Optimizing Ewald summation for Monte Carlo simulations

A.C. Maggs

Laboratory de Physicochimie théorique, Gulliver, CNRS-ESPCI, 10 rue Vauquelin, 75005, Paris.

We consider the Ewald formula for the electrostatic interaction between charges. We introduce three optimizations. First when working with few particles we show how to simplify the evaluation of the Fourier part of the interaction. Second we reduce the artefacts that arise from truncating the sum. Last we show that when working with a large number of particles energy changes can be calculated with an effort scaling with the square root of the number particles.

I. INTRODUCTION

The Ewald formula is the main workhorse for the evaluation of electrostatic interactions in the presence of periodic boundary conditions. The conditionally convergent Coulomb sum is split into rapidly convergent real-space and Fourier-space series, which are truncated in order to evaluate the energy to a desired accuracy. When working with $N \gg 1$ particles one optimizes the splitting between real and Fourier series in such a way that the numerical effort needed to calculate the total energy varies as $\tau \sim N^{3/2}$, Ref.[1]. Variations of the method that use fast Fourier methods on a grid to solve the Poisson equation lead to algorithms that have even more favorable scalings with system size $\tau \sim N \log N$, Ref.[2], even if the precision of the result is lower. These grid methods are widely used in molecular simulation though considerable effort is needed to control lattice artefacts.

Just occasionally one desires a code that gives the Ewald sum for a small numbers of particles. It is this case which is the subject of the first part of the present note. We show that rather elementary identities allow one to improve the efficiency of codes. We have found the ideas useful, for instance, when developing and calibrating local $O(N)$ electrostatics solvers based on auxiliary field updates[3,4].

When working with the Ewald formula the question arises as to how to optimally truncate the summation in order to reduce artefacts. We introduce a simple method of generating smooth truncations which considerably reduces the numerical error.

In the last part of the paper we demonstrate that changes in energy, such as those needed in the Metropolis method, can be evaluated with an effort which varies as $\sqrt{N}$ once the full energy has been calculated.

II. EWALD FORMULA

We use a normalization for the potential such that the Coulomb interaction in a non-periodic space is $1/r$ and consider a cubic, unit simulation cell. The Ewald method splits the interaction between a pair of particles into real and Fourier space components in the following way

$$U_w(\mathbf{r}) = \sum_{\mathbf{n}} U_r(|\mathbf{r} + \mathbf{n}|) + U_f(\mathbf{r}) + U_0$$

$$U_r(r) = \frac{\text{erfc}(\alpha r)}{r}$$

$$U_f(\mathbf{r}) = \frac{1}{\pi} \sum_{\mathbf{m} \neq 0} \frac{e^{-\pi^2 \mathbf{m}^2/\alpha^2}}{\mathbf{m}^2} \cos(2\pi \mathbf{m} \cdot \mathbf{r}) \qquad (1)$$

$$U_0 = -\frac{\pi}{\alpha^2}$$

We remind the reader that the long range expansion of erfc(x) is given by

$$\text{erfc}(x) = \frac{e^{-x^2}}{x\sqrt{\pi}} \left(1 - \frac{1}{2x^2} \cdots \right)$$

The zero body potential, U_0 is chosen such that the spatial average of the potential is zero. The sum of these three terms is independent of α *if the sums are not truncated.* When the sums are truncated there are numerical errors that we now estimate in order to indicate how to choose the parameters in the sum.

Both the real and Fourier sums converge exponentially fast with cut-off radius; for such rapidly decreasing functions we estimate the cut-off radii by solving the equations

$$e^{-\alpha^2 r_c^2} = \epsilon \qquad (2)$$

$$e^{-\pi^2 m_c^2/\alpha^2} = \epsilon$$

where r_c and m_c are the cut-offs in the real and Fourier sums; ϵ is the truncation error. Let $z = \log(1/\epsilon)$. We assume that the Fourier and real-space function evaluations require times τ_f and τ_r per evaluation. The time required to evaluate the energy with the Ewald formula is comparable to

$$\tau = \frac{4\pi}{3}\left(\tau_r r_c^3 + \tau_f m_c^3\right) = \frac{4\pi}{3} z^{3/2} \left(\tau_r \frac{1}{\alpha^3} + \tau_f \frac{\alpha^3}{\pi^3}\right)$$

We minimize over α to find that $\alpha^6 = \pi^3 \frac{\tau_r}{\tau_f}$ so that

$$\tau \sim z^{3/2}\sqrt{\tau_r \tau_f} \qquad (3)$$

In order to improve the speed of the algorithm by one order of magnitude one requires a two orders of magnitude increase in the speed of τ_f, or faster converging sums in order to reduce the cut-off radii. Our first optimizations will play on both of the features.

