SIAM J. SCI. COMPUT.
Vol. 34, No. 1, pp. C1–C23

REACTIVE MOLECULAR DYNAMICS: NUMERICAL METHODS AND ALGORITHMIC TECHNIQUES*

HASAN METIN AKTULGA†, SAGAR A. PANDIT‡, ADRI C. T. VAN DUIN§, AND ANANTH Y. GRAMA†

Abstract. Modeling atomic and molecular systems requires computation-intensive quantum mechanical methods such as, but not limited to, density functional theory [R. A. Friesner, *Proc. Natl. Acad. Sci. USA*, 102 (2005), pp. 6648–6653]. These methods have been successful in predicting various properties of chemical systems at atomistic scales. Due to the inherent nonlocality of quantum mechanics, the scalability of these methods ranges from $O(N^3)$ to $O(N^7)$ depending on the method used and approximations involved. This significantly limits the size of simulated systems to a few thousand atoms, even on large scale parallel platforms. On the other hand, classical approximations of quantum systems, although computationally (relatively) easy to implement, yield simpler models that lack essential chemical properties such as reactivity and charge transfer. The recent work of van Duin et al. [*J. Phys. Chem. A*, 105 (2001), pp. 9396–9409] overcomes the limitations of nonreactive classical molecular dynamics (MD) approximations by carefully incorporating limited nonlocality (to mimic quantum behavior) through an empirical bond order potential. This reactive classical MD method, called ReaxFF, achieves essential quantum properties, while retaining the computational simplicity of classical MD, to a large extent. Implementation of reactive force fields presents significant algorithmic challenges. Since these methods model bond breaking and formation, efficient implementations must rely on complex dynamic data structures. Charge transfer in these methods is accomplished by minimizing electrostatic energy through charge equilibration. This requires the solution of large linear systems (10^8 degrees of freedom and beyond) with shielded electrostatic kernels at each time-step. Individual time-steps are themselves typically in the range of tenths of femtoseconds, requiring optimizations within and across time-steps to scale simulations to nanoseconds and beyond, where interesting phenomena may be observed. In this paper, we present implementation details of sPuReMD (serial Purdue reactive molecular dynamics program), a unique reactive classical MD code. We describe various data structures, and the charge equilibration solver at the core of the simulation engine. This Krylov subspace solver relies on a preconditioner based on incomplete LU factorization with thresholds (ILUT), specially targeted to our application. We comprehensively validate the performance and accuracy of sPuReMD on a variety of hydrocarbon systems. In particular, we show excellent per-time-step time, linear time scaling in system size, and a low memory footprint. sPuReMD is a freely distributed software with GPL and is currently being used to model diverse systems ranging from oxidative stress in biomembranes to strain relaxation in Si-Ge nanorods.

Key words. reactive classical molecular dynamics, bond order potential, ReaxFF, charge equilibration, Krylov subspace solvers, ILUT-based preconditioner

AMS subject classifications. 70F10, 65P10, 65F50, 65F10, 68Nxx, 65D07

DOI. 10.1137/100808599

1. Introduction.

Molecular-scale simulation techniques provide important computational tools in diverse domains, ranging from biophysical systems (protein folding, membrane modeling, etc.) to materials engineering (design of novel materials,

*Submitted to the journal's Software and High-Performance Computing section September 14, 2010; accepted for publication (in revised form) August 30, 2011; published electronically January 31, 2012. The research of the first, second, and fourth authors was supported by the National Science Foundation and the U.S. Department of Energy.

http://www.siam.org/journals/sisc/34-1/80859.html

†Department of Computer Science, Purdue University, West Lafayette, IN 47907 (haktulga@cs.purdue.edu, ayg@cs.purdue.edu).

‡Department of Physics, University of South Florida, Tampa, FL 33620 (pandit@usf.edu).

§Department of Mechanical and Nuclear Engineering, Pennsylvania State University, University Park, PA 16802 (acv13@psu.edu). This author's research was partly supported by KISK startup grant C000032472.

nanoscale devices, etc.). These methods can model physical reality under extreme conditions not easily reproducible in the laboratory. Conventional molecular simulation methods range from quantum-scale to atomistic methods. Quantum-scale methods are based on the principles of quantum mechanics, i.e., on the explicit solution of the Schrödinger equation. They start with first principles quantum mechanics and make few approximations while deriving the solution. For these reasons, quantum molecular dynamics (MD) methods are often referred as *ab-initio* methods. Due to their modeling fidelity, ab-initio methods usually produce more accurate and reliable results than atomistic methods.

Due to the nonlocality of interactions in quantum mechanics, the accuracy of ab-initio methods comes at a high computational cost. Brute force approaches to ab-initio calculations suffer from the exponential growth of computational complexity with the number of electrons in the simulated system. Several approaches, with varying degrees of complexity and accuracy, have been developed to solve the ab-initio problems (please refer to [1] for an excellent review). These approaches are broadly classified into two categories: (i) wavefunction-based approaches (which include Hartree–Fock (HF), second-order Moller–Plesset perturbation theory (MP2), coupled cluster with single, double, and triple perturbative excitations (CCSD(T)), complete active space with second-order perturbation theory (CASPT2) as the more popular ones), and (ii) density functional theory (DFT) based approaches. Even with the approximations to reduce computational costs, ab-initio methods do not scale well with the system size. N being the number of electrons, CCSD(T) scales as N^7, MP2 as N^5, and localized variants of MP2 can bring the scaling factor down to N^3, making thousand-atom simulations feasible. DFT-based methods such as Car–Parrinello molecular dynamics (CPMD), CASTEP, and the Vienna Ab-initio Simulation Package (VASP) are still methods of choice for medium to large scale systems, since they can deliver better accuracy and performance in most cases [1].

High computational costs associated with ab-initio methods restrict their applicability to the systems on the order of thousands of atoms and a few picoseconds of simulation time. For many real-life problems, however, systems of this length and time scale are rarely sufficient to observe the phenomena of interest. Techniques such as atomistic simulations based on empirical force fields are developed to overcome this system size limitation. Empirical force fields model the nuclear core together with its orbital electrons as a single basis, thus making the problem "local." Since electrons are not treated explicitly, their roles and effects are approximated by means of functional forms and parameters. Often, empirical force fields are dependent on a large number of tunable parameters. Evaluation of these parameters is a critical step in the modeling process. Due to the correlations between virtually all the parameters, development of a high-quality force field is usually a very tedious task. Typically, all parameters are tuned iteratively to match well-studied experimental properties of the target subsystem of the real system. Once this task is accomplished, resulting force field parameters can be used in simulations of systems that reflect real-life scenarios.

MD methods produce snapshots of the time evolution of the input system using Newton's laws of motion. While it may be argued that an MD simulation would deviate significantly from the target system's real trajectory due to significant simplifications and approximations, along with numerical errors associated with implementations, MD methods find their basis in the fundamental postulate of classical statistical mechanics: "All states accessible to the system and having a prescribed energy, volume and number of particles are equally likely to be visited in the course of time (the ergodic hypothesis)" [3]. Consequently, even though we cannot hope

to capture real trajectories from MD simulations, we can still compute ensembles of snapshots for physically accessible configurations of systems. This allows us to use ideas from statistical thermodynamics to study time-averaged properties such as density, temperature, and free energies.

While traditional atomistic modeling techniques are successful in reproducing features of real systems to varying degrees, they are limited in many respects. Some of these limitations are as follows: (i) Due to specific parameterization, these methods are not generic and cannot be used for arbitrary systems. (ii) While this is true for any empirical force field, conventional atomistic methods start with the assumption of static bonds in their target system, and therefore they cannot be used to model reactive systems. (iii) In most atomistic methods, charges are kept fixed throughout the simulation. Although polarizable force fields were introduced almost two decades ago [4], they have only recently gained significant attention for modeling charge transfer in empirical force fields [5]. Polarization is achieved either by inducible point dipole methods or by fluctuating charge models. Even though polarizable force fields are built upon their nonpolarizable counterparts, their development still requires considerable effort since charges are not assumed to be fixed in the target system. This requires most parameters to be retuned. Better characterization of target systems have been reported in literature through the use of polarizable force fields [6].

With a priori knowledge of the reactions in a system of interest and spatially localized reactivity, quantum mechanics/molecular mechanics (QM/MM) methods can be used to study large scale reactive systems. Mixed QM/MM methods use QM methods to simulate the reactive region while applying MM methods to the rest of the system. The reactive region must be localized, otherwise computational cost of QM methods dominates the overall simulation cost. A natural question relates to effective and efficient ways of coupling QM and MM methods, which provides a bridge between these disparate modeling regimes. If there are no covalent bonds between the QM and MM regions, then coupling is easily achieved by introducing long range interactions between the two regions. However, more complicated models are necessary when there are covalent bonds. While impressive performance results have been achieved using QM/MM methods on large scale reactive systems, factors such as knowledge of reactive site a priori, size limitation on the reactive site, difficulties in coupling QM/MM regions, and the intricacies of setting up and running such simulations have prevented mixed QM/MM methods from being widely adopted to the study of large scale reactive systems [1].

