MATHEMATICS OF COMPUTATION
Volume 82, Number 283, July 2013, Pages 1631–1645
S 0025-5718(2013)02672-5
Article electronically published on February 27, 2013

A FAST ALGORITHM FOR BROWNIAN DYNAMICS SIMULATION WITH HYDRODYNAMIC INTERACTIONS

SHIDONG JIANG, ZHI LIANG, AND JINGFANG HUANG

Abstract. One of the critical steps in Brownian dynamics simulation with hydrodynamic interactions is to generate a normally distributed random vector whose covariance is determined by the Rotne-Prager-Yamakawa tensor. The standard algorithm for generating such a random vector calls for the Cholesky decomposition of a $3N \times 3N$ matrix and thus requires $O(N^3)$ operations for N particles, which is prohibitively slow for large scale simulations. In this paper, we present a fast algorithm for generating such random vectors. Our algorithm combines the Chebyshev spectral approximation for the square root of a positive definite matrix and kernel independent fast multipole method. The overall complexity of the algorithm is $O(\sqrt{\kappa}N)$ with κ the condition number of the matrix and N the size of the particle system. Numerical experiments show that the algorithm can be applied to various particle configurations with essentially $O(N)$ operations since κ is usually small and independent of N. Thus, our fast algorithm 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. Finally, the algorithm can be extended to speed up the computation involving the matrix square root for many other matrices, which has potential applications on areas such as statistical analysis with certain spatial correlations and model reduction in dynamic control theory.

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 the 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 leads 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. 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

Received by the editor July 20, 2011 and, in revised form, December 1, 2011.
2010 *Mathematics Subject Classification.* Primary 76M35, 65Z05, 65C60, 65F60.
The first author was supported in part by NSF under grant CCF-0905395.
The third author was supported by NSF under grant CCF-0905473. The author's support is thankfully acknowledged.

setup that does not consider hydrodynamics, the hydrodynamics interactions accelerate the dynamics of the particle system. The results presented in [6] 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 [1, 27, 33]. There has been recent interest in the rheological and conformational properties of dilute solutions of DNA and other proteins [12, 19, 23]. Moreover, the hydrodynamic interaction profoundly influences the dynamics of diffusional encounters [11, 32] and the description of the transport properties of multisubunit structures in terms of subunit frictional coefficients [7, 26]. Computer simulations should be useful for studying certain aspects of protein folding [22], particle coagulation, and other biochemical processes. However, when hydrodynamic interactions are included in a Brownian dynamics simulation, the random displacements become *correlated* [5], 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.

In this paper, we consider the Ermak-McCammon algorithm [6, 13] 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 [28, 34] defined as follows, which models the Stokes flow for two spheres and neglects the hydrodynamic rotation-rotation and rotation-translation coupling,

$$D_{ii} = \frac{k_B T}{6\pi\eta a} I, \tag{1}$$

$$D_{ij} = \frac{k_B T}{8\pi\eta r_{ij}}[(I + \frac{\mathbf{r}_{ij}\mathbf{r}_{ij}}{r_{ij}^2}) + \frac{2a^2}{3r_{ij}^2}(\mathbf{I} - 3\frac{\mathbf{r}_{ij}\mathbf{r}_{ij}}{r_{ij}^2})] \quad \text{for} \quad i \neq j \quad \text{and} \quad r_{ij} \geq 2a, \tag{2}$$

$$D_{ij} = \frac{k_B T}{6\pi\eta a}[(1 - \frac{9}{32}\frac{r_{ij}}{a})I + \frac{3}{32a}\frac{\mathbf{r}_{ij}\mathbf{r}_{ij}}{r_{ij}}] \quad \text{for} \quad i \neq j \quad \text{and} \quad r_{ij} < 2a. \tag{3}$$

