Summer 8-31-2012

# Fast algorithms for Brownian dynamics simulation with hydrodynamic interactions

Zhi Liang
*New Jersey Institute of Technology*

# Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

## ABSTRACT

## FAST ALGORITHMS FOR BROWNIAN DYNAMICS SIMULATION WITH HYDRODYNAMIC INTERACTIONS

**by**
**Zhi Liang**

In the Brownian dynamics simulation with hydrodynamic interactions, one needs to generate the total displacement vectors of Brownian particles consisting of two parts: a deterministic part which is proportional to the product of the Rotne-Prager-Yamakawa (RPY) tensor $D$ [1, 2] and the given external forces $\mathbf{F}$; and a hydrodynamically correlated random part whose covariance is proportional to the RPY tensor. To be more precise, one needs to calculate $D\mathbf{u}$ for a given vector $\mathbf{u}$ and compute $\sqrt{D}\mathbf{v}$ for a normally distributed random vector $\mathbf{v}$. For an arbitrary $N$-particle configuration, $D$ is a $3N \times 3N$ matrix and $\mathbf{u}$, $\mathbf{v}$ are vectors of length $3N$. Thus, classical algorithms require $O(N^2)$ operations for computing $D\mathbf{u}$ and $O(N^3)$ operations for computing $\sqrt{D}\mathbf{v}$, which are prohibitively expensive and render large scale simulations impossible since one needs to carry out these calculations many times in a Brownian dynamics simulation.

In this dissertation, we first present two fast multipole methods (FMM) for computing $D\mathbf{u}$. The first FMM is a simple application of the kernel independent FMM (KIFMM) developed by Ying, Biros, and Zorin [3], which requires 9 scalar FMM calls. The second FMM, similar to the FMM for Stokeslet developed by Tornberg and Greengard [4], decomposes the RPY tensor into harmonic potentials and its derivatives, and thus requires only four harmonic FMM calls. Both FMMs reduce the computational cost of $D\mathbf{u}$ from $O(N^2)$ to $O(N)$ for an arbitrary $N$-particle configuration.

We then discuss several methods of computing $\sqrt{D}\mathbf{v}$, which are all based on the Krylov subspace approximations, that is, replacing $\sqrt{D}\mathbf{v}$ by $p(D)\mathbf{v}$ with $p(D)$ a

low degree polynomial in $D$. We first show rigorously that the popular Chebyshev spectral approximation method (see, for example, [5, 6]) requires $\sqrt{\kappa}\log\frac{1}{\epsilon}$ terms for a desired precision $\epsilon$, where $\kappa$ is the condition number of the RPY tensor $D$. In the Chebyshev spectral approximation method, one also needs to estimate the extreme eigenvalues of $D$. We have considered several methods: the classical Lanczos method, the Chebyshev-Davidson method, and the safeguarded Lanczos method proposed by Zhou and Li [7]. Our numerical experiments indicate that $\kappa$ is usually very small when the particles are distributed uniformly with low density, and that the safeguarded Lanczos method is most effective for our cases with very little additional computational cost. Thus, when combined with the FMMs we described earlier, the Chebyshev approximation method with safeguarded Lanczos method as eigenvalue estimators essentially reduces the cost of computing $\sqrt{D}\mathbf{v}$ from $O(N^3)$ to $O(N)$ for most practical particle configurations. Finally, we propose to combine the so-called spectral Lanczos decomposition method (SLDM) (see, for example, [8]) and the FMMs to compute $\sqrt{D}\mathbf{v}$. Our numerical experiments show that the SLDM is generally more efficient than the popular Chebyshev spectral approximation method.

The fast algorithms developed in this dissertation will be useful for the study of diffusion limited reactions, polymer dynamics, protein folding, and particle coagulation as it enables large scale Brownian dynamics simulations. Moreover, the algorithms can be extended to speed up the computation involving the matrix square root for many other matrices, which has potential applications in areas such as statistical analysis with certain spatial correlations and model reduction in dynamic control theory.

# FAST ALGORITHMS FOR BROWNIAN DYNAMICS SIMULATION WITH HYDRODYNAMIC INTERACTIONS

by
Zhi Liang

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology and
Rutgers, The State University of New Jersey – Newark
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Mathematical Sciences

Department of Mathematical Sciences
Department of Mathematics and Computer Science, Rutgers-Newark

August 2012

**APPROVAL PAGE**

**FAST ALGORITHMS FOR BROWNIAN DYNAMICS SIMULATION WITH HYDRODYNAMIC INTERACTIONS**

**Zhi Liang**

---

Dr. Shidong Jiang, Dissertation Advisor — Date
Associate Professor of Mathematical Sciences, NJIT

---

Dr. Jingfang Huang, Committee Member — Date
Professor of Mathematics, University of North Carolina, Chapel Hill

---

Dr. Peter G. Petropoulos, Committee Member — Date
Associate Professor of Mathematical Sciences, NJIT

---

Dr. Michael S. Siegel, Committee Member — Date
Professor of Mathematical Sciences, NJIT

---

Dr. Yuan-nan Young, Committee Member — Date
Associate Professor of Mathematical Sciences, NJIT

## BIOGRAPHICAL SKETCH

**Author:** Zhi Liang

**Degree:** Doctor of Philosophy

**Date:** August 2012

### Undergraduate and Graduate Education:

- Doctor of Philosophy in Mathematical Sciences,
  New Jersey Institute of Technology, Newark, NJ, 2012
- Master of Science in Applied Mathematics,
  New Jersey Institute of Technology, Newark, NJ, 2009
- Bachelor of Science in Mathematics and Applied Mathematics,
  Beijing Normal University, Beijing, China, 2006

**Major:** Applied Mathematics

### Presentations and Publications:

Zhi Liang, Zydrunas Gimbutas, Leslie Greengard, Jingfang Huang, Shidong Jiang, "A Fast Multipole Method for the Rotne-Prager-Yamakawa Tensor and Its Applications", *Journal of Computational Physics*, submitted.

Shidong Jiang, Zhi Liang, Jingfang Huang, "A Fast Algorithm for Brownian Dynamics Simulation with Hydrodynamic Interactions", *Mathematics of Computation*, accepted.

Zhi Liang, Shidong Jiang, "A Fast Algorithm for Brownian Dynamics Simulation with Hydrodynamic Interactions", *the Eighth Annual Conference on Frontiers in Applied and Computational Mathematics (FACM'11)*, Department of Mathematical Sciences, NJIT, June 9-11, 2011.

Zhi Liang, "A Fast Algorithm for Brownian Dynamics Simulation with Hydrodynamic Interactions", *Applied Mathematics Seminar*, Department of Mathematical Sciences, NJIT, June 7, 2011.

Zhi Liang, "The Uniform Distribution Fast Multipole Methods in 1D", *Applied Mathematics Seminar*, Department of Mathematical Sciences, NJIT, July 14, 2010.

*Stay Hungry. Stay Foolish.*

—Steve Jobs

## ACKNOWLEDGMENT

First and foremost, I would like to express my greatest gratitude and appreciation to my advisor, Dr. Shidong Jiang, without whom this dissertation would simply be impossible. He has generously helped me and supported me in various ways. Ever since I joined the PhD program of the Department of Mathematical Sciences in NJIT, he has been a steadfast source of information, ideas, support, and energy. I am deeply grateful for the patient, encouraging and resourceful guidance he has been given me throughout the last few years, and I sincerely wish for him all the best in his future endeavors. It is my great honor to have him as my advisor.

I also would like to extend my gratitude to the other members on my committee for their kind encouragement and support throughout all of these years while I was in the graduate school, as well as throughout the process of working on my dissertation: Professor Jingfang Huang, Professor Peter Petropoulos, Professor Michael Siegel, and Professor Yuan-nan Young. I would also like to thank Professor Jonathan Luke, Chair of the Department of Mathematical Sciences, for his support.

I also would like to thank all of my friends (students, faculty, and administrators) for providing me with such a pleasant environment during my study at NJIT, and for all of the help they have given me over the last five years. It has been a great honor for me to get to know all of them.

Last, but not the least, I would like to thank my family for their support. To my parents, for raising me, believing in me, supporting me, and encouraging me for all of these years.

## TABLE OF CONTENTS


**Chapter** **Page**

1 INTRODUCTION . . . . . 1

2 NUMERICAL PRELIMINARY . . . . . 6

2.1 The Square Root of a Real, Symmetric, and Positive Definite Matrix . . . . . 6

2.2 Spectral Approximation Using Chebyshev Polynomials . . . . . 7

2.3 The Rotne-Prager-Yamakawa Tensor . . . . . 8

2.4 A Brief Overview to the Fast Multipole Method . . . . . 11

2.4.1 Introduction . . . . . 11

2.4.2 Data Structure of Adaptive FMM . . . . . 15

2.4.3 Approximations and Translations . . . . . 17

3 FAST MULTIPOLE METHODS FOR THE ROTNE-PRAGER-YAMAKAWA TENSOR . . . . . 23

3.1 Kernel Independent FMM . . . . . 23

3.2 FMM with Harmonic Potentials . . . . . 25

3.3 Numerical Results . . . . . 30

4 FAST ALGORITHMS FOR GENERATING RANDOM DISPLACEMENT VECTORS . . . . . 33

4.1 Algorithm I: Lanczos Method with Chebyshev-Davidson Method plus Chebyshev Spectral Approximation . . . . . 33

4.1.1 Estimating the Extreme Eigenvalues of $D$ . . . . . 33

4.1.2 Chebyshev Spectral Approximation for Computing Matrix Square Root . . . . . 36

4.2 Algorithm II: Safeguarded Lanczos Method plus Chebyshev Spectral Approximation . . . . . 40

4.2.1 Estimating the Extreme Eigenvalues of $D$ . . . . . 40

4.2.2 Chebyshev Spectral Approximation for Computing Matrix Square Root . . . . . 41

4.3 Algorithm III: Spectral Lanczos Decomposition Method . . . . . 47

## TABLE OF CONTENTS
## (Continued)

**Chapter** **Page**

5 NUMERICAL RESULTS . . . . . 50

5.1 Condition Number and Terms Needed in Chebyshev Approximation . 50

5.2 Algorithm I: Lanczos Method with Chebyshev-Davidson Method plus Chebyshev Spectral Approximation . . . . . 52

5.3 Algorithm II: Safeguarded Lanczos Method plus Chebyshev Spectral Approximation . . . . . 53

5.4 Algorithm III: Spectral Lanczos Decomposition Method . . . . . 55

5.5 Comparison of the Algorithm I, II and III . . . . . 56

6 CONCLUSION . . . . . 58

BIBLIOGRAPHY . . . . . 61

## LIST OF TABLES


| Table | | Page |
|---|---|---|
| 3.1 | Relative Error and Timing Results of KIFMM | 24 |
| 3.2 | Relative Error and Timing Results of RPYFMM | 29 |
| 3.3 | Timing results for uniform distribution in a cube. | 30 |
| 3.4 | Timing results for nonuniform distribution on a sphere. | 31 |
| 5.1 | Number of Terms Needed in Chebyshev Approximation to Approximate $\sqrt{x}$ | 51 |
| 5.2 | The Condition Number $\kappa$ of the Tensor $D$ | 51 |
| 5.3 | Timing and Relative Error Results for Generating Random Displacement Vectors by Algorithm I, $Na/L = 1$ is Fixed | 52 |
| 5.4 | Timing and Relative Error Results for Generating Random Displacement Vectors by Algorithm I, with $L = 1000$ and $a = 0.1$ | 53 |
| 5.5 | Timing and Relative Error Results for Generating Random Displacement Vectors by Algorithm II, $Na/L = 1$ is Fixed | 54 |
| 5.6 | Timing and Relative Error Results for Generating Random Displacement Vectors by Algorithm II with $L = 1000$ and $a = 0.1$ | 54 |
| 5.7 | Timing and Relative Error Results for Generating Random Displacement Vectors by Algorithm III, $Na/L = 1$ is Fixed | 55 |
| 5.8 | Timing and Relative Error Results for Generating Random Displacement Vectors by Algorithm III with $L = 1000$ and $a = 0.1$ | 55 |
| 5.9 | Timing and Relative Error Results for Generating Random Displacement Vectors by Three Algorithm, $Na/L = 1$ is Fixed | 56 |
| 5.10 | Timing and Relative Error Results for Generating Random Displacement Vectors by Three Algorithm with $L = 1000$ and $a = 0.1$ | 57 |

## LIST OF FIGURES

**Figure** **Page**

2.1 FMM Data Structure: Box and Its Associated Lists . . . . . 17

# CHAPTER 1

# INTRODUCTION

In biophysics and biochemistry studies, the theory of Brownian motion was developed to describe the dynamic behavior of particles whose mass and size are much larger than those of solvent molecules. In the Fokker-Planck theory or Langevin equation based models, a configuration-dependent force field (due to interparticle interactions or external forces) was coupled with stochastic rules to update the location of each particle, which in turn lead to a displacement of the other particles and new configurations. Brownian dynamics simulation has been widely used to study the properties of dilute solutions of large molecules and colloidal particles.

For many particle systems in application, in addition to the commonly used short-ranged forces (e.g., the hard-sphere exclusion and Lennard-Jones forces) and electrostatic force, one also needs to consider the hydrodynamic effects between solvent molecules and Brownian particles in order to describe how the relative motion of the Brownian particles is coupled mechanically by the displaced solvent. Compared to a setup that does not consider hydrodynamics, the hydrodynamics interactions accelerate the dynamics of the particle system. The results presented in [9] demonstrates the importance of including hydrodynamic interactions in a dynamic simulation of many-particle Brownian systems. The hydrodynamic interaction is long-range and influences the dynamics of dilute polymer solutions [10, 11, 12]. There has been recent interest in the rheological and conformational properties of dilute solutions of DNA and other proteins [13, 14, 15]. Moreover, the hydrodynamic interaction profoundly influences the dynamics of diffusional encounters [16, 17] and the description of the transport properties of multisubunit structures in terms of subunit frictional coefficients [18, 19]. Computer simulations should be useful for studying certain

aspects of protein folding [20], particle coagulation, and other biochemical processes. However, when hydrodynamic interactions are included in a Brownian dynamics simulation, the random displacements become *correlated* [21], even though they still have the same (temperature dependent) magnitudes. In a numerical simulation, they now have to be determined from a factorization of the diffusion tensor of the complete system, which is numerically demanding. Here we would like to remark that Brady et al. [22, 23, 24] have developed fast algorithms for Stokesian dynamics simulation, which are $O(N \log N)$ methods.

In this dissertation, we consider the Ermak-McCammon algorithm [9, 25] for Brownian dynamics simulation, where the particles are assumed to be of spherical shape and the hydrodynamic interactions between $N$ particles are described by a $3N \times 3N$ diffusion tensor $D$. One of the popular choices for the diffusion tensor $D$ is the Rotne-Prager-Yamakawa tensor [1, 2], which will be defined in Chapter 2.

In the Ermak-McCammon algorithm, the total displacement $\Delta \mathbf{x}^m$ of the $m$th particle during a time step $\Delta t$ due to the force $\mathbf{F}^n$ and diffusion tensor $D$ is given by

$$\Delta \mathbf{x}^m (\Delta t) = \sum_n \frac{D(\mathbf{x}^m, \mathbf{x}^n)\mathbf{F}^n}{k_B T} \Delta t + \sum_n \frac{\partial D(\mathbf{x}^m, \mathbf{x}^n)}{\partial \mathbf{x}^n} \Delta t + \mathbf{R}^m (\Delta t) \tag{1.1}$$

where the hydrodynamically correlated random displacements $\mathbf{R}^m (\Delta t)$ are normally distributed with zero mean and finite covariance determined by the diffusion tensor $D$. To be more precise, we have

$$\langle \mathbf{R}^m (\Delta t) \rangle = 0, \quad \langle \mathbf{R}^m (\Delta t)\, \mathbf{R}^n (\Delta t) \rangle = 2D(\mathbf{x}^m, \mathbf{x}^n)\Delta t. \tag{1.2}$$

For the Rotne-Prager-Yamakawa tensor, which has the property

$$\sum_n \frac{\partial D(\mathbf{x}^m, \mathbf{x}^n)}{\partial \mathbf{x}^n} \Delta t \equiv 0 \tag{1.3}$$

it is shown in [9], so that this term can be dropped from (1.1). This greatly simplifies the calculation of the displacements as the gradient of the diffusion tensor does not have to be calculated. For computing the deterministic part of $\Delta\mathbf{x}^m$ $(m = 1, \cdots, N)$, i.e., the first term on the right side of (1.1), could be considered as calculating $D\mathbf{u}$ for a given vector $\mathbf{u}$. The classical algorithms require $O(N^2)$ operations and it could be computed in $O(N)$ or $O(N \log N)$ time using the fast multipole method or fast Fourier transform (see, for example, [26, 27, 28, 29, 30, 3, 31]). However, It is nontrivial to generate $3N$ normally distributed random vectors $\mathbf{R}^m$ $(m = 1, \cdots, N)$ with the particular correlation (1.2) efficiently. Indeed, the standard technique in statistics [32, 33] generates such a random vector in three steps. First, find the Cholesky factor $C$ of the diffusion matrix $D$ (i.e., $D = C^T \cdot C$ and $C$ is an upper triangular matrix). Second, generate an independent normally distributed random vector, say, $\mathbf{v}$. Third, multiply $C\sqrt{2\Delta t}$ with $\mathbf{v}$ and the resulting vector will be normally distributed with the correlation given by (1.2). The well-known algorithm for Cholesky factorization requires $O(N^3)$ operations and the third step for computing matrix-vector multiplication requires $O(N^2)$ operations via direct computation. Thus, generating the random vector $\mathbf{R}$ has become one of the bottlenecks in the large-scale Brownian dynamics simulations with hydrodynamic interactions.

