## REACTIVE MOLECULAR DYNAMICS: NUMERICAL METHODS AND ALGORITHMIC TECHNIQUES

# HASAN METIN AKTULGA \*, SAGAR A. PANDIT †, ADRI C. T. VAN DUIN ‡, AND ANANTH Y. GRAMA $\S$

Abstract. Modeling atomic and molecular systems requires computation-intensive quantum mechanical methods such as, but not limited to, density functional theory (DFT) [11]. These methods have been successful in predicting various properties of chemical systems at atomistic detail. 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 thousands of 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 [9] overcomes the limitations of classical molecular dynamics approximations by carefully incorporating limited nonlocality (to mimic quantum behavior) through empirical bond order potential. This reactive molecular dynamics method, called ReaxFF, achieves essential quantum properties, while retaining computational simplicity of classical molecular dynamics, 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 equilibriation. This requires the solution of large linear systems ( $10^8$  degrees of freedom and beyond) with shielded electrostatic kernels at each timestep. Individual timesteps are themselves typically in the range of tenths of femtoseconds, requiring optimizations within and across timesteps 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 molecular dynamics code. We describe various data structures, and the charge equilibration solver at the core of the simulation engine. This Krylov subspace solver relies on an ILU-based preconditioner, 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-timestep time, linear time scaling in system size, and a low memory footprint. sPuReMD is available over the public domain and is currently being used to model diverse systems ranging from oxidative stress in bio-membranes to strain relaxation in Si-Ge nanorods.

 ${\bf Key}$  words. Reactive Molecular Dynamics, Bond Order Potentials, ReaxFF, Charge Equilibration.

AMS subject classifications.

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, nano-scale devices, etc.). These methods can model physical reality under extreme conditions, not easily reproduced in 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. The methods start from first principles quantum mechanics and

<sup>\*</sup>Department of Computer Science, Purdue University, West Lafayette IN 47907 (haktulga@cs.purdue.edu)

<sup>&</sup>lt;sup>†</sup>Department of Physics, University of South Florida, Tampa, FL 33620 (pandit@usf.edu)

<sup>&</sup>lt;sup>‡</sup>Department of Mechanical and Nuclear Engineering, Pennsylvania State University, University Park, PA 16802 (acv13@psu.edu)

<sup>&</sup>lt;sup>§</sup>Department of Computer Science, Purdue University, West Lafayette, IN 47907 (ayg@cs.purdue.edu)

make few approximations while deriving the solution. For these reasons, the quantum dynamics methods are often referred as ab-inito methods. Because of their modeling fidelity, ab-initio methods usually produce more accurate and reliable results than atomistic methods.

Due to the non-locality of interactions in quantum mechanics, the high 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 ab*initio* problems (please refer to [11] 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 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 [11].

The high computational cost associated with *ab-initio* methods restrict their applicability to the systems on the order of thousands of atoms and few picoseconds of simulation time. For many real-life problems, 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 sub-system 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)" [12]. 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, therefore they cannot be used to model reactive systems, and (iii) in most atomistic methods, charges are kept fixed throughout the simulation. Although polarizable force fields were intruduced almost two decades ago [1], they have only recently gained significant attention for modeling charge transfer in empirical force fields [2]. Polarization is achieved either by inducible point dipole methods, or by fluctuating charge models. Even though polarizable force fields are built upon their non-polarizable counter-parts, their development still requires considerable effort since charges are not assumed to be fixed in the target system. This requires most parameters to be re-tuned. Better characterization of target systems have been reported in literature through the use of polarizable force fields [3].

With apriori knowledge of the reactions in a system of interest and spatially localized reactivity, we can use mixed quantum mechanics/molecular mechanics (QM/MM) methods 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 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 *apriori*, size limitation on the reactive site, difficulties in coupling QM/MM regions, and the intricacies of setting up/ running such simulations have prevented mixed QM/MM methods from being widely applied to the study of large scale reactive systems [11].

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 wave functions through a bond order term that describes the bonds in the system dynamically based on the local neighborhoods of each atom. These include Bond Energy Bond Order (BEBO) and VALBOND methods. A widely used bond order potential has been the Reactive Empirical Bond Order (REBO) potential [6]. REBO is built on the Tersoff potential [5], which was inspired by Abell's work [4]. 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 verison [7]. Like many other bond order potentials, this new version of REBO lacks long range interactions, which are important in modeling molecular systems. AIREBO was an attempt by Stuart *et al.* 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 [8].

The ReaxFF method of van Duin et al [9] is the first 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* [9], 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 though the bond order terms. In spite of these simplifications, the challenges associated with implementing an efficient and scalable reactive force field are formidable. In a 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 timestep, 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 accurately because electrostatic interactions play crucial roles in describing most systems and over/ under-estimation of charges would result in violation of energy conservation.

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 describe the potentials used in our implementation. 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. Simple pair-wise nature of non-bonded interactions allows the use of interpolation schemes to expedite the computation of energy and forces due to non-bonded interactions. The complexity of bonded interactions on one hand, and the possibility of expediting the non-bonded interactions on the other hand bring the computational time required for bonded interactions on par with that of non-bonded interactions (note that in classical MD methods, time required for bonded interactions is negligible compared to the time required for non-bonded interactions). As mentioned above, a highly accurate charge equilibration method is desirable to obtain reliable results from ReaxFF simulations. However, the QEq method [16] 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 a preconditioned GMRES method with an ILU-based preconditioner. The solver relies heavily on optimizations across iterations to achieve an excellent per timestep running time. The dynamic nature of bonds in ReaxFF requires dynamic bond lists, and subsequently dynamic angle and dihedral lists reconstructed at every timestep. 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 literature in Section 5. Characterization of efficiency and performance of our implementation on systems of different types is presented in Section 6. 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 classical molecular dynamics, atoms constitute molecules through static bonds, akin to a balls and springs

model in which springs are statically attached. This approach cannot simulate chemical reactions, since reactions correspond to bond breaking and formation. In reactive molecular dynamics using the Reax force field (ReaxFF), 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.

**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 (Eq. (2.1)), which computes the bond order in terms of the types of atoms i and j, and the distance between them:

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

In Eq. (2.1),  $\alpha$  corresponds to  $\sigma - \sigma$ ,  $\sigma - \pi$ , or  $\pi - \pi$  bonds,  $a_{\alpha}$  and  $b_{\alpha}$  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:

$$BO'_{ij} = BO^{\sigma'}_{ij} + BO^{\pi'}_{ij} + BO^{\pi\pi'}_{ij}$$
(2.2)

One cannot model the complex bond structure observed in real-life systems just by using pair-wise 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 moleculue, detach the two H atoms from the middle O atom and put them in vacuum while preserving the distance between them. Those two same Hatoms between which we do not observe any bonding in a water molecule would then share a weak covalent bond. These examples suggest the necessity of aforementioned corrections, which are applied in ReaxFF using Eq. (2.3).

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

$$(2.3)$$

Here,  $\Delta'_i$  is the deviation of atom *i* from its optimal coordination number,  $f_1(\Delta'_i, \Delta'_j)$ enforces over-coordination correction, and  $f_4(\Delta'_i, BO'_{ij})$ , together with  $f_5(\Delta'_j, BO'_{ij})$ account 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 system-wide, the simulation process resembles classical MD. Indeed, in ReaxFF the total energy of the system is comprised of partial energy contributions (Eq. (2.4)), 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.