Here k_B is the Boltzmann constant, T is the absolute temperature, η is the solvent viscosity, a represents the hydrodynamic radius of each particle, i and j label particle indices, I is the 3×3 identity matrix, $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$, and $r_{ij} = \sqrt{\mathbf{r}_{ij} \cdot \mathbf{r}_{ij}}$ with $\mathbf{r}_i$ the position vector of the ith particle. We would like to remark here that this tensor has been shown to be positive definite for all particle configurations [34]. In the Ermak-McCammon algorithm, the total displacement Δr_i of the ith particle during a time step Δt due to the force F_j and diffusion tensor D is given by

$$\Delta r_i\,(\Delta) = \sum_j \frac{D_{ij}F_j}{k_B T}\Delta t + \sum_j \frac{\partial D_{ij}}{\partial r_j}\Delta t + R_i\,(\Delta t)\,, \tag{4}$$

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

$$\langle R_i(\Delta t)\rangle = 0, \quad \langle R_i(\Delta t)R_j(\Delta t)\rangle = 2D_{ij}\Delta t. \tag{5}$$

Obviously, the deterministic part of Δr_i $(i = 1, \cdots, N)$ (i.e., the first two terms on the right side of (4)) can be computed in $O(N)$ or $O(N \log N)$ time using the fast multipole method or fast Fourier transform (see, for example, [9, 14, 17, 18, 25, 35, 36]). However, It is nontrivial to generate $3N$ normally distributed random vectors R_i $(i = 1, \cdots, N)$ with the particular correlation (5) efficiently. Indeed, the standard technique in statistics [3, 4] 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, v. Third, multiply $C\sqrt{2\Delta t}$ with v and the resulting vector will be normally distributed with the correlation given by (5). 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 R has become one of the bottlenecks in the large-scale Brownian dynamics simulations with hydrodynamic interactions.

The purpose of this paper is to discuss a fast algorithm for generating such random vectors 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 (5) 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 $\sqrt{D}v$ efficiently for an arbitrary vector v. We observe that, given an arbitrary vector v, the fast multipole method can compute Dv in $O(N)$ operations; our strategy is as follows. We first 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. The degree n of the approximate polynomial depends logarithmically on the prescribed precision ϵ and is proportional to the square root of the condition number κ of the matrix D, that is, $n \propto \log(\frac{1}{\epsilon})\sqrt{\kappa}$. We would like to remark here that n is usually a small number (say, less than 20 for most of our numerical experiments for four-digit accuracy). We then approximate $\sqrt{D}v$ by $p_n(D)v$. Since Dv can be computed via FMM efficiently, $p_n(D)v$ can also be computed efficiently. The overall complexity of generating one such random vector R is thus 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 [21], and applications on biomolecular systems will be reported in the future.

Remark 1.1. The Chebyshev polynomial approximation for the square root matrix has been applied to the Brownian dynamics simulation by some researchers. [8] seems to be the earliest one to propose this method. It has been used later by other researchers (see, e.g., [13, 20]). 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 $O(N)$.

The outline of this paper is as follows. The numerical tools needed for our algorithm are summarized in Section 2. In Section 3, we present some details of our fast algorithm. The performance of our fast algorithm is illustrated via several numerical examples in Section 4. Finally, we present a short conclusion and discuss possible extension and applications of our algorithm in Section 5.

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:

$$D = BB^*. \tag{6}$$

This decomposition is not unique. Indeed, if B satisfies (6), then $B \cdot U$ also satisfies (6) for any unitary matrix U since $BU \cdot (BU)^* = BUU^*B^* = BIB^* = D$. Thus there are infinitely many matrices satisfying (6) 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

$$D = \sqrt{D} \cdot \sqrt{D}. \tag{7}$$

$\sqrt{D}$ 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, [2]). 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. *Suppose that D is a real, symmetric, and positive definite matrix and that $\sqrt{D}$ is its square root defined by* (7)*. Then there exists a polynomial $p(\cdot)$ such that $\sqrt{D} = p(D)$, and the degree of p is equal to the number of distinct eigenvalues of D minus* 1*.*

Proof. Suppose $D = O\Lambda O^T$ with O an orthogonal matrix and Λ 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

$$\sqrt{\lambda_i} = p(\lambda_i) = \sum_{j=0}^{k-1} c_j {\lambda_i}^j, \qquad i = 1, \cdots, k. \tag{8}$$

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 (8) 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 Λ 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

$$T_n(x) = \cos(n \arccos x), \qquad x \in [-1, 1]. \tag{9}$$

They also satisfy the following recurrence relation

$$\begin{aligned} &T_0(x) = 1, \\ &T_1(x) = x, \\ &T_{n+1} = 2xT_n(x) - T_{n-1}(x) \quad n \geq 1. \end{aligned} \tag{10}$$

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

$$x_j = \cos\left[\frac{(2j+1)\pi}{2n+2}\right], \qquad j = 0, 1, \cdots, n. \tag{11}$$

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 [10, 30].

