

# MDSimAid: A Recommender System for Automatic Parameter Selection in Fast Electrostatic Algorithms

Michael S. Crocker      Scott S. Hampton

Jesús A. Izaguirre\*

Department of Computer Science and Engineering

University of Notre Dame

Notre Dame, IN 46556-0309, USA

July 29, 2004

## Abstract

Once the paper is finished, we will add the abstract here.

## 1 Introduction

In Molecular Dynamics (MD) simulations, it is necessary to compute force interactions between each pair of elements in the system. Direct computation of all interacting pairs of  $N$  particles leads to a  $\Theta(N^2)$  algorithm. Because the forces must be updated with each step in the simulation, much effort has been put into reducing the computational cost of calculating these interactions. The Ewald [9] and particle mesh Ewald (PME) [2, 6] methods can be optimized to run in  $O(N^{3/2})$  and  $\Theta(N \log N)$  time, respectively. The fast multipole (FMM) [11] and the multigrid summation (MG) [12–14] methods have improved the asymptotic running time to  $\Theta(N)$ .

---

\*Corresponding author: izaguirr@cse.nd.edu

While these *fast* solvers are asymptotically quicker, in practice there are numerous parameters to optimize. The size of the system, the desired accuracy and the available computational time must all be considered before choosing a method. Unfortunately, each method must be fine tuned to the current system of interest. The choice of parameters cannot be fully accomplished analytically due to the complexity of the methods. Performance is dependent upon the actual simulated system and the implementation of the algorithms.

The purpose of MDSimAid is to automatically choose parameters for the fast electrostatic solvers PME and MG. Users can benefit from these methods without having to do tedious optimizations or delve into complex formulas. Large improvements over published recommendations have been observed. For example, in an 8,000 atom system, the optimized runtime was 2 times faster than the unoptimized simulation, and for an 80,000 atom system, MDSimAid provided a configuration that was 15 times faster.

MDSimAid was tested with an MD object oriented software framework called PROTOMOL [17]. The framework supports the CHARMM 19 and 28a2 force fields and is able to process PDB, PSF, XYZ and DCD trajectory files. It is designed for flexibility, extendibility, and high performance demands, including parallelization. See the PROTOMOL website [19] to obtain source code and prebuilt binaries.

When resolving the interactions of  $N$  particles in a periodically replicated system, the following summation is used to compute the electrostatic potential energy:

$$U^{\text{electrostatic}} = \sum_{i=1}^N \sum_{j \notin \chi(i)} \sum_{\mathbf{m} \in \mathbb{Z}^3} \frac{q_i q_j}{4\pi\epsilon_0 |\mathbf{r}_j - \mathbf{r}_i + \mathbf{m}L|}, \quad (1)$$

where  $\epsilon_0$  is the dielectric coefficient,  $r_i$  and  $q_i$  are the position and charge of particle  $i$  respectively. The set  $\chi(i)$  contains atoms to be excluded from calculation with atom  $i$ , including those that are covalently bonded to  $i$ . The integral triplet  $\mathbf{m}$  is a particular periodic image and  $L$  is the length of a periodic cell with volume  $L^3$ . Eq. (1) gives a conditionally convergent, infinite summation over all periodically replicated cells. Conditional convergence means that the sum may or may not converge depending on the order in which the summation is performed.

While simulation in vacuum can be useful for observing certain properties of a particular molecule, periodic boundary conditions provide a more realistic study of continuous phenomenon. In addition, simulation of bulk material most closely resembles actual physical experiments. For this reason, the main focus here is on periodic

systems.

## 1.1 Automatic recommender systems

In many scientific fields where computational methods are utilized, the optimization dilemma arises. Computational solvers often have various methods and associated parameters. In many cases, finding an optimal or nearly optimal solution must be done on a case-by-case basis. This can be a tedious and time consuming process. Fortunately, programs are being developed that help the user choose parameters automatically.

An application similar in scope to MDSimAid is PBCAID. This software was developed by the Advanced Biomedical Computer Center (ABCC) to help with the simulation setup process. PBCAID is used to optimize the rotational position of a molecular system to reduce the size of the bounding box used in periodic MD simulations. Smaller bounding boxes, provide a substantial savings in computational resources, and their software<sup>1</sup> is designed to do that automatically. In the area of partial differential equations (PDE), there are solvers that require many parameters. The choices made by the user affect the efficiency and applicability of the tests. The PYTHIA-II [7] project has developed software that studies the input and output of PDE solvers in order to automatically recommend configurations that will obtain PDE solutions more quickly. In the field of signal processing, many tasks are performed using discrete signal transforms. These matrix operations are very complex, very large, and can be solved by different methods. The SPIRAL [8] project is designing software that will provide optimal configurations for the chosen methods. When navigating the parameter space to find the best configuration, various search routines are considered ranging from exhaustive and random searches to dynamic programming and sophisticated tree searches.