We briefly describe each type of interaction that constitutes the Reax force field and provide energy expressions for them. Since the negative gradient of an interaction energy yiends corresponding force, we omit formulae for forces. The total energy can be written as sum of different energy terms as follows:

$$E_{system} = E_{bond} + E_{lp} + E_{over} + E_{under} + E_{val} + E_{pen} + E_{3conj} + E_{tors} + E_{4conj} + E_{H-bond} + E_{vdW} + E_{Coulomb}$$
(2.4)

**2.2.** Bond Energy. Classical MD methods adopt a spring model in which the energy of a bond is determined solely by its deviation from the optimal bond distance, therefore ignoring the effects of neighboring bonds. ReaxFF, however, takes a more rigorous approach by computing the energy incident on a bond from all bond order constituents. The higher the bond order, the lower the energy and the stronger the force associated with the bond. The following equation (Eq. (2.5)) ensures that the energy and force due to a bond smoothly go to zero as the bond breaks:

$$E_{bond} = -D_e^{\sigma} \cdot BO_{ij}^{\sigma} \cdot exp \left\{ p_{be1} \left( 1 - \left( BO_{ij}^{\sigma} \right)^{p_{be2}} \right) \right\}$$

$$-D_e^{\pi} \cdot BO_{ij}^{\pi\pi} \cdot BO_{ij}^{\pi\pi}$$
(2.5)

**2.3.** Lone Pair Energy. This energy term accounts for unpaired electrons of an atom, therefore classical MD terms do not explicitly compute this term. In a nicely formed and equilibrated system, lone pair energy does not have a significant contribution to the total energy, however, it is important for describing atoms with defective bonds. Lone-pair energy is computed using the following equation:

$$E_{lp} = \frac{p_{lp2} \cdot \Delta_i^{lp}}{1 + exp\{-75 \cdot \Delta_i^{lp}\}}$$
(2.6)

In this equation (Eq. (2.6)),  $\Delta_i^{lp} = n_{opt}^{lp} - n_i^{lp}$  essentially corresponds to the number of unpaired electrons.

2.4. Over & Under-coordination Energy. Despite the valence correction applied during bond order corrections, there may still remain some over or undercoordinated atoms in the system. Over-coordination energy, as given by Eq. (2.7), penalizes over-coordinated atoms.

If there is a  $\pi$ -bond between atoms i and j, then the energy due to the resonant  $\pi$ -electron between these atomic centers is accounted by the Eq. (2.8) which is called under-coordination energy.

$$E_{over} = \Delta_i^{lpcorr} \cdot \frac{\sum_{j \in nbrs(i)} p_{ovun1} \cdot D_e^{\sigma} \cdot BO_{ij}}{\left(\Delta_i^{lpcorr} + Val_i\right) \left(1 + exp\{p_{ovun2} \cdot \Delta_i^{lpcorr}\}\right)}$$
(2.7)

$$E_{under} = -p_{ovun5} \cdot f_6(i, p_{ovun7}, p_{ovun8}) \cdot \frac{1 - exp\{p_{ovun6} \cdot \Delta_i^{lpcorr}\}}{1 + exp\{-p_{ovun2} \cdot \Delta_i^{lpcorr}\}}$$
(2.8)

$$\Delta_i^{lpcorr} = \Delta_i - \Delta_i^{lp} \cdot f_6(i, p_{ovun3}, p_{ovun4})$$
$$f_6(i, p_1, p_2) = \left(1 + p_1 \cdot exp\left\{p_2 \cdot \left[\sum_{j \in nbrs(i)} \left(\Delta_j - \Delta_j^{lp}\right) \cdot \left(BO_{ij}^{\pi} + BO_{ij}^{\pi\pi}\right)\right]\right\}\right)^{-1}$$

.)

**2.5. Valence Angle Energy.** The energy associated with vibration about the optimum valence angle between atoms i, j, k is computed from Eq. (2.9):

$$E_{val} = f_7(BO_{ij}, p_{val3}, p_{val4}) \cdot f_7(BO_{jk}, p_{val3}, p_{val4}) \cdot f_8(\Delta_j, p_{val5}, p_{val6}, p_{val7}) \cdot \left(p_{val1} - p_{val1} \cdot exp\left\{-p_{val2} \cdot (\Theta_0 - \Theta_{ijk})^2\right\}\right)$$
(2.9)

$$f_7(BO, p_1, p_2) = 1 - exp \{-p_1 \cdot BO^{p_2}\}$$

$$f_8(\Delta, p_1, p_2, p_3) = p_1 - (p_1 - 1) \cdot f_9(\Delta, p_2, p_3)$$

$$f_9(\Delta, p_1, p_2) = \frac{2 + exp \{p_1 \cdot \Delta\}}{1 + exp \{p_1 \cdot \Delta\} + exp \{-p_2 \cdot \Delta\}}$$

Similar to its classical counterparts, the energy on  $\Theta_{ijk}$  increases as it moves away from its *corrected* optima  $\Theta_0$ , which is obtained from the *theoretical* optima  $\Theta_{00}$ , by accounting for the effects of over/under-coordination on the central atom j as well as the influence of any lone electron pairs. Valence angle energy further depends on the strength of bonds  $BO_{ij}$  and  $BO_{jk}$ ;  $f_7(BO_{ij})$  and  $f_7(BO_{jk})$  terms in Eq. (2.9) ensure that valence angle energy goes smoothly to zero as either bond dissociates.

**2.6.** Torsion Angle Energy. Eq. (2.10) accounts for the energy resulting from torsions in a molecule.

$$E_{tors} = \frac{1}{2} \cdot f_{10}(BO_{ij}, BO_{jk}, BO_{kl}, p_{tor2}, 1) \cdot sin\Theta_{ijk} \cdot sin\Theta_{jkl} \cdot$$

$$[V_1 \cdot (1 + cos\omega_{ijkl}) + V_2 \cdot exp \left\{ p_{tor1} \cdot \left(2 - BO_{jk}^{\pi} - f_9(\Delta_j + \Delta_k, p_{tor3}, p_{tor4})\right)^2 \right\} \cdot (1 - 2cos(2\omega_{ijkl})) + V_3 \cdot (1 + cos(3\omega_{ijkl}))]$$
(2.10)

$$f_{10}(BO_1, BO_2, BO_3, p_1, p_2) = f_7(BO_1, p_1, p_2) \cdot f_7(BO_2, p_1, p_2) \cdot f_7(BO_3, p_1, p_2)$$

As in the valence angle energy term, the torsional conribution from a four-body structure should vanish as any of its bonds dissociate. Here,  $f_{10}(BO_{ij}, BO_{jk}, BO_{kl}, p_{tor2}, 1)$  enforce this constraint. If either of the two valence angles defined by these four atoms approaches  $\pi$ , torsional energy should again disappear; this is accomplished by the term  $sin\Theta_{ijk} \cdot sin\Theta_{ikl}$ .