Theorem 2.2. *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 ± 1. In particular, for $x \in [-1, 1]$,*

$$|f(x) - p_n(x)| = O(e^{-n(\phi(z_0)+\ln 2)}) \qquad as\ N \to \infty, \tag{12}$$

where the potential function ϕ is defined by the formula

$$\phi(z) = \log \frac{|z - \sqrt{z^2 - 1}|}{2}. \tag{13}$$

2.3. **A brief overview to the fast multipole method.** The original Fast Multipole Method [17] (FMM) aims at computing the following electrostatic potentials

$$\phi(x_i) = \sum_{j=1, j\neq i}^{N} \log|x_i - x_j| q_j, \qquad i = 1, \cdots, N. \tag{14}$$

While the direct method takes $O(N^2)$ operations, the FMM takes $O(N)$ operations for any prescribed precision ϵ. The fast multipole method has been extended and generalized to speed up the computation of various summations and transformations in the past twenty-five years or so, and its applications have exploded.

Recently, there have been many so-called kernel independent fast multipole methods (KIFMMs) (see, for example, [9, 14, 25, 35, 36]) which aim at computing the following general summation

$$\phi(\mathbf{x}_i) = \sum_{j=1}^{N} K(\mathbf{x}_i, \mathbf{y}_j) q(\mathbf{y}_j), \qquad i = 1, \cdots, N. \tag{15}$$

The constraint on the kernel K is fairly mild, requiring only that 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. All 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 [17].

In this paper, we use the KIFMM developed in [35] to speed up the computation of Dv for an arbitrary vector v. We refer the reader to the original award-winning paper [35] for a detailed description of the algorithm. Here we give a rough sketch on the KIFMM in [35]. The input of KIFMM consists of the vector v and N particle locations $\mathbf{r}_j$ $(j = 1, \cdots, N)$ 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 Dv 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 construct those equivalent potentials in $O(N)$ time.

In Table 1, we report the average timing results T_N for computing the N particle hydrodynamic interactions via the KIFMM in [35]. 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. The computation is carried out on a laptop with 2.53 GHz CPU and 1.89 GB memory. 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 2.3. 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. It is possible to reduce this to 4 FMM calls by combining the original FMM and the technique similar to [29].

TABLE 1. Relative error and timing results of KIFMM [35] for the Rotne-Prager-Yamakawa tensor.

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

3. Fast algorithm for generating random displacement vectors

3.1. **Estimating the extreme eigenvalues of D.** We use the safeguarded Lanczos method in [38] combined with kernel-independent FMM to estimate the extreme eigenvalues of D. Starting from an arbitrary vector v, the safeguarded 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. 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 β_{k+1} is the last subdiagonal element of T_{k+1}. During this process, the most expensive step, i.e., the calculation of D^jv $(j = 1, \cdots, k)$ is done via the kernel-independent 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 1.

Remark 3.1. We originally used the plain Lanczos method [15] (i.e., without the term $|e_k^T z_k|\beta_{k+1}$) to estimate the largest eigenvalue of D and a simplified Chebyshev-Davidson algorithm [37] to estimate the smallest eigenvalue of D. The safeguarded Lanczos method [38] was suggested by one of the anonymous referees to this paper during the review process. This has simplified the overall algorithm and reduced the computational time. We would like to thank the referee for suggesting the safeguarded Lanczos method.

Remark 3.2. 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 digit accuracy suffices for our purpose (please see Theorem 3.4 for details). We observe that the number of Lanczos steps is usually less than 8 in most of our test cases. We would also like to remark here that [38] actually provides several safeguard terms with different convergence properties. We have used Equation 2.6 in [38] since we find that it has the best performance for our problem.

