5\%$), among 4 MPI processes, using Morton coding. The spheres belonging to the last rank are suppressed in the figure to emphasize interface structure.

2.4.3 Collision Resolution

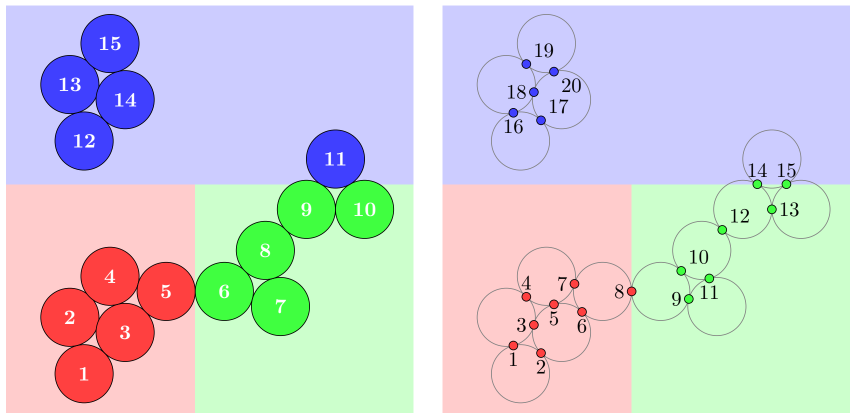
For each collision pair i, we store distance ϕ_i^k, unknown impulse γ_i and contact transformation matrix D_i on the same MPI rank that contains the collision pair. We use a distributed memory linear algebra library to manage the different parts of the distributed vectors Φ and γ, and the distributed matrix D.

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

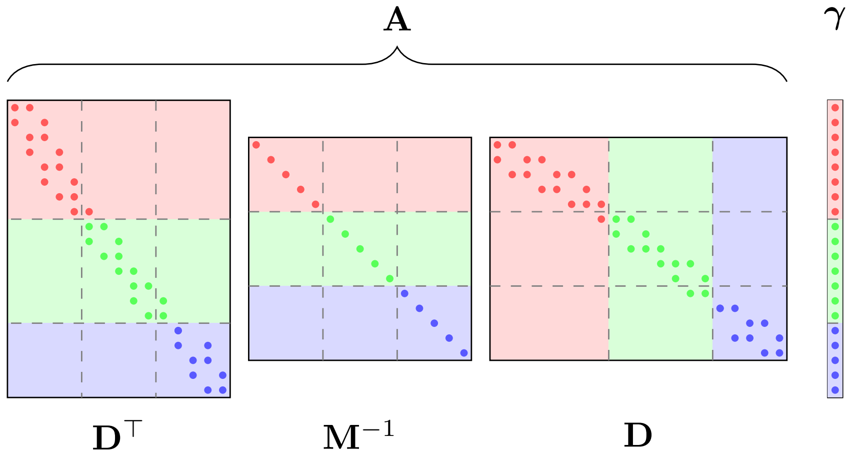
We exploit the sparsity structure of the three factor matrices to create an efficient mat-vec implementation. For granular media simulations, the mass matrix M is 6×6 block diagonal, and the columns of D contain at most 12 non-zero entries (6 per body in the corresponding collision pair). This allows us to store M^{-1} and $D^\top$ 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 γ is partitioned using the same schemes, matrix vector multiplication will be fast (see Figure 2.3).

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

The sparsity structure of A is dependent on contact structure; the 3×3 non-zero blocks correspond to contacts that share a body. In [24], 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.



(a)



(b)

Figure 2.3: Example of distributed construction of the collision matrix. (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 M^{-1} is $6{\times}6$ block diagonal. The columns of the contact transformation matrix D encode information about collision pairs. E.g. the 8th collision occurs between bodies 5 and 6; this corresponds to non-zero 6×3 blocks in the 8th 'column'.

2.5 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.

2.5.1 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 [44], Msgpack [36], Eigen [40] and TRNG [14] 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 [64]. 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).

2.5.2 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.

Table 2.1: Number of iterations required to reach prescribed residual tolerance for APGD and SCIP solvers for two different moderate-scale sedimentation simulations.

	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

2.5.3 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 2.1 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 [75]. However, since each SCIP iteration requires a linear solve, APGD converges faster w.r.t. walltime: each SCIP iteration step with a direct linear solver takes hours compared to the minutes for the APGD iteration steps.

2.5.4 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

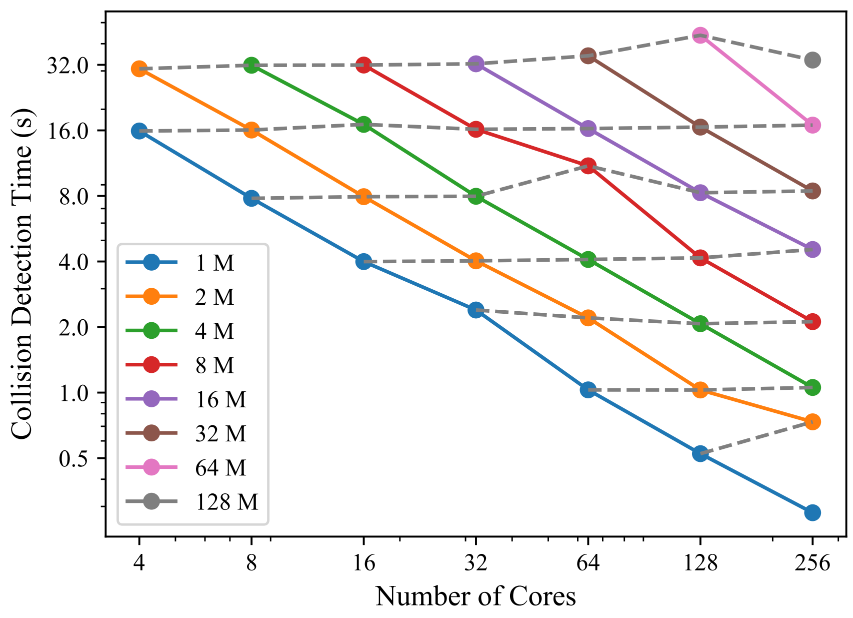


Figure 2.4: Scalability of collision detection algorithm. The solid lines connect simulations with same problem size (strong scaling), and the dashed lines connect simulations with same problem size per core (weak scaling).

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.

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 2.4 and Figure 2.5, 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 2.2). This results in

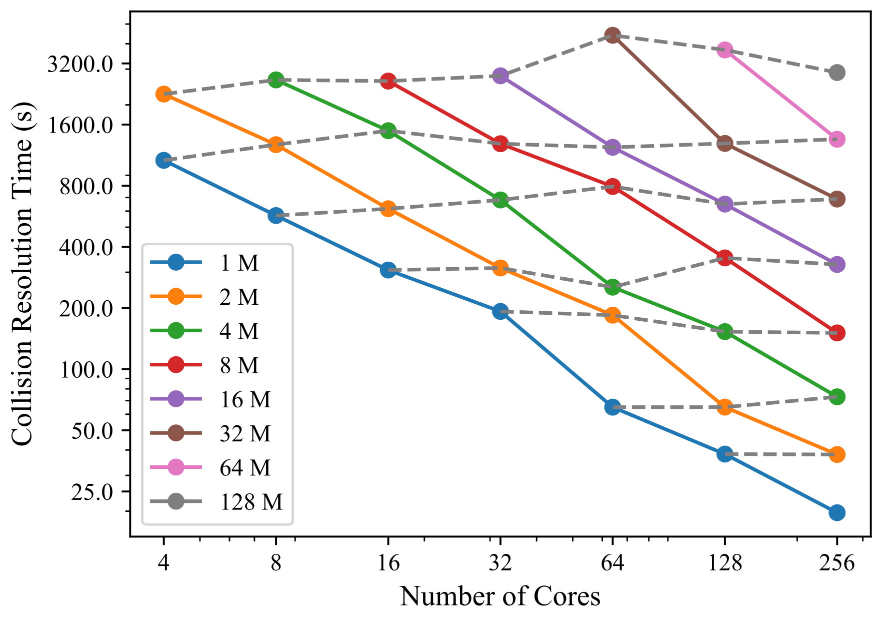


Figure 2.5: Scalability of collision resolution algorithm with APGD optimization solver. The solid lines connect simulations with same problem size (strong scaling), and the dashed lines connect simulations with same problem size per core (weak scaling).

Table 2.2: Average iteration count of APGD in weak scalability tests. As we vary the number of particles and the number of cores while keeping the amount of work per core approximately the same, we see a slight increase in the number of iterations before APGD converges.

# 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

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.

2.6 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.

CHAPTER 3

Fast Solvers for Stokes Boundary Integral Equations in Axisymmetric Geometries via Fourier Decoupling

Preamble. This chapter focuses on developing a fast simulator for three-dimensional Stokes flows past axisymmetric bodies. We transform the Stokes integration kernels into a rotationally invariant form using a linear transformation, and decouple the transformed kernels using the Fourier series. Combining a panel based high-order accurate 1D quadrature scheme with trapezoid rule along the azimuthal direction, we can solve the Stokes boundary integral equations quickly and accurately. This is ongoing work with Shravan Veerapaneni.

3.1 Introduction

Fluid flows laden with microscopic rigid and/or deformable particles are ubiquitous in biological and industrial systems; examples include blood flow [99], bacterial suspensions [32] and microfluidic chips [55]. At these length scales, especially if the flow is very slow or highly viscous, the Reynolds number is typically close to zero and the viscous and pressure forces dominate over any inertial or advective inertial forces, and the full Navier-Stokes flow reduces to the Stokes equations [90]:

$$\nabla^2 u = \nabla p, \quad \nabla \cdot u = 0. \tag{3.1}$$

Here u and p are the velocity and pressure fields of the fluid. This system of equations constitutes a linear and constant-coefficient system of elliptic partial differential equations (PDEs). Despite this relatively simple nature of the governing equations, Stokes flow tends to behave in unusual ways, especially in complex and/or moving geometries, e.g. particulate flows. Apart from laboratory experiments, numerical simulations provide the best way to study the behavior of these systems.

Traditional numerical solvers based on the finite element method (FEM) are not suitable for solving the Stokes PDEs in complex geometries. First, mesh generation for the entire fluid domain with arbitrary particle boundaries is a complex task, one we have to repeat at every timestep when the particles are dynamic. Additionally, when the particles are close by, resolving the interaction accurately might require a very dense mesh. In contrast, the methods based on boundary integral equation (BIE) formulation take advantage of the linear and elliptic structure of the PDEs and only solve for the flow-field generating unknown densities on the particle boundaries. This immediately reduces the number of unknowns by reducing the dimensionality of the problem by one. Additionally, assuming the particle deformations are not extreme, there is no need to change the quadrature schemes associated with the particles throughout the course of the simulation. Further, the linear systems obtained from the BIEs are typically well-conditioned, especially if the integral equations are of the second kind [60]. We can therefore use iterative solvers to efficiently solve these linear systems within a few iterations. Finally, the state-of-the-art fast multipole method (FMM) can be leveraged to construct efficient far-body interaction evaluations [39], leading to high-performance of these iterative solutions schemes even for dense suspensions.

One of the drawbacks of the BIE approach is that the kernels of the integral equations under consideration are typically singular. While it is possible to develop quadrature schemes for integral operators on arbitrary two-dimensional surfaces, the order of decay in error of these schemes are typically low. However, it is possible to leverage symmetries of the particles to simplify the problem. For instance, in [26], the authors demonstrate that the Stokes integral operators on spherical bodies diagonalize in the vectorized spherical harmonics basis. This leads to very accurate evaluation of the layer potentials even when the particles are nearby.

Simulation of Stokes equations in axisymmetric geometries is of great interest in biological systems. For instance, the pressure-driven motion of red blood cells through capillaries is often approximated using rotationally symmetric geometries. However, these models impose a fully axisymmetric assumption on the fluid-flow which is absent in more complex fluid-structure interactions. We aim for a more general scheme which can solve Stokes equations in rotationally symmetric domains, even when the flow itself is not necessarily fully axisymmetric. We use the Fourier series to decouple the surface BIE into a sequence of BIEs on a planar curve; this approach has been developed and tested for the Laplace [87], Helmholtz [87, 43] and Maxwell [33] boundary integral operators. Unfortunately, the Stokes integral kernels do not satisfy an invariance property necessary for the decoupling out-of-the-box. To address this, we developed a transformation which can put the Stokes kernels in the required form. Based on this observation, we design a $O(N^2)$ scheme for solving the Stokes BIEs, where N is the total number of quadrature nodes on the axisymmetric surface. This is a fundamental improvement over the $O(N^3)$ complexity of directly solving the surface BIEs. We also combine our method with FMM to construct a solver for Stokes flow generated by multiple axisymmetric particles with arbitrary axes of symmetry.

The rest of the chapter is structured as follows. In Section 3.2, we review the BIE formulation for the Stokes flow. Section 3.3 outlines the Fourier decoupling process for a general rotationally invariant kernel and the discretization of the resulting integral equations. In Section 3.4, we derive the analytic formulas for evaluation of modal Stokes kernels. Section 3.5 combines our numerical scheme with FMM to simulate multibody systems. In Section 3.6 we report results from our numerical simulations, and in Section 3.7, we present our concluding remarks.

3.2 Boundary Integral Formulation of the Stokes Flow

We present a very brief introduction to the BIE formulation for Stokes equations and set up the boundary value problems; for a detailed exposition of the associated theory, see [61, 90, 51].

3.2.1 Integral Kernels and Operators

Consider a closed, bounded, orientable surface Γ of class C^2. Let $\nu : \Gamma \to \mathbb{R}^3$ denote the outward unit normal field on this surface. Denote the interior and exterior domains of this surface by D^- and D^+. We define the Stokes single layer velocity and pressure potentials as

$$\mathcal{V}_i^S[\rho](x) = \int_\Gamma V_{ij}^S(x,x')\rho_j(x')\,\mathrm{d}\Gamma(x') \tag{3.2}$$

$$\mathcal{P}^S[\rho](x) = \int_\Gamma P_j^S(x,x')\rho_j(x')\,\mathrm{d}\Gamma(x') \tag{3.3}$$

and the Stokes double layer velocity and pressure potentials as

$$\mathcal{V}_i^D[\mu](x) = \int_\Gamma V_{ijk}^D(x,x')\mu_j(x')\nu_k(x')\,\mathrm{d}\Gamma(x') \tag{3.4}$$

$$\mathcal{P}^D[\mu](x) = \int_\Gamma P_{jk}^D(x,x')\mu_j(x')\nu_k(x')\,\mathrm{d}\Gamma(x') \tag{3.5}$$

where $\rho, \mu : \Gamma \to \mathbb{R}^3$ are continuous density functions.

Here (V^S, P^S) and (V^D, P^D) are the free-space Green's functions with different kind of singularities at source point x'. We adopt the explicit expressions for these functions from [90]: the Stokeslet is given by

$$V_{ij}^S(x,x') = \frac{1}{8\pi}\left[\frac{\delta_{ij}}{|\bar{x}|} + \frac{\bar{x}_i\bar{x}_j}{|\bar{x}|^3}\right] \tag{3.6}$$

$$P_j^S(x,x') = \frac{1}{4\pi}\frac{\bar{x}_j}{|\bar{x}|^3} \tag{3.7}$$

and the stresslet is given by

$$V^D_{ijk}(x,x') = \frac{3}{4\pi}\frac{\bar{x}_i\bar{x}_j\bar{x}_k}{|\bar{x}|^5} \tag{3.8}$$

$$P^D_{jk}(x,x') = \frac{1}{2\pi}\left[-\frac{\delta_{jk}}{|\bar{x}|^3} + 3\frac{\bar{x}_j\bar{x}_k}{|\bar{x}|^5}\right] \tag{3.9}$$

Here, for brevity, we denote $\bar{x} = x - x'$.

Now, let us define the neighborhood $\Gamma_c = \{x + t\nu(x) \in \mathbb{R}^3 : x \in \Gamma, -c < t < c\}$ of the surface Γ. For sufficiently small c, we can continuously extend the normal field ν to Γ_c. In this neighborhood, we define the traction potentials associated with the single and double layer formulations as

$$\mathcal{T}^S_i[\rho](x) = \int_\Gamma T^S_{ijl}(x,x')\rho_j(x')\nu_l(x)\,\mathrm{d}\Gamma(x') \tag{3.10}$$

$$\mathcal{T}^D_i[\mu](x) = \int_\Gamma T^D_{ijkl}(x,x')\mu_j(x')\nu_k(x')\nu_l(x)\,\mathrm{d}\Gamma(x') \tag{3.11}$$

The integration kernels are given by

$$T^S_{ijl}(x,x') = -\frac{3}{4\pi}\frac{\bar{x}_i\bar{x}_j\bar{x}_l}{|\bar{x}|^5}, \tag{3.12}$$

$$T^D_{ijkl}(x,x') = \frac{1}{4\pi}\left[\frac{2\delta_{ij}\delta_{kl}}{|\bar{x}|^3} + \frac{3(\delta_{ik}\bar{x}_j\bar{x}_l + \delta_{il}\bar{x}_j\bar{x}_k + \delta_{jk}\bar{x}_i\bar{x}_l + \delta_{jl}\bar{x}_i\bar{x}_k)}{|\bar{x}|^5} - \frac{30\bar{x}_i\bar{x}_j\bar{x}_k\bar{x}_l}{|\bar{x}|^7}\right] \tag{3.13}$$

3.2.2 Jump Conditions at the Boundary

The layer potentials $(\mathcal{V}^S[\rho], \mathcal{P}^S[\rho], \mathcal{T}^S[\rho])$ and $(\mathcal{V}^D[\mu], \mathcal{P}^D[\mu], \mathcal{T}^D[\mu])$ are well defined whenever $x \notin \Gamma$; in fact they are analytic in D^- and D^+. Moreover, since Stokes equations are linear and the layer potentials are convolutions of the fundamental solutions against density functions, they define Stokes flow fields in these domains. This observation allows us to pose certain Stokes boundary value problems as equivalent BIEs defined on the surface.

For this purpose, we need to be able to evaluate the layer potentials when the target x

is on the surface Γ; this involves computing integrals with singular kernels. Additionally, to match boundary data, we need to be able to take limits of the layer potentials as the target approaches the surface from the interior and exterior. To keep our notation clean, we introduce the following scheme: when we write, for example, $\mathcal{V}^S[\rho](x)$ with $x \in \Gamma$, it is understood that the integral is interpreted as an appropriate improper integral. Additionally, we use $\mathcal{V}^S_\pm[\rho](x)$ with $x \in \Gamma$ to denote the one-sided limits

$$\mathcal{V}^S_\pm[\rho](x) = \lim_{h\to 0+} \mathcal{V}^S[\rho](x \pm h\nu(x)) \tag{3.14}$$

It can be shown that the single layer velocity potential $\mathcal{V}^S[\rho]$, single layer traction potential $\mathcal{T}^S[\rho]$ and double layer velocity potential $\mathcal{V}^D[\mu]$ have weakly-singular kernels, and the integrals exist in an improper sense. Additionally, they satisfy the jump conditions [90]

$$\mathcal{V}^S_\pm[\rho](x) = \mathcal{V}^S[\rho](x) \tag{3.15}$$

$$\mathcal{T}^S_\pm[\rho](x) = \mp\frac{1}{2}\rho(x) + \mathcal{T}^S[\rho](x) \tag{3.16}$$

$$\mathcal{V}^D_\pm[\mu](x) = \pm\frac{1}{2}\mu(x) + \mathcal{V}^D[\mu](x) \tag{3.17}$$

for $x \in \Gamma$. The cases of the other three layer potentials are not as straightforward. The single layer pressure potential $\mathcal{P}^S[\rho]$ is singular, whereas the double layer pressure potential $\mathcal{P}^D[\mu]$ and the double layer traction potential $\mathcal{T}^D[\mu]$ are hyper-singular; they need to be interpreted as Cauchy principal value integral and Hadamard finite part integrals, respectively. The jump conditions for these layer potentials are given by [90, 112]

$$\mathcal{P}^S_\pm[\rho](x) = \pm\frac{1}{2}\nu(x)^\top\rho(x) + \mathcal{P}^S[\rho](x) \tag{3.18}$$

$$\mathcal{P}^D_\pm[\mu](x) = \pm(\tau_1(x)^\top\mu_{\tau_1}(x) + \tau_2(x)^\top\mu_{\tau_2}(x)) + \mathcal{P}^D[\mu](x) \tag{3.19}$$

$$\mathcal{T}^D_\pm[\mu](x) = \mathcal{T}^D[\mu](x) \tag{3.20}$$

Here $\tau_1(x)$ and $\tau_2(x)$ are orthonormal unit vectors spanning the tangent plane of Γ at $x \in \Gamma$, and μ_{τ_1}, μ_{τ_2} are derivatives of μ along these directions.

3.2.3 Boundary Integral Equations

The jump conditions can be used to reformulate Dirichlet boundary value problems as equivalent boundary integral equations. We present two examples:

Problem 3.1. Consider a Stokes flow in the exterior D^+ with boundary data

$$u = f \quad \text{on} \quad \Gamma \tag{3.21}$$

In the single layer formulation, we assume that the solution flow is generated by some unknown Stokeslet density ρ on the boundary: $u = \mathcal{V}^S[\rho]$. Using the jump condition (3.15) of the single layer potential, we can rewrite the boundary condition as

$$\mathcal{V}^S[\rho] = f \quad \text{on} \quad \Gamma \tag{3.22}$$

This is a Fredholm integral equation of the first kind defined on the two dimensional surface Γ. Once we solve for the unknown density ρ, we can use it to generate the flow field in the exterior D^+

$$u = \mathcal{V}^S[\rho] \quad \text{in} \quad D^+ \tag{3.23}$$

$$p = \mathcal{P}^S[\rho] \quad \text{in} \quad D^+ \tag{3.24}$$

or compute the pressure and traction on the fluid at the surface

$$p^+ = \frac{1}{2}\nu^\top \rho + \mathcal{P}^S[\rho] \quad \text{on} \quad \Gamma \tag{3.25}$$

$$t^+ = -\frac{1}{2}\rho + \mathcal{T}^S[\rho] \quad \text{on} \quad \Gamma \tag{3.26}$$

Problem 3.2. Now consider a Stokes flow in the interior D^- of the surface Γ with the same boundary data (3.21). Using the double layer formulation, we assume that the flow is generated by some unknown stresslet density $u = \mathcal{V}^D[\mu]$. Using the appropriate jump condition (3.17) across the boundary Γ, we can rewrite the boundary condition as a Fredholm integral equation of the second kind:

$$-\frac{1}{2}\mu + \mathcal{V}^D[\mu] = f \quad \text{on} \quad \Gamma \tag{3.27}$$

We prefer this formulation when possible, since the linear systems, constructed from discretizing Fredholm integral equations of the second kind, are usually well-conditioned. Once we have numerical approximation of the density μ, we can compute the velocity and pressure fields in the interior as

$$u = \mathcal{V}^D[\mu] \quad \text{in} \quad D^- \tag{3.28}$$

$$p = \mathcal{P}^D[\mu] \quad \text{in} \quad D^- \tag{3.29}$$

using a suitable quadrature rule.

3.3 Integral Equations on Axisymmetric Surfaces

Development of high-order accurate solvers for BIEs with singular kernels is an active area of research. In one dimension, many excellent quadrature schemes have been designed to handle singular integrands, such as Kapur-Rokhlin [54], Alpert [6] and Kolm-Rokhlin [59]. A review of these quadratures, as well as the comparison of their performance, is available in [41]. By comparison, singular quadratures for singular integrals on a two-dimensional surface is not as well developed. Quadrature by expansion (QBX) is a popular method [57, 2, 3, 4, 97], but schemes based on this approach usually achieve a low order of convergence.

However, when the surface is rotationally symmetric, it is possible to take advantage

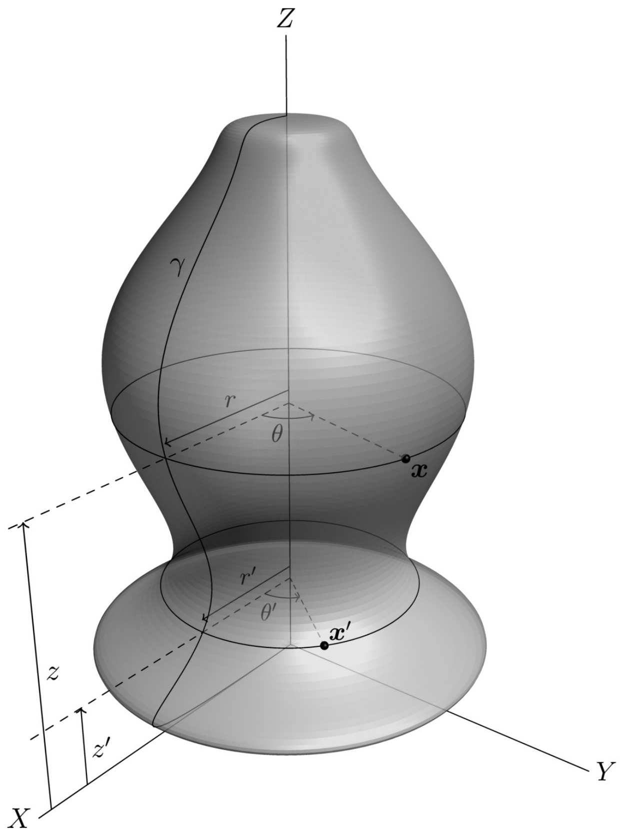


Figure 3.1: Parametrization of a axisymmetric surface Γ, which is generated by rotating a curve $\gamma : s \mapsto (r, z)$ about the z-axis; the curve γ is called the generating curve for the surface. The source and target points, x and x', are expressed in the cylindrical coordinate system, with $x = x(r, z, \theta)$ and $x' = x(r', z', \theta')$.

the symmetry and reduce the surface BIE into a sequence of curve BIEs. We can then use the high-order quadrature schemes designed for singular one-dimensional integrals. This approach has been utilized to device fast and accurate solvers for the Laplacian [87], Helmholtz [87, 43] and Maxwell [33] equations. Here, we give a brief overview of this approach.

3.3.1 Fourier Mode Decomposition

Consider a scalar Fredholm BIE of the second kind (the case of vector valued BIEs is analogous)

$$\tau + \mathcal{A}[\tau] = f \quad \text{with} \quad \mathcal{A}[\tau](x) := \int_\Gamma A(x, x')\tau(x')\,\mathrm{d}\Gamma(x') \tag{3.30}$$

defined on a smooth surface of revolution Γ, i.e. it is obtained by rotating a planar curve $\gamma : [0,1] \to \mathbb{R}_+ \times \mathbb{R}$ about the z-axis (see Figure 3.1). In this situation, it is natural to choose cylindrical coordinates (r, z, θ) and (r', z', θ') to parametrize the points x and x', respectively. In Cartesian coordinates, the components are then expressed by

$$x = \begin{bmatrix} r\cos\theta \\ r\sin\theta \\ z \end{bmatrix} \quad \text{and} \quad x' = \begin{bmatrix} r'\cos\theta' \\ r'\sin\theta' \\ z' \end{bmatrix} \tag{3.31}$$

It is easy to see that in this new coordinate system, the boundary density τ and boundary data f become 2π-periodic functions of the azimuthal angle, and can be expressed in a Fourier series:

$$\tau(x) = \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{+\infty} \tau_n(s) e^{in\theta} \quad \text{and} \quad f(x) = \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{+\infty} f_n(s) e^{in\theta} \tag{3.32}$$

where $s \mapsto (r, z)$ parametrizes the generating curve γ. Let us further assume that the kernel is rotationally invariant, i.e. it only depends on the difference of the azimuthal angles of the source and target points:

$$A(x, x') = A(\theta - \theta', s, s') \tag{3.33}$$

Under this assumption, we can also write down a Fourier series for the kernel

$$A(x, x') = \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{+\infty} A_n(s, s') e^{in(\theta - \theta')} \tag{3.34}$$

Then the BIE (3.30) is equivalent to a series of integral equations defined on the generating curve [87]:

$$\tau_n + \mathcal{A}_n[\tau_n] = f_n \quad \text{where} \quad \mathcal{A}_n[\tau_n](s) = \sqrt{2\pi} \int_\gamma A_n(s, s') \tau_n(s') r(s') \, \mathrm{d}\gamma(s') \tag{3.35}$$

and the solution to (3.30) can be constructed as

$$\tau(x) = \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} (\mathcal{I} + \mathcal{A}_n)^{-1}[f_n](s) e^{in\theta} \tag{3.36}$$

In practice, we truncate the Fourier series at a cutoff frequency n_f. Thus, we choose to solve the problem with an approximate boundary data

$$f_{\text{approx}}(x) = \frac{1}{\sqrt{2\pi}} \sum_{n=-n_f}^{n_f} f_n(s) e^{in\theta} \tag{3.37}$$

Our approximate solution is then given by

$$\tau_{\text{approx}}(x) = \frac{1}{\sqrt{2\pi}} \sum_{n=-n_f}^{n_f} (\mathcal{I} + \mathcal{A}_n)^{-1}[f_n](s) e^{in\theta} \tag{3.38}$$

In [87], the authors showed that under certain assumptions on the integral operator $\mathcal{A}$, the error in the boundary density is of the same order as the truncation error in the boundary data as long as n_f is large enough.