The purpose of this dissertation is to discuss the fast algorithms for generating the total displacement $\Delta\mathbf{x}^m$. We first present two fast multipole methods (FMM) for computing the first term on the right side of (1.1). The first FMM is a simple application of the kernel independent FMM (KIFMM) developed by Ying, Biros, and Zorin [3], which requires 9 scalar FMM calls. The second FMM, similar to the FMM for Stokeslet developed by Tornberg and Greengard [4], decomposes the RPY tensor into harmonic potentials and its derivatives, and thus requires only four harmonic FMM calls. Both FMMs reduce the computational cost of $D\mathbf{u}$ from $O(N^2)$ to $O(N)$ for an arbitrary $N$-particle configuration.

Next we present three fast algorithms to generate the random displacement vector $\mathbf{R}$. We first observe that the Cholesky factor in the above algorithm can be replaced by any matrix $B$ (not necessarily lower triangular) which satisfies the equation $D = B \cdot B^T$ since (1.2) characterizes the random vector $R$. In particular, one could replace $C$ by the so-called square root matrix $\sqrt{D}$ defined by the equation $D = \sqrt{D}^2$. Of course the direct computation of the square root matrix $\sqrt{D}$ is probably as hard as that of $C$. However, note here that it is more than sufficient if we can compute $\sqrt{D}\mathbf{v}$ efficiently for an arbitrary vector $\mathbf{v}$. We observe that given an arbitrary vector $\mathbf{v}$, the fast multipole method can compute $D\mathbf{v}$ in $O(N)$ operations. And our strategy is as follows:

In Algorithm I and II, which are all based on the Krylov subspace approximations, we replace $\sqrt{D}\mathbf{v}$ by $p(D)\mathbf{v}$ with $p(D)$ a low degree polynomial in $D$. First we try to find an accurate and efficient matrix polynomial approximation for the square root matrix $\sqrt{D}$, that is, $\sqrt{D} \approx p_n(D)$ with $p_n$ a polynomial of low degree. This is possible since $D$ is positive definite. In the popular Chebyshev spectral approximation method (see, for example, [5, 6]), degree $n$ of the approximate polynomial depends logarithmically on the prescribed precision $\epsilon$ and is proportional to the square root of the condition number $\kappa$ of the matrix $D$, that is, $n \propto \log(\frac{1}{\epsilon})\sqrt{\kappa}$. In the Chebyshev spectral approximation method, one also needs to estimate the extreme eigenvalues of $D$. We have considered several methods: the classical Lanczos method, the Chebyshev-Davidson method, and the safeguarded Lanczos method proposed by Zhou and Li [7]. Our numerical experiments indicate that $\kappa$ is usually very small when the particles are distributed uniformly with low density, and that the safeguarded Lanczos method is most effective for our cases with very little additional computational cost. Thus, when combined with the FMMs we described earlier, the Chebyshev approximation method with the safeguarded Lanczos method as eigenvalue estimators

essentially reduces the cost of computing $\sqrt{D}\mathbf{v}$ from $O(N^3)$ to $O(N)$ for most practical particle configurations.

Finally in Algorithm III, we combine the so-called Spectral Lanczos Decomposition Method (SLDM) [8] and the FMM to compute $\sqrt{D}\mathbf{v}$ for an arbitrary vector $\mathbf{v}$. The SLDM, like the Chebyshev polynomial based method, tries to find an approximation to $\sqrt{D}\mathbf{v}$ in the Krylov subspace $K_k = \text{span}\{v, Dv, \cdots, D^k v\}$, but the method is based on the standard Lanczos iteration (see, for example, [34]) and does not require the estimation of extreme eigenvalues. Our numerical experiments show that the SLDM is generally more efficient that the popular Chebyshev spectral approximation method. The overall complexity of generating one such random vector $\mathbf{R}$ by all the algorithms is essentially linear (i.e., $O(N)$) considering the fact that $n$ is very small. Our technique will be incorporated into existing Brownian dynamics simulation packages, including the open source Browndye [35], and applications on biomolecular systems will be reported in the future.

The outline of this dissertation is as follows. The numerical tools needed for our algorithms are summarized in Chapter 2. In Chapter 3, we discuss the fast multipole methods for the Rotne-Prager-Yamakawa tensor. In Chapter 4, we present the details of our three fast algorithms. The performance of our fast algorithms is illustrated via several numerical examples in Chapter 5. Finally, we present a short conclusion and discuss possible extension and applications of our algorithm in Chapter 6.

# CHAPTER 2

# NUMERICAL PRELIMINARY

## 2.1 The Square Root of a Real, Symmetric, and Positive Definite Matrix

Suppose that $D$ is a real, symmetric, and positive definite matrix. Then $D$ admits the following decomposition:

$$D = BB^*. \tag{2.1}$$

This decomposition is not unique. Indeed, if B satisfies (2.1), then $B \cdot U$ also satisfies (2.1) for any unitary matrix U since $BU \cdot (BU)^* = BUU^*B^* = BIB^* = D$. Thus there are infinitely many matrices satisfying (2.1) and these matrices are associated with unitary transformations.

Nevertheless, there are two natural choices for $B$. One is the Cholesky factor $C$, a real upper triangular matrix. The Cholesky factorization is a standard algorithm for finding $C$, which is essentially the LU decomposition with the symmetry of the matrix taken into account and requires $\frac{1}{6}N^3$ operations. The other is the so-called square root matrix $\sqrt{D}$, which satisfies

$$D = \sqrt{D} \cdot \sqrt{D}. \tag{2.2}$$

$\sqrt{D}$ is also real, symmetric, and positive definite. If the eigenvalue decomposition of $D$ is $O\Lambda O^T$ with $O$ an orthogonal matrix and $\Lambda = \mathrm{diag}(\lambda_1, \cdots, \lambda_N)$ the eigenvalue matrix, then $\sqrt{D} = O\sqrt{\Lambda}O^T$, where $\sqrt{\Lambda} = \mathrm{diag}(\sqrt{\lambda_1}, \cdots, \sqrt{\lambda_N})$. Thus $\sqrt{D}$ is actually unique. There are many algorithms for computing $\sqrt{D}$ (see, for example, [36]). However, the explicit construction of such a matrix requires at least $O(N^2)$ operations for a general matrix.

The following lemma is interesting and somewhat surprising.

**Lemma 2.1.1.** *Suppose that $D$ is a real, symmetric, and positive definite matrix and that $\sqrt{D}$ is its square root defined by (2.2). Then there exists a polynomial $p(\cdot)$ such that $\sqrt{D} = p(D)$, and the degree of $p$ equals to the number of distinct eigenvalues of $D$ minus 1.*

*Proof.* Suppose $D = O\Lambda O^T$ with $O$ an orthogonal matrix and $\Lambda$ the eigenvalue matrix of $D$. Then $\sqrt{D} = O\sqrt{\Lambda}O^T$.

Now if $D$ has $k$ distinct eigenvalues $\lambda_1, \cdots, \lambda_k$, then there exist $c_0, \cdots, c_{k-1}$ such that

$$\sqrt{\lambda_i} = p(\lambda_i) = \sum_{j=0}^{k-1} c_j {\lambda_i}^j, \qquad i = 1, \cdots, k. \tag{2.3}$$

The existence and uniqueness of these $k$ coefficients $c_j$ ($j = 0, \cdots, k-1$) are guaranteed since the coefficient matrix in the above linear system (2.3) is a Vandermonde matrix. Thus, there exists a polynomial $p$ of degree $k-1$ such that $p(x) = \sqrt{x}$ for $x = \lambda_1, \cdots, \lambda_k$. Hence, $p(\Lambda) = \sqrt{\Lambda}$ where $\Lambda$ is the eigenvalue matrix of $D$ and $p(D) = Op(\Lambda)O^T = O\sqrt{\Lambda}O^T = \sqrt{D}$. □

However, though $\sqrt{D}$ is exactly equal to a polynomial $p$ in $D$, the degree of $p$ might be very large if $D$ has a large number of distinct eigenvalues. Thus, instead of trying to find the exact polynomial $p$ in $D$ which equals $\sqrt{D}$, we will try to find an approximate polynomial $p$ in $D$ so that $p(D)$ is very close to $\sqrt{D}$ and the degree of $p$ is fairly low. For this, we need the spectral approximation of the square root function.

## 2.2 Spectral Approximation Using Chebyshev Polynomials

The Chebyshev polynomial of degree $n$, denoted by $T_n$, is defined by the formula

$$T_n(x) = \cos(n \arccos x), \qquad x \in [-1, 1]. \tag{2.4}$$

They also satisfy the following recurrence relation

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_{n+1} &= 2xT_n(x) - T_{n-1}(x) \quad n \geq 1. \end{aligned} \tag{2.5}$$

The Chebyshev points are the zeros of $T_{n+1}(x)$ in $[-1, 1]$, given explicitly by

$$x_j = \cos\left[\frac{(2j+1)\pi}{2n+2}\right], \qquad j = 0, 1, \cdots, n. \tag{2.6}$$

Given a sufficiently smooth function $f$, we will denote the polynomial interpolating $f$ on the Chebyshev points by $p_n(x)$. The following theorem states that the Chebyshev polynomial approximation has spectral accuracy for functions analytic on $[-1, 1]$. It can be found in [37, 34].

**Theorem 2.2.1.** *Suppose that $f$ is analytic on $[-1, 1]$ and $p_n(x)$ is its Chebyshev polynomial interpolant which interpolates $f$ on the Chebyshev points. Suppose further that $z_0$ is the closest singular point of $f$ to the interval $[-1, 1]$. Then $p_n(z)$ converges to $f(z)$ exponentially fast for any $z$ inside the ellipse that passes through $z_0$ and has foci $\pm 1$. In particular, for $x \in [-1, 1]$,*

$$|f(x) - p_n(x)| = O(e^{-n(\phi(z_0)+\ln 2)}) \qquad as\ N \to \infty, \tag{2.7}$$

*where the potential function $\phi$ is defined by the formula*

$$\phi(z) = \log \frac{|z - \sqrt{z^2 - 1}|}{2}. \tag{2.8}$$

## 2.3 The Rotne-Prager-Yamakawa Tensor

For a chain of $N$ particles of radius $a$, the Brownian motion in the absence of external forces is governed by the Kirkwood-Riseman diffusion [38]. The diffusion tensor normally used to treat hydrodynamic interactions in polymer molecules is not

necessarily positive definite, an unphysical behavior which results from the neglect of short-range contributions to the interaction between chain segments. The problem can be reformulated as a minimization of the rate at which energy is dissipated by the motion of the suspending fluid, and in this way Rotne and Prager [1] have obtained an approximate diffusion tensor which is positive definite for all configurations of the polymer molecule, approaches the Kirkwood-Riseman diffusion tensor at large separations between the interacting segments, and can be written as the true diffusion tensor plus a positive definite correction. Then for examining the possible effects of the correction term on the transport properties of flexible chains, Yamakawa [2] has improved the approximate diffusion tensor which describes the interaction when the particles adhibit together. The Rotne-Prager-Yamakawa tensor is defined as follows, which is the most common choice to model the Stokes flow for two spheres and neglect the hydrodynamic rotation-rotation and rotation-translation coupling:

$$D(\mathbf{x}^m, \mathbf{x}^m) = \frac{k_B T}{6\pi\eta a}\mathbf{I}, \tag{2.9}$$

$$\begin{gathered} D(\mathbf{x}^m, \mathbf{x}^n) = \frac{k_B T}{8\pi\eta r_{mn}}[(\mathbf{I} + \frac{\mathbf{r}_{mn}\mathbf{r}_{mn}}{r_{mn}^2}) + \frac{2a^2}{3r_{mn}^2}(\mathbf{I} - 3\frac{\mathbf{r}_{mn}\mathbf{r}_{mn}}{r_{mn}^2})] \\ for \quad m \neq n \quad and \quad r_{mn} \geq 2a, \end{gathered} \tag{2.10}$$

$$\begin{gathered} D(\mathbf{x}^m, \mathbf{x}^n) = \frac{k_B T}{6\pi\eta a}[(1 - \frac{9}{32}\frac{r_{mn}}{a})\mathbf{I} + \frac{3}{32a}\frac{\mathbf{r}_{mn}\mathbf{r}_{mn}}{r_{mn}}] \\ for \quad m \neq n \quad and \quad r_{mn} < 2a. \end{gathered} \tag{2.11}$$

Here $k_B$ is the Boltzmann constant, $T$ is the absolute temperature, $\eta$ is the solvent viscosity, $a$ represents the hydrodynamic radius of each particle, $m$ and $n$ label particle indices, $I$ is the $3 \times 3$ identity matrix, $\mathbf{r}_{mn} = \mathbf{x}^m - \mathbf{x}^n$, and $r_{mn} = \sqrt{\mathbf{r}_{mn} \cdot \mathbf{r}_{mn}}$ with $\mathbf{x}^m$ the position vector of the $m$th particle.

Following for the sake of simplicity, we mark $\mathbf{r}_{mn} = (x, y, z)$ and $r = r_{mn} = \sqrt{x^2 + y^2 + z^2}$. Then we can get,

$$\begin{cases} x\dfrac{\partial}{\partial x}\left(\dfrac{1}{r}\right) = -\dfrac{x^2}{r^3} & y\dfrac{\partial}{\partial x}\left(\dfrac{1}{r}\right) = -\dfrac{xy}{r^3} & z\dfrac{\partial}{\partial x}\left(\dfrac{1}{r}\right) = -\dfrac{xz}{r^3} \\ x\dfrac{\partial}{\partial y}\left(\dfrac{1}{r}\right) = -\dfrac{xy}{r^3} & y\dfrac{\partial}{\partial y}\left(\dfrac{1}{r}\right) = -\dfrac{y^2}{r^3} & z\dfrac{\partial}{\partial y}\left(\dfrac{1}{r}\right) = -\dfrac{yz}{r^3} \\ x\dfrac{\partial}{\partial z}\left(\dfrac{1}{r}\right) = -\dfrac{xz}{r^3} & y\dfrac{\partial}{\partial z}\left(\dfrac{1}{r}\right) = -\dfrac{yz}{r^3} & z\dfrac{\partial}{\partial z}\left(\dfrac{1}{r}\right) = -\dfrac{z^2}{r^3} \end{cases} \tag{2.12}$$

$$\begin{cases} \dfrac{\partial^2}{\partial x^2}\left(\dfrac{1}{r}\right) = -\dfrac{1}{r^3} + \dfrac{3x^2}{r^5} & \dfrac{\partial^2}{\partial xy}\left(\dfrac{1}{r}\right) = -\dfrac{3xy}{r^5} & z\dfrac{\partial^2}{\partial xz}\left(\dfrac{1}{r}\right) = -\dfrac{3xz}{r^5} \\ \dfrac{\partial}{\partial yx}\left(\dfrac{1}{r}\right) = -\dfrac{3xy}{r^5} & \dfrac{\partial}{\partial y^2}\left(\dfrac{1}{r}\right) = -\dfrac{1}{r^3} + \dfrac{3y^2}{r^5} & \dfrac{\partial}{\partial yz}\left(\dfrac{1}{r}\right) = -\dfrac{3yz}{r^5} \\ \dfrac{\partial}{\partial zx}\left(\dfrac{1}{r}\right) = -\dfrac{3xz}{r^5} & \dfrac{\partial}{\partial zy}\left(\dfrac{1}{r}\right) = -\dfrac{3yz}{r^5} & \dfrac{\partial}{\partial z^2}\left(\dfrac{1}{r}\right) = -\dfrac{1}{r^3} + \dfrac{3z^2}{r^5} \end{cases} \tag{2.13}$$

combined with $\mathbf{r}_{mn}\mathbf{r}_{mn} = \begin{bmatrix} xx & xy & xz \\ xy & yy & yz \\ xz & yz & zz \end{bmatrix}$, the Rotne-Prager-Yamakawa tensor can be written as following form:

$$D(\mathbf{x}^m, \mathbf{x}^m) = C_0 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

$$D(\mathbf{x}^m, \mathbf{x}^n) = C_1 \left\{ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \frac{1}{r} - \begin{bmatrix} x\frac{\partial}{\partial x} & y\frac{\partial}{\partial x} & z\frac{\partial}{\partial x} \\ x\frac{\partial}{\partial y} & y\frac{\partial}{\partial y} & z\frac{\partial}{\partial y} \\ x\frac{\partial}{\partial z} & y\frac{\partial}{\partial z} & z\frac{\partial}{\partial z} \end{bmatrix} \frac{1}{r} \right\} - C_2 \begin{bmatrix} \frac{\partial^2}{\partial x^2} & \frac{\partial^2}{\partial x \partial y} & \frac{\partial^2}{\partial x \partial z} \\ \frac{\partial^2}{\partial x \partial y} & \frac{\partial^2}{\partial y^2} & \frac{\partial^2}{\partial y \partial z} \\ \frac{\partial^2}{\partial x \partial z} & \frac{\partial^2}{\partial y \partial z} & \frac{\partial^2}{\partial z^2} \end{bmatrix} \frac{1}{r}$$
$$for \quad m \neq n \quad and \quad r_{mn} \geq 2a, \tag{2.15}$$

$$D(\mathbf{x}^m, \mathbf{x}^n) = C_0 \left\{ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \frac{9}{32a} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} r - \frac{3}{32a} \begin{bmatrix} x\frac{\partial}{\partial x} & y\frac{\partial}{\partial x} & z\frac{\partial}{\partial x} \\ x\frac{\partial}{\partial y} & y\frac{\partial}{\partial y} & z\frac{\partial}{\partial y} \\ x\frac{\partial}{\partial z} & y\frac{\partial}{\partial z} & z\frac{\partial}{\partial z} \end{bmatrix} r \right\}$$
$$for \quad m \neq n \quad and \quad r_{mn} < 2a. \tag{2.16}$$

Where $C_0 = \frac{k_B T}{6\pi\eta a}$, $C_1 = \frac{k_B T}{8\pi\eta}$ and $C_2 = \frac{k_B T a^2}{12\pi\eta}$. This kind form of Rotne-Prager-Yamakawa tensor will be used for harmonic fast multipole method in Chapter 3.