Algorithm 1. Estimating the eigenvalue bounds using the safeguarded Lanczos method and KIFMM

Generate a random vector v and normalize it $v_1 = v/\|v\|$.
Set $v_0 = 0$, $\beta_1 = 0$, $tol = 10^{-3}$ and $m = 12$.
for $k = 1, 2, ..., m$ **do**
  Use KIFMM 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 $\mu_1 \leq \cdots \leq \mu_k$ and associated eigenvectors $z_1, \cdots, z_k$ of T_k.
    Compute $Ub_k = \lambda_{\max}(T_k) + |e_k^T z_k|\beta_{k+1}$, $Lb_k = \lambda_{\min}(T_k) - |e_k^T z_1|\beta_{k+1}$.
    if $|Ub_k - Ub_{k-1}|/Ub_{k-1} < tol$ **then**
      Set $\lambda_{\max}(D) = Ub_k$, $\lambda_{\min}(D) = Lb_k$ and **return**.
    end if
  end if
end for
Set $\lambda_{\max}(D) = \lambda_{\max}(T_m) + |e_m^T z_m|\beta_{m+1}$, $\lambda_{\min}(D) = \lambda_{\min}(T_m) - |e_m^T z_1|\beta_{m+1}$.

3.2. **Chebyshev spectral approximation for the square root matrix.** 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 3.3. *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* $x \in [a, b]$,

$$|\sqrt{x} - p_n(x)| = O\left(\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n\right) \qquad \text{as } n \to \infty, \tag{16}$$

where $\kappa = b/a$.

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

$$\sqrt{x} = \sqrt{\frac{b+a}{2} + \frac{b-a}{2}z} \tag{17}$$

and $x \in [a, b] \Longleftrightarrow z \in [-1, 1]$. According to Theorem 2.2, we have

$$|\sqrt{\frac{b+a}{2} + \frac{b-a}{2}z} - p_n(z)| = O(e^{-n(\phi(z_0)+\log 2)}) \qquad \text{as } n \to \infty, \tag{18}$$

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

$$
\begin{aligned}
e^{-n(\phi(z_0)+\log 2)} &= e^{-n\left(\log\frac{|z_0-\sqrt{z_0^2-1}|}{2}+\log 2\right)} \\
&= \left(\frac{b+a}{b-a}+\sqrt{\left(\frac{b+a}{b-a}\right)^2-1}\right)^{-n} \\
&= \left(\frac{\kappa+1}{\kappa-1}+\sqrt{\left(\frac{\kappa+1}{\kappa-1}\right)^2-1}\right)^{-n} \\
&= \left(\frac{\kappa+1+\sqrt{4\kappa}}{\kappa-1}\right)^{-n} \\
&= \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-n} \\
&= \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{n}.
\end{aligned} \tag{19}
$$

Substituting (19) into (18), we obtain the desired result (16). □

We are now ready to state our main analytical result.

Theorem 3.4. *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*

$$
\|\sqrt{D}-p_n(D)\|_2 = O\left(\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n\right) \qquad \text{as } n\to\infty, \tag{20}
$$

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:

$$
n = O\left(\sqrt{\kappa}\log\frac{1}{\epsilon}\right) \qquad \text{as } \kappa\to\infty, \tag{21}
$$

where ϵ is the prescribed precision.

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

$$
\|\sqrt{D}-p_n(D)\|_2 = \|\sqrt{\Lambda}-p_n(\Lambda)\|_2 = O\left(\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n\right) \qquad \text{as } n\to\infty, \tag{22}
$$

where the first equality follows from the fact that the 2-norm is invariant under the orthogonal transformation and the second equality follows from (16). This shows that in order to achieve a prescribed precision ϵ for the polynomial approximation,

we need

$$
\begin{aligned}
n &= O(\log \epsilon / \log \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \right)) \\
&= O(\log \epsilon / \log \left(1 - \frac{2}{\sqrt{\kappa}} \right)) \\
&= O(\sqrt{\kappa} \log \frac{1}{\epsilon}), \qquad \text{as } \kappa \to \infty.
\end{aligned} \tag{23}
$$

□

Remark 3.5. The error estimate (20) 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 [31]) 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.