3.3.2 Singular Nyström Quadrature Schemes on Smooth Curves

We follow [41] in constructing a high-order modified Gauss-Legendre quadrature scheme for singular integrals of the form (3.35) on the generating curve. Note that by parametrizing the curve $\gamma : [0, 1] \to \mathbb{R}_+ \times \mathbb{R}$, we can rewrite the integral over the curve as

$$\begin{aligned} \mathcal{A}_n[\tau_n](s) &= \sqrt{2\pi} \int_\gamma A_n(s, s') \tau_n(s') r(s') \mathrm{d}\gamma(s') \\ &= \int_0^1 \left[\sqrt{2\pi} A_n(s, s') r(s') |\dot{\gamma}(s')| \right] \tau_n(s') \, \mathrm{d}s' \end{aligned} \tag{3.39}$$

where $\dot{\gamma}(s)$ is derivative of $\gamma(s)$ w.r.t. the curve parameter s. To simplify notation, we combine the terms inside the parenthesis into a single function and drop the Fourier mode subscripts:

$$g(s) = \int_0^1 B(s, s')\tau(s')\,\mathrm{d}s' \tag{3.40}$$

Now, we partition $[0, 1]$ into n_p panels I_k, and implement 10th-order Gauss-Legendre quadrature scheme for each of these panels; this leads to the following discretization

$$\begin{aligned} g(s_i^{(k)}) &= \sum_{l=1}^{n_p} \int_{I_l} B(s_i^{(k)}, s)\tau(s)\,\mathrm{d}s \\ &\approx \sum_{l=1}^{n_p} \sum_{j=1}^{10} B(s_i^{(k)}, s_j^{(l)}) w_j^{(l)} \tau(s_j^{(l)}) \end{aligned} \tag{3.41}$$

where $\{(s_i^{(k)}, w_i^{(k)}) : 1 \leq i \leq 10\}$ are the standard Gauss-Legendre quadrature nodes and weights for the panel I_k with $1 \leq k \leq n_p$.

While this simple quadrature rule works very well for continuous kernels, we cannot evaluate the kernel when the source and target nodes are in nearby panels ($k \approx l$). In this situation, specially constructed Kolm-Rokhlin auxiliary nodes and weights $\{(s_j^{(l)(k,i)}, w_j^{(l)(k,i)}) : 1 \leq j \leq m\}$ that depend on the target panel k and node i are used to evaluate the integral over the panel [59]

$$\int_{I_l} B(s_i^{(k)}, s)\tau(s)\,\mathrm{d}s \approx \sum_{j=1}^{m} B(s_i^{(k)}, s_j^{(l)(k,i)}) w_j^{(l)(k,i)} \tau(s_j^{(l)(k,i)}) \tag{3.42}$$

The function τ is computed at these auxiliary nodes using Lagrange interpolation. In our setup, a value of $m = 20$ is used when I_k and I_l are the same panel, and $m = 24$ when I_k and I_l are neighboring panels. In all other cases, the standard quadrature is used.

3.3.3 Overview of the Algorithm

We are now in a position combine all the pieces described in this section, and construct the full algorithm. Suppose that we want to solve the BIE (3.30)

$$\tau + \mathcal{A}[\tau] = f \tag{3.43}$$

on the axisymmetric surface Γ. Our input is boundary data f, number of panels n_p and cutoff frequency n_f. We solve the Fourier decoupled BIEs (3.35) on the generating curve γ

$$\tau_n + \mathcal{A}_n[\tau_n] = f_n, \quad -n_f \leq n \leq n_f \tag{3.44}$$

and construct the boundary density τ using the solved Fourier modes τ_n. The general structure of our algorithm is as follows:

1. Using a pre-computed Kolm-Rokhlin quadrature rule, create a discretization of the generating curve γ using n_p panels. This creates a total $10n_p$ standard Gaussian nodes along the generating curve, and discretizes the modal integral equations as

   $$\mathsf{\tau}_n + \mathsf{A}_n \mathsf{\tau}_n = \mathsf{f}_n, \quad -n_f \leq n \leq n_f \tag{3.45}$$

2. Compute the kernel matrices A_n.

3. Compute the Fourier modes f_n for the boundary value function. This can be achieved with $O(n_p)$ fast Fourier transforms (FFTs) of size $O(n_f)$ in $O(n_p n_f \log n_f)$ operations.

4. Solve $O(n_f)$ linear systems of size $O(n_p)$. Using direct methods, this take $O(n_p^3 n_f)$ time. However, if we need to solve for boundary density for the same surface, but with a different boundary condition, we can store the inverses, and during the subsequent solves, the cost would come down to $O(n_p^2 n_f)$.

5. Once we have the Fourier modes τ_n of the boundary density, we can use inverse fast Fourier transform (IFFT) to compute the boundary density using $O(n_p n_f \log n_f)$ operations.

This approach has one major drawback. Computation of the kernel matrices A_n require us to evaluate the kernel functions A_n. However, unlike the full kernel A, we often do not have access to analytical formulas for these Fourier modes. We can use FFT to compute approximations to these kernels. However, for the diagonal and near diagonal entries, the singularity in the kernel A makes the functions $t \mapsto A(t, s, s')$ sharply peaked around $t = 0$, and FFT is inaccurate in this regime.

3.4 Rotational Invariance and Evaluation of the Stokes Modal Kernels

For successful design of fast BIE solvers with rotationally invariant kernels on axisymmetric geometries, we need some way to compute the near diagonal entries. In [87], the authors presented analytical expressions for Fourier modes of Laplace and Helmholtz kernels. In [33], analytical expansions were constructed for Maxwell equations. In this section, we follow a similar approach and construct analytical expressions for the Stokes kernels.

3.4.1 Rotationally Invariant Forms

None of the kernels in Stokes boundary integral operators (3.2), (3.3), (3.4), (3.5), (3.10) and (3.11) satisfy the rotational invariance property (3.33). This is because of how the $\bar{x} = x - x'$ terms interact with each other; for instance, in the single layer velocity potential (3.2), the $\bar{x}_i \bar{x}_j$ terms are not individually rotationally invariant.

Proposition 3.3. *Let $U(\phi)$ be the linear transformation that rotates any vector in three-*

dimensional space by angle ϕ in the clockwise direction about the z-axis,

$$U(\phi) = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.46}$$

Let x and x' be two points in the 3D Euclidean space with azimuthal angles θ and θ' respectively, and denote $\bar{x} = x - x'$. Then the entries $U_{ii'}(\theta)\bar{x}_{i'}$ and $U_{jj'}(\theta')\bar{x}_{j'}$ of the vectors $U(\theta)\bar{x}$ and $U(\theta')\bar{x}$ are rotationally invariant. Consequently the entries $U_{ii'}(\theta)\bar{x}_{i'}\bar{x}_{j'}U_{jj'}(\theta')$ of the transformed matrix $U(\theta)\bar{x}\bar{x}^\top U(\theta')^\top$ are rotationally invariant.

Proof. Note that $x \mapsto U(\phi)x$ reduces the azimuthal angle of vector x by ϕ. Since the azimuthal angle of x is given by θ, it is clear that $U(\theta)x$ has zero azimuthal angle. Similarly $U(\theta)x'$ has azimuthal angle $\theta' - \theta$. Thus

$$U(\theta)\bar{x} = U(\theta)x - U(\theta)x' = \begin{bmatrix} r \\ 0 \\ z \end{bmatrix} - \begin{bmatrix} r'\cos(\theta' - \theta) \\ r'\sin(\theta' - \theta) \\ z' \end{bmatrix} = \begin{bmatrix} r - r'\cos t \\ r'\sin t \\ z - z' \end{bmatrix} \tag{3.47}$$

where we have defined $t := \theta - \theta'$. We can write down a similar expression for $U(\theta')\bar{x}$. It follows that the entries of

$$U(\theta)\bar{x}\bar{x}^\top U(\theta')^\top = [U(\theta)\bar{x}][U(\theta')\bar{x}]^\top = \begin{bmatrix} r - r'\cos t \\ r'\sin t \\ z - z' \end{bmatrix} \begin{bmatrix} r\cos t - r' \\ r\sin t \\ z - z' \end{bmatrix}^\top \tag{3.48}$$

are rotationally invariant. □

Proposition 3.4. *Let U be defined as in Proposition 3.3. Let Γ be a C^2 surface created by rotating a curve $\gamma : s \mapsto (r, z)$ about the z-axis. Then the entries $U_{ii'}(\theta)\nu_{i'}(x)$, $U_{ii'}(\theta)\nu_{i'}(x')$, $U_{ii'}(\theta')\nu_{i'}(x)$ and $U_{ii'}(\theta')\nu_{i'}(x')$ of the vectors $U(\theta)\nu(x)$, $U(\theta)\nu(x')$, $U(\theta')\nu(x)$ and $U(\theta')\nu(x')$*

are rotationally invariant. Consequently the terms $\nu_i(x)\bar{x}_i = \nu(x)^\top \bar{x}$ *and* $\nu_i(x')\bar{x}_i = \nu(x')^\top \bar{x}$ *are also rotationally invariant.*

Proof. On axisymmetric surfaces, the normal direction $\nu(x)$ has the same azimuthal angle as the point x. This can easily be seen by writing out the explicit form of the normal:

$$\nu(x) = \frac{1}{\sqrt{\dot{r}^2 + \dot{z}^2}} \begin{bmatrix} -\dot{z}\cos\theta \\ -\dot{z}\sin\theta \\ \dot{r} \end{bmatrix} \tag{3.49}$$

where the dot represents derivative of the r and z functions defining the generating curve γ w.r.t. the parameter s. The invariance of $U(\theta)\nu(x)$, $U(\theta)\nu(x')$, $U(\theta')\nu(x)$ and $U(\theta')\nu(x')$ is immediate following a similar argument as in Proposition 3.3; for example

$$U(\theta)\nu(x) = \frac{1}{\sqrt{\dot{r}^2 + \dot{z}^2}} \begin{bmatrix} -\dot{z} \\ 0 \\ \dot{r} \end{bmatrix} \quad \text{and} \quad U(\theta)\nu(x') = \frac{1}{\sqrt{\dot{r}'^2 + \dot{z}'^2}} \begin{bmatrix} -\dot{z}'\cos(\theta' - \theta) \\ -\dot{z}'\sin(\theta' - \theta) \\ \dot{r}' \end{bmatrix} \tag{3.50}$$

where we abuse the notation a little to write $\dot{r}' = \dot{r}(s')$ and $\dot{z}' = \dot{z}(s')$. In addition, we have

$$\nu(x)^\top \bar{x} = [U(\theta)\nu(x)]^\top [U(\theta)\bar{x}] \tag{3.51}$$

$$\nu(x')^\top \bar{x} = [U(\theta')\nu(x')]^\top [U(\theta')\bar{x}] \tag{3.52}$$

where each of the terms in the parenthesis are rotationally invariant. □

Finally, the $|\bar{x}| = \sqrt{r^2 + r'^2 - 2rr'\cos t + (z - z')^2}$ term is already in rotationally invariant form, with $t = \theta - \theta'$. With these three observations, we can now prove that all of the Stokes BIEs can be put in rotationally invariant forms:

Corollary 3.5. *Let* $\widetilde{\rho}_j(x') = U_{jj'}(\theta')\rho_{j'}(x')$. *Then the single layer Stokes operators* (3.2), (3.3) *and* (3.10) *are equivalent to*

$$U_{ii'}(\theta)\mathcal{V}^S_{i'}[\rho](x) =: \widetilde{\mathcal{V}}^S_i[\widetilde{\rho}](x) = \int_\Gamma \widetilde{V}^S_{ij}(x,x')\widetilde{\rho}_j(x')\mathrm{d}\Gamma(x') \tag{3.53}$$

$$\mathcal{P}^S[\rho](x) =: \widetilde{\mathcal{P}}^S[\widetilde{\rho}](x) = \int_\Gamma \widetilde{P}^S_j(x,x')\widetilde{\rho}_j(x')\mathrm{d}\Gamma(x') \tag{3.54}$$

$$U_{ii'}(\theta)\mathcal{T}^S_{i'}[\rho](x) =: \widetilde{\mathcal{T}}^S_i[\widetilde{\rho}](x) = \int_\Gamma \widetilde{T}^S_{ij}(x,x')\widetilde{\rho}_j(x')\mathrm{d}\Gamma(x') \tag{3.55}$$

with rotationally invariant integral kernels

$$\widetilde{V}^S_{ij}(x,x') = U_{ii'}(\theta)V^S_{i'j'}(x,x')U_{jj'}(\theta') \tag{3.56}$$

$$\widetilde{P}^S_j(x,x') = P^S_{j'}(x,x')U_{jj'}(\theta') \tag{3.57}$$

$$\widetilde{T}^S_{ij}(x,x') = U_{ii'}(\theta)T^S_{i'j'l}(x,x')\nu_l(x)U_{jj'}(\theta') \tag{3.58}$$

Proof. Proving the equivalence of the layer operators boils down to multiplying the velocity and traction operators on the left by $U(\theta)$, substituting the expression for $\widetilde{\rho}$ and rearranging the terms.

To show the rotational invariance of the transformed velocity kernel, we note that

$$\begin{aligned}\widetilde{V}^S_{ij}(x,x') &= U_{ii'}(\theta)V^S_{i'j'}(x,x')U_{jj'}(\theta') \\ &= \frac{1}{8\pi}U_{ii'}(\theta)\left[\frac{\delta_{i'j'}}{|\bar{x}|} + \frac{\bar{x}_{i'}\bar{x}_{j'}}{|\bar{x}|^3}\right]U_{jj'}(\theta') \\ &= \frac{1}{8\pi|\bar{x}|}U_{ij}(\theta-\theta') + \frac{1}{8\pi|\bar{x}|^3}[U_{ii'}(\theta)\bar{x}_{i'}][U_{jj'}(\theta')\bar{x}_{j'}]\end{aligned} \tag{3.59}$$

where the simplification of the first term,

$$U_{ii'}(\theta)\delta_{i'j'}U_{j'j}(\theta') = U_{ij}(\theta-\theta') \iff U(\theta)IU^\top(\theta') = U(\theta-\theta') \tag{3.60}$$

follows from the following observation. The operations in $U(\theta)IU^\top(\theta')$, from the right to the left, are: (i) rotation around z-axis by angle θ', (ii) identity operator and (iii) rotation around z-axis by angle $-\theta$. This is equivalent to a total rotation around z-axis by angle $-(\theta-\theta')$, which is encoded by $U(\theta-\theta')$. With this and Proposition 3.3, all the terms in $\widetilde{V}^S_{ij}$ are rotationally invariant.

Next, for the transformed pressure kernel, we note that

$$\widetilde{P}^S_j(x,x') = P^S_{j'}(x,x')U_{jj'}(\theta') = \frac{1}{4\pi|\bar{x}|^3}[U_{jj'}(\theta')\bar{x}_{j'}] \tag{3.61}$$

which is again rotationally invariant.

Finally, for the transformed traction kernel, we have

$$\begin{aligned}\widetilde{T}^S_{ij}(x,x') &= U_{ii'}(\theta)T^S_{i'j'l}(x,x')\nu_l(x)U_{jj'}(\theta') \\ &= -\frac{3}{4\pi|\bar{x}|^3}[U_{ii'}(\theta)\bar{x}_{i'}][U_{jj'}(\theta')\bar{x}_{j'}][\nu_l(x)\bar{x}_l]\end{aligned} \tag{3.62}$$

and the invariance follows by a combination of Propositions 3.3 and 3.4. □

Corollary 3.6. *Let $\widetilde{\mu}_j(x') = U_{jj'}(\theta')\mu_{j'}(x')$. Then the double layer Stokes operators* (3.4), (3.5) *and* (3.11) *are equivalent to*

$$U_{ii'}(\theta)\mathcal{V}^D_{i'}[\mu](x) =: \widetilde{\mathcal{V}}^D_i[\widetilde{\mu}](x) = \int_\Gamma \widetilde{V}^D_{ij}(x,x')\widetilde{\mu}_j(x')\mathrm{d}\Gamma(x') \tag{3.63}$$

$$\mathcal{P}^D[\mu](x) =: \widetilde{\mathcal{P}}^D[\widetilde{\mu}](x) = \int_\Gamma \widetilde{P}^D_j(x,x')\widetilde{\mu}_j(x')\mathrm{d}\Gamma(x') \tag{3.64}$$

$$U_{ii'}(\theta)\mathcal{T}^D_{i'}[\mu](x) =: \widetilde{\mathcal{T}}^D_i[\widetilde{\mu}](x) = \int_\Gamma \widetilde{T}^D_{ij}(x,x')\widetilde{\mu}_j(x')\mathrm{d}\Gamma(x') \tag{3.65}$$

with rotationally invariant integration kernels

$$\widetilde{V}^D_{ij}(x,x') = U_{ii'}(\theta)V^D_{i'j'k}(x,x')\nu_k(x')U_{jj'}(\theta') \tag{3.66}$$

$$\widetilde{P}^D_j(x,x') = P^D_{j'k}(x,x')\nu_k(x')U_{jj'}(\theta') \tag{3.67}$$

$$\widetilde{T}^D_{ij}(x,x') = U_{ii'}(\theta)T^D_{i'j'kl}(x,x')\nu_k(x')\nu_l(x)U_{jj'}(\theta') \tag{3.68}$$

Proof. Proving the equivalence of the layer operators, as in the single layer case, boils down to multiplying the velocity and traction operators on the left by $U(\theta)$, substituting the expression for $\widetilde{\rho}$ and rearranging the terms.

To show the rotational invariance of the transformed velocity kernel, we note that

$$\begin{aligned} \widetilde{V}^D_{ij}(x,x') &= U_{ii'}(\theta)V^D_{i'j'k}(x,x')\nu_k(x')U_{jj'}(\theta') \\ &= \frac{3}{4\pi|\bar{x}|^5}[U_{ii'}(\theta)\bar{x}_{i'}][U_{jj'}(\theta')\bar{x}_{j'}][\nu_k(x')\bar{x}_k] \end{aligned} \tag{3.69}$$

is a product of rotationally invariant terms by Propositions 3.3 and 3.4.

Next, for the rotational invariance of the transformed pressure kernel, we note that

$$\begin{aligned} \widetilde{P}^D_{j}(x,x') &= P^D_{j'k}(x,x')\nu_k(x')U_{jj'}(\theta') \\ &= -\frac{1}{2\pi|\bar{x}|^3}\delta_{j'k}\nu_k(x')U_{jj'}(\theta') + \frac{3}{2\pi|\bar{x}|^5}\bar{x}_{j'}\bar{x}_k\nu_k(x')U_{jj'}(\theta') \\ &= -\frac{1}{2\pi|\bar{x}|^3}[U_{jj'}(\theta')\nu_{j'}(x')] + \frac{3}{2\pi|\bar{x}|^5}[U_{jj'}(\theta')\bar{x}_{j'}][\nu_k(x')\bar{x}_k] \end{aligned} \tag{3.70}$$

which is again rotationally invariant.

Finally, for the transformed traction kernel, we have

$$\widetilde{T}^D_{ij}(x,x') = U_{ii'}(\theta)T^D_{i'j'kl}(x,x')\nu_k(x')\nu_l(x)U_{jj'}(\theta') \tag{3.71}$$

We check each term of the traction kernel $T^D_{i'j'kl}$ individually, dropping any obviously rotationally invariant parts (e.g. δ_{ij} and $|\bar{x}|$) for simplicity:

$$U_{ii'}(\theta)\delta_{i'k}\bar{x}_{j'}\bar{x}_l\nu_k(x')\nu_l(x)U_{jj'}(\theta') = [U_{ii'}(\theta)\nu_{i'}(x')][U_{jj'}(\theta')\bar{x}_{j'}][\nu_l(x)\bar{x}_l] \tag{3.72}$$

$$U_{ii'}(\theta)\delta_{i'l}\bar{x}_{j'}\bar{x}_k\nu_k(x')\nu_l(x)U_{jj'}(\theta') = [U_{ii'}(\theta)\nu_{i'}(x)][U_{jj'}(\theta')\bar{x}_{j'}][\nu_k(x')\bar{x}_k] \tag{3.73}$$

$$U_{ii'}(\theta)\delta_{j'k}\bar{x}_{i'}\bar{x}_l\nu_k(x')\nu_l(x)U_{jj'}(\theta') = [U_{ii'}(\theta)\bar{x}_{i'}][U_{jj'}(\theta')\nu_{j'}(x')][\nu_l(x)\bar{x}_l] \tag{3.74}$$

$$U_{ii'}(\theta)\delta_{j'l}\bar{x}_{i'}\bar{x}_k\nu_k(x')\nu_l(x)U_{jj'}(\theta') = [U_{ii'}(\theta)\bar{x}_{i'}][U_{jj'}(\theta')\nu_{j'}(x)][\nu_k(x')\bar{x}_k] \tag{3.75}$$

$$U_{ii'}(\theta)\bar{x}_{i'}\bar{x}_{j'}\bar{x}_k\bar{x}_l\nu_k(x')\nu_l(x)U_{jj'}(\theta') = [U_{ii'}(\theta)\bar{x}_{i'}][U_{jj'}(\theta')\bar{x}_{j'}][\nu_k(x')\bar{x}_k][\nu_l(x)\bar{x}_l] \tag{3.76}$$

Clearly, the traction kernel, as a whole, is also rotationally invariant. □

3.4.2 Analytic Expressions for Stokes Fourier Modes

Now that the kernels are in rotationally invariant form, we need to derive the analytical formulas for evaluating the Fourier modes. In this section, we reduce the Stokes single layer modal kernels to linear combinations of definite integrals of the form

$$I^m_{n,k}(r,z,r',z') = \frac{1}{\sqrt{2\pi}}\int_{-\pi}^{\pi} \frac{\cos nt \cos kt}{(r^2+r'^2-2rr'\cos t+(z-z')^2)^{m+1/2}}\,\mathrm{d}t \tag{3.77}$$

$$J^m_{n,k}(r,z,r',z') = \frac{-i}{\sqrt{2\pi}}\int_{-\pi}^{\pi} \frac{\sin nt \sin kt}{(r^2+r'^2-2rr'\cos t+(z-z')^2)^{m+1/2}}\,\mathrm{d}t \tag{3.78}$$

Proposition 3.7. *The Fourier modes of the transformed single layer velocity kernel*

$$\widetilde{V}^S_{ij}(x,x') = \frac{1}{8\pi|\bar{x}|}U_{ij}(t) + \frac{1}{8\pi|\bar{x}|^3}[U_{ii'}(\theta)\bar{x}_{i'}][U_{jj'}(\theta')\bar{x}_{j'}], \quad t=\theta-\theta' \tag{3.79}$$

are given by

$$\widetilde{V}^S_{11,n} = \frac{1}{8\pi}\left[I^0_{n,1} - \frac{rr'}{2}I^1_{n,2} + (r^2+r'^2)I^1_{n,1} - \frac{3rr'}{2}I^1_{n,0}\right] \tag{3.80}$$

$$\widetilde{V}^S_{21,n} = \frac{1}{8\pi}\left[-J^0_{n,1} + \frac{rr'}{2}J^1_{n,2} - r'^2 J^1_{n,1}\right] \tag{3.81}$$

$$\widetilde{V}^S_{31,n} = \frac{1}{8\pi}\left[rI^1_{n,1} - r'I^1_{n,0}\right](z-z') \tag{3.82}$$

$$\widetilde{V}^S_{12,n} = \frac{1}{8\pi}\left[-J^0_{n,1} - \frac{rr'}{2}J^1_{n,2} + r^2 J^1_{n,1}\right] \tag{3.83}$$

$$\widetilde{V}^S_{22,n} = \frac{1}{8\pi}\left[I^0_{n,1} - \frac{rr'}{2}I^1_{n,2} + \frac{rr'}{2}I^1_{n,0}\right] \tag{3.84}$$

$$\widetilde{V}^S_{32,n} = \frac{1}{8\pi}\left[rJ^1_{n,1}\right](z-z') \tag{3.85}$$

$$\widetilde{V}^S_{13,n} = \frac{1}{8\pi}\left[-r'I^1_{n,1} + rI^1_{n,0}\right](z-z') \tag{3.86}$$

$$\widetilde{V}^S_{23,n} = \frac{1}{8\pi}\left[r'J^1_{n,1}\right](z-z') \tag{3.87}$$

$$\widetilde{V}^S_{33,n} = \frac{1}{8\pi}\left[I^0_{n,0} + (z-z')^2 I^1_{n,0}\right] \tag{3.88}$$

Here we have suppressed the arguments of the kernel modes and the integrals for brevity.