## 2.4 A Brief Overview to the Fast Multipole Method

### 2.4.1 Introduction

Many methods in computational physics are based on the evolution of particle systems with pairwise interactions corresponding to potentials related to the fundamental solution of elliptic partial differential equations. The most important among these kernels is the single-layer Laplacian. Given $N$ particles each carrying a charge $q_j$ at location $y_j$ $(j = 1, \cdots N)$, the electrostatic potential field is described by

$$\phi(x_i) = \sum_{j=1, j\neq i}^{N} \frac{q_j}{\|x_i - y_j\|} \tag{2.17}$$

It also arises in the integral equation methods when a convolution $\int K(x,y)q(y)dy$ of the Green's function (kernel) $K(x,y)$ and a given density $q(y)$ is discretized using quadrature rules. Equation (2.17) can be equivalently represented as a matrix-vector multiplication, with the matrix having zeros on the diagonal and $\{1/\|x_i - y_j\|\}$ on the off-diagonals and the vector being $\{q_j\}$ for $i, j = 1, ..., N$. When a direct method is applied to the summation or the corresponding matrix-vector multiplication, $O(N^2)$ operations are required, which becomes a significant bottleneck for large-scale problems even on modern supercomputers. Indeed, the advance in computer architectures

requires innovative numerical algorithms. In particular, the asymptotically optimal $O(N)$ methods for the special structured matrix vector operations are in urgent need in science and engineering applications.

There have been many research efforts to develop $O(N)$ or $O(N \log N)$ algorithms for the summation in Equation (2.17), including the Fast Fourier Transform algorithms [39, 40, 41], multi-wavelet based schemes [42], multigrid multi-level methods [43], and the multipole expansion techniques [44, 45, 26, 27]. In [44], and [45] the "tree-code" algorithm, the complexity is reduced to $O(N \log N)$ by using a low order spherical harmonics expansion to represent the "far-field" of a cluster of particles and a downward (or upward) pass to transmit this "multipole" expansion to the "interaction" list. In 1987, by introducing the additional local expansion and using both the upward and downward passes, Greengard and Rokhlin invented the asymptotically optimal $O(N)$ fast multipole method (FMM) and applied it to many-body problems [26]. The FMM was elected as one of the top ten algorithms for the twentieth century and it has been successfully applied in many science and engineering fields such as computational electromagnetic, molecular dynamics, computational fluid and solid mechanics, etc.

Given $N$ source densities $\{q_i\}$ located at $N$ points $\{\mathbf{y}_i\}$ in $\mathbb{R}^d (d = 2, 3)$, we want to compute the potential $\{\phi_i\}$ at $N$ points $\{\mathbf{x}_i\}$ induced by a kernel $K$ (single layer, double layer or other kernels of a elliptic PDE) using the following relation:

$$\phi(\mathbf{x}_i) = \sum_{j=1}^{N} K(\mathbf{x}_i, \mathbf{y}_j) q(\mathbf{y}_j), \quad i = 1, \cdots, N. \tag{2.18}$$

The implementation of FMM depends on the analytic expansion of the kernel. Such expansions need to be carried differently for different kernels, which somewhat makes the implementation of efficient and accurate FMM accelerators tedious. The original FMM deals with the kernel $K(\mathbf{x}, \mathbf{y}) = \log |\mathbf{r}|$, with $\mathbf{r} = \mathbf{x} - \mathbf{y}$ in 2D and

$K(\mathbf{x}, \mathbf{y}) = 1/|\mathbf{r}|$ in 3D. Recently, there have been many so-called kernel independent fast multipole methods (KIFMMs) which aim at computing the summation (2.18) for a much broader class of kernels. The constraint on the kernel $K$ is fairly mild, requiring only that $K$ is hierarchically semi-separable (see, for example, [46]). In other words, $K$ is increasingly smooth for $\mathbf{x}$ further away from $\mathbf{y}$. To be more precise, the numerical rank of the off-diagonal submatrices of $K$ is very low regardless of their sizes.

Gimbutas and Rokhlin derived a modification of FMM in 2D applicable to non-oscillatory kernels [29]. In their scheme, the Taylor and Laurent expansions are replaced with tensor products of Legendre expansions, which are subsequently compressed using singular value decomposition (SVD). The advantage of this technique is that using SVD guarantees an optimal compression in the sense of $L^2$ form, hence the number of terms in the multipole expansion is minimal for a given approximation error.

Martinsson and Rokhlin proposed a scheme based on the so-called "skeletonization" for 1D problems [30]. For two sets of particles, one as the source and the other one as target, this scheme approximates the interaction matrix by a low-rank matrix to within some precision, say rank $k$. They then choose a subset of $k$ "proxy" sources to represent the source set, and another subset of $k$ target locations with the property that if the potential is known at these $k$ points, it can be interpolated to all of the remaining points.

A formulation of FMM for non-oscillatory kernels which are only known numerically was proposed by Fong and Darve [28]. This algorithm combines interpolation with SVD. Specifically, the far-field behavior of the kernel $K(x, y)$ is approximated by a Chebyshev interpolation scheme which results in a low-rank representation of the kernel. Then the multipole-to-local operator is to evaluate the field due to particles located at Chebyshev nodes, which is done using an SVD.

A kernel-independent FMM based on Fourier series expansions has been developed by Zhang and Huang [31] and applied to kernels with scaling property. The scheme only relies on the ability of kernel evaluation in a finite region, which extends the ideas of FMM to situations that the analytical kernel expansion is either unavailable or too expensive to compute.

Another 3D kernel independent fast multipole method has been developed by Ying et al. [3]. This algorithm is designed to generalize FMM to second-order constant coefficient nonoscillatory elliptic partial differential equations. Their scheme only requires the existence of the Green's function and relies on kernel evaluation. The potential generated by sources inside a box is represented by a continuous distribution of an equivalent density on a surface enclosing the box, which is found by matching its potential to the potential of the original sources at a surface in the far-field. This scheme is applicable to second-order constant coefficient non-oscillatory elliptical partial differential equations.

Generally speaking, all of these KIFMMs have $O(N)$ complexity for a prescribed precision; they are all based on a hierarchical tree structure; and their algorithmic structure are all quite similar to that of the original fast multipole method [26] with two types of expansions: multipole expansion which gives a low rank approximation for far-field interactions, local expansion which makes $O(N)$ algorithm possible (if we use far-field expansion alone, we will obtain an $O(N \log N)$ algorithm); and translation operators for converting multipole and local expansions between different levels and for converting multipole expansions to local expansions. For all of the kernel-independent FMM to be discussed, they are characterized by the basis used for kernel expansion and the corresponding translation operators.

### 2.4.2 Data Structure of Adaptive FMM

Given a set of $N$ points, we define the computational domain to be a box, the root box, large enough to contain all points. We construct a hierarchical tree (a quad-tree in 2D and an oct-tree in 3D) and refer to the tree nodes (squares in 2D and cubes in 3D) as boxes. If the particle distribution is uniform, a regular grid can be used; however, we are primarily interested in non-uniform particle distributions. In this case an adaptively refined grid is needed. The grid is recursively refined until the number of points in each leaf box is less than a fixed number $s$.

The root box is referred to as refinement level 0. Starting from level 0, we recursively obtain level $l+1$ through subdivision of the boxes at level $l$ with more than $s$ points into 4 boxes in 2D or 8 boxes in 3D of equal size. At each level of refinement, a table of non-empty boxes is maintained, so that once an empty box is encountered, its existence is forgotten and it is completely ignored by the subsequent process.

A box is called a *parent* box if it contains more than $s$ points. Otherwise, it is referred to as a *childless* box or *leaf* box. A box $C$ is said to be the *child* of box $B$ if it is obtained by a single subdivision of box $B$. On the other hand, such a box $B$ is the *parent* of the box $C$. Boxes resulting from the subdivision of a parent box are referred to as *siblings*. The *colleagues* of a box $B$ consist of the boxes at the same level of $B$ and adjacent to $B$, including box $B$ itself. Apparently, a given box can have up to 9 colleagues in 2D or 27 colleagues in 3D.

One key observation in the FMM algorithm is the so-called "low separation rank" property for the well-separated boxes in the tree structure. We say two sets $\{\mathbf{x}_i\}$ and $\{\mathbf{y}_i\}$ are *well-separated* if there exists points $\mathbf{x}_0$ and $\mathbf{y}_0$ and a real number $r > 0$ such that

$$
\begin{aligned}
&|\mathbf{x}_i - \mathbf{x}_0| < r \quad \forall i = 1, ..., m \\
&|\mathbf{y}_i - \mathbf{y}_0| < r \quad \forall i = 1, ..., n, and \\
&|\mathbf{x}_0 - \mathbf{y}_0| < c \cdot r
\end{aligned}
\tag{2.19}
$$

where $c$ is bounded below by some constant $c_0 > 2$. Two boxes $B_1$ and $B_2$ of the same size in the tree structure are said to be *well-separated* if they are at least one box of the same size apart, i.e., if any sets of points $\{\mathbf{x}_i\} \subset B_1$ and $\{\mathbf{y}_i\} \subset B_2$ are well-separated. The "information" in two well-separated boxes can be compressed either analytically using special basis functions, or numerically using singular value decomposition (SVD) before being sent out.

Well-separated boxes can also appear at different levels. Assume that a box $B$ can inherit information from its parents while traversing down the tree structure, which contains the information from all well-separated boxes of $B$'s parents. Then we can define the *interaction list* of $B$ as the union of boxes at the same level of $B$ that are well-separated from $B$, but whose parents are not well-separated from $B$'s parent. Clearly, the information received from $B$'s parent and the interaction list is equivalent to the information from all well-separated boxes of $B$.

For a given box $B$ in the adaptive tree structure, in Figure 2.1, we associate $B$ with four lists of other boxes, determined by their positions with respect to $B$. We store lists and pointers for parent-child relations for each box in the tree structure.

In the following, we give the detailed definition for each list:

- List $U$ of a box $B$ will be denoted by $U_B$. It is empty if $B$ is a parent box. If $B$ is leaf, $U_B$ consists of $B$ and all leaf boxes adjacent to $B$.

- List $V$ of a box $B$ will be denoted by $V_B$ and is formed by all the children of the colleagues of $B$'s parent that are well-separated from $B$. List $V$ is also referred to as the *interaction list*.

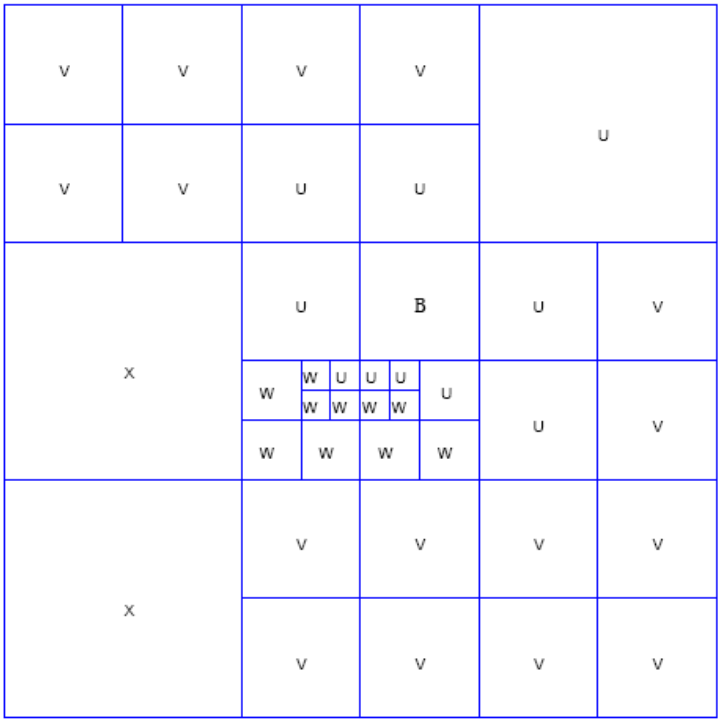


**Figure 2.1** FMM Data Structure: Box and Its Associated Lists

- List $W$ of a box $B$ will be denoted by $W_B$. $W_B$ is empty if $B$ is a parent box, and consists of all descendants of $B$'s colleagues whose parents are adjacent to $B$, but who are not adjacent to $B$ themselves, if $B$ is a leaf box.

- List $X$ of a box $B$ will be denoted by $X_B$ and is formed by all boxes $C$ such that $B \in W_C$. Note that all boxes in List $X$ are leaves and larger than $B$.

### 2.4.3 Approximations and Translations

Another important concept in the FMM algorithm is the extraction, storage, and transmission of the data or information based on the tree structure. In this section, using the single layer Laplacian kernel as an example, we introduce the analytical expansions and the translation operators which transmit the information from one box to another.

In two dimensions we have $K(\mathbf{x},\mathbf{y}) = -\frac{1}{2\pi}\log\rho$, with $\mathbf{r} = \mathbf{x} - \mathbf{y}$ and $\rho = |\mathbf{r}|$. In the FMM context it is convenient to use $K(\mathbf{x},\mathbf{y}) = Re(\log(z_x - z_y))$ where $z_x$ and $z_y$ are complex numbers corresponding to $\mathbf{x}$(target) and $\mathbf{y}$(source) points on the plane. The idea of FMM is to encode the potentials of a set of source densities using the *multipole expansion* and *local expansion* at places far away from these sources.

Suppose the source densities are supported in a disk centered at $z_c$ with radius $r$. Then for all $z$ outside the disk with radius $R(R > r)$, we can represent the potential at $z$ from the source densities using a set of coefficients $\{a_k, 0 \le k \ge p\}$.

(Multipole Expansion ($\tau_{SM}$ translation)): Suppose the $m$ source densities $\{q_j\}$ located at $\{z_j\}$, with $|z_j - z_C| < r$, then for any $|z - z_C| > R$, the induced potential $\phi(z)$ can be approximated by:

$$\phi(z) = a_0\log(z - z_C) + \sum_{k=1}^{p}\frac{a_k}{(z - z_C)^k} + O(\frac{r^p}{R^p}) \tag{2.20}$$

where $\{a_k, 0 \le k \le p\}$ satisfies

$$a_0 = \sum_{j=1}^{m} q_j \quad and \quad a_k = \sum_{j=1}^{m}\frac{-q_j(z_j - z_C)^k}{k}. \tag{2.21}$$

On the other hand, if the source densities are outside the disk with radius $R$, the potential at a point $z$ inside the disk with radius $r$ can be represented using a set of coefficients $\{c_k, 0 \le k \ge p\}$.

(Local Expansion ($\tau_{LT}$ translation)): Suppose the $m$ source densities $\{q_j\}$ located at $\{z_j\}$, with $|z_j - z_C| > R$, then for any $|z - z_C| < r$, the induced potential $\phi(z)$ can be approximated by:

$$\phi(z) = \sum_{k=0}^{p} c_k(z - z_C)^k + O(\frac{r^p}{R^p}) \tag{2.22}$$

where $\{c_k, 0 \leq k \leq p\}$ satisfies

$$c_0 = \sum_{j=1}^{m} q_j \log(z_C - z_j) \quad and \quad c_k = \sum_{j=1}^{m} \frac{-q_j}{k(z_j - z_C)^k}. \tag{2.23}$$

In both expansions, $p$ is usually a small constant determined from the desired accuracy of the result.

FMM employs the above representations in a recursive way. The computational domain, a box large enough to contain all source and target points, is hierarchically partitioned into a tree structure (a quadtree in 2D or an octtree in 3D). Each node of the tree corresponds to a geometric box (square or cube). The tree is constructed so that the leaves contain no more than a prespecified number of points. For each box, the potential induced by its source densities is represented using a multipole expansion, while the potential induced by the sources from non-adjacent boxes is encoded in a local expansion. The number of expansion terms $p$ is chosen so that, both expansions give an error which is less than a prescribed threshold.

