Focus Article

An overview of the Amber biomolecular simulation package

Romelia Salomon-Ferrer,1 David A. Case3 and Ross C. Walker1,2*

Molecular dynamics (MD) allows the study of biological and chemical systems at the atomistic level on timescales from femtoseconds to milliseconds. It complements experiment while also offering a way to follow processes difficult to discern with experimental techniques. Numerous software packages exist for conducting MD simulations of which one of the widest used is termed Amber. Here, we outline the most recent developments, since version 9 was released in April 2006, of the Amber and AmberTools MD software packages, referred to here as simply the Amber package. The latest release represents six years of continued development, since version 9, by multiple research groups and the culmination of over 33 years of work beginning with the first version in 1979. The latest release of the Amber package, version 12 released in April 2012, includes a substantial number of important developments in both the scientific and computer science arenas. We present here a condensed vision of what Amber currently supports and where things are likely to head over the coming years. Figure 1 shows the performance in ns/day of the Amber package version 12 on a single-core AMD FX-8120 8-Core 3.6GHz CPU, the Cray XT5 system, and a single GPU GTX680. © 2012 John Wiley & Sons, Ltd.

INTRODUCTION

The term Amber1 refers to more than just a molecular dynamics (MD) package. It includes the collection of numerous programs that work together to setup, perform, and analyze MD simulations, from the preparation of the necessary input files, to the analysis of the results. The name Amber also refers to a series of classical molecular mechanics force fields, primarily designed for the simulation of biomolecules. This includes the amino acid and nucleic acid parameter sets, termed, for example, ff94, ff99SB, and ff12SB2-4; carbohydrates termed Glycam5,6; phospholipids termed Lipid117; nucleic acids8; and general organic molecules termed GAFF.9 Together these parameter sets describe the most common components of biomolecular and condensed matter simulations, containing parameters for the most naturally occurring solvents, ions, amino acids, carbohydrates, and lipids plus, with GAFF and the Antechamber, most organic molecules. The Amber package also contains software designed to parameterize more complex molecules and fragments not currently present in the force field libraries.

Amber is the result of the collaboration of over 40 scientific researchers and additional external collaborators and contributors, actively working on the advancement of MD methods and on the study of numerous important biochemistry problems. Although the exact number of Amber users is hard to estimate, it is known to be installed in excess of 1000 sites worldwide. The widest used versions of the Amber force field, ff94,3 ff99SB,4 ff03,10 and GAFF,9 together have been cited almost 9000 times whereas the Amber software itself has in excess of 4000 citations, with over 2000 of those corresponding to the latest three versions. Maintaining a large set of programs that are used and developed by such a broad...

How to cite this article: WIREs Comput Mol Sci 2012. doi: 10.1002/wcms.1121
community leads to rapid development but inevitably implies trade-offs to ensure interoperability of the various components, acceptable performance, and long-term maintainability. As such, it has always been the philosophy of the Amber developers to focus on including and maintaining those methodologies that are actively being used and investigated within the research groups of developers and contributors. Thus, many methods from previous versions of the code have been deprecated or replaced by others over the years.

The history of the evolution of Amber can be found primarily in two references, the former referring to the early developments and the later to the evolution from there to Amber v9. In this paper, we present the development, evolution, and intelligent design of Amber over the latest three releases, up to and including Amber v12. The last few years have seen Amber be the object of significant software developments both in terms of scientific methodology and capitalizing on extensive paradigm shifts in computer hardware. Having a software that is both a platform for scientific methodology research as well as a robust and efficient computational tool, on modern hardware, has driven the development of the MD engine of the Amber package into three forks referred to as sander, pmemd, and pmemd.cuda. At the same time, the adoption of the Amber force fields within other MD packages has increased demand for the setup and analysis tools which were thus split out from the Amber package as of version 10, termed AmberTools, and made available for download under an open-source license. In this paper, we focus on the software description of Amber and, by association, AmberTools including only a brief description of the new developments and additions to the force fields. For further information, tutorials, and a more detailed description of any of the methods included in Amber, we encourage the reader to visit the Amber Web site and to consult the Amber manuals that contain a more complete list of references.

