ELSEVIER

Computer Physics Communications 91 (1995) 1–41

Computer Physics Communications

AMBER, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules

David A. Pearlman [a], David A. Case [b], James W. Caldwell [c], Wilson S. Ross [c], Thomas E. Cheatham III [c], Steve DeBolt [b], David Ferguson [d], George Seibel [e], Peter Kollman [c 1]

[a] *Vertex Pharmaceuticals, 40 Allston Street, Cambridge, MA 02138, USA*
[b] *Department of Molecular Biology and Chemistry, Scripps Research Institute, La Jolla, CA 92037, USA*
[c] *Department of Pharmaceutical Chemistry, University of California, San Francisco, CA 94143, USA*
[d] *Department of Medicinal Chemistry, University of Minnesota, Minneapolis, MN 9055455, USA*
[e] *Smith Kline Beecham Pharmaceuticals, 709 Swedeland Road, King of Prussia, PA 19406, USA*

Received 31 October 1994

Abstract

We describe the development, current features, and some directions for future development of the AMBER package of computer programs. This package has evolved from a program that was constructed to do Assisted Model Building and Energy Refinement to a group of programs embodying a number of the powerful tools of modern computational chemistry–molecular dynamics and free energy calculations.

Keywords: Molecular mechanics; Molecular dynamics; Normal modes; Free energies; Computer simulations

PROGRAM SUMMARY

Title of program: AMBER version 4.1

Program obtainable from: University of California, San Francisco (UCSF) or Oxford Molecular, Ltd. (OML)

Licensing provisions: The AMBER programs' source code is available upon receipt of a signed license agreement and the payment of a license fee, the amount depending on whether the use is commercial or non-commerical. To receive a license agreement and other ordering information, please send *e*-mail to amber-request@cgl. ucsf.edu or support @oxmol. co. uk or consult the Web site: http://www. amber. ucsf. edu/amber/amber. html.

Computer for which the programs are designed and others on which it has been tested: The FORTRAN programs are written primarily in standard f77, and should be readily portable to any system fully implementing the standard. Moreover the compilation system used for the main FORTRAN programs, which is available for free independent of the source code, is designed for easy adaptability to any UNIX system, as well as for producing code targeted for other operating systems. Example UNIX machines include: HP 735, SGI single and multiple (shared memory parallel) processors, DEC Alpha

[1] To whom reprint requests should be addressed.

SSDI 0010-4655(95)00041-0

machines, Convex 3820, Sun SPARC, Cray C90 (single processor and shared memory parallel) and T3D, IBM RS6000, and parallel networks and clusters of machines running MPI, PVM, and TCGMSG. The X-windows graphical model-building program includes R4/R5-compatible widget libraries and should be portable to any X11 R4/R5 system. Example machines include HP, SGI, SPARC, and linux PC.

Operating system: UNIX, with significant VMS and VM support

Programming languages used: The main computational programs are in FORTRAN, many others are in C, and some utilities and the compilation system use UNIX shell scripts awk and make.

Memory required to execute with typical data: suggested 16 Mbytes for systems of 10000 atoms

Number of bits in a word: runs on a variety of architectures

Number of processors used: When compiling for parallel execution, currently hardcoded to 512 processors or less for the message passing implementation and unlimited for the shared memory implementation. Effective speedup depends on the speed of communication, size of the problem, options enabled, and sequential parts of the code.

Number of lines in distributed programs, including test data: There are 332000 lines in 930 source files, of which 187000 are code. The UNIX distribution, including documentation and test data, comprises 68 Mbytes before compilation.

Keywords: molecular mechanics, molecular dynamics, NMR refinement, free energy, normal modes, parallel computation, AMBER

Nature of physical problem
Simulations and analysis of the structures, noncovalent interactions and classical dynamics of organic and biological molecules.

Method of solution
Classic equations of motion, with force fields based on harmonic bonded interactions as well as point charge and 6–12 nonbonded interactions. Atomic polarizabilities can be used. Nonbonded cutoffs can be applied, or the Ewald sum method can be used for dynamics in periodic systems. In molecular dynamics, the system is coupled to an external bath of constant temperature and/or pressure. Various restraints and constraints can be applied.

Restrictions on the complexity of the problem
Nonbonded pairlist generation is on order of N^2 in time, where N is the number of atoms. The number of nonbonded pairs is on the order of the number of atoms times the volume of the nonbonded cutoff sphere. Thus pairlist energy evaluation is on the order of $O(N, r^3)$ in space and time, where r is the cutoff radius. Together, these components usually account for over 90% of the execution time for systems of 100000 atoms (5% for list generation, 85% for evaluation). In practice, systems of more than 100000 atoms are rarely if ever modeled at the moment.

Typical running time
A DNA hexamer in a box of water (7700 atoms) takes 80–325 seconds (8 Ångstrom cutoff vs. Ewald sum) to run 100 time steps (0.2 ps) on a single SGI R8000 75 MHz processor. Thus it would take about 4.5–18.8 days to simulate a nanosecond.

Unusal features of the programs
The free energy features and NMR restraints are well-developed, a new AMBER additive force field is included, atomic polarizabilities can be used, a fast particle mesh algorithm is used for the Ewald sum periodic boundary, and the graphical model-building program runs on generic X machines without special hardware requirements.

LONG WRITE-UP

1. Introduction

The concept of AMBER dates back to 1978 when Paul Weiner joined Peter Kollman's laboratory. Weiner had been studying molecular mechanics and X-ray refinement as a postdoc with Martin Karplus at Harvard. He brought the computer code for energy and Cartesian coordinate analytical derivative evaluation, developed originally at the Weizmann Institute [1] and refined/improved by Gelin at Harvard [2], to UCSF. At the time, molecular mechanics programs tended to be specialized for given classes of

molecules. It was Weiner and Kollman's goal to develop a completely general molecular mechanics program, capable of energy component analysis and computer graphics display as well as energy calculations, that could handle small molecules as well as polymers with arbitrary connectivities and could do so in a reasonably user-friendly way. The goal was reached in the form of the program AMBER, as described in [3]. Using a tree structure format and algorithms to generate bonds, angles, dihedral angles and "excluded atoms" from the bond connectivity, Weiner constructed PREP to process topological information on a residue or molecule and LINK (to link residues together in general ways). Two other modules, EDIT (which enabled coordinates to be manipulated), and PARM (which mapped molecular mechanical parameters onto the atom types and topology) were constructed to feed into MIN, the energy minimization program. ANAL, a program to evaluate the molecular mechanical energy components for arbitary collections of atoms and a simple graphical display program completed the first version of AMBER.

Weiner and Kollman were fortunate to be located in the same department as the UCSF Computer Graphics Laboratory, whose Evans and Sutherland PS2 enabled real time color interactive computer graphics [4], something that was available at few other places at that time. It was also very good fortune that Andrew Dearing joined the Kollman lab at the time. Dearing, with previous experience in graphics programs at Oxford, constructed a general macromolecule display program CHEM [5], later marketed by Evans and Sutherland as PSSHOW. This program and later, MIDAS [6], developed in the Graphics Lab, provided a crucial complement to AMBER in studies of complex macromolecular systems because without the ability to effectively visualize and think three dimensionally about such systems, extracting useful insights from molecular mechanics on them is extremely difficult. Although CHEM and AMBER were not tightly coupled, communicating simply via PDB files, the complementarity in using them together to gain insight into nucleic acid and protein systems was very clear.

Shortly after the first version of AMBER was completed, David Case and Paul Weiner incorporated analytical second derivatives into the program, enabling normal mode analysis to be performed, and Weiner wrote a graphical program to visualize the normal modes. During this time, both energy minimization and normal modes were used in the development of a new force field [7,8] which also featured atomic charges derived by fit to the electrostatic potential [9]. This could be viewed as the completion of version 1 of AMBER [10].

Wilfred van Gunstersen had recently implemented molecular dynamics into his program GROMOS [11]. Around this time, Kollman met van Gunsteren at a meeting and inquired whether GROMOS was available, to be used to incorporate molecular dynamics into AMBER. Van Gunsteren was willing to make GROMOS available. Although Kollman may have been prescient in his vision in the initial construction of AMBER, his enthusiasm for molecular dynamics was not great at that time. Luckily for him, a very talented scientist, U.C. Singh, had joined his group, and with help from graduate student Scott Weiner "spliced" the molecular dynamics from GROMOS into AMBER in a very short time [12,13]. In addition, Singh wrote a molecular dynamics analysis program. This completed the development of AMBER 2 [14].

At around this time, reports from Terry Lybrand (a former graduate student at UCSF who was a postdoc in Andy McCammon's laboratory in Houston) on the use of molecular dynamics and statistical mechanics to calculate free energy differences and the appearance of a paper by Jorgensen and Ravimohan [15], employing Monte Carlo methods and the same statistical mechanical equations to calculate the difference in aqueous solvation free energy of methanol and ethane, galvanized Singh and Kollman. Singh thus began to incorporate these methods–termed 'free energy perturbation methods'–into AMBER and, working with Paul Bash and Frank Brown, tested his code on both simple and complex systems. Two papers in Science [16,17] and one in J. Amer. Chem. Soc. [18] describe the methodology and two exciting early applications. Singh also added two other important capabilities to AMBER. The first was the capability of including non-additive effects in molecular mechanics using atomic polarizabilities

[19]. Singh also coupled AMBER with a version of the Gaussian series of programs to form QUEST [20], a combined quantum/molecular mechanics program. The above culminated the development of AMBER 3 [21].

At this point, AMBER had become like an overgrown child, unruly and difficult to manage. Fortunately, George Seibel, a UCSF graduate student at the time, undertook the thankless task of cleaning up the code (including fixing bugs and merging divergent machine-specific versions of the programs), implementing a validation suite, and bringing the release under central management for the first time. This work resulted in AMBER 3A [22,23], which included a new residue-based periodic imaging algorithm, performance and reliability enhancements, and fundamental revisions to abstract machine dependencies and create the first generally portable version of AMBER. In a parallel development, David Spellmeyer, a postdoctoral fellow, working with William Swope of IBM Palo Alto and Erik-Robert Evensen at UCSF, began the development of a new molecular dynamics module, SPASMS [24].

The arrival of David Pearlman at UCSF began the development of AMBER 4 [25]. The major additions in AMBER 4.0 were a large number of new capabilities/improvements in GIBBS (Pearlman), the free energy module, the addition of SANDER, a module with software for restrained MD and NMR relaxation refinment (Case and Pearlman) and the addition of the capability of non-additive force fields in molecular dynamics (Caldwell) and normal mode analysis software (Case). Bill Ross carried on Seibel's organizational work, adding compilation and portability features, etc.

Continued development of the SPASMS module has been carried out by David Ferguson. Its presence has also allowed considerable cross-fertilization of ideas among the various MD modules (e.g. Swope's ideas for speeding up non-bonded interactions that have made their way into GIBBS and SANDER). These and the developments in AMBER 4.1 [26], soon to be released, are described below.

AMBER is now nearly 15 years old, not as old as Gaussian and MM2, but still rather long-lived. It has, in our opinion, remained young and vital because of the constant infusion of new ideas and algorithms, fueled by the increasing computer power that has enabled their application to interesting molecules. It is also ironic that AMBER began as a major step forward in a user-friendly interface to a general molecular mechanics program and it is now behind other software in not having a modern graphical, user-friendly interface. This is due to the sometimes. unappreciated difficulty of writing and maintaining such interfaces, particularly in a "society" which rewards scientific applications or methodology development much more than skill in writing general, graphical oriented interfaces. Hopefully, a new X-windows program by Christian Schafmeister, LEaP, [27] and Pearlman's powerful AMBER/INTERFACE front end [28] to SANDER and GIBBS will help close the "interface gap".

Below, we present a description of the most important features of AMBER. The sections are:

1. This section;
2. Overview of the programs and the code;
3. Basic ideas underlying molecular mechanics;
4. Inclusion of non-additive effects in the potential energy function;
5. SANDER, the program for energy minimization, molecular dynamics and NMR refinement;
6. GIBBS, the program for free energy calculations;
7. SPASMS, a new dynamics/free energy program;
8. NMODE, a program for normal mode analysis;
9. Structural and energy analysis tools;
10. INTERFACE, a scripting front-end for SANDER and GIBBS;
11. LEaP, a graphical molecular builder program;
12. Concluding remarks.