Not only these expansions (multipole and local) can be used for efficient evaluation, but translations between these expansions are also available which make an $O(N)$ complexity algorithm possible. In particular, the following types of translations are used:

($\tau_{MM}$ translation): The *multipole to multipole* translation transforms the multipole expansions of a box's children to its own multipole expansion. Suppose $z_C$ is the center of a box and $z_M$ is the center of its parent. Suppose further $\{a_k\}$ is the multipole expansion at $z_C$, then the multipole expansion at $z_M$ can be written as:

$$\phi(z) = b_0 \log(z - z_M) + \sum_{l=1}^{p} \frac{b_l}{(z - z_M)^l} + O(\epsilon) \tag{2.24}$$

where $\{b_l, 0 \leq l \leq p\}$ satisfies

$$b_0 = a_0 \quad and \quad b_l = -\frac{a_0(z_C - z_M)^l}{l} + \sum_{k=1}^{l} a_k (z_C - z_M)^{l-k} \binom{l-1}{k-1}. \tag{2.25}$$

($\tau_{ML}$ translation): The *multipole to local* translation transforms the multipole expansion of a box to the local expansion of another non-adjacent box. Suppose $z_M$ and $z_L$ are the centers of two non-adjacent boxes on the same level, $\{b_l\}$ is multipole expansion at $z_M$. Then the local expanion at $z_L$ transformed can be written as:

$$\phi(z) = \sum_{l=0}^{p} d_l (z - z_L)^l + O(\epsilon) \tag{2.26}$$

where $\{d_l, 0 \le l \le p\}$ satisfies

$$\begin{aligned} d_0 &= b_0 \log(z_L - z_M) + \sum_{k=1}^{p} \frac{b_k}{(z_L - z_M)^k} \\ d_l &= -\frac{b_0}{l(z_M - z_L)^l} + \frac{1}{(z_M - z_L)^l} \sum_{k=1}^{p} \frac{b_k}{(z_L - z_M)^k} \binom{l+k-1}{k-1}. \end{aligned} \tag{2.27}$$

($\tau_{LL}$ translation): The *local to local* translation of the local expansion of a box's parent to its own local expansion. Suppose $z_T$ is the center of a box and $z_L$ the center of its parent. Suppose further $\{d_l\}$ is the local expansion at $z_L$, then the local expansion at $z_T$ can be written as:

$$\phi(z) = \sum_{l=0}^{p} e_l (z - z_T)^l + O(\epsilon) \tag{2.28}$$

where $\{e_l, 0 \le l \le p\}$ satisfies

$$e_l = \sum_{k=l}^{p} d_k (z_T - z_L)^{k-l} \binom{k}{l}. \tag{2.29}$$

Using the tree structure, FMM algorithmic structure consists of two basic steps. During the first step, the *upward pass*, the tree is traversed in postorder to compute the multipole expansion for each box. At the leaves, the multipole expansions are built following the kernel expansion, also called *source to multipole* $\tau_{SM}$ translation. At each non-leaf node, the multipole expansion is shifted from its children using the *multipole to multipole* $\tau_{MM}$ translation.

In the second step, the *downwards pass*, the tree is traversed in a preorder to compute the local expansion. For each box $B$, the local expansion is the sum of two parts. First, the *local to local* $\tau_{LL}$ translation collects the local expansion of $B$'s parent, the result condenses the contributions from the sources in all the boxes which are not adjacent to $B$'s parent. Second, the *multipole to local* $\tau_{ML}$ translation collects the multipole expansions of the boxes which are the children of the neighbors of $B$'s parent but are not adjacent to $B$. These boxes compose the interaction list of $B$. The sum of these two parts encodes all of the contributions from the sources in the boxes which are not adjacent to $B$ itself. At the end, the far filed interaction for each box, which is evaluated using the local expansion at this box, the *local to target* $\tau_{LT}$ translation, is combined with the near filed interaction evaluated by iterating over all of the source points in the neighborhood of the target box to obtain the potential.

Following we present a pseudo-code to explain the algorithmic structure of the FMM, assume $N$ is the total number of points, $s$ is the maximum number of points allowed in leaf box, we have:

The pseudo-code for the algorithm:

STEP 1 TREE CONSTRUCTION

**for** each box $B$ in preorder traversal of the tree **do**

  subdivide $B$ if $B$ has more than $s$ points in it

**end for**

**for** each box $B$ in preorder traversal of the tree **do**

  construct $U_B$,$V_B$,$W_B$ and $X_B$ for $B$

**end for**

STEP 2 UPWARDS PASS

**for** each leaf box $B$ in postorder traversal of the tree **do**

  evaluate $B$'s multipole expansion using $\tau_{SM}$

**end for**

**for** each non-leaf box $B$ in postorder traversal of the tree **do**

  add to $B$'s multipole expansion the contribution from its children box using $\tau_{MM}$

**end for**

STEP 3 DOWNWARDS PASS

**for** each non-root box $B$ in preorder traversal of the tree **do**

  add to B's local expansion the contribution from points in $X_B$ using $\tau_{LT}$

  add to B's local expansion the contribution from its parent box using $\tau_{LL}$

  add to B's local expansion the contribution from points in $V_B$ using $\tau_{ML}$

**end for**

**for** each leaf box $B$ in preorder traversal of the tree **do**

  calculate the potential at each point in $B$ from local expansion

  add to the potential due to all points in $W_B$ from multipole expansion

  add to the potential due to all points in $U_B$ from direct interaction

**end for**

# CHAPTER 3

# FAST MULTIPOLE METHODS FOR THE ROTNE-PRAGER-YAMAKAWA TENSOR

In this dissertation, we consider the calculation of the following sums:

$$u_i^m = \sum_{n=1}^{N} \sum_{j=1}^{3} D_{ij}(\mathbf{x}^m, \mathbf{x}^n) v_j^n, \quad i = 1, 2, 3, \quad m = 1, \cdots, N, \tag{3.1}$$

where $\mathbf{v}^n = (v_1^n, v_2^n, v_3^n)$ are vector source strengths. For notational convenience, we will write $\mathbf{u} = D\mathbf{v}$ with $\mathbf{u}$, $\mathbf{v}$ vectors of length $3N$ and $D$ a $3N \times 3N$ matrix. Clearly, the direct evaluation of $u_i^m$ for $i = 1, 2, 3$ and $m = 1, \cdots, N$ requires $O(N^2)$ work. Here, we present both the kernel independent FMM and the harmonic FMM for the sums in (3.1) that reduce the computational cost to $O(N)$.

## 3.1 Kernel Independent FMM

In our work, we use the KIFMM developed in [3] to speed up the computation of $D\mathbf{v}$ for an arbitrary vector $\mathbf{v}$. We refer the reader to the original award-winning paper [3] for a detailed description of the algorithm. Here we give a rough sketch on the KIFMM in [3]. The input of the KIFMM consists of the vector $\mathbf{v}$ and $N$ particle locations $\mathbf{x}^m$ $(m = 1, \cdots, N)$ which determine the Rotne-Prager-Yamakawa tensor $D$. The KIFMM first builds an adaptive octtree of boxes by successively dividing the box into its children so that the leaf boxes contain no more than a certain number of particles, with the top box containing all particles. Next, the KIFMM performs an upward pass starting from leaf boxes in which the far equivalent potential is constructed for each box summarizing the far field interaction due to the particles in that box. The KIFMM then performs a downward pass in which the local equivalent potential is constructed for each box, so that the interaction

due to particles in all well-separated boxes can be computed using that potential alone. Finally, the KIFMM will compute each entry of $D\mathbf{v}$ by summing over the local interactions directly and the far interactions using the local equivalent potential. The speed-up from $O(N^2)$ to $O(N)$ is achieved in the following two key steps. First, the equivalent potentials for any box need only a constant number of terms regardless of number of particles in that box and the size of the box. Second, three translation operators (multipole-to-multipole, multipole-to-local, and local-to-local) are utilized to constrcut those equivalent potentials in $O(N)$ time.

**Table 3.1** Relative Error and Timing Results of KIFMM

| $N$ | *relative error* | $T_N$ |
|---|---|---|
| 500 | 1.09979e-15 | 0.024001 |
| 1000 | 1.57043e-15 | 0.088006 |
| 2000 | 1.99245e-06 | 0.232015 |
| 4000 | 2.45951e-06 | 0.516033 |
| 8000 | 5.97456e-06 | 1.62018 |
| 10000 | 5.06958e-06 | 2.71617 |
| 100000 | 2.34039e-05 | 31.5622 |
| 1000000 | 9.21691e-05 | 317.968 |

In Table 3.1, we report the average timing results $T_N$ for computing the $N$ particle hydrodynamic interactions via the KIFMM in [3, 47]. The first column contains the number of particles. The second column contains the relative error with the direct summation as the reference result; when $N$ is large, the error is computed over 200 randomly chosen points. The third column contains the CPU time of the computation in seconds. In these experiments, we require 4-digit accuracy and set the

maximum number of the points allowed in a leaf box to 150, and maximum number of levels to 10.

*Remark* 3.1.1. We observe that $T_N$ grows approximately linearly with respect to the number of particles $N$. Since $D$ is a tensor, the hydrodynamic interactions between $N$ particles require 9 KIFMM calls.

## 3.2 FMM with Harmonic Potentials

Here we present an FMM for computing sums involving the Rotne-Prager-Yamakawa tensor. The method, similar to the approach in [4] for the Stokeslet, decomposes the tensor vector product into a sum of harmonic potentials and fields induced by four different charge and dipole distributions.

We begin with a brief overview of the harmonic FMM for Coulombic interactions. The FMM is an $O(N)$ scheme for the evaluation of sums of the form:

$$P^m(q,p,\mathbf{d}) = \sum_{n=1,n\neq m}^{N} \frac{q^n}{r_{mn}} + \sum_{n=1,n\neq m}^{N} \frac{(\mathbf{d}^n \cdot \mathbf{r}_{mn})p^n}{r_{mn}^3}, \quad m = 1,\cdots,N \tag{3.2}$$

$$F_i^m(q,p,\mathbf{d}) = \frac{\partial P^m(q,p,\mathbf{d})}{\partial x_i^m}, \quad i = 1,2,3, \quad m = 1,\cdots,N, \tag{3.3}$$

where $\mathbf{r}_{mn} = \mathbf{x}^m - \mathbf{x}^n$ and $r_{mn} = |\mathbf{r}_{mn}|$. The input data consists of the source locations $\mathbf{x}^n$, the charge strengths $q^n$, the dipole strengths $p^n$, and the orientation vectors $\mathbf{d}^n$ $(n = 1,\cdots,N)$. The output data consists of the quantities $P^m(q,p,\mathbf{d})$ and $F_i^m(q,p,\mathbf{d})$, which we will refer to as potentials and fields, respectively.

In practice, the FMM makes use of an adaptive hierarchical oct-tree as a data structure, refined until each leaf node contains only $O(1)$ sources. If we consider a particle with index $m$ that lies in a leaf node $B$, then we denote the nearest neighbors of $m$ by $nborlist(m)$. These are the sources (distinct from $m$) that lie either in $B$ or in a leaf node adjacent to $B$. (See [48] and the references therein for a more thorough discussion.) The harmonic FMM splits the above sums into a local part and a far

part:

$$P^m(q,p,\mathbf{d}) = P^m_{\text{loc}}(q,p,\mathbf{d}) + P^m_{\text{far}}(q,p,\mathbf{d}), \tag{3.4}$$

$$P^m_{\text{loc}}(q,p,\mathbf{d}) = \sum_{n\in\text{nborlist}(m)} \frac{q^n}{r_{mn}} + \sum_{n\in\text{nborlist}(m)} \frac{(\mathbf{d}^n\cdot\mathbf{r}_{mn})p^n}{r^3_{mn}}, \tag{3.5}$$

$$P^m_{\text{far}}(q,p,\mathbf{d}) = \sum_{n\notin\text{nborlist}(m)} \frac{q^n}{r_{mn}} + \sum_{n\notin\text{nborlist}(m)} \frac{(\mathbf{d}^n\cdot\mathbf{r}_{mn})p^n}{r^3_{mn}}. \tag{3.6}$$

Analogous definitions hold for the fields $F^m_i$.

The number of sources in $nborlist(m)$ is $O(1)$ and $P^m_{\text{loc}}$, $F^m_{i,\text{loc}}$ are both evaluated directly. The far field contributions $P^m_{\text{far}}$ and $F^m_{i,\text{far}}$, on the other hand, are evaluated via multipole and local expansions. The overall complexity of the harmonic FMM is $O(N)$. One convenient feature of the FMM is that the local and far field calculations are uncoupled. This makes it straightforward to modify the code so that it only computes the far field contributions $P^m_{\text{far}}$ and $F^m_{i,\text{far}}$, which will be convenient below.

Turning now to the RPY tensor and the corresponding fast multipole method (RPYFMM), we again split the sums (3.1) into two parts:

$$u^m_i = u^m_{i,\text{loc}} + u^m_{i,\text{far}} = \sum_{n\in\text{nborlist}(m)} + \sum_{n\notin\text{nborlist}(m)} . \tag{3.7}$$

For the sake of simplicity, we assume that $2a$ is sufficiently small compared with the dimensions of a leaf node that all interactions with $r_{mn} < 2a$ fall into the local part $u^m_{i,\text{loc}}$, which is evaluated directly. Thus, it remains to consider the calculation of the far field contributions to $D(\mathbf{x}^m,\mathbf{x}^n)\mathbf{v}$ under the assumption that $|\mathbf{x}^m - \mathbf{x}^n| \geq 2a$. This we accomplish through a decomposition into four separate harmonic potential and field evaluations.

We first rewrite the $ij$th entry $D_{ij}(\mathbf{x}^m, \mathbf{x}^n)$ for $|\mathbf{x}^m - \mathbf{x}^n| \geq 2a$ with (2.15) as follows:

$$\begin{aligned} D_{ij}(\mathbf{x}^m, \mathbf{x}^n) &= C_1 \left( \frac{\delta_{ij}}{|\mathbf{x}^m - \mathbf{x}^n|} + \frac{(x_i - y_i)(x_j - y_j)}{|\mathbf{x}^m - \mathbf{x}^n|^3} \right) \\ &\quad + C_2 \left( \frac{\delta_{ij}}{|\mathbf{x}^m - \mathbf{x}^n|^3} - \frac{3(x_i - y_i)(x_j - y_j)}{|\mathbf{x}^m - \mathbf{x}^n|^5} \right) \\ &= C_1 \left( \frac{\delta_{ij}}{|\mathbf{x}^m - \mathbf{x}^n|} - (x_j - y_j) \frac{\partial}{\partial x_i} \frac{1}{|\mathbf{x}^m - \mathbf{x}^n|} \right) + C_2 \frac{\partial}{\partial x_i} \frac{x_j - y_j}{|\mathbf{x}^m - \mathbf{x}^n|^3}, \end{aligned} \tag{3.8}$$

where $C_1 = \frac{k_B T}{8\pi\eta}$, $C_2 = \frac{k_B T a^2}{12\pi\eta}$.

Using this expression for $D_{ij}(\mathbf{x}^m, \mathbf{x}^n)$, we obtain

$$\begin{aligned} u^m_{i,\text{far}} &= \sum_{n \notin \text{nborlist}(m)} \sum_{j=1}^{3} D_{ij}(\mathbf{x}^m, \mathbf{x}^n) v^n_j \\ &= \sum_{n \notin \text{nborlist}(m)} \sum_{j=1}^{3} \left[ C_1 \left( \frac{\delta_{ij}}{r_{mn}} - (x^m_j - x^n_j) \frac{\partial}{\partial x^m_i} \frac{1}{r_{mn}} \right) + C_2 \frac{\partial}{\partial x^m_i} \frac{x^m_j - x^n_j}{r^3_{mn}} \right] v^n_j \\ &= \sum_{n \notin \text{nborlist}(m)} C_1 \left( \frac{v^n_i}{r_{mn}} - \sum_{j=1}^{3} x^m_j \frac{\partial}{\partial x^m_i} \frac{v^n_j}{r_{mn}} + \frac{\partial}{\partial x^m_i} \frac{\mathbf{x}^n \cdot \mathbf{v}^n}{r_{mn}} \right) + C_2 \frac{\partial}{\partial x^m_i} \frac{\mathbf{v}^n \cdot \mathbf{r}_{mn}}{r^3_{mn}} \\ &= C_1 \sum_{n \notin \text{nborlist}(m)} \frac{v^n_i}{r_{mn}} - C_1 \sum_{j=1}^{3} x^m_j \frac{\partial}{\partial x^m_i} \sum_{n \notin \text{nborlist}(m)} \frac{v^n_j}{r_{mn}} \\ &\quad + \frac{\partial}{\partial x^m_i} \left( \sum_{n \notin \text{nborlist}(m)} \frac{C_1 (\mathbf{x}^n \cdot \mathbf{v}^n)}{r_{mn}} + \sum_{n \notin \text{nborlist}(m)} \frac{C_2 (\mathbf{v}^n \cdot \mathbf{r}_{mn})}{r^3_{mn}} \right). \end{aligned} \tag{3.9}$$

Comparing this with (3.3) and (3.6), we have

$$u^m_{i,\text{far}} = C_1 P^m_{\text{far}}(v_i, 0, 0) - C_1 \sum_{j=1}^{3} x^m_j F^m_{i,\text{far}}(v_j, 0, 0) + F^m_{i,\text{far}}(C_1(\mathbf{x} \cdot \mathbf{v}), C_2, \mathbf{v}). \tag{3.10}$$

In short, to compute $u^m_{i,\text{far}}$, we need to call the harmonic FMM four times, using the source locations $\{\mathbf{x}^n\}$. For the first three calls ($i = 1, 2, 3$), we let $\{v^1_i, \cdots, v^N_i\}$ as charge strengths. For the fourth call, we let $\{C_1 \mathbf{x}^1 \cdot \mathbf{v}^1, \cdots, C_1 \mathbf{x}^N \cdot \mathbf{v}^N\}$ be the charge strengths, $\{C_2, \cdots, C_2\}$ be the dipole strengths, and $\{\mathbf{v}^1, \cdots, \mathbf{v}^N\}$ be the dipole orientation vectors. In the last call, only the fields are required on output.

**Algorithm 1** Fast Multipole Method for the Rotne-Prager-Yamakawa Tensor

**Comment:** Given a precision requirement $\epsilon$, source locations $\{\mathbf{x}^1, \cdots, \mathbf{x}^N\}$, vector source strengths $\{\mathbf{v}^1, \cdots, \mathbf{v}^N\}$, constants $C_1$, $C_2$, and $a$, compute the Rotne-Prager-Yamakawa tensor vector product defined in (3.1).

1: **for** $j = 1 : 3$ **do**

2: Call the harmonic FMM with tolerance $\epsilon$ and charge strengths $\{v_j^1, \cdots, v_j^N\}$. Compute the far field part of the potentials and fields.