In the following section, we describe the three main MD engines contained in the Amber package, with a brief explanation of the place each of them have in the MD community. We outline, in more detail, the software development behind the performance-focused pmemd and pmemd.cuda packages and describe the methods presently available in each of them. In New MD Developments in Amber, we highlight developments that have been added to Amber since the last review paper while in Developments in AmberTools and Amber Forcefields we briefly outline the new additions to AmberTools and the Amber force field. Finally, in the last section, we discuss some of the potential future directions for Amber development.

SERIAL AND PARALLEL PERFORMANCE

As mentioned above, Amber now consists of three different, but highly coupled, MD simulation engines. Sander has traditionally been the most important platform for both computation and development in Amber, whereas pmemd and pmemd.cuda have focused on maximizing performance. Some of the more notable scientific methods added to sander since version 9 are discussed in New MD Developments in Amber.

Sander

There have been various improvements made to sander in the last few versions to both increase performance and improve usability. An implementation of binary trajectory files, based on the netCDF binary file format (joutfm = 1), is now available, as well as support for netCDF binary restart files as of version 12. This output format makes I/O from the master process more efficient, improves numerical precision and reliability, and substantially reduces file size. Support for a full three-dimensional (3D) decomposition of the reciprocal space fast Fourier transform (FFT) calculation helps to improve parallel scaling, whereas vectorization in key places and a complete rewrite of specific parts of the code, such as the quantum mechanical/molecular mechanical (QM/MM) support, has, in some cases, drastically improved serial performance.

Pmemd

There exist well-known limitations on increases in clock speed brought about by concerns with the power consumption of modern processors; hence the trend in workstations, clusters, and supercomputers has been toward increasing parallelization. In response to this, sander was rewritten with a focus on high performance and improved parallel scalability as a package named pmemd. Developed initially by Bob Duke (at National Institute of Environmental Health Sciences), pmemd began as a rewrite of sander as of Amber v6. It was officially released as part of the Amber package as of version 9. The original ethos of pmemd was to support the basic MD functions from sander but to run them as efficiently as possible while still producing output statistically equivalent to that of sander. Since Amber v10 numerous extensions have been made, mostly by the laboratory of Ross Walker (at the San Diego Supercomputer Center) and
other Amber developers to support additional, more complex scientific methods such as those discussed in the next section. These additions were made while ensuring that the carefully tuned parallel scalability and performance of pmemd were not adversely affected with each new addition.

Because pmemd represents a rewrite of sander with a focus on performance, substantial effort has been expended to guarantee the cross-compatibility of both codes. For the supported functionality, the input required is intended to exactly replicate that of sander and the output produced to be compatible and statistically equivalent to sander. For users, pmemd should simply feel like a version that runs more rapidly, scales better in parallel using the message passing interface (MPI), and can be used profitably on significantly higher numbers of processors. Pmemd is thus aimed at MD simulations of large solvated systems for long periods of time, especially if supercomputer resources are available. Some of the specific optimizations used in pmemd are described in the following paragraphs.

On the basis of a spatial decomposition scheme, which is optimized to minimize data exchange between processes, the speed and scaling is greatly improved compared with sander. Other tweaks are also included to maximize performance such as automatic selection of FFT decomposition, optimization of the writing of restart files, output buffering, axis optimization to maximize cache reuse, and the use of precisely tuned lookup tables for the direct space sums. For CPU communication, pmemd offers the possibility of using blocking and nonblocking calls for MPI point-to-point routines. Nonblocking communications are primarily used to overlap computation with communication and exploit possible performance gains. Pmemd can be compiled to use either protocol, with the nonblocking as default.