In ReaxFF, there are other bonded interaction terms shown in Eq. (2.4) for which we do not provide complete details. The stability of 3-body structures in which the central atom has two double bonds is achieved by adding the *penalty energy* term,  $E_{pen}$ . Three-body conjugation energy,  $E_{3conj}$ , and four-body conjugation energy,  $E_{4conj}$ , terms capture the energy contribution from conjugated systems. More details regarding these terms can be found in [9, 10]. This paper focuses on the algorithmic and numerical aspects of ReaxFF implementation. **2.7. Hydrogen Bond Energy.** The energy associated with a hydrogen bond in ReaxFF is given by:

$$E_{hbond} = p_{hb1} \cdot f_7(BO_{XH}, p_{hb2}, 1) \cdot \sin^4\left(\frac{\Theta_{XHZ}}{2}\right) \cdot exp\left\{-p_{hb3} \cdot \left(\frac{r_{hb}^0}{r_{HZ}} + \frac{r_{HZ}}{r_{hb}^0} - 2\right)\right\}$$
(2.11)

A hydrogen bond exists between an electronegative atom (denoted by Z in Eq. (2.11)) in the vicinity of a Hydrogen atom, covalently bonded to a Nitrogen, Oxygen or Fluorine atom (denoted by X). Similar to model for valence angle and torsion angle potentials, the  $f_7(BO_{XH}, p_{hb2}, 1)$  term ensures that contributions from hydrogen bonding smoothly disappear as the covalent bond breaks. For hydrogen bonding to be strong, it is crucial that all three atoms are aligned on a straight line. This is modeled by the term  $sin^4(\frac{\Theta_{XHZ}}{2})$ , which is maximized when  $\Theta_{XHZ} = \pi$ .

**2.8.** van der Waal's Interaction. A distance-corrected Morse-potential is used for *van der Waals* interactions, as shown in Eq. (2.12).

$$E_{vdWaals} = Tap(r_{ij}) \cdot D_{ij} \cdot$$

$$\left[ exp\left\{ \alpha_{ij} \cdot \left( 1 - \frac{f_{13}(r_{ij})}{r_{vdW}} \right) \right\} - 2 \cdot exp\left\{ \frac{1}{2} \cdot \alpha_{ij} \cdot \left( 1 - \frac{f_{13}(r_{ij})}{r_{vdW}} \right) \right\} \right]$$

$$Tap(r_{vi}) = Tap_{\overline{v}} \cdot r^{7} + Tap_{\overline{v}} \cdot r^{6} + Tap_{\overline{v}} \cdot r^{5} + Tap_{\overline{v}} \cdot r^{4}$$

$$(2.12)$$

$$Tap(r_{ij}) = Tap_7 \cdot r'_{ij} + Tap_6 \cdot r^3_{ij} + Tap_5 \cdot r^3_{ij} + Tap_4 \cdot r^4_{ij}$$

$$+ Tap_3 \cdot r^3_{ij} + Tap_2 \cdot r^2_{ij} + Tap_1 \cdot r_{ij} + Tap_0$$
(2.13)

$$f_{13}(r_{ij}) = \left(r_{ij}^{p_{vdW1}} + \gamma_w^{-p_{vdW1}}\right)^{\frac{1}{p_{vdW1}}}$$
(2.14)

Contrary to classical force fields where van der Waals interactions are computed only between non-bonded atom pairs, in ReaxFF all atom pairs, bonded or nonbonded, contribute to van der Waals energy. The reason for this is that exclusion of bonded pairs from van der Waals energy computation would result in discontinuties on the potential energy surface as bonds are formed or broken. To prevent extremely high repulsion forces between pairs at short distances, a shielding term is included, see Eq. (2.14). The Taper function in Eq. (2.13) ensures that the van der Waals energy smoothly goes to zero for pairs at distances beyond the non-bonded interaction cut-off distance,  $r_{nonb}$ .

**2.9. Coulomb Interaction.** Like van der Waals interactions, Coulomb interactions need to be computed between all atom pairs; shielding and Taper terms are included in the Coulomb potential as well (Eq. (2.15)).

$$E_{Coulomb} = C \cdot Tap(r_{ij}) \cdot \frac{q_i \cdot q_j}{\left[r_{ij}^3 + \gamma_{ij}^{-3}\right]^{\frac{1}{3}}}$$
(2.15)

All *Coulomb* interactions are confined within the  $r_{nonb}$  cut-off. There are no long-range electrostatic interactions in ReaxFF.

**2.10.** Charge Equilibration. Since bonding is dynamic in ReaxFF, charges on atoms cannot be fixed for the duration of simulations, as in most classical MD methods. Charge must be redistributed periodically (potentially at each time-step).

The most accurate way of doing this would be to employ *ab-inito* methods. However, this would render the ReaxFF method unscalable, therefore conflicting with our initial goal of building a highly scalable reactive force field. We resort to an alternate approximation to the charge equilibration problem called QEq [16].

The charge equilibration problem can be approximated as follows: find 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:

$$E(q_1 \dots q_N) = \sum_i \text{Atomic Energy of } i \text{ due to } q_i$$

$$+ \sum_{i < j} \text{Coulomb energy between } i \text{ and } j$$
(2.16)

where *i* and *j* denote atom indices and  $q_i$  denotes the partial charge on atom *i*. The coulomb interaction between atom pairs is relatively straighforward:  $\sum_{i < j} J_{ij} q_i q_j$ . The first summation in Eq. (2.16) requires more explanation, though. Charge dependency of an isolated atom's energy can be written as:

$$E_i(q) = E_{i0} + q_i \left(\frac{\partial E}{\partial q}\right)_{i0} + \frac{1}{2}q_i^2 \left(\frac{\partial^2 E}{\partial q^2}\right)_{i0} + \dots$$
(2.17)

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 Eq. (2.17), we can write:

$$E_i(0) = E_{i0} (2.18)$$

$$E_i(+1) = E_{i0} + \left(\frac{\partial E}{\partial q}\right)_{i0} + \frac{1}{2} \left(\frac{\partial^2 E}{\partial q^2}\right)_{i0} = E_{i0} + IP$$
(2.19)

$$E_i(-1) = E_{i0} - \left(\frac{\partial E}{\partial q}\right)_{i0} + \frac{1}{2} \left(\frac{\partial^2 E}{\partial q^2}\right)_{i0} = E_{i0} - EA$$
(2.20)

Solving for the unknowns in Eq. (2.19) and Eq. 2.20 gives:

$$\left(\frac{\partial E}{\partial q}\right)_{i0} = \frac{1}{2}(IP + EA) = \chi_i^0 \tag{2.21}$$

$$\left(\frac{\partial^2 E}{\partial q^2}\right)_{i0} = IP - EA = J_{ii}^0 \tag{2.22}$$

Here,  $\chi_i^0$  is referred to as the *electronegativity* and  $J_{ii}^0$  as the *idempotential* or *self-Coulomb* of *i*. We will revisit the computational solution of this QEq problem when we discuss algorithmic aspects of our implementation in Section 3. For now, we just restate the problem more formally in the light of the discussions above:

Minimize 
$$E(q_1 \dots q_N) = \sum_i (E_{i0} + \chi_i^0 q_i + \frac{1}{2} J_{ii}^0 q_i^2) + \sum_{i < j} (J_{ij} q_i q_j)$$
  
(2.23)  
subject to  $q_{net} = \sum_{i=1}^N q_i$ 

**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. The key components of the algorithm include: generate neighbors, compute energy and forces, move atoms under the effect of net forces, and repeat until the desired number of steps are reached. Each of these components, however, is considerably more complex in ReaxFF because of dynamic bonding and charge equilibration. 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 <i>nsteps</i> 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-timestep simulation time 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 almost 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 g/cm<sup>3</sup>.