To bridge the gap between quantum methods and classical MD methods, a number of models with empirical bond order potentials have been proposed. These techniques mimic quantum overlap of electronic wavefunctions through a bond order term that describes the bonds in the system dynamically based on the local neighborhoods of each atom. A widely used bond order potential has been the reactive empirical bond order (REBO) potential [7]. REBO is built on the Tersoff potential [8], which was inspired by Abell's work [9]. REBO was extended to describe interactions with Si, F, and Pt. Subsequently, Brenner et al. developed a newer formulation of REBO aimed at overcoming the shortcomings of the initial version [10]. Like many other bond order potentials, this new version of REBO lacks long range interactions, which are crucial in modeling molecular systems. AIREBO was an attempt by Stuart, Tutein, and Harrison to generalize REBO to include long range interactions. However, it retained the fundamental problems in the shapes of the dissociation and reactive potential curves of REBO [11].

The ReaxFF method of van Duin et al. [2] is the first classical reactive force field that contains dynamic bonds and polarization effects in its formulation. The flexibility and transferability of the force field allows ReaxFF to be easily extended to many systems of interest in diverse domains. In terms of accuracy, in a detailed comparison of ReaxFF against REBO and the semiempirical MOPAC method with PM3 parameters by van Duin et al. [2], it has been reported that results from ReaxFF on hydrocarbons are in much better agreement with DFT simulations than those of REBO and PM3.

Reactive force fields involve classical interactions that introduce limited nonlocality through the bond order terms. In spite of these simplifications, challenges associated with implementing an efficient and scalable reactive force field are formidable. In a typical classical MD simulation, bonds, valence angles, dihedral angles, and atomic charges are input to the program at the beginning and remain unchanged throughout the simulation. This allows for simple data structures and memory management schemes, in terms of static (interaction) lists. However, in a reactive force field where bonds are formed or broken, and where all three-body and four-body structures need to be updated at every time-step, efficient memory management becomes an important issue. When bonds incident on an atom are changing, it is evident that charge on that atom is going to change as well. Charge update needs to be done frequently and accurately because electrostatic interactions play crucial roles in describing most systems, and over-estimation or under-estimation of charges would, for example, result in violation of energy conservation principle for the microcanonical ensemble.

In this paper, we present algorithmic and numerical techniques underlying our implementation of ReaxFF, called sPuReMD (serial Purdue reactive molecular dynamics program). In section 2 we briefly describe the interactions in ReaxFF. Section 3 deals with the various algorithmic aspects of reactive modeling. In ReaxFF, electrostatic interactions are modeled as shielded interactions with Taper corrections. This obviates the need for computing long range electrostatic interactions. The simple pairwise nature of nonbonded interactions allows the use of interpolation schemes to expedite the computation of energy and forces due to nonbonded interactions. The complexity of bonded interactions on one hand and the possibility of expediting the nonbonded interactions on the other hand bring the computational time required for bonded interactions on par with that of nonbonded interactions. Note that in typical classical MD methods, time required for handling bonded interactions is negligible compared to the time required for handling nonbonded interactions. As mentioned above, a highly accurate charge update mechanism is desirable to obtain reliable results from ReaxFF simulations. However, the QEq method [12] that we use for determining partial charges on atoms at every time-step requires the solution of a very large sparse linear system, which can take up a significant fraction of the total compute time. Consequently, we develop a novel solver for the QEq problem using the generalized minimal residual method (GMRES) [13] and an incomplete LU factorization with thresholds (ILUT) based preconditioner [14]. This solver relies heavily on optimizations across iterations to achieve an excellent per time-step running time. The dynamic nature of bonds in ReaxFF requires dynamic bond lists, and subsequently dynamic angle and dihedral lists reconstructed at every time-step. In section 4 we describe our memory management and reallocation mechanisms, which form critical components of sPuReMD. We present detailed validation of performance and accuracy on hydrocarbon systems by comparing simulation results with the literature in section 5. Characterization of efficiency and performance of our implementation on

systems of different types is presented in section 5.2. Through these extensive simulations, we establish sPuReMD as a high-performance, scalable (in terms of system sizes), accurate, and lean (in terms of memory) software system. sPuReMD is currently in limited release and is being used at ten large institutions for modeling diverse reactive systems.

2. Reactive potentials for atomistic simulations. In nonreactive classical molecular models, atoms constitute molecules through static bonds, akin to a ball-and-spring model in which springs are statically attached. This approach cannot simulate chemical reactions, since reactions correspond to bond breaking and formation. In the Reax force field (ReaxFF) model, each atom is treated as a separate entity, whose bond structure is updated at every time-step. This dynamic bonding scheme, together with charge redistribution (equilibration to minimize electrostatic energy) constitutes the core of ReaxFF. We briefly explain them before we discuss algorithmic aspects of reactive modeling in the next section.

2.1. Bond orders. Bond order between a pair of atoms, i and j, is the strength of the bond between the two atoms. In ReaxFF, bond order is modeled by a closed form (equation (2.1)), which computes the bond order in terms of the types of atoms i and j and the distance between them:

$$BO_{ij}^{\alpha'}(r_{ij}) = exp\left[a_\alpha \left(\frac{r_{ij}}{r_{0\alpha}}\right)^{b_\alpha}\right]. \tag{2.1}$$

In (2.1), α corresponds to $\sigma-\sigma$, $\sigma-\pi$, or $\pi-\pi$ bonds, a_α and b_α are parameters specific to the bond type, and $r_{0\alpha}$ is the optimal length for this bond type. The total bond order (BO'_{ij}) is computed as the summation of $\sigma-\sigma$, $\sigma-\pi$, and $\pi-\pi$ bonds as follows:

$$BO'_{ij} = BO_{ij}^{\sigma'} + BO_{ij}^{\pi'} + BO_{ij}^{\pi\pi'}. \tag{2.2}$$

One cannot model the complex bond structure observed in real-life systems just by using pairwise bond order potentials; we must account for the total coordination number of each atom and 1–3 bond corrections in valence angles. For instance, the bond length and strength between O and H atoms in a hydroxyl group (OH) are different than those in a water molecule (H_2O). Alternately, taking the example of H atoms in a water molecule, if the two H atoms are detached from the middle O atom and put in vacuum while preserving the distance between them, those H atoms between which we do not observe any bonding in a water molecule would share a weak covalent bond in vacuum. These examples suggest the necessity of aforementioned corrections, which are applied in ReaxFF using (2.3).

$$BO_{ij} = BO'_{ij} \cdot f_1(\Delta'_i, \Delta'_j) \cdot f_4(\Delta'_i, BO'_{ij}) \cdot f_5(\Delta'_j, BO'_{ij}). \tag{2.3}$$

Here, Δ'_i is the deviation of atom i from its optimal coordination number, $f_1(\Delta'_i, \Delta'_j)$ enforces overcoordination correction, and $f_4(\Delta'_i, BO'_{ij})$, together with $f_5(\Delta'_j, BO'_{ij})$, accounts for 1–3 bond order corrections. Only corrected bond orders are used in energy and force computations in ReaxFF.

Once bond orders are calculated in this manner systemwide, a ReaxFF simulation proceeds much like a classical MD simulation. Indeed, in ReaxFF the total energy of the system is comprised of partial energy contributions (see (2.4), where we assume an implicit summation over atomic indices for each term), most of which are similar

to classical MD methods. However, due to the dynamic bonding scheme of ReaxFF, these potentials must be modified to ensure smooth potential energy curves as bonds form or break. Please refer to [2, 15] for a detailed formulation of the Reax force field.

$$\begin{aligned} E_{system} &= E_{bond} + E_{lp} + E_{over} + E_{under} \\ &\quad + E_{val} + E_{pen} + E_{3conj} \\ &\quad + E_{tors} + E_{4conj} + E_{H-bond} + E_{vdW} + E_{Coulomb}. \end{aligned} \tag{2.4}$$

2.2. Charge equilibration. Since bonding is dynamic in ReaxFF, charges on atoms cannot be fixed for the duration of simulations, as in most nonreactive classical MD models. Charges must be redistributed periodically (potentially at each time-step). The most accurate way of doing this would be to employ ab-initio methods. However, this would render the ReaxFF method unscalable, therefore conflicting with our initial goal of building a highly scalable reactive force field. To tackle the charge equilibration problem, we resort to the QEq [12] method which approximates the problem by finding an assignment of charges to atoms that minimizes the electrostatic energy of the system, while keeping the system's net charge constant. The total electrostatic energy is given as

$$\begin{aligned} E(q_1 \dots q_N) = &\sum_i \quad \text{Atomic energy of } i \text{ due to } q_i \\ &+ \sum_{i<j} \quad \text{Coulomb energy between } i \text{ and } j, \end{aligned} \tag{2.5}$$

where i and j denote atom indices and q_i denotes the partial charge on atom i. Charge dependency of an isolated atom's energy can be written as

$$E_i(q) = E_{i0} + q_i \left(\frac{\partial E}{\partial q}\right)_{i0} + \frac{1}{2} q_i^2 \left(\frac{\partial^2 E}{\partial q^2}\right)_{i0} + \cdots . \tag{2.6}$$