Proof. Writing the kernel in matrix form

$$\widetilde{V}^S(x,x') = \frac{1}{8\pi}\left(\frac{1}{|\bar{x}|}\begin{bmatrix}\cos t & \sin t & 0\\ -\sin t & \cos t & 0\\ 0 & 0 & 1\end{bmatrix} + \frac{1}{|\bar{x}|^3}\begin{bmatrix} r - r'\cos t\\ r'\sin t\\ z-z'\end{bmatrix}\begin{bmatrix} r\cos t - r\\ r\sin t\\ z-z'\end{bmatrix}^{\top}\right) \tag{3.89}$$

Note that $t \mapsto \widetilde{V}^S_{ij}(t,r,z,r',z') = V^S_{ij}(t,s,s')$ is an even (resp. odd) function if $i-j$ is even (resp. odd). This simplifies the expression of the Fourier coefficients:

$$\widetilde{V}^S_{ij,n}(s,s') = \begin{cases} \dfrac{1}{\sqrt{2\pi}}\displaystyle\int_{-\pi}^{+\pi} \widetilde{V}^S_{ij}(t,s,s')\cos nt\,\mathrm{d}t & \text{if} \quad i-j \text{ is even}\\[2ex] \dfrac{-i}{\sqrt{2\pi}}\displaystyle\int_{-\pi}^{+\pi} \widetilde{V}^S_{ij}(t,s,s')\sin nt\,\mathrm{d}t & \text{if} \quad i-j \text{ is odd}\end{cases} \tag{3.90}$$

Next, numerators of $\widetilde{V}^S_{ij}$ can be written as a linear combination of cosine (resp. sine) functions when $i-j$ is even (resp. odd); for example

$$\begin{aligned}\widetilde{V}^S_{11}(t,s,s') &= \frac{1}{8\pi}\left[\frac{\cos t}{q^{1/2}} + \frac{(r-r'\cos t)(r\cos t - r')}{q^{3/2}}\right]\\ &= \frac{1}{8\pi}\left[\frac{\cos t}{q^{1/2}} + \frac{-rr'\cos^2 t + (r^2+r'^2)\cos t - rr'}{q^{3/2}}\right]\\ &= \frac{1}{8\pi}\left[\frac{\cos t}{q^{1/2}} + \frac{-\frac{rr'}{2}\cos(2t) + (r^2+r'^2)\cos t - \frac{3rr'}{2}}{q^{3/2}}\right]\end{aligned} \tag{3.91}$$

and

$$\begin{aligned}\widetilde{V}^S_{12}(t,s,s') &= \frac{1}{8\pi}\left[\frac{\sin t}{q^{1/2}} + \frac{(r-r'\cos t)r\sin t}{q^{3/2}}\right]\\ &= \frac{1}{8\pi}\left[\frac{\sin t}{q^{1/2}} + \frac{-rr'\sin t\cos t + r^2\sin t}{q^{3/2}}\right]\\ &= \frac{1}{8\pi}\left[\frac{\sin t}{q^{1/2}} + \frac{\frac{-rr'}{2}\sin(2t) + r^2\sin t}{q^{3/2}}\right]\end{aligned} \tag{3.92}$$

where we define

$$q = q(t, s, s') = |x - x'|^2 = r^2 + r'^2 + (z - z')^2 - 2rr' \cos t \tag{3.93}$$

Rest of the proof is a simple matter of plugging these formulas into (3.90) as appropriate, using the linearity of integrals and using the definitions of the I- and J-integrals. □

Proposition 3.8. *The Fourier modes of the transformed single layer pressure kernel*

$$\widetilde{P}^S_j(x, x') = \frac{1}{4\pi|\bar{x}|^3}[U_{jj'}(\theta')\bar{x}_{j'}] \tag{3.94}$$

are given by

$$\widetilde{P}_{1,n} = \frac{1}{4\pi}\left[rI^1_{n,1} - r'I^1_{n,0}\right] \tag{3.95}$$

$$\widetilde{P}_{2,n} = \frac{1}{4\pi}\left[rJ^1_{n,1}\right] \tag{3.96}$$

$$\widetilde{P}_{3,n} = \frac{1}{4\pi}\left[I^1_{n,0}\right](z - z') \tag{3.97}$$

Proof. Follow the same steps as in the proof of Proposition 3.7 □

Proposition 3.9. *Let*

$$A = -\frac{\nu_r r'}{2}, \quad B = \nu_r r + \nu_z(z - z') \tag{3.98}$$

Then the Fourier modes of the transformed single layer traction kernel

$$\widetilde{T}^S_{ij}(x, x') = -\frac{3}{4\pi|\bar{x}|^3}[U_{ii'}(\theta)\bar{x}_{i'}][U_{jj'}(\theta')\bar{x}_{j'}][\nu_l(x)\bar{x}_l] \tag{3.99}$$

are given by

$$\widetilde{T}^S_{11,n} = -\frac{3}{4\pi}\Bigg[-\frac{Arr'}{2}I^2_{n,3} + \left(A(r^2+r'^2) - \frac{Brr'}{2}\right)I^2_{n,2} \tag{3.100}$$

$$+\big(B(r^2+r'^2) - 2Arr'\big)I^2_{n,1} + \left(A(r^2+r'^2) - \frac{3Brr'}{2}\right)I^2_{n,0}\Bigg] \tag{3.101}$$

$$\widetilde{T}^S_{21,n} = -\frac{3}{4\pi}\left[\frac{Arr'}{2}J^2_{n,3} + \left(\frac{Brr'}{2} - Ar'^2\right)J^2_{n,2} + \left(\frac{Arr'}{2} - Br'^2\right)J^2_{n,1}\right] \tag{3.102}$$

$$\widetilde{T}^S_{31,n} = -\frac{3}{4\pi}\left[ArI^2_{n,2} + (Br - Ar')I^2_{n,1} + (Ar - Br')I^2_{n,0}\right](z - z') \tag{3.103}$$

$$\widetilde{T}^S_{12,n} = -\frac{3}{4\pi}\left[-\frac{Arr'}{2}J^2_{n,3} + \left(Ar^2 - \frac{Brr'}{2}\right)J^2_{n,2} + \left(Br^2 - \frac{Arr'}{2}\right)J^1_{n,1}\right] \tag{3.104}$$

$$\widetilde{T}^S_{22,n} = -\frac{3}{4\pi}\left[-\frac{Arr'}{2}I^2_{n,3} - \frac{Brr'}{2}I^2_{n,2} + \frac{Brr'}{2}I^2_{n,0}\right] \tag{3.105}$$

$$\widetilde{T}^S_{32,n} = -\frac{3}{4\pi}\left[ArJ^2_{n,2} + BrJ^2_{n,1}\right](z - z') \tag{3.106}$$

$$\widetilde{T}^S_{13,n} = -\frac{3}{4\pi}\left[-Ar'I^2_{n,2} + (Ar - Br')I^2_{n,1} + (Br - Ar')I^2_{n,0}\right](z - z') \tag{3.107}$$

$$\widetilde{T}^S_{23,n} = -\frac{3}{4\pi}\left[Ar'J^2_{n,2} + Br'J^2_{n,1}\right](z - z') \tag{3.108}$$

$$\widetilde{T}^S_{33,n} = -\frac{3}{4\pi}\left[AI^2_{n,1} + BI^2_{n,0}\right](z - z')^2 \tag{3.109}$$

Proof. Note that our choice of A and B implies

$$\nu_l(x)\bar{x}_l = 2A\cos t + B, \quad t = \theta - \theta' \tag{3.110}$$

Using trigonometric identities

$$\begin{aligned}(2A\cos t + B)(a\cos 2t + b\cos t + c) = {} & Aa\cos 3t \\ & + (Ab + Ba)\cos 2t \\ & + (Aa + Ac + Bb)\cos t \\ & + (Ab + Bc)\end{aligned} \tag{3.111}$$

and

$$
\begin{aligned}
(2A\cos t + B)(a\sin 2t + b\sin t) = {} & Aa\sin 3t \\
& + (Ab + Ba)\sin 2t \\
& + (Aa + Bb)\sin t
\end{aligned} \tag{3.112}
$$

we can reduce the numerators of $\widetilde{T}_{ij}^{S}$ into linear combinations of cosine and sine functions, just as in the case of single layer velocity kernel. The rest follows. $\square$

The transformed Stokes double layer kernels also admit similar expressions in terms of the I- and J-integrals. In particular, the expressions for the double layer velocity potential modes are nearly identical to those of the single layer traction kernel. While the expressions for the pressure, and in particular, the traction kernel are significantly more complex, they are still computable using algebraic manipulation programs like Mathematica.

3.4.3 Expansion of the Integrals in Terms of Toroidal Functions

The analytic expressions for the modal transformed Stokes kernels in the previous section are expressed in terms of the I- and J-integrals. In this section, we connect the evaluation of these integrals with toroidal functions, which finally gives us an efficient way to compute the modal kernels, especially near the singularity.

Toroidal functions are Legendre functions of the second kind with half-integer degrees: $Q_{n-1/2}^{m}$ with $m \in \mathbb{N}_0$ and $n \in \mathbb{Z}$. Analogous to spherical harmonics, they show up when we solve the Laplace equation in the toroidal coordinates (a orthogonal coordinate system suitable for axisymmetric geometries). We will now show that we can write the I- and J-integrals as appropriate combinations of toroidal functions:

Proposition 3.10. *Let*

$$
\chi = \frac{r^2 + r'^2 + (z - z')^2}{2rr'} \tag{3.113}
$$

Then

$$I^m_{n,k} = \frac{(-1)^m}{\Gamma(m+1/2)} \frac{1}{(2rr')^{m+1/2}(\chi^2-1)^{m/2}} \left[Q^m_{n+k-1/2}(\chi) + Q^m_{n-k-1/2}(\chi)\right] \tag{3.114}$$

$$J^m_{n,k} = \frac{(-1)^m}{\Gamma(m+1/2)} \frac{i}{(2rr')^{m+1/2}(\chi^2-1)^{m/2}} \left[Q^m_{n+k-1/2}(\chi) - Q^m_{n-k-1/2}(\chi)\right] \tag{3.115}$$

Proof. We can use the trigonometric identities

$$\cos(nt)\cos(kt) = \frac{1}{2}\left[\cos(n+k)t + \cos(n-k)t\right] \tag{3.116}$$

$$-\sin(nt)\sin(kt) = \frac{1}{2}\left[\cos(n+k)t - \cos(n-k)t\right] \tag{3.117}$$

and that $|\bar{x}| = 2rr'(\chi - \cos t)$ to reduce the integrals $I^m_{n,k}$ and $J^m_{n,k}$ to sums or differences of integrals of the form

$$\frac{1}{(2rr')^{m+1/2}} \int_{-\pi}^{+\pi} \frac{\cos nt}{(\chi - \cos t)^{m+1/2}}\, \mathrm{d}t = \frac{1}{(2rr')^{m+1/2}} \int_{-\pi}^{+\pi} \frac{e^{-int}}{(\chi - \cos t)^{m+1/2}}\, \mathrm{d}t \tag{3.118}$$

Next, from [84, Eq. 14.3.10] and [84, Eq. 14.19.6], after substituting $\mu = m$ and $\chi = \cosh \xi$, we obtain the identity

$$\frac{(-1)^m Q^m_{-1/2}(\chi)}{\Gamma(m+1/2)} + 2\sum_{n=1}^{\infty} \frac{(-1)^m Q^m_{n-1/2}(\chi)}{\Gamma(m+1/2)} \cos nt = \left(\frac{\pi}{2}\right)^{1/2} \frac{(\chi^2-1)^{m/2}}{(\chi - \cos t)^{m+1/2}} \tag{3.119}$$

Using the relations $\cos nt = (e^{int} + e^{-int})/2$ and $Q^m_{-n-1/2}(\chi) = Q^m_{n-1/2}(\chi)$ and after rearranging the terms, we obtain

$$\frac{1}{(\chi - \cos t)^{m+1/2}} = \frac{1}{2\pi} \sum_{n=-\infty}^{\infty} \left[(-1)^m \frac{\sqrt{2\pi}}{\Gamma(m+1/2)} \frac{2}{(\chi^2-1)^{m/2}} Q^m_{n-1/2}(\chi)\right] e^{int} \tag{3.120}$$

This is simply the Fourier series for $(\chi - \cos t)^{-m-1/2}$; the Fourier coefficients are given by

$$\int_{-\pi}^{\pi} \frac{e^{-int}}{(\chi - \cos t)^{m+1/2}}\, \mathrm{d}t = (-1)^m \frac{\sqrt{2\pi}}{\Gamma(m+1/2)} \frac{2}{(\chi^2-1)^{m/2}} Q^m_{n-1/2}(\chi) \tag{3.121}$$

The rest of the proof is simple algebra. □

3.4.4 Numerical Computation of Toroidal Functions

We compute the $-\frac{1}{2}$ and $\frac{1}{2}$ degree zero-order Legendre functions of the second kind using the complete elliptic integrals [87]:

$$Q^0_{-1/2}(\chi) = \mu K(\mu) \tag{3.122}$$

$$Q^0_{1/2}(\chi) = \mu\chi K(\mu) - \sqrt{2(\chi+1)}E(\mu) \tag{3.123}$$

where K and E are the complete elliptic integrals of the first and second kind, respectively.

$$\mu = \sqrt{\frac{2}{1+\chi}} \tag{3.124}$$

Once we have these two values, we can use them in the forward recurrence relation [87]

$$Q^0_{n-1/2}(\chi) = 4\frac{n-1}{2n-1}\chi Q^0_{n-3/2}(\chi) - \frac{2n-3}{2n-1}Q^0_{n-5/2}(\chi) \tag{3.125}$$

to compute the higher degree function values. Once we have the higher degree functions, we can use the recurrence relation for the order [35]

$$(\chi^2-1)^{1/2}Q^{m+1}_{n+1/2} = (n-m+1/2)\chi Q^m_{n+1/2} - (n+m+1/2)Q^m_{n-1/2} \tag{3.126}$$

for $n = 0, 1, \ldots$, and

$$(\chi^2-1)^{1/2}Q^{m+1}_{-1/2} = (1/2-m)Q^m_{1/2}(\chi) - (1/2+m)\chi Q^m_{-1/2}(\chi) \tag{3.127}$$

for $n = -1$. To compute the Legendre function with degree lower than $-\frac{1}{2}$, we use the identity

$$Q^m_{n-1/2}(\chi) = Q^m_{-n-1/2}(\chi) \tag{3.128}$$

Unfortunately, the forward recurrence relation (3.125) is unstable for all $\chi > 1$, so we need to be careful how we use it:

- If $\chi \approx 1$, then the instability is relatively mild, and we only see it when computing the Legendre functions with very large degree [87]. So we can use the algorithm described above below a certain threshold value, e.g. $\chi < 1.0005$.
- For $\chi \geq 1.0005$, the points s and s' on the curve γ are well-separated. Hence we can use the fast Fourier transform algorithm to directly compute the Fourier coefficients.

Assuming that we ignore any Fourier frequencies with magnitude higher than n_f, this hybrid approach allows us to compute all $(2n_f + 1)$ Fourier coefficients given a value of χ (equivalently, a pair of points s and s' on the curve) in $O(n_f \log n_f)$ steps. Thus, the total cost of computing the modal kernel matrices for Stokes equations with n_p panels is given by $O(n_p^2 n_f \log n_f)$.

3.5 Extension to Multibody Systems

In this section, we extend our algorithm to solve Stokes equation in a multibody system, where each of the bodies are axisymmetric, but with potentially different axes of symmetries.

3.5.1 Bodies with Arbitrary Axes of Rotational Symmetry

So far, we have only considered surfaces which are symmetric about the z-axis. We will now generalize our algorithm for surfaces with arbitrary axis of symmetry. Suppose we are solving the integral equation

$$\mathcal{V}^S[\rho] = f \iff \int_\Omega V_{ij}^S(y, y')\rho_j(y')\,\mathrm{d}\Omega(y') = f_i(y) \tag{3.129}$$

where Ω has the unit vector v as the axis of symmetry. Pick any orthogonal matrix R whose last column is v, then the surface

$$\Gamma = R^{-1}\Omega := \left\{x \in \mathbb{R}^3 : Rx \in \Omega\right\} \tag{3.130}$$

forms a rotationally invariant surface symmetric about the z-axis. Note that for $y = Rx$ and $y' = Rx'$, we have

$$\bar{y}_i\bar{y}_j = R_{ii'}\bar{x}_{i'}\bar{x}_{j'}R_{jj'} \tag{3.131}$$

Further, since R is orthogonal, distances are between two points don't change under this transformation. Hence we obtain

$$V_{ij}^S(y, y') = R_{ii'}V_{i'j'}^S(x, x')R_{jj'} \tag{3.132}$$

With this, we can rewrite (3.129) as

$$R_{ii'}\int_\Gamma V_{i'j'}^S(x, x')R_{jj'}\rho_j(Rx')\,\mathrm{d}\Gamma(x') = f_i(Rx) \tag{3.133}$$

and after multiplying both sids by $R^\top$,

$$\int_\Gamma V_{i'j'}^S(x, x')[R_{jj'}\rho_j(Rx')]\mathrm{d}\Gamma(x') = [R_{ii'}f_i(Rx)] \tag{3.134}$$

Thus, if we use modified boundary data $R^\top f(Rx)$ and modified boundary density $R^\top\rho(Rx)$, this has the exact same form as (3.2), and we can then use our standard solver. Other layer potentials can be modified in a similar manner to handle bodies with arbitrary axes of symmetry.

3.5.2 Fast Multipole Method and Multibody Solver

Now, suppose that we are solving a multibody system, i.e. we have axisymmetric surfaces $\Omega^{(i)}$ for $1 \leq i \leq n_b$, each with their own axis of symmetry. Suppose we have boundary data $f^{(i)}$ on the i-th body. Then our goal is to solve

$$\sum_{j=1}^{n_b} \mathcal{V}^{S,(i)}[\rho^{(i)}](x) = f^{(i)}(x), \quad x \in \Omega^{(i)} \tag{3.135}$$

Here we have absorbed the rotation matrices $R^{(i)}$ for the axisymmetric bodies inside the layer potential definition. Discretizing this system leads to the linear system

$$\begin{bmatrix} \mathsf{V}^{(1,1)} & \cdots & \mathsf{V}^{(1,n_b)} \\ \vdots & \ddots & \vdots \\ \mathsf{V}^{(n_b,1)} & \cdots & \mathsf{V}^{(n_b,n_b)} \end{bmatrix} \begin{bmatrix} \mathsf{\rho}^{(1)} \\ \vdots \\ \mathsf{\rho}^{(n_b)} \end{bmatrix} = \begin{bmatrix} \mathsf{f}^{(1)} \\ \vdots \\ \mathsf{f}^{(n_b)} \end{bmatrix} \tag{3.136}$$

We solve this system using iterative methods (e.g. GMRES). As a preconditioner, we use the diagonal blocks; i.e. we split the matrix V into two parts

$$\mathsf{V} = \mathsf{D} + \mathsf{N} \tag{3.137}$$

with

$$\mathsf{D} = \begin{bmatrix} \mathsf{V}^{(1,1)} & & \\ & \ddots & \\ & & \mathsf{V}^{(n_b,n_b)} \end{bmatrix}, \quad \mathsf{N} = \begin{bmatrix} 0 & \cdots & \mathsf{V}^{(1,n_b)} \\ \vdots & \ddots & \vdots \\ \mathsf{V}^{(n_b,1)} & \cdots & 0 \end{bmatrix}, \tag{3.138}$$

Then the solution of (3.136) is given by

$$(\mathsf{D}+\mathsf{N})\rho = \mathsf{f} \implies \rho + \mathsf{D}^{-1}\mathsf{N}\rho = \mathsf{D}^{-1}\mathsf{f} \tag{3.139}$$

We use our single-body solver to speed up the evaluation of $\mathsf{D}^{-1}\mathsf{v}$ given an array v. We further accelerate the evaluation of $\mathsf{N}\rho$ using FMM, so that the left hand side of (3.139) can be computed very quickly. This leads to considerable speedup in solving the system, since GMRES repeatedly evaluates the matrix-vector product as part of its algorithm.

3.6 Numerical Results

We now outline the numerical experiments we performed to inspect the efficiency and accuracy of the algorithm presented in this paper. We implemented the algorithm in MATLAB, and used the `stfmmlib3d` library [37] for Stokes FMM. We ran the single body simulations on a personal laptop equipped with a 8-core 4.9 GHz Intel Core i9 11900H processor with 32 GB RAM, and the multibody simulations on the Haswell nodes (each equipped with two 12-core 2.5 GHz Intel Xeon E5-2680 v3 processors and 128 GB RAM) of the Flux cluster at University of Michigan.

3.6.1 Accuracy and Performance of the Single Body Solver

We solved a sequence of problems with the Stokes single layer velocity BIE using different discretizations ($2 \leq n_p \leq 30$ and $10 \leq n_f \leq 150$) of the prolate ellipsoid $\Gamma : \frac{r^2}{0.5^2} + z^2 = 1$. We generated the boundary condition in these simulations placing 5 random Stokeslets on the surface $\frac{r^2}{0.3^2} + \frac{z^2}{0.8^2} = 1$ interior to Γ. After solving for the density function on Γ, we then evaluated the velocity field on a regular grid on the surface $\frac{r^2}{0.7^2} + \frac{z^2}{1.2^2} = 1$, and compared these velocities against those generated by the Stokeslets. The entire setup is transformed through a random orthogonal matrix.

We use two error criteria:

$$\epsilon_{\text{abs}} = \max_x \max_i \left| u_i^{\text{exact}}(x) - u_i^{\text{apprx}}(x) \right| \tag{3.140}$$

$$\epsilon_{\text{rel}} = \max_x \max_i \frac{|u_i^{\text{exact}}(x) - u_i^{\text{apprx}}(x)|}{|u_i^{\text{exact}}(x)|} \tag{3.141}$$

The outer maximum is taken over all target points in the exterior of the ellipsoid, and the inner maximum over the three components of the velocities.

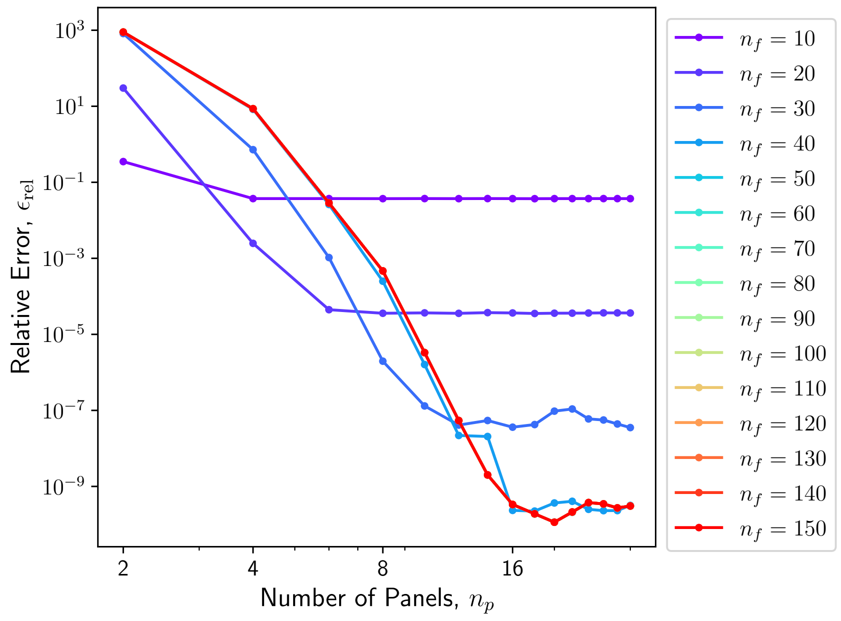
In Figure 3.2, we plot the relative error against the number of panels and number of Fourier modes used to discretize the ellipsoidal surface Γ. We see that this scheme attains accuracy of the order 10^{-9} as we increase the number of panels and Fourier modes simultaneously. We note that for steady improvement in accuracy, both the number of panels and number of Fourier modes should be increased simultaneously; otherwise the convergence suffers. For the rest of numerical experiments in the paper, we keep these two quantities proportional.

To verify that the complexity of the scheme matches our estimates in subsection 3.3.3, we plot the time spent in different stages of the algorithm against total number of quadrature nodes $N = 10n_p(2n_f+1)$ in Figure 3.3. We assume a power-law relation between runtime and quadrature points: $T \propto N^\alpha$. In the figure, we report the empirical α-values computing from the simulation data; we observe that these values closely match the theoretical predictions.

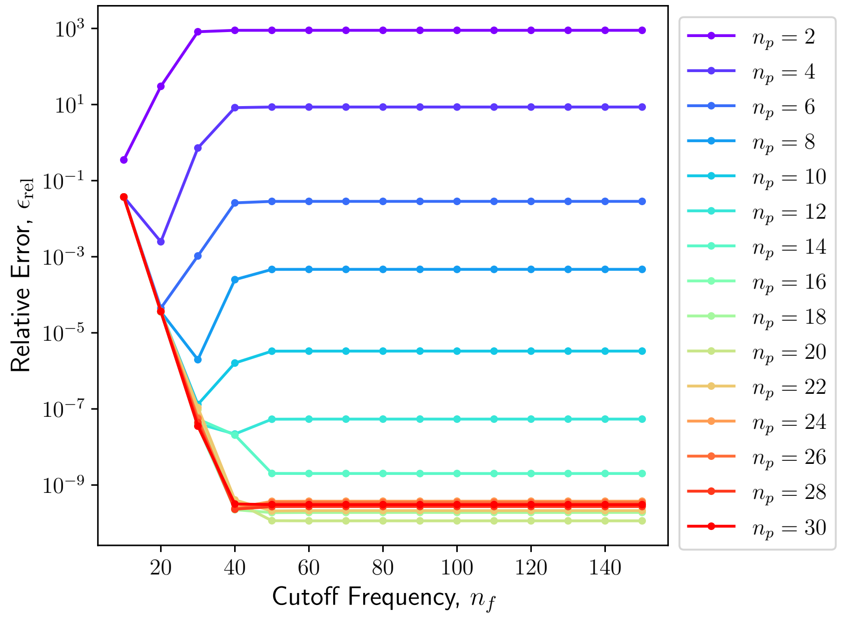
3.6.2 Accuracy and Performance of Multibody Solver

We test the accuracy and performance of the multibody solver with the single layer formulation. We place prolate and oblate ellipsoids randomly in the free space, and place Stokeslets at the center of these bodies. The boundary data on the ellipsoid surfaces are computed by summing up the velocity fields generated by the individual Stokeslets.

We use a combination of our fast axisymmetric solvers and the FMM to solve for the boundary data; stabilized bi-conjugate gradient descent (BiCGStab) is used as the iterative



(a)



(b)

Figure 3.2: Convergence of the relative error ϵ_{rel} of the Stokes single layer velocity field compared against the analytic flow generated by a random placement of Stokeslets as the number of panels n_p and cutoff frequency n_f are varied. We see that the performance of our scheme is best when we keep $n_p \propto n_f$. In addition, the error drops algebraically with increasing n_p at a high-order, and it drops spectrally with n_f.

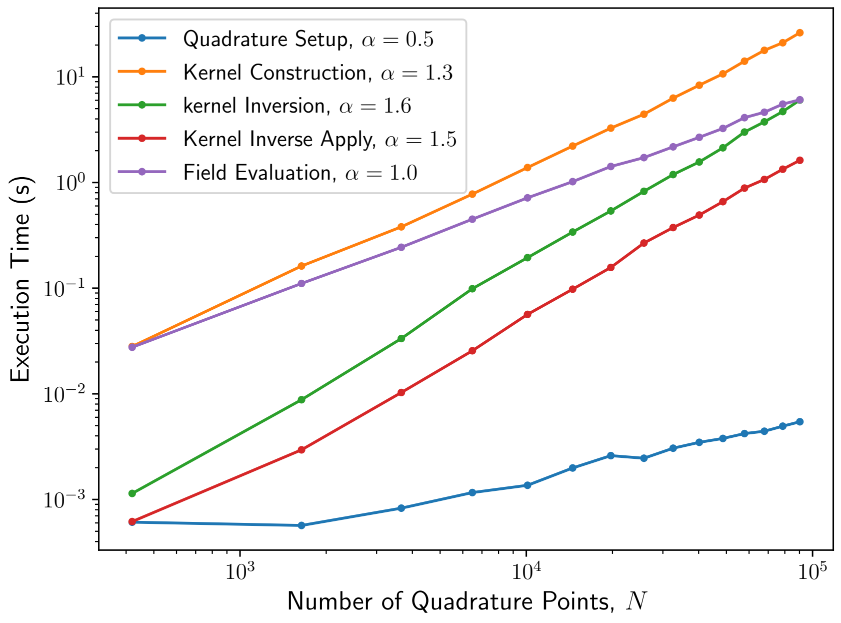


Figure 3.3: The execution time of various stages of our axisymmetric Stokes solver plotted against the total number N of quadrature nodes on the surface. We fit a curve $T \propto N^{\alpha}$ for each of these time segments, and report the best-fit α value in the legends.

solver for (3.139). We then evaluate the velocity field at test points in the exterior domain, and compare it with the analytical values (from the Stokeslets) using the absolute error metric.

In Table 3.1, we report the execution times and observed errors. We see that the kernel computation and factorization times, and the times for computing the far body interactions using the FMM, and inverting near body interactions per iteration scale approximately linearly. Additionally, the error in these simulations are fairly small, especially when we only used 5 panels and 25 Fourier modes per ellipsoid.

3.6.3 Simulation of Stokes Flow past Active Matter

We applied our multi-body solver to simulate Stokes flow past active matter. The flow is drive by 125 randomly shaped and oriented ellipsoids in a $5 \times 5 \times 5$ grid, each imposing a slip boundary condition on the external fluid. The slip velocity for each particle points from one pole to the other; the velocity is 10 units near the equator and drops to zero at the poles.

Table 3.1: Performance and accuracy of the multibody Stokes solver with n_b bodies and n_{omp} OpenMP threads using 2550 quadrature points per body ($n_p = 5$ and $n_f = 25$). We report the one-time kernel matrix construction and inversion times T_{ker} and T_{inv}, respectively. The metric n_{op} represents the total number of matrix-vector multiplication operations in the BiCGStab iterative solvers required before convergence, and T_{near} and T_{fmm} are execution times for near and far evaluations. The accuracy of the multibody solver is captured using the absolute error metric ϵ_{abs}. All execution times are reported in seconds, and the missing entries correspond to situations where the BiCGStab solver did not converge within 200 iterations.

n_{omp}	n_b	T_{ker}	T_{inv}	n_{op}	T_{near}	T_{fmm}	ϵ_{abs}
1	2	3.8	0.1	14	1.1	2.0	4.4471×10^{-7}
	4	4.2	1.3	22	2.5	4.5	1.0709×10^{-6}
	8	6.6	3.1	27	5.1	16.5	8.4709×10^{-7}
	16	13.5	6.1	29	10.1	37.9	7.1951×10^{-7}
	32	24.5	12.1	54	21.0	85.7	7.1479×10^{-7}
	64	49.5	26.8	177	43.0	196.1	4.4704×10^{-6}
2	4	3.5	0.9	23	1.8	2.7	7.6096×10^{-7}
	8	4.1	1.6	22	3.7	9.1	2.5302×10^{-6}
	16	7.0	3.2	46	7.8	21.0	1.3312×10^{-6}
	32	14.2	6.9	56	15.7	47.6	7.3557×10^{-7}
	64	26.9	14.7	123	27.6	100.5	7.9819×10^{-7}
	128	52.6	26.1	—	56.6	214.0	—
4	8	3.7	1.2	27	3.4	6.3	1.1476×10^{-6}
	16	4.5	2.1	62	7.0	17.9	3.9708×10^{-7}
	32	7.3	5.8	36	14.1	28.3	8.5426×10^{-7}
	64	13.7	10.4	179	28.5	63.0	2.1424×10^{-6}
	128	28.5	21.2	—	58.3	145.6	—
	256	50.5	46.2	—	117.4	280.5	—

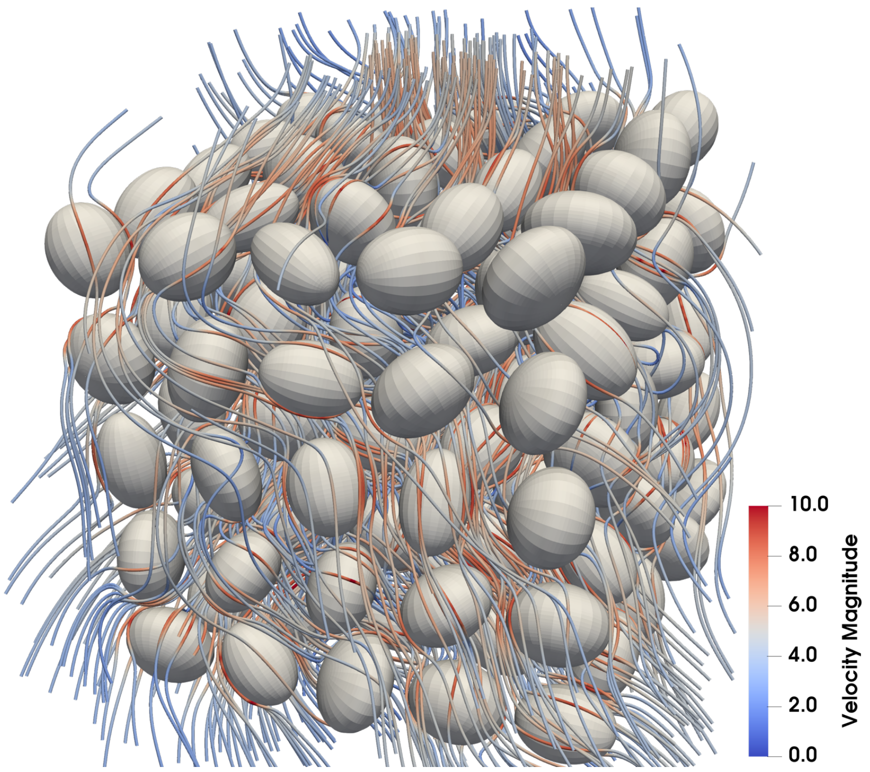


Figure 3.4: Streamlines of a Stokes flow generated by 125 randomly shaped and oriented ellipsoids, each imposing a slip boundary condition on the fluid.