**3.1.** Neighbor Generation. In ReaxFF, both bonded and non-bonded interactions are truncated after a cut-off distance (which is typically 4-5 Åfor bonded interactions and 10-12 Åfor non-bonded interactions). Given this truncated nature of interactions, we use a procedure based on "binning" (or link-cell method) [14]. First, a 3D 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  $r_{nbrs}$  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  $(3r_{nbrs})^3$  per atom. Setting the cell dimensions to be  $\frac{1}{2}r_{nbrs}$ , further decreases the search space to  $(\frac{5}{2}r_{nbrs})^3$ , 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. An empirical study using our ReaxFF implementation on the bulk water system described above reveals that the lowest neighbor list generation time is achieved when the cell size is half of  $r_{nbrs}$  (Tab. 3.1).

#### Table 3.1

Comparison of different  $r_{nbrs}$  to cell size ratios in terms of the time and memory required to generate neighbors in our code. For benchmarking, we have performed an NVT (constant N, volume, and temperature) simulation of a bulk water system at 300 K. Values below are the average neighbor generation times per step measured in seconds averaged over 100 steps. Memory usage reported is the space required by the 3D grid structure only.

$r_{nbrs}$ /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

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 further than  $r_{nbrs}$ , it is evident that none the atoms in the cell are potential neighbors. For the experiment described in Tab. 3.1, when  $r_{nbrs}$  to cell size ratio is set to 2, neighbor generation takes 0.14 seconds on average without this optimization (as opposed to 0.11 seconds with it) reflecting an improvement over 20%.

*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 we look up the position information of an atom, the 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 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 only needs to update its distance only to the atoms in its Verlet list and to determine its neighbors from this list. We note that neighbor generations are 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 more memory lookups. This situation may incur additional overhead due to iterations over a larger list for force computations as well. In our ReaxFF implementation, we observe that gains from delayed reneighboring strategy are only marginal, most likely because of the effectiveness of other optimizations in the code.

Algorithm 2 Neighbor lists

```
r_{nbrs} \leftarrow r_{nonb} + r_{buf}
grid \leftarrow \text{Setup}_Grid(simulation\_box, atom\_list})
Bin_Atoms( grid, atom_list )
num\_nbrs \leftarrow \text{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 cell<sup>2</sup> in cell1_{nbrs} do
           d_{cell2} \leftarrow \text{distance of } atom_i \text{ to closest point of } cell2
           if d_{cell2} \ll r_{nbrs} then
              for all atom_i in cell2 do
                d_{ij} \leftarrow |x_i - x_j|
                if d_{ij} \ll r_{nbrs} then
                   nbr\_list_i \leftarrow j
                 end if
              end for
           end if
        end for
     end for
  end for
else
  for all i, j in nbr_{list} do
     d_{ij} \leftarrow |x_i - x_j|
  end for
end if
```

Our neighbor list generation routine is given in Alg. 2 with all of the optimizations. The neighbor list is the most memory consuming data-structure. Therefore, we store it as a half-list, to reduce the overall memory footprint. The neighbor list size is dynamically controlled to optimize the memory usage according to the constantly changing geometry of the simulated system. Our dynamic memory management scheme is desribed in detail in Section 4.

**3.2. Bonded and Non-bonded Force Computations.** The dynamic bonding and charge equilibration in reactive models add to the complexity of force computation, both algorithmically and performance-wise. In Alg. 3, we outline the key components of force computations in ReaxFF. In the following subsections, we discuss algorithms and techniques used to achieve excellent performance for force computations in ReaxFF.

Algorithm 3 Computation of energy and forces in a reactive force field.	
Compute bonds	
Compute bonded forces	
Update charges	
Compute non-bonded forces	

**3.2.1. Eliminating bond order derivative lists.** As described in Section 2, all bonded potentials (including even the hydrogen bond potential) primarily depend on the strength of the bonds between the atoms it involves. Therefore all forces arising from bonded interactions will depend on the derivative of the bond order terms.

A close examination of Eq. (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 non-zero 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, we may need to evaluate the expression  $dBO_{ij}/dr_k$  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 timestep 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 going to physical memory by delaying the computation of the derivative of bond orders until the end of a timestep. During the computation of bonded potentials, we accumulate the coefficients for the corresponding bond order derivative terms arising from various interactions into a scalar variable  $CdBO_{ij}$ . In the final stage of a timestep, 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.

$$(c_1 \times \frac{dBO_{ij}}{dr_k} + c_2 \times \frac{dBO_{ij}}{dr_k} + \dots + c_n \times \frac{dBO_{ij}}{dr_k}) = (c_1 + c_2 + c_n) \times \frac{dBO_{ij}}{dr_k} = CdBO_{ij} \times \frac{dBO_{ij}}{dr_k}$$

This simple technique 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.

**3.2.2.** Lookup tables for non-bonded interactions. In general, computing non-bonded forces is more expensive than computing bonded forces, due to the larger number of interactions within the longer cut-off radii associated with non-bonded interactions. In ReaxFF, the formulations of non-bonded interactions are more complex compared to their classical counterparts. These two factors increase the fraction of time required to compute non-bonded 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 [22]. In sPuReMD,

## 14 H.M. AKTULGA AND S.A. PANDIT AND A.C.T. VAN DUIN AND A.Y. GRAMA

we use a cubic spline interpolation to achieve accurate approximations of non-bonded energies and forces, while using a compact lookup table. Performance gains and additional memory required by this technique are summarized in Tab. 3.2.2. It is evident that approximating non-bonded forces with interpolation gives significant performance improvements, as much as eight times, when we look at only the time to compute non-bonded interactions or up to a factor of three in the overall running time.

#### TABLE 3.2

Sample bulk water system. Effect of approximating non-bonded energy and forces through cubic spline interpolations. Size of the lookup table (in MB) is also shown. Last column gives the RMS deviation of net forces due to interpolation.

# of splines	total (s)	init (s)	nonb (s)	table size (MB)	$\mathbf{rms}(f_{net})$
1000	0.53	0.15	0.09	1.2	$2.2 \times 10^{-8}$
10000	0.56	0.16	0.11	12	$2.1 \times 10^{-12}$
50000	0.62	0.18	0.15	61	$6.9 \times 10^{-15}$
none	1.80	0.31	1.20	0	0

The memory size of the interpolation table is directly proportional to the number of splines used and the number of different element pairs in the system to be simulated. This is clearly reflected in the memory requirements of RuReMD, working on the same system with different number of splines used for interpolation. Tab. 3.2.2 shows that as we increase the number of splines from 1000 to 50,000, the size of the lookup table increases from 1.2 MB to 61 MB. Not surprisingly, besides the memory overhead, increasing the number of splines adversely affects the performance of force computations as well. Due to the nature of our neighbor generation routine, neighbors of an atom are roughly clustered by their distances to that atom (since we handle the search space cell by cell). A small lookup table allows sPuReMD to make good use of the cache, however, increasing the number of splines adversely impacts cache usage and overall performance.

An important implementation choice corresponds to the size of the lookup table. To address this, we compute the root-mean-squared (RMS) deviation of net forces from their actual values in Tab. 3.2.2. Even when we use only 1000 splines, the RMS deviation of net forces is on the order of  $10^{-8}$ . This is well within required tolerance for MD simulations [15].

**3.3.** A Fast ILU preconditioning-based solver for the charge equilibration problem. One of the most compute-intensive parts of a ReaxFF simulation is the procedure for (re)assining partial charges to atoms at each timestep. This component does not exist in conventional MD formulations, since they rely on static charges on atoms. The charge reassignment problem is formulated as charge equilibration, with the objective of minimizing electrostatic energy.