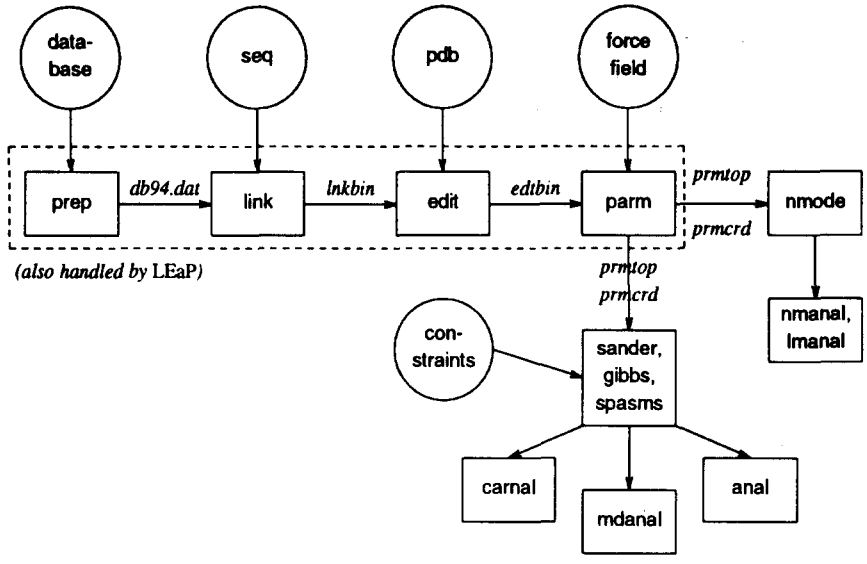

Fig. 1. Basic information flow in AMBER.

2. The programs: source code management, portability and performance

The following figure illustrates the relationships between the main programs in the AMBER package.

AMBER now consists of about 65 programs, with 930 source files containing over 332 000 lines, of which about 187 000 lines are code. AMBER code has traditionally been all FORTRAN, but with the 4.1 release, 30% of the codes is in C (LEaP is 20% of the total). The release is managed under UNIX; the UNIX version as distributed contains all the management tools, including the ability to generate source code for non-UNIX systems and utilities for e.g. recursive ftp of a source tree to VMS machines. All versions include all the source code. An email reflector has been created for discussion of user issues and the distribution of bug fixes.

A compilation system has been devised that allows the user to compile everything on any version of UNIX by copying a single file containing machine dependencies from a library covering about 22 types of UNIX, then running a script which invokes 50 Makefiles. The same system allows source code to be generated for VMS, VM/CMS, DOS, and CTSS operating systems. Unlike the source code, this compilation system has been placed in the public domain. The machine dependency files set UNIX environment variables which define complier commands for various optimization levels as well as various flags which are used to by the cpp preprocessor to screen each FORTRAN source file for machine dependencies (including shared memory and message-passing parallelism), which are delimited with "#ifdef...#endif" constructs. Cpp also handles "#include" statements, which are typically used for common blocks. The environment variables are used by a shell script ('Compile') which is invoked by the Makefiles to compile and link the FORTRAN programs or to generate target source for a given machine. The machine file also indicates a machine-specific directory where a library, sys.a, is built, which can include basic C library routines such as getarg() to be linked with FORTRAN if the machine's FORTRAN library is deficient. A small library of machine-dependent FORTRAN routines is provided: mexit() allows the programs to return status to the operating system; amopen() hides variations in open(); and second() and wall() hide details of cpu and wallclock timing. A portable *namelist* library is also provided courtesy of Nelson Beebe, Department of Mathematics at the University of Utah.

An exception to the above machine-dependency system is the new SPASMS program, for which an auxiliary FORTRAN program is provided to mask the program's source code for machine dependencies. This has the advantage of allowing one to build source for any machine on any machine (i.e. the 'management' machine does not need to be UNIX). A disadvantage is that machine dependencies are much harder to program in the masking program. Some UNIX tools have also been provided to split the single, canonical SPASMS source file into many for faster recompilation of individual modules during development, and remerge then into the canonical, single file form if desired.

A demo tree is provided with examples of various chemical systems, and a complementary test tree is provided to run the code after compilation and compare the results with the demo tree. There is also a benchmark tree with a variety of larger systems in water and vacuum to test the speed of new algorithms and to compare the performance of various machines.

3. Basis for molecular mechanics

Molecular mechanics is an approach to representing the energy surface for bond and non-bonded interactions with a simple, analytical and easily differentiable energy function. From the beginning, the following additive potential energy function has been supported in AMBER:

$$V = \sum_{\text{bonds}} K_r(r - r_{\text{eq}})^2 + \sum_{\text{angles}} K_\theta(\theta - \theta_{\text{eq}})^2 + \sum_{\text{dihedrals}} \tfrac{1}{2}V_n[1 + \cos(n\phi - \gamma)]$$

$$+ \sum_{1<j} \left(\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\varepsilon R_{ij}} \right) + \sum_{\text{H-bonds}} \left(\frac{C_{ij}}{R_{ij}^{12}} - \frac{D_{ij}}{R_{ij}^{10}} \right). \tag{1}$$

In Amber, version 1, minimization of V (total potential energy) was supported using both steepest descent and conjugate gradient techniques [3]. The same energy function, V, and its derivatives, are used in second derivative minimization and normal mode analysis, in molecular dynamics simulations with or without restraints, and in free energy calculations, (where the V for one molecular system can be 'mutated' into the V for another system).

The physical reality of simulations using AMBER or any other such program depends on the accuracy of Eq. (1). One of the strengths of AMBER has been the accuracy of its potential energy function [29]. That said, it must be emphasized that this accuracy pertains to confromational and non-bonded energies, not high frequency intramolecular motions. Some programs (e.g. MM2/MM3 [30] and BIOSYM's DISCOVER [31]) support a much more complex energy function for intramolecular interactions (e.g. higher order term in bonds, $r - r_{\text{eq}}$, angles, $\Theta - \Theta_{\text{eq}}$ and coupling terms between these). It is our opinion that if one's focus is on non-bonded and conformational energies of complex molecules, the inclusion of such terms makes the model unnecessarily cluttered with little gain in representing the properties of interest. More important are efforts to optimize further the non-bonded and electrostatic terms in Eq. (1) [32], and the inclusion of non-additive effects (see below).

One should not forget, however, that the most daunting challenge in computational applications of Eq. (1) to complex, polymeric molecules is the local minimun problem [33], i.e. the difficulty in accurately representing the relative free energy energies in solution of all the important populated conformations of a molecule [34]. This is a challenge for the furture.

Finally, a limitation of Eq. (1), should be noted: electronic structure changes (excited states, bond breaking and making) are not represented. Thus, for simulations of reactive processes, one must couple molecular mechanics with quantum mechanics as has been done by Singh [20] and Merz and co-workers [35] using AMBER.

4. Non-additive extensions to the 'Amber' Hamiltonian

One of the major differences between the classical physics methods used to study molecular phenomena and the more rigorous quantum mechanical methods is the ability of the latter to account for charge redistribution. There is a method available to add that capability to classical Hamiltonia. This approach ('induced polarization') adds polarizable units to the Hamiltonian which then interact with each other and the fixed charges. In the most general terms, both the fixed charges and the inducible units can be located at arbitary points in space, subject to the constraint that the calculated properties 'resemble' the molecule. In practice, fixed charges are usually restricted to locations corresponding to the nuclei of the atoms (although widely used four point models for water are a notable exception) in keeping with the notion of point atoms and that charge is a scalar property. However, molecular polarizability is rigorously a tensor phenomenon and as such is amenable to being described by a nonlocal expression. It is initially difficult to choose between a 'chemical' mode of choosing the polarization approach and a 'model' approach. In the 'chemical' approach, units of standard chemical description (e.g. C =0, C_αN—(=0, C_6 etc.) are representated by general 3×3 α_{ij} tensors.

$$\alpha = \begin{pmatrix} \alpha_{11} & \alpha_{21} & \alpha_{31} \\ \alpha_{12} & \alpha_{22} & \alpha_{32} \\ \alpha_{13} & \alpha_{23} & \alpha_{33} \end{pmatrix}.$$

The 'model' approach makes the physically reasonable approximation that the atoms relevant to most interesting biochemistry can be treated as uniformly deformable inducible units described by a single parameter:α(essentially α_{ij} reduced to a diagonal matrix with equal elements). The advantage of the 'chemical' approach is that there will be far fewer interactions to consider. However the complexitity of the underlying mathematics makes the 'model' approach our present choice. In this method the interaction of the diagonal polarizabilities with each other and the point charges gives rise to induced dipoles at each atomic center.

The groundwork for this 'model' method was performed in the laboratory of J. B. Applequist [36], who studied the molecular rotation of light. That group described a small set of α's for hydrogen, carbon, nitrogen and oxygen in a variety of chemical environments which reproduce the observed molecular rotations [37] as well as the circular dichroic spectra of polypeptides [38]. The Kollman group recognized early that this effect was important and the effect of the induced polarization was studied by Kollman and Lybrand in 1985 [39]. However, it was only possible to determine derivatives of the induced polarization term by numerical methods until the work of Ahlström [40] found reasonable approximations that allowed the derivation of analytical derivatives of the energy arising from the interaction of the induced dipoles.

Sprik and Klein [41] have described an alternative approach to representing polarizabilities wherein the inducible moments are represented by positionally fluctuating (in a self-consistent interaction scheme) 'mini-charges' positioned about the 'center' of a rigid molecular unit. This approach is computationally somewhat simpler and works well for water but is is likely to be difficult to parameterize for general molecules. Straatsma and McCammon [42] have suggested a non-iterative scheme to include some of the affect of polarizability which only allows the induced moments to interact with the fixed charges and not each other.

The inducible dipoles are fit into the AMBER approach as shown below

$$E_{\text{total}} = \sum_{\text{bonds}} K_r(r - r_{\text{eq}})^2 + \sum_{\text{angles}} K_\theta(\theta - \theta_{\text{eq}})^2 + \sum_{\text{dihedrals}} \tfrac{1}{2}V_n[1 + \cos(n\phi - \gamma)]$$
$$+ \sum_{i<j}^{\text{atoms}} \left(\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\varepsilon R_{ij}} \right) + \sum_{i}^{\text{atoms}} (\boldsymbol{\mu} \cdot \boldsymbol{E}_i), \tag{2}$$

where:

$$\boldsymbol{\mu}_i = \alpha_i \boldsymbol{E}_i, \tag{3}$$

$$\boldsymbol{E}_i = \boldsymbol{E}_i^0 + \sum_{i=1_{j\neq i}} T_{ij}\boldsymbol{\mu}_j, \tag{4}$$

$$E_i^0 = \sum_{j=1_{j\neq i}} q_j \frac{\boldsymbol{r}_{ij}}{r_{ij}^3}, \tag{5}$$

$$T_{ij} = \frac{1}{r_{ij}}\left[3\boldsymbol{r}_{ij}\left(\frac{\boldsymbol{r}_{ij}}{r_{ij}^2}\right) - 1\right]. \tag{6}$$

In practice there are two methods to determine the induced fields: (1) Perform a matrix inversion of the last term in Eq. (2) and (2) from an initial guess of no induction, iterate until a stable value of the induced field is reached. Method (1) is exact; however (2), is faster, according to Applequist [43], as well as adequate in the light of all the other approximations that go into the classical approximation. We have implemented method 2.

The method of induced polarization was implemented in AMBER by Caldwell and Dang [44] following the Ph.D. thesis of Ahlström [40]. The approximation of a uniform diagonal inducible dipole was the final outcome of extensive tests (method 2 above). The program implements

$$\alpha_{\text{atomic}} = \begin{pmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{pmatrix},$$

which allows a low level verison of method 1 if the user wishes to do minor tinkering to the setup code.

The addition of self consistent atomic polarization (SCAP) to the normal additive Hamiltonian is implemented in a very simple manner for the user. One merely needs to specify atomic polarizabilities for each atom type in the input to PARM in the same lines as the atomic masses and set the 'POLA' flag on line 2 of the input. In Sander, Gibbs and Nmode the user then sets 'ipol = 1' (the default is 0).

