PHYSICAL REVIEW E **81**, 020902(R) (2010)

Treecode algorithm for pairwise electrostatic interactions with solvent-solute polarization

Zhenli Xu*

Department of Mathematics, Shanghai Jiao Tong University, Shanghai 200240, China

and Department of Mathematics and Statistics, University of North Carolina, Charlotte, North Carolina 28223, USA

(Received 26 September 2009; revised manuscript received 18 January 2010; published 9 February 2010)

An $O(N \log N)$ treecode algorithm is presented for computing pairwise interactions of electrostatic free energy for reaction potentials with polarization effects due to the macroscopic solvent. A multipole expansion for a cluster is used to account for particles inside the cluster, where a spatial difference is applied to obtain the expansion coefficients of the polarization function. Numerical tests are performed to illustrate the accuracy and efficiency of the approach. The algorithm is significant in speeding up generalized Born methods for biomolecular simulations under the framework of macroscopically treating solvents.

DOI: 10.1103/PhysRevE.81.020902 PACS number(s): 87.15.ap, 41.20.Cv, 02.70.Ns

I. INTRODUCTION

Hierarchical treecode algorithms [1–7] have been used extensively in studying particle interactions in various applications such as astrophysics, biophysics, computational chemistry, and mechanical engineering. To introduce the algorithm, the particles are usually divided into several nested clusters with an octree structure. A multipole expansion is used for a particle-cluster interaction when the particle is far from the cluster, otherwise a direct summation is used for a short-range interaction. By this way, the computational complexity is reduced from $O(N^2)$ to $O(N \log N)$ operations for Barnes-Hut-type treecodes [2], or $O(N)$ for the so-called fast multipole method (FMM) [3] if an ingenious step converting the multipole expansion to a local expansion is performed.

Treecodes have been developed for computing the Coulomb, van der Waals, Yukawa, and Lennard-Jones potentials, and so on; see, for example, [8]. In this work, a treecode algorithm is presented for calculating the following "polarized" Coulomb interactions of an N-particle system:

$$G = \sum_{j=1}^{N} q_j \Phi_j, \tag{1}$$

with a reaction-field potential of particle j given by

$$\Phi_j = \sum_{i=1}^{N} \frac{q_i}{\sqrt{|\mathbf{r}_i - \mathbf{r}_j|^2 + c_{ij}}}, \tag{2}$$

where q_i and $\mathbf{r}_i$ are the charge and location of particle i, and c_{ij} is a positive, smooth, and slowly varying function of locations. In the generalized Born (GB) theory [9–14] of biomolecular simulations, G is known as the solvation free energy of a biomolecule immersed into an aqueous solution. Also the function is assumed to be separable or approximately separable; i.e., $c_{ij} \approx c_i c_j$, where c_i is the approximate Born radius reflecting the polarization effect of the bulk solvent and usually represented by a volume integral over the solvent region. It should be remarked that, in most GB methods, c_{ij} takes the form of $c_i c_j \exp(-r_{ij}^2/4c_i c_j)$. As the exponential term is actually very close to 1, c_{ij} can be easily separated by taking an average of this term in a cluster.

In molecular-dynamics simulations of biological systems, there are two principal approaches of computational models for electrostatic interactions, namely, the explicit and implicit models. Explicit solvation models, which represent the solvent in full atomistic detail and the particle interactions are governed by the Coulomb's law, are limited by high computational cost. Macroscopic implicit solvation models [14–16] replace the solvent with a high dielectric continuum media and thus offer a cheaper way for simulations of the biological systems by greatly reducing the number of the degree of freedom. However, although implicit simulation methods have advanced dramatically in recent years, it is well known that the lack of fast and accurate methods to calculate electrostatic interactions remains the bottleneck of computational speedup, where the accuracy and speed are usually two opposing objectives. An accurate description of the solvent environment with the Poisson-Boltzmann (PB) theory is essential for realistic simulations, for which numerical solutions such as finite difference and finite element methods have to be used and it is still at a high computational cost. The GB formula produces an accurate approximation to the Poisson result, while still fast enough to be applied in molecular-dynamics simulations. The GB methods have been widely studied [10,12,17] in last two decades owing to their applicability and performance. However, the desiring demand in accuracy and speed remains unresolved when implementing the GB methods.