For long-range electrostatics in explicit solvent, the particle mesh Ewald15 algorithm has the option of using a ‘block’ or pencil FFT rather than the usual slab FFT algorithm. The block FFT algorithm allows the reciprocal space workload to be distributed to more processors, but at a cost of higher communications overhead, both in terms of the distributed FFT transpose and in terms of communication of the data necessary to set up the FFT grids at the beginning of the calculation. The use of block FFTs can be beneficial at high processor counts because it allows for better overlap of computation with communication. However, at low processor counts (typically <32), it can actually hurt performance. Pmemd handles the selection of block versus slab FFTs automatically. By default, a heuristic scheme, based on atom and MPI task count and FFT grid size, is used to identify how the block division should be done and whether direct force work is also assigned to tasks doing reciprocal space work, or whether the master thread is given any force and energy computation work to do, or is reserved strictly for handling output and load balancing.

Pmemd.cuda
With the emergence of graphics processing units (GPUs) as a practical and powerful platform for scientific computing, pmemd has been ported to the GPU platform using NVIDIA’s Compute Unified Device Architecture (CUDA) language.16 This work has been led by Ross Walker’s laboratory in close collaboration with Scott Le Grand and others at NVIDIA. The performance enhancements are remarkable as can be appreciated from Figure 1 and Table 1. The performance enhances are discussed in greater detail on the Amber Web site and in Refs 17–19.

The primary reason for using GPUs as an alternative hardware technology is based on their high computational power and memory bandwidth. GPUs have been present in personal computers for years, backed primarily by the gaming industry and generally used for 3D image rendering. Their success and strong demand has allowed a significant industrial development of GPU technology, not only continuously increasing the computational power and memory bandwidth, but at the same time reducing the prices substantially. When programmed carefully, GPUs can significantly outperform CPUs on highly mathematical and naturally parallel tasks. Today, a majority of computers and workstations in research laboratories already contain one or more GPUs, or can easily be upgraded to include them. For a review of the history of MD on GPUs, we refer the reader to a recent review article.20

Programming GPUs used to be very challenging, but the release of the programming language CUDA by NVIDIA and the subsequent development of OpenCL have reduced the difficulties significantly. There has been a true explosion of scientific codes that have been recently ported to work, at least partially, on GPUs. The computational complexity and fine-grained parallelism of MD simulations of macromolecules makes them an ideal candidate for implementation on GPUs.

Pmemd.cuda is based on the existing Fortran code in pmemd extended with calls to specific CUDA kernels for the GPU acceleration. This allows developers not only to use the robust infrastructure of pmemd, but also provides a simple framework to add
new features, by using GPU uploading and downloading routines to transfer vectors containing information, such as position, velocity, and so on, to and from the GPU in a simple fashion. At the time of writing, performance in excess of 75 ns/day for the joint amber charmm (JAC) production benchmark, can be achieved on a single commodity GPU (GTX 680). This is made possible by ensuring that all the key aspects of a time step are computed on the GPU, for example, bonding terms, direct space electrostatics, and van der Waals as the FFT-based reciprocal sum.

Integration and thermostats are also carried out on the GPU and hence copies from GPU to CPU memory are only required every time I/O is needed, typically every few thousand steps. Performance has also been maximized by carefully auditing the use of single, double and fixed point precision within the code to maximize performance while keeping the accuracy of the calculation equivalent to one run on the CPU.

Modern GPUs offer support for double precision floating-point arithmetic (DP), however this comes with a significant performance penalty. In Amber, for historical reasons, both the CPU codes, sander and pmemd, are written entirely using DP. In pmemd.cuda, there were initially three different precision models implemented. In one mode, the contributions to the nonbonded forces are calculated in single precision but bonded terms and force accumulation are calculated in double precision (SPDP), whereas in the other modes everything is computed and accumulated in single precision (SPSP) or double precision (DPDP). It has been shown that the use of SP in all places within the code can lead to substantial instabilities in the MD simulations. The released version of Amber v12, therefore, uses the mixed SPDP precision model as the default because the numerical results obtained are comparable with those of the full double-precision DPDP model, and the double-precision CPU implementations, but at significantly lower computational cost.