The SANS project [5] talks about the recommender software as turning a non-expert user into an expert one. By using the intermediate program, a new user can draw from the collected knowledge of the recommender. MD is an interdisciplinary field that has many scientists (chemists, physicists, biologists, etc.) that commonly have little or no notion about how MD simulators work. All the researchers want are the results of the simulation as quickly and as accurately as possible. If they can avoid the technical aspects, then they can be more efficient. Because the recommender program acts in a fast and intelligent way, expert users can perform many tests and

---

<sup>1</sup><http://www.abcc.ncifcrf.gov/app/htdocs/appdb/appinfo.php?appname=PBCAID>

searches in the same time that they would have spent finding one optimal configuration.

## 2 The Search Algorithm

MDSimAid is designed to optimize the PME and MG methods. The optimization works in three stages: First, default parameter values for each method are generated depending on the accuracy desired and the number of atoms in the system. Default values come from rules and tables generated from extensive empirical testing of the methods and analysis of the algorithms available in the literature. Second, a limited run time search, starting from the default parameter values, is performed in the parameter space of the methods. Initially the search is performed with the goal of reducing the error of the simulation through the parameter adjustments. Third, after the desired error tolerance is achieved, MDSimAid makes final configuration changes to reduce the runtime of the simulation while still preserving the accuracy. MDSimAid then presents the user with several different parameter sets containing the best configurations. The user can accept the suggested method or choose among the alternate options which show different error and time constraints.

Throughout the recommendation process, each configuration and the corresponding results from testing are saved. These values are stored, and at any time, the configuration with the best error or runtime can be accessed for use or comparison. Once the recommendation process is finished, the optimal configuration is collected and presented to the user as a recommendation configuration file.

MDSimAid's rules were obtained by testing solvated protein systems from 1,000 to 100,000 atoms. The runtime improvements that MDSimAid makes upon the analytical models depend on the accuracy desired by the user. If a lower accuracy is desired, MDSimAid increases the effort for improving runtime performance. MDSimAid ensures that the accuracy is sufficient and then optimizes the parameters for improved time. Higher accuracies require MDSimAid spend greater amounts of time searching the parameter space. Many times, especially for MG simulations, the analytical equations provide a starting point that is not sufficiently close to the desired accuracy. Because attaining higher accuracy is more difficult than attaining higher performance, MDSimAid will do more to reach the desired accuracy, only afterward improving runtime within the bounds of that accuracy. Improvements in accuracy from the starting points to the recommendations are quite significant.

### 3 Optimization methodology

In this section, we briefly describe each method, followed by an explanation of the rules and models used to optimize them. Both MG and PME are compared against Ewald to determine accuracy.

#### 3.1 Multigrid summation

The MG summation splits the contributions to the potential energy kernel into a short-range, non-smooth part that is evaluated using cutoff, and a smooth part. Switching functions are used that bring the kernel smoothly to zero at a cutoff point. The smooth part is approximated on a grid, by first doing the adjoint interpolation of the particle charges. This process is repeated recursively on the grid, creating a hierarchy of grids, hence the name MG summation.

MG has five parameters: (i) the cutoff,  $r_c$ , used at the particle level; (ii) the grid size,  $h$ , at the coarsest level; (iii) the number of grid levels,  $l$ ; (iv) the smoothness of the switching function,  $k$ , ( $C^1, C^2$ , etc.); and (v) the interpolation order,  $p$ , from one level to the next. The cutoff, or softening distance, defines as many values as there are levels since the cutoff used in each grid will be proportional to this value. Similarly, the grid size at the finest level will be related to the grid size at the coarsest level. The exact proportion is determined by the smoothness of the switching function used to partition the potential energy kernel, and by the order of the interpolation used to transfer charges from particles to grids and forces from grids to particles.

Skeel and coworkers [20] attempt to balance the work between the short-range and the smooth parts. For an approximation of order  $p$  to the smooth part of the force, they find the following optimal values:

$$h = \left(1 + \frac{4}{p}\right)^{1/6} \left(\frac{64}{7}\theta\right)^{1/6} h_*, \quad (2)$$

where  $h_*$  is the distance between two nearest neighbors, and  $\theta$  is the ratio of the cost of calculating a pairwise interaction between grid points to that for particles. Hence, the cutoff can be varied to obtain the desired accuracy.

The computation cost increases proportionally to the cube power of the cutoff, which can be estimated as

$$r_c = \left(h^p h_*^2 \frac{C_p}{\epsilon}\right)^{1/(p+2)}, \quad (3)$$

where  $\epsilon$  is the desired accuracy and  $C_p$  is some unknown constant. A search is still necessary for this constant, but once it is found, the cutoff can be scaled proportionally to the error tolerance  $\epsilon$ .

The separation between grid points could reasonably be anywhere between 1 and 5 Å. Such spacing is suggested by the equations above. Once the optimal  $h$  is determined, the number of grid points  $m$  in the finest grid is obtained based on the physical size of the system. Improvements in accuracy are obtained by increasing the number of grid points (thus decreasing  $h$ ), by increasing the softening distance  $r_c$  (up to size of the bounding box), and through an increase in the number of levels  $l$ . MG's advantage is its ability to interpolate to and from coarse and fine grids to get similar accuracy for less cost. Using additional levels is a major focus of MDSimAid's recommendations for MG.