SCAP is certainly not free; it typically requires 3.5–5 times as much CPU time as the additive Hamiltonian. In a typical additive molecular dynamics or minimization run on a liquid of small molecules (where SHAKE time is typically negligible), approximately 90% of the CPU time is consumed in the nonbonded energy evaluation. For the same system with SCAP included, the CPU time breakdown is approximately: nonbond energy and derivative evaluation 12–15%; SCF induced field determination 53–57%; and SCAP derivative evaluation 27–30%. This is for converging to a 0.01 RMS difference in the induced electric field. We feel that this in the loosest tolerance that is acceptable; the convergence is very rapid, usually only requiring four steps, with the final RMS much below the cutoff. SCAP is expensive for two somewhat related reasons: (1) there is simply much more arithmetic to do; (2) the arithmetic is completely scalar. It is possible that new alogrithms for matrix inversion written with hardware-specific tricks could actually help with the SCF portion of the calculation; this is on the short list of things to do. It will be a considerable challenge to speed up the derivatives.

5. SANDER

SANDER is the main AMBER module which carries out minimization, molecular dynamics, and NMR refinements. The acronym stands for Simulated Annealing with NMR-Derived Energy Restraints, but in fact SANDER is perhaps most often used for things not related to NMR. The two principal

algorithms contained in the code are a conjugate-gradient procedure to minimize the molecular mechanics energy, and a molecular dynamics procedure that uses the leap-frog Verlet scheme to propagate the trajectory. We first discuss some of the general features of the code, followed by a discussion of the NMR-specific portion.

5.1. Energy functions

SANDER provides direct support for the AMBER [7,8,32] and AMBER/OPLS [45] force fields for proteins and nucleic acids, and for the TIP3P and TIP4P models for water [45,46]. Other types of parameters can be entered, and the code allows some variation in functional form as well as in parameters. These variations include alternate functions for 'improper' torsions and Urey-Bradley interactions, so that force fields like that of version 22 of CHARMm [47] can be implemented. In addition, 'non-additive' force fields based on atom-centered dispole polarizabilities can be included, as described in Section 4.

The relative weights of various terms in the force field can be varied over time. In addition, it is also straightforward to implement a 'geometric' force field, in which bonds and angles are kept fixed, torsions are free and non-bonded interactions consist solely of 'soft' interactions to prevent steric overlap. This sort of potential can be useful when major conformational changes are anticipated, or when one is concerned that errors in the more realistic atomic potentials are biasing the results.

Two geometries of periodic boundary conditions are included: the closest-image convention is used within either a rectangular parallelopiped or a truncated octaherdron (box with corners chopped off). Nonbonded interactions are computed for the closest image as long as the distance is below a user-based cutoff. These cutoffs are computed on a per-residue basis, i.e. if the distance between any pair of atoms falls below the cutoff, all atom-pairs between the corresponding residues are included; for neutral residues, this ensures that dipoles are not 'split', with one part inside the cutoff and the other outside. Atom pairs lying between the cutoff and a second, larger cutoff can be evaluated at less frequent intervals, and have their contributions to the forces and energy remain constant until the next time they are updated, thus implementing a very simple version of a multiple-time-step algorithm.

5.2. Energy minimization and molecular dynamics

Energy minimization in SANDER uses a conjugate gradient procedure [48], usually following a few steps of steepest descent to ameliorate very bad initial energies. Although these procedures can be effective at finding local minima, it can often take many iterations to find accurate minima, i.e., conformations where the root-mean-square of the elements of the gradient is less than 10^{-3} kcal/mol-A^2. For these problems it can be advantageous to use the Newton–Raphson or truncated Newton (TNPACK) routines [49] that are implemented in the nmode module. It is also increasingly common to use dynamical simulated annealing or other more 'global' techniques to find low-energy structures.

The molecular dynamics scheme in AMBER basically follows ideas first developed in the GROMOS programs [11,50]. This uses a 'leap-frog' version of the Verlet algorithm, in which the positions are determined at times t, $t + \Delta t$, $t + 2\Delta t$,..., and the velocities at time $t - \Delta t/2$, $t + \Delta t/2$, $t + 3\Delta t/2$,... Bond length constraints are maintained with the SHAKE procedure, using an interative procedure close to that originally described [51] for bonds in the protein or nucleic acid, and a special analytical version for 3-point water [52].

Temperature regulation can either be carried out by periodic scaling of velocities, or by a 'weak coupling' procedure, where at every step the velocities are slightly modified by a factor that depends upon the instantaneous and target temperatures [50]. This procedure is designed to have the system temperature approach the target temperature in an exponential fashion, with a time constant set by the user,

usually in the range of 0.5 to 2 ps, although longer values can be used for cooling steps in a simulated annealing cycle. In periodic boundary condition simulations, pressure regulation can be handled in a similar way, by scaling the coordinates of all atoms by a small amount at each step, where the scaling factor is determined by the ratio of the current internal pressure to the target pressure. In this way, the size of the simulation box can vary until the appropriate density is achieved. Although these procedures are not guaranteed (or designed) to generate populations that exactly match conventional thermodynamic ensembles, the observed fluctuations in temperature and pressure (which can be sizeable for small simulation systems) are close to those expected for the conventional ensembles, and the modifications to Newton's equations are relatively small, especially for long relaxation times. Other temperature and pressure regulation methods are available in the SPASMS program.

Two additional and optional modifications of the molecular dynamics algorithm are provided. The first implements the PEACS routine, which attempts to force the trajectory to follow contours of constant potential energy [53]. If the target contour is slowly lowered, this can be an effective alternative to simulated annealing to find low energy regions of conformational space. A second modification limits the absolute magnitude of any component of the velocity at each step. This cutoff value is placed well above the main portion of the Maxwell–Boltzmann velocity distribution, and serves to prevent in-stabilities in the SHAKE procedure that can occasionally occur when integration errors allow two atoms to approach each other too closely. We find this velocity limitation procedure to be helpful in eliminating problems with SHAKE, particularly when performing NMR refinement.

5.3. NMR-based restraints

The ways in which experimental restraints based on NMR data can be used to generate solution structures is a topic too large to be considered in any detail here. In addition to general reviews [54–58], several papers have outlined the particular approach taken in the AMBER codes [57,59,60]. For the 'basic' types of distance and angle constraints, we have tried to make the specification as flexible as possible, and to make the input as natural as possible for practicing spectroscopists. SANDER also incorporates some less-commonly-used constraints, including time-averaged distance and angle constraints, and penalty functions based on relaxation-matrix estimates of NOESY intensities or on chemical shifts.

5.4. Distance and angle penalties

The most common approach at present is to interpret NOESY intensities as distance constraints and coupling constants in terms of dihedral angle constraints. In sander, the user can define internal restraints on bonds, valence angles, and torsions, and the force constants and target values for the restraints can vary during the simulation. Restraints can also be imposed directly in terms of Karplus-like relations for torsion angles [61,62]. The penalty function can consist of as many as three types of region: it can be flat between one set of upper and lower bounds (called r^2 and r^3); then rise parabolically when the internal coordinate violates these bounds; and finally, since large violations may lead to excessive parabolic penalties, these parabolas can smoothly turn into linear penalties outside even wider upper and lower bounds (called $r1$ and $r4$). The imposition of restraints can be made dependent upon the distance that residues are apart in the amino-acid sequence, so that much of the functionality of programs like DISMAN [63] is available.

Internal restraints can be defined to be 'time-averaged', that is, restraint forces are applied based on the averaged value of an internal coordinate over the course of the dynamics trajectory, not only on its current value [62,64–68]. Time-averaged bonds and angles are calculated as

$$\bar{r} = (1/C)\left(\int_0^t e^{(t'-t)/\tau} r(t')^{-i} dt'\right)^{-1/i}, \tag{7}$$

where $\bar{r}$ is the time-averaged value of the internal coordinate, τ is the exponential decay constant, $r(t')$ is the value at time t'. Usually i is set to 3 or 6 for NOE distances, and -1 for angles and torsions (linear averaging). Time-averaged torsions are calculated as

$$\langle\phi\rangle = \text{atan}\,(\langle\sin(\phi)\rangle/\langle\cos(\phi)\rangle), \tag{8}$$

where ϕ is the torsion, and $\langle\sin\phi\rangle$ and $\langle\cos(\phi)\rangle$ are calculated using the equation above with sin $[\phi(t')]$ or cos $[\phi(t')]$ substituted for $r(t')$. Forces for time-averaged restraints can be calculated either of two ways. In the first,

$$\partial E/\partial x = (\partial E/\partial \bar{r})(\partial \bar{r}/\partial r(t))(\partial r(t)/\partial x) \tag{9}$$

(and analogously for y and z). The forces then correspond to the standard flat-bottomed well functional form, with the instantaneous value of the internal replaced by the time-averaged value. In the second option, forces are calculated as

$$\partial E/\partial x = (\partial E/\partial \bar{r})(\partial r(t)/\partial x). \tag{10}$$

Integration of this equation yields a non-intuitive expression for the energy (although one that still forces the bond to the target range). The reason that it may sometimes be preferable to use the second option is that the term $\partial \bar{r}/\partial(t)$, which occurs in the exact expression (Eq. (9)), varies as $(r/r(t))^{1+i}$. When $i = 3$, this means the forces can be varying with the fourth power of the distance, which can possibly lead to very large transient forces and instabilities in the molecular dynamics trajectory. The best ways to handle time-averaged restraints is a subject of active research [62, 64–68].

5.5. Penalties based on NOESY volumes or chemical shifts

Refinement directly against measured NOESY and chemical shift restraints in the newest and most specialized functionality of the SANDER module. These can be used in either in single-point mode, to provide a 'back calculation' of spectra for an input structure. The basis relaxation matrix approach provides an estimate of multi-spin effects in cross-relaxation experiments [58]. These use the Solomon equations, which arise by projection of more general equations of motion onto the operator space spanned by the z-components of the individual spins:

$$dm(t)/dt = -Rm(t), \tag{11}$$

where m gives the deviation of the spin populations from their equilibrium values, the off-diagonal elements of R are the cross relaxation rates, and the diagonal elements represent the self-relaxation rates of each spin. After a mixing time τ_m, the NOESY cross-peak volume or integrated intensity is thus

$$I_{ij} = \exp\,(-R\tau_m)_{ij}. \tag{12}$$

Details of the ways in which the intensities and their derivatives are solved are presented elsewhere [60]. For a rigid molecule, the elements of the rate matrix R depend upon the inverse sixth power of the distance between spins, and on the overall rotational tumbling time, which is often assumed to be isotropic. In addition to overall rotational tumbling, internal motion and conformational disorder can contribute to the observed cross-peak intensities. SANDER allows one to estimate these effects using

data provided from separate molecular dynamics [69] or normal mode calculations [70], or from 'master equations' in which alternate conformations and transitions between them are specified [60,71]. In the dynamic refinement mode, the computation can are included in penalty functions that depend upon $I - I_{obs}$, where I_{obs} is the experimentally measured value, and I is the value corresponding the current conformation. The best functional form for this sort of penalty function is still a subject of active research. Many details of the SANDER algorithms for relaxation matrix calculations are given in separate review [60]. Chemical shift calculations are based on a combination of ring-current, magnetic anisotropy and electrostatic interactions [72,73]. Briefly, structural chemcial shifts for protons attached to carbons in proteins can be estimated by:

$$\delta_{protein} - \delta_{randomcoil} = \delta_{rc} + \delta_{m} + \delta_{el} - \delta_{const}, \tag{13}$$

where the experimentally measured proton chemcial shifts protein are referenced to the corresponding values in the random coil state. One the right-hand side, the protein shifts are approximated by the calculated ring current (rc), magnetic anisotropy (m), and electrostatic (el) contributions of various aromatic and polar groups in the protein. A penalty function can then be designed to minimize the difference between calculated and observed shifts. An initial application to myoglobin illustrates some of the prospects for this sort of analysis [74].