3: **end for**

4: Call the harmonic FMM with tolerance $\epsilon$, charge strengths $\{C_1\mathbf{x}^1 \cdot \mathbf{v}^1, \cdots, C_1\mathbf{x}^N \cdot \mathbf{v}^N\}$, dipole strengths $\{C_2, \cdots, C_2\}$, and dipole orientation vectors $\{\mathbf{v}^1, \cdots, \mathbf{v}^N\}$. Compute the far field part of the fields only.

5: Compute the far part $u_{i,\text{far}}^m$ according to (3.10).

6: Call the harmonic FMM to reconstruct the *nborlist*.

7: Compute the local part $u_{i,\text{loc}}^m$ directly using (2.10) and (2.11).

8: Add $u_{i,\text{loc}}^m$ and $u_{i,\text{far}}^m$ to obtain $u_i^m$ $(i = 1, 2, 3,\ m = 1, \cdots, N)$ and if necessary, compute the relative error by comparing the result with the exact result obtained via direct computation on a few sampling points.

In Table 3.2, we report the average timing results $T_N$ for computing the $N$ particle hydrodynamic interactions via the RPYFMM in [49]. The first column contains the number of particles. The second column contains the relative error with the direct summation as the reference result; when $N$ is large, the error is computed over 200 randomly chosen points. The third column contains the CPU time of the computation in seconds. In these experiments, we require 4-digit accuracy.

**Table 3.2** Relative Error and Timing Results of RPYFMM

| $N$ | *relative error* | $T_N$ |
|---|---|---|
| 500 | 1.29675e-08 | 0.124006 |
| 1000 | 2.50972e-06 | 0.244021 |
| 2000 | 3.59364e-06 | 0.312022 |
| 4000 | 7.37061e-06 | 0.556034 |
| 8000 | 1.62885e-05 | 2.54818 |
| 10000 | 1.98404e-05 | 3.01224 |
| 100000 | 3.21268e-05 | 33.5138 |
| 1000000 | 3.46643e-05 | 331.919 |

*Remark* 3.2.1. Here we use the publicly available software package FMM3DLIB at *http://www.cims.nyu.edu/cmcl/fmm3dlib/fmm3dlib.html* for the harmonic FMM. This package is reasonably fast but not highly optimized. It assumes, for example, that all charge strengths are complex, and uses "point-and-shoot" translation operators instead of the diagonal translation operators of [48]. Optimization along these lines would yield a factor of 2-3 acceleration at low precision and an order of magnitude or more at high precision.

### 3.3 Numerical Results

Here we present some timing results for the Rotne-Prager-Yamakawa Tensor by kernel-independent and harmonic FMMs. All calculations were carried out on a laptop computer with 2.53 GHz CPU and 1.89 GB memory.

**Table 3.3** Timing results for uniform distribution in a cube.

| $N$ | $Prec$ | $T_{KIFMM}$ | $T_{RPYFMM}$ | $T_{Direct}$ | $E_{KIFMM}$ | $E_{RPYFMM}$ |
|---|---|---|---|---|---|---|
| 10000 | 3 | 2.71617 | 3.01224 | 8.00117 | 5.06958e-06 | 1.98404e-05 |
| 100000 | 3 | 31.5622 | 33.5138 | 680.052 | 2.34039e-05 | 3.21268e-05 |
| 1000000 | 3 | 317.968 | 331.919 | 69604.3 | 9.21691e-05 | 3.46643e-05 |
| 10000 | 6 | 10.8847 | 3.22422 | 7.99995 | 8.33041e-08 | 1.20964e-08 |
| 100000 | 6 | 100.278 | 47.8955 | 720.059 | 4.06636e-07 | 2.35994e-08 |
| 1000000 | 6 | 895.072 | 605.893 | 70807.4 | 1.92014e-06 | 2.66861e-08 |
| 10000 | 9 | 28.5351 | 5.15633 | 8.00128 | 2.86116e-09 | 7.26890e-10 |
| 100000 | 9 | 265.397 | 78.2052 | 719.992 | 6.08592e-09 | 1.85852e-09 |
| 1000000 | 9 | 2087.01 | 924.901 | 70007.5 | 2.66637e-08 | 2.01941e-09 |

Tables 3.3 and 3.4 show the results for computing $D\mathbf{v}$ using different methods for a uniform distribution in a cube, and a nonuniform distribution on a sphere, respectively. In each of these tables, the first column contains the number of particles. The second column contains the number of digits requested from the FMM. The third column contains the time required by the kernel-independent FMM (KIFMM) in [3]. The fourth column contains the time required by the harmonic FMM (RPYFMM) in

**Table 3.4** Timing results for nonuniform distribution on a sphere.

| $N$ | $Prec$ | $T_{KIFMM}$ | $T_{RPYFMM}$ | $T_{Direct}$ | $E_{KIFMM}$ | $E_{RPYFMM}$ |
|---|---|---|---|---|---|---|
| 10000 | 3 | 2.13213 | 1.75611 | 8.00016 | 2.11836e-05 | 1.98755e-05 |
| 100000 | 3 | 16.3694 | 21.9493 | 720.057 | 9.41318e-05 | 2.76244e-05 |
| 1000000 | 3 | 162.386 | 194.984 | 70404.2 | 4.55086e-04 | 2.83529e-05 |
| 10000 | 6 | 12.8128 | 4.32432 | 7.99995 | 2.74212e-07 | 1.85629e-08 |
| 100000 | 6 | 48.4713 | 56.6241 | 680.082 | 1.67471e-06 | 1.31362e-07 |
| 1000000 | 6 | 329.897 | 534.750 | 70401.4 | 4.89467e-06 | 3.41533e-07 |
| 10000 | 9 | 75.7567 | 7.76053 | 8.00091 | 6.31121e-09 | 1.30443e-08 |
| 100000 | 9 | 194.802 | 99.4217 | 720.058 | 2.73113e-08 | 7.51752e-08 |
| 1000000 | 9 | 752.019 | 891.092 | 70404.2 | 6.84249e-08 | 3.07682e-07 |

[49]. The fifth column contains the time required by the direct computation (which is obtained by extrapolating the time needed for twenty direct calculations). And the last two contain the relative $l_2$ error returned by the KIFMM and the RPYFMM, where the relative $l_2$ error is calculated by comparison with the direct calculations at twenty randomly selected source points.

*Remark* 3.3.1. We observe that both the KIFMM and the RPYFMM scale approximately linearly, $O(N)$, as expected. The timings are somewhat erratic due to the fact that the adaptive FMM builds a different data structure for each precision for each source distribution.

*Remark* 3.3.2. Unlike the approach based on the KIFMM in [3], which requires nine scalar FMM calls, the RPYFMM requires only four scalar harmonic FMM calls, so that any improvements to the harmonic FMM itself will automatically speed up the RPYFMM. At same time, both of the FMMs can compute $D\mathbf{v}$, but the RPYFMM also could easy to get the calculation of the following sums:

$$\frac{\partial u_i^m}{\partial x_k^n} = \sum_{n=1}^{N} \sum_{j=1}^{3} \frac{\partial D_{ij}(\mathbf{x}^m, \mathbf{x}^n)}{\partial x_k^n} v_j^n, \quad i = 1, 2, 3, \quad m = 1, \cdots, N. \tag{3.11}$$

Where the KIFMM requires twenty-seven scalar FMM calls, the RPYFMM still requires only four scalar harmonic FMM calls. It is very useful and more efficient in other kinds of simulation.

# CHAPTER 4

# FAST ALGORITHMS FOR GENERATING RANDOM DISPLACEMENT VECTORS

## 4.1 Algorithm I: Lanczos Method with Chebyshev-Davidson Method plus Chebyshev Spectral Approximation

The Chebyshev polynomial approximation for the square root matrix has been applied to the Brownian dynamics simulation by some researchers. Fixman [5] seems to be the earliest one to propose this method. It has been used later by other researchers (see, e.g., [25, 6]). Our contributions to this problem are that we have rigorously shown that the number of terms needed in the Chebyshev approximation depends logarithmically on the desired precision, and linearly on the square root of the condition number of the tensor, and that we have used the fast multipole method to reduce the complexity of the algorithm from $O(N^2)$ to essentially $O(N)$.

### 4.1.1 Estimating the Extreme Eigenvalues of $D$

We use the Lanczos method [50] combined with the FMM to estimate the largest and smallest eigenvalues of $D$. Starting from an arbitrary vector $v$, the Lanczos method successively computes $Dv, D^2v, \cdots, D^kv$, constructs an orthogonal basis $Q_k$ for the Krylov subspace $K_k = \text{span}\{v, Dv, \cdots, D^kv\}$, and forms a much smaller $k \times k$ tridiagonal matrix $T_k = Q_k^T D Q_k$ using three term recurrence. It then applies any standard algorithm to compute the eigenvalues and corresponding eigenvectors of $T_k$ and take $\lambda_{\max}(D) \approx \lambda_{\max}(T_k)$. During this process, the most expensive step, i.e., the calculation of $D^jv$ $(j = 1, \cdots, k)$ is done via the FMM. The computational cost of other steps is negligible since they involve much smaller matrices. Thus the overall complexity of the algorithm is $O(kN)$, where $N$ is the size of $D$ and $k$ is the number of Lanczos steps.

We summarize the above algorithm in Algorithm 2.

**Algorithm 2** Estimating the eigenvalue upper bound using the Lanczos method and FMM

Generate a random vector $v$ and normalize it $v_1 = v/\|v\|$.
Set $v_0 = 0$, $\beta_1 = 0$, $tol = 10^{-3}$ and $p = 12$.
**for** $k = 1, 2, ..., p$ **do**
  Use FMM to compute $Dv_k$ and set $w_j = Dv_k - \beta_k v_{k-1}$.
  Compute $\alpha_k = w_k \cdot v_k$.
  Set $w_k = w_k - \alpha_k v_k$ and $\beta_{k+1} = \|w_k\|$.
  Set $v_{k+1} = w_k/\beta_{k+1}$.
  **if** $k \geq 4$ **then**
    Construct a tridiagonal matrix $T_k$ with the diagonals equal to $(\alpha_1, \cdots, \alpha_k)$ and super- and sub-diagonals equal to $(\beta_2, \cdots, \beta_k)$.
    Use any standard method to compute the eigenvalues $\mu_1 \leq \cdots \mu_k$ and associated eigenvectors $z_1, \cdots, z_k$ of $T_k$.
    Compute $Ub_k = \lambda_{\max}(T_k)$.
    **if** $|Ub_k - Ub_{k-1}|/Ub_{k-1} < tol$ **then**
      Set $\lambda_{\max}(D) = Ub_k$, and **return**.
    **end if**
  **end if**
**end for**
Set $\lambda_{\max}(D) = \lambda_{\max}(T_p)$ and $\lambda_{\min}(D) = \lambda_{\min}(T_p)$.

Algorithm 2 is fairly effective in estimating the largest eigenvalue of $D$. Indeed, the size of the tridiagonal matrix $T_k$ is usually very small. We found that $k \leq 5$ is suffcient to achieve 3-digit accuracy for our numerical experiments. However, though it can also be used to estimate the smallest eigenvalue of $D$ with $\lambda_{\min}(D) = \lambda_{\min}(T_k)$,

it is not effcient. In order to estimate the smallest eigenvalue of $D$ effciently, we use a simplifed Chebyshev-Davidson method developed in [51].

**Algorithm 3** Estimating the eigenvalue lower bound using the Chebyshev-Davidson method and FMM

1: Use Algorithm 2 to obtain estimates about the smallest and largest eigenvalues of $D$ and denote them by $a$ and $b$, respectively.
2: Generate a random vector $v$ and normalize it $v = v/\|v\|$.
3: Set $c = (a+b)/2$ and $e = (b-a)/2$.
4: Use FMM and the Chebyshev recurrence relation to compute $Av, A^2v, \cdots, A^kv$ with $A = T_l\left(\frac{D-cI}{e}\right)$.
5: Form a $N \times k$ matrix $B = [v, Av, A^2v, \cdots, A^kv]$.
6: Do QR decomposition for $B$ so that $B = \tilde{Q}_k R$.
7: Form a $k \times k$ matrix $\tilde{T}_k = \tilde{Q}_k^T D \tilde{Q}_k$.
8: Find the eigenvalues of $\tilde{T}_k$ using any standard eigensolver.
9: Set $\lambda_{\min}(D) \approx \lambda_{\min}(\tilde{T}_k)$.

The basic idea of our algorithm is as follows. We first use Algorithm 2 to obtain a rough estimate of $\lambda_{\min}(D)$ (which is always greater than the exact value of $\lambda_{\min}(D)$) and an accurate estimate of $\lambda_{\max}(D)$. We will denote them by $a$ and $b$ correspondingly. Now, instead of considering the Krylov subspace $K_k = \text{span}\{v, Dv, \cdots, D^kv\}$, we will consider the transformed Krylov subspace $K_k = \text{span}\{v, Av, \cdots, A^kv\}$, where $A = T_l\left(\frac{D-cI}{e}\right)$. Here $T_l$ is the Chebyshev polynomial of degree $l$, $c = (a+b)/2$ and $e = (b-a)/2$. It is well known that Chebyshev polynomials are bounded by 1 on $[-1, 1]$, but grow very fast outside $[-1, 1]$. Thus, $T_l((x-c)/e)$ is bounded by 1 on $[a, b]$, but grows quickly outside $[a, b]$. Therefore, the purpose of the above transformation is to map the smallest eigenvalue of $D$ to the largest eigenvalue of $A$. The simplified Chebyshev-Davidson algorithm then constructs an orthogonal basis

$\tilde{Q}_k$ for $\tilde{K}_k$, computes the eigenvalues of $\tilde{T}_k = \tilde{Q}_k^T D \tilde{Q}_k$, and set $\lambda_{\min}(D) \approx \lambda_{\min}(\tilde{T}_k)$. The algorithm is summerized as Algorithm 3.

The overall complexity of the Algorithm 3 is $O(dN)$, where $N$ is the size of $D$ and $d = l(k+1)$. In our numerical experiments, we found that $l = 2$ and $k = 4$ are generally suffcient to achieve 3-digit accuracy for $\lambda_{\min}(D)$, where $d = 10$. However the cost of computational time for $\lambda_{\min}(D)$ is much more than $\lambda_{\max}(D)$.

### 4.1.2 Chebyshev Spectral Approximation for Computing Matrix Square Root

We now consider the Chebyshev polynomial approximation for the square root of $D$. First, we have the following lemma concerning the Chebyshev polynomial approximation for the simple square root function $\sqrt{x}$ on $[a, b]$ with $a > 0$.

**Lemma 4.1.1.** *Let* $y_j = \frac{b+a}{2} + \frac{b-a}{2}\cos(\frac{(2j+1)\pi}{2n+2})$ , $j = 0, 1, \cdots, n$ *be the transformed Chebyshev points on* $[a, b]$. *Suppose that* $p_n(x)$ *is the polynomial interpolating* $\sqrt{x}$ *on* $y_j$, $j = 0, 1, \cdots, n$. *Then for* $x \in [a, b]$,

$$|\sqrt{x} - p_n(x)| = O\left(\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n\right) \qquad as\ n \to \infty, \tag{4.1}$$

*where* $\kappa = b/a$.

*Proof.* We first use a linear transformation to transform $[a, b]$ back to $[-1, 1]$. That is,

$$\sqrt{x} = \sqrt{\frac{b+a}{2} + \frac{b-a}{2}z} \tag{4.2}$$

and $x \in [a, b] \Longleftrightarrow z \in [-1, 1]$. According to Theorem 2.2.1, we have

$$|\sqrt{\frac{b+a}{2} + \frac{b-a}{2}z} - p_n(z)| = O(e^{-n(\phi(z_0)+\log 2)}) \qquad as\ n \to \infty, \tag{4.3}$$

where $\phi(z) = \log \frac{|z-\sqrt{z^2-1}|}{2}$ and $z_0$ is the closest singular point of the complex function $\sqrt{\frac{b+a}{2} + \frac{b-a}{2}z}$ to $[-1, 1]$.

Now the closest singular point of $\sqrt{\frac{b+a}{2}+\frac{b-a}{2}z}$ to $[-1,1]$ is obviously the branch point of this square root function. That is, $z_0=-\frac{b+a}{b-a}$. Thus, we have

$$\begin{aligned} e^{-n(\phi(z_0)+\log 2)} &= e^{-n\left(\log\frac{|z_0-\sqrt{z_0^2-1}|}{2}+\log 2\right)} \\ &= \left(\frac{b+a}{b-a}+\sqrt{\left(\frac{b+a}{b-a}\right)^2-1}\right)^{-n} \\ &= \left(\frac{\kappa+1}{\kappa-1}+\sqrt{\left(\frac{\kappa+1}{\kappa-1}\right)^2-1}\right)^{-n} \\ &= \left(\frac{\kappa+1+\sqrt{4\kappa}}{\kappa-1}\right)^{-n} \\ &= \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-n} \\ &= \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{n}. \end{aligned} \tag{4.4}$$

Substituting (4.4) into (4.3), we obtain the desired result (4.1). □

We are now ready to state our main analytical result.