Here, $E_i(0)$ corresponds to the energy of an isolated neutral atom. If we detach one electron from a neutral atom, we end up with an ion of +1 charge, and the energy required to do so is called the *ionization potential* (IP). There is a related term, *electron affinity* (EA), which corresponds to the energy released when we attach one electron to a neutral atom, creating a negative ion. IP and EA are well-known quantities that can be measured for any element through physical experiments. Including only terms through second order in (2.6), we can write

$$E_i(0) = E_{i0}, \tag{2.7}$$

$$E_i(+1) = E_{i0} + \left(\frac{\partial E}{\partial q}\right)_{i0} + \frac{1}{2}\left(\frac{\partial^2 E}{\partial q^2}\right)_{i0} = E_{i0} + IP, \tag{2.8}$$

$$E_i(-1) = E_{i0} - \left(\frac{\partial E}{\partial q}\right)_{i0} + \frac{1}{2}\left(\frac{\partial^2 E}{\partial q^2}\right)_{i0} = E_{i0} - EA. \tag{2.9}$$

Solving for the unknowns in (2.8) and 2.9 gives

$$\left(\frac{\partial E}{\partial q}\right)_{i0} = \frac{1}{2}(IP + EA) = \chi_i^0, \tag{2.10}$$

$$\left(\frac{\partial^2 E}{\partial q^2}\right)_{i0} = IP - EA = H_{ii}^0. \tag{2.11}$$

Here, χ_i^0 is referred to as the *electronegativity* and H_{ii}^0 as the *idempotential* or *self-Coulomb* of i. H_{ij} corresponds to the Coulomb interaction between atoms i and j. We will revisit the computational solution of the charge equilibration problem in section 3. Below, we restate the problem more formally in the light of the discussions above:

$$\begin{aligned} &\text{Minimize } E(q_1 \dots q_N) = \sum_i \left(E_{i0} + \chi_i^0 q_i + \frac{1}{2} H_{ii}^0 q_i^2 \right) + \sum_{i<j} (H_{ij} q_i q_j) \\ &\text{subject to } q_{net} = \sum_{i=1}^{N} q_i. \end{aligned} \tag{2.12}$$

3. Algorithmic aspects of reactive modeling. Since ReaxFF is a classical MD method at its core (albeit with some important modifications), the high-level structure of our ReaxFF implementation reflects that of a classical MD code, as shown in Algorithm 1. Key components of a ReaxFF simulation include generating neighbors, computing energy and forces, and moving atoms under the effect of net forces until the desired number of steps is reached. Each of these components, however, is considerably more complex compared to a nonreactive classical MD implementation due to dynamic bonding and charge equilibration requirements. It is this complexity that forms the focus of our study.

Algorithm 1 General structure of an atomistic modeling code.

```
Read geometry, force field parameters, user control file
Initialize data structures
for t = 0 to nsteps do
   Generate neighbors
   Compute energy and forces
   Evolve the system
   Output system info
end for
```

In this section, we discuss the algorithmic techniques and optimizations in sPuReMD that deliver excellent per time-step performance and linear time scaling in system size. We also present a detailed performance analysis of the techniques and optimizations on a sample bulk water system. We choose a water system as our sample system for two reasons—first, water is ubiquitous in diverse applications of molecular simulation; and second, the ReaxFF model for water involves nearly all types of interactions present in the Reax force field—thus validating accuracy and performance of all aspects of our implementation. The bulk water system we use contains 6540 atoms (2180 water molecules) inside a $40.3 \times 40.3 \times 40.3\,\text{Å}^3$ simulation box yielding the ideal density of $1.0\,\text{g/cm}^3$.

3.1. Neighbor generation. In ReaxFF, both bonded and nonbonded interactions are truncated after a cut-off distance (which is typically 4–5 Å for bonded interactions and 10–12 Å for nonbonded interactions). Given this truncated nature of interactions, we use a procedure based on "binning" (or link-cell method) [16, 17]. First, a three-dimensional grid structure is built by dividing the domain of simulation into small cells. Atoms are then binned into these cells based on their spatial coordinates. It is easy to see that potential neighbors of an atom are either in the same cell or in neighboring cells that are within the neighbor cut-off distance r_{nbrs}

TABLE 3.1

Comparison of different r_{nbrs} to cell size ratios in terms of the time and memory required for neighbor generation in sPuReMD. Performance results from an NVT *(constant N, volume, and temperature) simulation of a bulk water system at 300 K averaged over* 100 *steps are presented. Memory space allocated for the 3D grid is also given.*

r_{nbrs}/cell size	Time (s)	Memory (MB)
1	0.15	2
2	0.11	4
3	0.13	23
4	0.18	106
5	0.27	370

of its own cell. Using this binning method, we can realize O(k) neighbor generation complexity for each atom, where k is the average number of neighbors of any atom. The associated constant with this asymptote can be high, depending on the actual method and choice of cell size. In most real-life systems, k is a constant (bounded density), resulting in an overall O(N) computational complexity for neighbor generation. Even though neighbor lists can be generated in linear time, this part is one of the most computationally expensive components of an MD simulation. Therefore it is desirable to lower the large constant associated with the O(1) computational complexity of generating the neighbors of a single atom.

Cell dimensions. The dimensions of cells play a critical role in reducing the neighbor search time. Choosing either dimension of a cell to be greater than the cutoff distance is undesirable, since this increases the search space per atom. Choosing the dimensions of cells to be roughly equal to r_{nbrs} is a reasonable choice, creating a search space of about $(3r_{\text{nbrs}})^3$ per atom. Setting the cell dimensions to be $\frac{1}{2}r_{\text{nbrs}}$, however, further decreases the search space to $(\frac{5}{2}r_{\text{nbrs}})^3$, which is clearly a better choice. Continuing to reduce the cell dimensions further would further reduce the search space (the shape of the search space approaching a perfect sphere in the lower limit), but there is an overhead associated with managing the increased number of cells. Given the significant decrease in the search space volume and small overhead associated with managing the increased number of cells, $\frac{1}{2}r_{\text{nbrs}}$ is used as the default value of cell dimensions in sPuReMD. Unless the cells are extremely sparse or overcrowded, we expect the neighbor generation routine of sPuReMD to perform well. An empirical study on a bulk water simulation, indeed, confirms this (see Table 3.1).

Atom to cell distance. Another optimization in neighbor search is to first look at the distance of an atom to the closest point of a cell before starting the search for neighboring atoms inside that cell. If the closest point of a cell to an atom is farther than r_{nbrs}, it is evident that none the atoms in that cell are potential neighbors. For the study presented in Table 3.1, this optimization yields about a 20% performance improvement.

Regrouping. Regrouping atoms that fall into the same cell together in the atom list improves neighbor search performance, since it makes better use of the cache. When the position information of an atom is retrieved, position information of other atoms adjacent to it in the list would be brought to the cache as well. If these adjacent atoms happen to be the ones inside the same cell, then it is more likely that we will find the position information of the next atom in neighbor search in the cache. Regrouping also improves the performance of force computation routines for exactly the same reason.

Verlet lists. Delayed neighbor generation using Verlet lists is another common optimization. The idea of Verlet lists is based on the observation that in a typical

simulation, atoms do not have large displacements between successive integration steps. Consequently, by choosing a suitable buffer region r_{buf} and an appropriate reneighboring frequency f_{renbr}, neighbor list generation from scratch can be delayed to every f_{renbr} steps. In this approach, at each step, every atom needs to update its distance only to the atoms in its Verlet list and determine its neighbors from within this list. But note that neighbor generations are now more expensive, due to the increased search space, and there is still the cost of updating distances between atom pairs in the Verlet lists at each step. Furthermore, the Verlet list is larger than the actual neighbors list, requiring this approach to iterate over a larger list for force computations as well. In our ReaxFF implementation, we observe that gains from a delayed reneighboring strategy are only marginal, most likely because of the effectiveness of other optimizations in the code.

Our neighbor list generation routine is given in Algorithm 2 with all the optimizations described above. A neighbor list is the most memory-consuming data structure in sPuReMD. Therefore, it is stored as a half-list to reduce the overall memory footprint. In addition, the size of the neighbor list is dynamically controlled to optimize the memory usage according to the constantly changing geometry of the simulated system. Our dynamic memory management scheme is described in detail in section 4.

Algorithm 2 Neighbor lists

```
r_nbrs ← r_nonb + r_buf
grid ← Setup_Grid( simulation_box, atom_list )
Bin_Atoms( grid, atom_list )
num_nbrs ← Estimate_Neighbors( grid, atom_list )
nbr_list gets Allocate_Neighbor_List( n, num_nbrs )
if tmodf_renbr == 0 then
  for all cell1 in grid do
    for all atom_i in cell1 do
      for all cell2 in cell1_nbrs do
        d_cell2 ← distance of atom_i to closest point of cell2
        if d_cell2 <= r_nbrs then
          for all atom_j in cell2 do
            d_ij ← |x_i − x_j|
            if d_ij <= r_nbrs then
              nbr_list_i ← j
            end if
          end for
        end if
      end for
    end for
  end for
else
  for all i, j in nbr_list do
    d_ij ← |x_i − x_j|
  end for
end if
```

3.2. Bonded and nonbonded force computations. The dynamic bonding and charge equilibration in reactive models add to the complexity of force computa-

tion, both algorithmically and with regard to performance. Four major steps in the computation of forces in ReaxFF are determining the bond orders between atom pairs in the system, computing the forces due to those bonds, updating the partial charges on each atom, and finally computing the nonbonded forces in the system. In the following subsections, we discuss algorithms and techniques used to achieve excellent performance for force computations in ReaxFF.