Very recently, work has focused on a combined single precision or fixed-point precision model termed SPFP. This new approach preserves the bitwise determinacies of the SPDP model but with substantial reduction in memory requirements and significant performance improvements on the latest generation of GPUs.
TABLE 2 | Methods and Features Currently Available in the Different Molecular Dynamics Engines Contained within Amber (v12)

<table>
<thead>
<tr>
<th>Methods</th>
<th>Sander</th>
<th>pme md</th>
<th>pme md-cuda</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMD Temp¹</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REMD Hamiltonian¹</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Langevin REMD¹</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>AMD</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PIMD¹</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PIMD-derived methods (CMD, RPMD, etc.)¹</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Constant pH¹</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>QM/MM</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NEB¹</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>TI¹</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extra points</td>
</tr>
<tr>
<td>NMR restraints</td>
</tr>
<tr>
<td>IPS</td>
</tr>
<tr>
<td>GBSA</td>
</tr>
<tr>
<td>GB</td>
</tr>
<tr>
<td>Soft core potentials</td>
</tr>
<tr>
<td>Harmonic restraints</td>
</tr>
<tr>
<td>Temperature Scaling</td>
</tr>
</tbody>
</table>

¹These features are available only in the MPI version of the corresponding codes.

NVIDIA GPUs (GTX680/690 and K10). The SPFP model has been developed to be indistinguishable from SPDP and is described in more detail in Refs 18 and 19.

Summary

With the introduction of pme md and pme md-cuda, the development model of the Amber package has changed. Sander is now considered the vehicle to explore new features, whereas pme md, running either on CPUs or GPUs, is designed to be the production code that implements sander’s most-used features in extensively tested software that performs well on high-performance architectures. All methods and features included in Amber are available in sander, whereas only a selected, but increasing, list of them are available in the other two codes. In the following section, we present the new methodology developments added to sander since version 9, and Table 2 describes their availability in sander, pme md, and pme md-cuda. Care has been taken that all features included in the three codes have been extensively tested to reproduce the same results, so in essence all codes are equivalent for those features, and the decision of whether to use one or other should rely exclusively on the amount and type of resources available.

Table 1 (section 1) shows timings for a standard Amber benchmark running on CPUs. The first entries consider only algorithm improvements over the last nine versions of Amber comparing the serial performance on identical hardware. The benchmark is dihydrofolate reductase (DHFR) in TIP3P water (23,558 total atoms). The current code is more than 3.6 times as fast as it was in Amber v.4 from just serial algorithm improvements. These numbers do not factor in changes in hardware speed or differences in parallel scalability, which are complete stories in themselves. The second section of Table 1 shows performance for Amber 11 and 12 on a single GTX580 GPU/GTX680 graphics card, which in the latter case runs this benchmark at over 75 ns/day. In comparison, using all cores of a Dual x Hex Core Intel X5670 2.93GHz CPU the performance tops out at 14 ns/day. The parallel scaling of Amber has also improved significantly; the same benchmark on 48 CPUs has a performance of 35.92 ns/day while as a point of comparison the same benchmark on 2 GTX580 GPUs gives performance of 67.55 ns/day, whereas the latest version, Amber 12, running the new SPFP precision model on a single NVIDIA GTX680 graphics card, exceeds 75 ns/day.

NEW MD DEVELOPMENTS IN AMBER

Amber developers, as well as external contributors, have made a large number of significant developments...
and improvements to algorithms since the release of Amber v9. Space limitations mean that we can only briefly cover the major additions here; however, for further information, please refer to the papers cited here, and for a more exhaustive list please refer to the latest edition of the Amber manual.

Replica-Exchange Method
In replica-exchange method (REMD)\textsuperscript{21–23} noninteracting copies of the system (replicas) are simulated concurrently at different values of some independent variable. Replicas are then subjected to Monte Carlo evaluations periodically to decide whether or not to exchange values of the independent variable. REMD enables simulation in a generalized ensemble, weighted by non-Boltzmann probabilities, so a replica trapped in a local minimum can escape via exchange to a different value of the independent variable. In Amber, REMD can now be used with either temperature or different Hamiltonians as the choice for independent variables, with recent modifications allowing both single-dimension and multiple-dimension REMD.