III.   OPTIMIZATION FOR SMALL NUMBERS OF PARTICLES

Our first optimization is based on the trivial remark that when the components of $\mathbf{m}$ are all non-zero there are 8 symmetry-related contributions to the Fourier sum eq. (1) corresponding to changing the sign of each component of $\mathbf{m}$: $m_i \to \pm m_i$ for which the modulus of $\mathbf{m}$, and thus the weights are identical. Elementary manipulations then show that

$$\sum_{i=1}^{8} \cos\left(2\pi\mathbf{m}_i \cdot \mathbf{r}\right) =$$
$$8\cos\left(2\pi m_1 x\right)\cos\left(2\pi m_2 y\right)\cos\left(2\pi m_3 z\right) \qquad (4)$$

This identity allows considerable simplification of the evaluation of the Fourier sum.[5] Rather than requiring $\frac{4\pi}{3}m_c^3$ evaluations of the costly function cosinus we can calculate all the trigonometric functions with only $3m_c$ function calls. The exponential weight is always evaluated for the same arguments, and can be pre-calculated and stored in an array. Thus in the calculation of the Fourier space contribution to the interaction the only part of the calculation that scales as m_c^3 in complexity is the multiplication of the pre-calculated array with $3m_c$ elements. On a typical modern workstation this multiplication is the slowest step. Since multiplications are about a factor 10 faster than complex function evaluations and we reduce the calculation to a sum over the positive octant we find an approximately 80-fold decrease in τ_f from this simple re-arrangement of the calculation. We note the use of similar manipulations of the Ewald sum in a more complicated planar geometry[6].

Until now we have assumed a spherical cut-off in the Fourier sum. We found that using a cubic cut-off such that $0 \le m_i < m_c$ was much more efficient for the multiplication of the weights; the multiplication code vectorizes on modern desktop processors and is thus rather efficient. Introducing the extra logic to restrict the summation so that to a smaller volume, $0 < |\mathbf{m}| < m_c$, $m_i \ge 0$, led to *substantial slow down* (by a factor two) in the code, even if less arithmetical operations are performed. A bonus is that the use of a cubic cut-off leads to typical fivefold decrease in errors from the truncation; more modes are included in the sum for the same choice of m_c.

IV.   DECREASING THE FOURIER TRUNCATION ERROR

As noted above the full summation gives an energy independent of α. We can thus replace the Ewald energy E_w by

$$E_w \to \left(1 + \gamma_1 \frac{d}{d\alpha} + \gamma_2 \frac{d^2}{d\alpha^2}\right)E_w \qquad (5)$$

without changing the result when the sums are fully converged. However we can choose the γ_i so that truncation errors are reduced when the sums are cut-off. We select values of γ_i so that the Fourier weight in eq. (1) is now equal to

$$\frac{e^{-\pi^2 \mathbf{m}^2/\alpha^2}}{\mathbf{m}^2}\left(1 - \frac{\mathbf{m}^2}{\mathbf{m}_0^2}\right)^2 \qquad (6)$$

This choice is motivated by the fact that the truncation of the Fourier sum is equivalent to introducing a discontinuous weight function in Fourier space. Standard results in asymptotic analysis show that this leads to a long ranged "ringing" in the real-space error[7] with oscillatory power-law decay of the error. By introducing a truncation of the Fourier space interaction at a point where the function and its derivative are both zero we expect that these artefacts can be reduced in such a way that the decay of the truncated function is faster in real-space.