In this section, we first briefly desribe the mathematical formulation of the charge equilibration problem. We use various aspects of the formulation to motivate our choice of a Krylov subspace solver with an ILU preconditioner. We investigate various performance and stability aspects of our solver, and validate its accuracy on model systems. We use two physical systems 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<sup>3</sup> box, and

a PETN crystal with 48,256 atoms again inside an orthogonal box with dimensions  $570 \times 38 \times 28$  Å<sup>3</sup>.

**3.3.1. Mathematical formulation.** We extend the mathematical formulation of the charge equilibration problem by Nakano et al. [17]. 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.  $\chi$  is a vector of parameters determined based on the types of atoms in the system and it has a size of N. 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 fictituous charges based on the following formula:

$$q_i = s_i - \frac{\sum_{i=1}^{i} s_i}{\sum_{i=1}^{i} t_i} t_i \tag{3.3}$$

The linear systems in Eq. (3.1) and Eq. (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. H is a sparse linear system and we can use well-known Krylov subspace methods such as CG [18] and GMRES [19] to solve the linear systems in Eq. (3.1) and Eq. (3.2). The sparsity of the coefficient matrix results from the fact that independent of system size, we use neighboring atom information only within the non-bonded interaction cut-off distance  $r_{nonb}$ . Even though  $r_{nonb}$ , and the number of atoms that can be found within the cut-off radius change from system to system, this number is bounded (typically on the order of a few hundred atoms). Therefore the number of non-zeros in H will be on the order of a few hundred entries per row independent of the system size.

Based on our observations on different molecular systems, H carries a heavy diagonal and this motivates diagonal scaling as an effective accelerator for the solver. Individual timesteps in reactive MD simulations are also much shorter than those in conventional MD (less than a femtosecond in *ReaxFF*). Therefore the system does not change drastically between consecutive timesteps. This observation implies that solutions to Eq. (3.1) and Eq. (3.2) in one timestep yield good initial guesses about the solutions the next timestep. 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 at the current timestep.

## 16 H.M. AKTULGA AND S.A. PANDIT AND A.C.T. VAN DUIN AND A.Y. GRAMA

One may derive a simple solver based on the observations above: A *GMRES* or *CG* solver with a diagonal preconditioner, which extrapolates from the solutions of previous timesteps to make a good initial guess. Tab. 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  $(10^{-6}$  is generally a satisfactory value). In Tab. 3.3, we set the QEq tolerance to  $10^{-6}$  and use linear extrapolation for both systems.

Performance of basic approaches for bulk water and bilayer systems. QEq tolerance is set to  $10^{-6}$ , which suffices for most applications. Extrapolation scheme is linear extrapolation.

system	solver	matvecs	$\mathbf{QEq}\ (\mathbf{s})$	<b>QEq</b> (%)
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%

In Tab. 3.4, we use a much lower tolerance level  $(10^{-10})$  to observe the performance of these basic solvers, when a high accuracy solution is desired. Results in Tab. 3.3 and 3.4 suggest that GMRES is a better solver for the QEq problem compared to CG. However, QEq still takes up to 47% of total computation time, even with the GMRES solver at a modest  $10^{-6}$  tolerance level. This motivates the use of more powerful solvers/ preconditioners. We address this issue in the remainder of this section.

TABLE 3.4 Performance of basic solvers for bulk water and bilayer systems. QEq tolerance is set to  $10^{-10}$  to observe performance at low tolerance.

system	solver	matvecs	QEq~(s)	<b>QEq</b> (%)
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%

**3.3.3. ILU-based preconditioning.** Preconditioning techniques based on incomplete LU factorization (ILU) are effective and widely used for solving sparse linear systems [20]. ILU-based preconditioners help in reducing the iteration counts reported in Tab. 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. However, the observation regarding the use of the solutions from previous timesteps 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, the same factors L and U can be used effectively as preconditioners over several steps.

**ILU factorization.** In sPuReMD, we use factors from the ILU factorization of the H matrix with zero fill-in and a threshold. Zero fill-in means that we drop values in the L and U matrices that correspond to zero-entries in the H matrix. The threshold used in the ILU factorization ensures that all entries in the L and U factors

with values less than a specified threshold are dropped. In ILU factorization with a threshold (ILUT), the smaller the threshold, the better the preconditioners. However, factorization takes considerably longer.

Tests on our sample bulk water system reveal that the QEq solver's performance is not very sensitive to the threshold for ILU factorization; different thresholds chosen from a large range (0.03, 0.01, 0.005 and 0.001) exhibit comparable performance. When the threshold value is on the high side, factorization is relatively cheap but the preconditioners are not as effective and iteration counts increase slightly. As the threshold value is decreased, ILU factorization and applying the resulting factors as preconditioners become more expensive, however, we observe improvements in iteration counts compensating for the more expensive factorization and preconditioning steps. Based on this empirical evidence, we fix the ILUT threshold at 0.01 for the experiments presented in this section.

Longevity of the ILU-based preconditioners. Experiments on the longevity of ILU-based preconditioners suggests that 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.

In Fig. 3.1, we show how the characteristics of the system affect the longevity of ILU-based preconditioners. 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. We compute the ILU-based preconditioner at the start of each simulation and use the same preconditioner over the lifetime of the simulation. As can be seen, for the PETN system, we can use the same preconditioner over a few picoseconds (several thousands of steps) without a significant increase in the number of iterations (and time) required by the PGMRES or PCG solvers. On the other hand, atoms in bulk water tend to have a higher displacement rate. Therefore ILU-based preconditioners lose their effectiveness much quicker. Even in this case, the number of steps that ILU-based preconditioners proves to be effective is on the order of a few hundred steps – long enough to amortize the ILU factorization cost.

We also examine how the desired accuracy of the solution affects the longevity of ILU-based preconditioners. In Fig. 3.2, we show the immediate effects of decreasing the QEq tolerance on the bulk water system: increased number of iterations, 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 ILU preconditioners are still quite effective for long durations (0.5 ps). However, even for the PETN crystal simulated at the QEq tolerance of  $10^{-10}$ , there is a quicker increase in the number of iterations of the QEq solver compared to the simulations at  $10^{-6}$  tolerance level.

One last observation can be made regarding the linear solvers we use, PGMRES vs. PCG. Besides delivering a better performance than PCG, PGMRES holds the edge in the longevity of ILU-preconditioners as well. For these reasons, PGMRES is supported as the default solver in sPuReMD.

**Performance gain with ILU-based preconditioners**. Even when the longevity of ILU-based preconditioners is on the order of tens of steps (which actually is the case for the bulk water system with QEq threshold at  $10^{-10}$ ), this is long enough to amortize the cost of the ILU factorization step. The significant improvement in the number of iterations and computation times result in impressive overall performance.



48256 atom petn crystal - QEq tolerance = 1e-6

FIG. 3.1. Longevity of the ILU-based preconditioners for systems with different characteristics. At the top a bulk water simulation is shown and at the bottom a PETN crystal simulation.

0.25

time (ps)

0.3

0.35

0.4

0.45

0.5

0.2

10

5

0

0

0.05

0.1

0.15

In Tab. 3.5, we show the performance of QEq solvers with ILU-based preconditioners for the bulk water system at  $10^{-6}$  and  $10^{-10}$  tolerance levels. Compared to the corresponding values in Tab. 3.3 and 3.4, QEq solvers with ILU-based preconditioners deliver three to four times better performance than the basic solvers. Even when we consider the case of  $10^{-10}$  tolerance level, the overall fraction of QEq computations in total computation time is only 15%-18%. This implies that the (re)assignment of partial charges at high accuracies can be accomplished without a significant degradation in the overall performance.