Accelerated Molecular Dynamics
Accelerated molecular dynamics (AMD)\textsuperscript{24,25} adds a bias to the potential function that can facilitate crossing of high energy barriers without advance knowledge of the location of either the potential energy wells or saddle points. The added potential alters the underlying shape of the true potential in a very simple way, allowing it to be recovered by a reweighting procedure. Amber v12 supports acceleration based on the entire potential or only the potential term arising from the dihedral angle contributions. Amber v12 also supports acceleration of every step or at intervals specified by the user.

Self-Guided Langevin Dynamics
Self-guided Langevin dynamics (SGLD)\textsuperscript{26–28} is a method that enhances conformational sampling by accelerating low-frequency modes through the use of an ad hoc time-averaged momentum term. The running average of the momentum over a short-period simulation time is added back as an external force to the simulation system to accelerate low-frequency motions. SGLD selectively enhances and suppresses molecular motion based on their frequency; it thus accelerates conformational searching without the modification of the energy surface or the increase of temperature. Improved ideas for SGLD are included to enhance sampling along soft degrees of freedom. New developments, such as the force-momentum-based SGLD, allow the direct sampling of the canonical ensemble without the need for reweighting.\textsuperscript{27,28}

Nudged Elastic Band Method
Nudged elastic band method (NEB),\textsuperscript{29,30} supported initially with Amber v9 and then to a greater extent in Amber v11, offers a way to find minimal energy paths between two configurations. The path for a conformational change is approximated with a series of images of the molecule describing the path. Each image in-between is connected to its neighboring images by springs along the path that serve to keep each image from sliding down the energy landscape onto adjacent images. The newer Amber implementation supports partial NEB in which only part of the system is connected by springs. This allows both explicit solvent and more focused NEB calculations to be run.

Generalized Born Methods
Two new generalized Born (GB) solvation models are available specified by igb 7 (v10) and 8 (v11). These models use a pairwise correction term to the Hawkins, Cramer, and Truhlar implementation\textsuperscript{31} to approximate a molecular surface dielectric boundary that serve to eliminate interstitial regions of high dielectric smaller than a solvent molecule. These models have the same functional form and carry little additional computational overhead relative to the other GB models. The parameters in igb 7 correspond to the Mongan, Simmerling, McCammon, Case, and Onufriev implementation,\textsuperscript{32} whereas the parameters in igb 8 correspond to the ones by Nguyen and Simmerling.\textsuperscript{33}

Solvation Models
In addition to the explicit and implicit solvation models, Amber now also includes a third class of solvation model for molecular mechanics simulations, the reference interaction site model (RISM) of molecular solvation.\textsuperscript{34} RISM is an inherently microscopic approach, calculating the equilibrium distribution of the solvent, from which all thermodynamic properties are then derived. As of Amber v12, an enhanced 3D-RISM model using a variety of closure approximations, and with a better treatment of aqueous electrolytes, is available.

Poisson–Boltzmann Solvation
Amber v10 and onward now offers support for calculating the reaction field and nonbonded interactions
using a numerical Poisson–Boltzmann (PB) solver\cite{35,36,37} as an alternative continuum solvent model. As of Amber v12, models for membranes and support for periodic systems are available.

**Isotropic Periodic Sum**
A polarized formulation for Isotropic Periodic Sum (IPS)\cite{38} and the discrete fast Fourier transform (DFFT)\cite{39} version was added to sander in Amber as of v11 as an electrostatic and long-range interaction model. These additions allow IPS to describe properly systems with polarized molecules and, with the use of DFFTs, allows for the use of a smaller cutoff radius. IPS requires a coarser grid for the DFFT mode than particle mesh ewald (PME) and thus tends to scale better in parallel\cite{40}. As of Amber v12 IPS is now supported in sander, pmemd, and pmemd.cuda.