3.3. **Computation of the matrix square root.** 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

$$
p_n(x) = \sum_{k=0}^{n} c_k \tilde{T}_k(x), \tag{24}
$$

where the expansion coefficients c_k are given by

$$
c_k = \frac{2}{n} \sum_{l=1}^{n} \sqrt{x_l} \tilde{T}_k(x_l), \tag{25}
$$

$$
x_l = \frac{\lambda_{\max}(D) + \lambda_{\min}(D)}{2} + \frac{\lambda_{\max}(D) - \lambda_{\min}(D)}{2} \cos(\frac{(2l+1)\pi}{2n+2}), \tag{26}
$$

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

$$
\begin{aligned}
&\tilde{T}_0 = 1, \\
&\tilde{T}_1(x) = t_a x + t_b, \\
&\tilde{T}_{l+1}(x) = 2(t_a x + t_b)\tilde{T}_l(x) - \tilde{T}_{l-1}(x)
\end{aligned} \tag{27}
$$

with

$$
\begin{aligned}
t_a &= \frac{2}{\lambda_{\max} - \lambda_{\min}}, \\
t_b &= -\frac{\lambda_{\max} + \lambda_{\min}}{\lambda_{\max} - \lambda_{\min}}.
\end{aligned} \tag{28}
$$

Obviously, we have

$$
p_n(D) = \sum_{k=0}^{n} c_k \tilde{T}_k(D), \tag{29}
$$

and thus

$$
p_n(D)v = \sum_{k=0}^{n} c_k \tilde{T}_k(D)v = \sum_{k=0}^{n} c_k v_k, \tag{30}
$$

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

$$\begin{aligned} v_0 &= v, \\ v_1 &= t_a D v_0 + t_b v_0, \\ v_{k+1} &= 2(t_a D v_k + t_b v_k) - v_{k-1}. \end{aligned} \tag{31}$$

Here the kernel-independent fast multipole method 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.

Now we have summarized our algorithm as follows.

Algorithm 2. Chebyshev spectral approximation for computing $\sqrt{D}v$

STEP 1. Precomputation stage

a. Use Algorithm 1 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 ϵ for $\sqrt{x}$ on $[\lambda_{\min}(D), \lambda_{\max}(D)]$.

STEP 2. Computation of the matrix square root

a. Use the recurrence relation (31) and kernel-independent 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 $\sqrt{D}v \approx p_n(D)v$.

Remark 3.6. When v is a normally distributed random vector with mean zero, as in 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. And the expensive step of computing the first m vectors Dv_k $(k = 1, \cdots, m)$ is avoided. We have implemented this reduction in our code.

Remark 3.7. Obviously, 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 (see the above remark) 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 3.4, the computational cost of our algorithm is $O(\sqrt{\kappa}N)$.

4. Numerical results

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

Although Theorem 3.4 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; the computational cost of this step is negligible compared with that of the other steps. In Table 2, we report the number of terms needed in the Chebyshev polynomial approximation. The first column indicates the desired precision ϵ. The first row indicates the condition number κ of the matrix D.

TABLE 2. The 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 ϵ. The first row indicates the condition number κ.

$\epsilon \backslash \kappa$	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

Next, we observe that the overall complexity of our algorithm is $O(\sqrt{\kappa}N)$. Thus it is important to investigate the condition number κ 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 3, 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.

TABLE 3. The condition number κ 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

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 the condition number κ is roughly constant along the diagonals in Table 3. This indicates that κ 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

TABLE 4. Relative error and timing results for generating random displacement vectors. $Na/L = 1$ is fixed.

N	T_{Lanc}	m	κ	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

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.

We have tested the performance of our algorithm with Na/L fixed for various Ns. The average timing results for various particle configurations are summarized in Table 4. In Table 4, the first column contains the total number of particles. The second column contains the time needed (in seconds) for the safeguarded 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 $\sqrt{D}v$. The last column contains the relative error of the computational result, where the relative error is defined as the average of $\left|\frac{V_{\text{eval}}^n - V_{\text{true}}}{V_{\text{true}}}\right|$ and $\frac{\|v_n - v_{n-1}\|}{\|v_n\|}$ with $v_n = p_n(D)v$ the nth approximation of $\sqrt{D}v$, $v_{n-1} = p_{n-1}(D)v$, $V_{\text{eval}}^n = v_n^T v_n$, and $V_{\text{true}} = v^T Dv$. 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.