Recently, much interest has emerged regarding fast numerical implementations of macroscopic electrostatic models [18–22] such as for the PB model and the hybrid reaction-field model. Their applications in large macromolecular simulations confront many issues, for example, the mesh generation and force calculations. In contrast, the use of the GB model is straightforward in these simulations once the faster algorithms have been developed for computing the Born radii and the pairwise summation. The GB methods have been widely used in molecular-dynamics simulations for a long time as they are simple and at the same time very accurate macroscopic electrostatic models. A linear scaling implementation of the pairwise interactions is significant for studying macromolecules, where the method of the direct summation is limited if the particle number reaches a mag-

*xuzl@ustc.edu

nitude of a hundred thousand. The fast Fourier transform techniques [23,24] have been developed for Born radii evaluations. The present algorithm then aims to be an accelerator of the pairwise summation involved in the GB model.

The primary aim of this Rapid Communication is to present a treecode of computing [Eq. (1)]. Extension to the force calculation is solvable, as we have a process of reconstructing the Born radii derivatives. However, how to improve the accuracy in force calculations and couple it with a fast Born radii solver will be important issues, which would be pursued in future publications. Dimensionless units are used throughout this work.

II. TREECODE ALGORITHM

To compute Eq. (2) in $O(\log N)$ operations for each particle, the core idea is to use a multipole expansion to the potential function for particles in a cluster. Let us consider the interaction between particles in a cubic cluster A with center $\mathbf{r}_A$ and a distant particle j. The reaction potential for a location $\mathbf{r}$ inside the cluster due to charge j has the multidimensional Taylor expansion with respective to $\mathbf{r}_A$:

$$\Phi(\mathbf{r}) = \frac{q_j}{\sqrt{|\mathbf{r}-\mathbf{r}_j|^2 + c_j c(\mathbf{r}-\mathbf{r}_A)}} = \sum_{\|\alpha\|\geq 0} \frac{1}{\alpha!}\partial^\alpha \Phi(\mathbf{r}_A-\mathbf{r}_j)(\mathbf{r}-\mathbf{r}_A)^\alpha, \tag{3}$$

where function $c(\mathbf{r}-\mathbf{r}_A)$ is an approximate function from $\{c_i, i\in A\}$ and vector $\alpha=(\alpha_1,\alpha_2,\alpha_3)$ is represented by the multi-index notation often used in multivariable calculus. By using a pth order Taylor approximation, the reaction potential of j due to the cluster is given by

$$\Phi_{j,A} \approx \sum_{\|\alpha\|=0}^{p} \frac{1}{\alpha!}\partial^\alpha \Phi(\mathbf{r}_A-\mathbf{r}_j)\sum_{i\in A} q_i(\mathbf{r}_i-\mathbf{r}_A)^\alpha = \sum_{|\alpha|=0}^{p} T_\alpha M_A^\alpha, \tag{4}$$

where $T_\alpha=(1/\alpha!)\partial^\alpha\Phi(\mathbf{r}_A-\mathbf{r}_j)$ is the αth order coefficient of the Taylor expansion and $M_A^\alpha=\Sigma_{i\in A}q_i(\mathbf{r}_i-\mathbf{r}_A)^\alpha$ is the αth multipole moment of cluster A. As we only need to compute once the moments for each cluster, there are $\frac{3}{2}p(p+1)+1$ operations, provided the Taylor expansion coefficients, for each particle-cluster interaction, which is independent of the particle number in the cluster. This process results in an $O(N\log N)$ calculation of the pairwise summation, where $\log N$ comes from the number of clusters.