**Quantum Dynamics for Nuclear Motions**
The path-integral molecular dynamics (PIMD) method\cite{41,42,43} has been implemented in sander as of Amber v9. PIMD is a general method for calculating equilibrium properties of a quantum many-body system based on Feynman’s formulation of quantum statistical mechanics in terms of path integrals. The current implementation in Amber allows the ‘quantization’ of either the entire system or just a subsection of it. Several methods based on the path-integral formalism are also available: (1) centroid molecular dynamics (CMD)\cite{44,45} and ring polymer molecular dynamics (RPMD)\cite{46,47} to perform approximate quantum dynamical calculations, these two methods can be used to calculate quantum time-correlation functions plus RPMD has been shown to be a powerful method to calculate reaction rates and dynamics in enzyme environments\cite{48,49}. (2) Linearized semiclassical initial value representation (LSC-IVR)\cite{50,51} based on the semiclassical initial value representation (SC-IVR)\cite{52,53} can be shown to be exact in the classical limit, high-temperature limit, and harmonic limit. The LSC-IVR can treat both linear and nonlinear operators in a consistent way, and can be applied to nonequilibrium as well as the above-equilibrium correlation functions, and can also be used to describe electronically nonadiabatic dynamics, i.e., processes involving transitions between several potential energy surfaces. (3) Quantum instanton\cite{54,55,56} is a theoretical approach for computing thermal reaction rates in complex molecular systems, based on the semiclassical instanton approximation (QI).\cite{57} The essential feature of the QI rate constant is that it is expressed wholly in terms of the quantum Boltzmann operator, so it can be evaluated for complex molecular systems using the path-integral methods. (4) Finally, equilibrium and kinetic isotope effects can be calculated using a PIMD-based method that does thermodynamic integration (TI)\cite{58,59,60} over mass.

**Free Energy Tools**
Amber has supported a range of free energy methods through its history with the latest approaches being umbrella sampling and TI. As of Amber v10, umbrella sampling has been enhanced such that atom groups may be used not only in distance restraints, but also in angle, torsion, and plane restraints, as well as new generalized restraint methods. Thermodynamic Integration now supports ‘single’ or ‘dual’ topologies and soft-core potentials for improved sampling allowing atoms to appear and disappear without the need for the use of dummy atoms. This avoids the issues with sampling that complicated previous implementations of TI. Tighter integration with replica-exchange simulations is also available.

**Constant pH Dynamics**
Constant pH was first implemented in sander\cite{61} as of version 9 and has been expanded in later versions as well as being added to pmemd as of v12. Constant pH addresses issues present with regular MD simulations where titratable residues are assigned a fixed protonation state. This restriction can lead to inaccuracies in situations where the pKa can change substantially due to changes in conformation. This is especially true in cases where the pKa is close to the pH of the solvent used. In Amber, the Monte Carlo sampling of the Boltzmann distribution of protonation states concurrent with the MD simulation addresses those issues. Support for implicit solvent GB models was added as of version 9. Support was extended in later versions to also include explicit solvation.

**QM/MM Methodology**
Many additions and changes have been made to the QM/MM algorithms in sander. A complete rewrite of the QM/MM support was introduced in v9 and substantial improvements have been made with each new version of Amber. New semiempirical neglect of diatomic differential overlap (NDDO)\cite{62} type and density functional based tight binding (DFTB)\cite{63} implementations were first introduced in version 9, supporting most modern Hamiltonians including, as of version 11, PM6\cite{64} and AM1/d. The QM/MM algorithms now conserve energy for long MD simulations. Extensive optimization provides the fastest semiempirical implementation available while also
supporting parallel execution. Explicit solvent long-range QM/MM electrostatic interactions are accounted for using a QM/MM compatible version of the PME approach,\textsuperscript{65} whereas implicit solvation is handled by a QM/MM compatible version of the regular Amber GB models.\textsuperscript{65} Support is also available, as of version 12, for \textit{ab initio} and DFT QM potentials via interfaces to external quantum chemistry software packages including Gaussian,\textsuperscript{66} Orca,\textsuperscript{67} Terachem,\textsuperscript{68} GAMESS,\textsuperscript{69} NWChem,\textsuperscript{70} and ADF.\textsuperscript{71}