6. GIBBS

GIBBS is the AMBER module that is used to perform free energy calculations. Free energies are calculated using statistical mechanical equations that relate an emsemble depending on potential energy to a free energy difference. The necessary ensemble average is evaluated within GIBBS using molecular, dynamics. One can think of GIBBS as an elaborate molecular dynamics program, with a tremendous amount of additional code included to perform a wide variety of free-energy-specific tasks.

In the most common application, GIBBS calculates the free energy difference, G, between two states '0' and '1':

$$\Delta G = G_1 - G_0. \tag{14}$$

State 1 is defined in the AMBER PREP module and state 0 is defined in the PARM module. For reasons that will be discussed below, the free energy difference is calculated in a series of incremental steps which connect physical states 1 and 0 through a not necessarily-physical intermediates. The character of the system at each of these intermediate steps is related to a parameter λ. For the bond, angle and torsional terms, the force constants, K, and target values, T, are linearly interpolated as a function of λ:

$$K(\lambda) = \lambda K_1 + (1-\lambda)K_0, \tag{15a}$$

$$T(\lambda) = \lambda T_1 + (1-\lambda)T_0. \tag{15b}$$

The non-bonded interactions are defined by scaling the individual charges (q), well depths (ε) and van der Waals interaction distances (r^*):

$$q(\lambda) = \lambda q_1 + (1-\lambda)q_0, \tag{15c}$$

$$\varepsilon_{mix}(\lambda) = \lambda \varepsilon_{mix,1} + (1-\lambda)\varepsilon_{mix,0}, \tag{15d}$$

$$r^*_{mix}(\lambda) = \lambda r^*_{mix,1} + (1-\lambda)r^*_{mix,0}. \tag{15e}$$

The Lennard-Jones non-bonded coefficients A and B are defined in terms of ε and r^* as:

$$A = \varepsilon_{mix} r^{*12}_{mix}, \tag{15f}$$

$$B = 2\varepsilon_{mix} r^{*6}_{mix}. \tag{15g}$$

The hydrogen bond (10–12) interaction coefficients are defined in an analogous fashion.

6.1. Free energy techniques available in GIBBS

There are several techniques available in GIBBS/AMBER 4.1 for evaluating the free energy difference between two states, all based on various statistical mechanical relationships. (For a general introductory review of free energy methods, see Ref. [75]). The methods offered by GIBBS include:

(1) Free energy perturbation (FEP): The free energy is calculated at discrete and uniformly spaced intervals of λ using the formulae:

$$G_{\lambda(i+1)} - G_{\lambda(i)} = -RT\ln\langle\exp - [(V(x)_{\lambda(i+1)} - V(x)_{\lambda(i)})/RT]\rangle_{\lambda(i)}, \tag{16}$$

$$\Delta G = G_1 - G_0 = \sum_i G_{\lambda(i+1)} - G_{\lambda(i)}, \tag{17}$$

where G_0 and G_1 are the free energies of states 0 and 1, respectively, $V_{\lambda(i)}$ is the potential energy function representative of state $\lambda(i)$, and $\langle\rangle_{\lambda(i)}$ means use the ensemble average of the enclosed quantity, representative of state $\lambda(i)$. The ensemble is evaluated from an MD trajectory run with $V = V_{\lambda(i)}$ The user specifies the numbers of steps of equilibration and data collection for each $\lambda(i) \rightarrow \lambda(i+1)$ 'window'.

(2) Thermodynamic integration (TI): Instead of Eqs. (16) and (17), we use

$$G_1 - G_0 = \int_0^1 \langle \partial V(x,\lambda)\partial\lambda\rangle_\lambda d\lambda \tag{18}$$

to calculate the free energy difference. The integral is approximated by evaluating the integrand a at a series of discrete values of λ, followed by integration using a standard algorithm, such as the trapezodial rule.

(3) Slow growth: This is the limiting case of both FEP and TI where an extremely large number of windows is used. The assumption is that λ changes slowly enough that the system remains in equilibrium at every point, so that no equilibration and only one step of data collection is required at every window to evaluate the ensemble. Under these conditions, both Eq. (16)–(17) and Eq. (18) reduce to

$$\Delta G = \sum_i V(x)_{\lambda(i+1)} - V(x)_{\lambda(i)}. \tag{19}$$

(4) Dynamically modified windows: [76] The equations of FEP (16) and (17) are used as described for method 1 above. But instead of using pre-chosen uniformly-spaced intervals of λ, the width $\Delta\lambda = \lambda(i+1) - \lambda(i)$ of each window is determined during the run, based on the recent value of the slope, $\partial G/\partial\lambda$, of the accumulated free energy versus λ curve. This allows the simulation to be run more 'slowly' when the free energy is changing very quickly, and more 'quickly' when it is not.

(5) Dynamically modified thermodynamic integration: Uses the same λ adjustment algorithm as for FEP (method 4), but the intervals in λ correspond to the points at which the integrand in Eq. (18) is evaluated to approximate the integral.

(6) Potential of mean force (PMF) calculations: The user can elect to constrain any chosen set of internals (distances, angles, torsions) to a chosen lambda-dependent path-way. By selecting the appropriate option, the contribution to the free energy from such constraints will be calculated. This constitutes a PMF calculation. PMF calculations can be carried out in conjunction with any of the methods described above.

6.2. Free energy simulation options

Gibbs allows one to carry out a large number of types of calculations, and offers substantial flexibility to define how a particular calculation type will be performed. The following section briefly outline some of the options available to the user for these simulations. For the most part, options which are not

specific to GIBBS and free energy calculations–that is, options which primarity control how the MD simulation trajectory is generated–are not described here. Some of these are discussed in the description of the SANDER module.

6.3. Basic control options

These options determine the basic method used for the free energy simulation, how long the simulation will be run, and in what direction ($\lambda = 0 \rightarrow 1$ or $\lambda = 1 \rightarrow 0$). These are the most basic questions that one must answer before running a free energy simulation. The five methods (FEP, TI, slow growth, and the 'dynamically modified' variants of FEP and TI) have their relative strengths and weaknesses:

FEP: The equations of the window growth, or free energy perturbation (FEP) are exact, and, in principal, if one has the computer resources to perform sufficient sampling, one can obtain very accurate results. In practice, FEP suffers from two significant difficulties. The first is that, in reality, we do not always sample to convergence. Unfortunately, no reliable test to prove convergence has been developed. The second problem with FEP is that Eq. (16) requires that we obtain the ensemble average of a quantity which relies of the *difference* between the potential functions representative of both states $\lambda(i)$ and $\lambda(i+1)$. But the average is evaluated from the ensemble of states visited when MD is run using the potential function for state $\lambda(i)$. Thus, if states $\lambda(i)$ and $\lambda(i+1)$ are too dissimilar, it will be very very difficult to obtain reliable statistics. This has been termed the 'orthogonality problem'. Reducing the spacing between adjacent λ states helps circumvent this problem, but at a significant additional cost. And even then we do not have any reliable methods for assuring the problem has been avoided [77].

TI: TI (Eq. 18) is appealing because it avoids the orthogonality problem inherent in FEP. But TI has its own potential problem: The driving equation of TI is an integral, which in practice must be calculated approximately by evaluating the integrand at finite intervals of λ. Fortunately, for the most free energy simulations the accumulated free energy *versus* λ curve is slowly enough varying that an accurate estimate of the integral can be obtained from a modest number of integration points [77]. Of course, TI is also susceptible to errors when a simulation is not run sufficiently long to obtain a converged value of the averaged quantity which serves as the integrand. Recent studies have indicated that TI will generally be at least as efficient as FEP, and will be considerably more efficient in some cases [78].

By default, within GIBBS the integral is approximated from a discrete set of evaluated integrand points using the trapezoidal algorithm:

$$\Delta G_i = G(\lambda(i+1)) + G(\lambda(i)) = (\langle \partial V/\partial\lambda \rangle_{\lambda(i+1)}) - \langle \partial V/\partial\lambda \rangle_{\lambda(i)}(\lambda(i+1) - \lambda(i))/2. \tag{20}$$

This integration method should be reasonably accurate in most cases. If the user wishes to try their own integration scheme, GIBBS can also be forced to report the integrand values necessary to do so.

Slow growth: Some early studies indicated that slow growth might be a more efficient technique for free energy calculation than fixed-width windows. [79] More recent work [80] has indicated that the implicit assumption of slow growth – that λ changes slowly enough that the system can be assumed to be in equilibrium at each step – does not strictly hold. The consequences of this 'Hamiltonian lag' have not been fully quantified. Similarly, the assertion that the ensemble can be approximated by a single snapshot has been questioned.

Dynamically modified windows (DMW): In dynamically modified windows [76], the $\Delta\lambda$ spacing between consecutive windows in FEP or TI is continually changing, to achieve a relatively constant free energy change per window. It has been demonstrated for model compounds that this can improve the efficiency of some calculations, by focusing proportionately more simulation time on those ranges of λ where the free energy is changing more rapidly [76]. However, the usefulness of this approach will depend–at least

in part–on whether the variance in the average being accumulated (Eq. (16)–(17) and Eq. (18) is correlated to the new value of ΔG over a window [81]. These quantities will sometimes – but not always – be significantly correlated [82].

For the FEP/TI methods, a significant question remains as to how much equilibration and how much data collection should be carried out at each window. While no general prescription can be made, recent work has indicated that a minimum of 0.5–0.7 ps of sampling is required in model systems to obtain one statistically distinct observation at each window [82]. This should serve as a lower-bound on sampling per window for such methods.

6.4. Options that should increase the accuracy of the calculation

It has been demonstrated that the contributions from bond stretching should be included in any free energy simulation [83]. In the same work, it was shown that these contributions can be easily determined by constraining the bonds and then calculating the contributions to the free energy due to the constraints (the 'PMF bond contribution'). The procedures for calculating these contributions in FEP simulations [83] and in TI simulations [84,85] have been described and have been implemented in GIBBS. Note that for TI, two different methods have been outlined: The 'potential force' (PF) method [85] and the 'constraint force' (CF) method [84]. PF has the advantages of analytical exactness but the disadvantage that it cannot be applied to λ-dependent bonds within a closed ring. CF has the advantage of being applicable to all bonds, but the disadvantages that it requires numerical approximation of a derivative, and that the CF method requires a second simulation to be performed to correct for center-of-mass changes during the simulation of interest.

In addition to the PMF bond contribution, it has similarly been shown that it is often important to include *all* contributions to the free energy in a calculation [83,86]. In some early free energy work, intra-perturbed-group contributions were omitted under the assertion that they would introduce noise into the calculation and would cancel out in the net free energy cycle. Gibbs now allows the reporting of all appropriate contributions.

6.5. Options that may improve the convergence profile

When a charged atom 'vanishes' at an endpoint of a simulation, and MD trajectory instability can sometimes be encountered due to a finite charge on an atom with a disppearingly small van der Waals interaction parameter. When this is a problem, one can use the method of 'electrostatic decoupling' [17] within GIBBS. In electrostatic decoupling, the atomic charges are removed from the atoms in one free energy simulation which can be followed by a second simulation where the van der Waals parameters are removed.

GIBBS also allows the user to control the mixing rules for 'vanishing' atoms. By default (and without exception in pre-4.0 versions of GIBBS), the optimal interaction r_{ij}^* between two atoms i and j is given by

$$r_{ij}^*(\lambda) = r_i^*(\lambda) + r_j^*(\lambda). \tag{21}$$

This is fine when neither atom 'vanishes' at either λ endpoint. But now consider the case where atom i vanishes at $\lambda = 0$. Then

$$r_{ij}^*(0) = r_i^*(0) + r_j^*(0) = r_j^*(0). \tag{22}$$

Thus, r_{ij}^* never gets smaller then $r_j^*(0)$. At $\lambda = 0$, the mixed well depth, $\varepsilon(0)$, will also be 0. But at any value of λ just slightly > 0, $\varepsilon \neq 0$, and suddenly a steric 'gap' between atoms i and j of r_j^* will be

required. This can lead to sampling inefficiencies. A better solution is to shrink $r_{ij}^{*}(\lambda)$ to a user-chosen small value as one of the atoms 'vanishes' [76].