Suppose function $c(\mathbf{r}-\mathbf{r}_A)$ has the following Taylor expansion:

$$c(\mathbf{r}-\mathbf{r}_A) = \sum_{\|\alpha\|=0}^{\infty} \frac{c_\alpha}{\alpha!}(\mathbf{r}-\mathbf{r}_A)^\alpha, \tag{5}$$

where c_α are the coefficients. Then T_α can be calculated by a Taylor series expansion. When $\|\alpha\|\leq 2$, these are written as

$$T_{\mathbf{0}} = \frac{1}{(r_{j,A}^2 + c_j c_{\mathbf{0}})^{1/2}}, \tag{6}$$

$$T_{\mathbf{e}_k} = -\frac{1}{2}(2x_k + c_j c_{\mathbf{e}_k})(T_{\mathbf{0}})^3, \tag{7}$$

$$T_{2\mathbf{e}_k} = \frac{3}{2}\frac{(T_{\mathbf{e}_k})^2}{T_{\mathbf{0}}} - \frac{2 + c_j c_{2\mathbf{e}_k}}{4}(T_{\mathbf{0}})^3, \tag{8}$$

$$T_{\mathbf{e}_k+\mathbf{e}_l} = \frac{3T_{\mathbf{e}_k}T_{\mathbf{e}_l}}{T_{\mathbf{0}}} - \frac{c_j c_{\mathbf{e}_k+\mathbf{e}_l}}{2}(T_{\mathbf{0}})^3, \quad k\neq l, \tag{9}$$

where $r_{j,A}=|\mathbf{r}_j-\mathbf{r}_A|$, $x_k=(\mathbf{r}_A-\mathbf{r}_j)\cdot\mathbf{e}_k$, and $\mathbf{e}_k$ is the unit vector of the kth coordinate.

A difficulty is we have only the information of point values $c(\mathbf{r})$ at the particle locations. The present algorithm truncates the expansion to $p=1$ and currently higher-order truncations remain a challenge due to the difficulty of reconstructing coefficients c_α. Nevertheless, it leads to a relative error of a few 0.1% in free-energy calculations, which is often sufficiently accurate in practical simulations. The code is a modified version of that developed by Krasny and his collaborators [5,25–27]. The octree is constructed by dividing the simulation box into a hierarchy of clusters, where the root cluster is the smallest box containing all the particles and it is bisected in each dimension into eight children. Recursive subdivisions are applied to each child until the number of particles in a cube is less than a specified number N_0, which is a leaf of the created tree data structure. For each cluster A, the zero-order Taylor coefficient $c_{\mathbf{0}}^A$ of function $c(\mathbf{r}-\mathbf{r}_A)$ is computed by an average over the particles within it, and each first-order coefficient is constructed by a difference of $c_{\mathbf{0}}^A$ of its eight children as

$$c_{\mathbf{e}_k}^A = \frac{\sum_{B\in A_{\text{child}}^+} c_{\mathbf{0}}^B - \sum_{B\in A_{\text{child}}^-} c_{\mathbf{0}}^B}{2\Delta x_k}, \tag{10}$$

where A_{child}^+ (or A_{child}^-) are the four children clusters of A satisfying the center coordinate x_k larger (or smaller) than that of its father cluster, and Δx_k is the size of box A along x_k direction. If the corresponding child is null, the value $c_{\mathbf{0}}^B$ takes $c_{\mathbf{0}}^A$. Furthermore, if cluster A is a leaf, a linear least-squares fitting [28] is applied to construct the approximate linear function and the first-order Taylor coefficient $c_{\mathbf{e}_k}^A$.

The potential Φ_j in Eq. (2) of a particle j is computed by traversing the tree. This is done recursively by starting from the root cluster. At each step relevant to cluster A, a multipole acceptance criterion (MAC) [27],

$$\frac{\sqrt{\Delta x_1^2+\Delta x_2^2+\Delta x_3^2}}{2|\mathbf{r}_j-\mathbf{r}_A|} \leq \theta, \tag{11}$$

for a specified error-control parameter θ is first decided. If the MAC is satisfied, i.e., j and A are well separated, $\Phi_{j,A}$ is computed by the pth order multipole expansion, where, if the moments M_A^α and the coefficients c_α^A are not available, they are first calculated and stored for the use of other particle-cluster interactions relevant to cluster A. If the MAC is not satisfied, then the algorithm computes $\Phi_{j,A}$ by a direct summation if A is a leaf of the tree or descends the tree to the next level if A is not a leaf.