In Tab. 3.6, we present the performance of QEq solvers with ILU-based precondi-



48256 atom petn crystal - QEq tolerance = 1e-10

FIG. 3.2. Longevity of the ILU-based preconditioners at very low QEq tolerances. At the top a bulk water simulation is shown and at the bottom a PETN crystal simulation.

0.025

time (ps)

0.03

0.035

0.04

0.045

0.05

0.02

5

0

0

0.005

0.01

0.015

tioners for the lipid bilayer system at  $10^{-6}$  and  $10^{-10}$  tolerance levels. Although the number of iterations required by QEq solvers at  $10^{-6}$  tolerance level are the same as those in the bulk water system, the share of QEq in total computational time is much higher for the bilayer simulation (23-31% vs. 6-9%). This is due to less effective use of cache during the matrix-vector multiplication in each iteration. As the tolerance is decreased to  $10^{-10}$ , the share of QEq increases even further to 53-57%, because of the increase in the number of matrix-vector products. Optimizations for improved cache performance are currently being implemented in sPuReMD to improve this fraction.

TABLE	3.5
TUDDD	0.0

Performance of ILU-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.

qeq_tol	solver	matvecs	QEq (s)	<b>QEq</b> (%)
$t_{0} = 10^{-6}$	$CG + ILUT(10^{-2})$	9	0.06	9%
<b>tol</b> =10	$GMRES(50) + ILUT(10^{-2})$	6	0.04	6%
$t_{0} = 10^{-10}$	$CG + ILUT(10^{-2})$	18	0.13	18%
101=10	$GMRES(50) + ILUT(10^{-2})$	15	0.11	15%

TABLE	3	6
LADLL	ം.	.υ

Performance of ILU-based preconditioners for the bilayer system averaged over 100 steps. Refactorization frequency was 100 and 50 for the  $10^{-6}$  and  $10^{-10}$  QEq tolerance levels, respectively.

$qeq_tol$	solver	matvecs	QEq (s)	<b>QEq</b> (%)
$t_{0} = 10^{-6}$	$CG+ILUT(10^{-2})$	9	2.39	31%
<b>LOI</b> =10	$GMRES(50) + ILUT(10^{-2})$	6	1.58	23%
$t_{0} = 10^{-10}$	$CG+ILUT(10^{-2})$	27	6.93	57%
101=10	$GMRES(50) + ILUT(10^{-2})$	23	5.96	53%

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, 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, and four-body interactions, together with significant amount of book-keeping required for these interactions necessitate larger memory requirement and more sophisticated procedures for managing memory. The second important aspect is the adaptation of data-structures to 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 datastructures 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 mechanisms for each of them.

4.1. Dynamic data-structures. A number of dynamic data structures are used in sPuReMD to ensure high access rate and low footprint.

**4.1.1.** Neighbors list. The neighbor list in sPuReMD is similar to its counterpart in conventional MD implementations. We represent the adjacency matrix in CSR (compressed sparse row) format and keep only the upper half of the adjacency matrix to reduce its memory requirement by half. Keeping only the upper half of the adjacency matrix does not degrade the performance of sPuReMD, due to our optimized access technique. In fact, its compactness improves overall performance by reducing the memory region over which accesses are iterated.

Neighbor list is used in the computation of non-bonded interactions as well as the construction of all other lists described below. A neighbor list storage unit contains the atom list index of the neighbor, the distance and the distance vector between the pair, and the imaginary box information for keeping track of neighbors through periodic boundaries.

**4.1.2. QEq matrix.** We represent the QEq matrix as a list separate from the neighbor list for two reasons. First, as discussed in Section 3.1, with the use of delayed re-neighboring the cut-off for the neighbors list,  $r_{nbrs}$ , can be larger than the cut-off for the QEq matrix,  $r_{nonb}$ . Keeping a separate list ensures efficient usage of memory. More importantly, matrix-vector products during QEq require several passes over the coefficient matrix. Keeping a separate list for the QEq matrix, where each unit is compactly packed with the column information and the matrix entry only, requires access to a much smaller memory region. This ensures better usage of the cache due to spatial locality.

We only store the upper half of the QEq matrix, also in CSR format. At the start of each timestep, the QEq matrix, together with the bonds and hydrogen bonds lists are constructed in a single pass over the neighbor list.

**4.1.3.** Bonds lists. In ReaxFF, bonds need to be identified and their strengths need to be calculated at each step. We use the neighbor list described above to identify potential bonds. However, as discussed in Section 2.1 calculating the actual bond order between two atoms is not a pair-wise interaction, it requires full knowledge of all other potential bonds of these two atoms. Furthermore, higher order interactions such three-body, four-body, and multi-body terms also change every timestep. Constructing these interactions and computing forces due to them require full knowledge of bonds incident on all interacting atoms. For these reasons, we maintain the bond list as a full list in a *modified* CSR format. Even though a bond order storage unit is a large structure with all the quantities required for force computations, our use of a full list is motivated by the need to improve the performance of force computations for these interactions.

We call the format of our bond list, modified CSR format, because the space reserved for each atom in the list is contiguous, but the actual data stored is not. This is required because the neighbors list is a half list and we need to be able to access the memory reserved for the bonds of both i and j while processing their neighbors information. sPuReMD starts by estimating the number of bonds of each atom based on the neighbor list at the start of the simulation. The estimation task is carried out before any force computations are performed. Let  $eb_i$  denote the number of estimated bonds for atom i. Then  $max(2eb_i, MIN\_BONDS)$  slots are reserved for atom i in the bond list. This conservative allocation scheme prevents any overwrites in subsequent steps and reduces the frequency of bond list reallocations (this is discussed in more detail in Section 4.2) throughout the simulation.

**4.1.4. Hydrogen bonds list.** In sPuReMD, due to performance and space considerations, we maintain the neighbor list as a half list, as mentioned before. This means that neighbors of a hydrogen atom within the hydrogen bonding cut-off  $r_{hbond}$  are spread all over the neighbor list. To make hydrogen bonding computations easier, we maintain a separate list in which we collect hydrogen bonding pairs together, using the same *modified* CSR format as the bonds list.

In hydrogen bonds list, for each hydrogen atom, we only store its neighbors belonging to certain chemical species that can establish hydrogen bonds and fall within the  $r_{hbond}$  cut-off. To save memory, instead of replicating the distance, distance vector, and imaginary box information already stored in the neighbors list, in a hydrogen bonds list unit we store a pointer to the corresponding neighbors list storage unit and a flag that describes the neighbors list entry as an H-X or X-H pair (where X denotes the acceptor in a hydrgon bond).

**4.1.5.** Three-body interactions list. Three-body interactions in sPuReMD are constructed from the bonds list. For each atom j, we iterate over its bond list (which is a full list) to locate all pairs i and k such that the angle term i, j, k satisfies certain criteria to be included in three-body force computations. We store the information about all such three-body structures for use while building the four-body structures. We store the three-body structures in CSR format indexed not by the atoms but by the bonds in the system.

**4.1.6. Four-body interactions.** Four-body structures are constructed in a manner similar to three-body structures. We bring together two three-body structures i, j, k and j, k, l to see whether they satisfy certain criteria for a four-body interaction. 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.