MDSimAid will start its MG runtime tests by evaluating an analytical model to determine a starting point for system parameters. Once the starting point is determined, it will then do a local search. The configuration from the local search that has the smallest error will be used to start the accuracy optimizations.

First,  $r_c$  is increased to reduce the error, until no additional change is observed. Then,  $m$  will be increased to further reduce the error, if necessary. For MG,  $m$  plays a significant role affecting greatly both error and runtime. Next,  $l$  is increased while keeping the finest grid size the same. Generally, an increase in  $l$  causes the greatest reduction of error when compared to changes in  $m$  or  $r_c$ . Finally, MDSimAid will try to increase  $p$  from. In some cases, the higher order interpolation will reduce the simulation error. The kernel parameter for MG is not affected by MDSimAid but rather coded into configuration files. There is no obvious advantage to any of the kernels.

For the second stage, MDSimAid does the reverse of what it did for accuracy. First, it changes  $l$ , then  $m$ , then  $r_c$ . Time improvement for levels involves increasing  $l$  but also decreasing  $m$ . While making the grid coarser would normally increase the error, increasing the levels makes up for the accuracy while also improving time significantly. Optimal parameters from these tests are used to optimize  $m$  and then  $r_c$  for shorter runtimes, which can lead to very efficient configurations.

### 3.2 Ewald summation

A concrete derivation and analysis of the Ewald summation method can be found in [3, 9]. Ewald splits Eq. (1) into the sum of two rapidly converging series:

$$\frac{1}{r} = \underbrace{\left(\frac{1}{r} - f_{\text{smooth}}(r)\right)}_{\text{Short-ranged}} + \underbrace{f_{\text{smooth}}(r)}_{\text{Long-ranged}}, \quad (4)$$

where  $f_{\text{smooth}}(r)$  is a softening function. The short-ranged equation is of the form:

$$\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N q_i q_j \sum_{\mathbf{m}}' \left( \frac{1}{r_{ij,m}} - \frac{\text{erf}(\alpha r_{ij,m})}{r_{ij,m}} \right), \quad (5)$$

and the long range equation:

$$\frac{1}{2\pi\epsilon_0 L^3} \sum_{\mathbf{m} \neq 0} \frac{1}{\mathbf{m}^2} \exp\left(\frac{-\pi^2 \mathbf{m}^2}{\alpha^2}\right) \left| \sum_j q_j \exp(2\pi i \mathbf{m} \cdot \mathbf{r}_j) \right|^2. \quad (6)$$

The parameter  $\alpha$  is a measure of the convergence rates in real and reciprocal space. With small  $\alpha$ , the short-ranged sum converges faster. Conversely, with large  $\alpha$ , the long-range sum converges more quickly [4, 21]. Fincham [10] and other researchers [18, 23] recommend that  $\alpha$  vary with  $N$  such that

$$\alpha = c\sqrt{\pi} \left(\frac{N}{V^2}\right)^{1/6}, \quad (7)$$

where  $V$  is the volume of the system and  $c$  is the ratio of execution time of the real part to that of the reciprocal. Since  $c$  may vary from one platform to another, this ratio is computed by PROTOMOL at run time.

In addition to  $\alpha$ , the direct space cutoff distance  $r_c$ , controls accuracy and CPU time in Ewald. The relationship between  $\alpha$  and  $r_c$  is given by

$$r_c = \frac{\sqrt{-\ln \epsilon}}{\alpha}, \quad (8)$$

where  $\epsilon$  is the desired accuracy. A typical value for  $\epsilon$  is  $10^{-5}$ . Large  $\alpha$  yields a small cutoff distance and faster execution time. Since Ewald is easily optimized and highly accurate, MDSimAid uses values computed by Ewald

to determine the error in PME and MG.

### 3.3 Particle Mesh Ewald

PME has three method parameters: the first,  $\alpha$ , determines the amount of work that is done in real and reciprocal space; the second is the cutoff  $r_c$  at which the real part is evaluated; the third is the number of points  $m$  in the grid used by the FFT. As the last two parameters get bigger, the accuracy and the cost increase as well. Occasionally the method becomes sensitive to variations, but for the most part it is fairly consistent and predictable.

The  $\alpha$  and  $r_c$  parameters have the same qualitative effect in PME as in Ewald. Petersen [18] shows that the optimal  $\alpha$  is approximately  $N^{1/3}$ , whereas the optimal for  $r_c$  is proportional to  $\log_2^{1/2} N$ . Therefore,  $\alpha$  is quite sensitive to  $N$  while  $r_c$  is hardly dependent on  $N$ . In particular, once  $\alpha$  is determined (based on  $N$ ), the  $r_c$  can be estimated as follows:

$$\frac{\text{erfc}(\alpha r_c)}{r_c} = \epsilon, \tag{9}$$

where  $\epsilon$  is the desired accuracy.