The resulting streamlines are shown in Figure 3.4. We can see that the fluid picks up speed near the equator of the ellipsoids (the streamlines attach to the body), as expected from this scenario.

3.7 Conclusions

In this chapter, we explored BIEs with rotationally invariant kernels on axisymmetric surfaces can be decoupled using the Fourier series. However, the Stokes kernels do not satisfy this rotational invariance property out-of-the-box. We introduced a linear change of coordinates for the boundary integral equations and proved that all of the single and double layer velocity, pressure and traction kernels can be transformed into the required rotationally invariant form.

Using this observation, we developed a fast solution scheme for the Stokes BIEs, analogous to the ones developed in [87, 43, 33] for Laplace, Helmholtz and Maxwell equations. Our kernel decoupling strategy leads to an order of magnitude improvement over direct discretization of the surface BIEs, and we achieved $O(10^{-9})$ relative errors in the velocity field using relatively few panels and frequency cutoff. We also combined our solver with the fast multipole method (FMM) to construct a fast solver for Stokes flow past multiple axisymmetric bodies.

CHAPTER 4

Compression of Discrete Element Method Simulation Data using the Tensor-Train Decomposition

Preamble. In this chapter, we introduce the tensor-train decomposition as a tool for compressing scientific data. We focus on compressing output data from discrete element method (DEM) simulations, concentrating on both raw (e.g. particle position, velocity etc.) and derived (e.g. stress, strain etc.) datasets. Using a geometry-driven "tensorization" to increase the dimensionality of the datasets, we achieve high compression ratios on both types of datasets. This is ongoing work with Eduardo Corona, Paramsothy Jayakumar and Shravan Veerapaneni.

4.1 Introduction

The Discrete Element Method (DEM) is widely recognized as an effective simulation framework for granular media modeled as a collection of individual particles. It uses simple first-principle physical laws to describe the particle response to body forces (e.g. gravity and inertia) and dissipative contact forces (e.g. friction), leading to high-fidelity and robust predictions. In recent years, substantial progress has been made to address algorithmic challenges in DEM simulations: in fast optimization solvers [101, 105, 75, 24], parallel computing frameworks [27, 74, 79, 16, 111], and multiscale modeling [65, 113, 94]. Unfortunately, for large-scale problems such as those in vehicle-terrain simulation, each run can still take days or even weeks to complete. In learning scenarios such as mobility map construction [53, 72] and autonomous driving, each of these expensive runs then corresponds to only one point in the exploration of a high-dimensional parametric space.

As algorithmic and learning frameworks continue to improve, addressing large dataset storage and management problems is imperative. DEM vehicle-soil simulations, for example, generate a lot of data, including particle positions and velocities, pairwise contact forces and soil-vehicle interaction forces at each timestep. One simulation can easily produce tens of gigabytes of data. In current state-of-the-art setups, only the bare minimum of the computational results is kept. This is not only wasteful of weeks or months of expensive, energy-intensive computational work; it precludes any current or future knowledge we might obtain from the entire dataset.

We note that these issues are far from exclusive to DEM. Across our current computational landscape, we are producing tremendous amounts of data, outpacing our storage capacity. This limits our ability to learn, and makes data transfer, visualization and analysis significant bottlenecks in our pipelines. This is most acutely felt in online, interactive environments in which streaming data must be processed under restrictive computational and storage capabilities. Data compression techniques are therefore of great interest across the entire computational simulation paradigm.

4.1.1 Related Works

There is a vast literature on general scientific data reduction methods; for an informative survey, see [62]. While general purpose encoders such as `fpzip` [67] have achieved a degree of success exploiting data coherence in scientific datasets, more efficient representations require us to incorporate assumptions on specific types of data-sparsity or inherent structures we want to target. Techniques such as principal component analysis (PCA), wavelet transforms and multi-resolution schemes have been traditionally deployed to exploit inherent structure due to smoothness and data locality.

Scientific data, such as that obtained in DEM simulations, is almost inevitably multi-dimensional, as variables of interest are tracked in space, time and across a range of parameters. Tensor factorization schemes are a natural extension of matrix-factorization based data compression methods like the PCA. They explicitly tackle the high-dimensional nature of scientific data. Key features of tensor decompositions include addressing the curse of dimensionality and producing significantly reduced representations of datasets that allow rapid full or partial reconstruction and fast linear algebra in compressed format at the expense of allowing user-specified error tolerance.

Due to recent algorithmic developments, there is a rapidly growing body of literature for tensor decomposition algorithms and their applications to data mining, data fusion and scientific data compression [58, 1, 38, 23]. The work reviewed in these articles suggests the various tensor representations (CP, Tucker, Tensor-Train) and their hierarchical counterparts ($\mathcal{H}$-Tucker and Quantized Tensor-Train) can be incredibly effective as tools for data compression and analysis. For instance, in [12], the authors developed a distributed implementation of the Tucker decomposition, demonstrating dramatic compression of a number of combustion datasets (e.g. 760-200K fold compression for a 10^{-2} relative tolerance) and fast partial reconstruction in limited computing environments. More recently, there has been a large push towards developing tensor compression techniques for streaming data with CP [83, 98], Tucker [71, 103] and tensor train [69, 68, 107] factorizations.

In the context of DEM simulation, methods exploiting low-rank data structure have not focused on data compression and reconstruction, but on producing reduced order dynamical models. Many of these approaches use the singular value decomposition (SVD) to construct the reduced representation. In particular, [117] uses the related proper orthogonal decomposition (POD) to extract the principal modes of the dynamics from time snapshots of a damped DEM simulation.

4.1.2 Our Contributions

We develop tensor-train (TT) [85] methods tailored to both raw and derived data from DEM simulations. Our contributions to the state of the art of scientific data compression are as follows:

- We present what is, to our knowledge, the first adaptation of tensor compression methods to nonsmooth granular media simulation data.
- Our methods feature a compression scheme for streaming data, in which batches of simulation data can be efficiently incorporated to a compressed representation as soon as they are generated.
- We find most applications of tensor methods in the literature focus on compression along the dimensions suggested by the data (e.g. x, y, z, time). By "tensorizing" input data using a standard space-filling curve ordering, we demonstrate the ability of Quantized Tensor Train (QTT) methods to provide further compression by exploiting redundancies in the data at different levels of resolution.

We demonstrate high compression ratios of $O(10^4)$ for both raw and derived DEM datasets of size $O(1)$ GB while keeping the relative reconstruction error at $O(10^{-1})$. This enables us to store the reduced datasets using only $O(100)$ KB memory, at a fraction of the cost of the original dataset, while still being able to use it for post-processing and learning tasks.

4.2 The Tensor-Train Decomposition

From a computational perspective, we consider tensors to be a direct generalization of two-dimensional matrices. They are represented as multi-dimensional arrays of real numbers. The entries of a d-dimensional tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ are expressed as $\mathcal{X}(i_1, \ldots, i_d)$ with indices $1 \leq i_k \leq n_k$ for $1 \leq k \leq d$. Given $n = \max\{n_1, \ldots, n_d\}$, assuming tensor dimensions are comparable in size, it can be readily observed that storing all of its entries incurs $O(n^d)$ storage cost. Additionally, any operation accessing all of the entries of the tensor will have a computational complexity that is similarly exponential in the dimensionality d.

Tensor factorizations leverage low-rank structures of the underlying tensor in order to overcome this "curse of dimensionality". [58] provides a detailed review of the canonical polyadic (CP) factorization and the Tucker decomposition. These two formats are direct generalizations of the popular singular value decomposition (SVD) for matrices, and are widely used in high-dimensional data analysis.

In this work, we instead focus on using the tensor-train (TT) decomposition, introduced in [86], as our main tool for data compression. The reasons for this choice are threefold:

- We can compute the TT decomposition of a tensor utilizing the SVD in a numerically stable manner.
- Many linear algebra operations are very efficient in the TT format, a fact we utilize in constructing streaming data compression schemes.
- The outputs from linear algebra operations in TT format often inflates the tensor ranks, leading to inefficient storage. We can optimize the ranks of such a TT representation via basic matrix operations: orthogonalization and SVD.

In this section, we explore each of these key elements in more detail.

4.2.1 TT Format

From linear algebra, we know that a rank-r matrix $X \in \mathbb{R}^{n_1 \times n_2}$ can be factorized as

$$X = AB, \quad A \in \mathbb{R}^{n_1 \times r}, \quad B \in \mathbb{R}^{r \times n_2} \tag{4.1}$$

The individual entries of the data matrix then can be recovered as

$$X(i_1, i_2) = A(i_1)B(i_2), \quad 1 \le i_1 \le n_1, \quad 1 \le i_2 \le n_2 \tag{4.2}$$

where $A(i_1) \in \mathbb{R}^{1\times r}$ is the i_1-th row of A, and $B(i_2) \in \mathbb{R}^{r\times 1}$ is the i_2-th column of B. The TT format is a direct generalization of this decomposition, expressing an entry of a d-dimensional tensor $\mathcal{X} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ as

$$\mathcal{X}(i_1, \ldots, i_d) = X_1(i_1)\cdots X_d(i_d), \quad X_k(i_k) \in \mathbb{R}^{r_{k-1}\times r_k}, \quad 1 \le i_k \le n_k, \quad 1 \le k \le d \tag{4.3}$$

The matrix sizes $r_0, \ldots, r_d$ are called the TT ranks of the decomposition. Note that since the matrix product on the right-hand side produces a scalar, this equality requires $r_0 = r_d = 1$. We can stack the matrices $\{X_k(i_k) : 1 \le i_k \le n_k\}$ corresponding to the k-th dimension of the data tensor into a three-dimensional tensor $\mathcal{X}_k$:

$$\mathcal{X}_k(\alpha_{k-1}, i_k, \alpha_k) = X_k(i_k)(\alpha_{k-1}, \alpha_k) \tag{4.4}$$

for $1 \le \alpha_{k-1} \le r_{k-1}$, $1 \le i_k \le n_k$, $1 \le \alpha_k \le r_k$ and $1 \le k \le d$. This gives us the familiar form of the TT decomposition introduced in [86]:

$$\mathcal{X}(i_1, \ldots, i_d) = \sum_{\alpha_0=1}^{r_0} \cdots \sum_{\alpha_d=1}^{r_d} \mathcal{X}_1(\alpha_0, i_1, \alpha_1)\cdots \mathcal{X}_d(\alpha_{d-1}, i_d, \alpha_d) \tag{4.5}$$

The component tensors $\mathcal{X}_k \in \mathbb{R}^{r_{k-1}\times n_k\times r_k}$ for $1 \le k \le d$ are called the TT cores. The advantage of using the TT format in terms of storage complexity is immediate: given a tensor with maximum TT rank $r = \max\{r_1, \ldots, r_{d-1}\}$, storing the TT cores requires $O(dnr^2)$ memory. A large array of tensors relevant to practical applications can be well approximated by tensors with $r \ll \sqrt{n^d}$, which leads to significant storage savings.

4.2.2 TT Decomposition of a Full Tensor

In [86], the author presents an algorithm that computes the TT decomposition of a full tensor entirely using a series of SVD of auxiliary matrices. The k-th unfolding of the d-dimensional tensor $\mathcal{X}$ constructs a matrix from by grouping the first k indices of the tensor into rows, and the remaining indices into columns

$$X^{(k)}(\overline{i_1, \dots, i_k}, \overline{i_{k+1}, \dots, i_d}) = \mathcal{X}(i_1, \dots, i_d) \tag{4.6}$$

where

$$\overline{i_1, \dots, i_k} = i_1 + n_1(i_2 - 1) + \cdots + n_1 \cdots n_{k-1}(i_k - 1) \tag{4.7}$$

$$\overline{i_{k+1}, \dots, i_d} = i_{k+1} + n_{k+1}(i_{k+2} - 1) + \cdots + n_{k+1}n_{k+1} \cdots n_{d-1}(i_d - 1) \tag{4.8}$$

are long-indices in the column-major (Fortran) order. The ranks of these unfolding matrices are intimately related to the TT ranks of the corresponding tensor:

Theorem 4.1 (Exact TT Decomposition, [86]). *Let r_k be the rank of the k-th unfolding matrix of tensor $\mathcal{X}$ for $1 \leq k \leq d-1$. Then there exists a TT factorization with TT ranks $r_1, \dots, r_{d-1}$.*

In practice, data tensors are usually noisy which precludes the existence of an exact low-rank TT factorization. In this situation, we aim to find an approximate TT factorization of the form

$$\mathcal{X}(i_1, \dots, i_d) \approx \mathcal{Y}(i_1, \dots, i_d) = \sum_{\alpha_0=1}^{r_0} \cdots \sum_{\alpha_d=1}^{r_d} \mathcal{Y}_1(\alpha_0, i_1, \alpha_1) \cdots \mathcal{Y}_d(\alpha_{d-1}, i_d, \alpha_d) \tag{4.9}$$

and we aim to minimize the Frobenius error of the approximation

$$\|\mathcal{X} - \mathcal{Y}\|_F = \sqrt{\sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} [\mathcal{X}(i_1, \ldots, i_d) - \mathcal{Y}(i_1, \ldots, i_d)]^2} \tag{4.10}$$

Existence of such an approximation is established by the following result:

Theorem 4.2 (Approximate TT Decomposition, [85]). *Suppose the unfolding matrices of tensor $\mathcal{X}$ satisfies*

$$X^{(k)} = A_k + E_k, \quad \operatorname{rank}(A_k) = r_k, \quad \|E_k\|_F = \delta_k, \quad 1 \leq k \leq d-1. \tag{4.11}$$

Then there exists a tensor $\mathcal{Y}$ with inner TT ranks $r_1, \ldots, r_{k-1}$ satisfying

$$\|\mathcal{X} - \mathcal{Y}\|_F \leq \sqrt{\sum_{k=1}^{d-1} \delta_k^2}. \tag{4.12}$$

The proof of this theorem is constructive, and it outlines an algorithm for computing a TT format tensor $\mathcal{Y}$ that approximates a full tensor $\mathcal{X}$ via SVD given an relative tolerance τ satisfying

$$\|\mathcal{X} - \mathcal{Y}\|_F \leq \tau \|\mathcal{X}\|_F. \tag{4.13}$$

This procedure, known as the TT-SVD, is presented in Algorithm 4.1. This is the main algorithm we use for compressing data tensors.

4.2.3 TT-Rounding of a Sub-Optimal TT Tensor

As we perform fast arithmetic on TT-compressed arrays such as sums, products and concatenation, TT ranks are often over-estimated, leading to sub-optimal representations requiring more storage than necessary. Fortunately, we can recompress these tensors by applying ma-

Algorithm 4.1 TT-SVD

```
Input: $d$-dimensional tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, relative tolerance $\tau$
Output: Approximation $\mathcal{Y}$ with TT cores $\mathcal{Y}_1, \dots, \mathcal{Y}_d$ satisfying $\|\mathcal{X} - \mathcal{Y}\|_F \leq \tau \|\mathcal{X}\|_F$
 1: Compute truncation parameter $\delta \leftarrow \tau \|\mathcal{X}\|_F / \sqrt{d-1}$
 2: Initialize $M \leftarrow \mathsf{reshape}(\mathcal{X}, [n_1, n_2 \cdots n_d])$
 3: Initialize $r_0 \leftarrow 1$
 4: Initialize $r_d \leftarrow 1$
 5: for $k = 1, \dots, d-1$ do
 6:     $U, \Sigma, V \leftarrow \mathsf{svd}_\delta(M)$
 7:     $r_k \leftarrow \mathsf{rank}(\Sigma)$
 8:     $\mathcal{Y}_k \leftarrow \mathsf{reshape}(U, [r_{k-1}, n_k, r_k])$
 9:     $M \leftarrow \mathsf{reshape}(\Sigma V^\top, [r_k n_{k+1}, n_{k+2} \cdots n_d])$
10: end for
11: $\mathcal{Y}_d \leftarrow \mathsf{reshape}(M, [r_{d-1}, n_d, r_d])$
```

trix factorizations to each of the TT cores [86]. This TT-rounding process constructs from input tensor $\mathcal{X}$ and relative tolerance τ a new tensor $\mathcal{Y}$ satisfying

$$\|\mathcal{X} - \mathcal{Y}\|_F \leq \tau \|\mathcal{X}\|_F \tag{4.14}$$

This is achieved in two sweeps:

- First, the cores are orthogonalized to remove any linear dependence among its slices.
- Then the cores are recompressed using SVD to further optimize the ranks to the extent allowed by the relative tolerance.

The detailed steps are reproduced in Algorithm 4.2. Computation of the Frobenius norm of the input tensor $\mathcal{X}$ in TT-format can be achieved solely from the tensor cores, and we do not need to uncompress the full tensor [86]. This process has computational complexity of $O(dnr^3)$, where r is the maximal TT rank of the sub-optimal tensor.

Algorithm 4.2 TT-rounding

```
Input: Tensor $\mathcal{X}$ with TT cores $\mathcal{X}_1, \ldots, \mathcal{X}_d$ and ranks $r_0, r_1, \ldots, r_{d-1}, r_d$, relative tolerance $\tau$
Output: TT cores $\mathcal{Y}_1, \ldots, \mathcal{Y}_d$ of tensor $\mathcal{Y}$ with optimal ranks satisfying $\|\mathcal{X} - \mathcal{Y}\|_F \leq \tau \|\mathcal{X}\|_F$
 1: Compute truncation parameter $\delta \leftarrow \tau \|\mathcal{X}\|_F / \sqrt{d-1}$
 2: Initialize cores $\mathcal{Y}_k \leftarrow \mathcal{X}_k$ for $1 \leq k \leq d$
 3: for $k = d, d-1, \ldots, 2$ do
 4:     $M \leftarrow \mathsf{reshape}(\mathcal{Y}_k, [r_{k-1}, n_k r_k])$
 5:     $Q, R \leftarrow \mathsf{qr}(M^\top)$
 6:     $r'_{k-1} \leftarrow \mathsf{rank}(R)$
 7:     $\mathcal{Y}_k \leftarrow \mathsf{reshape}(Q^\top, [r'_{k-1}, n_k, r_k])$
 8:     $M \leftarrow \mathsf{reshape}(\mathcal{Y}_{k-1}, [r_{k-2} n_{k-1}, r_{k-1}])$
 9:     $\mathcal{Y}_{k-1} \leftarrow \mathsf{reshape}(M R^\top, [r_{k-2}, n_{k-1}, r'_{k-1}])$
10:     $r_{k-1} \leftarrow r'_{k-1}$
11: end for
12: for $k = 1, \ldots, d-1$ do
13:     $M \leftarrow \mathsf{reshape}(\mathcal{Y}_k, [r_{k-1} n_k, r_k])$
14:     $U, \Sigma, V \leftarrow \mathsf{svd}_\delta(M)$
15:     $r'_k \leftarrow \mathsf{rank}(\Sigma)$
16:     $\mathcal{Y}_k \leftarrow \mathsf{reshape}(U, [r_{k-1}, n_k, r'_k])$
17:     $M \leftarrow \mathsf{reshape}(\mathcal{Y}_{k+1}, [r_k, n_{k+1} r_{k+1}])$
18:     $\mathcal{Y}_{k+1} \leftarrow \mathsf{reshape}(\Sigma V^\top M, [r'_k, n_{k+1}, r_{k+1}])$
19:     $r_k \leftarrow r'_k$
20: end for
```

4.3 Tensor-Train for Data Compression

We now build upon the basic aspects of the TT factorization introduced in the last section, and adapt them for compressing scientific simulation data.

4.3.1 Tensorization

One of the main advantages of the TT decomposition is the linear scaling of storage complexity with the dimensionality of the data tensor. This makes it ideal for compressing very high-dimensional tensors. We take full advantage of this aspect of TT factorization by artificially and systematically increasing the dimensionality of the data (a.k.a. tensorizing the data) before constructing the decomposition. This approach is motivated by hierarchical domain decomposition techniques used in partial differential equation solvers, especially in the context of boundary integral equation (BIE) methods [39, 48, 50, 49, 25].

Example: Compression of Integral Kernel Matrix

We first demonstrate how domain decomposition can be used to compress an integral kernel matrix more efficiently. Let us consider the following integral operator evaluation

$$f(x) = \int_0^1 k_\delta(x,y)\sigma(y)dy \tag{4.15}$$

where the integration kernel is given by

$$k_\delta(x,y) = \log \frac{1}{|x-y|+\delta}, \quad \delta > 0 \tag{4.16}$$

and $\sigma : [0,1] \to \mathbb{R}$ is some density function. Discretizing (4.15) via a quadrature rule $\{(t_i, w_i) : 1 \le i \le N\}$ with $N = 2^d$ nodes for some $d \ge 1$ leads to the linear system

$$f = K_\delta \mathrm{diag}(w)\sigma \tag{4.17}$$

with $\sigma = [\sigma(t_j)]_{j=1}^N$, $w = [w_j]_{j=1}^N$, $K_\delta = [k_\delta(t_i,t_j)]_{i,j=1}^N$ and $f \approx [f(t_i)]_{i=1}^N$. Thus, evaluating f for a given density σ using (4.15) boils down to a dense matrix multiplication, which naively takes $O(N^2)$ operations.

However, we can take advantage of the structure of the matrix K_δ imposed by the underlying kernel function k_δ. In particular, when the source and target points x and y are far away, we can evaluate accurately within machine precision $k_\delta(x,y)$ via a low-rank approximation of the kernel. This observation forms the basis for many fast algorithms in the integral equations research community, such as tree-codes and fast multipole methods [39]. At their heart, these algorithms construct a hierarchical domain decomposition and take advantage of the low-rank structure of the far-interactions to construct $O(N)$ algorithms for

Relative Tolerance	Metric	Tensorization Level		
		$l = 6$	$l = 8$	$l = 10$
$\tau = 10^{-2}$	Compression, ρ	1.9×10^{2}	1.4×10^{3}	1.6×10^{3}
	Error, ϵ	4.2×10^{-1}	1.5×10^{0}	7.4×10^{-1}
$\tau = 10^{-5}$	Compression, ρ	1.2×10^{2}	6.0×10^{2}	6.5×10^{2}
	Error, ϵ	2.3×10^{-3}	1.1×10^{-3}	1.1×10^{-3}
$\tau = 10^{-8}$	Compression, ρ	8.3×10^{1}	3.6×10^{2}	3.9×10^{2}
	Error, ϵ	2.4×10^{-7}	3.5×10^{-7}	3.5×10^{-7}
$\tau = 10^{-11}$	Compression, ρ	6.9×10^{1}	2.7×10^{2}	3.0×10^{2}
	Error, ϵ	1.1×10^{-9}	1.1×10^{-9}	1.1×10^{-9}
$\tau = 10^{-14}$	Compression, ρ	5.9×10^{1}	1.8×10^{2}	1.9×10^{2}
	Error, ϵ	9.2×10^{-12}	1.0×10^{-11}	2.3×10^{-11}

Table 4.1: Compression ratios ρ and reconstruction errors $\epsilon = \|K_\delta - \hat{K}\|_2$ in constructing a TT approximations $\hat{K}$ of the $2^d \times 2^d$ integral kernel matrix K_δ with $d = 10$ and $\delta = 10^{-5}$ at various tensorization levels l and TT-SVD relative tolerance τ. We note that the compression ratio increases as we increase the level of tensorization and decreases as we reduce TT-SVD relative tolerance and ask for a more accurate reconstruction.

matrix-vector multiplications.

The idea of hierarchical domain decomposition can also be used to increase the dimensionality of the kernel matrix K_δ. Let $(i_1^{(l)}, \ldots, i_l^{(l)})$ be a level l multi-index with $i_k^{(l)} \in \{1, 2\}$ for $1 \leq k \leq l-1$ and $i_l^{(l)} \in \{1, \ldots, 2^{d-l+1}\}$; then the row-major unwrapping defines a map between this binary multi-index and a linear index $i \in \{1, \ldots, 2^d\}$:

$$i = \overline{i_1^{(l)}, \ldots, i_l^{(l)}} := \sum_{k=1}^{l} 2^{l-k}(i_k^{(l)} - 1) + 1 \tag{4.18}$$

This allows us to recast the $2^d \times 2^d$ kernel matrix with entries $K_\delta(i, j)$ as a l dimensional tensor of size $4 \times \cdots \times 4 \times 4^{d-l+1}$ with entries $\mathcal{K}_\delta^{(l)}(\overline{i_1^{(l)}, j_1^{(l)}}, \ldots, \overline{i_l^{(l)}, j_l^{(l)}})$. Note that we have interlaced the multi-indices of the rows and columns of K_δ such that each index $\overline{i_k, j_k}$ describes the same level in the domain decomposition hierarchy.

In Table 4.1 we demonstrate the compression ratios and reconstruction errors when compressing this tensorized kernel matrix at various tensorization levels l. We note that as we increase tensorization level up to $l = d$, the compression ratio increases.

Application to DEM Datasets: Tensorization with Space Filling Curves

In the context of compressing data from differential and integral equation discretization on structured grids, established hierarchical domain decomposition leads to a natural tensorization scheme, as illustrated by the 1D example above.

DEM simulation data, on the other hand, typically consists of relevant physical properties (e.g. velocity, force, stress) associated to individual particles or collision sites; additionally, these locations will shift as the simulation progresses. It is thus not immediately obvious how to tensorize this dataset.

Recall from Chapter 2 how Morton order was used to impose locality among the particles in a DEM simulation; we can utilize the same idea here. We first sort the particles in a DEM simulation dataset using the Morton order. Then the particles that are nearby will, for the most part, also end up close by in in 3D simulation space. We can expect the properties (such as velocities) for these particles to be similar/correlated, especially if the particles are dense in space. We can then construct a hierarchy of dimensions using a simple binary tree structure on this linearly ordered list of particles. Note that the time dimension is already linear, and we can impose a similar hierarchical partitioning to further increase the dimensionality of the dataset. With this new high-dimensional dataset, we can expect to exploit the similarities in particle properties at different scales, leading to better compressibility.

4.3.2 Streaming Data Compression

One of the disadvantages of the direct TT-SVD algorithm presented in Algorithm 4.1 is the SVD itself. Given a $m \times n$ matrix with $m \geq n$, computing its full SVD decomposition (including the singular vectors) costs $O(m^2n + mn^2 + n^3)$ floating point operations. Thus, for a tensor with large sizes, or one with high dimensionality, performing the SVD itself becomes a challenge.