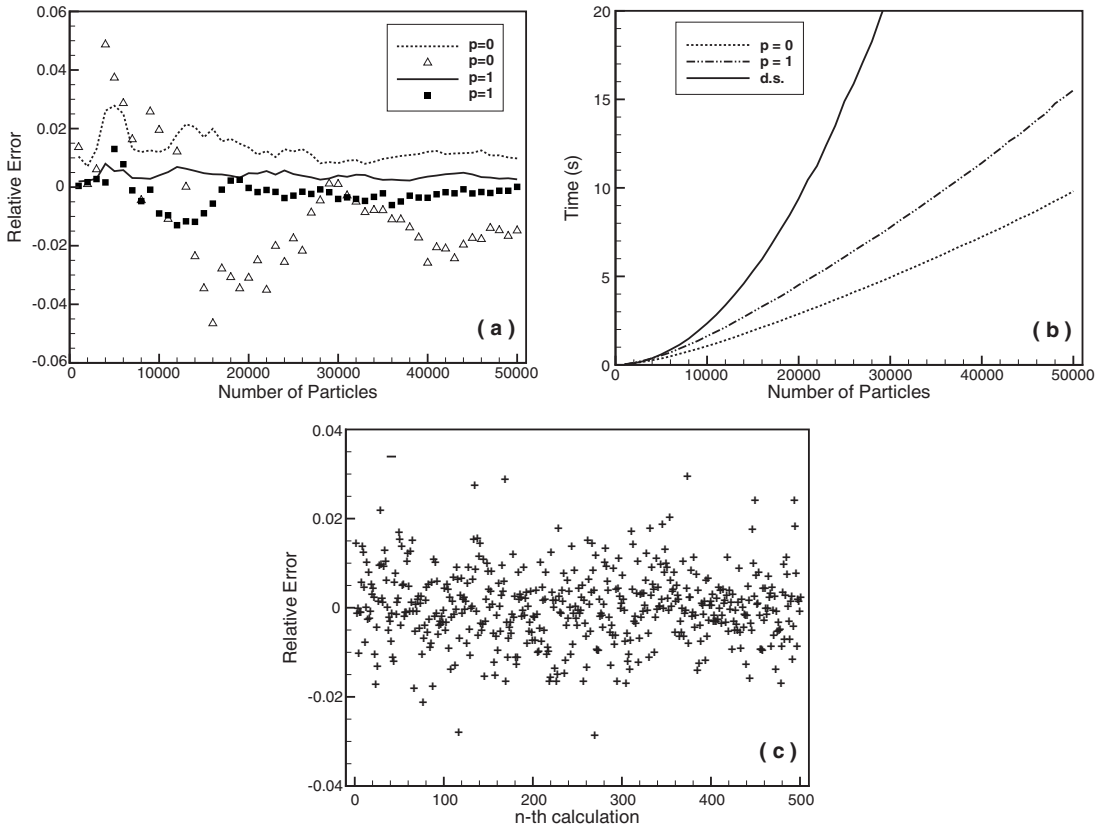

FIG. 1. Accuracy and CPU time performance of the tree code for sphere. (a) The relative errors E_1 (lines) and E_G (symbols) for the results of the treecode algorithm with $p=0$ and 1. (b) The CPU times of the treecode algorithm and the direct summation. (c) E_G errors of 500 tests for 4000 randomly generated particles.

III. RESULTS

Numerical examples are performed to test the algorithm by a model of a spherical geometry and two biomolecules. In all the calculations, we set $\theta=0.3$ for the MAC parameter and $N_0=20$ for the maximum particle number in a leaf. To illustrate the performance, the results of $p=1$ is compared to those of $p=0$ by taking those of direct summation as the exact solution. The calculations are run on a Windows PC machine with a 3.0 GHz dual-core CPU.

In the first test, particles are randomly generated inside a unit sphere ($r_i<0.95$ in order to avoid singularity) and each particle has a random charge between −0.5 and 0.5. The Born radius function is taken as a quadratic function of r as $c_i=1-r_i^2$ which is from the R6 model [13,29,30] of the sphere. For each calculation, we compute the L_1 relative error of the potential Φ_j,

$$E_1=\frac{1}{N}\sum_{j=1}^{N}\frac{|\Phi_j-\Phi_j^{\mathrm{ds}}|}{\max|\Phi^{\mathrm{ds}}|}, \tag{12}$$

and the free-energy relative error,

$$E_G=\frac{G-G^{\mathrm{ds}}}{G^{\mathrm{ds}}}, \tag{13}$$

where the superscript ds represents the results by the direct summation.