**Theorem 4.1.2.** *Suppose that $p_n(x)$ is the polynomial interpolating $\sqrt{x}$ on the scaled and shifted Chebyshev points $y_j=\frac{\lambda_{\max}(D)+\lambda_{\min}(D)}{2}+\frac{\lambda_{\max}(D)-\lambda_{\min}(D)}{2}\cos(\frac{(2j+1)\pi}{2n+2})$, $j=0,1,\cdots,n$. Then*

$$\|\sqrt{D}-p_n(D)\|_2 = O\left(\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n\right) \qquad \text{as } n\to\infty, \tag{4.5}$$

*where $\|\cdot\|_2$ is the matrix 2-norm and $\kappa=\lambda_{\max}(D)/\lambda_{\min}(D)$ is the 2-norm condition number of $D$. Thus, the degree of the polynomial satisfies the following relation:*

$$n = O\left(\sqrt{\kappa}\log\frac{1}{\epsilon}\right) \qquad \text{as } \kappa\to\infty, \tag{4.6}$$

*where $\epsilon$ is the prescribed precision.*

*Proof.* Let the eigenvalue decomposition of $D$ be $D = O\Lambda O^T$ with $O$ an orthogonal matrix and $\Lambda$ the eigenvalue matrix. Then $\sqrt{D} = O\sqrt{\Lambda}O^T$ and $p_n(D) = Op_n(\Lambda)O^T$. Thus, we have

$$\|\sqrt{D} - p_n(D)\|_2 = \|\sqrt{\Lambda} - p_n(\Lambda)\|_2 = O\left(\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n\right) \qquad \text{as } n \to \infty, \tag{4.7}$$

where the first equality follows from the fact that the 2-norm is invariant under the orthogonal transformation and the second equality follows from (4.1). This shows that in order to achieve a prescribed precision $\epsilon$ for the polynomial approximation, we need

$$\begin{aligned} n &= O(\log \epsilon / \log\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)) \\ &= O(\log \epsilon / \log\left(1 - \frac{2}{\sqrt{\kappa}}\right)) \\ &= O(\sqrt{\kappa} \log \frac{1}{\epsilon}), \qquad \text{as } \kappa \to \infty. \end{aligned} \tag{4.8}$$

□

*Remark* 4.1.3. The error estimate (4.5) for the polynomial approximation of the square root matrix is very similar to that of the Conjugate Gradient (CG) method (see, for example, page 299 in [52]) for solving a linear system. Thus, the above theorem shows that computing $\sqrt{A}b$ via the Chebyshev polynomial approximation takes about the same number of operations as computing $A^{-1}b$ via the CG method.

We now present some details about computing $\sqrt{D}v$ using the Chebyshev spectral approximation of $\sqrt{D}$. Let $p_n(x)$ be the Chebyshev polynomial approximation of the scalar function $\sqrt{x}$ over the range $[\lambda_{\min}, \lambda_{\max}]$. Then $p_n(x)$ can be expressed as

$$p_n(x) = \sum_{k=0}^{n} c_k \tilde{T}_k(x), \tag{4.9}$$

where the expansion coefficients $c_k$ are given by

$$c_k = \frac{2}{n}\sum_{l=1}^{n}\sqrt{x_l}\tilde{T}_k(x_l), \tag{4.10}$$

$$x_l = \frac{\lambda_{\max}(D)+\lambda_{\min}(D)}{2} + \frac{\lambda_{\max}(D)-\lambda_{\min}(D)}{2}\cos(\frac{(2l+1)\pi}{2n+2}), \tag{4.11}$$

and the scaled and shifted Chebyshev polynomials $\tilde{T}_k$ satisfy the following recurrence relation

$$\begin{aligned} &\tilde{T}_0 = 1, \\ &\tilde{T}_1(x) = t_a x + t_b, \\ &\tilde{T}_{l+1}(x) = 2(t_a x + t_b)\tilde{T}_l(x) - \tilde{T}_{l-1}(x) \end{aligned} \tag{4.12}$$

with

$$\begin{aligned} t_a &= \frac{2}{\lambda_{max}-\lambda_{min}}, \\ t_b &= -\frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}}. \end{aligned} \tag{4.13}$$

Obviously, we have

$$p_n(D) = \sum_{k=0}^{n} c_k \tilde{T}_k(D), \tag{4.14}$$

and thus

$$p_n(D)v = \sum_{k=0}^{n} c_k \tilde{T}_k(D)v = \sum_{k=0}^{n} c_k v_k, \tag{4.15}$$

where $v_k$ $(k = 0, 1, \cdots, n)$ can be computed via the following recurrence relation:

$$\begin{aligned} &v_0 = v, \\ &v_1 = t_a D v_0 + t_b v_0, \\ &v_{k+1} = 2(t_a D v_k + t_b v_k) - v_{k-1}. \end{aligned} \tag{4.16}$$

Here the FMM is applied to calculate $Dv_k$ at each step. Thus the complexity of computing $p_n(D)v$ is $O(nN)$, with $n$ the degree of the approximating Chebyshev polynomial and $N$ the size of the matrix $D$.

We now summarized our algorithm as follows.

---

**Algorithm 4** Chebyshev spectral approximation I for computing $\sqrt{D}v$

---

STEP 1. Precomputation stage

a. Use Algorithm 2 and Algorithm 3 to estimate $\lambda_{\max}(D)$ and $\lambda_{\min}(D)$.

b. Compute the Chebyshev coefficients $c_k$ $(k = 0, 1, \cdots, n)$ and determine the degree of the Chebyshev expansion $n$ so that it has the prescribed precision $\epsilon$ for $\sqrt{x}$ on $[\lambda_{\min}(D), \lambda_{\max}(D)]$.

STEP 2. Computation of the matrix square root

a. Use recurrence relation (4.16) and FMM to compute $v_k$, $k = 1, \cdots, n$.

b. Compute $p_n(D)v = \sum_{k=0}^{n} c_k v_k$ and set $\sqrt{D}v \approx p_n(D)v$.

---

Obviously, the precomputation stage requires $O((m + d)N)$ operations with $m$ the number of Lanczos steps and $d$ depends on the parameter set in Chebyshev-Davidson method. For Brownian dynamics simulation, the second step requires $O(nN)$ operations with $n$ the degree of the approximating Chebyshev polynomial for $\sqrt{D}$. Thus, the total computational cost is $O((m + d + n)N)$ for Brownian dynamics simulation. In general, since $m$ and $d$ are usually very small numbers in practice (say, $m$ less than 5 and $d = 10$), and $n$ is proportional to $\sqrt{\kappa}$ as shown in Theorem 4.1.2, the computational cost of our algorithm is $O(\sqrt{\kappa}N)$.

## 4.2 Algorithm II: Safeguarded Lanczos Method plus Chebyshev Spectral Approximation

### 4.2.1 Estimating the Extreme Eigenvalues of $D$

Note that there is no need to obtain a very accurate estimate of the extreme eigenvalues of $D$ since what we really need is an interval $[a, b]$ which contains all the eigenvalues

of $D$ but not far from $[\lambda_{\min}, \lambda_{\max}]$. Thus, two or three digits of accuracy suffices for our purpose. We can also observe that the cost of computational time for $\lambda_{\min}(D)$ from the Chebyshev-Davidson method is much more than $\lambda_{\max}(D)$.

To simplify the overall algorithm and reduce the computational time, we use the safeguarded Lanczos method in [7] combined with the FMM to estimate the extreme eigenvalues of $D$. As with the plain Lanczos method [50] above, we form a $k \times k$ tridiagonal matrix $T_k = Q_k^T D Q_k$ and apply any standard algorithm to compute the eigenvalues and corresponding eigenvectors of $T_k$. Finally, the estimates of the extreme eigenvalues of $D$ are given by $\lambda_{\max}(D) \approx \lambda_{\max}(T_k) + |e_k^T z_k|\beta_{k+1}$, $\lambda_{\min}(D) \approx \lambda_{\min}(T_k) - |e_k^T z_1|\beta_{k+1}$, respectively. Here $e_k$ is the kth column of the $k \times k$ identity matrix, $z_k$ is the eigenvector associated with the largest eigenvalue of $T_k$, $z_1$ is the eigenvector associated with the smallest eigenvalue of $T_k$, and $\beta_{k+1}$ is the last subdiagonal element of $T_{k+1}$. During this process, the most expensive step, i.e., the calculation of $D^j v$ $(j = 1, \cdots, k)$ is done via the FMM. The computational cost of other steps is negligible since they involve much smaller matrices. Thus the overall complexity of the algorithm is $O(kN)$, where $N$ is the size of $D$ and $k$ is the number of Lanczos steps.

We summarize the above algorithm in Algorithm 5.

*Remark* 4.2.1. We observe that the number of Lanczos steps is usually less than 8 in most of our testing cases. We would also like to remark here that [7] actually provides several safeguard terms with different convergence properties. We have used Equation 2.6 in [7] since we find that it has the best performance for our problem.

### 4.2.2 Chebyshev Spectral Approximation for Computing Matrix Square Root

Here we also use the Chebyshev spectral approximation to compute $\sqrt{D}v$. However, we observed that when $v$ is a normally distributed random vector with mean zero,

**Algorithm 5** Estimating the eigenvalue bounds using the safeguarded Lanczos method and FMM

1: Generate a random vector $v$ and normalize it $v_1 = v/\|v\|$.
2: Set $v_0 = 0$, $\beta_1 = 0$, $tol = 10^{-3}$ and $p = 12$.
3: **for** $k = 1, 2, ..., p$ **do**
4: Use FMM to compute $Dv_k$ and set $w_j = Dv_k - \beta_k v_{k-1}$.
5: Compute $\alpha_k = w_k \cdot v_k$.
6: Set $w_k = w_k - \alpha_k v_k$ and $\beta_{k+1} = \|w_k\|$.
7: Set $v_{k+1} = w_k/\beta_{k+1}$.
8: **if** $k \geq 4$ **then**
9: Construct a tridiagonal matrix $T_k$ with the diagonals equal to $(\alpha_1, \cdots, \alpha_k)$ and super- and sub-diagonals equal to $(\beta_2, \cdots, \beta_k)$.
10: Use any standard method to compute the eigenvalues $\mu_1 \leq \cdots \mu_k$ and associated eigenvectors $z_1, \cdots, z_k$ of $T_k$.
11: Compute $Ub_k = \lambda_{\max}(T_k) + |e_k^T z_k|\beta_{k+1}$, $Lb_k = \lambda_{\min}(T_k) - |e_k^T z_1|\beta_{k+1}$.
12: **if** $|Ub_k - Ub_{k-1}|/Ub_{k-1} < tol$ **then**
13: Set $\lambda_{\max}(D) = Ub_k$, $\lambda_{\min}(D) = Lb_k$ and **return**.
14: **end if**
15: **end if**
16: **end for**
17: Set $\lambda_{\max}(D) = \lambda_{\max}(T_p) + |e_p^T z_p|\beta_{p+1}$, $\lambda_{\min}(D) = \lambda_{\min}(T_p) - |e_p^T z_1|\beta_{p+1}$.

as is the case of Brownian dynamics simulations, we could combine the step of estimating extreme eigenvalues of $D$ with the step of computing $p_n(D)v$ to reduce the computational cost. To be more precise, we could use exactly the same vector $v$ to estimate the extreme eigenvalues of $D$. Obviously, the Lanczos iteration will generate a sequence of orthonormal vectors $q_1, \cdots, q_m$ which form a basis for the Krylov subspace $K_m = \text{span}\{v, Dv, \cdots, D^m v\}$. But $\tilde{T}_k(D)v$ $(k = 1, \cdots, m)$ are also in the same Krylov space $K_m$. Thus, $\tilde{T}_k(D)v$ $(k = 1, \cdots, m)$ can be obtained by using $q_1, \cdots, q_m$ and solving a small linear system of size $m$.

After using Lanczos iteration, we obtain a sequence of orthonormal vectors $q_1, q_2, \cdots, q_m, q_{m+1}$ and form a $m \times m$ tridiagonal matrix

$$T_m = Q_m^T D Q_m = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ & & \beta_{m-1} & \alpha_m \end{bmatrix}. \tag{4.17}$$

Then from the Equation 36.6 in [34], we have

$$DQ_m = Q_{m+1} S_m, \tag{4.18}$$

where

$$\begin{aligned} Q_m &= [q_1\, q_2\, \cdots\, q_m]_{n\times m} \\ Q_{m+1} &= [q_1\, q_2\, \cdots\, q_m\, q_{m+1}]_{n\times(m+1)} \\ S_m &= \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ & & \beta_{m-1} & \alpha_m \\ & & & \beta_m \end{bmatrix} = \begin{bmatrix} & & T_m & \\ & & & \\ 0 & \cdots & 0 & \beta_m \end{bmatrix}_{(m+1)\times m} \end{aligned} \tag{4.19}$$

Let

$$Q_m = C_m P_m \tag{4.20}$$

where $P_m$ is an $m \times m$ upper triangular matrix and $C_m = [\tilde{T}_0(D)v\, \tilde{T}_1(D)v\, \cdots\, \tilde{T}_{m-1}(D)v]$. We are trying to determine $P_m$ by induction. First,

$$Q_1 = q_1 = \frac{v}{||v||} = C_1 P_1 = v p_{11} \Rightarrow \quad P_1 = p_{11} = \frac{1}{||v||} \tag{4.21}$$

By induction, suppose we already find $P_m$, now we try to find $P_{m+1}$. Combining (4.18) and (4.20), we have

$$DC_m P_m = C_{m+1} P_{m+1} S_m \tag{4.22}$$

Since by (4.16) the $\tilde{T}_i(D)v$ follows

$$\begin{aligned} D\tilde{T}_0(D)v &= 2a\tilde{T}_1(D)v + b\tilde{T}_0(D)v \\ D\tilde{T}_l(D)v &= a\tilde{T}_{l+1}(D)v + a\tilde{T}_{l-1}(D)v + b\tilde{T}_l(D)v \end{aligned} \tag{4.23}$$

with $a = 1/2t_a$ and $b = -t_b/t_a$. So

$$\begin{aligned} DC_m P_m &= D[\tilde{T}_0(D)v\, \tilde{T}_1(D)v\, \cdots\, \tilde{T}_{m-1}(D)v]P_m \\ &= [\tilde{T}_0(D)v\, \tilde{T}_1(D)v\, \cdots\, \tilde{T}_{m-1}(D)v\, \tilde{T}_m(D)v]R_m P_m \end{aligned} \tag{4.24}$$

where

$$R_m = \begin{bmatrix} b & a & & & & \\ 2a & b & a & & & \\ & a & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & a \\ & & & & a & b \\ & & & & 0 & a \end{bmatrix}_{(m+1)\times m} \tag{4.25}$$

Substituting (4.24) to (4.22), we have

$$C_{m+1} R_m P_m = C_{m+1} P_{m+1} S_m \tag{4.26}$$

Since the columns of $C_{m+1}$ are linearly independent, then

$$R_m P_m = P_{m+1} S_m \tag{4.27}$$

We already found $P_m$ at this stage, let

$$P_m = \begin{bmatrix} \vec{p}_1 \\ \vec{p}_2 \\ \vdots \\ \vec{p}_m \end{bmatrix}_{m\times m} \qquad and \quad R_m P_m = \begin{bmatrix} \vec{r}_1 \\ \vec{r}_2 \\ \vdots \\ \vec{r}_m \\ \vec{r}_{m+1} \end{bmatrix}_{(m+1)\times m} \tag{4.28}$$

where $\vec{p}_i$ is the ith row of $P_m$, $\vec{r}_i$ is the ith row of $R_m P_m$. Then (4.27) becomes

$$\begin{bmatrix} \vec{r}_1 \\ \vdots \\ \vec{r}_m \\ \vec{r}_{m+1} \end{bmatrix} = \begin{bmatrix} \vec{p}_1 & p_{1,m+1} \\ \vdots & \vdots \\ \vec{p}_m & p_{m,m+1} \\ 0 & p_{m+1,m+1} \end{bmatrix} \begin{bmatrix} & T_m & \\ 0 & \cdots & 0 & \beta_m \end{bmatrix}$$

$$\Rightarrow \begin{cases} \vec{r}_1 = \vec{p}_1 T_m + p_{1,m+1}[0 \cdots 0\, \beta_m] \\ \quad \vdots \\ \vec{r}_m = \vec{p}_m T_m + p_{m,m+1}[0 \cdots 0\, \beta_m] \\ \vec{r}_{m+1} = p_{m+1,m+1}[0 \cdots 0\, \beta_m] \end{cases} \tag{4.29}$$

In (4.27), only $p_{i,m+1}$ are unknowns, we can obtain

$$\begin{cases} p_{1,m+1} = [(\vec{r}_1 - \vec{p}_1 T_m)_m]/\beta_m \\ \quad \vdots \\ p_{m,m+1} = [(\vec{r}_m - \vec{p}_m T_m)_m]/\beta_m \\ p_{m+1,m+1} = [(\vec{r}_{m+1})_m]/\beta_m \end{cases} \tag{4.30}$$

This means we can determine $P_{m+1}$ from $P_m$. Since $m$ is a small number of the Lanczos steps, by $Q_m = C_m P_m$, we can find all $\tilde{T}_1(D)v, \tilde{T}_2(D)v, \cdots, \tilde{T}_m(D)v$. The expensive step of computing the first m vectors $Dv_k$ $(k = 1, \cdots, m)$ is avoided.

*Remark* 4.2.2. The algorithm above do not work for the Chebyshev-Davidson method, since it computes $Av$ instead of $Dv$, where $A = T_l\left(\frac{D-cI}{e}\right)$.

We summarize the above algorithm in Algorithm 6.

**Algorithm 6** Chebyshev spectral approximation II for computing $\sqrt{D}v$

STEP 1. Precomputation stage

a. Use Algorithm 5 with $m$ steps to estimate $\lambda_{\max}(D)$ and $\lambda_{\min}(D)$.

b. Compute the Chebyshev coefficients $c_k$ $(k = 0, 1, \cdots, n)$ and determine the degree of the Chebyshev expansion $n$ so that it has the prescribed precision $\epsilon$ for $\sqrt{x}$ on $[\lambda_{\min}(D), \lambda_{\max}(D)]$.