DEVELOPMENTS IN AMBER TOOLS AND AMBER FORCEFIELDS

A significant change in the structure of the Amber software package over the last years has been the separation of Amber into two major package collections, Amber and AmberTools. AmberTools consists of an independent set of software tools that, for the most part, used to be part of Amber. Generally, the tools are used to prepare coordinate and topology files using the Amber force fields as well as analyze the resulting output and trajectory files. AmberTools also includes programs that perform many simulation tasks. A key feature of AmberTools is that it is entirely an open source providing access to many of the key features of Amber without requiring a license. Some of the main components of this suite of tools are discussed below. For further information on each one of them, please refer to the AmberTools manual.\textsuperscript{13}

Force Fields

The Amber force fields, distributed as part of AmberTools, have been greatly expanded in the last years and an in-depth discussion of the various changes is beyond the scope of this paper and so only a brief overview is provided here. Amber now supports a wide range of force fields including all of the pair wise amino and nucleic acid variants\textsuperscript{2–4,8,10} of which the ff10 force field represents a collection of the most widely used combinations. FF10 consists of the ff99SB amino acid parameters,\textsuperscript{4} the BSC0 DNA parameters,\textsuperscript{72} the Cheatham et al. updated ion parameters,\textsuperscript{73–74} and modifications to RNA.\textsuperscript{75–76} There is also a new fixed-charge protein force field, ff12SB, provided with Amber v12 along with enhanced support for polarizable potentials as well as the Charmm force fields via an auxiliary program called Chamber,\textsuperscript{77} described below. Carbohydrates are supported through the Glycam series of force fields\textsuperscript{5,6} whereas phospholipids are supported both through the Charmm force fields and through the recent addition of the Lipid11\textsuperscript{7} force field released with Amber v12. Support for ligands is provided by the General Amber Force Field (GAFF)\textsuperscript{9} with setup and parameterization automated through an auxiliary program called Antechamber. Polarizable force fields are supported for the induced point dipole model\textsuperscript{78,79} as well as the Amoeba force field.\textsuperscript{80,81} The most widely used explicit solvent models are supported including the major water models: TIP3P, TIP4P, TIP5P, TIP4PEW, SPCFW, and so on.\textsuperscript{82–89}

Preparation Tools

AmberTools contains a range of preparation tools that are designed to work together in a loose fashion. Some of the key ones are

- \textit{LEaP} is a module from the Amber suite of programs, which can be used to generate force field files compatible with the Amber MD packages and Nucleic Acid Builder (NAB). There are two versions currently available as of Amber v12. One command line program termed \textit{tleap} and a command line editor termed \textit{xleap} and \textit{tleap} are equivalent and refer to the same code underneath.
- \textit{Antechamber} is a set of tools to generate files primarily for organic molecules. It is designed to be used with the GAFF force field and will automatically assign atom types and attempt to generate missing parameters. A range of input file formats are supported and the output files are designed to be read into \textit{LEaP} as part of the build procedure for proteins containing organic ligands.
- \textit{Metal Center Parameter Builder} (MCPB) provides a means to rapidly build, prototype, and validate MM models of metalloproteins. It uses bonded and electrostatics models to expand existing pairwise additive force fields.
- \textit{Chamber}\textsuperscript{77} is a program that can read Charmm\textsuperscript{90,91} topology (psf), coordinate (crd), and parameter files (par & dat) and will produce Amber compatible prmtop and inpcrd files. This allows the simulation of the Charmm force field in sander, pmemd, and pmemd.cuda to machine precision.
- \textit{Paramfit} is a program that allows specific force field parameters to be optimized or created by fitting to quantum energy data when parameters are missing in default force fields and \textit{antechamber} cannot find a replacement.
### Simulation Tools

In addition to the setup tools, AmberTools has also evolved to include a number of codes that can be used to carry out a range of simulations:

- **Nucleic Acid Builder** is a high-level language that facilitates manipulations of macromolecules and their fragments. Some force field calculations such as MD, minimization, and normal mode analysis can also be carried out in NAB. A parallel version of this tool is available as mpinab. NAB provides an interface to PB and RISM integral-equation solvent models.