The effect of  $m$  on the error and runtime of PME is modest. Assume a coarse grid to have a separation twice that of a medium grid and four times that of a fine grid. A fine grid will incur about 20% more overhead than a medium grid size, but it has an error lower than that of the coarse grid. Therefore, if high accuracy is important, a fine grid should be used. But if faster evaluation is needed, then a coarse grid should be used.

In this work we use B-spline interpolation for PME, since it is smoother and it is easier to control its accuracy, cf. []. By default, cubic B-splines are used, but for higher accuracy quintic interpolation is available. Also available, and quite significant, is the accuracy parameter  $X$ . With a better accuracy parameter, large improvements in accuracy will be possible with only medium increases in runtime. This will be the case up until a relative error of about  $10^{-6}$  in the forces.

The initial parameters are calculated from the analytical models, while, at the same time, an appropriate  $m$  based on the system size is determined. After that, a local search is performed. The configuration with the smallest error from that search is passed on. Next, like with MG,  $r_c$  is reduced while the error from the test runs is closely observed. MDSimAid then checks to see if the desired accuracy has been reached. If not, it makes a change to the accuracy parameter to a value ten times smaller, hoping to reach the desired error tolerance. The

fastest configuration found up to that point that is also within the desired accuracy is chosen for the next stage. Finally, parameters are reworked so that each step improves the runtime. This is done until the simulation error becomes larger than the desired accuracy. In most cases, this involves reducing  $r_c$  distance little by little, because  $m$  is not very sensitive and rarely changed after initial determination. From all of the test runs, the optimal can then be selected.

## 4 Results and discussion

### 4.1 Ways of Evaluating Accuracy

PROTOMOL computes the relative error in force and potential energy by using a specified method as a reference. In the case of MDSimAid, the reference method is Ewald. Errors are computed by PROTOMOL comparing Ewald to MG or Ewald to PME. The metrics for relative errors are defined as follows:

$$\text{rFavg} = \frac{\sum_i \sqrt{\frac{\|F_i - \tilde{F}_i\|^2}{m_i}}}{\sum_i \sqrt{\frac{\|F_i\|^2}{m_i}}}, \quad (10)$$

$$\text{rFmax} = N \frac{\max_i \sqrt{\frac{\|F_i - \tilde{F}_i\|^2}{m_i}}}{\sum_i \sqrt{\frac{\|F_i\|^2}{m_i}}}, \quad (11)$$

$$\text{rPE} = \left| \frac{E_p - \tilde{E}_p}{E_p} \right|, \quad (12)$$

where  $m_i$  is the mass of atom  $i$ ;  $F$  is the force of the reference algorithm reference and  $\tilde{F}$  is the force of the algorithm being tested.

MDSimAid optimizes MG and PME for lower  $rFavg$ . Changes in  $rFavg$  are important for all of MDSimAid’s algorithmic choices. The error in the potential energy  $rPE$  produces smaller errors on average, but the behavior of  $rPE$  is less consistent than  $rFavg$  for different configurations. In addition, a metric based on the forces is more useful than one based only on the energies.  $rFmax$  is useful for examining extremes in the error, but not as a comparator among different methods.

MDSimAid uses three different values for analyzing the effectiveness of each configuration, or set of parameters.

The first is runtime  $t$ , which is simply the amount of time used by PROTOMOL to complete one simulation step. The second, average error  $rFavg$ , comes from PROTOMOL’s built-in force comparator. Finally, the cost metric,  $M_e$ , is defined as

$$M_e = \frac{t}{|\log(rFavg)|}, \quad (13)$$

and is a simple way to combine runtime and error in a single value.  $M_e$  can be viewed as the time it takes to achieve a tenfold improvement in the error. A lower  $M_e$  means either that the runtime cost to attain the desired accuracy was lower, or that a smaller error was attained with virtually the same runtime cost.

## 4.2 Parameter Behavior

For MG, the behavior of the grid-size  $m$  and softening distance  $r_c$  is simple to understand and predict. The curve for increasing  $r_c$  is almost parabolic, and it is easy to find the point at which growth will no longer produce any reduction in  $rFavg$ . Because increasing  $r_c$  will slow down the simulation, there is no reason to try any larger values once that floor is reached. See Figs. 1 and 2.

In Fig. 1, increases in  $r_c$  level off and no longer provide improvement in  $rFavg$ . Eventually, increasing  $r_c$  will cause the error to amplify as well. Generally, the growth in  $rFavg$  is observed starting close to the size of the bounding box of the molecular system. For example, the 1,029 atom water system has a bounding box of approximately  $19 \text{ \AA}^3$ , and  $rFavg$  begins to increase with an  $r_c$  of  $17 \text{ \AA}$ . The same is true for the 2,001 atom water systems with a bounding box approximately  $25 \text{ \AA}^3$  where  $rFavg$  starts to increase with  $r_c$  equal to  $22 \text{ \AA}$ .