Finally, GIBBS allows the user to use the method of dynamically modified windows (DMW). In DMW, the slope of the ΔG versus λ function over the next window, $M_{i+1} = \partial G/\partial\lambda|_{\lambda=\lambda(i+1)}$, is estimated using the slope over recent windows. This slope approximation is then used to modify $\Delta\lambda$ on the next window to keep the free energy per window approximately constant, using the relationship

$$\Delta\lambda = \Delta G_{\text{target}}/M_i \tag{23}$$

These parameters allow you to determine how many points are used in approximating the slope, the minimum acceptable correlation coefficient for the fit, the minimum and maximum allowable step sizes, the target free energy per window, ΔG_{target}, and a number of more minor parameters related to controlling DMW. With these parameters, one can also specify different values of ΔG_{target}, and of the minimum and maximum λ moves, for different ranges of λ.

6.6. Options that allow more detailed output and analysis

In addition to the total free energy, within GIBBS one can decompose the free energy into the contributions from various atoms or groups in the system, and from different terms in the force field. A number of recent studies have attempted to draw additional insights from such component analysis [87,88]. However, it must be noted that the free energy components unlike the total free energy, do not represent state functions. And, in addition the components may converege much more slowly than the total free energy. This type of analysis, therefore, must be carried out with caution [89,90].

It is also possible to request the calculation of free energy derivatives. Free energy derivatives [82,85] are terms of the form $\partial G/\partial Q_i$, where G is the free energy, and Q_i is any parameter of the system [91]. These derivatives are calculated as

$$\partial G/\partial Q_i = \langle \partial V/\partial Q_i \rangle_{Q_i} \tag{24}$$

and are directly related to the net $\langle \partial V/\partial\lambda \rangle_\lambda$ required for TI simulations through the chain rule. These free energy derivatives have two valuable applications. First, they can be used to discern the rate of convergence of a free energy simulation through straightforward statistical analysis. Second, they may be useful for drug design. The free energy derivatives can be used to suggest changes in the molecule that will – at least in the instantaneous limit – have the desired change in the net free energy of the system. [91] Whether one can proceed from these suggestions to an actual molecule with the desired change in free energy properties remain to be proven [92].

6.7. Options that allow more sophisticated types of calculations

GIBBS allows the user to request that the enthalpy and entropy changes be reported in addition to the free energy (which is always reported). The method used to calculate the entropy (and dependent enthalpy) depends on whether a FEP or TI simulation is being performed. Note that, in either case, the expressions for the enthalpy and entropy include terms which depend on the *total* energy of the system (which is often very large). This differs from the expressions for the total free energy change, which depend only on the changes in free energy for terms that change with λ. For this reason, enthalpy/entropy estimates are typically more than an order of magnitude less accurate than their free energy countepart [90,93]. One should be very cautious when interpreting them. A free energy simulation where the entropy/enthalpy are determined takes no more cpu time than one where only the net free energy is calculated. But because the entropy/enthalpy values are so prone to inaccuracy, they are only reported when the user explicity requests them.

For FEP, the approximate equations derived in Ref. [93] are used. These approximate the required temperature derivatives by a finite difference. The equations used are derived from the FEP expression. For TI, the enthalpy and entropy are evaluated using the exact-form integral relationships presented in Ref. [94].

For a GIBBS run, the user may define additional restraints or constraints on the system. As many added re-/con-straints may be specified as are desired. Each definition specifies the distance, angle or torsion to be affected, whether a (harmonic) restraint or a holonomic constraint [51,95] is desired, and the values of the internal at the two end-points of the simulation. The free energy contributions from a constraint are calculated when requested. A potential of mean force (PMF) calculation, wherein one determines the free energy profile as a function of an internal coordinate, can be determined by adding an internal constraint and calculating the free energy contributions. For restraints one can specify that the restraint is either to be considered as part of the force field, or to be considered as an 'umbrella' term.

The added constraint definitions also allow one generate information that could be used to define a 2-dimensional free energy calculation. In such a calculation, a free energy map is determined as a function of two independent internal coordinates [96].

6.8. Changing parameters versus dual topologies

In 'standard' operation, free energy changes in GIBBS are effected by transforming the potential representative of state 1 to that representative of state 2. The topology of the system does not change. To make atoms non-interacting at one of the endpoints, they are assigned zeroed non-bond and electrostatic parameters at this endpoint. This has been termed the 'single topology' approach.

The improved mixing rules included in GIBBS (version 4.0 and newer) allow a second method to be used. One result of these new mixing rules is that if any pair of atoms 'exist' only at mutually exclusive endpoints (e.g. atom i exists in state 1 but no state 2; atom j exists in state 2, but not in state 1), then effectively no non-bonded interactions are ever calculated between them. This means that, in lieu of the 'standard' method which uses a single topology, we can define dual topologies, one corresponsing to the $\lambda = 0$ endpoint, and the other corresponding to the $\lambda = 1$ endpoint. For the fomer topology, all non-bonded parameters would be defined to be 0 in the $\lambda = 1$ state. Similarly, all non-bonded parameters for the latter topology would be 0 at $\lambda = 0$. The two toplogies would then never 'see' each other at intermediate values of λ.

Defining dual topologies makes it possible to calculate a free energy difference for a system where the *topology* of a closed ring changes (e.g. the difference between systems with purine and pyrimidine nucleic acid bases). Such systems are not amenable to single topology calculations. A recent study [78] has indicated that for simulations where both 'single' and 'dual' topology approaches are applicable, the 'single topology' approach typically used with AMBER may be more efficient than the 'dual topology' approach frequently used with CHARMm. For a detailed example of AMBER used for a 'dual topologies' calculation, see Ref. [78].

6.9. Other features of GIBBS

Apart from the specific free energy capabilities described above, GIBBS includes the following features:

(1) Full ability to carry out MD simulations with or without periodic boundary conditions, at constant volume or constant pressure, and (optionally) with inclusion of polarization.
(2) Analytic SHAKE for 3-atom molecules. Allows a significant speedup in SHAKE for TIP3P waters.

(3) Interactions which change from hydrogen-bonding type (handled by a '10–12' potential) to non-hydrogen-bonding type (handled by a '6–12') potential are smoothly and properly handled.
(4) Special code to appreciably speed up non-bonded interaction calculations among TIP3P waters.
(5) Both single and dual-range non-bonded cutoffs supported. A separate cutoff for the perturbed group atoms is also supported.
(6) An abbreviated logfile, MICSTAT, which contains the free energy trajectory in a concise, easy-to-read format.
(7) File assignment can be performed either with command line flags (under Unix), or from a machine-independent assignment file.

7. SPASMS

SPASMS was orginally developed as a second generation simulation package to the AMBER MINMD and GIBBS program modules. The acronym stands for the San Franciso program of applications for the simulation of molecular systems [24]. The program is, in fact, a 're-tooling' of the basic code and algorithms found in the AMBER programs, sharing similar force fields, datafile structures and many options. Although SPASMS had earlier been viewed as a possible replacement to the minimization and dynamics based modules of AMBER, the program has more evolved as a complement that allows users to try alternative methods to study computational problems. In addition, the code structure is designed to be user friendly, with most nomenclature borrowed from the popular textbook on simulation by Allen and Tildesley [97]. The program therefore provides an excellent basis for learning the computational techniques and methods implemented in these codes as well.

The revised code structure, however, is just one of the features of the SPASMS program. In reworking the code and algorithms the authors of SPASMS also took the opportunity to add features not then available in the molecular dynamics modules MINMD and GIBBS (AMBER 3A). This was done to incorporate alternate methods and techniques regarded to be more robust for molecular simulations. Once again, the text by Allen and Tildesley served as a model from which the majority of algorithms were adapted. The text is therefore regarded as a major reference for the SPASMS code and functionality. Besides internal components, SPASMS was also modified to use an updated user interface. Although this was originally intended to be generated graphically (via the LEaP interface), the resulting keyword driven format is nevertheless a valuable aspect of the SPASMS program.

In the sections that follow, we will discuss the features of SPASMS not currently available in the GIBBS or SANDER programs. It is also important to point out that in SPASMS, the code is still relatively primitive, which has both pros and cons. The basic structure and functions supported may be attractive to researchers interested in algorithm development and/or code modifications, but the limited options of SPASMS may be frustrating to applications-minded users. In any case, the program does offer an alternative approach to calculating trajectories, free energies, and conformers; and provides a valuable tool to evaluate methodologies and results as well as a different approach to solving difficult simulation problems (as discussed further below).

7.1. Energy conservation

One of the main objectives addressed in the development of SPASMS was energy conservation. In the other AMBER programs, non-bonded interactions are abruptly truncated if the closest intra-residue atomic distance exceeds a predetermined cut-off distance, R_{cut}, which is typically set to values near the third minimum in the radial distribution function of water (about 8.0–9.0 Å). This practice is commonly

used in other program packages as well. Truncation has the beneficial effect of reducing the problem size, or non-bonded pairlist, in large, complex systems. Physically, however, this approach can have rather severe consequences [98]. By truncating the potential energy, the function defining the system is no longer continuous through the cut-off distance. While the energy is properly defined as residues' closest atoms cross R_{cut}, the interatomic forces $\partial E/\partial R_{(ij)}$ are not, and are simply 'truncated' as well. The force is therefore inconsistent with the behavior of the energy, leading to poor energy conservation in the equations of motion.

Although this produces a number of physical effects in the system, the most obvious and well-known is heating. The system continually gains energy that must be removed if the finite difference equation is to remain stable. The heating is due to the migration of particles in and out of the cut-off distance. Since molecules exit the cut-off correlated, and re-enter decorrelated, the net effect is a gain in the potential energy of the system. In the other AMBER programs, this problem is handled using Berendsen's velocity scaling algorithm [99]. While the method has been shown to have little effect on the other physical properties of the system, the averages calculated during the simulation do not correspond to a well-defined statistical mechanical ensemble. Furthermore, the scaling method often exacerbates the heating problem, as hot parts often get hotter while cool parts get cooler. This can be partially alleviated in GIBBS and SANDER by using separate solute and solvent temperature coupling.

The heating problem is overcome in SPASMS by applying a switching function to the potential energy to smooth interactions to zero at the cut-off. The following function is applied over the interval, r_{lower} to r_{upper}

$$S(r) = c_0 + c_1\left(\frac{r - r_{lower}}{r_{upper} - r_{lower}}\right) + c_2\left(\frac{(r - r_{lower}}{r_{upper} - r_{lower})}\right)^2 + c_3\left(\frac{r - r_{lower}}{r_{upper} - r_{lower}}\right)^3 + c_4\left(\frac{r - r_{lower}}{r_{upper} - r_{lower}}\right)^4 + c_5\left(\frac{r - r_{lower}}{r_{upper} - r_{lower}}\right)^5, \tag{25}$$

where r is the interatomic separation. The choice of coefficients is critical and must ensure that the potential energy function remains unaltered inside the lower switch distance and is thrice differentiable as well. This latter requirement is necessary by definition for all Verlet-based integration schemes. The coefficients that meet this criterion are

$$c_0 = 0, \quad c_1 = 0, \quad c_2 = 0, \quad c_3 = -10, \quad c_4 = 15, \quad c_5 = -6.$$

This switching function has been implemented in SPASMS for use with residue-based pairlists. Although this approach is sufficient to guarantee energy conservation in molecular dynamics algorithms, we have gone slightly further in SPASMS by implementing the velocity verlet method in this code, instead of the leap-frog. While all verlet algorithms produce identical trajectories, the velocity form is slightly more robust and shows better numerical stability for energy conservation. In addition, the velocity form allows the current position, velocity, and acceleration to be known simultaneously. This has advantages when coupling the system to thermal baths or external pressures using alternative schemes that may be more appropriate for specific simulations. The effect of non-bonded truncation schemes on molecular dynamics in an aqueous protein system has been examined by Guenot and Kollman [100].