- **mdgx** is an MD engine with functionality that mimics some of sander and pmemd, but featuring simple C code and an atom sorting routine that simplifies the flow of information during force calculations.

- **Sqm** is a standalone semiempirical quantum chemistry program, originally extracted from the QM/MM portions of sander. It is primarily used by Antechamber for calculating AM1BCC point charges but also serves as a QM library for sander. It is envisioned that this will ultimately become a fully featured quantum package.

- **Pupil** is a systematic approach to software integration in multiscale simulations. It is primarily used within the context of Amber as an interface for performing QM/MM simulations using sander and various quantum chemistry packages.

- **rism1d and rism3d.snglpnt** are 1D and 3D-RISM solvers for single-point calculations.

### Analysis Tools

A range of tools dedicated to the analysis of MD simulation results are part of AmberTools including:

- **ptraj** is the main trajectory analysis tool in Amber, it is able to perform many types of analyses, and can process multiple trajectories. The list of analysis procedures included in ptraj is very extensive and the reader should consult the ptraj chapter of the AmberTools manual for further information (13). An MPI parallel version of this tool is also available as ptraj.MPI.

- **cpptraj** is a complimentary program to ptraj, written in C++, that can process trajectory files with different topology files in the same run. Some key differences between ptraj and cpptraj are highlighted in Table 3. Some parallelization has been added for multicore machines using OpenMP. It is ultimately envisioned that cpptraj will replace ptraj in later versions of Amber.
• **pbsa** is a package containing several efficient finite-difference numerical solvers, both linear and nonlinear, for various applications of the PB method. An MPI parallel version of this tool is available as pbsa.MPI.

• **Mmpbsa**[^92] is a method for calculating binding-free energies. Amber now has two scripts to perform MM/Poisson–Boltzmann (or generalized Born) Surface Area [MM/GBSA] calculations, `mmpbsa.pl` written in Perl and `mmpbsa.py` written in Python. This is a postprocessing method in which representative snapshots from an ensemble of conformations are used to calculate the free energy change between two states.

### CONCLUSIONS AND FUTURE OUTLOOK

MD simulations have increased dramatically in size, complexity, and simulation timescale in recent years while the questions being answered with these methods have also diversified. In the last six years, the typical system size in publications of MD simulations has grown from 50 to 100K atom regime and/or ten to hundreds of nanoseconds to millions of atoms and/or microseconds and more. The evolution of the Amber package in recent years has been significant; in only a few years, the MD codes have been completely restructured. Performance in Amber has increased dramatically from around 941 ps/day for a simulation of DHFR on a single-core desktop in Amber v9 to over 75 ns/day in Amber v12 using a single GTX680 GPU in such a desktop. The latest developments, both in methodology and particularly in performance enhancements have established Amber as a modern and widely used MD software package.

The Amber community is both active and wide, and new ideas emerge frequently. Although predicting the exact direction that Amber might take is difficult, we can give a small glimpse of some implementations that will be available in the near future. Adaptive QM/MM methods that allow the partitioning of the system into QM and MM regions that change during the course of the simulation will soon be included. More complex and higher fidelity force fields will follow, as will a continuous trend to migrate more methods to pmemd and pmemd.cuda. Heterogeneous architectures, such as GPUs and possibly Intel’s MIC architecture will likely dominate on the performance front in the near to medium term with performance improvements of an order of magnitude likely over the next 3–5 years.

It is hoped that the continuation of the implementation of smart ideas coupled with good programming that take advantage of the increasing computer power as well as the emergence of new computing platforms will bring more exciting advances to Amber in the future.

### ACKNOWLEDGMENTS

This work was funded in part by the National Science Foundation through the Scientific Software Innovations Institutes program—NSF SI2-SSE (NSF1047875 & NSF1148276)—grants to R.C.W and also by the University of California (UC Lab 09-LR-06-117792) grant to R.C.W. Computer time was provided by the San Diego Supercomputer Center through National Science Foundation award TGMCB090110 to R.C.W. The work was also supported by a CUDA fellowship to R.C.W. from NVIDIA.

### REFERENCES