Figs. 3 and 4 illustrate the effect of  $m$  on accuracy and runtime. Increasing  $m$  leads to decreases in  $rFavg$  for small values. At a certain point, however, the trend reverses.

Increasing levels  $l$  is another way to improve error. With no changes to  $m$  or  $r_c$ , increasing  $l$  has a somewhat predictable effect on  $rFavg$ . For each different sized systems, increasing  $l$  will improve  $rFavg$ , but the amount of improvement is different and unpredictable. In less common situations, increasing  $l$  does not improve  $rFavg$ , however it is still important to continue trying more levels because. Because it is unpredictable but generally beneficial, MDSimAid tries more levels for many different configurations to reduce the error in simulation. See Fig. 5.

When increasing only the number of levels, the resulting runtime can also be affected. The behavior of  $t$  for

---

**Algorithm 1** MDSimAid

---

1. For each method, compute the initial configuration. See Section 3 for the source of these equations.
  2. Perform optimizations on each parameter in the designated order of effectiveness to decrease the error in the simulation.
    - (a) The cutoff or softening distance (PME and MG)
    - (b) The grid-size (MG)
    - (c) The number of levels (MG)
    - (d) The accuracy parameter (PME)
    - (e) The order parameter (MG)
  3. Perform optimizations on each parameter in the designated order of effectiveness to decrease the runtime of the simulation.
    - (a) The number of levels (MG)
    - (b) The grid-size (MG)
    - (c) The cutoff (PME and MG)
  4. Repeat steps 2–3 to determine if further improvements are possible.
  5. Choose method that meets user constraints.
  6. Give user suggested setup with several alternate options.
- 

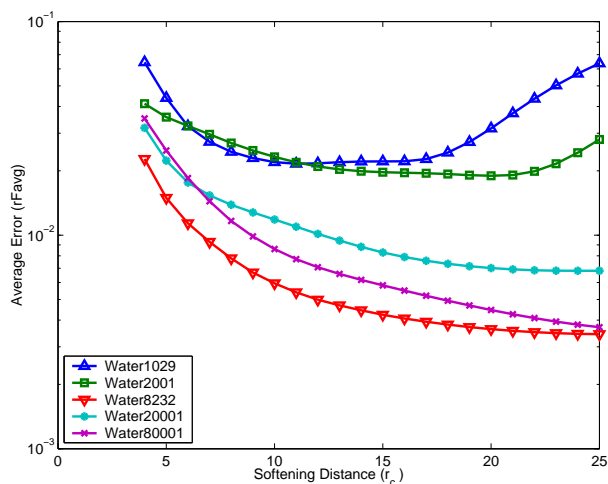


Figure 1: The effect of  $r_c$  on the average error  $rF_{avg}$  for different sized TIP3P water boxes.

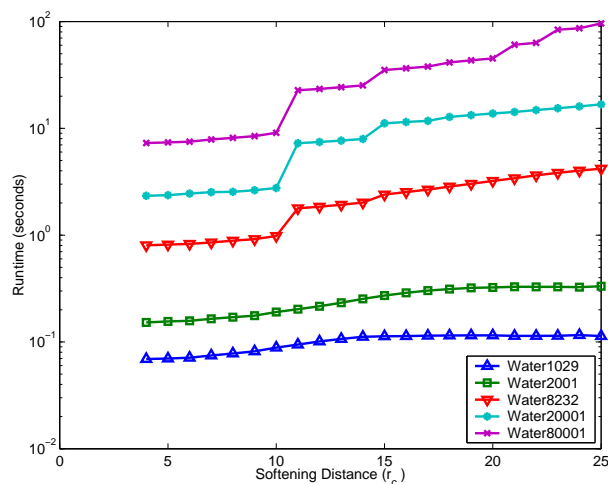


Figure 2: The effect of  $r_c$  on the running time for different sized TIP3P water boxes.

this parameter is more unpredictable than for  $rFavg$ . In some cases increased levels will decrease  $t$  and other times an increase will occur. See Fig. 6.

### 4.3 Runtime Improvements

MDSimAid’s biggest advantage comes in the form of runtime tests and optimizations. There are many studies that give analytical methods for determining optimal parameters for the various fast force evaluators. These equations and tables could be used with any molecular system to calculate a configuration. However, the effect that different configurations have on simulation error is much less predictable than the analytical formulas would suggest. To be certain that the final recommendation is near optimal, actual test runs of the MD simulator are performed to confirm and carefully improve results.

The runtime tests are most important for MG, as it has more parameters in its configuration, and the combination of those parameters leads to  $rFavg$  having a more chaotic behavior. MDSimAid’s recommendations are compared to the Skeel equations [20] in order to demonstrate the advantage of runtime improvements. The tests were run with PROTOMOL using MG on molecular systems of sizes 8,000 to 80,000 atoms. Results are presented showing both  $rFavg$ . The second plot is more informative, and the metric value  $M_e$  that considers  $t$  as well as  $rFavg$ . See Fig. 7. A second graph is given for a higher requested accuracy. See Fig. 8.