STEP 2. Computation of the matrix square root

a. Use result from Step 1.a to compute $v_k$, $k = 1, \cdots, m$.

b. If $n > m$, use the recurrence relation (4.16) and FMM to compute $v_k$, $k = m + 1, \cdots, n$.

c. Compute $p_n(D)v = \sum_{k=0}^{n} c_k v_k$ and set $\sqrt{D}v \approx p_n(D)v$.

The precomputation stage requires $O(mN)$ operations with $m$ the number of Lanczos steps. For Brownian dynamics simulation, the second step requires $O((n - m)N)$ operations (using the algorithm above) with $n$ the degree of the approximating Chebyshev polynomial for $\sqrt{D}$ if $n > m$ and $O(1)$ operations if $n \leq m$. Thus the total computational cost is $O(\max(m, n)N)$ for Brownian dynamics simulation. In general, since $m$ is usually a very small number in practice (say, less than 12) and $n$ is proportional to $\sqrt{\kappa}$ as shown in Theorem 4.1.2, the computational cost of our algorithm is $O(\sqrt{\kappa}N)$.

*Remark* 4.2.3. Obviously, Algorithm II has two improvements over Algorithm I: (1) Using the Chebyshev-Davidson method to estimate the smallest eigenvalue costs much more time than using the plain Lanczos method to estimate the largest eigenvalue (in our practice, more than twice). The Safeguarded Lanczos Method can estimate the largest and smallest eigenvalues of $D$ together, it simplified the overall algorithm and reduced the computational time. (2) In Algorithm II, we could combine the step of estimating extreme eigenvalues of $D$ with the step of computing $p_n(D)v$. After

solving a small linear system of size $m$, the expensive step of computing the first $m$ vectors $Dv_k$ $(k = 1, \cdots, m)$ is avoided.

### 4.3 Algorithm III: Spectral Lanczos Decomposition Method

The degree of the Chebyshev approximation for the matrix square root depends on $\sqrt{\kappa}$, which could be very large if the condition number of the matrix is very large. Algorithms I and II may be very inefficient in this case. So here we use the Spectral Lanczos Decomposition Method (SLDM) [8] to avoid that problem. The SLDM is based on the Lanczos method only. There is no need to estimate the largest and smallest eigenvalues, and $\sqrt{D}v$ can be computed regardless of the condition number $\kappa$ of $D$.

Likewise, starting from an arbitrary vector $v$, the Lanczos method successively computes $Dv, D^2v, \cdots, D^kv$, constructs an orthogonal basis $Q_k$ for the Krylov subspace $K_k = \text{span}\{v, Dv, \cdots, D^kv\}$, and forms a much smaller $k \times k$ tridiagonal matrix $T_k = Q_k^T D Q_k$ using three term recurrence. It then applies any standard algorithm to compute $\sqrt{T_k}$, the square root matrix of $T_k$. Since $D$ is a real symmetric $n \times n$ matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, and $\sqrt{x}$ is a function analytic on $[\lambda_1, \lambda_n]$, from [8] we will have the approximation

$$\sqrt{D}v = ||v|| Q_k \sqrt{T_k} e_1^{(k)} \tag{4.31}$$

with $e_1^{(k)}$ the first unit $k$-vector. This technique appeared in the literature from the mid-eighties [53, 54, 55, 56, 57, 58, 59], and it can now be viewed as standard. During this process, the most expensive step, i.e., the calculation of $D^jv$ $(j = 1, \cdots, k)$ is done via the FMM. The computational cost of other steps is negligible since they involve much smaller matrices. Thus, the overall complexity of the algorithm is $O(kN)$, where $N$ is the size of $D$ and $k$ is the number of Lanczos steps.

We summarize the above algorithm in Algorithm 7.

**Algorithm 7** Estimating the matrix square root using the SLDM and FMM

Generate a random vector $v$ and normalize it $v_1 = v/\|v\|$.
Set $v_0 = 0$, $\beta_1 = 0$, $tol = 10^{-3}$ and $p = 12$.
**for** $k = 1, 2, ..., p$ **do**
    Use FMM to compute $Dv_k$ and set $w_j = Dv_k - \beta_k v_{k-1}$.
    Compute $\alpha_k = w_k \cdot v_k$.
    Set $w_k = w_k - \alpha_k v_k$ and $\beta_{k+1} = \|w_k\|$.
    Set $v_{k+1} = w_k/\beta_{k+1}$.
    **if** $k \geq 4$ **then**
        Construct a tridiagonal matrix $T_k$ with the diagonals equal to $(\alpha_1, \cdots, \alpha_k)$ and super- and sub-diagonals equal to $(\beta_2, \cdots, \beta_k)$ and construct matrix $Q_k = [v_1\, v_2 \cdots v_k]$.
        Use any standard method to compute $\sqrt{T_k}$.
        Compute $U_k = ||v||Q_k\sqrt{T_k}e_1^{(k)}$.
        **if** $|U_k - U_{k-1}|/U_{k-1} < tol$ **then**
            Set $\sqrt{D}v = U_k$, and **return**.
        **end if**
    **end if**
**end for**
Set $\sqrt{D}v = U_p$.

*Remark* 4.3.1. Suppose $U$ is the approximate value of $\sqrt{D}v$ by the SLDM, $V$ is the approximate value of $\sqrt{D}v$ by least squares procedure in the Krylov subspace $K_k(D, v) = \text{span}\{v, Dv, \cdots, D^k v\}$. For the Rotne-Prager-Yamakawa tensor, our extensive numerical tests show that $||U - \sqrt{D}v||_2 \leq ||V - \sqrt{D}v||_2$. So this method has enough accuracy.

*Remark* 4.3.2. The Spectral Lanczos Decomposition Method is an iterative method and thus its convergence rate depends only on the spectrum of $D$, while the Chebyshev approximation tries to construct an approximation which is uniformly good on the whole interval $[\lambda_{\min}(D), \lambda_{\max}(D)]$. Therefore, the SLDM may converge much faster than the Chebyshev approximation when the condition number of $D$ is large but its eigenvalues are clustered. In general, since $k$ is usually a very small number in our numerical implements (say, less than 12), the computational costs of computing $\sqrt{T_k}$ and $U_k$ are also small. Thus, the total computational cost is $O(kN)$ for Brownian dynamics simulation. Compared with Algorithms I and II, the SLDM does not need to estimate the extreme eigenvalues, it has simplified the overall algorithm and reduced the computational time.

# CHAPTER 5

# NUMERICAL RESULTS

A C++ and Fortran mixed code has been written implementing the algorithms described in the preceding chapter. In this chapter, we present some numerical experiments to check the accuracy and explore the applicable scope of the algorithms. All of the programs are run on a laptop with 2.53 GHz CPU and 1.89 GB memory.

## 5.1 Condition Number and Terms Needed in Chebyshev Approximation

Though Theorem 4.1.2 shows that the number of terms $n$ needed in the approximating Chebyshev polynomial depends linearly on $\log(\frac{1}{\epsilon})\sqrt{\kappa}$, in practice it is better to simply compute the necessary number of terms needed by comparing the approximation with $\sqrt{x}$ on $[\lambda_{\min}, \lambda_{\max}]$. This will give a much more accurate estimate of the necessary number of terms needed for a prescribed precision. And the computational cost of this step is negligible compared with that of the other steps. In Table 5.1, we report the number of terms needed in the Chebyshev polynomial approximation. The first column indicates the desired precision $\epsilon$. The first row indicates the condition number $\kappa$ of the matrix $D$.

Next, we observe that the overall complexity of our algorithm is $O(\sqrt{\kappa}N)$. Thus, it is important to investigate the condition number $\kappa$ of the tensor $D$ for various numbers $N$ of particles and particle configurations. We assume that the particles are contained in a box of side length $L$. The numerical results are summarized in Table 5.2, where the first column indicates the total number of Brownian particles, and the first row indicates the ratio $L/a$ with $L$ the side length of the box and $a$ the radius of each particle.

Our numerical experiments show that the condition number of the tensor $D$ is small if very few Brownian particles are close to each other. Indeed, we observe that

**Table 5.1** Number of Terms Needed in Chebyshev Approximation to Approximate $\sqrt{x}$ on $[a, b]$ with $a > 0$ and $\frac{b}{a} = \kappa$. The first column indicates the desired precision $\epsilon$. The first row indicates the condition number $\kappa$.

| $\epsilon \backslash \kappa$ | 2 | 10 | 100 | 1000 | 10000 |
|---|---|---|---|---|---|
| $10^{-3}$ | 3 | 4 | 7 | 8 | 8 |
| $10^{-4}$ | 4 | 7 | 12 | 17 | 18 |
| $10^{-6}$ | 5 | 12 | 28 | 54 | 79 |
| $10^{-9}$ | 9 | 22 | 57 | 137 | 297 |

**Table 5.2** The Condition Number $\kappa$ of the Tensor $D$. The first column indicates the total number of particles $N$. The first row indicates the ratio $L/a$ with $L$ the side length of the simulation box and $a$ the radius of each particle.

| $N \backslash \frac{L}{a}$ | $10^2$ | $10^3$ | $10^4$ | $10^5$ |
|---|---|---|---|---|
| $10^2$ | 2.6627 | 1.25848 | 1.13958 | 1.12345 |
| $10^3$ | 19.7834 | 2.28714 | 1.24435 | 1.13326 |
| $10^4$ | 128.585 | 12.2358 | 2.21383 | 1.23047 |
| $10^5$ | 1228.94 | 111.124 | 11.9808 | 2.20866 |
| $10^6$ | 10649.1 | 1084.43 | 109.581 | 11.9612 |

the condition number $\kappa$ is roughly constant along the diagonals in Table 5.2. This indicates that $\kappa$ depends only on the "density" $Na/L$. When the density is low, the condition number is small. And when the density is high, the condition number is high. In practice, the density of the Brownian particles will be fairly low, hence the condition number of the tensor $D$ will be small, say, less than 200, and independent of the total number of particles $N$ in the simulation.

## 5.2 Algorithm I: Lanczos Method with Chebyshev-Davidson Method plus Chebyshev Spectral Approximation

For Algorithm I, we have tested the performance of our algorithm with $Na/L$ fixed for various $N$s. The average timing results for various particle configurations are summarized in Table 5.3.

**Table 5.3** Timing and Relative Error Results for Generating Random Displacement Vectors by Algorithm I, $Na/L = 1$ is Fixed

| $N$ | $T_{Lanc}$ | $m$ | $T_{Davi}$ | $\kappa$ | $n$ | $T_{Cheb}$ | $T_{total}$ | $error$ |
|---|---|---|---|---|---|---|---|---|
| $10^3$ | 0.356022 | 4 | 0.924058 | 2.23841 | 4 | 0.416026 | 1.69611 | 4.07816e-05 |
| $10^4$ | 9.97262 | 4 | 24.9416 | 2.21096 | 4 | 10.0246 | 44.9388 | 4.11333e-05 |
| $10^5$ | 125.960 | 4 | 314.501 | 2.20644 | 4 | 126.022 | 566.483 | 4.07739e-05 |
| $10^6$ | 1266.13 | 4 | 3423.03 | 2.20386 | 4 | 1269.42 | 5958.58 | 3.08514e-05 |

In Table 5.3, the first column contains the total number of particles. The second column contains the time needed (in seconds) for the Lanczos method. The third column contains the number of Lanczos steps. The fourth column contains the time needed for Chebyshev-Davidson method. The fifth column contains the condition number of the tensor. The sixth column contains the number of terms needed in

the Chebyshev approximation. The seventh column contains the computational time of computing the Chebyshev approximations. The eighth column contains the total computational time for computing $\sqrt{D}v$. And the last column contains the relative error of the computational result, where the relative error is defined as the average of $\left|\frac{V_{eval}^n - V_{true}}{V_{true}}\right|$ and $\frac{\|v_n - v_{n-1}\|}{\|v_n\|}$ with $v_n = p_n(D)v$ the nth approximation of $\sqrt{D}v$, $v_{n-1} = p_{n-1}(D)v$, $V_{eval}^n = v_n^T v_n$, and $V_{true} = v^T D v$. The first term in the relative error serves as a check with the true value $\sqrt{D}v$, and the second term serves as a self consistency check.

We have also tested Algorithm I for various particle configurations with $L$ and $a$ fixed. The results with $L = 1000$ and $a = 0.1$ are reported in Table 5.4.

**Table 5.4** Timing and Relative Error Results for Generating Random Displacement Vectors by Algorithm I, with $L = 1000$ and $a = 0.1$

| $N$ | $T_{Lanc}$ | $m$ | $T_{Davi}$ | $\kappa$ | $n$ | $T_{Cheb}$ | $T_{total}$ | $error$ |
|---|---|---|---|---|---|---|---|---|
| $10^3$ | 0.384024 | 4 | 0.888056 | 1.23428 | 3 | 0.316021 | 1.58810 | 1.17941e-06 |
| $10^4$ | 9.95262 | 4 | 24.8456 | 2.21357 | 4 | 9.97262 | 44.7708 | 1.05025e-04 |
| $10^5$ | 125.481 | 4 | 313.504 | 11.9944 | 6 | 188.640 | 627.625 | 7.70808e-04 |
| $10^6$ | 1260.57 | 4 | 3149.75 | 109.258 | 16 | 5046.75 | 9457.07 | 7.69059e-04 |

## 5.3 Algorithm II: Safeguarded Lanczos Method plus Chebyshev Spectral Approximation

For Algorithm II, we have tested the performance of our algorithm with $Na/L$ fixed for various $N$s. The average timing results for various particle configurations are summarized in Table 5.5.

**Table 5.5** Timing and Relative Error Results for Generating Random Displacement Vectors by Algorithm II, $Na/L = 1$ is Fixed

| $N$ | $T_{Lanc}$ | $m$ | $\kappa$ | $n$ | $T_{Cheb}$ | $T_{total}$ | $error$ |
|---|---|---|---|---|---|---|---|
| $10^3$ | 0.596037 | 7 | 2.29495 | 4 | 0.032002 | 0.628039 | 3.36403e-04 |
| $10^4$ | 17.5611 | 7 | 2.22923 | 4 | 0.036003 | 17.5971 | 3.11699e-04 |
| $10^5$ | 221.146 | 7 | 2.19521 | 4 | 0.120007 | 221.266 | 2.49343e-04 |
| $10^6$ | 2244.88 | 7 | 2.18724 | 4 | 1.34408 | 2246.22 | 2.40625e-04 |

In Table 5.5, the first column contains the total number of particles. The second column contains the time needed for the Lanczos method. The third column contains the number of Lanczos steps. The fourth column contains the condition number of the tensor. The fifth column contains the number of terms needed in the Chebyshev approximation. The sixth column contains the computational time of computing the Chebyshev approximations. The seventh column contains the total computational time for computing $\sqrt{D}v$. And the last column contains the relative error of the computational result, where the relative error is defined as before.

**Table 5.6** Timing and Relative Error Results for Generating Random Displacement Vectors by Algorithm II with $L = 1000$ and $a = 0.1$

| $N$ | $T_{Lanc}$ | $m$ | $\kappa$ | $n$ | $T_{Cheb}$ | $T_{total}$ | $error$ |
|---|---|---|---|---|---|---|---|
| $10^3$ | 0.660041 | 7 | 1.22813 | 3 | 0.032002 | 0.692043 | 6.51492e-04 |
| $10^4$ | 17.8331 | 7 | 2.20676 | 4 | 0.040003 | 17.8731 | 3.88933e-04 |
| $10^5$ | 220.958 | 7 | 12.2125 | 7 | 0.108007 | 221.066 | 5.02846e-04 |
| $10^6$ | 2547.04 | 8 | 117.534 | 16 | 2565.53 | 5112.57 | 6.63551e-04 |

We have also tested Algorithm II for various particle configurations with $L$ and $a$ fixed. The results with $L = 1000$ and $a = 0.1$ are reported in Table 5.6.

### 5.4 Algorithm III: Spectral Lanczos Decomposition Method

For Algorithm III, we have tested the performance of our algorithm with $Na/L$ fixed for various $N$s. The average timing results for various particle configurations are summarized in Table 5.7.

**Table 5.7** Timing and Relative Error Results for Generating Random Displacement Vectors by Algorithm III, $Na/L = 1$ is Fixed

| $N$ | $k$ | $T_{total}$ | $error$ |
|---|---|---|---|
| $10^3$ | 4 | 0.372023 | 1.38721e-07 |
| $10^4$ | 4 | 10.0486 | 1.91785e-05 |
| $10^5$ | 4 | 128.596 | 6.80112e-04 |
| $10^6$ | 7 | 2250.12 | 2.59537e-04 |

**Table 5.8** Timing and Relative Error Results for Generating Random Displacement Vectors by Algorithm III with $L = 1000$ and $a = 0.1$

| $N$ | $k$ | $T_{total}$ | $error$ |
|---|---|---|---|
| $10^3$ | 4 | 0.308019 | 1.22036e-07 |
| $10^4$ | 4 | 10.1166 | 2.49304e-05 |
| $10^5$ | 4 | 126.324 | 8.07139e-04 |
| $10^6$ | 7 | 2240.19 | 4.24709e-04 |

In Table 5.7, the first column contains the total number of particles. The second column contains the number of Lanczos steps. The third column contains the total

computational time for computing $\sqrt{D}v$. The last column contains the relative error of the computational result, where the relative error is the same as defined before.

We have also tested Algorithm III for various particle configurations with $L$ and $a$ fixed. The results with $L = 1000$ and $a = 0.1$ are reported in Table 5.8.

## 5.5 Comparison of the Algorithm I, II and III

Finally, the timing and relative error results by our three algorithms with $Na/L = 1$ fixed for various $N$s are shown in Table 5.9. The first column contains the total number of particles. $T_i$ and $error_i$, $i = 1, 2, 3$, denote the time required and the relative error by Algorithms I, II and III.