3.2.1. Eliminating bond order derivative lists. All bonded potentials (including even the hydrogen bond potential) depend primarily on the strength of the bonds between the atoms involved. Therefore, all forces arising from bonded interactions will depend on the derivative of the bond order terms.

A close examination of (2.3) suggests that BO_{ij} depends on all the uncorrected bond orders of both atoms i and j, which could be as many as 20–25 in a typical system. This also means that when we compute the force due to the i-j bond, the expression dBO_{ij}/dr_k evaluates to a nonzero value for all atoms k that share a bond with either i or j. Considering the fact that a single bond takes part in various bonded interactions, the same dBO_{ij}/dr_k expression may need to be evaluated several times over a single time-step. One approach to efficiently computing forces due to bond order derivatives is to evaluate the bond order derivative expressions at the start of a time-step and then use them repeatedly as necessary. Besides the large amount of memory required to store the bond order derivative list, this approach also has implications for costly memory lookups during the time-critical force computation routines.

We eliminate the need for storing the bond order derivatives and frequent lookups to the physical memory by delaying the computation of the derivative of bond orders until the end of a time-step. During the computation of bonded potentials, coefficients for the corresponding bond order derivative terms arising from various interactions are accumulated into a scalar variable $CdBO_{ij}$. After all bonded interactions are computed, we evaluate the expression dBO_{ij}/dr_k and add the force $CdBO_{ij} \times \frac{dBO_{ij}}{dr_k}$ to the net force on atom k directly. This technique, summarized below, enables us to work with much larger systems on a single processor by saving us considerable memory. It also saves considerable computational time during force computations.

$$\left(c_1 \times \frac{dBO_{ij}}{dr_k} + c_2 \times \frac{dBO_{ij}}{dr_k} + \cdots + c_n \times \frac{dBO_{ij}}{dr_k}\right) = (c_1 + c_2 + c_n) \times \frac{dBO_{ij}}{dr_k} = CdBO_{ij} \times \frac{dBO_{ij}}{dr_k}.$$

3.2.2. Lookup tables for nonbonded interactions. In general, computing nonbonded forces is more expensive than computing bonded forces, due to the larger number of interactions within the longer cut-off radii associated with nonbonded interactions. In ReaxFF, formulations of nonbonded interactions are more complex compared to their classical counterparts. These two factors increase the fraction of time required to compute nonbonded energy and forces in ReaxFF simulations.

Using a lookup table and approximating complex expressions by means of interpolation is a common optimization technique for MD simulations [18, 19]. In sPuReMD, we use a cubic spline interpolation to achieve accurate approximations of nonbonded energies and forces, while using a compact lookup table. Performance gains and additional memory required by this technique are summarized in Table 3.2. It is evident that approximating nonbonded forces with interpolation gives significant performance

TABLE 3.2

Sample bulk water system. Effect of approximating nonbonded interactions through cubic spline interpolations in terms of performance and accuracy. Time per time-step (total), time for initializations (init), and time for computing nonbonded forces (nonb) are shown for various sampling widths (w_{sampling}) and in the case of no interpolation (last row). t_{size} is the size of the lookup table in MB. rms(f_{net}) gives the RMS deviation of net forces due to interpolation.

w_{sampling} (Å)	Total (s)	init (s)	nonb (s)	t_{size} (MB)	rms(f_{net})
0.1	0.53	0.15	0.09	1.2	2.2×10^{-8}
0.01	0.56	0.16	0.11	12	2.1×10^{-12}
0.002	0.62	0.18	0.15	61	6.9×10^{-15}
none	1.80	0.31	1.20	0	0

improvements with acceptable sacrifice in terms of accuracy. For our sample water system, we observe speedups over 3x in the overall execution time.

The memory size of the interpolation table is directly proportional to the number of sample points used and the number of different element pairs in the system to be simulated. This is clearly reflected in the memory requirements of the lookup table in Table 3.2 at different sampling widths. Increasing the number of sampling points adversely affects the performance of force computations as well. In the neighbor generation phase, since we explore the search space cell by cell, neighbors of an atom are roughly clustered by their distances to that atom. A compact lookup table (requires larger w_{sampling} values) allows sPuReMD to make less memory lookups and better use of the cache data, giving a better overall performance. Even when $w_{\text{sampling}} = 0.1$ Å, nonbonded energy and force functions of ReaxFF can be approximated with enough accuracy. At this sampling width, the RMS deviation of net forces is on the order of 10^{-8}, which is well within the required tolerance for preserving the accuracy of MD simulations [20].

3.3. A fast QEq solver. One of the most computation-intensive parts of a ReaxFF simulation is the procedure for (re)assigning partial charges to atoms at each time-step. This component does not exist in typical classical MD formulations, since they rely on static charges on atoms. The charge reassignment problem is formulated as charge equilibration, with the objective of minimizing the electrostatic energy.

In this section, we first briefly describe the mathematical formulation of the charge equilibration problem and use various aspects of the formulation to motivate our choice of a Krylov subspace solver with an ILUT preconditioner. We investigate various performance and stability aspects of our solver and validate its accuracy on model systems. Two physical systems are used in addition to the bulk water system described before: a biological system composed of a lipid bilayer (biomembrane) surrounded by water molecules (56,800 atoms in total) in an orthogonal $82.7 \times 81.5 \times 80.0$ Å^3 box, and a Pentaerythritol tetranitrate (PETN) crystal with 48,256 atoms again inside an orthogonal box with dimensions $570 \times 38 \times 28$ Å^3.

3.3.1. Mathematical formulation. We extend the mathematical formulation of the charge equilibration problem by Nakano [21]. Using the method of Lagrange multipliers to solve the electrostatic energy minimization problem, we obtain the following linear systems:

$$-\chi_k = \sum_i H_{ik} s_i, \tag{3.1}$$

$$-1 = \sum_i H_{ik} t_i. \tag{3.2}$$

Here, H denotes the QEq coefficient matrix which is an N by N sparse matrix, N being the number of atoms in the simulation. The diagonal elements of H correspond to the polarization energies of atoms, and off-diagonal elements contain the electrostatic interaction coefficients between atom pairs. χ is a vector of size N, comprised of parameters determined based on the types of atoms in the system. s and t are fictitious charge vectors of size N which come up while solving the minimization problem. Finally, partial charges, q_i's, are derived from the fictitious charges based on the following formula:

$$q_i = s_i - \frac{\sum_i s_i}{\sum_i t_i} t_i. \tag{3.3}$$

The linear systems in (3.1) and (3.2) can be solved using a direct solver. However, for moderate to large sized systems, direct solvers are observed to be much more expensive than Krylov subspace methods.

3.3.2. Basic solution approaches. In ReaxFF, nonbonded interactions are computed only within the cut-off radius of r_{nonb}. Even though r_{nonb} and the number of atoms that can be found within the cut-off radius change from system to system, the maximum number of neighbors of an atom is practically bounded (typically on the order of a few hundred atoms). Therefore the number of nonzeros in H will be on the order of a few hundred entries per row, independent of the system size, making it a sparse matrix. Consequently, well-known Krylov subspace methods such as CG [22] or GMRES [13] can be used to solve the sparse linear systems of (3.1) and (3.2). Krylov subspace solvers are iterative methods requiring a costly sparse matrix-vector multiplication (SpMV) at each iteration.

Due to the large size of the linear system, the QEq solve phase is a severe bottleneck if a plain Krylov subspace solver is used. Preconditioning technique is used to transform the linear system into an equivalent one with improved spectral properties [21, 14]. The heavy diagonal of the H matrix motivates diagonal scaling as an effective accelerator for the solver.

Individual time-steps in reactive MD simulations must be much shorter than those in nonreactive classical MD methods (less than a femtosecond in ReaxFF). Therefore the geometry of the system does not change drastically between consecutive time-steps. This observation implies that solutions to (3.1) and (3.2) in one time-step yield good initial guesses about the solutions in the next time-step. Indeed, by making linear, quadratic, or higher order extrapolations on the solutions from previous steps, better initial guesses can be obtained for the charge equilibration problem.

One may derive a simple solver based on the observations above: GMRES or CG solver with a diagonal preconditioner, which extrapolates from the solutions of previous time-steps for good initial guesses. Table 3.3 summarizes the performance of these solvers on our sample systems. The stopping criteria (tolerance) for both solvers is determined by the norm of the relative residual. Even at a relatively high tolerance value of 10^{-6}, performance of the simple solver described above clearly creates a bottleneck for large simulations.

Table 3.4 explores the performance of these basic solvers, when a high accuracy solution is desired (tolerance of 10^{-10}). Results in Tables 3.3 and 3.4 show that GMRES is a better solver for the QEq problem compared to CG. Nevertheless, the performance of both solvers suggests that more powerful preconditioners and solvers

TABLE 3.3

Performance of basic approaches for bulk water and bilayer systems. GMRES(x) means that a GMRES solver with restarts after x iterations is used. The stopping criteria (tolerance) for both solvers is set to 10^{-6}, which suffices for most applications. Linear extrapolation is used for initial guesses.

System	Solver	# iterations	QEq (s)	QEq (%)
bulk water	CG+diagonal	31	0.18	23%
(6540 atoms)	GMRES(50)+diagonal	18	0.11	15%
bilayer system	CG+diagonal	38	9.44	64%
(56800 atoms)	GMRES(50)+diagonal	19	4.82	47%

TABLE 3.4

Performance of basic approaches for bulk water and bilayer systems at 10^{-10} tolerance level. Linear extrapolation is used for initial guesses.