The analytical models produce configurations with smaller  $t$  than did the MDSimAid recommendation for some of the systems for smaller error tolerances. Such results are reasonable because they were accompanied by higher  $rFavg$  values. MDSimAid was able to find configurations that produced much smaller errors, especially in the  $10^{-3}$  range. Considering that  $rFavg$  for the recommender was 4 to 50 times better than that of the equations, the few times that the analytical models provided slightly faster results means little when the overall efficiency is poor. The  $M_e$  plot seen here better demonstrates the results of the recommender versus the equations. As the system size increases, the recommender becomes more and more efficient at achieving an acceptable error and also keeping  $t$  relatively small.

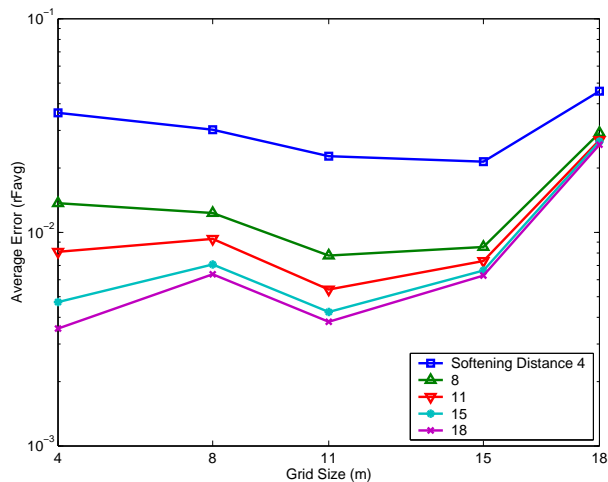


Figure 3: The effect of grid size  $m$  on average error  $rFavg$  for TIP3P water with 8232 atoms.

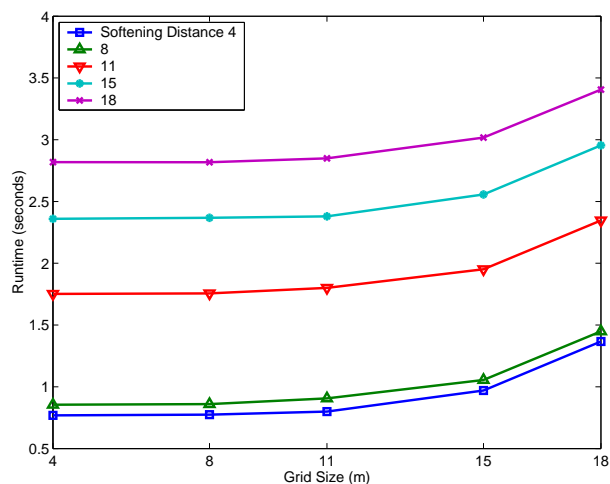


Figure 4: The effect of grid size  $m$  on runtime for TIP3P water with 8232 atoms.

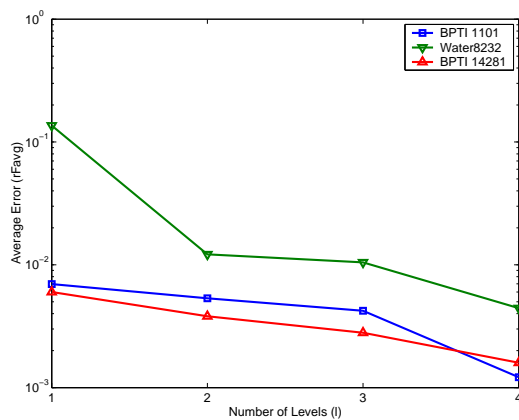


Figure 5: How Levels Affect Average Error (Finest Level Grid = 32)

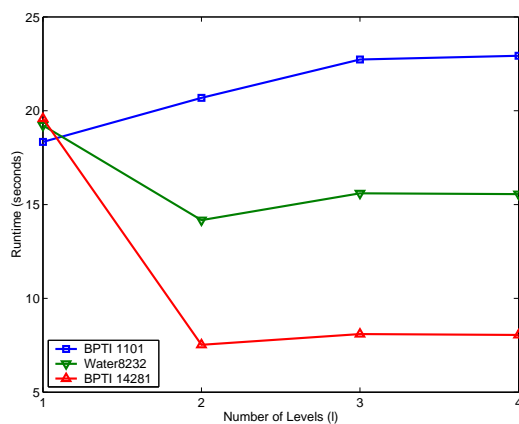


Figure 6: How Levels Affect Runtime (Finest Level Grid = 32)

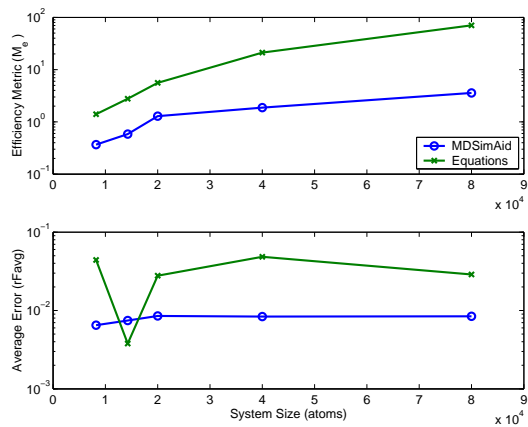


Figure 7: Analytical Methods vs MDSimAid Recommendation ( $10^{-2}$ ).