Figure 1 shows the relative errors and the CPU time as a function of the number of particles N increasing from 1 to 50 k for two truncation orders of the multipole expansion. The break-even points of $p=0$ and 1 are about 2 and 4 k, respectively. It can be observed that the treecode algorithm is with an order of $O(N \log N)$ and it is significantly faster than the direct summation; when the particle number is 50 k, the speed up of $p=1$ is a factor around 4.0. The L_1 relative errors of the $p=1$ case are a few 0.1% and all less than 1%, which averagely have about four times less than those of the $p=0$ case. The performance measured by the free-energy error is also reasonable; most of calculations are less than 1%, which are significantly better than $p=0$. A tendency can be also seen that the fluctuation of free-energy errors is decreased with increasing of particles. In Fig. 1(c), we also illustrates 500 tests for 4000 randomly generated particles in order to verify the accuracy for different distributions of the same number of particles. We can see from the plot that most of the errors (97.8% tests) are less than 2%, and all of them are less than 3%, demonstrating that although the free energy is a sum of signed data and the denominator in (13) may be a small number, the error performance of the algorithm remains attractive.

Next, we test the algorithm by two protein molecules with PDB access codes 3GB1 and 1OCA. Our concern is to show the accuracy of the proposed algorithm for proteins which are the primary objects. These two proteins contain 855 atoms and 2503 atoms, respectively, which have significantly different sizes and thus serve well the aim of testing the

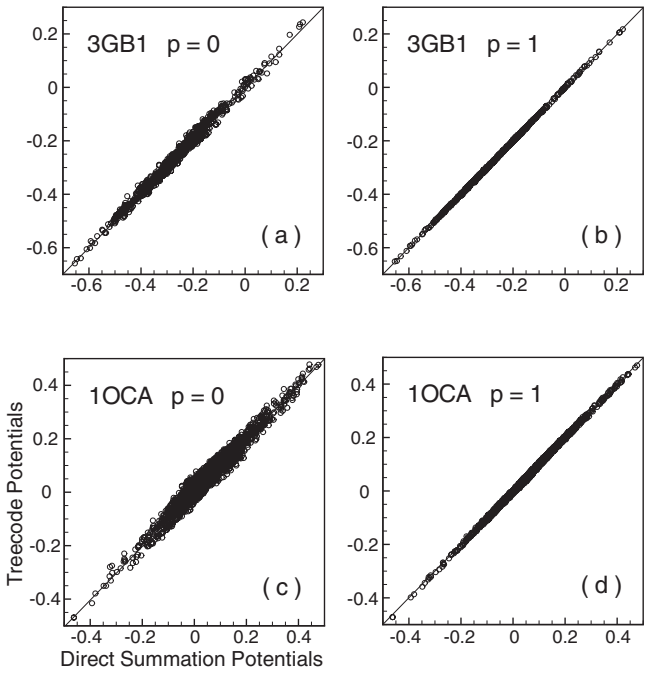

FIG. 2. Treecode ($p=0$ and 1) vs direct summation for the results of Φ_j. Two protein molecules 3GB1 and 1OCA are calculated.

performance of the algorithm. Before the calculations, the Born radii are provided by using the PBEQ module of the macromolecular modeling package CHARMM [31]. The treecode vs direct summation results of the reaction potential Φ_j are shown in Fig. 2. Clear improvement in accuracy can be observed for the $p=1$ case in comparison to the $p=0$ one. We also calculate the E_1 and E_G errors for $p=1$. The E_1 relative errors of the 3GB1 and 1OCA are 0.31% and 0.96%, respectively, and the E_G errors are 0.12% and 0.25%, which are sufficiently accurate for practical applications.