Algorithm 4.3 TT-Concatenation along Existing Data Dimension

```
Input: Data tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ with $n_1 \gg \max\{n_2, \ldots, n_d\}$
Output: TT cores $\mathcal{X}_1, \ldots, \mathcal{X}_d$ approximating input tensor $\mathcal{X}$
1: Partition $\mathcal{X}$ into two pieces $\mathcal{X}^{(i)} \in \mathbb{R}^{n_1^{(i)} \times n_2 \times \cdots \times n_d}$, $i \in \{a, b\}$, $n_1^{(a)} + n_1^{(b)} = n_1$ such that
       (4.19)  $\mathcal{X}^{(a)}(i_1, \ldots, i_d) = \mathcal{X}(i_1, i_2, \ldots, i_d) \quad \text{for} \quad 1 \le i_1 \le n_1^{(a)}$
       (4.20)  $\mathcal{X}^{(b)}(i_1, \ldots, i_d) = \mathcal{X}(i_1 + n_1^{(a)}, i_2, \ldots, i_d) \quad \text{for} \quad 1 \le i_1 \le n_1^{(b)}$
2: Compute TT decompositions $\mathcal{X}^{(i)}(i_1, \ldots, i_d) = X_1^{(i)}(i_1) \cdots X_d^{(i)}(i_d)$ of the sub-tensors
3: Construct zero-padded tensors $\widehat{\mathcal{X}}^{(i)} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ in the TT format by modifying the first core
       (4.21)  $\widehat{X}_1^{(a)}(i_1) = \begin{cases} X_1^{(a)}(i_1) & \text{if} \quad 1 \le i_1 \le n_1^{(a)} \\ 0 & \text{if} \quad n_1^{(a)} + 1 \le i_1 \le n_1 \end{cases}$
       (4.22)  $\widehat{X}_1^{(b)}(i_1) = \begin{cases} 0 & \text{if} \quad 1 \le i_1 \le n_1^{(a)} \\ X_1^{(b)}(i_1 - n_1^{(a)}) & \text{if} \quad n_1^{(a)} + 1 \le i_1 \le n_1 \end{cases}$
4: By construction, we have $\mathcal{X} = \widehat{\mathcal{X}}^{(a)} + \widehat{\mathcal{X}}^{(b)}$; construct the TT cores of the full tensor as
       (4.23)  $X_1(i_1) = \begin{bmatrix} \widehat{X}_1^{(a)}(i_1) & \widehat{X}_1^{(b)}(i_1) \end{bmatrix}$
       (4.24)  $X_k(i_k) = \begin{bmatrix} X_k^{(a)}(i_k) & \\ & X_k^{(b)}(i_k) \end{bmatrix} \quad \text{for} \quad 1 < k < d$
       (4.25)  $X_d(i_d) = \begin{bmatrix} X_d^{(a)}(i_d) \\ X_d^{(b)}(i_d) \end{bmatrix}$
5: Optimize the TT ranks using TT-rounding
```

Many simulations have a natural time-like parameter, and the dataset grows along this dimension as the simulation progresses. With this in mind, we can reduce the computational complexity of TT-SVD by dividing the tensor along that dimension and compressing the resulting sub-tensors individually. We outline the details of this approach in Algorithm 4.3 where the original tensor is divided into two parts; a k-part generalization is straightforward. This approach works equally well in splitting up datasets along spatial dimensions.

However, this approach is ill-suited for tensorized dimensions, where it is more advantageous to create new dimensions instead of growing an existing dimension. We therefore

Algorithm 4.4 TT-Concatenation along New Data Dimension

```
Input: Data tensors $\mathcal{X}^{(a)}, \mathcal{X}^{(b)} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ with same size
Output: TT cores $\mathcal{X}_1, \ldots, \mathcal{X}_d$ approximating $\mathcal{X} \in \mathbb{R}^{n_1\times\cdots n_d\times 2}$ with
    (4.26)    $\mathcal{X}(i_1,\ldots,i_d,1) = \mathcal{X}^{(a)}(i_1,\ldots,i_d)$
    (4.27)    $\mathcal{X}(i_1,\ldots,i_d,2) = \mathcal{X}^{(b)}(i_1,\ldots,i_d)$
1: Compute TT decompositions $\mathcal{X}^{(i)}(i_1,\ldots,i_d) = X_1^{(i)}(i_1)\cdots X_d^{(i)}(i_d)$ for $i \in \{a,b\}$
2: Define first $d$ TT cores
    (4.28)    $X_1(i_1) = \begin{bmatrix} X_1^{(a)}(i_1) & X_1^{(b)}(i_1) \end{bmatrix}, \quad X_k(i_k) = \begin{bmatrix} X_k^{(a)}(i_k) & \\ & X_k^{(b)}(i_k) \end{bmatrix}$ for $2 \le k \le d$
3: Construct the last TT core
    (4.29)    $X_{d+1}(i_{d+1}) = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \text{for} \quad i_{d+1} = 1 \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{for} \quad i_{d+1} = 2 \end{cases}$
   By direct computation, we have $\mathcal{X}(i_1,\ldots,i_{d+1}) = X_1(i_1)\cdots X_{d+1}(i_{d+1})$
4: Optimize the TT ranks using TT-rounding
```

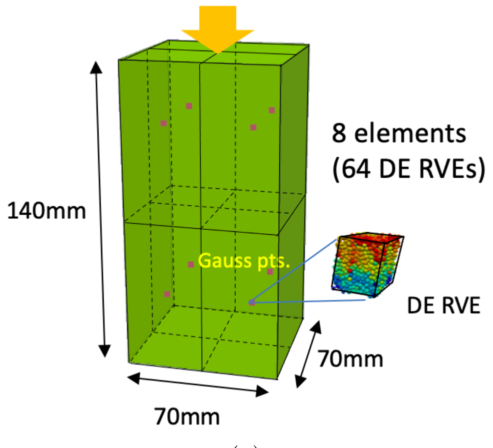
adapt our approach to suit this situation in Algorithm 4.4. We once again address combining two tensors, but generalization to an arbitrary number of tensors is straightforward.

4.4 Numerical Results

We used two distinct types of datasets in testing out the capabilities of our compression framework:

- **Raw DEM Datasets:** These datasets were constructed by running various DEM simulations and saving the particle-level data (i.e. position, velocity etc. of all the particles in the simulation). We compress the data from the following scenarios:

  SEDIMENT We release $n = 16384$ spherical particles of radius $r = 0.01$ m and density



(a)

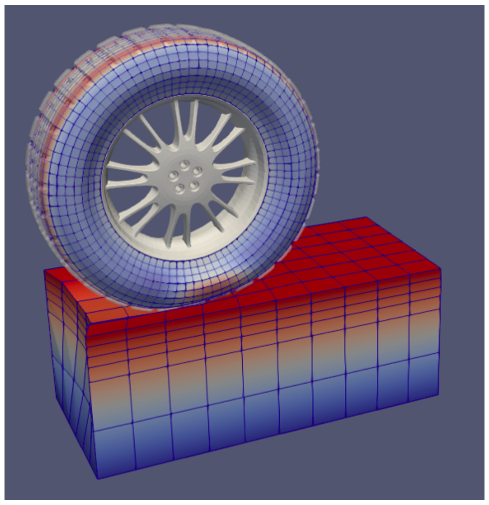

(b)

Figure 4.1: Setup for the stress-strain datasets from multi-scale coupled finite element/discrete element (FE-DE) simulations conducted by Hiroyuki Sugiyama and his research group at the University of Iowa. (a) Triaxial soil compression test models a column of soil using 8 elements; the sides of the soil are confined by a constant pressure and a load is applied to the top. (b) Soil-wheel simulation models a patch of soil using $10 \times 6 \times 6$ elements and the wheel moves on top of the soil at a constant speed. In both cases, reference volume elements (RVEs) are placed at 8 Gaussian quadrature points inside each element. Stress, strain and tangent moduli are computed at these RVEs from individual DEM simulations.

$\rho = 2500$ kg/m^3 from rest inside a rectangular box $[-1, 1] \times [-1, 1] \times [-\frac{1}{2}, \frac{1}{2}]$, and allow them to free-fall under gravity ($g = 9.81$ m/s^2). In the course of the simulation, the particles collide with each other and the inner walls of the box: we resolve the collisions using the complementarity formulation of Coulomb friction model [30]. We record the location, angular orientation, velocity and angular velocity for each particles for $m = 1024$ timesteps with stepsize $\Delta t = 1$ ms. The full dataset is approximately 8 GB when saved to disk in ASCII format.

- **Stress-Strain Response Datasets:** These datasets were generated from running a multi-scale model of soil in various simulation scenarios. The soil is modeled using a finite element mesh, and within each mesh element, the stress-strain response of the soil is computed by running small-scale DEM simulations ($\sim$ 1000 particles) at Gaussian quadrature nodes (a.k.a. reference volume elements, RVEs). At each timestep, we save the stress, strain, incremental stress, incremental strain and tangent moduli of each RVE. These datasets are generated and provided to us by Professor Hiroyuki Sugiyama and his research group at the University of Iowa.

TRIAXIAL A column of soil is placed in a chamber confined by some fluid, and load presses the column from the top. The fluid pressure is kept at a constant value (the confining pressure). As the load at the top of the chamber is increased incrementally, we compute the stress-strain data at 64 RVEs inside the soil column (see Figure 4.1a). This creates a three-dimensional dataset, the dimensions corresponding to the load on the top, different RVEs and various components of the recorded data.

SOILWHEEL A tire is rolled on top of a soil patch. The tire is modeled using the finite element method (FEM), whereas for the soil we use the multiscale model with a $10 \times 6 \times 6$ FEM grid, the RVEs are placed at a $2 \times 2 \times 2$ Gaussian quadrature grid within each element. Soil response data is generated for 602 parameter values as the tire rolls across the soil patch with a constant speed (see Figure 4.1b).

In each of the experiments, we use the following quantities as a measure of the quality of compression:

- **Compression Ratio:** It is the ratio of the number of parameters needed to represent the dataset in full vs. that in the TT format. If a $n_1 \times \cdots \times n_d$ tensor can be represented in TT format with TT ranks $\{r_1, \ldots r_{d-1}\}$, then we define

$$\text{compression ratio} = \frac{n_1 \cdots n_d}{\sum_{k=1}^{d} r_{k-1} n_k r_k} \tag{4.30}$$

  with $r_0 = r_d = 1$ by convention.

- **Maximal TT Rank:** It is given by $r = \max\{r_1, \ldots, r_{d-1}\}$ and is another measure of how compressible the dataset is; note that

$$\text{compression ratio} \geq \frac{1}{r^2}\frac{n_1 \cdots n_d}{n_1 + \cdots + n_d} \tag{4.31}$$

- **Normalized RMSE:** It is a measure of the error incurred when reconstructing the dataset from its TT representation. For a series $\{y_1, \ldots, y_n\}$ and its reconstruction $\{\hat{y}_1, \ldots, \hat{y}_n\}$, the normalized RMSE is defined as

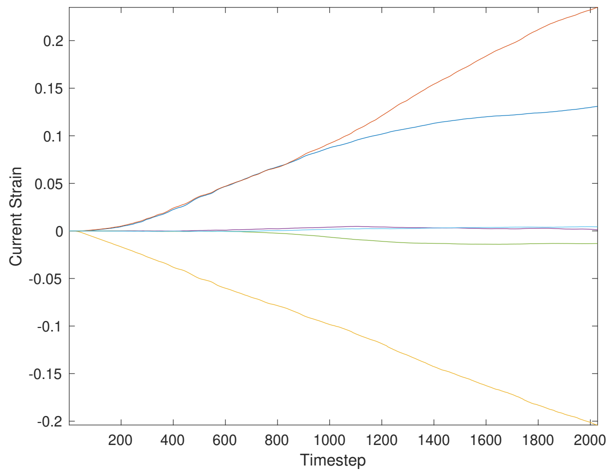
$$\text{nRMSE} = \frac{1}{y_{\max} - y_{\min}}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.32}$$

  with $y_{\min} = \min\{y_1, \ldots, y_n\}$ and $y_{\max} = \max\{y_1, \ldots, y_n\}$.

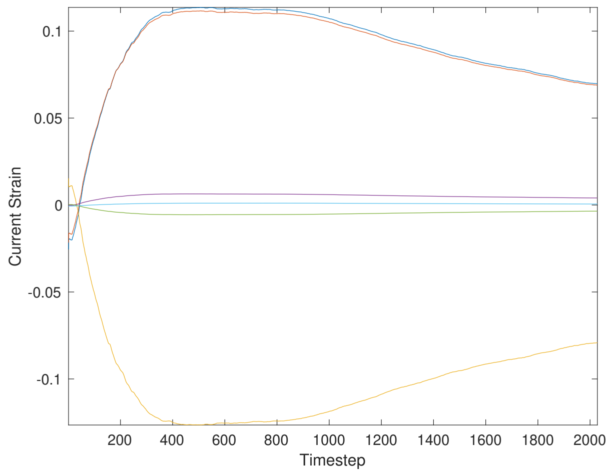
4.4.1 General Observations

Note that all the datasets introduced above are a mixture of different variables. For instance, the soil-response datasets consist of stress, strain and tangent moduli, and there are huge differences on the magnitudes of the data—the strain is of $O(10^{-1})$ magnitude while the tangent moduli are of $O(10^7)$ magnitude. This is generally true for a majority of scientific simulation data; they consist of several variables taking values in different orders of magnitudes. Compressing this type of heterogeneous datasets as a single tensor typically leads to inconsistencies in the reduced representations of the constituent variables. This is demonstrated in Figure 4.2, where we compress the TRIAXIAL dataset as a single tensor using the TT factorization at relative accuracy level $\tau = 10^{-1}$. In the reconstruction of the current strain data, we can see features from the tangent moduli data creeping in and drastically altering the recovered behavior. This observation led us to compressing different variables in a single scientific dataset as separate tensors in our numerical experiments.

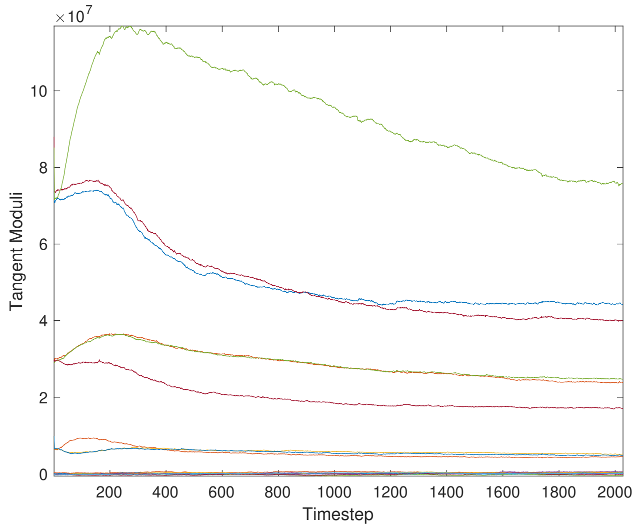
We also observe a close relationship between the level of correlation in a dataset, and the compressibility of that dataset. We use time-dimension autocorrelation as a measure of



(a) Original Current Strain

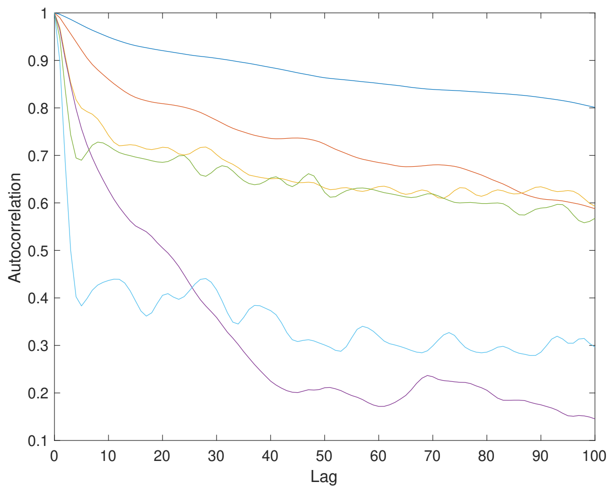


(b) Recovered Current Strain

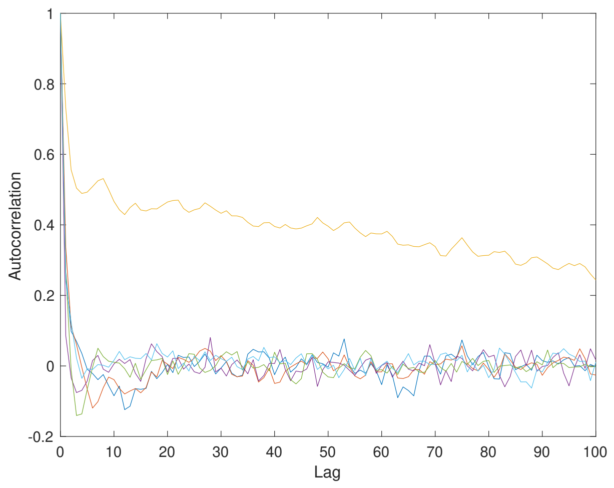


(c) Original Tangent Moduli

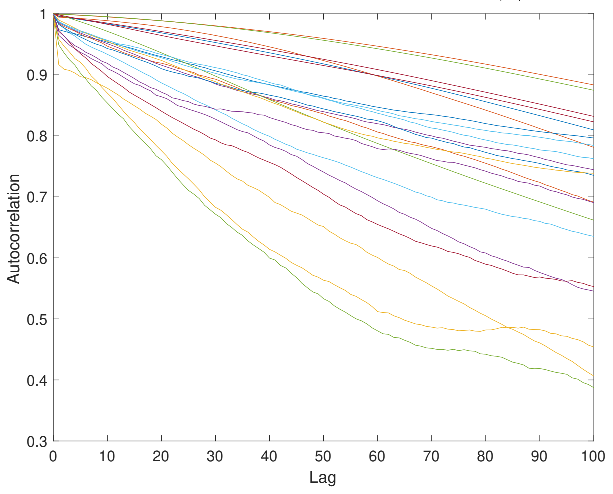
Figure 4.2: Inconsistencies in the recovered current strain data from TRIAXIAL dataset when all of it is compressed as a single tensor at relative accuracy level $\tau = 10^{-1}$. Panel (a) plots the time evolution of the six components of current strain from a single RVE in the dataset and panel (b) is its reconstruction. The significant deviations between these are obvious from these two plots—we can see general features from the tangent moduli timeseries, plotted in panel (c), creep in this reconstruction (e.g. the peak near timestep 400).



(a) Incremental Strain



(b) Incremental Stress



(c) Tangent Moduli

Figure 4.3: Time-dimension autocorrelation factors for different variables from a single RVE of the TRIAXIAL dataset. We note that as the lag increases, the autocorrelation for incremental stress drops sharply to zero, indicating minimal correlation in the timeseries. On the other hand, the autocorrelation for tangent moduli decays slowly, implying highly correlated timeseries data; correlation for incremental strain is somewhere in between. Compressibility of these variables follow the same trend: the compression ratios for incremental stress, incremental strain and tangent moduli are approximately 8.2×10^{-1}, 8.4×10^{0} and 1.3×10^{3}, respectively.

correlation in the dataset: given a timeseries $\{x_i : 1 \leq i \leq n\}$ the k-lag autocorrelation is defined as

$$\text{AutoCorr}(k) = \frac{\sum_{i=1}^{n-k}(x_i - \bar{x})(x_{i+k} - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \quad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.33}$$

We note that AutoCorr(0) = 1 for any timeseries. Additionally, for $k \geq 1$, a fast decay in the autocorrelation is indicative of uncorrelated data. We also define the integrated autocorrelation as a scalar measure for level of correlation

$$\text{IAC} = 1 + 2\sum_{k=1}^{\infty} \text{AutoCorr}(k) \tag{4.34}$$

In Figure 4.3 we demonstrate the link between autocorrelation and compression ratios for the TRIAXIAL dataset. We note that incremental stress has smaller autocorrelation ($4.8 \times 10^{-1} \lesssim \text{IAC} \lesssim 7.9 \times 10^{1}$) when compared against tangent moduli ($1.2 \times 10^{2} \lesssim \text{IAC} \lesssim 1.9 \times 10^{2}$). Correspondingly, incremental stress is much less compressible (compression ratio $\approx 8.2 \times 10^{-1}$) than tangent moduli (compression ratio $\approx 1.3 \times 10^{3}$).

4.4.2 Tensorization for Raw DEM Simulation Data

We now explore the advantages of tensorization in compressing the SEDIMENT dataset. To increase the dimensionality of the dataset, we create $\log_2 n = 14$ hierarchical indices out of the list of particles sorted using the Morton ordering, and $\log_2 m = 10$ hierarchical indices from the time snapshots.

We compress the SEDIMENT datasets 32 timesteps at a time, sorting the particles based on the first and seventeenth timesteps of each chunk and retaining that order for the subsequent timesteps. In Figure 4.4, we compare the compression ratios of these data chunks when they are compressed as the natural three-dimensional and tensorized 20-dimensional tensors as they evolve in time. We see that when the particles initially free-fall, the datasets are highly compressible; additionally the compression ratio of the tensorized dataset is far

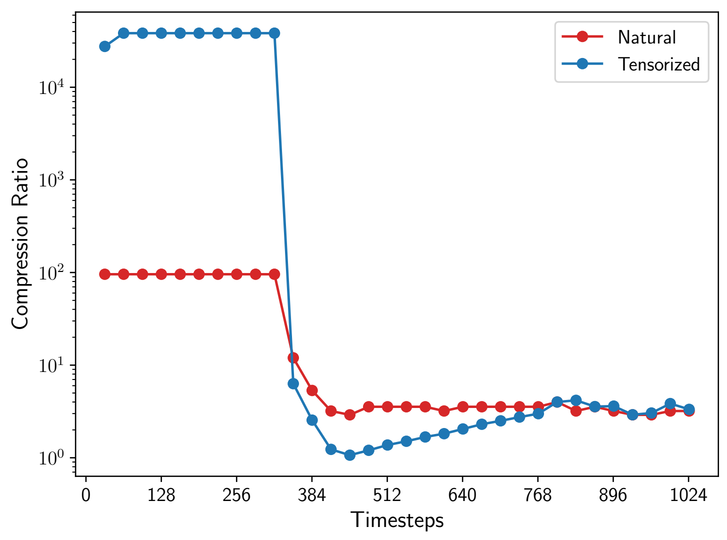


Figure 4.4: Compression of the velocity variable from the SEDIMENT dataset; the dataset is compressed 32 timesteps at a time. The blue dots correspond to compression ratios of the data chunks in their natural three dimensions, and the red dots correspond to the compression ratios with tensorized dimensions. Initially, as the particles are in free-fall, the compression ratio is high. But as the particles start to collide with each other and the simulation box, the compression ratio drops. Eventually, as the particles settle down, the compression ratio again slowly increases.

superior to that of the natural dataset. But as the particles start to collide with the walls of the box and each other (around the timestep 300), the compression ratio drops drastically. However, as the particles settle down, the compression ratios start increasing again.

4.4.3 Streaming Compression for Raw DEM Simulation Data

We now apply the streaming compression setup to the compressed chunks of the SEDIMENT dataset. We combine the TT compressed datasets from the previous section following Algorithm 4.4. In Figure 4.5 we plot the compression ratios of these incrementally constructed compressed datasets. We observed the same compressibility features as in the previous section: an initially high compression ratio, followed by a drop when the particles collide and the data is nonsmooth, and finally an increase in the compression ratio when the particles settle down.

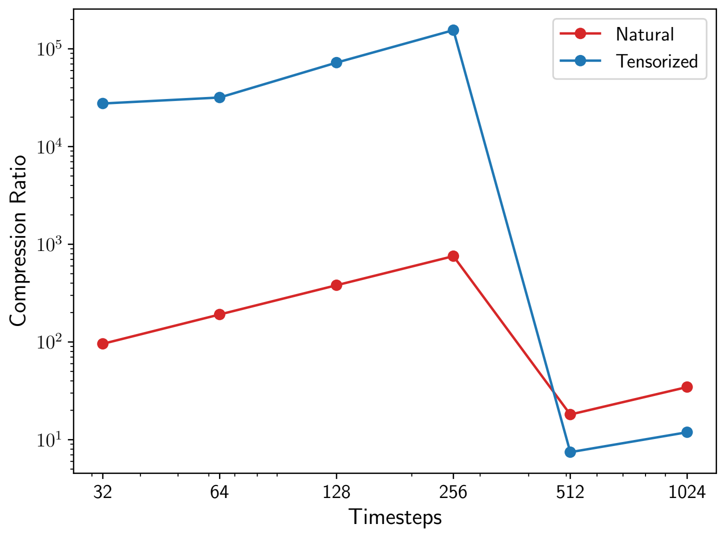


Figure 4.5: Streaming compression of the velocity variable from the SEDIMENT dataset; the dataset is first compressed 32 timesteps at a time and the compressed datasets are then concatenated in the TT format. The blue dots correspond to compression ratios of the data chunks in their natural three dimensions, and the red dots correspond to the compression ratios with tensorized dimensions. Initially, as the particles are in free-fall, the compression ratio is high. But as the particles start to collide with each other and the simulation box, the compression ratio drops. Eventually, as the particles settle down, the compression ratio again slowly increases.

4.4.4 Compression of Stress-Strain Data

In this section, we present the data compression results from the SOILWHEEL dataset. In Table 4.2, we record the maximum TT rank, compression ratios and normalized RMSEs from compressing the different variables in the dataset at different relative accuracy levels. We see that the compressibility of the different variables varies drastically: at $\tau = 10^{-1}$ accuracy level, the compression ratio for tangent moduli is 3.0×10^4 whereas that for incremental stress is only 2.7.

4.5 Conclusions

In this chapter, we introduced the tensor-train decomposition as a tool for compressing scientific data, focusing on DEM simulation outputs. We were able to demonstrate high compression ratios for both raw and derived DEM datasets. We were able to achieve com-

Dataset	Metric	Relative Tolerance		
		$\tau = 10^{-1}$	$\tau = 10^{-2}$	$\tau = 10^{-3}$
Current Strain	Maximum Rank	41	163	301
	Compression Ratio	6.6×10^{2}	8.2×10^{1}	2.2×10^{1}
	Normalized RMSE	1.1×10^{-2}	9.5×10^{-4}	8.8×10^{-5}
Current Stress	Maximum Rank	128	726	1910
	Compression Ratio	1.0×10^{2}	3.2×10^{0}	8.3×10^{-1}
	Normalized RMSE	4.3×10^{-3}	3.2×10^{-4}	3.1×10^{-5}
Incremental Strain	Maximum Rank	203	654	1214
	Compression Ratio	2.6×10^{1}	3.3×10^{0}	1.3×10^{0}
	Normalized RMSE	2.3×10^{-3}	2.0×10^{-4}	2.0×10^{-5}
Incremental Stress	Maximum Rank	795	2097	2680
	Compression Ratio	2.7×10^{0}	7.6×10^{-1}	5.8×10^{-1}
	Normalized RMSE	1.9×10^{-3}	1.7×10^{-4}	1.2×10^{-5}
Tangent Moduli	Maximum Rank	8	270	3323
	Compression Ratio	3.0×10^{4}	1.4×10^{2}	1.4×10^{0}
	Normalzied RMSE	1.2×10^{-2}	1.4×10^{-3}	9.7×10^{-5}

Table 4.2: Maximum TT ranks, compression ratios and normalized RMSEs when compressing the different variables in the SOILWHEEL dataset at various tolerance levels. We see that the different variables have differing compression ratios (e.g. the tangent moduli are much more compressible than incremental stress). Also, the compression ratios drop as we demand more and more accurate reconstructions by reducing the relative tolerance τ.

pression ratios of $O(10^4)$ on datasets of size $O(1)$ GB, reducing them to mere $O(100)$ KB datasets while keeping $O(10^{-1})$ relative accuracy. We expect these compression ratios to persist, and potentially even increase, for larger datasets as we have more redundancies to exploit. These high levels of compression enable us to store simulation datasets at a fraction of the storage cost of the full dataset.

CHAPTER 5

Efficient MCMC Sampling for Bayesian Matrix Factorization by Breaking Posterior Symmetries

Preamble. In this chapter, we explore how to recover missing entries of matrices. This is a widely studied problem with numerous practical applications. We focus on recovering low-rank data matrices using a fully Bayesian inference framework based on Markov-chain Monte Carlo (MCMC) sampling methods. We propose a simple modification that provably breaks the posterior symmetries observed in the standard setup, and leads to better sampling performance and lower reconstruct errors. This is joint work with Hadi Salehi and Alex Gorodetsky, and is currently under review [31].

5.1 Introduction

The *matrix completion* problem seeks to use partial observations of a matrix to estimate missing entries. Formally, let $X \in \mathbb{R}^{m\times n}$ be our target data matrix, and let $\Lambda \subseteq \{1,\dots,m\}\times\{1,\dots,n\}$ be a set of matrix indices where the matrix element is observed:

$$y = \mathcal{P}_\Lambda(X) + \eta. \tag{5.1}$$

Here, $\mathcal{P}_\Lambda : \mathbb{R}^{m\times n} \to \mathbb{R}^{|\Lambda|}$, $|\Lambda|$ being the cardinality of the set Λ, is the linear projection map

$$\mathcal{P}_\Lambda(X) = (x_\lambda : \lambda = (\lambda_1, \lambda_2) \in \Lambda), \tag{5.2}$$

and $\eta \in \mathbb{R}^{|\Lambda|}$ is a vector of additive noises. Our goal is then to recover the matrix X from the observations y. This problem commonly arises in many practical applications such as recommender system design [104], drug-target interaction prediction [110, 116], image inpainting [42, 63], social network topology recovery [70] and sensor localization [109].