System	Solver	# iterations	QEq (s)	QEq (%)
bulk water	CG+diagonal	95	0.54	47%
(6540 atoms)	GMRES(50)+diagonal	81	0.49	44%
bilayer system	CG+diagonal	159	39.6	88%
(56800 atoms)	GMRES(50)+diagonal	138	34.7	86%

are necessary to achieve a fast and highly scalable ReaxFF implementation. We address this issue in the remainder of this section.

3.3.3. ILUT-based preconditioning. Preconditioning techniques based on incomplete LU factorization (ILU) are effective and widely used for solving sparse linear systems [14]. As we will show below, ILU-based preconditioners help in reducing the iteration counts reported in Tables 3.3 and 3.4 dramatically. However, the QEq problem needs to be solved at each step of a ReaxFF simulation, and computing the ILU factors and applying them as preconditioners, frequently, is computationally expensive. Our observation regarding the use of the solutions from previous time-steps as initial guesses has further implications. The fact that the simulation environment evolves slowly in a ReaxFF simulation implies that the QEq coefficient matrix H and its ILU factors L and U evolve slowly as well. Therefore, same factors L and U can be used effectively as preconditioners over several steps.

ILUT factorization. In sPuReMD, we use factors from the ILU factorization of the H matrix with a threshold (ILUT). In ILUT factorization, all entries in the L and U factors with values less than the specified threshold are dropped. As the threshold gets smaller, the factors L and U become higher quality preconditioners. The downside, however, is that factorization and application of the preconditioner takes considerably longer. Empirical evidence suggests that 0.01 is an optimal threshold value for the experiments presented in this section.

Depending on the displacement rate of atoms in the system (which is determined mostly by the type of the material and simulation temperature, among other factors) and the desired accuracy of the solution, the same preconditioner can be used over tens to thousands of time-steps with only a slight increase in the iteration count. Figure 3.1 shows how the characteristics of the system affect the longevity of our ILUT-based preconditioner. In addition to our sample bulk water system, a liquid, we perform simulations on a large PETN crystal, which is a large solid crystal with 48256 atoms. QEq tolerance is set to 10^{-6} in all cases. ILUT factors are computed at the start of each simulation and used as preconditioners over the lifetime of the

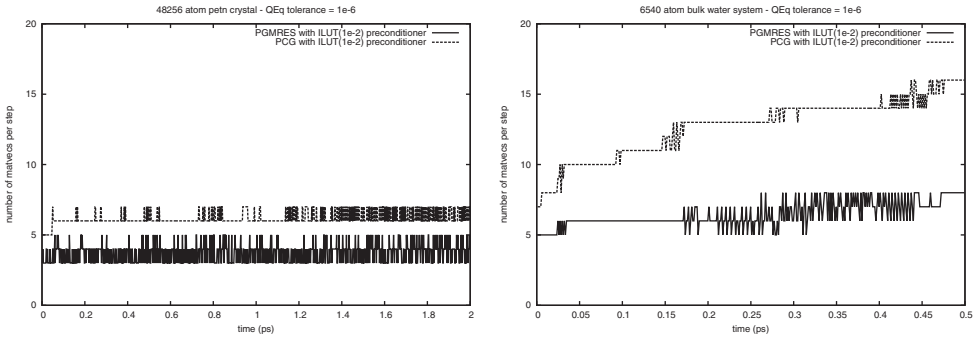

FIG. 3.1. *Number of iterations required for the convergence of PGMRES and PCG solvers using ILUT-based preconditioners at* 10^{-6} *tolerance level plotted against the simulation time. The threshold for the ILUT factorization is* 10^{-2}*, i.e., ILUT*(10^{-2})*. At the top a PETN crystal simulation is shown, and at the bottom is a bulk water simulation simulation.*

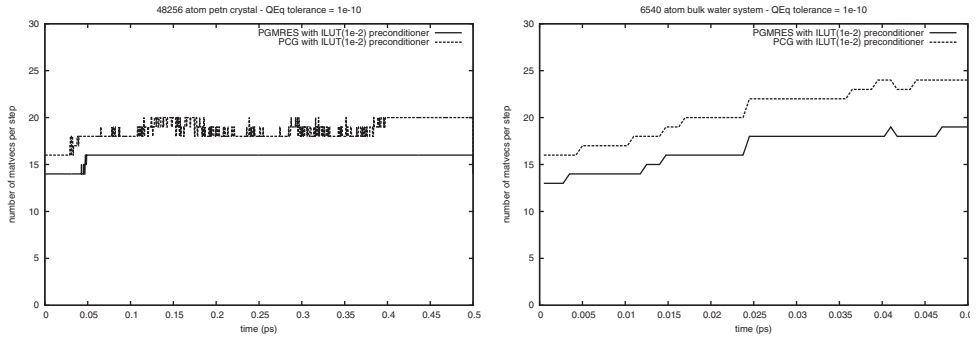

FIG. 3.2. *Number of iterations required for the convergence of PGMRES and PCG solvers using ILUT-based preconditioners at* 10^{-10} *tolerance level plotted against the simulation time. The threshold for the ILUT factorization is* 10^{-2}*, i.e., ILUT*(10^{-2})*. At the top a PETN crystal simulation is shown, and at the bottom is a bulk water simulation simulation.*

simulations. For the crystalline PETN system, the same preconditioner can be used over a few picoseconds (several thousands of steps) without a significant increase in the number of iterations (and time) required by the preconditioned GMRES (PGMRES). The same is true for the preconditioned CG (PCG) solver, albeit with slightly higher number of iterations per time-step. On the other hand, atoms in bulk water tend to have a higher displacement rate. Therefore ILUT-based preconditioners lose their effectiveness much quicker, as illustrated in Figure 3.1. Even in this case, the number of steps that these preconditioners prove to be effective is on the order of a few hundred steps—long enough to amortize the ILUT factorization cost.

We also examine how the desired accuracy of the solution affects the longevity of ILUT-based preconditioners. Figure 3.2 shows the immediate effects of lowering the QEq tolerance on the bulk water system: increased number of iterations and quickly deteriorating preconditioners (note the much shorter simulation time of 0.05 ps). On the other hand, for the PETN crystal we observe that the ILUT preconditioners are still quite effective for long durations (0.5 ps). One final observation can be made regarding the PGMRES and PCG solvers. Besides delivering a better performance than PCG, PGMRES holds the edge in the longevity of preconditioners as well. For these reasons, PGMRES is supported as the default solver in sPuReMD.

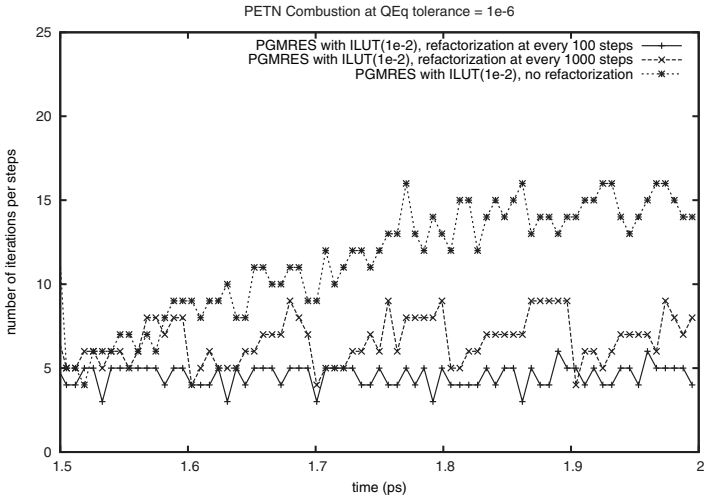

FIG. 3.3. *Number of iterations required for the convergence of PGMRES solver using ILUT-based preconditioning at* 10^{-6} *tolerance level in the final* 0.5 *ps of a PETN combustion simulation, where the system temperature is above* 2000 *K. Quick deterioration problem is alleviated by recomputing the ILUT factors at regular intervals (*100 *steps* = 0.01 *ps and* 1000 *steps* = 0.1 *ps).*

Finally, we examine the effectiveness of the ILUT-based preconditioning technique on a reactive system, where the use of ReaxFF is better justified. PETN is a very powerful explosive, with an autoignition temperature of 463 K. We simulate a PETN combustion process by heating it from 300 K to 2500 K and letting the simulation box expand freely in a 2 ps time-frame. Figure 3.3 shows the number of iterations required by the PGMRES solver at 10^{-6} tolerance level in the final 0.5 ps of the simulation where the temperature of the system is above 2000 K. At the 1.5 ps mark, new ILUT factors are computed but, as can be seen, their effectiveness quickly deteriorates due to the quickly changing simulation environment. However, this problem can be alleviated by recomputing the ILUT factors at regular intervals (at every 100 steps or 1000 steps as in Figure 3.3). Considering that the cost of ILUT factorization can be amortized in as few as 10 steps, PETN combustion simulation shows that the ILUT-based preconditioning scheme that we propose is an efficient way to tackle the computation-intensive charge equilibration problem.