Since SPASMS addresses the heating problem, it is possible to simulate molecular systems under constant energy conditions, regardless of system size or the existence of periodic boundary conditions. The dynamics of large macromolecules can then be explored reliably and analyzed for time-dependent properties if desired. This is particularly important if the user is interested in spectroscopic or diffusional properties. In addition, the integrity of the finite difference equation can also be checked as a function of time step [101]. To simulate canonical ensembles, Andersen's minor and massive collisions method for temperature coupling to a heat bath has been implemented [102]. This approach simulates the collisions

a macromolecule has with an external heat source by reassigning velocities randomly at some predetermined frequency that depends on the thermal conductivity of the surroundings. At one end of the spectrum, collisions can be performed sparsely yielding trajectories that are nearly constant energy, while at the opposite end, colisions can be so frequent that the trajectory is Brownian. In addition, by using massive collisions it is possible to simulate numerous constant energy trajectories that when combined produce a trajectory that is equivalent to that of a canonical ensemble. Since the system collision is performed at fairly long time intervals, dynamical properties can be derived from each microcanonical ensemble simulation during the overall process. [103]

7.2. Acceptance ratio free energy methods

Although SPASMS supports relatively few options compared to the GIBBS module of AMBER, the code does support an alternative approach to obtaining free energy estimates. While most would agree that longer sampling times correspond to more reliable estimates, it is not always possible to obtain converged, or even near-converged averages in Eq. (16). To address this problem, we have implemented a free energy algorithm based on acceptance ratio methods to optimize the estimate. This approach was first described by Bennett [104] and is discussed and compared with other methods in further detail by Ferguson [105]. Essentially, the scheme optimizes the overlap between samples from adjacent windows during the simulation by reducing the variance in the estimate. The basic AR method represents the free energy change as a ratio of ensemble averages where an arbitrary weighting function has been introduced into the configurational integrals

$$\frac{Q_0}{Q_1} = \frac{W \exp(-\beta E_0)_1}{W \exp(-\beta E_1)_0} \exp(+C). \tag{26}$$

By choosing W in such a way as to minimize the variance in the partition function ratio, the following sampling equation can be derived

$$\frac{Q_0}{Q_1} = \frac{F(\beta(C-(E_1-E_0)))_1}{F(\beta E_1 - E_0 - C)_0}, \tag{27}$$

with the requirement that

$$C = \ln\frac{Q_0}{Q_1}, \tag{28}$$

where C is a shift constant to the configurational energy and F is the Fermi function [104,105]. These simultaneous equations are then solved iteratively until C converges, minimizing the variance.

These equations have been implemented in SPASMS as part of the double-ended sampling protocol of statistical mechanical perturbation theory. Although the applications have been limited, Ferguson has shown that better estimates are possible using this algorithm given limited sample sizes and/or large molecular changes. In addition, it has also been shown that the methodology can yield information regarding the shape and overlap of the energy samples at each window, allowing the sizes to be optimized for specific systems and molecular changes.

7.3. Conformational searching

A valuable analysis tool has been incorporated in SPASMS to explore the conformational space available to molecules as a function of torsions. This method is often referred to as dihedral driving [106]. The basic protocol followed drives the torsions through predetermined intervals, generating a set of

conformers that can either be minimized or used as starting points for other refinements. Although the most familiar application of such methodology is to the energy profile of simple molecules like butane or cyclohexane, nearly any system can be examined adiabatically as a function of its torsions. While the technique is quite simple, the addition of these routines to SPASMS allows barrier heights and minima to be examined in detail as a function of the force field.

The dihedral driver option offered in SPASMS allows up to 15 torsions to be simultaneously varied, with or without minimization after each increment. In theory, a complete phi/psi map of an oligopeptide could be constructed in this manner. In addition, a variation of the code is available that performs automated conformational searches of molecules. The Monte Carlo/torsion-based approach implemented in SPASMS has been described in detail by Fergusion *et al.* and has been further shown to globally optimize several relatively large structures, including a poly-alanine fragment using the Weiner *et al.* atom force field [107].

SPASMS also supports several other features not discussed above. These include flatwell constraint methods for use with NMR data, fast water routines capable of efficiently simulating macromolecules in both TIP3P and TIP4P molecules, ensemble dynamics for simulating multiple ligand binding in drug design studies, and last but not least, the code is ANSI standard FORTRAN-77 allowing a high degree of portability across machines. Further details of functionality and applicability can be found in the SPASMS user manual.

8. NMODE

This program performs molecular mechanics calculations on proteins and nucleic acids, using first and second derivative information to find local minima, transition states, and to perform vibrational analyses. There are accompanying programs NMANAL (normal mode analysis) and LMANAL (Langevin mode analysis) that use the output of these programs to compute molecular fluctuations, time correlation functions, and NMR order parameters. NMODE was originally written at the University of California, Davis, by D.T. Nguyen and D.A. Case, based in part on code in the Amber 2.0 package. Major revisions were made at The Scripps Research Institute by J. Kottalam and D.A. Case. The second derivative routines are based on expressions used in the Consistent Force Field programs [108], and the general use of normal mode analysis in macromolecules has recently been reviewed [109]. Since these routines all require the construction and storage of a second derivative matrix (of size '$3N$ cross $3N$' for 'N' atoms^2), they become most unwieldy for systems with more than about 2500 atoms.

NMODE performs five basic tasks, depending on the value of the input variable *ntrun*:

(1) Perform a normal mode analysis starting coordinates, by constructing and diagonalizing the mass-weighted second derivate matrix in Cartesian coordinates. This requires an input structure that has already been minimized, from process (4), below, or by some other method. In addition to the computation of normal mode frequencies, the entropy, free energy and heat capacity are estimated in the rigid-rotor/ harmonic-oscillator (RRHO) approximation.

The accompanying routine NMANAL carries out analysis of the normal mode description by projecting the Cartesian normal mode eigenvectors onto internal coordinates or onto 'rigid groups'. For each internal coordinate, the program will determine the projection of each normal mode onto that coordinate, and the fraction of the total potential energy change along the normal mode that is contributed by the internal coordinate (the 'potential energy distribution' for each mode.) It will also sum over all modes to obtain thermal fluctuations for any particular internal coordinate, using either classical (Boltzmann) or

quantum (Bose–Einstein) statistics. The NMANAL program can also compute the rms thermal fluctuations of atoms in a Cartesian coordinate frame, and will compute time correlation functions and fluctuations of internal coordinates. Order parameters for NMR relaxation can also be computed; details of the connections between normal modes and spin relaxation are given elsewhere [58,110].

(2) Search for transition state, starting (generally) from a minimum. This procedure uses a modification of the procedure of Cerjan and Miller [111], which can follow normal-mode-like directions uphill to transition states. The basic approach is a modification of a Newton–Raphson minimization step to allow uphill motion along one coordinate direction while simultaneously minimizing along other directions. Our implementation includes modifications suggested by the Simons groups [112], and an account of its incorporation into AMBER and applications to peptides and other systems is given elewhere [113].

(3) Does a Newton–Raphson minimization from starting coordinates. A constant (t_λ) is added to the diagonal elements of the Hessian matrix to make it positive definite. This constant is chosen in a manner such that the step is always downhill in all directions. Whenever the change in energy is greater than some cutoff, or the root-mean-square of step length exceeds a second cutoff, the step length is scaled back repeatedly until the two cutoffs are satisfied. This is essentially a modification of the Cerjan–Miller procedure outlined above, but which converges to minima rather than transition states. It is generally an effective and robust local minimizer, even on fairly difficult surfaces, but is undoubtedly more expensive than some alternative modifications of Newton-like methods.

(4) Perform a Langevin mode calculation, starting from a minimized structure. This option is similar to (1), but includes the viscous effects of a solvent in the calculation. Langevin modes are analogous to normal modes, but in the presence of a viscous coupling to a continuum solvent. The basic ideas are presented by Lamm and Szabo [114] and applications to proteins and nucleic acids in the AMBER environment are reported in detail elewhere [115]. The LMANAL program computes time correlation functions from Langevin modes. Note that since the time-independent aspects of the molecular normal mode description are indpendent of solvent viscosity, all of the equal-time correlations (such as rms fluctuations in cartesian or internal coordinates) will be the same as for the regular normal mode calculation, so that the NMANAL routines described above would be used for these.

(5) Perform a quasi-harmonic analysis from the output of a molecular dynamics trajectory. This procedure diagonalizes the fluctuation matrix provided by the trajectory, yielding a harmonic description that reproduces as well as possible those determined form the dynamics simulation [116–119]. Those directions with the lowest effective frequencies are the most important for the overall fluctuations, and often mimic the true normal modes rather well. The analysis of these modes is then carried out using the NMANAL program.

9. Parallelization of AMBER programs

AMBER 4 is the first *distributed* version of AMBER to support some level of parallelism. In the following, we acknowledge the significant effort by numerous people to parallelize AMBER, discuss the early work on AMBER 3A and AMBER 4.0 at UCSF, then discuss the parallelism supported within AMBER 4.1.

The early work on parallelization within AMBER began with version 3A. This includs the work at UCSF by Steve DeBolt [120,121] on porting AMBER to the nCUBE/2 hypercube (discussed below) and Tom Darden's work at NIEHS on shared memory computers. With AMBER 4.0 began a larger push, with work continuing at UCSF by Steve DeBolt and later David Case on message passing versions and

work by Silicon Graphics Incorporated (SGI) and Thomas Cheatham at UCSF at developing a shared memory parallel implementation for SGI multiprocessors [122]. In addition to the work at UCSF, other groups were also independently involved in parallelizing AMBER including Terry Lybrand and Eric Swanson at the University of Washington (master/slave PVM, eventually distributed with AMBER 4.0), Ken Merz and Jim Vincent at Pennslyvania State University (MPI), Tom Darden at NIEHS, Thomas Huber at Ludwig Maximilian Universitaet in Muenchen Germany (KSR,TCGMSG), and various vendors (KSR, IBM, TMC, Cray), among others.

9.1. AMBERCUBE MD for distributed memory parallel architectures

In 1993, a fully functional, parallel MIMD (muliple instruction multiple data) version of the AMBER3A molecular dynamcis (MD) module, called AMBERCUBE, was dscribed [120]. It was developed as a data-replicated message-passing adaptation of the sequential version to MIMD architectures in general and was fully benchmarked for parallel speedup and efficiency on the nCUBE/2 hypercube architecture for up to 128 concurrent processors. Several molecular systems of varied size and complexity were used: pure liquid water, water-saturated 1-octanol, crambin/water in a periodic box, and alpha-lytic protease with a bound inhibitor surrounded by a 'cap' of waters. Subsequently, AMBERCUBE was also ported to the Intel Paragon by David Case. Those procedures which would otherwise limit overall parallel speedup (more so as a greater number of processors were added) were parallelized: the calculation of long-range nonbonded Coulomb and Lennard-Jones interactions, generation of the pairlist, intramolecular bond, angle, dihedral, 1–4 nonbonded interaction terms, coordinate restraints, and the SHAKE [123] bond constraint algorithm. Algorithmic advancements were made in the subtasking and partitioning of the work contained in an MD timestep, the dynamic load balancing of nonbonded interactions, and the parallelization of the SHAKE routine for all bond types: hydrogen-containing, heavy-atom only, crosslinked (as for protein disulfide bonds). Detailed descriptions of these algorithms are given in [120].

The tasks in the calculation of molecular interactions during each individual time step are broken down into subtasks and 'farmed out', after appropriate data decomposition and load balancing, over the number of available processors. Then, the large data array containing the intermolecular forces is updated and regathered globally on all distributed containing the intermolecular forces is updated and regathered globally on all distributed memory processors. It was found that parallel speedup scaled linearly with added processors for those procedures that did not rely on message passing. As expected, procedures that require significant amounts of interprocessor communication did not scale linearly, with efficiency typically decreasing below 70% (although stil scaling positively) beyond the addition of 16 concurrent processors. AMBERCUBE MD performed about one-third the speed of a CRAY Y-MP processor when a typical nonbonded interaction cutoff radius was used. When no nonbonded interaction cutoff was used, the interprocessor message-passing communications bottleneck did not dominate execution time. This simulated the effect of increased molecular system size or faster interprocessor communication channels. In this case, the speedup and efficiency obtained were much greater, giving nearly linear efficiency for as many as 64 nCUBE/2 processors, achieving one-half the speed of a CRAY Y-MP processor on a machine an order of magnitude less expensive.