5.1.1 Related Works

This matrix problem is naturally ill-posed, and obtaining robust and accurate solutions requires imposing additional *regularity* conditions on the underlying data matrix such as sparsity or low-rank structures. In this paper, we consider the problem of low-rank matrix completion, where we might attempt to recover the matrix by nuclear norm minimization

$$\underset{X \in \mathbb{R}^{m \times n}}{\text{minimize}} \quad \|X\|_* \quad \text{subject to} \quad \|y - \mathcal{P}_\Lambda(X)\| \leq \delta, \tag{5.3}$$

for some constant $\delta \geq 0$ that depends on the level of noise. Note that the nuclear norm is the convex relaxation of rank of a matrix [18], hence the objective in the optimization problem naturally encourages low-rank structure of the reconstruction. In [18], [19] and [17] the authors establish that, under mild assumptions about certain incoherence properties of X, solving (5.3) leads to accurate recovery of the underlying data matrix with surprisingly few observations. In fact, in absence of noise (that is, when $\eta = 0$ and $\delta = 0$) exact recovery is possible with high probability.

The standard solution method for optimization (5.3) is semi-definite programming, which is expensive when the matrix sizes m and n are large. In this setup, it is advantageous to explicitly use the low-rank factorization

$$X = AB^\top, \quad A \in \mathbb{R}^{m \times r}, \quad B \in \mathbb{R}^{n \times r}, \quad r \ll \min\{m, n\}, \tag{5.4}$$

and solve the optimization problem

$$\underset{A\in\mathbb{R}^{m\times r}, B\in\mathbb{R}^{n\times r}}{\text{minimize}} \quad \|A\|_F^2 + \|B\|_F^2 \quad \text{subject to} \quad \|y - \mathcal{P}_\Lambda(AB^\top)\| \leq \delta, \tag{5.5}$$

We can show that the optimization problems (5.3) and (5.5) are equivalent as long as the estimated rank r is chosen to be larger than the true rank [93]. We also consider the unconstrained version of (5.5):

$$\underset{A\in\mathbb{R}^{m\times r}, B\in\mathbb{R}^{n\times r}}{\text{minimize}} \left\|y - \mathcal{P}_\Lambda(AB^\top)\right\|_2^2 + \frac{\omega}{2}\|A\|_F^2 + \frac{\omega}{2}\|B\|_F^2. \tag{5.6}$$

Generally speaking, these optimization-based approaches attempt to minimize the distance between the observed entries and their corresponding low-rank predictions while regularizing over the low-rank factors, and they serve as the base of a wide class of methods for low-rank matrix recovery [100, 76, 29].

In this paper we instead focus on probabilistic approaches for which the solution also quantifies uncertainty in the predictions. Within this context the objective function (5.6) can easily be viewed as the negative log-posterior in a Bayesian parameter estimation problem with data y and parameters A and B—it corresponds to i.i.d. zero-mean Gaussian priors on the entries of the factor matrices and observations corrupted by Gaussian noise. [66] and [92] each apply variational Bayes approximations of this inference model to analyze the Netflix prize challenge [15] to great success. Further, [77] and [78] develop a theoretical framework to analyze the variational Bayes low-rank matrix factorization.

A fundamental issue with the variational Bayes approach lies in one of its modeling assumptions—namely the factors A and B are taken to be independent. [95] argues a fully Bayesian framework, which does not need this assumption, can outperform the variational models. Their proposed Markov-chain Monte-Carlo (MCMC) based approach is used in several later works [22, 5]. More recently, it has been adapted for recovering low-rank tensor factorizations as well [91, 114, 115].

5.1.2 Our Contributions

In most of these variational and fully Bayesian inference setups, the priors and observations models are the same—they use Gaussian priors with zero means on the columns of the factor matrices, and assume observations are corrupted by additive Gaussian noise. One of the reasons for the popularity of this choice of priors is rooted in the fact that it leads to simple analytical conditional posteriors, which is ideal for devising a Gibbs MCMC sampler [7, 8]. Unfortunately, this choice of priors cannot fully mitigate the non-identifiability of the low-rank factorization (5.4)—given any $r \times r$ non-singular matrix W we can construct the new factors

$$\tilde{A} = AW \text{ and } \tilde{B} = BW^{-\top} \implies X = \tilde{A}\tilde{B}^{\top}. \tag{5.7}$$

The effect of this *invertible invariance* shows up in the Bayes posteriors in the form of symmetries, as illustrated in Figure 5.1, where we plot the joint posterior between various components of the factor matrices, constructed from Hamiltonian Monte-Carlo (HMC) samples, for the fully observed 4×4 rank-2 matrix described in Example 1 (Section 5.4). Effective MCMC sampling from distributions with such wide varying multi-modal and non-connected geometries is, in general, a difficult task.

The main goal of this paper is to advocate a simple change in the prior specification, which does not break the local conjugacy required for Gibbs samplers, but breaks the posterior symmetries. Our contributions are threefold:

- In Theorem 5.3, we derive the exact conditions on the matrix W such that the posterior, assuming zero-mean Gaussian priors, is invariant under the transformation $(A, B) \mapsto (AW, BW^{-\top})$.
- In Corollary 5.8 we prove that choosing a system of linearly independent prior means is sufficient for breaking this posterior symmetry. An illustration is given in Figure 5.2,

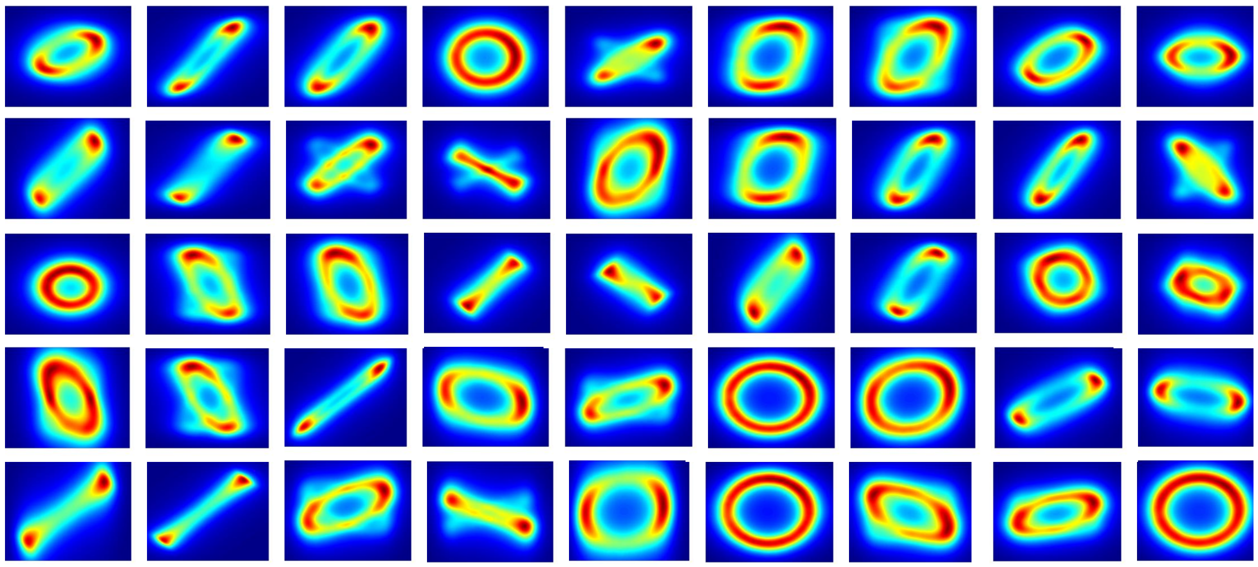


Figure 5.1: Joint posterior between a few components of the factor matrices. Results obtained using Hamiltonian Monte Carlo with zero mean priors.

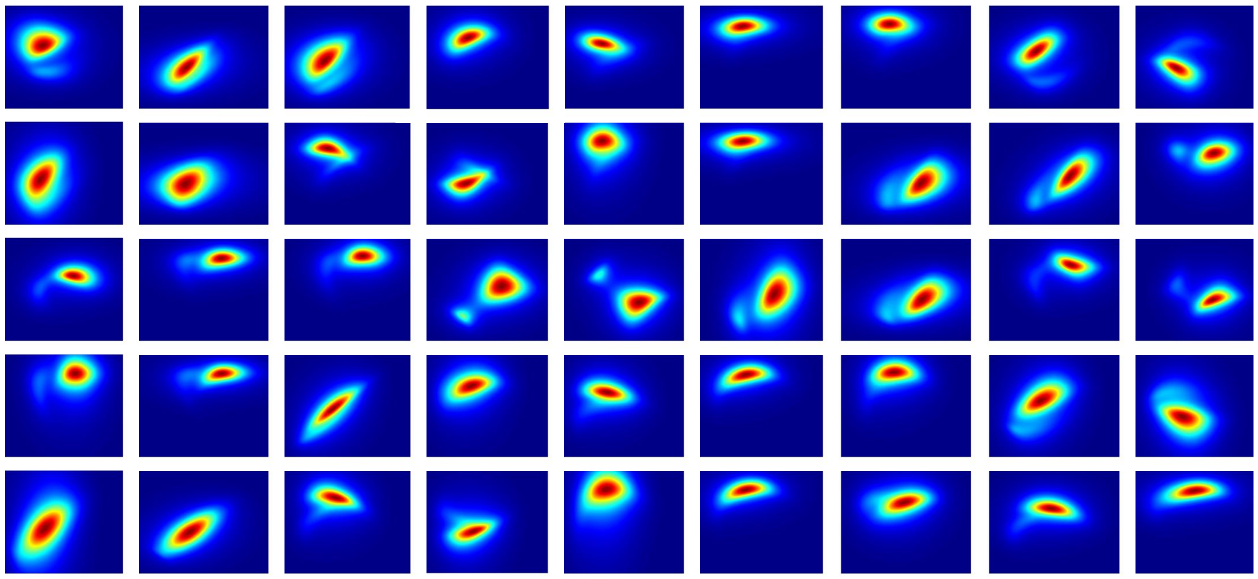


Figure 5.2: Joint posterior between a few components of the factor matrices (same as in Figure 5.1). Results obtained using Hamiltonian Monte Carlo, but this time with non-zero mean priors.

where we plot the same set of joint posteriors as in Figure 5.1, but now constructed from HMC samples obtained with non-zero mean priors.

- By breaking the symmetry via introducing non-zero mean priors, we counter the non-identifiability of low-rank factorizations. We demonstrate this ultimately leads to better performance for MCMC sampling algorithms on matrices constructed from both synthetic and real-world data. We observe up to an order of magnitude decrease in the autocorrelations of generated samples and corresponding improvement in reconstruction errors of the underlying data matrices in our numerical examples in Section 5.4.

Note that our proposed change in prior means is very simple to implement—one needs to change at most a few lines of code in any existing applications. Constructing the appropriate prior means is also straightforward. From random matrix theory, we know that if the entries of a tall-and-thin matrix are sampled i.i.d. from *any* continuous random variable in $\mathbb{R}$, then the columns of the matrix form a linearly independent system with probability 1.

The rest of the paper is structured as follows. In Section 5.2, we formally introduce the Bayesian inference setup and quantify the symmetries arising from zero-mean Gaussian priors. In Section 5.3, we show that choosing the priors means to be non-zero in a systematic fashion breaks the invertible invariance. In Section 5.4, we present the numerical experiments. Finally, in Section 5.5, we present our concluding remarks and some directions for future work.

5.2 Notations and Bayesian Inference Setup

In this section, we introduce the notations we use throughout the rest of the paper, and introduce our Bayesian inference setup for the low-rank matrix factorization problem.

5.2.1 Notations

A vector x is always represented as a column, a row-vector is represented as $x^\top$. The vector e_i is the i-th standard basis of appropriate (and inferable from context) size—all but its i-th entries are zero, and the non-zero entry is one.

Given a matrix $X \in \mathbb{R}^{m\times n}$, we use x_{ij} to denote its (i, j)-th entry. Additionally, $\bar{x}_i \in \mathbb{R}^n$ and $x_j \in \mathbb{R}^m$ denotes i-th row and the j-th column of the matrix; thus

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} = \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} = \begin{bmatrix} \bar{x}_1^\top \\ \vdots \\ \bar{x}_m^\top \end{bmatrix}. \tag{5.8}$$

The matrix I_n denotes the $n \times n$ identity matrix.

We adopt MATLAB's notation for indexing—given index sets $\Lambda_r \subseteq \{1, \ldots, m\}$ and $\Lambda_c \subseteq \{1, \ldots, n\}$ we use $X[\Lambda_r, \Lambda_c]$ to denote the intersection of the rows of $X \in \mathbb{R}^{m\times n}$ indexed by Λ_r and columns indexed by Λ_c. We also adopt the following slight abuses of notation:

$$X[i, \Lambda_c] = X[\{i\}, \Lambda_c], \qquad X[:, \Lambda_c] = X[\{1, \ldots, m\}, \Lambda_c], \tag{5.9}$$

$$X[\Lambda_r, j] = X[\Lambda_r, \{j\}], \qquad X[\Lambda_r, :] = X[\Lambda_r, \{1, \ldots, n\}]. \tag{5.10}$$

5.2.2 Prior and Likelihood Models

Given a low-rank factorization (5.4), we impose independent Gaussian priors on the columns of the factor matrices

$$a_k \sim \mathcal{N}(\mu_{a,k}, \tau_{a,k}^{-1} I_m), \quad b_k \sim \mathcal{N}(\mu_{b,k}, \tau_{b,k}^{-1} I_n). \tag{5.11}$$

The joint prior is then given by

$$p(A, B) = \prod_{k=1}^{r} \mathcal{N}(a_k \mid \mu_{a,k}, \tau_{a,k}^{-1} I_m) \mathcal{N}(b_k \mid \mu_{b,k}, \tau_{b,k}^{-1} I_n). \tag{5.12}$$

We assume real-valued matrices and a standard additive Gaussian noise model

$$y_\lambda = x_\lambda + \eta_\lambda, \quad \eta_\lambda \sim \mathcal{N}(0, \tau_\eta^{-1}), \quad \lambda \in \Lambda. \tag{5.13}$$

Here Λ is the set of location indices where the matrix was observed. The corresponding likelihood is given by

$$p(y \mid A, B) = \prod_{\lambda \in \Lambda} \mathcal{N}(y_\lambda \mid \bar{a}_{\lambda_1}^\top \bar{b}_{\lambda_2}, \tau_\eta^{-1}). \tag{5.14}$$

The effect of invertible invariance of the likelihood is immediately obvious:

Proposition 5.1. *The likelihood $p(y \mid A, B)$ defined in* (5.14) *is invariant under invertible transformations. In particular, if $W \in \mathbb{R}^{r \times r}$ is an invertible matrix, then*

$$p(y \mid A, B) = p(y \mid AW, BW^{-\top}) \quad \textit{for all} \quad A \in \mathbb{R}^{m \times r} \textit{ and } B \in \mathbb{R}^{n \times r}. \tag{5.15}$$

Proof. Let $\tilde{A} = AW$ and $\tilde{B} = BW^{-\top}$. Then we note

$$\tilde{A}\tilde{B}^\top = (AW)(BW^{-\top})^\top = AWW^{-1}B^\top = AB^\top. \tag{5.16}$$

Since all the matrix entries appearing in (5.14) are the same for both (A, B) and $(\tilde{A}, \tilde{B})$, it follows that the likelihood is invariant. □

5.2.3 The Posterior and its Symmetries

Using Bayes rule, the posterior is

$$p(A, B \mid y) \propto p(A, B)p(y \mid A, B) = p(A)p(B)p(y \mid A, B). \tag{5.17}$$

The negative log posterior is therefore

$$\begin{aligned} -\ln p(A, B \mid y) = {} & \frac{mr}{2}\ln(2\pi) + \frac{m}{2}\sum_{k=1}^{r}\ln\frac{1}{\tau_{a,k}} + \frac{1}{2}\sum_{k=1}^{r}\tau_{a,k}\|a_k - \mu_{a,k}\|^2 \\ & + \frac{nr}{2}\ln(2\pi) + \frac{n}{2}\sum_{k=1}^{r}\ln\frac{1}{\tau_{b,k}} + \frac{1}{2}\sum_{k=1}^{r}\tau_{b,k}\|b_k - \mu_{b,k}\|^2 \\ & + \frac{|\Lambda|}{2}\ln(2\pi) + \frac{|\Lambda|}{2}\ln\frac{1}{\tau_\eta} + \frac{\tau_\eta}{2}\sum_{\lambda\in\Lambda}(y_\lambda - \bar{a}_{\lambda_1}^\top \bar{b}_{\lambda_2})^2 + \text{const.} \end{aligned} \tag{5.18}$$

The first two lines correspond to the priors on A and B, and the first three terms of the final line to the likelihood. The constant term corresponds to the evidence $p(y)$ of the observations, and can generally be ignored. We have already seen that the likelihood term is invariant under invertible transformations of the form $(A, B) \mapsto (AW, BW^{-\top})$. Now we investigate any effect the prior terms might have. We immediately note the following:

Proposition 5.2. *The posterior corresponding to* (5.18) *is invariant under invertible transformation $W \in \mathbb{R}^{r\times r}$, i.e.*

$$p(A, B \mid y) = p(AW, BW^{-\top} \mid y) \quad \text{for all} \quad A \in \mathbb{R}^{m\times r} \text{ and } B \in \mathbb{R}^{n\times r}, \tag{5.19}$$

if and only if the terms

$$f_1(A) = \sum_{k=1}^{r}\tau_{a,k}\|a_k\|^2, \qquad f_2(A) = \sum_{k=1}^{r}\tau_{a,k}\mu_{a,k}^\top a_k, \tag{5.20}$$

$$f_3(B) = \sum_{k=1}^{r}\tau_{b,k}\|b_k\|^2, \qquad f_4(B) = \sum_{k=1}^{r}\tau_{b,k}\mu_{b,k}^\top b_k \tag{5.21}$$

are individually invariant under the $A \mapsto AW$ and $B \mapsto BW^{-\top}$ transformations.

Proof. Note that

$$\|a_k - \mu_{a,k}\|^2 = \|a_k\|^2 - 2\mu_{a,k}^\top a_k + \text{const.}, \tag{5.22}$$

and similarly

$$\|b_k - \mu_{b,k}\|^2 = \|b_k\|^2 - 2\mu_{b,k}^\top b_k + \text{const.} \tag{5.23}$$

Thus, we can rewrite the negative log posterior as

$$-\log p(A, B \mid y) = \frac{1}{2}f_1(A) - f_2(A) + \frac{1}{2}f_3(B) - f_4(B) - \log p(y \mid A, B) + \text{const.} \tag{5.24}$$

From Proposition 5.1, we see that the likelihood term is already invariant under the invertible transformation. It follows invariance of f_1, f_2, f_3 and f_4 is sufficient for invariance of the posterior.

We now establish that invariance of f_1, f_2, f_3 and f_4 is necessary for invariance of the posterior. To show this, first note that f_1, f_3 are homogeneous of degree two, and f_2, f_4 are homogeneous of degree one. More explicitly, for all $t, s \in \mathbb{R}$ we have

$$f_1(tA) = t^2 f_1(A), \qquad f_2(tA) = t f_2(A), \tag{5.25}$$

$$f_3(sB) = s^2 f_3(B), \qquad f_4(sB) = s f_4(B). \tag{5.26}$$

Now, fix A and B, then for invariance we must have $p(tA, sB \mid y) = p(tAW, sBW^{-\top} \mid y)$ for all $s, t \in \mathbb{R}$. Expanding this out using (5.24) and applying the homogeneity properties, we obtain

$$\begin{aligned}(5.27)\qquad &\frac{t^2}{2}f_1(A) - tf_2(A) + \frac{s^2}{2}f_3(B) - sf_4(B) = \\ &\quad \frac{t^2}{2}f_1(AW) - tf_2(AW) + \frac{s^2}{2}f_3(BW^{-\top}) - sf_4(BW^{-\top}),\end{aligned}$$

where the likelihood term cancels out. Comparing the coefficients of like-powered terms on the both sides, we conclude that we must have

$$(5.28)\qquad f_1(A) = f_1(AW) \qquad\qquad f_2(A) = f_2(AW)$$

$$(5.29)\qquad f_3(B) = f_3(BW^{-\top}) \qquad\qquad f_4(B) = f_4(BW^{-\top})$$

Since A and B are arbitrary, it follows that f_1, f_2, f_3 and f_4 must be individually invariant under the $A \mapsto AW$ and $B \mapsto BW^{-\top}$ transformations. □

Clearly, the addition of prior imposes further restrictions on W for invertible invariance of the posterior (compared to the invariance of the likelihood). In particular, with zero-mean priors, the invariance exhibited by the likelihood under the transformation $(A, B) \mapsto (AW, BW^{-\top})$ holds for the posterior only when W is restricted to a very particular subclass of invertible matrices:

Theorem 5.3 (Posterior symmetries with zero mean priors)**.** *Let* $\mu_{a,k} = 0$ *and* $\mu_{b,k} = 0$ *for all* $1 \le k \le r$ *and denote the diagonal matrices of the precision of the priors on the columns as*

$$(5.30)\qquad T_a = \mathrm{diag}(\tau_{a,1}, \dots, \tau_{a,r}), \quad T_b = \mathrm{diag}(\tau_{b,1}, \dots, \tau_{b,r}).$$

Let $\{\Lambda_1, \dots, \Lambda_q\}$ *be a partition of* $\{1, \dots, r\}$ *defined by the following:*

$$(5.31)\qquad k, k' \in \Lambda_\ell \iff \tau_{a,k}\tau_{b,k} = \tau_{a,k'}\tau_{b,k'}.$$

Then the posterior corresponding to (5.18) *is invariant under the transformation* $(A, B) \mapsto (AW, BW^{-\top})$ *with invertible* $W \in \mathbb{R}^{r\times r}$ *if and only if we can decompose*

$$W = T_a^{1/2} Q T_a^{-1/2} = T_b^{-1/2} Q T_b^{1/2}, \tag{5.32}$$

where Q *is orthogonal and block diagonal w.r.t. the partition* $\{\Lambda_1, \ldots, \Lambda_q\}$*, i.e. the submatrices*

$$Q[\Lambda_{\ell_1}, \Lambda_{\ell_2}] \text{ are } \begin{cases} \text{orthogonal} & \text{if } \ell_1 = \ell_2 \\ \text{zero} & \text{if } \ell_1 \neq \ell_2 \end{cases}. \tag{5.33}$$

We note that one direction of this theorem (that the above form of W is sufficient for the invariance of posterior) appears in [77]. We claim this structure of the matrix W is also necessary for invertible invariance as well.

Proof of Theorem 5.3. From Proposition 5.2, invertible invariance of the posterior with zero mean priors holds if and only if the terms

$$f_1(A) = \sum_{k=1}^{r} \tau_{a,k} \|a_k\|^2, \quad f_3(B) = \sum_{k=1}^{r} \tau_{b,k} \|b_k\|^2 \tag{5.34}$$

are invariant under the $A \mapsto AW$ and $B \mapsto BW^{-\top}$ transformations.

We divide the rest of the proof into three steps:

- Step 1 derives the following condition on W necessary for invertible invariance:

  $$Q_a = T_a^{-1/2} W T_a^{1/2} \quad \text{and} \quad Q_b = T_b^{1/2} W T_b^{-1/2} \tag{5.35}$$

  must be orthogonal matrices.

- Step 2 establishes $Q_a = Q_b$, and derives the block diagonal structure (5.33) on this common orthogonal matrix Q. This establishes (5.32) as a necessary condition for invertible invariance.

- Step 3 proves that (5.32) is in fact sufficient for invertible invariance.

Step 1. Let us choose

$$A = \begin{bmatrix} \alpha_1 & \cdots & \alpha_r \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}_{m\times r}. \tag{5.36}$$

Then we have

$$AW = \begin{bmatrix} \alpha^\top \\ 0^\top \\ \vdots \\ 0^\top \end{bmatrix} \begin{bmatrix} w_1 & \cdots & w_r \end{bmatrix} = \begin{bmatrix} \alpha^\top w_1 & \cdots & \alpha^\top w_r \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}_{m\times r} \tag{5.37}$$

for arbitrary $\alpha \in \mathbb{R}^r$. Then we have

$$f(A) = \sum_{k=1}^{r} \tau_{a,k}\alpha_k^2 = \left\|T_a^{1/2}\alpha\right\|^2 \tag{5.38}$$

$$f(AW) = \sum_{k=1}^{r} \tau_{a,k}(\alpha^\top w_k)^2 = \sum_{k=1}^{r} \tau_{a,k}(w_k^\top \alpha)^2 = \left\|T_a^{1/2}W^\top\alpha\right\|^2 \tag{5.39}$$

These equalities follow from the following observations:

$$T_a^{1/2}\alpha = \begin{bmatrix} \tau_{a,1}^{1/2}\alpha_1 \\ \vdots \\ \tau_{a,r}^{1/2}\alpha_r \end{bmatrix}, \quad T_a^{1/2}W^\top\alpha = T_a^{1/2}\begin{bmatrix} w_1^\top\alpha \\ \vdots \\ w_r^\top\alpha \end{bmatrix} = \begin{bmatrix} \tau_{a,1}^{1/2}w_1^\top\alpha \\ \vdots \\ \tau_{a,r}^{1/2}w_r^\top\alpha \end{bmatrix} \tag{5.40}$$

Thus, invariance of f_1 under $A \mapsto AW$ transformation requires

$$\left\|T_a^{1/2}\alpha\right\| = \left\|T_a^{1/2}W^\top\alpha\right\| \quad \text{for all} \quad \alpha \in \mathbb{R}^r \tag{5.41}$$

Set $\beta = T_a^{1/2}\alpha$, then by the invertibility of T_a we can rewrite this condition as

$$\|\beta\| = \left\|\underbrace{T_a^{1/2}W^\top T_a^{-1/2}}_{Q_a^\top}\beta\right\| \quad \text{for all} \quad \beta \in \mathbb{R}^r \tag{5.42}$$

Clearly, $Q_a^\top$ preserves lengths, and it must be orthogonal; orthogonality of Q_a follows. Proceeding in a similar manner, we can show that the invariance of f_3 under the $B \mapsto BW^{-\top}$ transformation would require Q_b (again, as defined in (5.35)) to be orthogonal.

Step 2. Using (5.35) we can compute

$$Q_a^{-1} = T_a^{-1/2}W^{-1}T_a^{1/2} \implies Q_a^{-\top} = T_a^{1/2}W^{-\top}T_a^{-1/2} \tag{5.43}$$

But since Q_a is orthogonal we have

$$Q_a = Q_a^{-\top} \implies T_a^{-1/2}WT_a^{1/2} = T_a^{1/2}W^{-\top}T_a^{-1/2} \implies WT_a = T_aW^{-\top} \tag{5.44}$$

Similarly, from (5.35) and orthogonality of Q_b, we can derive

$$T_bW = W^{-\top}T_b \tag{5.45}$$

Using (5.44) and (5.45) along with associativity of matrix multiplication, we obtain

$$T_aT_bW = T_a(T_bW) \overset{(5.45)}{=} T_a(W^{-\top}T_b) = (T_aW^{-\top})T_b \overset{(5.44)}{=} (WT_a)T_b = WT_aT_b \tag{5.46}$$

Now, equating (i,j)-th entries of the two boundary matrices in the above chain (which are easy to compute given diagonal T_a and T_b), we get

$$\tau_{a,i}\tau_{b,i}w_{ij} = w_{ij}\tau_{a,j}\tau_{b,j} \quad \text{for all} \quad 1 \le i,j \le r \tag{5.47}$$

Clearly, if $\tau_{a,i}\tau_{b,i} \neq \tau_{a,j}\tau_{b,j}$ for some pair of indices (i,j), then we must have $w_{ij} = 0$. This leads us to the block-diagonal structure of W w.r.t. partition $\{\Lambda_1, \dots, \Lambda_q\}$, i.e. $W[\Lambda_{\ell_1}, \Lambda_{\ell_2}]$ is non-zero only if $\ell_1 = \ell_2$. Using this with the diagonal nature of T_a and T_b in (5.35), we can conclude Q_a and Q_b have the same block-diagonal structure.