4.2. Reallocation Mechanism. In sPuReMD, the reallocation mechanism ensures that we retain a small memory footprint and adapt our data-structures to the changing simulation needs. The general mechanism consists of three parts: estimation, monitoring, and reallocation. At the start of simulation, sPuReMD runs a number of estimation routines before storing data. Actual neighbor generation and force computations are executed after all the lists are initialized based on their estimated sizes. Throughout the simulation, memory utilization in each list is monitored. When utilization within a list reaches above a high threshold or below a low threshold, the list is reallocated. To avoid overheads associated with reallocations (such as copying stored data), we ensure that the reallocation deamon kicks in when none of the lists contain important data, thus reducing the cost of a list reallocation to a deallocate/allocate call.

5. Validation. We validate our algorithms and implementation comprehensively on a well-studied class of materials, hydrocarbons. Hydrocarbons are mostly used as combustable 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, therefore considerable experimental data is available for validation, (ii) the diversity of interactions in hydrocarbons stress all parts of the code, and underlying algorithms, (iii) due to reactive aspects of typical applications, conventional MD methods are not feasible, and (iv) systems of various sizes can be prepared relatively easily. We choose to work with a *hexane*,  $C_6H_{14}$  system in liquid form for validation.

**5.1. Preparation of systems.** Initial configurations are prepared by randomly placing the desired number of hexane backbone chains inside a large box to minimize overlaps among hexane molecules. Initial configurations are first energy minimized and then run under the NPT ensemble using GROMACS [22] to quickly bring them to equilibrium under 1 atm pressure and 200 K temperature. Systems of various sizes (343, 512, 1000, 1728, 3375 molecules) are prepared in this manner for the scalability analysis. Results presented in this section are derived from the hexane<sub>343</sub> system – 343 hexane molecules (6860 atoms) inside an orthogonal box with periodic boundary conditions.

5.2. ReaxFF simulations. Configurations equilibrated using GROMACS are used for our ReaxFF simulations. The force field for hexane simulations in GRO-

MACS does not treat H atoms separately, they are modeled as parts of super-atomic 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 sub-femtoseconds, the simulation time-steps can be stretched to a few femtoseconds.

In ReaxFF, all atoms must be modeled separately to correctly capture reactions. We use the Avogadro program [23] 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 only, 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 suggest that sPuReMD produces hexane structures that are in near-perfect agreement with experimental results in [21] and geometry optimizations based on ab-initio methods (CPMD software [24] using PBE Troullier-Martins pseudopotentials). These results are presented in Tab. 5.1.

			'I'ABL	le 5.1				
Structural	properties	of hexan	e molecules	obtained	through	different	simulation	methods
(ReaxFF, a rea	active force	$field, \ and$	$CPMD, \ an$	$ab\mathchar`-initio$	MD meth	nod) comp	ared to expe	erimenta
results.								

property	ReaxFF	ab-initio	experimental [21]
C-H bond	$1.09\pm0.01$	1.100	$1.118\pm0.006$
C-C bond	$1.57\pm0.01$	1.533	$1.533 \pm 0.003$
<c-c-c< td=""><td><math display="block">108.0\pm2.9</math></td><td>114.2</td><td><math display="block">111.9\pm0.4</math></td></c-c-c<>	$108.0\pm2.9$	114.2	$111.9\pm0.4$
<c-c-h< td=""><td><math display="block">111.0\pm0.0</math></td><td>109.5</td><td><math display="block">109.5\pm0.5</math></td></c-c-h<>	$111.0\pm0.0$	109.5	$109.5\pm0.5$
<h-c-h< td=""><td><math display="block">106.6\pm0.0</math></td><td>106.5</td><td>—</td></h-c-h<>	$106.6\pm0.0$	106.5	—
$q_{\rm C-tip}$	-0.171	-0.205	—
$q_{\rm C-mid}$	-0.080	0.033	_
$q_{ m H-tip}$	0.040	0.047	—
$q_{ m H-mid}$	0.040	$-0.10\sim 0.10$	_

6. Performance Analysis. We compare the performance of our ReaxFF implementation, sPuReMD, to other MD codes. First, 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 then compare the performance of sPuReMD to the state of the art ReaxFF implementation available over the public domain, the REAX package [26] inside LAMMPS [14, 28].

**6.1. Comparison with other MD methods.** We examine the time-scales and system sizes where different molecular simulation methods, namely classical MD, ReaxFF, and ab-initio methods, are applicable. We use GROMACS as a representative of classical MD methods, sPuReMD as a reactive MD method, and CPMD as an ab-initio method. For this purpose, we prepare various hexane systems of different sizes (343, 512, 1000, 1728, 3375 molecules) as described in Section 5 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 overheads. 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 closeby molecules from the large simulation boxes used with sPuReMD. Due to the small size of CPMD systems, we drop the periodic boundary conditions in CPMD simulations and use the PBE approximation to DFT and corresponding Trouiller-Martins norm-conserving pseudo-potentials with a wave-function cut-off of 75Rydberg for both C and H atoms.



FIG. 6.1. 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).

Fig. 6.1 shows scaling of the run-time per time-step and total memory requirements of various molecular simulation methods. All simulations are performed under the micro-canonical ensemble (NVE), and reported run-times are averaged values over several hundred steps. Reported memory requirements are the peak values observed during the life time of 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 Fig. 6.1. Growing box size causes a similar growth in memory requirements of CPMD simulations, as well.

sPuReMD and GROMACS, on the other hand, 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 non-bonded pairs, decreasing the number of such pairs by more than an order of magnitude. Considering that non-bonded force computations takes about 90% of the total time in a typical classical MD simulations [25] and that most of the allocated space is used for storing non-bonded pairs, the run-time per time-step and memory footprint of GROMACS simulations would approach those of sPuReMD, had H atoms been modeled explicitly. There is the additional burden of computing bonds, 3-body, and 4-body structures dynamically 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 Fig. 6.1 is the jump in sPuReMD's run-time per timestep between hexane<sub>512</sub> and hexane<sub>1000</sub>. To shed more light on the reasons for this super-linear 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 3-body and 4-body structures in the system.
- ilu: the part where we compute the ILU-based preconditioners for the QEq matrix.
- **QEq:** the 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 Tab. 6.1, we show how the run-time for each of these parts (except for **ilu**) 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. However, due to cache effects, as we move beyond hexane<sub>512</sub> system (10240 atoms), the matrix-vector product becomes costlier and causes the jump mentioned above. This is reflected as the increased share

of  $\mathbf{QEq}$  within the total run-time per time-step beyond hexane<sub>512</sub>.

TABLE 6.1

Break-down 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 <sub>343</sub>	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

ILU factorization is not performed at every step, to amortize its cost as discussed in Section 3.3. Therefore we do not explicitly present **ilu** running times in Tab. 6.1. For all sPuReMD experiments reported in this Section, the threshold for the QEq solver was set to  $10^{-6}$  and we perform the ILU factorizations once every 100 steps. The ILU-based preconditioning scheme described in Section 3.3.3 takes only 5-6 iterations per step using PGMRES algorithm to solve the QEq problem. The cost of ILU factorization is about 0.30 s for the hexane<sub>343</sub> system and 2.80 s for the hexane<sub>3375</sub> system, which is negligible when averaged over 100 steps. These figures also suggest that **ilu** scales nicely with the system size.