The calculation is best organized by noting that we can bring down a multiplicative factor of $\mathbf{m}^2$ in the Fourier weight by using the operator

$$D_\alpha = \frac{\partial}{\partial(\pi^2/\alpha^2)} = -\frac{\alpha^3}{2\pi^2}\frac{\partial}{\partial\alpha} \qquad (7)$$

The choice eq. (6) then corresponds to the choice for eq. (5)

$$E_w \to (1 + 2\mathbf{m}_0^{-2}D_\alpha + \mathbf{m}_0^{-4}D_\alpha^2)E_w \qquad (8)$$

For the real space correction we make use of the result

$$\frac{\partial}{\partial\alpha}\frac{\text{erfc}(\alpha r)}{r} = -\frac{2}{\sqrt{\pi}}e^{-\alpha^2 r^2} \qquad (9)$$

The corrections in eq. (5) corresponding to eq. (6) are

$$U_{r,cor} = \frac{2\alpha^3}{\pi^{5/2}\mathbf{m}_0^2}\left(1 - \frac{\alpha^2}{4\pi^2\mathbf{m}_0^2}(3 - 2\alpha^2 r^2)\right)e^{-\alpha^2 r^2}$$
$$U_{0,corr} = -\frac{2}{\pi\mathbf{m}_0^2}$$

We see that $U_{r,cor}$ decays exponentially quickly. The exponential function can be evaluated much quicker than erfc so that the overhead of the extra function evaluations is small. We chose $\mathbf{m}_0 = (m_0, 0, 0)$ as the first mode beyond the cubic truncation in the direction $(1, 0, 0)$. Again the Fourier weight eq. (6) is pre-calculated, so that the evaluation of the Fourier expression is just as fast as the un-transformed sum.

The use of this smoothed truncation leads to a two-orders of magnitude decrease in the error in the Fourier sum compared to the use of the simple discontinuous truncation (which is in addition to the factor five discussed above). This improvement in convergence can also be compared to previous work[8,9]; these authors achieved substantial error reduction by performing an expansion using splines or orthogonal polynomials and using the coefficients to optimize the error. We note also interesting recent work on optimum truncation of Fourier transforms

in a more general setting[10]. Our method has the advantage of being expressed as an analytic modification to Ewald, rather than the result of an elaborate optimization of orthogonal functions or splines.

In order to test the implementation of the Ewald sum we wrote two codes, the first based on the expression eq. (1) in a direct implementation. The second used the simplified Fourier evaluation together with a smoothed truncation. We worked with a relatively high quality error bound $\epsilon = 10^{-9}$. After using the estimates eq. (2) as a starting point we tune by hand α to maximize code speed, while keeping errors within the required bound. We found that the best behavior of our optimized sum corresponded to $r_c = 0.73$, $m_c = 6$, with $\alpha = 6.2$. By combining the optimized truncation with the simplified evaluation of interactions in Fourier space our code was accelerated some 15 times compared with the direct implementation as written in eqs. (1-4).

V. OPTIMIZING FOR LARGE NUMBERS OF PARTICLES

For a collection of N particles the Fourier contribution to the energy is generalized by replacing the cosinus in eq. (1) by $|S(\mathbf{q})|^2/2 - N$, where S is the structure factor of the system. In a unit simulation cell N particles are separated on average by a distance $N^{-1/3}$ so that the average density $\rho = N$.

We now consider the complexity of updating the energy when a single charge is displaced. It has been widely assumed that this update requires a complexity that is linear in N. Let us displace a single particle and consider real-space interactions with range $1/\alpha$. We must recalculate the interaction of $\rho/\alpha^3 = N/\alpha^3$ neighbors of the moving particle. In Fourier space we require the structure factor of α^3 modes to evaluate the reciprocal summation. If they are recalculated from scratch each mode requires N operations, leading to a total complexity $\tau = \frac{N}{\alpha^3} + N\alpha^3$ This has a minimum of $\tau = N$ for $\alpha \sim 1$. Indeed Monte Carlo codes based on Ewald summation are normally believed to have a complexity of $O(N^2)$ per sweep through the system.