Finally, we have also tested our algorithm for various particle configurations with L and a fixed. The results with $L = 1000$ and $a = 0.1$ are reported in Table 5.

TABLE 5. Relative error and timing results for generating random displacement vectors. $L = 1000$, $a = 0.1$.

N	T_{Lanc}	m	κ	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

5. CONCLUSION

We have presented a fast algorithm for generating random vectors whose spatial correlation is determined by the hydrodynamic interactions, the Rotne-Prager-Yamakawa tensor in particular. The complexity of the algorithm is $O(\sqrt{\kappa}N)$ with κ the condition number of the tensor and N the total number of Brownian particles. Our algorithm 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 algorithm is useful in many other applications, including the statistical analysis with certain spatial correlations and model

reduction in dynamic control theory. Finally, we observe 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, a fast direct algorithm which compresses the square root matrix (an idea similar to [16, 24]) would be more useful. This approach is currently under investigation and results will be reported in the future.

Acknowledgements

The authors would like to thank the anonymous referees for carefully reading the manuscript and for suggesting the safeguarded Lanczos method in [38] for estimating eigenvalue bounds of large Hermitian matrices.

References

[1] G. K. Batchelor, *Brownian Diffusion of Particles with Hydrodynamic Interaction*. J. Fluid Mech. **74** (1976), 1-29. MR0406082 (53:9874)

[2] 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. **22** (2001), 1112-1125. MR1825853 (2002a:65069)

[3] G. Dahlquist and A. Bjorck, *Numerical Methods*, Prentice Hall, Englewood Cliffs (1974). MR0368379 (51:4620)

[4] M. H. DeGroot and M. J. Schervish, *Probability and Statistics*, third edition, Addison Wesley, 2002.

[5] J. M. Deutch and I. Oppenheim, *Molecular Theory of Brownian Motion for Several Particles*. J. Chem. Phys. **54** (1971), 3547-3554.

[6] D. L. Ermak and J. A. McCammon, *Brownian Dynamics with Hydrodynamic Interactions*. J. Chem. Phys. **69** (1978), 1352-1360.

[7] B. U. Felderhof and J. M. Deutch, *Frictional Properties of Dilute Polymer Solutions I. Rotational Friction Coefficient*. J. Chem. Phys. **62** (1975), 2391-2397.

[8] M. Fixman, *Implicit Algorithm for Brownian Dynamics of Polymers*. Macromolecules **19** (1986), 1195–1204.

[9] W. Fong and E. Darve, *The Black-Box Fast Multipole Method*. J. Comp. Phys. **228** (2009), 8712-8725. MR2558773 (2010m:65009)

[10] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, 1998. MR1386891 (97g:65001)

[11] H. L. Friedman, *A Hydrodynamic Effect in the Rates of Diffusion Controlled Reactions*. J. Chem. Phys. **70** (1966), 3931-3933.

[12] P. G. de Gennes, *Passive Entry of a DNA Molecule into a Small Pore*. Proc. Natl. Acad. Sci. USA **96** (1999), 7262-7264.

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

[14] Z. Gimbutas and V. Rokhlin, *A Generalized Fast Multipole Method for Nonoscillatory Kernels*. SIAM J. Sci. Comput. **24** (2003), 796-817. MR1950512 (2004a:65176)

[15] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press (1996). MR1417720 (97g:65006)

[16] L. Greengard, D. Gueyffier, P.G. Martinsson and V. Rokhlin, *Fast Direct Solvers for Integral Equations in Complex Three-Dimensional Domains*. Acta Numerica **18** (2009), 243-275. MR2506042 (2010e:65252)

[17] L. Greengard and V. Rokhlin, *A Fast Algorithm for Particle Simulations*. J. Comp. Phys. **73** (1987), 325-348. MR918448 (88k:82007)