**Table 5.9** Timing and Relative Error Results for Generating Random Displacement Vectors by Three Algorithm, $Na/L = 1$ is Fixed

| $N$ | $T_1$ | $error_1$ | $T_2$ | $error_2$ | $T_3$ | $error_3$ |
|---|---|---|---|---|---|---|
| $10^3$ | 1.69611 | 4.07816e-05 | 0.628039 | 3.36403e-04 | 0.372023 | 1.38721e-07 |
| $10^4$ | 44.9388 | 4.11333e-05 | 17.5971 | 3.11699e-04 | 10.0486 | 1.91785e-05 |
| $10^5$ | 566.483 | 4.07739e-05 | 221.266 | 2.49343e-04 | 128.596 | 6.80112e-04 |
| $10^6$ | 5958.58 | 3.08514e-05 | 2246.22 | 2.40625e-04 | 2250.12 | 2.59537e-04 |

The timing and relative error results by our three algorithms have also shown various particle configurations with $L$ and $a$ fixed. The results with $L = 1000$ and $a = 0.1$ are reported in Table 5.10.

**Table 5.10** Timing and Relative Error Results for Generating Random Displacement Vectors by Three Algorithm with $L = 1000$ and $a = 0.1$

| $N$ | $T_1$ | $error_1$ | $T_2$ | $error_2$ | $T_3$ | $error_3$ |
|---|---|---|---|---|---|---|
| $10^3$ | 1.58810 | 1.17941e-06 | 0.692043 | 6.51492e-04 | 0.308019 | 1.22036e-07 |
| $10^4$ | 44.7708 | 1.05025e-04 | 17.8731 | 3.88933e-04 | 10.1166 | 2.49304e-05 |
| $10^5$ | 627.625 | 7.70808e-04 | 221.066 | 5.02846e-04 | 126.324 | 8.07139e-04 |
| $10^6$ | 9457.07 | 7.69059e-04 | 5112.57 | 6.63551e-04 | 2240.19 | 4.24709e-04 |

# CHAPTER 6

# CONCLUSION

We have presented three fast algorithms for generating random vectors whose spatial correlation is determined by the hydrodynamic interactions, the Rotne-Prager-Yamakawa tensor in particular.

First, we presented two fast multipole methods for computing $D\mathbf{u}$. The first FMM was the kernel indepedent FMM, which required 9 scalar FMM calls. The second FMM, similar to the FMM for Stokeslet, decomposed the Rotne-Prager-Yamakawa tensor into harmonic potentials and its derivatives, and thus required only four harmonic FMM calls, which have been trivially modified so that the net cost scaled like three harmonic interactions. Both FMMs reduced the computational cost of $D\mathbf{u}$ from $O(N^2)$ to $O(N)$ for an arbitrary $N$-particle configuration.

Second, for computing $\sqrt{D}\mathbf{v}$ in Algorithm I, we used the Lanczos method with the Chebyshev-Davidson method to estimate the extreme eigenvalues of $D$. Then the Chebyshev spectral approximation was used for computing matrix square root. In Algorithm II, we used the safeguarded Lanczos method to estimate the extreme eigenvalues, it simplified the overall algorithm and reduced the computational time. In the Chebyshev spectral approximation, we combined the step of estimating extreme eigenvalues of $D$ with the step of computing $p_n(D)v$, the expensive step of computing the first m vectors $Dv_k$ $(k = 1, \cdots, m)$ was avoided. We observed that Algorithm II achieved a factor of 2-3 speed-up as compared with Algorithm I. The complexity of Algorithm I and Algorithm II are $O(\sqrt{\kappa}N)$ with $\kappa$ the condition number of the tensor and $N$ the total number of Brownian particles.

Third, we observed that the degree of the Chebyshev approximation for the matrix square root depends on the square root of the condition number of the

matrix, which could be very large for certain matrices. For these cases, we developed Algorithm III using the spectral Lanczos decomposition method. There was no need to estimate the extreme eigenvalues, and $\sqrt{D}v$ could be computed regardless of the condition number $\kappa$ of $D$. The complexity of Algorithm III is $O(mN)$, where the $m$ is the number of Lanczos steps. Our numerical experiments showed that the SLDM is generally more effective than the Chebyshev approximation method, which is currently the most widely used approach in Brownian dynamics simulation community.

The work in this dissertation will be continued in several ways in the future.

- Our algorithms can be easily generalized to compute the matrix-vector product $\sqrt{A}v$ for many other matrices when $Av$ can be computed via fast algorithms such as the fast multipole method. Thus, the algorithms are useful in many other applications, including the statistical analysis with certain spatial correlations and model reduction in dynamic control theory.

- For the RPYFMM, a further speed-up could be achieved if we grouped all four harmonic FMM calls into a single routine (although this would require more memory). This would avoid the repeated calculation of various special functions (such as spherical harmonics).

- In Algorithm III, the complexity is $O(mN)$, but the $m$ may still depend on the matrix. So we try to find the relationship or more information about it.

- Except this, a fast direct algorithm which compresses the square root matrix (an idea similar to [60, 61]) would be developed for the matrix which has the large condition number. This approach is currently under investigation and results will be reported in the future.

- Currently we are working on the numerical simulation of Brownian dynamics with the Rotne-Prager-Yamakawa tensor. Our algorithm can be utilized to

accelerate the computation of the total displacement $\Delta\mathbf{x}^m$ in the Ermark-McCammon algorithm. This work will be reported in a later publication.

## BIBLIOGRAPHY


[1] J. Rotne and S. Prager, "Variational Treatment of Hydrodynamic Interaction in Polymers," *J. Chem. Phys.*, vol. 50, pp. 4831–4837, 1969.

[2] H. Yamakawa, "Transport Properties of Polymer Chains in Dilute Solution: Hydrodynamic Interaction," *J. Chem. Phys.*, vol. 53, pp. 436–443, 1970.

[3] L. Ying, G. Biros, and D. Zorin, "A Kernel-Independent Adaptive Fast Multipole Algorithm in Two and Three Dimensions," *J. Comp. Phys.*, vol. 196, pp. 591–626, 2004.

[4] A. Tornberg and L. Greengard, "A Fast Multipole Method for the Three-dimensional Stokes Equations," *J. Comput. Phys.*, vol. 3, pp. 1613–1619, 2008.

[5] M. Fixman, "Implicit Algorithm for Brownian Dynamics of Polymers," *Macromolecules*, vol. 19, pp. 1195–1204, 1986.

[6] J. P. Hernandez-Ortiz, J. J. de Pablo, and M. D. Graham, "Fast Computation of Many-Particle Hydrodynamic and Electrostatic Interactions," *Phys. Rev. Letters*, vol. 98, p. 140602, 2007.

[7] Y. Zhou and R. C. Li, "Bounding the Spectrum of Large Hermitian Matrices," *Linear Algebra and its App.*, vol. 435, pp. 480–493, 2011.

[8] V. Druskin and L. Knizhnerman, "Extended Krylov Subspaces: Approximation of the Matrix Square Root and Related Functions," *SIAM. J. Matrix Anal. Appl.*, vol. 19, pp. 755–771, 1998.

[9] D. L. Ermak and J. A. McCammon, "Brownian Dynamics with Hydrodynamic Interactions," *J. Chem. Phys.*, vol. 69, pp. 1352–1360, 1978.

[10] G. K. Batchelor, "Brownian Diffusion of Particles with Hydrodynamic Interaction," *J. Fluid Mech.*, vol. 74, pp. 1–29, 1976.

[11] D. Petera and M. Muthukumar, "Brownian Dynamics Dimulation of Bead-rod Chains under Shear with Hydrodynamic Interaction," *J. Chem. Phys.*, vol. 111, pp. 7614–7623, 1999.

[12] P. G. Wolynes and J. M. Deutch, "Dynamical Orientation Correlations in Solution," *J. Chem. Phys.*, vol. 67, pp. 733–741, 1977.

[13] P. G. de Gennes, "Passive Entry of a DNA Molecule into a Small Pore," *Proc. Natl. Acad. Sci. USA*, vol. 96, pp. 7262–7264, 1999.

[14] B. B. Haab and R. A. Mathies, "Single-Molecule Detection of DNA Separations in Microfabricated Capillary Electrophoresis Chips Employing Focused Molecular Streams," *Anal. Chem.*, vol. 71, pp. 5137–5145, 1999.

[15] R. G. Larson, H. Hua, D. E. Smith, and S. Chu, "Brownian Dynamics Simulations of a DNA Molecule in an Extensional Flow Field," *J. Rheol.*, vol. 43, pp. 267–304, 1999.

[16] P. G. Wolynes and J. M. Deutch, "Slip Boundary Conditions and the Hydrodynamic Effect on Diffusion Controlled Reactions," *J. Chem. Phys.*, vol. 65, pp. 450–454, 1976.

[17] H. L. Friedman, "A Hydrodynamic Effect in the Rates of Diffusion Controlled Reactions," *J. Chem. Phys.*, vol. 70, pp. 3931–3933, 1966.

[18] B. U. Felderhof and J. M. Deutch, "Frictional Properties of Dilute Polymer Solutions I. Rotational Friction Coefficient," *J. Chem. Phys.*, vol. 62, pp. 2391–2397, 1975.

[19] J. A. McCammon, J. M. Deutch, and B. U. Felderhof, "Frictional Properties of Multisubunit Structures," *Biopolymers*, vol. 14, pp. 2613–2623, 1975.

[20] M. Karplus and D. L. Weaver, "Protein Folding Dynamics," *Nature*, vol. 260, pp. 404–406, 1976.

[21] J. M. Deutch and I. Oppenheim, "Molecular Theory of Brownian Motion for Several Particles," *J. Chem. Phys.*, vol. 54, pp. 3547–3554, 1971.

[22] A. Sierou and J. Brady, "Accelerated Stokesian Dynamics Simulations," *J. Fluid Mech.*, vol. 448, pp. 115–146, 2001.

[23] J. Brady, R. Phillips, J. Lester, and G. Bossis, "Dynamic Simulation of Hydrodynamically Interacting Suspensions," *J. Fluid Mech.*, vol. 195, pp. 257–280, 1988.

[24] T. Phung, J. Brady, and G. Bossis, "Stokesian Dynamics Simulation of Brownian Suspensions," *J. Fluid Mech.*, vol. 313, pp. 181–207, 1996.

[25] T. Geyer and U. Winter, "An $O(n^2)$ Approximation for Hydrodynamic Interactions in Brownian Dynamics Simulations," *J. Chem. Phys.*, vol. 130, p. 114905, 2009.

[26] L. Greengard and V. Rokhlin, "A Fast Algorithm for Particle Simulations.," *J. Comp. Phys.*, vol. 73, pp. 325–348, 1987.

[27] L. Greengard and V. Rokhlin, "A New Version of the Fast Multipole Method for the Laplace Equation in Three Dimensions.," *Acta Numerica*, vol. 6, pp. 229–269, 1997.

[28] W. Fong and E. Darve, "The Black-Box Fast Multipole Method," *J. Comp. Phys.*, vol. 228, pp. 8712–8725, 2009.

[29] Z. Gimbutas and V. Rokhlin, "A Generalized Fast Multipole Method for Nonoscillatory Kernels.," *SIAM J. Sci. Comput.*, vol. 24, pp. 796–817, 2003.

[30] P. G. Martinsson and V. Rokhlin, "An Accelerated Kernel-Independent Fast Multipole Method in One Dimensions," *SIAM J. Sci. Comput.*, vol. 29, pp. 1160–1178, 2007.

[31] B. Zhang, "Integral-Equation-Based Fast Algorithms and Graph-Theoretic Methods for Large-Scale Simulations.," *PhD thesis, University of North Carolina, Chapel Hill*, 2010.

[32] G. Dahlquist and A. Bjorck, *Numerical Methods*. Prentice Hall, 1974.

[33] M. H. DeGroot and M. J. Schervish, *Probability and Statistics*. Addison Wesley, 3rd ed., 2002.

[34] L. N. Trefethen, *Spectral Methods in Matlab*. SIAM, 2000.

[35] G. A. Huber and J. A. McCammon, "Browndye: A Software Package for Brownian Dynamics," *Computer Physics Communications*, vol. 181, pp. 1896–1905, 2010.

[36] S. H. Cheng, N. J. Higham, C. S. Kenney, and A. J. Laub, "Approximating the Logarithm of a Matrix to Specified Accuracy," *SIAM J. Matrix Anal. Appl.*, vol. 22, pp. 1112–1125, 2001.

[37] B. Fornberg, *A practical guide to pseudospectral methods*. Cambridge University Press, 1998.

[38] J. G. Kirkwood, "The Statistical Mechanical Theory of Irreversible Processes in Solutions of Flexible Macromolecules. Visco-elastic Behavior," *Rec. Trav. Chim.*, vol. 68, p. 649, 1949.

[39] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*. Taylor and Francis, Inc, 1988.

[40] T. Darden, D. York, and L. Pedersen, "Particle Mesh Ewald: An $N \log(N)$ Method for Ewald Sums in Large Systems," *J. Chem. Phys.*, vol. 98, pp. 10089–10092, 1993.

[41] J. R. Phillips, J. K. White, and A. Member, "A Precorrected-FFT Method for Electrostatic Analysis of Complicated 3-D Structures," *IEEE Trans. Comput.-Aided Des. Integrat. Circ. Syst.*, vol. 16, pp. 1059–1072, 1997.

[42] R. Harrison, G. Fann, T. Yanai, Z. Gan, and G. Beylkin, "Multiresolution Quantum Chemistry:Basic Theory and Initial Applications," *J. Chem. Phys.*, vol. 121, pp. 11587–11598, 2004.

[43] A. Brandt, "Multilevel Computations of Integral Transforms and Particle Interactions with Oscillatory Kernels," *Comp. Phys. Commun.*, vol. 65, pp. 24–38, 1991.

[44] A. W. Appel, "An Efficient Program for Many-Body Simulation," *SIAM J. Sci. Stat. Comput.*, vol. 6, pp. 85–103, 1985.

[45] J. Barnes and P. Hut, "A Hierarchical $O(N \log N)$ Force-Calculation Algorithm," *Nature*, vol. 324, pp. 446–449, 1986.

[46] M. Gu, X. S. Li, and P. S. Vassilevski, "Direction-Preserving and Schur-Monotonic Semiseparable Approximations of Symmetric Positive Definite Matrices," *Nucleic Acids Research*, vol. 31, pp. 2650–2664, 2010.

[47] S. Jiang, Z. Liang, and J. Huang, "A Fast Algorithm for Brownian Dynamics Simulation with Hydrodynamic Interactions," *Mathematics of Computation, accepted*.

[48] H. Cheng, L. Greengard, and V. Rokhli, "A Fast Adaptive Multipole Algorithm in Three Dimensions," *J. Comput. Phys.*, vol. 155, pp. 468–498, 1999.

[49] Z. Liang, Z. Gimbutas, L. Greengard, J. Huang, and S. Jiang, "A Fast Multipole Method for the Rotne-Prager-Yamakawa Tensor and Its Applications," *J. Comp. Phys., submitted*.

[50] G. H. Golub and C. F. V. Loan, *Matrix Computations*. The Johns Hopkins University Presss, 1996.

[51] Y. Zhou and Y. Saad, "A Chebyshev-Davidson Algorithm for Large Symmetric Eigenvalue Problems," *SIAM J. Matrix Anal. App.*, vol. 29, pp. 954–971, 2007.

[52] L. N. Trefethen, *Numerical Linear Algebra*. SIAM, Philadelphia, 1997.

[53] A. Nauts and R. E. Wyatt, "New Approach to Many State Quantum Dynamics," *Phys. Rev. Lett.*, vol. 51, pp. 2238–2241, 1983.

[54] B. Nour-Omid and R. W. Clough, "Dynamic Analysis of Structure using Lanczos Co-ordinates," *Earthquake Engrg. Structur. Dynamics*, vol. 12, pp. 565–577, 1984.

[55] T. J. Park and J. C. Light, "Unitary Quantum Time Evolution by Iterative Lanczos Reduction," *J. Chem. Phys.*, vol. 85, pp. 5870–5876, 1986.

[56] H. A. Vorst, "An Iterative Solution Method for Solving f(A)x = b, Using Krylov Subspace Information Obtained for the Symmetric Positive Definite Matrix A," *J. Comput. Appl. Math.*, vol. 18, pp. 249–263, 1987.

[57] V. L. Druskin and L. A. Knizhnerman, "A Spectral Semi-discrete Method for the Numerical Solution of 3D Nonstationary Problems in Electrical Prospecting," *Izv. Akad. Sci. U.S.S.R.,Physics of Solid Earth*, vol. 24, pp. 641–648, 1988.

[58] R. A. Frishner, L. S. Tukerman, B. C. Dornblacer, and T. V. Russo, "A Method of Exponential Propagation of Large Systems of Stiff Nonlinear Dfferential Equations," *J. Sci. Comput.*, vol. 4, pp. 327–335, 1989.

[59] E. Gallopoulos and Y. Saad, "Effcient Solution of Parabolic Equations by Krylov Approximation Methods," *SIAM J. Sci. Statist. Comput.*, vol. 13, pp. 1236–1264, 1992.

[60] P. G. Martinsson and V. Rokhlin, "A Fast Direct Solver for Boundary Integral Equations in Two Dimensions," *J. Comput. Phys.*, vol. 1, pp. 1–23, 2005.

[61] L. Greengard, D. Gueyffier, P. Martinsson, and V. Rokhlin, "Fast Direct Solvers for Integral Equations in Complex Three-Dimensional Domains," *Acta Numerica*, vol. 18, pp. 243–275, 2009.