However if the previous value of the structure factor is still available then we can update it in a rather trivial manner:

$$S(\mathbf{q}) \to S(\mathbf{q}) - q_i e^{i\mathbf{q}\cdot\mathbf{r}_{i0}} + q_i e^{i\mathbf{q}\cdot\mathbf{r}_i} \qquad (10)$$

where $\mathbf{r}_{i0}$ is the old position $\mathbf{r}_i$ is the new and q_i is the value of the charge. The *update* in one mode of the structure factor requires an effort which is $O(1)$, not the $O(N)$ which was required for the original calculation.

This leads to a complexity

$$\tau = \frac{N}{\alpha^3} + \alpha^3 \qquad (11)$$

which has a minimum $\tau \sim N^{1/2}$ for $\alpha \sim N^{1/6}$. For the method to work we require a modestly dimensioned table of size $N^{1/2}$ to stock the evolving structure factor.

We programmed this version of the algorithm. As expected for small values of N it is slower than the version described in Section III, it does not use vectorized multiplication to evaluated the Fourier sum.

There is an obvious fear that following the structure factor using eq. (10) will lead to numerical drift and noise in long simulations. This point is potentially dangerous, as we will now indicate. Fortunately careful management of round-off errors enables one to mitigate this problem. Let us suppose that we have initialized the structure factor to an initial value S_0, a complex number of modulus $O(N^{1/2})$. As we move particles we make *relative* errors in the sum which are given by a machine dependent constant $u \sim 10^{-16}$. Thus the absolute error is $O(u\sqrt{N})$ after a single update. The error can drift randomly so we expect the error after P steps is $u\sqrt{NP}$. We estimate the error in $|S(\mathbf{q})|^2$ from the term linear in u in the expression $(\sqrt{N} + u\sqrt{NP})^2$ and find $\Delta S^2 \sim uN\sqrt{P}$. We find the typical error in the Fourier sum, by adding randomly the error from $\sqrt{N}$ modes. We find

$$(\Delta U)^2 \sim \sqrt{N}(\Delta S^2)^2 = \sqrt{N}(uN\sqrt{P})^2 \qquad (12)$$

In order not to slow the algorithm down by repeated initialization of the structure factor we require that $P \gtrsim N$ We conclude that $\Delta U \sim uN^{7/4}$ after a full sweep of the system. This result seems quite disappointing if we wish to keep a high precision on the total sum, say errors of 10^{-9}; it implies that $N < 10,000$. The solution is to use the Kahan compensated summation algorithm[11] which uses two accumulators to boost the effective machine error *for summation* to u^2. In this case we do not obtain a breakdown in the accuracy of the sum before system sizes which are $N = 10^{14}$. It is clear that when programming for very large system sizes extreme care is need in any calculation involving the accumlation of a large number of independently evaluated contributions. Similar, though slower, results to the Kahan algorithm are found if one accumulates $S(\mathbf{q})$ in quadruple precision.

Finally let us note that the use of multiscale Monte Carlo techniques enable further improvements in efficiency[12] allowing one to use low quality energy estimates interspersed by rarer corrections using a high quality energy evaluation.

VI. NUMERICAL TESTS

We performed some numerical tests to compare the time of a single Monte Carlo sweep in several different implementations Fig. 1. The gold curve corresponds to eq. 1 with pre-calculation of the exponential factor in Fourier space which is then stocked in an array within the program. Optimizing the Fourier calculation using eq. (4) gives the red curve which corresponds to an order of magnitude increase in the code speed. Finally the use
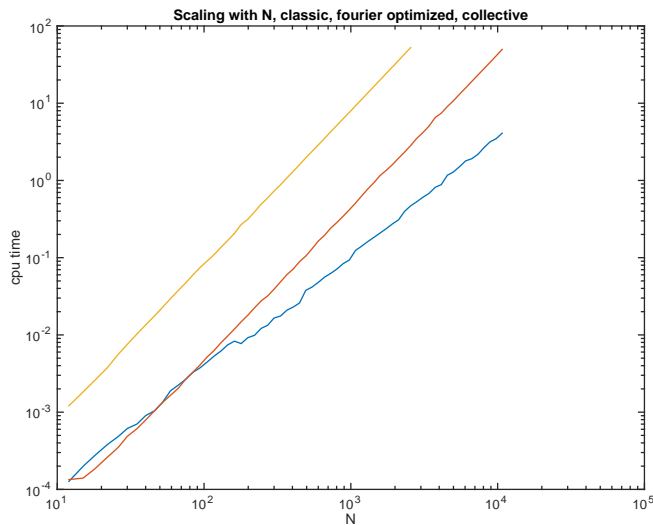
FIG. 1. Computer time per sweep as a function of system size. Gold classical algorithm. Red, simple Fourier optimizations. Blue caching structure factor

of eq. (10) gives the curve in blue. For the largest systems this algorithm is clearly the fastest. The complexity of the scaling is indeed compatible with $N^{3/2}$.