Performance gain with ILUT-based preconditioners. Even when the longevity of ILUT-based preconditioners is on the order of tens of steps (which actually is the case for the bulk water system with QEq tolerance at 10^{-10}), this is long enough to amortize the cost of the ILUT factorization step. The significant improvement in the number of iterations and computation times results in impressive overall performance. Table 3.5 shows the performance of our solvers with ILUT-based preconditioners for the bulk water system at 10^{-6} and 10^{-10} tolerance levels. Compared to the corresponding values in Tables 3.3 and 3.4, QEq solvers with ILUT-based preconditioners deliver up to a 4x speedup. The fact that the overall fraction of QEq computations in the total computation time is only about 15% implies that the (re)assignment of partial charges with a high accuracy can be accomplished without any significant degradation in the overall performance.

In Table 3.6, we present the performance of our solvers with ILUT-based preconditioners for the lipid bilayer system at 10^{-6} and 10^{-10} tolerance levels. Although

TABLE 3.5
Performance of PGMRES and PCG solvers with ILUT-based preconditioners for the bulk water system averaged over 1000 *steps. Refactorization frequency is* 100 *and* 30 *for the* 10^{-6} *and* 10^{-10} *QEq tolerances, respectively. The threshold for the ILUT factorization is* 10^{-2}.

Tolerance	Solver	# iterations	QEq (s)	QEq (%)
tol=10^{-6}	*PCG*	9	0.06	9%
	PGMRES	6	0.04	6%
tol=10^{-10}	*PCG*	18	0.13	18%
	PGMRES	15	0.11	15%

TABLE 3.6
Performance of PGMRES and PCG solvers with ILUT-based preconditioners for the bilayer system averaged over 100 *steps. Refactorization frequency is* 100 *and* 50 *for the* 10^{-6} *and* 10^{-10} *QEq tolerances, respectively. The threshold for the ILUT factorization is* 10^{-2}.

Tolerance	Solver	# iterations	QEq (s)	QEq (%)
tol=10^{-6}	*PCG*	9	2.39	31%
	PGMRES	6	1.58	23%
tol=10^{-10}	*PCG*	27	6.93	57%
	PGMRES	23	5.96	53%

the number of iterations required by QEq solvers at 10^{-6} tolerance level is the same as that in the bulk water system, the share of QEq in total computational time is much higher for the bilayer simulation (23% vs. 6% for PGMRES). This is due to less effective use of cache during the sparse matrix-vector multiplication in each iteration. As the tolerance is lowered to 10^{-10}, the fraction of QEq increases even further to 53–57%, with the effect of the increase in the number of iterations required for convergence. Optimizations for improved cache performance are currently being implemented in sPuReMD to improve this fraction.

4. Memory management. There are two important aspects of designing a flexible and efficient memory manager for atomistic simulations. First is the choice of appropriate data structures for representing various lists required during force computations in a way that provides efficient access to these lists while minimizing the memory footprint. For most classical MD simulations, the memory footprint is not a major concern because the neighbor list is the only data structure that requires significant space. However, in a reactive force field the dynamic nature of bonds, three-body interactions, and four-body interactions, together with the significant amount of bookkeeping required for these interactions, necessitate larger memory regions and more sophisticated procedures for managing that region. The second important aspect is the adaptation of data structures to a constantly changing simulation environment. Since the interaction patterns in a reactive environment cannot be predicted at the start of a simulation, we cannot precisely estimate the amount of memory required by all lists throughout the simulation. A reallocation mechanism in sPuReMD constantly adapts its data structures based on the needs of the system being simulated. In this section, we first describe various data structures used in sPuReMD and then discuss the reallocation mechanism implemented.

We store neighbor lists and the QEq matrix in compressed sparse row (CSR) format. Both of these lists are half-lists; i.e., we store only the upper half of the matrix in each case. The manner in which these lists are generated and accessed is well suited to the CSR format. Bond lists are stored as full lists in *modified* CSR format, which is a full list because higher order bonded interactions (three-body and

four-body) are derived from the bond list. We call the format of our bond lists *modified* CSR format, because the space reserved for each atom on the list is contiguous, but the actual data stored are not. Before allocating the bond lists, the number of bonds for each atom is estimated. Let eb_i denote the number of estimated bonds for atom i; then $\max(2eb_i, \text{MIN\_BONDS})$ slots, where MIN_BONDS is a predefined constant, are allocated to atom i in the bond list. This conservative allocation scheme prevents any overwrites in subsequent steps, while reducing the frequency of bond list reallocations through the simulation as the system evolves and bonding patterns change.

A three-body interaction list is built from the bond list and stored in CSR format indexed not by the individual atoms but by the bonds in the bond list. Four-body structures are constructed from the three-body interactions. Four-body structures are not stored, since there are no higher order interactions in ReaxFF. Energy and forces due to the discovered four-body interactions are computed on the fly.

We maintain a dedicated hydrogen bond list, because in ReaxFF, hydrogens are often bonded to more than one atom, and the neighbors of hydrogen atoms are spread throughout the entire neighbors list (recall that the neighbors list is a half-list). The hydrogen bond list uses the same *modified* CSR format described above.

In order to minimize the memory footprint of sPuReMD, we adopt a three-stage memory management scheme: estimation, monitoring, and reallocation. At the start of the simulation, sizes of all lists are estimated just by counting (but not storing) the neighbors and bonds of each atom—this is a one-time, inexpensive calculation. During each step of the simulation, the utilization of lists are monitored carefully. If the utilization of a list reaches a prescribed threshold, that list is reallocated by conservative estimations on its size based on the current utilization. To avoid significant overhead with reallocations (such as copying of stored data), we ensure that the reallocation daemon is invoked only at specific instances.

5. ReaxFF vs. other simulation methods. In this section, we compare the quality of the results and the scale of simulations possible (both in terms of time and system size) using ReaxFF with quantum MD methods (represented by CPMD [25]) and nonreactive classical MD methods (represented by GROMACS [18]) comprehensively on a well-studied class of materials, hydrocarbons. Hydrocarbons are mostly used as combustible fuel sources and comprise one of the most important sources of energy. Our choice of this system is motivated by several factors: (i) they are relatively well studied, and therefore considerable experimental data is available for validation; (ii) the diversity of interactions in hydrocarbons stresses all parts of our code; and (iii) systems of various sizes can be prepared relatively easily. We choose to work with hexane, C_6H_{14}, in liquid form.

Initial configurations are prepared by randomly placing the desired number of hexane backbone chains inside a large box to minimize overlap among hexane molecules. Initial configurations are first energy minimized and then run under the NPT ensemble using GROMACS to quickly bring them to equilibrium under 1 atm pressure and 200 K temperature. The united atom force field 43A-S3 developed by Chiu et al. [32] is used for these simulations. This force field is designed to reproduce liquid properties of hydrocarbons and biological materials accurately. In our opinion, even if this force field does not have explicit hydrogens, it is a most desirable choice for such simulations. Results presented for structural comparison are derived from the hexane_{343} system—343 hexane molecules (6860 atoms) inside an orthogonal box with periodic boundary conditions. Systems of various sizes (343, 512, 1000, 1728, 3375 molecules) are prepared for the scalability analysis.

TABLE 5.1

Structural properties of hexane molecules obtained through different simulation methods (ReaxFF, a reactive classical force field, and CPMD, a quantum MD method) compared to experimental results.

Property	ReaxFF	ab-initio	Experimental [23]
C-H bond	1.09 ± 0.01	1.100	1.118 ± 0.006
C-C bond	1.57 ± 0.01	1.533	1.533 ± 0.003
<C-C-C	108.0 ± 2.9	114.2	111.9 ± 0.4
<C-C-H	111.0 ± 0.0	109.5	109.5 ± 0.5
<H-C-H	106.6 ± 0.0	106.5	–
$q_{\text{C-tip}}$	-0.171	-0.205	–
$q_{\text{C-mid}}$	-0.080	0.033	–
$q_{\text{H-tip}}$	0.040	0.047	–
$q_{\text{H-mid}}$	0.040	$-0.10 \sim 0.10$	–

5.1. Structural comparison. Configurations equilibrated using GROMACS are used for our ReaxFF simulations. The force field for hexane simulations in GROMACS does not treat H atoms separately; they are modeled as parts of superatomic structures in the force field. This has two advantages for GROMACS simulations. First, the number of atoms in the simulation is lowered significantly (from 20 atoms to 6 atoms per molecule). Second, with the elimination of individual H atoms, which typically have bond vibrational frequencies at subfemtoseconds, the simulation timesteps can be stretched to a few femtoseconds. In ReaxFF, all atoms must be modeled explicitly to correctly capture reactions. We use the Avogadro program [24] for approximately placing the missing H atoms on the GROMACS output configurations. Avogadro places H atoms onto the hexane backbone based on the local topology; it does not take into account the presence of nearby hexane molecules. Therefore, we energy minimize the resulting configurations with H atoms added for 2.5 ps using sPuReMD and equilibrate them under NVT at 200 K for another 2.5 ps. Analysis on the final configurations presented in Table 5.1 suggests that sPuReMD produces hexane structures that are in near-perfect agreement with experimental results in [23] and geometry optimizations based on CPMD simulations using PBE Troullier–Martins pseudopotentials. Note that most of the analyses in Table 5.1 are not applicable for GROMACS simulations since H atoms are not explicitly modeled and partial charges on atoms are given as inputs to the simulation.