Next, for each $1 \le \ell \le q$ we have $\tau_{a,i}\tau_{b,i} = \tau_{a,j}\tau_{b,j}$ for all $i,j \in \Lambda_\ell$. Let us call this common value c_ℓ, then we have

$$\begin{aligned} T_a[\Lambda_\ell, \Lambda_\ell]T_b[\Lambda_\ell, \Lambda_\ell] &= \operatorname{diag}(\tau_{a,i} : i \in \Lambda_\ell)\operatorname{diag}(\tau_{b,i} : i \in \Lambda_\ell) \\ &= \operatorname{diag}(\tau_{a,i}\tau_{b,i} : i \in \Lambda_\ell) \\ &= \operatorname{diag}(c_\ell : i \in \Lambda_\ell) \\ &= c_\ell I_{r_\ell} \end{aligned} \tag{5.48}$$

where we denote $r_\ell = |\Lambda_\ell|$. We conclude

$$\begin{aligned} T_b[\Lambda_\ell, \Lambda_\ell] &= c_\ell T_a[\Lambda_\ell, \Lambda_\ell]^{-1} \\ \implies Q_b[\Lambda_\ell, \Lambda_\ell] &= T_b[\Lambda_\ell, \Lambda_\ell]^{1/2} W[\Lambda_\ell, \Lambda_\ell] T_b[\Lambda_\ell, \Lambda_\ell]^{-1/2} \\ &= c_\ell^{1/2} T_a[\Lambda_\ell, \Lambda_\ell]^{-1/2} W[\Lambda_\ell, \Lambda_\ell] c_\ell^{-1/2} T_a[\Lambda_\ell, \Lambda_\ell]^{1/2} \\ &= T_a[\Lambda_\ell, \Lambda_\ell]^{-1/2} W[\Lambda_\ell, \Lambda_\ell] T_a[\Lambda_\ell, \Lambda_\ell]^{1/2} \\ &= Q_a[\Lambda_\ell, \Lambda_\ell] \end{aligned} \tag{5.49}$$

Combining these for all the blocks, we obtain $Q_a = Q_b$. We call this common value Q, and (5.32) is trivially satisfied.

Step 3. This part of the proof is taken from Appendix G.3 of [77]:

Note that we can write

$$f_1(A) = \sum_{k=1}^{r} \tau_{a,k} \|a_k\|^2 = \operatorname{tr}((T_a A^\top) A) = \operatorname{tr}(A(T_a A^\top)) = \operatorname{tr}(A T_a A^\top) \tag{5.50}$$

where the second equality follows from the following observation:

$$\begin{aligned}
(T_a A^\top) A &= \left(\begin{bmatrix} \tau_{a,1} & & \\ & \ddots & \\ & & \tau_{a,r} \end{bmatrix} \begin{bmatrix} a_1^\top \\ \vdots \\ a_r^\top \end{bmatrix} \right) \begin{bmatrix} a_1 & \cdots & a_r \end{bmatrix} \\
&= \begin{bmatrix} \tau_{a,1} a_1^\top \\ \vdots \\ \tau_{a,r} a_r^\top \end{bmatrix} \begin{bmatrix} a_1 & \cdots & a_r \end{bmatrix} \\
&= \begin{bmatrix} \tau_{a,1} a_1^\top a_1 & \cdots & \tau_{a,1} a_1^\top a_r \\ \vdots & \ddots & \vdots \\ \tau_{a,1} a_r^\top a_1 & \cdots & \tau_{a,1} a_r^\top a_r \end{bmatrix}
\end{aligned} \tag{5.51}$$

Now, using $W = T_a^{1/2} Q T_a^{-1/2}$ from (5.32), we get

$$\begin{aligned}
f_1(AW) &= \operatorname{tr}((AW) T_a (AW)^\top) \\
&= \operatorname{tr}(A W T_a W^\top A^\top) \\
&= \operatorname{tr}(A(T_a^{1/2} Q T_a^{-1/2}) T_a (T_a^{-1/2} Q^\top T_a^{1/2}) A^\top) \\
&= \operatorname{tr}(A T_a^{1/2} Q (T_a^{-1/2} T_a T_a^{-1/2}) Q^\top T_a^{1/2} A^\top) \\
&= \operatorname{tr}(A T_a^{1/2} (Q Q^\top) T_a^{1/2} A^\top) \\
&= \operatorname{tr}(A (T_a^{1/2} T_a^{1/2}) A^\top) \\
&= \operatorname{tr}(A T_a A^\top) \\
&= f_1(A)
\end{aligned} \tag{5.52}$$

We can similarly prove that $f_3(BW^{-\top}) = f_3(B)$ with $W = T_b^{-1/2} Q T_b^{1/2}$. Thus f_1 and f_3

satisfy the desired invariance property. □

Two extreme cases of invariance with zero mean priors can be derived immediately from this theorem:

Corollary 5.4. *Let $\mu_{a,k} = 0$ and $\mu_{b,k} = 0$, $1 \leq k \leq r$. Further suppose that $\tau_{a,1} = \cdots = \tau_{a,r}$ and $\tau_{b,1} = \cdots = \tau_{b,r}$. Then the posterior corresponding to* (5.18) *is invariant under invertible transformation W if and only if W is orthogonal.*

Proof. We have $\tau_{a,1}\tau_{b,1} = \cdots = \tau_{a,r}\tau_{b,r}$. Hence Q only has one block. Consequently invertible invariance holds if and only if Q is orthogonal (by Theorem 5.3). Additionally, we can write $T_a = \tau_{a,1} I_r$ where I_r is the $r \times r$ identity matrix. It follows that

$$W = T_a^{1/2} Q T_a^{-1/2} = \tau_{a,1}^{1/2} I_r Q \tau_{a,1}^{-1/2} I_r = Q, \tag{5.53}$$

i.e. Q is orthogonal if and only if W is also orthogonal. □

Corollary 5.5. *Let $\mu_{a,k} = 0$ and $\mu_{b,k} = 0$, $1 \leq k \leq r$. Further suppose that the products $\tau_{a,1}\tau_{b,1}, \ldots, \tau_{a,r}\tau_{b,r}$ are all distinct. Then the posterior corresponding to* (5.18) *is invariant under invertible transformation W if and only if W is diagonal with non-zero entries ± 1.*

Proof. It follows from Theorem 5.3 that Q must be block diagonal with block size 1, and the each block has to be ± 1 (these are the only 1×1 orthogonal matrices). Let us write $Q = \operatorname{diag}(q_1, \ldots, q_r)$ with each $q_i = \pm 1$. Then we have

$$W = \operatorname{diag}(\tau_{a,1}^{1/2}, \ldots, \tau_{a,r}^{1/2}) \operatorname{diag}(q_1, \ldots, q_r) \operatorname{diag}(\tau_{a,1}^{-1/2}, \ldots, \tau_{a,r}^{-1/2}) = \operatorname{diag}(q_1, \ldots, q_r). \tag{5.54}$$

It follows that invertible invariance holds if and only if W is diagonal with entries ± 1. □

These corollaries make it clear that under zero-mean priors, there are at the very least 2^r symmetries in the posterior, corresponding to the $W = \operatorname{diag}(\pm 1, \ldots, \pm 1)$ transformation matrices.

5.3 Breaking the Posterior Symmetries with Non-Zero Mean Priors

It is clear from Proposition 5.2 and Theorem 5.3 that the secret to completely breaking the symmetries can only hide in the f_2 and f_4 terms, which involve the prior means. This leads to our main result:

Theorem 5.6 (Breaking posterior symmetries). *Let T_a, T_b and $\{\Lambda_1, \dots, \Lambda_q\}$ be as defined in the statement of Theorem 5.3. Define the prior mean matrices*

$$M_a = \begin{bmatrix} \mu_{a,1} & \cdots & \mu_{a,r} \end{bmatrix} \quad \text{and} \quad M_b = \begin{bmatrix} \mu_{b,1} & \cdots & \mu_{b,r} \end{bmatrix}. \tag{5.55}$$

Then the posterior $p(A, B \mid y)$ is not invariant under the $(A, B) \mapsto (AW, BW^{-\top})$ transformation for any non-identity invertible $r \times r$ matrix W if and only if the matrices

$$\begin{bmatrix} M_a[:, \Lambda_\ell] T_a[\Lambda_\ell, \Lambda_\ell]^{1/2} \\ M_b[:, \Lambda_\ell] T_b[\Lambda_\ell, \Lambda_\ell]^{1/2} \end{bmatrix} \tag{5.56}$$

have full column rank for all $1 \le \ell \le q$.

To prove this, we will need the following:

Lemma 5.7. *Let $P \in \mathbb{R}^{m \times n}$ with $m \ge n$. Then the matrix equation*

$$PW = P, \quad W \in \mathbb{R}^{n \times n} \text{ orthogonal} \tag{5.57}$$

has the unique solution $W = I$ if and only if P has full column rank.

Proof. Multiplying both sides of the matrix equation by $P^\top$ we obtain $P^\top PW = P^\top P$. If P has full column rank, then $P^\top P$ is invertible, and it follows that $W = I$ is the unique solution.

Conversely, suppose P is not full rank. Then there exists a non-zero $x \in \mathbb{R}^n$ with unit norm such that $Px = 0$. Let $W = I - 2xx^\top$, then clearly

$$PW = P(I - 2xx^\top) = P - 2(Px)x^\top = P \tag{5.58}$$

and

$$W^\top W = (I - 2xx^\top)^\top(I - 2xx^\top) = I - 2xx^\top - 2xx^\top + 4x(x^\top x)x^\top = I \tag{5.59}$$

since $x^\top x = 1$. Thus we have constructed a second solution to the matrix equation, demonstrating the non-uniqueness of the solution W. □

Proof of Theorem 5.6. In this proof, we attempt to reduce the set of all possible $r \times r$ invertible matrices W, for which invertible invariance

$$p(A, B \mid y) = p(AW, BW^{-\top} \mid y) \quad \text{for all} \quad A \in \mathbb{R}^{m\times r}, B \in \mathbb{R}^{n\times r} \tag{5.60}$$

holds, to the singleton $\{I_r\}$. We will demonstrate that this reduction is possible if and only if the P_ℓ matrices (as defined in the theorem statement) have full column rank. We achieve this as follows:

- We have already shown in Proposition 5.2 that invertible invariance of the posterior holds if and only if the f_1, f_2, f_3 and f_4 terms (as defined in the aforementioned proposition) are individually invariant under the $A \mapsto AW$ and $B \mapsto BW^{-\top}$ transformations.
- Theorem 5.3 established that in order for the f_1 and f_3 terms to invariant under the transformation above, W must have the structure

$$W = T_a^{1/2} Q T_a^{-1/2} = T_b^{-1/2} Q T_b^{1/2} \tag{5.61}$$

  where Q is block-diagonal w.r.t. partition $\{\Lambda_1, \ldots, \Lambda_q\}$ as defined in the statement of the aforementioned theorem, and the nonzero diagonal blocks $Q[\Lambda_\ell, \Lambda_\ell]$ are orthogonal for all $1 \leq \ell \leq q$.

- In Step 1 below, we consider the terms f_2 and f_4, and derive simpler and equivalent conditions on matrix W (more specifically, the matrix Q) to ensure invariance under the $A \mapsto AW$ and $B \mapsto BW^{-\top}$ transformations. These conditions are formulated in terms of the prior means $\mu_{a,k}$, $\mu_{b,k}$ and precisions $\tau_{a,k}$, $\tau_{b,k}$ for invariance.
- In Step 2, we further analyze these simpler conditions and frame them as matrix equations on diagonal blocks of Q.
- Finally, in Step 3, we will use Lemma 5.7 to demonstrate $W = I$ is the only solution of this matrix system if and only if the matrices P_ℓ are full rank for $1 \leq \ell \leq q$.

Step 1. Let us explicitly write out the invariance of f_2: we have $f_2(AW) = f_2(A)$, i.e.

$$\begin{aligned} \sum_{k'=1}^{r} \tau_{a,k'} \mu_{a,k'}^\top (AW)_{k'} &= \sum_{k'=1}^{r} \tau_{a,k'} \mu_{a,k'}^\top a_{k'} \\ \implies \sum_{k'=1}^{r} \tau_{a,k'} \mu_{a,k'}^\top AWe_{k'} &= \sum_{k'=1}^{r} \tau_{a,k'} \mu_{a,k'}^\top a_{k'} \end{aligned} \tag{5.62}$$

for all A. Note that we can write

$$A = \sum_{i'=1}^{r} a_{i'} e_{i'}^\top \tag{5.63}$$

Substituting this expression on the left hand size of (5.62), we obtain

$$
\begin{aligned}
\sum_{k'=1}^{r} \tau_{a,k'}\mu_{a,k'}^{\top} AWe_{k'} &= \sum_{k'=1}^{r} \tau_{a,k'}\mu_{a,k'}^{\top} \left(\sum_{i'=1}^{r} a_{i'}e_{i'}^{\top}\right) We_{k'} \\
&= \sum_{k'=1}^{r} \tau_{a,k'}\mu_{a,k'}^{\top} \sum_{i'=1}^{r} a_{i'}e_{i'}^{\top}(T_a^{1/2}QT_a^{-1/2})e_{k'} \\
&= \sum_{k'=1}^{r} \tau_{a,k'}\mu_{a,k'}^{\top} \sum_{i'=1}^{r} a_{i'}(e_{i'}^{\top}T_a^{1/2})Q(T_a^{-1/2}e_{k'}) \\
&= \sum_{k'=1}^{r} \tau_{a,k'}\mu_{a,k'}^{\top} \sum_{i'=1}^{r} a_{i'}(\tau_{a,i'}^{1/2}e_{i'}^{\top})Q(\tau_{a,k'}^{-1/2}e_{k'}) \qquad (5.64)\\
&= \sum_{k'=1}^{r} \tau_{a,k'}^{1/2}\mu_{a,k'}^{\top} \sum_{i'=1}^{r} \tau_{a,i'}^{1/2}a_{i'}(e_{i'}^{\top}Qe_{k'}) \\
&= \sum_{k'=1}^{r} \tau_{a,k'}^{1/2}\mu_{a,k'}^{\top} \sum_{i'=1}^{r} \tau_{a,i'}^{1/2}a_{i'}q_{i',k'} \\
&= \sum_{k'=1}^{r} \tilde{\mu}_{a,k'}^{\top} \sum_{i'=1}^{r} \tilde{a}_{i'}q_{i',k'}
\end{aligned}
$$

where we denote

$$
\tilde{\mu}_{a,k'} = \tau_{a,k'}^{1/2}\mu_{a,k'}, \quad \tilde{a}_{k'} = \tau_{a,k'}^{1/2}a_{k'}, \quad k' \in \{1,\dots,r\} \qquad (5.65)
$$

The right hand side of (5.62) can be rewritten using this notation as

$$
\sum_{k'=1}^{r} \tau_{a,k'}\mu_{a,k'}^{\top}a_{k'} = \sum_{k'=1}^{r} \tilde{\mu}_{a,k'}^{\top}\tilde{a}_{k'} \qquad (5.66)
$$

These two computations simplifies (5.62) to

$$
\sum_{k'=1}^{r} \tilde{\mu}_{a,k'}^{\top} \sum_{i'=1}^{r} \tilde{a}_{i'}q_{i',k'} = \sum_{k'=1}^{r} \tilde{\mu}_{a,k'}^{\top}\tilde{a}_{k'} \qquad (5.67)
$$

Switching the order of summation on the left side, changing the summation index on the right side, and using $\tilde{a}_{i'}^\top \tilde{\mu}_{a,k'} = \tilde{\mu}_{a,k'}^\top \tilde{a}_{i'}$, we obtain

$$\sum_{i'=1}^{r} \tilde{a}_{i'}^\top \sum_{k'=1}^{r} \tilde{\mu}_{a,k'} q_{i',k'} = \sum_{i'=1}^{r} \tilde{a}_{i'}^\top \tilde{\mu}_{a,i'} \tag{5.68}$$

It has to hold for any arbitrary $\tilde{a}_{i'} \in \mathbb{R}^m$ for $i' \in \{1, \dots, r\}$ (since the columns $a_{i'}$ of matrix A are arbitrary and $\tau_{a,i'}$ are positive reals). Let us fix $1 \leq i \leq r$ and assume all but the i-th of these vectors $\tilde{a}_{i'}$ are zeros. Then (5.68) reduces to

$$\tilde{a}_i^\top \sum_{k'=1}^{r} \tilde{\mu}_{a,k'} q_{i,k'} = \tilde{a}_i^\top \tilde{\mu}_{a,i} \tag{5.69}$$

Since this holds for arbitrary $\tilde{a}_i \in \mathbb{R}^m$, we conclude

$$\sum_{k'=1}^{r} \tilde{\mu}_{a,k'} q_{i,k'} = \tilde{\mu}_{a,i} \tag{5.70}$$

Conversely, if (5.70) holds for all $1 \leq i \leq r$, then (5.68) is trivially satisfied.

Let us pause and review our progress. Under the $A \mapsto AW$ transformation, where W has the form defined in (5.32) and (5.33) (required for invariance of the f_1 and f_3 terms, c.f. Theorem 5.3), the term f_2 is invariant if and only if identity (5.62) holds if and only if identity (5.68) holds if and only if equation (5.70) is true.

Note that $W^{-\top} = T_b^{1/2} Q^{-\top} T_b^{-1/2} = T_b^{1/2} Q T_b^{-1/2}$ where the last equality follows from orthogonality of Q. We can now repeat the same process as above, and establish that f_4 is invariant under the $B \mapsto BW^{-\top}$ transformation if and only if

$$\sum_{k'=1}^{r} \tilde{\mu}_{b,k'} q_{i,k'} = \tilde{\mu}_{b,i} \tag{5.71}$$

holds.

We combine these two arguments, and conclude f_2 and f_4 are invariant (after assuming the conditions (5.32) and (5.33) equivalent to invariances of f_1 and f_3) if and only if (5.70) and (5.71) holds.

Step 2. We now frame (5.70) and (5.71) as matrix equations for the diagonal blocks of the Q matrix. In (5.70), let us assume $i \in \Lambda_\ell$ for some $\ell \in \{1, \ldots, q\}$. Then, since Q is block-diagonal w.r.t. partitions $\{\Lambda_1, \ldots, \Lambda_\ell\}$, we have

$$q_{i,k'} = 0 \quad \text{for all} \quad k' \notin \Lambda_\ell \tag{5.72}$$

and (5.70) further reduces to

$$\sum_{k' \in \Lambda_\ell} \tilde{\mu}_{a,k'} q_{i,k'} = \tilde{\mu}_{a,i} \implies \sum_{k' \in \Lambda_\ell} \tau_{a,k'}^{1/2} \mu_{a,k'} q_{i,k'} = \tau_{a,i}^{1/2} \mu_{a,i} \tag{5.73}$$

This is a linear system with unknown $Q[i, \Lambda_\ell]$; in matrix form, we can write it as

$$M_a[:, \Lambda_\ell] T_a[\Lambda_\ell, \Lambda_\ell]^{1/2} Q[i, \Lambda_\ell]^\top = \tau_{a,i}^{1/2} \mu_{a,i} \tag{5.74}$$

We can similarly pose (5.71) as a matrix equation

$$M_b[:, \Lambda_\ell] T_b[\Lambda_\ell, \Lambda_\ell]^{1/2} Q[i, \Lambda_\ell]^\top = \tau_{b,i}^{1/2} \mu_{b,i} \tag{5.75}$$

Combining (5.74) and (5.75) for all $i \in \Lambda_\ell$, we obtain the system

$$\begin{bmatrix} M_a[:, \Lambda_\ell] T_a[\Lambda_\ell, \Lambda_\ell]^{1/2} \\ M_b[:, \Lambda_\ell] T_b[\Lambda_\ell, \Lambda_\ell]^{1/2} \end{bmatrix} Q[\Lambda_\ell, \Lambda_\ell]^\top = \begin{bmatrix} M_a[:, \Lambda_\ell] T_a[\Lambda_\ell, \Lambda_\ell]^{1/2} \\ M_b[:, \Lambda_\ell] T_b[\Lambda_\ell, \Lambda_\ell]^{1/2} \end{bmatrix} \tag{5.76}$$

Denoting the matrix on the right hand side as P_ℓ, we obtain

$$P_\ell Q[\Lambda_\ell, \Lambda_\ell]^\top = P_\ell \tag{5.77}$$

In summary, given the block-diagonal structure of Q from invariance of f_1 and f_3 terms, we have derived matrix equation (5.77) which is equivalent to (5.70) and (5.71). These later two conditions are both necessary and sufficient for invariance of f_2 and f_4 to hold.

Step 3. By Lemma 5.7, the solution $Q[\Lambda_\ell, \Lambda_\ell] = I_{r_\ell}$ of (5.77) among orthogonal $r_\ell \times r_\ell$ matrices is unique if and only if the matrix P_ℓ has full column rank. Collecting this result for all $\ell \in \{1, \ldots, q\}$, we conclude that $Q = I$ is the unique matrix generating the invertible invariance matrix W if and only if the matrices P_ℓ are full rank.

Finally note that

$$Q = I \implies W = T_a^{1/2} Q T_a^{-1/2} = T_a^{1/2} I T_a^{-1/2} = I \tag{5.78}$$

and

$$W = I \implies Q = T_a^{-1/2} W T_a^{1/2} = T_a^{-1/2} I T_a^{1/2} = I \tag{5.79}$$

Thus $Q = I$ if and only if $W = I$ and we conclude our proof. □

We now present an immediate corollary, which provides an extremely simple way to ensure full rank matrices (5.56). It essentially states that if the means are chosen to be linearly independent (e.g. the entries of the mean matrix are sampled i.i.d. from a zero-mean Gaussian distribution—the columns would be independent with probability 1), then the invariance is broken.

Corollary 5.8. *Suppose either $\{\mu_{a,k} : 1 \leq k \leq r\}$ or $\{\mu_{b,k} : 1 \leq k \leq r\}$ form a linearly independent set in $\mathbb{R}^m$ or $\mathbb{R}^n$. Then the posterior $p(A, B \mid y)$ is not invariant under any non-identity invertible transformations.*

Proof. Suppose $\{\mu_{a,k} : 1 \leq k \leq r\}$ is a linearly independent set in $\mathbb{R}^m$. Then $M_a[:, \Lambda_\ell]$, and consequently $M_a[:, \Lambda_\ell] T_a[\Lambda_\ell, \Lambda_\ell]^{1/2}$, are full rank for all ℓ. It follows that

$$P_\ell = \begin{bmatrix} M_a[:, \Lambda_\ell] T_a[\Lambda_\ell, \Lambda_\ell]^{1/2} \\ M_b[:, \Lambda_\ell] T_b[\Lambda_\ell, \Lambda_\ell]^{1/2} \end{bmatrix} \tag{5.80}$$

has full rank for all ℓ, and Theorem 5.6 applies. A similar proof can be constructed when the prior means $\{\mu_{b,k} : 1 \leq k \leq r\}$ form a linearly independent set in $\mathbb{R}^n$. $\square$

Thus, by carefully choosing non-zero means for the priors on matrix factors, we can ensure that for any $r \times r$ invertible matrix $W \neq I$, the posteriors $p(A, B \mid y)$ and $p(AW^\top, BW^{-\top} \mid y)$ are distinct. In other words, with this choice of prior distributions, we can keep the identifiability issue from affecting Bayesian inference.

5.4 Numerical Results

We now demonstrate that symmetry breaking improves MCMC sampling, both in terms of efficiency (by decreasing autocorrelation) and accuracy (by reducing reconstruction error), with four numerical experiments. Examples 1 and 2 apply Bayesian matrix factorization with synthetic data, and Examples 3 and 4 work with real-world data. For each example, the entries of the non-zero prior mean matrices are sampled from a uniform distribution. We use the root mean squared error (RMSE) as a measure of error between the true matrix $X \in \mathbb{R}^{m \times n}$ and its reconstruction $\hat{X} \in \mathbb{R}^{m \times n}$:

$$\text{RMSE} = \sqrt{\frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (x_{ij} - \hat{x}_{ij})^2}. \tag{5.81}$$

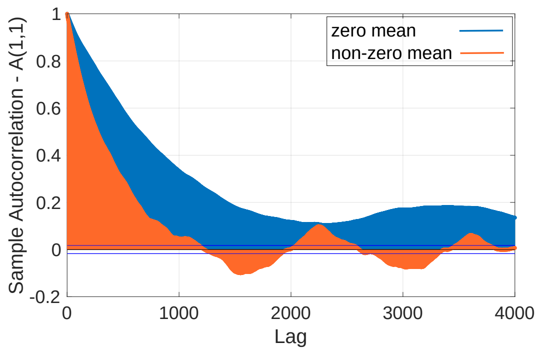
5.4.1 Example 1: Fully Observed Synthetic Matrix

We contrast the results obtained from running the HMC and Gibbs samplers on a rank-2 4×4 matrix

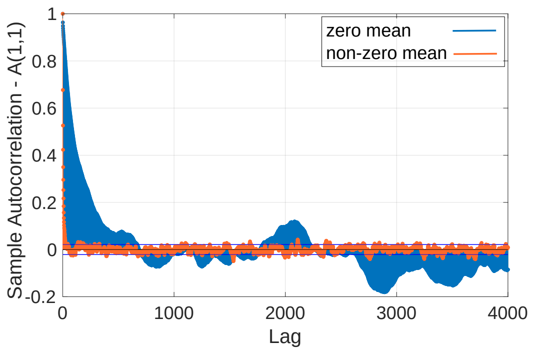
$$X = \begin{bmatrix} 1 & 0 & 1 & 5 \\ 2 & -1 & 1 & 4 \\ 4 & -1 & 3 & 14 \\ 3 & -1 & 2 & 9 \end{bmatrix}, \tag{5.82}$$

with zero and non-zero mean priors; the non-zero prior means M_a and M_b are constructed by sampling each of the entries from the Uniform$(0,1)$ distribution. We observe the full matrix with noise precision $\tau_\eta = 10^4$, and MCMC results are obtained using both Gibbs and HMC samplers that use 10 chains, each with 20000 samples. We also assume that the precision is unknown, and follow the standard procedure of hierarchical Bayes by inferring the precision with prior $\tau_\eta \sim \text{Ga}(3, 10^{-2})$. This leads to a conditionally conjugate posterior on τ_η [7].[1] In Figure 5.1 and Figure 5.2, we plot various joint posteriors, obtained using the samples generated using HMC, corresponding to the zero and non-zero mean priors. We can clearly see that the symmetries of the posterior in Figure 5.1 (corresponding to the zero mean priors) are not observed when using non-zero mean priors in Figure 5.2. This symmetry-breaking leads to better performance for MCMC samplers. For example, in Figure 5.3, we plot the autocorrelations for factor $a_{1,1}$ of the samples generated from both HMC and Gibbs samplers. The autocorrelations are significantly lower, for both samplers, when non-zero mean priors are used. This indicates using non-zero mean priors lead to better/faster mixing and smaller autocorrelations—regardless of algorithm used because the geometry of the posteriors is more favorable to the types of structures exploited by virtually all MCMC techniques. Further, in Figure 5.4, we plot the histogram of RMSE values of the matrix

[1]Even though our theory was constructed assuming τ_η is constant, it generalizes in a very straightforward manner for arbitrary prior on τ_η.



(a) Gibbs



(b) HMC

Figure 5.3: Autocorrelation for factor $a_{1,1}$ in Example 1 with zero and non-zero mean priors, computed using 10th chain of Gibbs and HMC samplers.

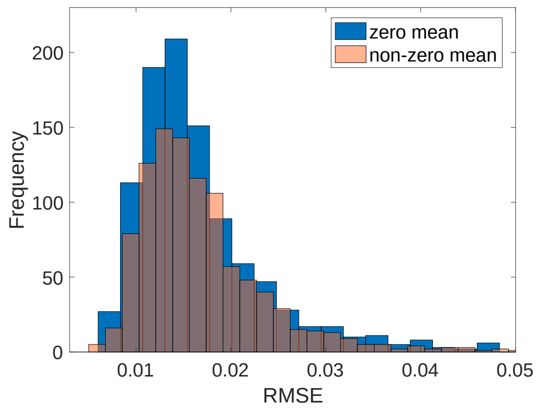


Figure 5.4: Reconstruction RMSE in Example 1 with zero and non-zero mean priors.

reconstruction for 1000 repetitions of the data gathering procedure—each experiment differs due to the noise realization and randomly sampled prior mean. For each of the repetitions we run 10 chains of the Gibbs sampler. Here the RMSE is on the order of 10^{-2} for both samplers—this is exactly what we expected because it is the order of the noise standard deviation. We can clearly see that, for this example, there is little difference in the RMSE based on the prior mean.