[18] L. Greengard and V. Rokhlin, *A New Version of the Fast Multipole Method for the Laplace Equation in Three Dimensions*. Acta Numerica **6** (1997), 229-269. MR1489257 (99c:65012)

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

[20] J. P. Hernandez-Ortiz, J. J. de Pablo, and M. D. Graham, *Fast Computation of Many-Particle Hydrodynamic and Electrostatic Interactions*. Phys. Rev. Letters **98** (2007), 140602.
[21] G.A. Huber, and J.A. McCammon, *Browndye: A Software Package for Brownian Dynamics*. Computer Physics Communications **181** (2010), 1896-1905.
[22] M. Karplus and D. L. Weaver, *Protein Folding Dynamics*. Nature **260** (1976), 404-406.
[23] 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. **43** (1999), 267-304.
[24] P. G. Martinsson and V. Rokhlin, *A Fast Direct Solver for Boundary Integral Equations in Two Dimensions*. J. Comput. Phys. **205** (2005), no. 1, 1–23. MR2132300 (2005k:65292)
[25] P. G. Martinsson and V. Rokhlin, *An Accelerated Kernel-Independent Fast Multipole Method in One Dimensions*. SIAM J. Sci. Comput. **29** (2007), 1160-1178. MR2318701 (2008f:65084)
[26] J . A. McCammon, J. M. Deutch, and B. U. Felderhof, *Frictional Properties of Multisubunit Structures*. Biopolymers **14** (1975), 2613-2623.
[27] D. Petera and M. Muthukumar, *Brownian Dynamics Dimulation of Bead-rod Chains under Shear with Hydrodynamic Interaction*. J. Chem. Phys. **111** (1999), 7614-7623.
[28] J. Rotne and S. Prager, *Variational Treatment of Hydrodynamic Interaction in Polymers*. J. Chem. Phys. **50** (1969), 4831-4837.
[29] A. Tornberg and L. Greengard, *A Fast Multipole Method for the Three-dimensional Stokes Equations*. J. Comput. Phys. **227** (2008), no. 3, 1613–1619. MR2450963 (2009g:76110)
[30] L. N. Trefethen, *Spectral Methods in Matlab*, SIAM (2000). MR1776072 (2001c:65001)
[31] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997. MR1444820 (98k:65002)
[32] P. G. Wolynes and J. M. Deutch, *Slip Boundary Conditions and the Hydrodynamic Effect on Diffusion Controlled Reactions*. J. Chem. Phys. **65** (1976), 450-454.
[33] P. G. Wolynes and J. M. Deutch, *Dynamical Orientation Correlations in Solution*. J. Chem. Phys. **67** (1977), 733-741.
[34] H. Yamakawa, *Transport Properties of Polymer Chains in Dilute Solution: Hydrodynamic Interaction*. J. Chem. Phys. **53** (1970), 436-443.
[35] L. Ying, G. Biros, and D. Zorin, *A Kernel-Independent Adaptive Fast Multipole Algorithm in Two and Three Dimensions*. J. Comp. Phys. **196** (2004), 591-626. MR2054351 (2005d:65235)
[36] B. Zhang, *Integral-Equation-Based Fast Algorithms and Graph-Theoretic Methods for Large-Scale Simulations*. PhD thesis, University of North Carolina, Chapel Hill, 2010.
[37] Y. Zhou and Y. Saad, *A Chebyshev-Davidson Algorithm for Large Symmetric Eigenvalue Problems*. SIAM J. Matrix Anal. App. **29** (2007), 954-971. MR2365900 (2008k:65081)
[38] Y. Zhou and R. C. Li, *Bounding the Spectrum of Large Hermitian Matrices*. Linear Algebra and its App. **435** (2011), 480-493. MR2794587

Department of Mathematical Sciences, New Jersey Institute of Technology, Newark, New Jersey 07102
E-mail address: shidong.jiang@njit.edu

Department of Mathematical Sciences, New Jersey Institute of Technology, Newark, New Jersey 07102
E-mail address: zl28@njit.edu

Department of Mathematics, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina 27599
E-mail address: huang@amath.unc.edu