IV. CONCLUSION

In this work, we have introduced a treecode algorithm for evaluating electrostatic interactions in presence of solvent-solute polarization, which is significant in generalized Born theory of macromolecular simulations. Test calculations for spherical models and two protein molecules demonstrate the accuracy and efficiency of the method. As the performance of the algorithm is shown attractive, we plan to further study this algorithm for force calculations and couple it with a fast Born radii solver to obtain a complete package for biomolecular electrostatic computation.

ACKNOWLEDGMENTS

The author thanks Professor W. Cai and Professor A. Baumketner for discussions of GB methods and the financial support from the Program for New Century Excellent Talents in University.

[1] A. Appel, SIAM J. Sci. Comput. (USA) **6**, 85 (1985).
[2] J. Barnes and P. Hut, Nature (London) **324**, 446 (1986).
[3] L. Greengard and V. Rokhlin, J. Comput. Phys. **73**, 325 (1987).
[4] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems* (MIT, Cambridge, 1987).
[5] Z. H. Duan and R. Krasny, J. Chem. Phys. **113**, 3492 (2000).
[6] H. Cheng, L. Greengard, and V. Rokhlin, J. Comput. Phys. **155**, 468 (1999).
[7] L. Ying, G. Biros, and D. Zorin, J. Comput. Phys. **196**, 591 (2004).
[8] B. Shanker and H. Huang, J. Comput. Phys. **226**, 732 (2007).
[9] W. C. Still, A. Tempczyk, R. C. Hawley, and T. Hendrickson, J. Am. Chem. Soc. **112**, 6127 (1990).
[10] D. Bashford and D. A. Case, Annu. Rev. Phys. Chem. **51**, 129 (2000).
[11] M. S. Lee, F. R. Salsbury, and C. L. Brooks III, J. Chem. Phys. **116**, 10606 (2002).
[12] J. Chen, C. L. Brooks III, and J. Khandogin, Curr. Opin. Struct. Biol. **18**, 140 (2008).
[13] T. Grycuk, J. Chem. Phys. **119**, 4817 (2003).
[14] M. Feig and C. L. Brooks III, Curr. Opin. Struct. Biol. **14**, 217 (2004).
[15] C. J. Cramer and D. G. Truhlar, Chem. Rev. **99**, 2161 (1999).
[16] B. Roux, *Implicit Solvent Models, Computational Biochemistry, and Biophysics* (Dekker, New York, 2001).
[17] Z. Xu and W. Cai (unpublished).
[18] A. H. Boschitsch, M. O. Fenley, and H. X. Zhou, J. Phys. Chem. B **106**, 2741 (2002).
[19] B. Lu, X. Cheng, J. Huang, and J. A. McCammon, Proc. Natl. Acad. Sci. U.S.A. **103**, 19314 (2006).
[20] B. Z. Lu, Y. C. Zhou, M. J. Holst, and J. A. McCammon, Comm. Comp. Phys. **3**, 973 (2008).
[21] M. Baptista, R. Schmitz, and B. Dunweg, Phys. Rev. E **80**, 016705 (2009).
[22] Y. Lin, A. Baumketner, S. Deng, Z. Xu, D. Jacobs, and W. Cai, J. Chem. Phys. **131**, 154103 (2009).
[23] W. Cai, Z. Xu, and A. Baumketner, J. Comput. Phys. **227**, 10162 (2008).
[24] T. Schupbach, V. Zoete, B. Tsakam-Sotche, and O. Michielin, J. Comput. Chem. **31**, 649 (2010).
[25] Z. H. Duan and R. Krasny, J. Comput. Chem. **22**, 184 (2001).
[26] K. Lindsay and R. Krasny, J. Comput. Phys. **172**, 879 (2001).
[27] P. Li, H. Johnston, and R. Krasny, J. Comput. Phys. **228**, 3858 (2009).
[28] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing* (Cambridge University Press, New York, 1992).
[29] J. Mongan, W. A. Svrcek-Seiler, and A. Onufriev, J. Chem. Phys. **127**, 185101 (2007).
[30] H. Tjong and H.-X. Zhou, J. Phys. Chem. B **111**, 3055 (2007).
[31] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaninathan, and M. Karplus, J. Comput. Chem. **4**, 187 (1983).