5.4.2 Example 2: Partially Observed Synthetic Matrix

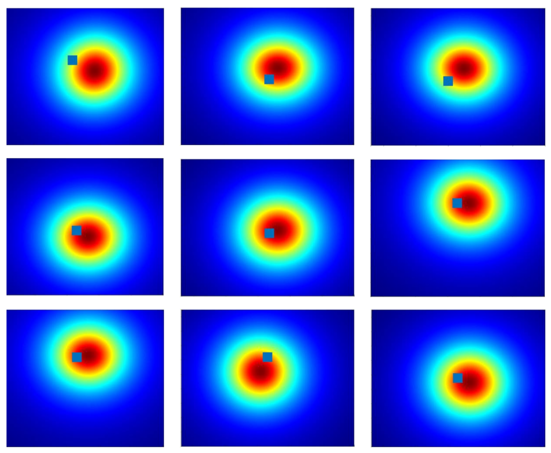
As we pointed out earlier, we often do not have access to observations of all entries of a matrix. To demonstrate the utility of our theory in context of partial observations, we use the Gibbs sampler to reconstruct a rank-5 100×100 matrix from observing only 20% of its entries. Sampling is performed with 5 chains, each with 5000 samples, and non-zero prior means are once again sampled from the $\text{Uniform}(0, 1)$ distribution. The prior on the precision is the same as the previous example. We show the posterior predictions for some elements in Figure 5.5 corresponding to zero and non-zero mean priors; results are presented for same elements in Figure 5.5(a) and Figure 5.5(b). We can clearly see that using non-zero mean priors result in better MCMC performance. The corresponding order-of-magnitudes improvement in autocorrelation of the factor $b_{50,4}$ by choosing non-zero mean priors is showcased in Figure 5.6. The RMSE histogram constructed from 50 repetitions of the experiment is shown in Figure 5.7—here we see that using non-zero mean priors lead to better reconstruction errors.

5.4.3 Example 3: Impaired Driving Dataset

We now apply our theory to reconstruct the Impaired Driving Death Rate by Age and Gender data set available publicly from CDC [20]. We assume 60% of the data is observed. Gibbs sampling is initiated with 5 chains, each with 10000 samples. Identical setup as in previous examples is used for this experiment. We demonstrate that introducing non-zero mean priors improve the autocorrelation for the $a_{20,1}$ and $a_{26,7}$ in Figure 5.8. Again we see significant improvement in efficiency in the same MCMC sampler.

5.4.4 Example 4: Mice Protein Expression Dataset

Finally, we apply our theory to the mice-protein expression data set [46] available from the UCI Machine Learning Repository [108]. The data set consists of 77 protein expressions, measured in terms of nuclear fractions, from 1080 mice specimens. For our experiments, we randomly sub-sampled this 1080×77 matrix, creating a 50×50 fully-observed sub-matrix. We then constructed a rank 10 factorization of the form (5.4) using Gibbs sampling while observing only 50% of the entries. Sampling was initiated for 4 chains, each with 20000


(a) Zero mean priors

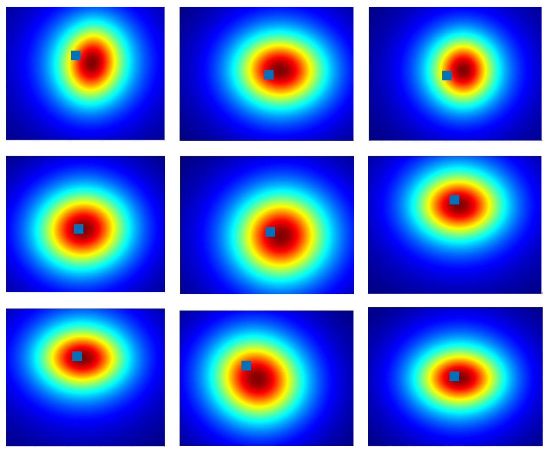

(b) Non-zero mean priors

Figure 5.5: Posterior predictions for some elements in Example 2 with Gibbs sampler. Blue marks indicate truth.

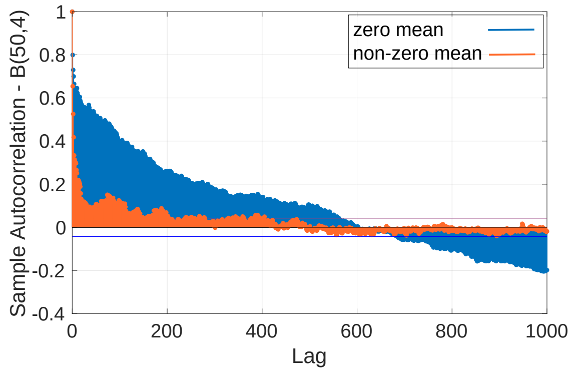


Figure 5.6: Autocorrelation for factor $b_{50,4}$ in Example 2 corresponding to zero and non-zero mean priors, computed with 5th chain of Gibbs sampler.

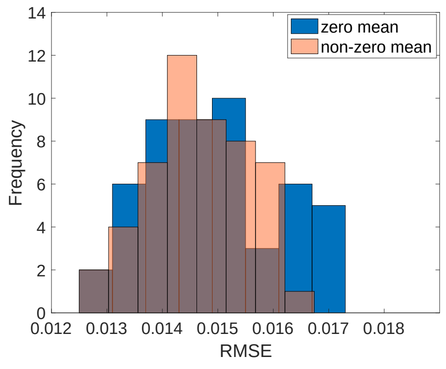
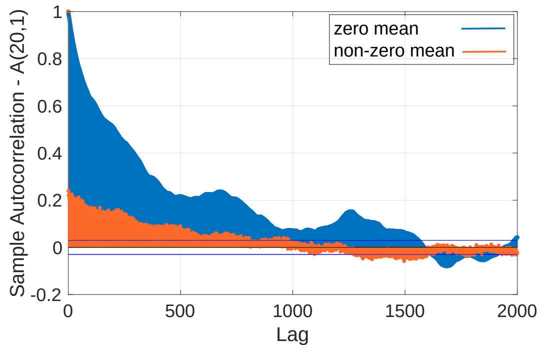
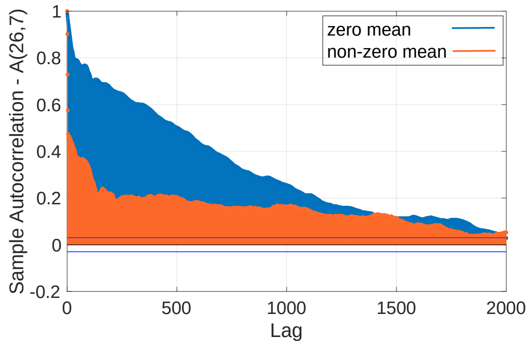


Figure 5.7: Reconstruction RMSE in Example 2 with zero and non-zero mean priors.



(a) Factor $a_{20,1}$



(b) Factor $a_{26,7}$

Figure 5.8: Autocorrelation for some factors in Example 3 corresponding to zero and non-zero mean priors, computed from 3rd chain of Gibbs sampler.

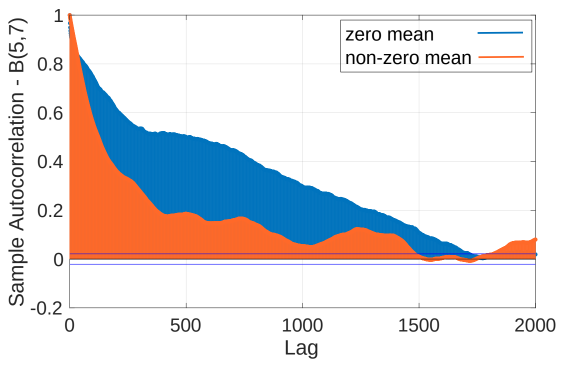


Figure 5.9: Autocorrelation for factor $b_{5,7}$ in Example 4 corresponding to zero and non-zero means, computed from 4th chain of Gibbs sampler.

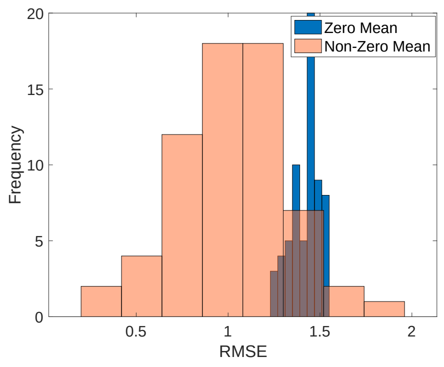


Figure 5.10: Reconstruction RMSE in Example 4 from 64 Gibbs sampling experiments with zero and non-zero mean priors.

samples. We set the parameter values

$$\tau_{a,1} = \cdots = \tau_{a,10} = 25, \quad \tau_{b,1} = \cdots = \tau_{b,10} = 25, \quad \tau_\eta = 10^2. \tag{5.83}$$

Entries of the non-zero prior mean matrices were sampled from the Uniform$(-\frac{7}{2}, \frac{7}{2})$ distribution. The first 1000 samples from all chains were discarded as burn-in. We observe that using non-zero mean priors lead to improvement of sample autocorrelation for the $b_{5,7}$ factor in Figure 5.9. We also see improved reconstruction errors in Figure 5.10—the RMSE histograms were computed from 64 independent repetitions of the experiment described above, with ten-fold sample thinning.

5.5 Conclusions

We have presented a full theoretical treatment of the symmetries of posteriors that arise from non-identifiability in Bayesian low-rank matrix factorization due to the standard choice of Gaussian priors on the matrix factors. We established that using a carefully chosen set of prior means, we can eliminate these symmetries, leading to better performance of MCMC sampling algorithms both with synthetic and real-world data. In future, we intend to extend this framework to address similar non-identifiability issues for low-rank tensor factorizations.

CHAPTER 6

Summary

In this thesis, we presented a scalable framework for resolving frictional contact between rigid particles using the cone-complementarity problem, and a fast boundary integral equation (BIE) solver for simulating Stokes flow past a multibody system of axisymmetric particles. We addressed the necessity of compressing high-fidelity simulation output data by using the tensor-train decomposition to reduce the datasets. Finally, we proposed improvements to a commonly used fully Bayesian matrix completion setup.

The scalable collision resolution scheme from Chapter 2 can be used to simulate large-scale systems. In our experiments, we were able to simulate systems consisting of 256 million spherical particles using 512 cores. Given the empirically observed near-optimal weak and strong scalability of our algorithm, we argue that our framework can simulate even larger systems using bigger supercomputers.

The fast BIE solvers presented in Chapter 3 allow us to decouple the Stokes kernels using the Fourier series and reduce the dimensionality of the integral equations by one. Taking advantage of the high-order accurate singular integration schemes available for one-dimensional curves and state-of-the-art fast multipole method implementations, we then constructed fast and accurate Stokes solvers. When the geometry is dynamic (e.g. in particulate flows), we can easily incorporate the collision resolution framework from Chapter 2 with our scheme, leading to an efficient and robust simulation framework.

In Chapter 4, we introduced the tensor-train decomposition as a tool for the compression of scientific data. We were able to compress discrete element method (DEM) simulation datasets by a factor of $O(10^4)$, reducing $O(1)$ GB datasets to $O(100)$ KB range with relative accuracy of $O(10^{-1})$. Extrapolating these compression ratios to larger datasets, we can potentially reduce terabytes of data to mere megabytes and post-process/visualize/learn from the reduced data on our personal computers.

Chapter 5 described a standard Bayesian inference setup for inferring missing values of a low-rank matrix. We proposed a change of priors which leads to better sampling and reconstruction performance to the commonly used setup. We argue that when the computational budget for a high-fidelity simulation is low, we can use this matrix completion as a reduced order model, and predict the output of the simulation from existing data.

There are many possible future directions. In the collision resolution framework, we use direct solvers in the Newton step of second order optimization algorithms; switching to iterative solvers is likely to provide better scalability. We also want to incorporate our methods into existing soil-vehicle simulation modules.

In the axisymmetric Stokes solver setup, accurate evaluation of the solution field near the boundary remains a challenge. This is also necessary when building a dense particulate flow simulator with contact resolution.

The Bayesian matrix factorization setup, at its current form, can only handle two-dimensional datasets (or a two-parameter family of simulations). The canonical polyadic (CP) tensor decomposition [58] is a straightforward generalization of the low-rank matrix factorization to higher dimensions. We aim to develop a similar set of theoretical guarantees for posterior symmetry breaking in that context as well.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Evrim Acar and Bülent Yener. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, 21(1):6–20, 2008.

[2] Ludvig af Klinteberg and Anna-Karin Tornberg. A fast integral equation method for solid particles in viscous flow using quadrature by expansion. *Journal of Computational Physics*, 326:420–445, 2016.

[3] Ludvig af Klinteberg and Anna-Karin Tornberg. Error estimation for quadrature by expansion in layer potential evaluation. *Advances in Computational Mathematics*, 43(1):195–234, 2017.

[4] Ludvig af Klinteberg and Anna-Karin Tornberg. Adaptive quadrature by expansion for layer potential evaluation in two dimensions. *SIAM Journal on Scientific Computing*, 40(3):A1225–A1249, 2018.

[5] Sungjin Ahn, Anoop Korattikara, Nathan Liu, Suju Rajan, and Max Welling. Large-scale distributed Bayesian matrix factorization using stochastic gradient MCMC. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 9–18, 2015.

[6] Bradley K Alpert. Hybrid Gauss-trapezoidal quadrature rules. *SIAM Journal on Scientific Computing*, 20(5):1551–1584, 1999.

[7] Pierre Alquier, Vincent Cottet, Nicolas Chopin, and Judith Rousseau. Bayesian matrix completion: Prior specification. *arXiv preprint arXiv:1406.1440*, 2014.

[8] Pierre Alquier et al. A Bayesian approach for noisy matrix completion: Optimal rate under general sampling distribution. *Electronic Journal of Statistics*, 9(1):823–841, 2015.

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

[10] 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.

[11] Mihai Anitescu and Alessandro Tasora. An iterative approach for cone complementarity problems for nonsmooth dynamics. *Computational Optimization and Applications*, 47(2):207–235, 2010.

[12] Woody Austin, Grey Ballard, and Tamara G Kolda. Parallel tensor compression for large-scale scientific data. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 912–922. IEEE, 2016.

[13] Satish Balay, Shrirang Abhyankar, Mark Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, W Gropp, et al. PETSc users manual. Technical report, Argonne National Laboratory, 2019.

[14] Heiko Bauke and Stephan Mertens. Random numbers for large-scale distributed Monte Carlo simulations. *Physical Review E*, 75(6):066701, 2007.

[15] James Bennett, Stan Lanning, et al. The Netflix prize. In *Proceedings of KDD Cup and Workshop*. New York, NY, USA, 2007.

[16] Richard Berger, Christoph Kloss, Axel Kohlmeyer, and Stefan Pirker. Hybrid parallelization of the LIGGGHTS open-source DEM code. *Powder Technology*, 278:234–247, 2015.

[17] Emmanuel J Candès and Yaniv Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

[18] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[19] Emmanuel J Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

[20] Impaired driving death rate, by age and gender, 2012 & 2014, all states. https://data.cdc.gov/Motor-Vehicle/Impaired-Driving-Death-Rate-by-Age-and-Gender-2012/ebbj-sh54.

[21] Timothy M Chan. Closest-point problems simplified on the RAM. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 472–473, 2002.

[22] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, pages 1683–1691. PMLR, 2014.

[23] Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, and Danilo P Mandic. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9(4-5):249–429, 2016.

[24] Eduardo Corona, David Gorsich, Paramsothy Jayakumar, and Shravan Veerapaneni. Tensor train accelerated solvers for nonsmooth rigid body dynamics. *Applied Mechanics Reviews*, 2019.

[25] Eduardo Corona, Abtin Rahimian, and Denis Zorin. A tensor-train accelerated solver for integral equations in complex geometries. *Journal of Computational Physics*, 334:145–169, 2017.

[26] Eduardo Corona and Shravan Veerapaneni. Boundary integral equation analysis for suspension of spheres in Stokes flow. *Journal of Computational Physics*, 362, 2018.

[27] Erwin Coumans et al. Bullet physics library. https://bulletphysics.org.

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

[29] Mark A Davenport and Justin Romberg. An overview of low-rank matrix recovery from incomplete observations. *IEEE Journal of Selected Topics in Signal Processing*, 10, 2016.

[30] Saibal De, Eduardo Corona, Paramsothy Jayakumar, and Shravan Veerapaneni. Scalable solvers for cone complementarity problems in frictional multibody dynamics. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7. IEEE, 2019.

[31] Saibal De, Hadi Salehi, and Alex Gorodetsky. Efficient MCMC sampling for Bayesian matrix factorization by breaking posterior symmetries. *arXiv preprint arXiv:2006.04295*, 2020.

[32] A. Decoene, S. Martin, and B. Maury. Microscopic modelling of active bacterial suspensions. *Mathematical Modelling of Natural Phenomena*, 6(5):98–129, 2011.

[33] Charles L. Epstein, Leslie Greengard, and Michael O'Neil. A high-order wideband direct solver for electromagnetic scattering from bodies of revolution. *Journal of Computational Physics*, 387, 2017.

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

[35] Henry E. Fettis. A new method for computing toroidal harmonics. *Mathematics of Computation*, 24, 1970.

[36] Sadayuki Furuhashi et al. Msgpack. https://msgpack.org.

[37] Zydrunas Gimbutas and Leslie Greengard. STKFMMLIB3D 1.2, 2012.

[38] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[39] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

[40] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org.

[41] S. Hao, A. H. Barnett, P. G. Martinsson, and P. Young. High-order accurate methods for Nyström discretization of integral equations on smooth curves in the plane. *Advances in Computational Mathematics*, 40, 2014.

[42] Wei He, Hongyan Zhang, Liangpei Zhang, and Huanfeng Shen. Total-variation-regularized low-rank matrix factorization for hyperspectral image restoration. *IEEE Transactions on Geoscience and Remote Sensing*, 54, 2015.

[43] Johan Helsing and Anders Karlsson. An explicit kernel-split panel-based Nyström scheme for integral equations on axially symmetric surfaces. *Journal of Computational Physics*, 272, 2014.

[44] Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P. Pawlowski, Eric T. Phipps, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. An overview of the Trilinos project. *ACM Transactions on Mathematical Software*, 31(3), 2005.

[45] 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.

[46] Clara Higuera, Katheleen J Gardiner, and Krzysztof J Cios. Self-organizing feature maps identify proteins critical to learning in a mouse model of Down syndrome. *PloS One*, 10, 2015.

[47] Alan C Hindmarsh, Peter N Brown, Keith E Grant, Steven L Lee, Radu Serban, Dan E Shumaker, and Carol S Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

[48] K. Ho and Leslie Greengard. A fast semidirect least squares algorithm for hierarchically block separable matrices. *SIAM Journal on Matrix Analysis and Applications*, 35(2):725–748, 2014.

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

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

[51] George C. Hsiao and Wolfgang L. Wendland. *Boundary Integral Equations*. Springer-Verlag, 2008.

[52] Klaus Iglberger. Blaze C++ linear algebra library. https://bitbucket.org/blaze-lib, 2012.

[53] Paramsothy Jayakumar and Dave Mechergui. Efficient generation of accurate mobility maps using machine learning algorithms. Technical report, US Army TARDEC, Warren, United States, 2019.

[54] S. Kapur and V. Rokhlin. High-order corrected trapezoidal quadrature rules for singular functions. *SIAM Journal on Numerical Analysis*, 34(4):1331–1356, 1997.

[55] Brian J Kirby. *Micro- and Nanoscale Fluid Mechanics: Transport in Microfluidic Devices*. Cambridge University Press, New York, 2010.

[56] Jan Kleinert. *Simulating Granular Material using Nonsmooth Time-Stepping and a Matrix-Free Interior Point Method*. Fraunhofer Verlag, 2015.

[57] A. Klöckner, A. Barnett, L. Greengard, and M. O'Neil. Quadrature by expansion: A new method for the evaluation of layer potentials. *Journal of Computational Physics*, 252:332–349, 2013.

[58] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[59] P. Kolm and V. Rokhlin. Numerical quadratures for singular and hypersingular integrals. *Computers and Mathematics with Applications*, 41, 2001.

[60] Rainer Kress. *Linear Integral Equations*. Springer New York, 2014.

[61] Olga A. Ladyzhenskaya. *The Mathematical Theory of Viscous Incompressible Flow*. Gordon and Breach, New York, 1969.

[62] Shaomeng Li, Nicole Marsaglia, Christoph Garth, Jonathan Woodring, John Clyne, and Hank Childs. Data reduction techniques for simulation, visualization and data analysis. *Computer Graphics Forum*, 37(6):422–447, 2018.

[63] Xiao Peng Li, Qi Liu, and Hing Cheung So. Rank-one matrix approximation with ℓ^p-norm for image inpainting. *IEEE Signal Processing Letters*, 27:680–684, 2020.

[64] Xiaoye S Li. An overview of SuperLU: Algorithms, implementation, and user interface. *ACM Transactions on Mathematical Software*, 31(3), 2005.

[65] Xikui Li, Z Wang, Y Du, and Q Duan. Advances in multiscale FEM-DEM modeling of granular materials. In *International Conference on Discrete Element Methods*, pages 267–279. Springer, 2016.

[66] Yew Jin Lim and Yee Whye Teh. Variational Bayesian approach to movie rating prediction. In *Proceedings of KDD Cup and Workshop*, 2007.

[67] Peter Lindstrom and Martin Isenburg. Fast and efficient compression of floating-point data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1245–1250, 2006.

[68] Huazhong Liu, Laurence T Yang, Jihong Ding, Yimu Guo, Xia Xie, and Zhi-Jie Wang. Scalable tensor-train-based tensor computations for cyber–physical–social big data. *IEEE Transactions on Computational Social Systems*, 7(4):873–885, 2020.

[69] Huazhong Liu, Laurence T Yang, Yimu Guo, Xia Xie, and Jianhua Ma. An incremental tensor-train decomposition for cyber-physical-social big data. *IEEE Transactions on Big Data*, 2018.

[70] Gunjan Mahindre, Anura P Jayasumana, Kelum Gajamannage, and Randy Paffenroth. On sampling and recovery of topology of directed social networks: A low-rank matrix completion based approach. In *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, pages 324–331. IEEE, 2019.

[71] Osman Asif Malik and Stephen Becker. Low-rank Tucker decomposition of large tensors using tensorsketch. *Advances in Neural Information Processing Systems*, 31:10096–10106, 2018.

[72] Gary R Marple, David Gorsich, Paramsothy Jayakumar, and Shravan Veerapaneni. An active learning framework for constructing high-fidelity mobility maps. *arXiv preprint arXiv:2003.03517*, 2020.

[73] 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.

[74] Hammad Mazhar, Toby Heyn, Arman Pazouki, Daniel Melanz, Andrew Seidl, Aaron Bartholomew, Alessandro Tasora, and Dan Negrut. Chrono: A parallel multi-physics library for rigid-body, flexible-body, and fluid dynamics. *Mechanical Sciences*, 4(1):49–64, 2013.

[75] 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.

[76] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, 2008.

[77] Shinichi Nakajima and Masashi Sugiyama. Theoretical analysis of Bayesian matrix factorization. *Journal of Machine Learning Research*, 12, 2011.

[78] Shinichi Nakajima, Masashi Sugiyama, S. Derin Babacan, and Ryota Tomioka. Global analytic solution of fully-observed variational Bayesian matrix factorization. *Journal of Machine Learning Research*, 14, 2013.

[79] Dan Negrut, Radu Serban, Hammad Mazhar, and Toby Heyn. Parallel computing in multibody system dynamics: Why, when, and how. *Journal of Computational and Nonlinear Dynamics*, 9(4), 2014.

[80] Dan Negrut, Alessandro Tasora, Mihai Anitescu, Hammad Mazhar, Toby Heyn, and Arman Pazouki. Solving large multibody dynamics problems on the GPU. In *GPU Computing Gems Jade Edition*, pages 269–280. Elsevier, 2012.

[81] Y E Nesterov and M J Todd. Self-scaled barriers and interior-point methods for convex programming. *Mathematics of Operations Research*, 22(1), 1997.

[82] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Sov. Math. Dokl.*, 27(2):372–376, 1983.

[83] Viet-Dung Nguyen, Karim Abed-Meraim, and Nguyen Linh-Trung. Fast adaptive PARAFAC decomposition algorithm with linear complexity. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6235–6239. IEEE, 2016.

[84] F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain eds. NIST digital library of mathematical functions. http://dlmf.nist.gov/, Release 1.1.2 of 2021-06-15.

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

[86] 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.

[87] P. G. Martinsson P. Young, S. Hao. A high-order Nyström discretization scheme for boundary integral equations defined on rotationally symmetric surfaces. *Journal of Computational Physics*, 231, 2012.

[88] 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.

[89] 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.

[90] C. Pozrikidis. *Boundary Integral and Singularity Methods for Linearized Viscous Flow*. Cambridge University Press, 1992.

[91] Piyush Rai, Yingjian Wang, Shengbo Guo, Gary Chen, David Dunson, and Lawrence Carin. Scalable Bayesian low-rank decomposition of incomplete multiway tensors. In *International Conference on Machine Learning*, 2014.

[92] Tapani Raiko, Alexander Ilin, and Juha Karhunen. Principal component analysis for large scale problems with lots of missing values. In *European Conference on Machine Learning*, 2007.

[93] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52, 2010.

[94] Antonio Recuero, Radu Serban, Bryan Peterson, Hiroyuki Sugiyama, Paramsothy Jayakumar, and Dan Negrut. A high-fidelity approach for vehicle mobility simulation: Nonlinear finite element tires operating on granular material. *Journal of Terramechanics*, 72:39–54, 2017.

[95] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[96] Johannes Schauer and Andreas Nüchter. Collision detection between point clouds using an efficient k-d tree implementation. *Advanced Engineering Informatics*, 29(3):440–458, 2015.

[97] Michael Siegel and Anna-Karin Tornberg. A local target specific quadrature by expansion method for evaluation of layer potentials in 3D. *Journal of Computational Physics*, 364:365–392, 2018.

[98] Shaden Smith, Kejun Huang, Nicholas D Sidiropoulos, and George Karypis. Streaming tensor factorization for infinite data sources. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 81–89. SIAM, 2018.

[99] Saverio E. Spagnolie, editor. *Complex Fluids in Biological Systems: Experiment, Theory, and Computation.* Springer, New York, 2015.

[100] Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1329–1336, 2005.

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

[102] 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.

[103] Yiming Sun, Yang Guo, Charlene Luo, Joel Tropp, and Madeleine Udell. Low-rank Tucker approximation of a tensor from streaming data. *SIAM Journal on Mathematics of Data Science*, 2(4):1123–1150, 2020.

[104] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Investigation of various matrix factorization methods for large recommender systems. In *2008 IEEE International Conference on Data Mining Workshops*, 2008.

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

[106] 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.

[107] Le Thanh, Karim Abed-Meraim, Nguyen Linh-Trung, and Remy Boyer. Adaptive algorithms for tracking tensor-train decomposition of streaming tensors. In *European Signal Processing Conference (EUSIPCO'20)*, 2021.

[108] Mice protein expression data set. https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression.

[109] Bo Xue, Linghua Zhang, Yang Yu, and Weiping Zhu. Locating the nodes from incomplete Euclidean distance matrix using Bayesian learning. *IEEE Access*, 7:37406–37413, 2019.

[110] Yoshihiro Yamanishi, Masaaki Kotera, Minoru Kanehisa, and Susumu Goto. Drug-target interaction prediction from chemical, genomic and pharmacological data in an integrated framework. *Bioinformatics*, 26, 2010.

[111] Wen Yan, Eduardo Corona, Dhairya Malhotra, Shravan Veerapaneni, and Michael Shelley. A scalable computational platform for particulate Stokes suspensions. *Journal of Computational Physics*, page 109524, 2020.

[112] Lexing Ying, George Biros, and Denis Zorin. A high-order 3D boundary integral equation solver for elliptic PDEs in smooth domains. *Journal of Computational Physics*, 219, 2006.

[113] Jidong Zhao. Hierarchical multiscale modeling of strain localization in granular materials: A condensed overview and perspectives. In *International Workshop on Bifurcation and Degradation in Geomaterials*, pages 349–359. Springer, 2017.

[114] Qibin Zhao, Liqing Zhang, and Andrzej Cichocki. Bayesian CP factorization of incomplete tensors with automatic rank determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37, 2015.

[115] Qibin Zhao, Guoxu Zhou, Liqing Zhang, Andrzej Cichocki, and Shun-Ichi Amari. Bayesian robust tensor factorization for incomplete multiway data. *IEEE Transactions on Neural Networks and Learning Systems*, 27, 2015.

[116] Xiaodong Zheng, Hao Ding, Hiroshi Mamitsuka, and Shanfeng Zhu. Collaborative matrix factorization with multiple similarities for predicting drug-target interactions. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013.

[117] Xinran Zhong and WaiChing Sun. An adaptive reduced-dimensional discrete element model for dynamics responses of granular materials with high-frequency noises. *International Journal for Multiscale Computational Engineering*, 16(4):345–366, 2018.