9.2 Free energy perturbation calculations on parallel computers

In 1994, a parallel implementation of the free energy perturbation (FEP) module of the GIBBS module was described [121] and demonstrated using five progressively more challenging molecular sytems. The approach takes advantage of the underlying coarse-grain character of the FEP algorithm. Because no interprocessor communication is required, this approach scales efficiently and is applicable in general for any parallel architecture or network of machines. FEP calculations run on the nCUBE/2

using 50 or 100 parallel processors completed in clock times equivalent to or twice as fast as a CRAY YMP processor. In the process of implementing this coarse-grain approach to FEP calculations, analyses were done regarding the potential problem of maintaining system equilibrium when a gradual configurational reorientation follows the mutation of the Hamiltonian. The results of a successful protocol for overcoming this equilibration problem were presented [121].

The overal FEP task subdivides into N identical time-dependent procedures, which can be computed simultaneously rather than successively as would be required in standard time-sequential MD simulation. The total free energy change for a process can be broken down into N summable free energy increments or 'window', with each window is assignable to an individual processor. The parallel speedup obtained relative to a single processor is thus directly proportional to the number of processors applied to the problem.

The task of submitting N parallel jobs, one for each FEP window, and managing multiple file-set input/output is relegated to a UNIX C-shell command script, ensuring program generality for any multiprocessor setup. Communications between the executing image, the submission command script, and the operating directory are instituted during submission so that any redundant files are eliminated as they are generated. This prevents duplication and accumulation of large files, thereby maintaining acceptable run-time memory requirements. Input files are updated and handed off from one processor to the next as the multiple submission of N jobs progresses. Upon job completion, a command script gathers and sums the resultant window free energies to yield the overall free energy and statistics.

The critical issue for simultaneous and independent processing of free energy increments is in how 'semiequilibrated' starting states are prepared for each window. This issue has been discussed at length [121]. Briefly, two methods are offered as useful ways to overcome the equilibration dilemma: the Blocking method (breaking an overall MD/FEP trajectory into several smaller blocks, each of which uses a preceding block's endstate as its pre-equilibrated initial state), and the RAM method (generating a rapidly accessible manifold of starting states for independent processing of each free energy 'window' by periodically writing out structures and velocities at the appropriate increments, during a very quick initial perturbation).

Of the five FEP calculations carried out [121], three of them demonstrated actual overall parallel speedup 1:1 proportional to the number of concurrent processors applied and the remaining two gave cost-effectivness over a CRAY Y-MP. The difference in hydration free energy for a lithium versus a cesium ion was computed. A zero-sum perturbation of a methyl group from one side of ethane to the other was performed to evaluate the method's convergence properties. The efficient calculation of enzyme/ligand binding free energy was demonstrated for the HIV protease inhibitor bound to a peptidic inhibitor. The potential of mean force (free energy) along the reaction coordinate or the dissociation of a potassium ion from the ionophore salinomycin immersed in methanol solvent was computed in parallel. An extended (2.6 nanosecond) parallel FEP calculation of the relative octanol/water partition coefficients of phenol versus benzene gave excellent agreement with experiment. Many FEP calculations can be carried out efficiently using the coarse-grain parallel approach currently implemented in AMBER. This method promises to be of great value to computational chemists in modeling drug/receptor binding, in peptide and protein design, and in general where there is a question of how differences in biological systems.

9.3. Parallelism in AMBER 4.1

Due to the success of groups at UCSF and elsewhere at parallelizing AMBER, when planning for the release of AMBER 4.1, it was decided that supporting parallelism within the general code distribution was crucial for both SANDER and GIBBS. Deciding what to put in was not an easy task and led to difficult decisions regarding exactly what level of parallelism to support, how to make the version portable

and clean, and how to merge the excellent work contributed by the various groups. Initially it was decided to support both a shared memory parallel implementation for SGI multiprocessors [122] and a general message passing version.

9.4. Shared memory implementation

AMBER 4.1 SANDER and GIBBS support shared memory parallelization on SGI multiprocessors for many of the most time consuming parts of a typical molecular dynamics run. Specifically, this involves parallelization of the nonbond forces and energies, the creation of the residue based pairlist, SHAKE [123], and also the bonds, angles, and dihedrals. This work was performed by Roberto Gomperts (Boston) and Michael Schlenkrich (Switzerland) of Silicon Graphics Incorporated (SGI) based on AMBER 4.0 and modified and integrated into the AMBER 4.0 and AMBER 4.1 distributions by Thomas Cheatham.

Generally, parallelism is supported through the use of C$DOACROSS comments in the source code which instruct the compiler to insert code to spawn multiple threads to run the next DO loop in parallel. In AMBER, an extra DO iproc = 1, numproc loop is added around calls to various time consuming routines so that each of the numproc threads does 1 iteration of the loop. This provides a general and very simple means of implementing parallelism. The only difficulties are in figuring out for the parallel threads what variables need to be local or shared, making sure that each thread has its own scratch arrays, load balancing the time spent in each parallel thread, and accumulating the results at the end of the loop. After execution in each parallel region, the results are accumulated into the standard AMBER data structures, implying that no special handling is needed for regions that have yet to be parallelized. The first thread uses the standard AMBER scratch arrays, whereas the other threads utilize a portion of integer and real memory allocated once, as needed. This is facilitated through the of (non standard) FORTRAN pointers, supported in the SGI FORTRAN-77 release. The FORTRAN pointers associate the address of a variable with the variable name, thereby allowing routines such as malloc( ) to associate space with the variable. In the implementation, the pointers and current sizes are stored in a common block. This method for handling the space allocation is preferred over the standard C$DOACROSS method of declaring each scratch array LOCAL, since it prevents needless reallocation of the space on subsequent C$DOACROSS calls in different scratch memory array contexts or in other words, one integer and one real scratch array can be allocated to fulfill all the scratch space needs. All of the SGI shared memory code is wrapped by the CPP define SGI_MP.

Overall this provides an efficient implementation with fairly minimal code modification and should be readily portable to other shared memory computers. Extensions of this work are underway to cleanup the code, make the shared memory implementation slightly more portable, port if to new shared memory machines, and to add some means to turn off parallelism in the code when it clearly doesn't provide any speed-up.

9.5. Message passing implementation

After careful review of the available message passing implementations of AMBER 4.0, the developers decided to integrate the Message Passing Interface Standard 1.0 (MPI) [124] version developed independently under AMBER 4.0 (minmd), and ported to AMBER 4.1 (sander and gibbs), by Jim Vincent and Ken Merz from the Department of Chemistry at Pennsylvania State University [125]. The primary reason this version was chosen was that the implementation was very elegant, the emerging MPI standard very portable, it was a SPMD (single program multiple data) implementation, and the code modifications required were actually very small. Integration and modification of the original AMBER/MPI work was performed in an active collaboration among Jim Vincent, Thomas Cheatham and David Case [126].

The implementation in Sander and Gibbs provides for parallelization of the nonbond forces and energies, the creation of the residue-based pairlist, and the bond, angles, and dihedrals (force, energy).

The MPI calls are fully compliant with the FORTRAN interface of MPI 1.0. In order to provide portability to systems where MPI currently is not supported yet, the MPI calls were implemented as subroutine wrappers of the MPI calls to PVM version 3.2 and higher [127], TCGMSG, Cray T3D SHMEM, IBM SP2 MPL, and Intel Paragon message passing calls, and in principle can be ported to any message passing library. The AMBER/MPI verison was developed and tested on clusters of workstations running PVM, the Cray T3D at Pittsburgh Supercomputing Center, IBM SP1 and SP2, the Intel Paragon at San Diego Supercomputing Center and an 8-processor Convex Exemplar, and can be easily adapted to run on any message passing computer.

The version is very similar to native AMBER and all the code is CPP wrapped with the CPP define MPI. One node is chosen as the master node and performs the initial startup code and all the I/O. After startup is complete, the master broadcasts (MPI_BCAST) the necessary common blocks, including all the necessary control data and initial coordinates, to each of the nodes. Then, in SANDER, the main dynamics routine (RUNMD) is called by all the parallel nodes. Within AMBER, most of the time consuming routines are driven by loops over residues. Load balancing is accomplished in general by having each node work on a subset of residues. Therefore, each node creates a portion of the total pairlist, calculates the nonbond energies and forces for this pairlist, and does a subset of the bonds, angles, and dihedrals; then the partial forces, energies, and virial are combined on each node (MPI_ALLREDUCE). Each node then performs an update of the coordinates. For routines that are not parallelized yet, each node performs redundant work. The only communications necessary beyond the initial broadcast and combining of the forces, energies, and virial are for dynamic load balancing of the non-bond pairlist and for the routine which implements the fast 3-point water analytical SHAKE. The polarization code is not yet parallelized, nor fully supported in the AMBER/MPI implementation since these routines expect the standard AMBER full pairlist, rather than the distributed pairlist implemented in the AMBER/MPI version. Overall, the implementation seems reasonably scalable and eminently portable.

Currently work is underway to optimize the MPI version for particular architectures and to parallelize SHAKE, polarization, and the Ewald summation code. Both the AMBER/MPI and shared memory code for SGI multiprocessor implementations are generally released and supported with the standard AMBER 4.1 code distribution.

10. Analysis programs in AMBER

There are several analysis programs in AMBER. An overview is presented first, then a more detailed description of one of the newer additions.

10.1 Structure analysis

ANAL

This is the static analysis module of AMBER. Its purpose is to do energetic and structural analyses of static structures. An important function of this program is decomposition of the energy among different groups of atoms in order to find the interaction energies between different parts of the system.

MDANAL

This molecular dynamics analysis program was written in September 1983, based on software provided by W.F. van Gunsteren and updated on Dec 1985 for version 3.0. Revised Nov 1989 for version 3.0 Rev A, and was unchanged for version 4.0. It does correlations and averaging of nucleic acid-type measurements and coordinates as well as RMS fitting and the preparation of data for normal mode movies. The

code was declared impossible to understand and a replacement program has been written, which is described next.

CARNAL

A new program for structure analysis, CARNAL is included in the 4.1 release. It interprets a special-purpose programming language that allows users ot build complex measurements from basic capabilities. It recognizes the various coordinate files (trajectories) can be read simultaneously, and the structures aligned, averaged, and compared. The primitive measurement types include: basic geometry (distances, angles, and torsions involving points and vectors); least-squares coordinate fitting and measurement of root mean square deviation; hydrogen bond analysis; and pucker analysis. Groups of atoms may be specified in a set-theoretic manner, i.e. using union, intersection, and negation over atom and residue names, types and numbers. Distributions of various measurements can be accumulated. Portions of trajectories can be selected to be output in various file formats, and atoms can be screened from the output based on various criteria.

RMS structure fitting is performed using the algorithm of Kabsch [128] torsional averaging is according to the method of [129] and pucker analysis following Cremer and Pople [130] with output for nucleic acids adjusted to the Altona and Sundaralingham [131] convention.

In Fig. 2 an example control file is listed to illustrate the form of the CARNAL command language. The keywords are in block capitals, and section delimiters are placed flush with the margin for clarity.

Fig. 3 lists another example involving RMS fitting and some geometrical measurements. For each coordinate set, the molecule is positioned so that one region fits a reference structure, and then the fit and orientation of other parts of the molecule are measured.