Each algorithm has been tuned to its optimal perfor-

mance by varying α. For the blue curve the algorithm is optimized for $N = 3000$, we then scale α with $N^{1/6}$. For moderate sized systems with N=10,000 the use of the optimal method can lead to a 100 fold speed up in the Ewald sum.

VII. CONCLUSION

We have introduced three simple optimizations of the Ewald method. Our construction of optimized truncations should be useful in all settings and implementations. We have also introduced specific optimizations that are valid for small and large numbers of particles The c^{++} code of these methods is available from the author.

We note that an entirely different approach to the problem is to change the geometry of the space in which one performs the simulation. For instance the interaction between a pair of particles on a hypersphere can be evaluated rather rapidly without the rather cumbersome machinery of Ewald summation[13]. These methods, however, change the physics of the problem being studied.

On a lattice the set of possible separations is know at the outset of the simulation. Thus the Ewald sum can be performed a single time for each separation and stored as a large look-up table,[14].

[1] J. W. Perram S. W. De Leeuw and E. R. Smith, "Simulation of electrostatic systems in periodic boundary conditions. i. lattice sums and dielectric constants," Proc. R. Soc. Lond. A **373**, 27–56 (1980).

[2] Tom Darden, Darrin York, and Lee Pedersen, "Particle mesh ewald: An n [center-dot] log(n) method for ewald sums in large systems," The Journal of Chemical Physics **98**, 10089–10092 (1993).

[3] A. C. Maggs and V. Rossetto, "Local simulation algorithms for coulomb interactions," Phys. Rev. Lett. **88**, 196402 (2002).

[4] Jörg Rottler and A. C. Maggs, "Local molecular dynamics with coulombic interactions," Physical Review Letters **93**, 170201 (2004).

[5] When one or more of (m_1, m_2, m_3) is zero more care is needed in the pre-calculated weights- one divides the weight by 2 for each zero in $\mathbf{m}$.

[6] Axel Arnold, Jason de Joannis, and Christian Holm, "Electrostatics in periodic slab geometries. i," The Journal of Chemical Physics **117**, 2496–2502 (2002).

[7] M.J. Lighthill, *An Introduction to Fourier Analysis and Generalised Functions* (Cambridge University Press, 2008).

[8] Vincent Natoli and David M. Ceperley, "An optimized method for treating long-range potentials," Journal of Computational Physics **117**, 171 – 178 (1995).

[9] G. Rajagopal and R. J. Needs, "An optimized ewald method for long-ranged potentials," Journal of Computational Physics **115**, 399 – 405 (1994).

[10] Eduardo Anglada and José M. Soler, "Filtering a distribution simultaneously in real and fourier space," Physical Review B (Condensed Matter and Materials Physics) **73**, 115122 (2006).

[11] W. Kahan, "Pracniques: further remarks on reducing truncation errors," Commun. ACM **8**, 40 (1965).

[12] Katarzyna Bernacki, Balázs Hetényi, and B. J. Berne, "Multiple "time step" monte carlo simulations: Application to charged systems with ewald summation," The Journal of Chemical Physics **121**, 44–50 (2004).

[13] J. M. Caillol and D. Levesque, "Numerical simulations of homogeneous and inhomogeneous ionic systems: An efficient alternative to the ewald method," The Journal of Chemical Physics **94**, 597–607 (1991).

[14] Athanassios Z. Panagiotopoulos and Sanat K. Kumar, "Large lattice discretization effects on the phase coexistence of ionic fluids," Phys. Rev. Lett. **83**, 2981–2984 (1999).