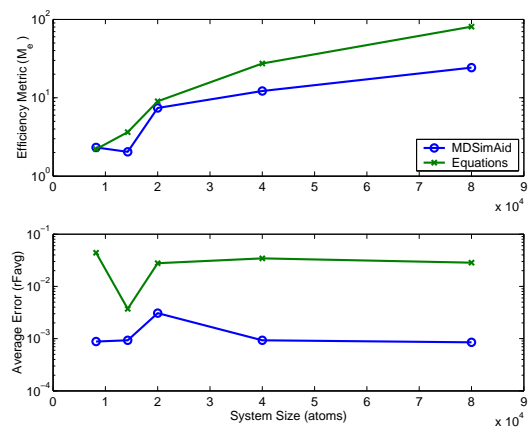


Figure 8: Analytical Methods vs MDSimAid Recommendation ( $10^{-3}$ ).

## 4.4 Comparison of different methods

The final MDSimAid tests compared the recommendations between different force evaluators. PME was compared to MG for different system sizes and different error tolerances. The purpose was to see which force evaluator MDSimAid could find smaller errors for. Results were also compared using runtime and also efficiency. For small error tolerances, namely  $10^{-2}$  and  $10^{-3}$ , MG did better than PME for all tests except for the one. In the case of the the 20,000 atom water system at a  $10^{-3}$  requested tolerance, MDSimAid failed to find a competitive  $t$  or  $rFavg$ , and thus a poor  $M_e$ . See Figs. 9 and 10.

For smaller error tolerances, MG could not compete with PME. In many cases, MDSimAid could not configure MG to achieve an error of  $10^{-4}$  for any configuration. In failing to produce a sufficient accuracy, MDSimAid then recommended the configuration with the smallest error, even if accompanied by very large runtimes. PME, on the other hand, has an accuracy parameter that can be adjusted for different desired values. A ten time improvement in error does not cause nearly as much additional overhead for PME as is necessary for MG. For PME, the limit of MDSimAid's ability to provide accurate results is about  $10^{-6}$ .

## 5 Future Work

MDSimAid could be improved to do many things. From its current algorithm, it could easily be expanded to recommend for different force evaluation methods and for different MD simulators. Attempts were made to complete a version of MDSimAid that optimizes for multiple time stepping (MTS) methods [1]. Due to certain complications with the method itself, the attempt was abandoned, but once the method is fixed, it would be easy to do. There are also different MD simulators out there. NAMD [15,16] is a good example of this. NAMD works with PME and Ewald, and also has some MTS functionality. With a little bit of research into the formatting of configuration and output files, MDSimAid could work with NAMD.

Another idea that came up during the development of MDSimAid involves making the online version more adaptable to the various tests that it runs. It could become self-adapting. The results of each test run on MDSimAid could be collected into a database, and the patterns found in the results could be used to more quickly recommend for future runs. This is a more complicated idea, but is possible. In fact, there are already

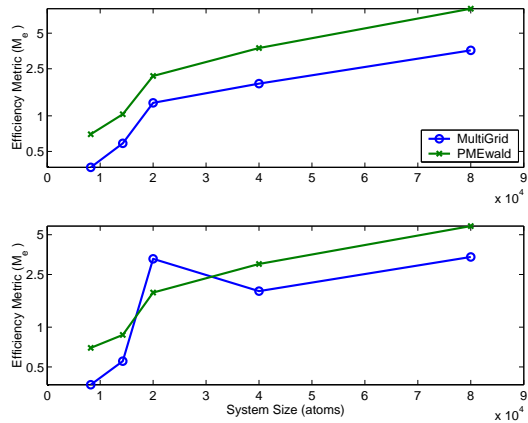


Figure 9: PME vs MG ( $10^{-2}$ ,  $10^{-3}$ ).

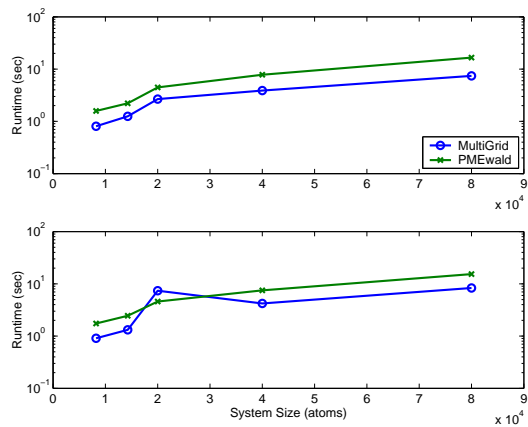


Figure 10: PME vs MG ( $10^{-2}$ ,  $10^{-3}$ ).

groups looking into the possibility of self adaptivity in various other areas of study, particularly in the area of grid computing [22].