5.2. Scalability comparison. Here, we compare sPuReMD with other MD methods to provide an idea of where ReaxFF sits within the spectrum of MD methods in terms of simulation capabilities. We examine the time scales and system sizes where each method, namely, GROMACS (as a nonreactive classical MD method), sPuReMD (as a ReaxFF implementation), and CPMD (as a quantum MD method), is applicable. For this purpose, various hexane systems of different sizes (343, 512, 1000, 1728, 3375 molecules) have been prepared as described above to be used in GROMACS and sPuReMD simulations. For each method, only single-processor simulations are performed to quantify true computational cost, independent of parallelization overhead. This limits our ability to work with large systems using CPMD, though. Systems used in CPMD simulations (1, 3, and 5 molecules) are formed by selecting nearby molecules from the large simulation boxes used with sPuReMD. Due to the small size of CPMD systems, periodic boundary conditions are dropped in CPMD simulations. Perdew, Burke, and Ernzerhof functionals (PBE) [26, 27] for DFT and corresponding Trouiller–Martins norm-conserving pseudopotentials with a wavefunction cut-off of 75 Rydberg for both C and H atoms are used.

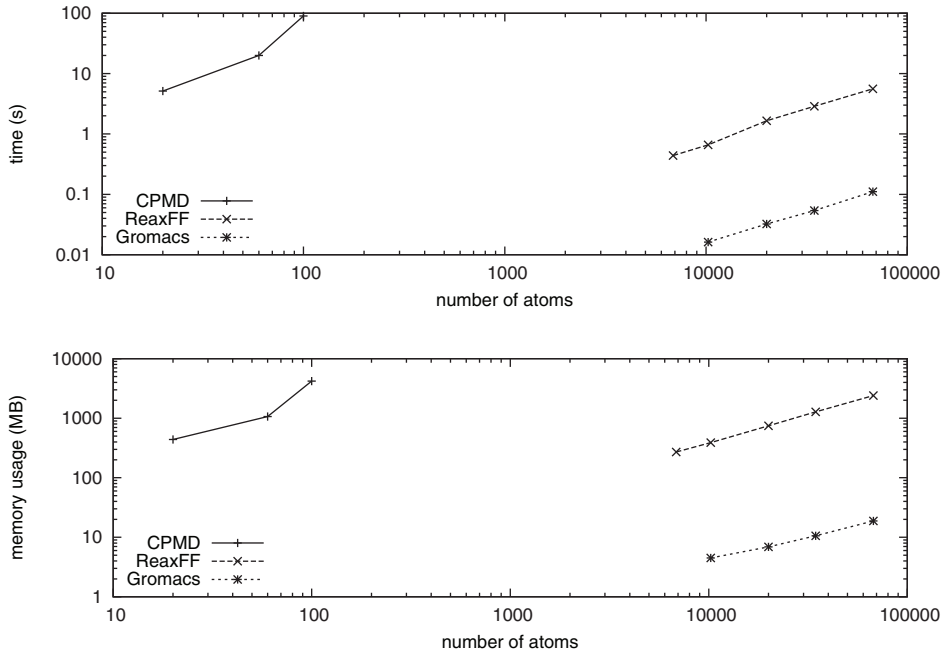

FIG. 5.1. *Log-log plot of the time spent per simulation step and total allocated memory for different molecular simulation methods, namely, CPMD (an ab-initio method), ReaxFF (a reactive MD method), and GROMACS (a classical MD method).*

Figure 5.1 shows scaling of the run-time per time-step and total memory requirements of these three molecular simulation methods. All simulations are performed under the microcanonical ensemble (NVE), and reported run-times are averaged values over several hundred steps. Reported memory requirements are the peak values observed during the lifetime of the simulations. The computational complexity of CPMD scales as $O(N^3)$, and as we increase the system size, we must increase the box dimensions to maintain accuracy without periodic boundary conditions. Therefore, we observe a rapid increase in the run-time per time-step for CPMD simulations in Figure 5.1. A growing box size causes a similar growth in memory requirements of CPMD simulations as well.

On the other hand, sPuReMD and GROMACS exhibit similar scaling behaviors, both in terms of run-time and memory allocated as indicated by almost parallel scaling curves for both methods. Not surprisingly, sPuReMD is slower (by a factor of about 50) and requires significantly more memory (about two orders of magnitude) than GROMACS. However, as noted before, GROMACS does not explicitly model the H atoms and represents a hexane molecule with only six particles, as opposed to the 20 particles used in ReaxFF. While this simplified model reduces the number of bonded interactions in GROMACS by about a factor of four, it has more significant impact on the number of nonbonded pairs, decreasing the number of such pairs by more than an order of magnitude. Considering that nonbonded force computations take about 90% of the total time in a typical classical MD simulations [28] and that most of the allocated space is used for storing nonbonded pairs, the run-time per time-step and memory footprint of GROMACS simulations would come to the same order as those of sPuReMD, had H atoms been modeled explicitly. There is the additional burden of computing bonds and three-body and four-body structures dynamically

TABLE 5.2

Breakdown of sPuReMD run-time into major components. For each system, we show the total time, nbrs, init, bonded, nonb, and QEq running times in seconds. The last column presents the percentage of QEq run-time within the total time.

System	#atoms	Total	nbrs	init	Bonded	nonb	QEq	QEq(%)
hexane_{343}	6860	0.44	0.02	0.17	0.09	0.11	0.04	9
hexane_{512}	10240	0.66	0.03	0.25	0.13	0.16	0.06	9
hexane_{1000}	20000	1.66	0.06	0.50	0.26	0.32	0.48	29
hexane_{1728}	34560	2.88	0.10	0.87	0.45	0.58	0.81	28
hexane_{3375}	67500	5.59	0.18	1.69	0.88	1.10	1.59	28

and equilibrating charges at every step of a ReaxFF simulation, and this explains the larger running time per time-step and memory footprint figures for sPuReMD. Nevertheless, we believe the demonstrated performance of sPuReMD is extremely promising, considering its significantly enhanced simulation scope.

Of notable interest in Figure 5.1 is the jump in sPuReMD's run-time per time-step between hexane_{512} and hexane_{1000}. To shed more light on the reasons for this superlinear scaling, we identify six major components in sPuReMD:

- *nbrs:* Neighbor generation, where all atom pairs falling within the neighbor cut-off distance r_{nbrs} are identified.
- *init:* Generation of the charge equilibration (QEq) matrix, bond list, and H-bond list based on the neighbor list.
- *bonded:* The part that includes computation of forces due to all interactions involving bonds (hydrogen bond interactions are included here as well). This part also includes identification of three-body and four-body structures in the system.
- *ilut:* The part where we compute the ILUT-based preconditioners for the QEq matrix.
- *QEq:* Charge equilibration part that requires the solution of a large sparse linear system. This involves computationally expensive matrix-vector products.
- *nonb:* The part that computes nonbonded interactions (van der Waals and Coulomb).

In Table 5.2, we show how the run-time for each of these parts (except for ilut) changes as the system size increases. As can be seen, the run-time for every part except for QEq scales linearly with the system size. As we move beyond the hexane_{512} system (10240 atoms), the matrix-vector product in QEq becomes much costlier and causes the jump mentioned above. This is also reflected as the increased share of QEq within the total run-time per time-step beyond hexane_{512}.

ILUT factorization is not performed at every step, to amortize its cost as discussed in section 3.3. Therefore we do not explicitly present ilut running times in Table 5.2. For all sPuReMD experiments reported in this section, the threshold for the QEq solver was set to 10^{-6}, and we perform the ILUT factorizations once every 100 steps. The ILUT-based preconditioning scheme described in section 3.3.3 takes only 5–6 iterations per step using the PGMRES algorithm to solve the QEq problem. The cost of ILUT factorization is about 0.30 s for the hexane_{343} system and 2.80 s for the hexane_{3375} system, which is negligible when averaged over 100 steps. These figures also suggest that ilut scales nicely with the system size.

6. Comparison with LAMMPS. The first-generation ReaxFF implementation of van Duin et al. [2] demonstrated the validity of the ReaxFF force-field in the

TABLE 6.1

Comparison of LAMMPS and sPuReMD performance on a single processor for different systems. For each system, we present the number of atoms in the simulation, LAMMPS and sPuReMD run-times per time-step (in seconds) on average, and the speedup achieved by sPuReMD over LAMMPS.

System	#atoms	LAMMPS (s)	sPuReMD (s)	Speedup
PETN crystal	3712	2.61	0.35	7.5
bulk water	6540	3.61	0.58	6.2
hexane_{343}	6860	3.73	0.44	7.8

TABLE 6.2

Comparison of the QEq solvers in LAMMPS and sPuReMD codes on different systems. For each system, we present the charge equilibration time (in seconds) and the number of iterations taken per time-step by the QEq solvers in LAMMPS and sPuReMD.

	LAMMPS		sPuReMD	
System	QEq (s)	# iterations	QEq (s)	# iterations
PETN crystal	0.19	32	0.02	5
bulk water	0.38	34	0.04	6
hexane_{343}	0.40	35	0.04	6

context of various applications. Thompson and Cho [29] successfully ported this initial implementation, which was not developed for a parallel environment, into their parallel MD package LAMMPS [16]. Except for the charge equilibration part, the ReaxFF implementation in LAMMPS is based on the original Fortran code of van Duin [2], significant portions of which were included directly (as Fortran routines called from C++) to ensure consistency between the two codes. So in this section, we compare the performance of sPuReMD to the state-of-the-art ReaxFF implementation available over the public domain, the REAX package [29] inside LAMMPS [16, 31].