**6.2.** Comparison with existing implementations. The first-generation ReaxFF implementation of van Duin et al. [9] demonstrated the validity of the force-field in the context of various applications. Thompson et al. [26] successfully ported this initial implementation, which was not developed for a parallel environment, into their parallel MD package LAMMPS [14]. Except for the charge equilibration part, the ReaxFF implementation in LAMMPS is based on the original FORTRAN code of van Duin [9], significant portions of which were included directly (as Fortran routines called from C++) to ensure consistency between the two codes.

TABLE 6.2

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 speed-up achieved by sPuReMD over LAMMPS.

system	#atoms	LAMMPS (s)	sPuReMD (s)	speed-up
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

In Tab. 6.2, we compare the performance of sPuReMD to the LAMMPS code for validation systems described above. For each system, we report the average run-time per step for both codes . Our results show that sPuReMD achieves a six to seven fold speed-up over LAMMPS, due to various algorithmic and numerical enhancements presented in this paper. Note that LAMMPS is a parallel MD simulation program, therefore it contains some potential overheads due to communications. On a single processor these overheads are not significant – at the end of each simulation, communication time reported by LAMMPS accounts for less than 1% of the total time.

#### Table 6.3

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 step by the QEq solvers in LAMMPS and sPuReMD.

	LAN	MMPS	sPuReMD		
system	QEq (s)	QEq iters	QEq (s)	QEq iters	
PETN crystal	0.19	32	0.02	5	
bulk water	0.38	34	0.04	6	
$hexane_{343}$	0.40	35	0.04	6	

Among various optimizations, our ILU-based preconditioning scheme for solving the QEq problem stands out. In Tab. 6.3, 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 [27]. Compared to this, our advanced QEq solver delivers about an order of magnitude improvement in performance.

TABLE 6.4 Comparison of the memory foot-prints of LAMMPS and sPuReMD codes on different systems. For each system, we present the amount of memory required by LAMMPS and sPuReMD in MBs.

$\mathbf{system}$	LAMMPS (MB)	sPuReMD (MB)
PETN crystal	2500	170
bulk water	2500	250
$hexane_{343}$	2500	270

Finally, in Tab. 6.4 we compare the memory foot-print of both codes. Since the LAMMPS implementation uses static allocation for atom, neighbor, and interaction lists, it exhibits a constant memory foot-print for all simulations. However, sPuReMD is able to achieve a considerably smaller memory foot-print owing to its intelligent memory management scheme. This scheme allows us to work with systems much larger sizes without having to tune compilation parameters.

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 is: (i) 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 nano-scale 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).

Acknowledgments. The authors HMA, SAP, AYG would like to thank the National Science Foundation and the US Department of Energy for their support of this work. ACTvD acknowledges funding from KISK startup grant C000032472. Special thanks are also due to Aidan Thompton and Steve Plimpton (Sandia), whose work on LAMMPS motivated many of our optimizations.

### REFERENCES

- A.C.T. van Duin, J.M.A. Baas, 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, 2881-2895, 1994.
- [2] T.A. Halgren, W. Damm, *Polarizable Force Fields*, Current Opinion in Structural Biology, 11, 236-242, 2001.
- J.E. Davis, G.L. Warren, S. Patel, Revised Charge Equilibration Potential for Liquid Alkanes, J Phys Chem B, 112, 8298-8310, 2008.
- [4] .C. Abell. Empirical chemical pseudopotential theory of molecular and metallic bonding., Phys Rev B, 31, 6184-6196, 1985.
- [5] J. Tersoff. Empirical interatomic potential for carbon, with applications to amorphous carbon, Phys Rev Lett, 61, 2879-2882, 1988.
- [6] D.W. Brenner, Empirical potential for hydrocarbons for use in simulating the chemical vapor deposition of diamond films, Phys Rev B, 42, 9458-9471, 1990.
- [7] D.W. Brenner, O.A. Shenderova, J.A. Harrison, S.J. Stuart, S.B. Sinnott, A second-generation reactive empirical bond order (REBO) potential energy expression for hydrocarbons, J Phys Condens Matter, 14, 783802, 2002.
- [8] S.J. Stuart, A.B. Tutein, J.A. Harrison, A reactive potential for hydrocarbons with intermolecular interactions, J Chem Phys, 112, 6472, 2000.
- [9] A.C.T. van Duin, S. Dasgupta, F. Lorant, W.A. Goddard III, ReaxFF: A Reactive Force Field for Hydrocarbons, J Phys Chem A, 105, 9396-9409, 2001.
- [10] A.C.T. van Duin, A. Strachan, S. Stewman, Q. Zhang, X. Xu, W.A. Goddard III. ReaxFFSiO Reactive Force Field for Silicon and Silicon Oxide Systems, J Phys Chem A, 107, 3803-3811, 2003.
- [11] R.A. Friesner. Ab initio quantum chemistry: Methodolgy and applications, PNAS, 102, 6648-6653, 2005.
- [12] J.M. Thijssen. Computational Physics, Cambridge University Press, 2003.
- [13] A. Erdemir, J.M. Martin. *Superlubricity*, Elsevier, 2007.
- [14] S.J. Plimpton, Fast Parallel Algorithms for Short-Range Molecular Dynamics, J Comp Phys, 117, 1-19, 1995.
- [15] R. Zhou, E. Harder, H. Xu, B.J. Berne. Efficient multiple time step method for use with Ewald and particle mesh Ewals for large biomolecular systems, J Chem Phys, 115(5), 2348-2358, 2001.
- [16] A.K. Rappe, W.A. Goddard III, Charge equilibration for molecular dynamics simulations, J Phys Chem, 95, 33583363, 1991.
- [17] Aiichiro Nakano, "Parallel multilevel preconditioned conjugate-gradient approach to variablecharge molecular dynamics", Comp Phys Comm 104, 59-69, 1997.
- [18] J.R. Shewchuk, An introduction to the conjugate gradient method without the agonizing pain, School of Computer Science, Carnegie Mellon University, Tech Report, August, 1994.
- [19] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual method for solving nonsymmetric linear systems, SIAM J Sci Stat Comput,7, 856-869, 1986.
- [20] Y. Saad, Iterative methods for sparse linear systems, 2nd Edition, SIAM, 2003.
- [21] R.A. Bonham, L.S. Bartell, D.A. Kohl, The Molecular Structures of n-Pentane, n-Hexane and n-Heptane, J Am Chem Soc, 81(18), 4765-4769, 1959.
- [22] B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl, GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation J Chem Theory Comput, 4(3), 435447, 2008.
- [23] M.D. Hanwell, D.E. Curtis, G.R. Hutchinson. (2009, Sep.) Avogadro: A Framework for Quantum Chemistry Calculation and Visualization [Online]. Available: http://avogadro. openmolecules.net/
- [24] CPMD consortium. (2008, Jan.) CPMD: CPMD consortium page [Online]. Available: http: //www.cpmd.org/
- [25] 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, S.C. Wang, Anton: A Special-

Purpose Machine for Molecular Dynamics Simulation, ISCA'07, San Diego, California, Jun 913, 2007.

- [26] A. Thompson, H. Cho. (2010, Apr.) LAMMPS/ReaxFF potential. [Online]. Available: http://lammps.sandia.gov/doc/pair\\_reax.html
  [27] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo,
- [27] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, 2nd Edition, SIAM, 1994.
- [28] S.J. Plimpton, P. Crozier, A. Thompson. (2010, Apr.) LAMMPS Molecular Dynamics Simulator. [Online]. Available: http://lamps.sandia.gov/index.html