## 5.1 Acknowledgments

This work was partially supported by an NSF Career Award ACI-*xxxxxxx*. Computations were performed in part through a Beowulf cluster supported by NSF grant DMR-0079647 and also by an Equipment Renovation grant through the University of Notre Dame. Michael Crocker was supported by an ICSB Undergraduate Research scholarship and NSF REU ACI-*xxxxxxx*. Scott Hampton was supported through an Arthur J. Schmitt fellowship.

## References

- [1] J. J. Biesiadecki and R. D. Skeel. Dangers of multiple-time-step methods. *J. Comput. Phys.*, 109(2):318–328, 1993.
- [2] T. Darden, D. York, and L. Pedersen. Particle mesh ewald: An  $w \log(n)$  method for Ewald sums in large systems. *J. Chem. Phys.*, 98(12):10089–10092, 1993.
- [3] S. W. de Leeuw, J. W. Perram, 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.
- [4] M. Deserno and C. Holm. How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines. *J. Chem. Phys.*, 109(18):7678–7693, 1998.
- [5] J. Dongarra and V. Eijkhout. Self-adapting numerical software for next generation applications. *International Journal of High Performance Computing Applications*, 17:125–131, 2003.
- [6] U. Essmann, L. Perera, and M. L. Berkowitz. A smooth particle mesh Ewald method. *J. Chem. Phys.*, 103(19):8577–8593, 1995.
- [7] E. N. H. et al. Pythia-ii: A knowledge/database system for managing performance data and recommending scientific software. pages 227–253, 2000.

- [8] M. P. et al. Spiral: A generator for platform-adapted libraries of signal processing algorithms. *Journal of High Performance Computing and Applications*, pages 21–45, 2004.
- [9] P. Ewald. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Ann. Phys.*, 64:253–287, 1921.
- [10] D. Fincham. Optimisation of the Ewald sum for large systems. *Mol. Sim.*, 13:1–9, 1994.
- [11] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73:325–348, 1987.
- [12] M. Holst. *Multigrid Solution of the Poisson–Boltzmann Equation*. PhD thesis, Dept. of Computer Science, Univ. of Illinois at Urbana–Champaign, 1993.
- [13] M. Holst, R. E. Kozack, F. Saied, and S. Subramaniam. Protein electrostatics: Rapid multigrid-based Newton algorithm for solution of the full nonlinear Poisson–Boltzmann equation. *J. Biomol. Struct. Dyn.*, 11(6):1437–1445, 1994.
- [14] J. A. Izaguirre and T. Matthey. <http://sourceforge.net/projects/protomol>, 2001. User Guide 1.05: Protomol: An Object-Oriented Molecular Dynamics Framework.
- [15] L. Kalé, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadara-jan, and K. Schulten. NAMD2: Greater scalability for parallel molecular dynamics. *J. Comput. Phys.*, 1999. In press.
- [16] L. V. Kalé, M. Bhandarkar, R. Brunner, N. Krawetz, J. Phillips, and A. Shinozaki. NAMD: A case study in multilingual parallel programming. In *Proc. 10th International Workshop on Languages and Compilers for Parallel Computing*, pages 367–381, Minneapolis, Minnesota, Aug. 1997.
- [17] T. Matthey, T. Cickovski, S. Hampton, A. Ko, Q. Ma, M. Nyerges, T. Raeder, T. Slabach, and J. A. Izaguirre. PROTOMOL: An object-oriented framework for prototyping novel algorithms for molecular dynamics. *ACM Trans. Math. Softw.*, 2004. In press.
- [18] H. G. Petersen. Accuracy and efficiency of the particle mesh Ewald method. *J. Chem. Phys.*, 103(3668-3679), 1995.

- [19] PROTOMOL. PROTOMOL: An object oriented framework for molecular dynamics. <http://sourceforge.net/projects/protomol/>, July 2004.
- [20] R. D. Skeel, I. Tezcan, and D. J. Hardy. Multiple grid methods for classical molecular dynamics. *J. Comp. Chem.*, 23(6):673–684, 2002.
- [21] A. Y. Toukmaji and J. A. Board. Ewald summation techniques in perspective: A survey. *Comput. Phys. Commun.*, 95:73–92, 1996.
- [22] S. S. Vadhiyar and J. J. Dongarra. Self adaptivity in grid computing. *Concurrency: Practice and Experience*, 2003.
- [23] Z. Wang and C. Holm. Estimate of the cutoff errors in the Ewald summation for dipolar systems. *J. Chem. Phys.*, 115(14):6351–6359, 2001.