In Table 6.1, we compare the performance of sPuReMD to the LAMMPS code for the experimental systems described above. For each system, we report the average run-time per time-step for both codes. Our results show that sPuReMD achieves a six-to-eightfold speedup over LAMMPS, due to various algorithmic and numerical enhancements presented in this paper. Note that LAMMPS is a parallel MD simulation program, and therefore it contains some potential overhead due to communications. On a single processor such overhead is not significant—at the end of each simulation, communication time reported by LAMMPS accounts for less than 1% of the total time.

Among various optimizations, our ILUT-based preconditioning scheme for solving the QEq problem stands out. In Table 6.2, we compare the average time and number of iterations per step required by the QEq solvers in both codes. The charge equilibration calculation currently in LAMMPS uses a standard parallel conjugate gradient algorithm [30]. Compared to this, our advanced QEq solver delivers about an order of magnitude improvement in performance.

Finally, in Table 6.3 we compare the memory footprint of both codes. Since the LAMMPS implementation uses static allocation for atom, neighbor, and interaction lists, it exhibits a constant memory footprint for all simulations. However, sPuReMD is able to achieve a considerably smaller memory footprint owing to its intelligent memory management scheme. This scheme allows us to work with systems of much larger sizes without having to tune compilation parameters.

TABLE 6.3
Comparison of the memory footprints of LAMMPS and sPuReMD codes on different systems. For each system, we present the amount of memory required by LAMMPS and sPuReMD in MB.

System	LAMMPS (MB)	sPuReMD (MB)
PETN crystal	2500	170
bulk water	2500	250
hexane_{343}	2500	270

7. Concluding remarks. In this paper, we present the design, implementation, algorithms, data structures, and optimizations underlying the state-of-the-art sPuReMD package for reactive molecular dynamics simulations. We demonstrate that sPuReMD (i) is highly accurate in terms of modeling accuracy, (ii) has linear scaling memory and run-time characteristics in terms of system size, (iii) is significantly faster than existing/comparable packages, and (iv) has been demonstrated in diverse application contexts, ranging from biomembranes (lipid bilayers) to explosives (PETN).

sPuReMD represents a unique resource for the scientific simulations community. It is in limited release and is currently in use at over ten institutions (including MIT, CalTech, Penn State, Illinois, Purdue, USF, Sandia, LANL, ORNL, and ARL). It is being used to model systems ranging from novel nanoscale devices (ORNL), materials models (Purdue, Penn State), explosives (ARL), and biophysical simulations (USF, CalTech). It also provides a basis for software, model, and method development (Sandia, Penn State).

Acknowledgment. Special thanks are due to Aidan Thompson and Steve Plimpton (Sandia), whose work on LAMMPS motivated many of our optimizations.

REFERENCES

[1] R.A. FRIESNER, *Ab initio quantum chemistry: Methodology and applications*, Proc. Natl. Acad. Sci., 102 (2005), pp. 6648–6653.
[2] A.C.T. VAN DUIN, S. DASGUPTA, F. LORANT, AND W.A. GODDARD III, *ReaxFF: A reactive force field for hydrocarbons*, J. Phys. Chem. A, 105 (2001), pp. 9396–9409.
[3] J.M. THIJSSEN, *Computational Physics*, Cambridge University Press, Cambridge, UK, 2003.
[4] A.C.T. VAN DUIN, J.M.A. BAAS, AND B. VAN DE GRAAF, *Delft molecular mechanics: A new approach to hydrocarbon force fields. Inclusion of a geometry-dependent charge calculation*, J. Chem. Soc. Faraday Trans., 90 (1994), pp. 2881–2895.
[5] T.A. HALGREN AND W. DAMM, *Polarizable force fields*, Current Opinion in Structural Biology, 11 (2001), pp. 236–242.
[6] J.E. DAVIS, G.L. WARREN, AND S. PATEL, *Revised charge equilibration potential for liquid alkanes*, J. Phys. Chem. B, 112 (2008), pp. 8298–8310.
[7] D.W. BRENNER, *Empirical potential for hydrocarbons for use in simulating the chemical vapor deposition of diamond films*, Phys. Rev. B, 42 (1990), pp. 9458–9471.
[8] J. TERSOFF, *Empirical interatomic potential for carbon, with applications to amorphous carbon*, Phys. Rev. Lett., 61 (1988), pp. 2879–2882.
[9] G.C. ABELL, *Empirical chemical pseudopotential theory of molecular and metallic bonding*, Phys. Rev. B, 31 (1985), pp. 6184–6196.
[10] D.W. BRENNER, O.A. SHENDEROVA, J.A. HARRISON, S.J. STUART, AND S.B. SINNOTT, *A second-generation reactive empirical bond order (REBO) potential energy expression for hydrocarbons*, J. Phys. Condens. Matter, 14 (2002), pp. 783–802.
[11] S.J. STUART, A.B. TUTEIN, AND J.A. HARRISON, *A reactive potential for hydrocarbons with intermolecular interactions*, J. Chem. Phys., 112 (2000), pp. 6472–6486.
[12] A.K. RAPPE AND W.A. GODDARD III, *Charge equilibration for molecular dynamics simulations*, J. Phys. Chem., 95 (1991), pp. 3358–3363.
[13] Y. SAAD AND M.H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[14] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[15] A.C.T. van Duin, A. Strachan, S. Stewman, Q. Zhang, X. Xu, and W.A. Goddard III, *ReaxFFSiO reactive force field for silicon and silicon oxide systems*, J. Phys. Chem. A, 107 (2003), pp. 3803–3811.
[16] S.J. Plimpton, *Fast parallel algorithms for short-range molecular dynamics*, J. Comput. Phys., 117 (1995), pp. 1–19.
[17] D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications*, 2nd ed., Comput. Sci. Ser. 1, Academic Press, Orlando, FL, 2001.
[18] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl, *GROMACS* 4: *Algorithms for highly efficient, load-balanced, and scalable molecular simulation*, J. Chem. Theory Comput., 4 (2008), pp. 435–447.
[19] D. Wolff and W.G. Rudd, *Tabulated potentials in molecular dynamics simulations*, Comput. Phys. Comm., 120 (1999), pp. 20–32.
[20] R. Zhou, E. Harder, H. Xu, and B.J. Berne, *Efficient multiple time step method for use with Ewald and particle mesh Ewals for large biomolecular systems*, J. Chem. Phys., 115 (2001), pp. 2348–2358.
[21] A. Nakano, *Parallel multilevel preconditioned conjugate-gradient approach to variable-charge molecular dynamics*, Comput. Phys. Comm., 104 (1997), pp. 59–69.
[22] J.R. Shewchuk, *An Introduction to the Conjugate Gradient Method without the Agonizing Pain*, Technical report, School of Computer Science, Carnegie Mellon University, 1994.
[23] R.A. Bonham, L.S. Bartell, and D.A. Kohl, *The molecular structures of n-Pentane, n-Hexane and n-Heptane*, J. Amer. Chem. Soc., 81 (1959), pp. 4765–4769.
[24] M.D. Hanwell, D.E. Curtis, and G.R. Hutchinson, *Avogadro: A Framework for Quantum Chemistry Calculation and Visualization*, 2009; available online from http://avogadro.openmolecules.net/.
[25] CPMD consortium, *CPMD: CPMD consortium page*, 2008; available online from http://www.cpmd.org/.
[26] J.P. Perdew, K. Burke, and M. Ernzerhof, *Generalized gradient approximation made simple*, Phys. Rev. Lett., 77 (1996), pp. 3865–3868.
[27] J.P. Perdew, K. Burke, and M. Ernzerhof, *Errata: Generalized gradient approximation made simple*, Phys. Rev. Lett., 78 (1997), p. 1396.
[28] D.E. Shaw, M.M. Deneroff, R.O. Dror, J.S. Kuskin, R.H. Larson, J.K. Salmon, C. Young, B. Batson, K.J. Bowers, J.C. Chao, M.P. Eastwood, J. Gagliardo, J.P. Grossman, C.R. Ho, D.J. Ierardi, I. Kolossvry, J.L. Klepeis, T. Layman, C. McLeavey, M.A. Moraes, R. Mueller, E.C. Priest, Y. Shan, J. Spengler, M. Theobald, B. Towles, and S.C. Wang, *Anton: A special-purpose machine for molecular dynamics simulation*, ISCA'07, San Diego, CA, 2007.
[29] A. Thompson and H. Cho, *LAMMPS/ReaxFF Potential*, 2010; available online from http://lammps.sandia.gov/doc/pair_reax.html.
[30] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, 2nd ed., SIAM, Philadelphia, 1994.
[31] S.J. Plimpton, P. Crozier, and A. Thompson, *LAMMPS Molecular Dynamics Simulator*, 2010; available online from http://lammps.sandia.gov/index.html.
[32] S.-W. Chiu, S.A. Pandit, H.L. Scott, and E. Jakobsson, *An improved united atom force field for simulation of mixed lipid bilayers*, J. Phys. Chem. B, 113 (2009), pp. 2748–2763.