CARNAL is programmed in C. The architecture of the program turns on the interpretation of the carnal analysis language: as the command file is parsed, a linked list of 'action nodes' is created and ordered; then in the execution phase, the list of nodes is executed repeatedly. The nodes at the beginning of the list cause streams to be advanced to the next coordinate set of interest. The next coordinate set in the stream is the default, but sets can also be selected by 'windows': skipping a number of sets, then reading a number, for use with free energy equilibration/data collection protocols. The stream input nodes are followed by measurement nodes, and finally output nodes. Each node after the stream nodes refers in some way to the results of a previous node. For example, a group node might cause the center of mass or center of geometry to be calculated for a group of atoms in a stream's current coordinate set, and then another node might calculate the distance between that point and an atom or another group center (which could be in another stream). A table output node in the last section might write the measurement as one column in a table.

```
# Select some coord sets from mdcrd files and output them in pdb format:
FILES_IN
    PARM p1 ketop;              ! keyword, id, filename
    STREAM s1 kecrd kfcrd;      ! keyword, id, 2 filenames
FILES_OUT
    COORD c1 ke.p PDB;          ! keyword, id, filename, output format
DECLARE
      ! no measurements declared for this simple case
OUTPUT
    COORD c1 s1 SELECT (1 3 5 200);
                                ! keyword, files_out id, files_in id:
                                ! command to select sets 1, 3, 5, 200
END
```

Fig. 2. Example control file to illustrate the form of the CARMAL command language.

```
# RMS fit a group of atoms and compare the orientations of other groups
FILES_IN
    PARM p1 ketop;                ! keyword, id, filename
    STREAM s1 kecrd kfcrd;        ! keyword, id, 2 filenames
    STATIC s2 k.pdb;              ! keyword, id, 1 filename;
                                  ! STATIC structures are pre-loaded
                                  ! and remain in memory for reference
FILES_OUT
    TABLE t1 k.tbl;               ! keyword, id, filename
DECLARE
    GROUP g1 (RES 12-16,26-28);   ! keyword, id, residue list;
                                  ! default coord set: first STREAM, s1
    GROUP g2 (RES 1-9,38-48);     ! keyword, id, residue list
    GROUP g3 (RES 1-9,38-48) s2;  ! keyword, id, residue list, setid
    DISTANCE d1 g1 g2;            ! keyword, id, point1, point2;
                                  ! defaults: group ids are interpreted
                                  ! as geometrical centers
    RMS r1 FIT g1 s1 s2;          ! keyword, id, keyword, groupid,
                                  !              streamid, refcrdid;
                                  ! FIT = do mass-weighted best RMS fit
                                  ! of STREAM to STATIC set using GROUP
    RMS r2 g2 r1;                 ! keyword, id, groupid, previous rmsid:
                                  ! measure fit of group 2 resulting
                                  ! from minimized FIT of group 1
    ANGLE a1 g3%cm g1%cm g2%cm;   ! keyword, id, point1, point2, point3;
                                  ! measure angular deviation of second
                                  ! group from reference structure,
                                  ! using centers of mass
OUTPUT
    TABLE t1 r1 r2 d1 a1;         ! keyword, files_out id, measurement ids
END
```

Fig. 3. Example involving RMS fitting.

The nodes are C structures that have a common header section containing a pointer to the next node, the user's name for the node, and information on the type of node. The remainder of the node is a variable C 'union' structure that is interpreted according to the node type.

Various extensions are contemplated for CARNAL. Some basic functions yet to be added are correlation and energy analysis. Another feature would be selection of coordinate sets from a stream by measurement criteria, allowing commands like 'output all coordinate sets in which the interaction energy between two groups of atoms is greater than 5 kcal/mol and the RMS fit of a third group to the crystal structure is better than that of the previously output set'.

Beyond the addition of measurement capabilities lies the prospect of embedding CARNAL in a graphical environment. This would allow the user to define measurements on a sketchpad by placing boxes representing the major functions and connecting them with lines representing the flow of information. These dependencies are currently handled by the id strings convention seen in the examples, but a graphical representation would provide a more meaningful and less ambiguous interface. Groups of atoms could also be selected graphically from a molecular viewer. The 'specification picture' might then be automatically parsed into the existing command language for execution within a broader context in which the AMBER runs creating the input could also be specified, and the tabular output and coordinate sets would be further visualized and manipulated. AVS, Explorer, Khoros, and Tk/Tcl are all possible vehicles for this extension.

11. The AMBER/INTERFACE front end

A new front end to the SANDER and GIBBS programs has been written. The AMBER/INTERFACE front-end is written in the Interface Programming Script (IPS) language [28] – an interpreted language designed specifically to make the design of flexible and powerful program interfaces straightforward. Both INTERFACE (the IPS language interpretor) and AMBER/INTERFACE were written by David Pearlman.

Features of the AMBER/INTERFACE front-end include:

(1) Simple-to-use commands control all input to MINMD (4.0) SANDER and GIBBS.
(2) The input for the two programs is rationalized in the AMBER/INTERFACE. This means that the input for the two programs is identical for those elements which have the same effect in each program. Commands which are unique to a particular program are read-but-ignored if specified as input to other programs.
(3) The commands specific to the AMBER/INTERFACE front-end are supplemented by various functions of the Interface language. These include control constructs (DO loops, IF ( ) THEN...ELSE, GOTO), general variable assignment, and a large list of intrinsic functions (e.g. +, −, *, /, SIN, COS, EXP, exponentiation, etc.).
(4) The AMBER/INTERFACE front-end script is identical on different computers (with a small number of necessary exceptions, such as the file path specifications within the file).
(5) The command set defined in the AMBER/INTERFACE fron-end can be easily modified or extended by the user.
(6) Commands are (with a few exceptions) order independent.
(7) Comments can be freely interspersed with commands.
(8) It is straightforward to program in the IPS language to create front ends to other programs, or to add additional commands.

The specific commands and feature available in the AMBER/INTERFACE are too numerous to be covered here. A sample input script is given in Fig. 4. Full details are provided in the reference manual [28].

12. LEaP

LEaP [27] is an acronym for Link, Edit and Parm, the three most commonly used AMBER programs that it replaces (it also replaces PREP). LEaP can be used to prepare structure and force field files for SANDER, GIBBS, SPASMS and NMODE. There are two versions of LEaP: terminal (command-line only) and X-windows, which includes a molecular viewer and table editing for force field parameters. In the molecular viewer, one can sketch new residues, load pdb files, bond atoms by distance, add hydrogens automatically, manipulate structures, relax internal coordinates, and call up a table editor to set atom types and charges. General capabilities include geometric measurements, placement of ions by electrostatic potential and solvatation with any molecule or box of molecules. It inaugurates a new file format for internal use, object file format, which is a generalized object-oriented format suitable for a variety of purposes and may be adopted in future by the other AMBER programs. LEaP was written by Christian Schafmeister; since it is a new program with the 4.1 release, the old setup programs are still provided and supported.

```
! Run 30 simulations in vacuo with NMR-derived distance restraints;
! change the random number for each simulation. Each
! simulation is 10000 steps (20ps) of annealed MD. For the first
! 10 simulations, ramp up to a top temperature of 600K before
! cooling back down. For comparison, for the second set of 10 simulations
! ramp up to a top temperature of of 750K before cooling back down,
! and for the third set of 10 simulations, ramp up to a top temperature
! of 900K before cooling.
!
! Author: David A. Pearlman
!
! set up the md run

 do iloop = 1,30
     sander
     md

! assign vr, which is the version of this run. Make it the
! same as iloop. All the files created for this iteration of the
! loop will be md<vr>.*

     assign vr = iloop

! assign the changing (on each loop) value of the random seed

     random_seed = 71200 + iloop

     info/off
     title = "md.sample"
     time_limit = 9999999.0
     read/formatted
     dielectric = 1r
     pairlist / update =  50 / cutoff = 8.0
     report/steps = 1000

! ------------------------------------------------------------------
! Set the parameters specific to MD:

     steps = 10000/runs=1

! Use shake; Use standard temp. coupling with a single scale factor

    shake/toler=0.0005
    temp/target=300./constant=temp/tau_solute=0.04/initial=300.0 \
        /coupling=single/dtemp=10.0

    timestep = 0.0020
    com/remove/steps = 1000

! ------------------------------------------------------------------
! Set the parameters that govern the refinement variables

! ramp the TEMP up from 300 to TOPTP over the first 2000 steps, then
! keep it at TOPTP for steps 2000-> 8,000. Then bring it back down to
! 300 over the final  2000 steps. TOPTP is the maximum temperature,
! which is 600K for the first 10 runs, and 750K for runs 11-20 and
! 900K for runs 21-30.

    if (iloop.le.10) then
       assign toptp = 600.0
    else if (iloop.le.20) then
       assign toptp = 750.0
    else
       assign toptp = 900.0
    end if

    change TEMP0 /from= 300.0 /to= toptp /step1=    1/step2= 2000/every=50
    change TEMP0 /from= toptp /to= toptp /step1= 2001/step2= 8000/every=50
    change TEMP0 /from= toptp /to= 300.0 /step1= 8001/step2=10000/every=50
```

Fig. 4. Sample AMBER INTERFACE input file for runnning restrained MD.

```
! Raise the restraint weights from 0.1 to 36.0 over the first 1000 steps
! and then keep them there.

    change REST  /from=  0.1/to= 36.0  /step1=    1/step2= 1000/every=50/geom
    change REST  /from= 36.0/to= 36.0  /step1= 1001/step2=    0

! The following, if uncommented, would force time-averaged refinement

!   time_average = distance/tau=10.0/power=3/pseudo_force
!   time_average = torsion /tau=10.0/power=-1

! The restraints are in file "min.noe", in the &rst namelist format
! described in the Amber manual.

    nmr/disang  = "min.noe"
       /listout = md<vr>.edv

    generate /output=md<vr>.inp

! Now run the md. Note that any filename that contains a
! forward slash (/) must be enclosed in double quotes.

    run \
        /input    = md<vr>.inp                              \
        /param    = "../parm.prm"                           \
        /coord    = "min1.rst"                              \
        /output   = md<vr>.out                              \
        /restart  = md<vr>.rst                              \

           /inf      = md<vr>.inf

  ! Make sure you issue a clearall statement between iterations of the
  ! loop to clear all of the sander setup options and to reclaim memory.

     clearall

   end do
```

Fig. 4. (Continued)

13. Concluding remarks

We have outlined the development and current status of the AMBER package of programs. Some of the code has reached a high level of development (e.g. GIBBS, SANDER and NMODE), some parts need more 'filling out' (e.g. CARNAL) and the graphical interface (LEaP) still needs to be made more robust and error-free. One of the strengths of AMBER has always been its 'open system' philosophy, which has led to some very exciting developments and extensions of its capabilities, for example the QM/MM coupling [35] and parallel programming paradigm [125] in the Merz group, and the Ewald approach to molecular dynamics in crystals by York *et al.* [132]. Hopefully, AMBER will continue to evolve to allow ever-more-realistic and useful simulation of the structures and energies of complex molecules in various environments.

Acknowledgements

Peter Kollman is pleased to acknowledge research support from the the NIH (GM-29072 and CA-25644) and the NSF (CHE94-17458) Tom Cheatham is a Biotechnology Trainee. David Case acknowledges NIH support (GM-45811).

Appendix. AMBER file specifications

AMBER 4.0 PARAMETER/TOPOLOGY FILE SPECIFICATION

FORMAT (20a4) (ITITL (i), i = 1,20)
ITITL:title

FORMAT (12i6) NATOM, NTYPES, NBONH, MBONA, NTHETH, MTHETA,
NPHIH, MPHIA, NHPARM, NPARM, NNB, NRES,
NBONA, NTHETA, NPHIA, NUMBND, NUMANG, NPTRA,
NATYP, NPHB, IFPERT, NBPER, NGPER, NDPER,
MBPER, MGPER, MDPER, IFBOX, NMXRS, IFCAP
NATOM:total number of atoms
NTYPES:total number of distinct atom types
NBONH:number of bonds containing hydrogen
MBONA:number of bonds not containing hydrogen
NTHETH:number of angles containing hydrogen
MTHETA:number of angles not containing hydrogen
NPHIH:number of dihedrals containing hydrogen
MPHIA:number of dihedrals not containing hydrogen
NHPARM:currently not used
NPARM:currently not used
NEXT:number of excluded atoms
NRES:number of residues
NBONA:MBONA + number of constraint bonds
NTHETA:MTHETA + number of constraint angles
NPHIA: MPHIA + number of constraints dihedrals
NUMBND:number of unique bond types
NUMANG:number of unique angle types
NPTRA:number of unique dihedral types
NATYP:number of atom types in parameter file, see SOLTY below
NPHB:number of distinct 10-12 hydrogen bond pair types
IFPERT:set to 1 if perturbation info is to be read in
NBPER:number of bonds to be perturbed
NGPER:number of angles to be perturbed
NDPER:number of dihedrals to be perturbed
MBPER:number of bonds across boundary to non-perturbed groups
MGPER:number of angles across boundary to
non-perturbed goups
MDPER:number of dihedrals across boundary to non-perturbed groups
IFBOX:set to 1 if standard periodic box information is to be read in
NMXRS:number of atoms in the largest residue
IFCAP:set to 1 if the CAP option from edit was specified

FORMAT(20a4) (IGRAPH(i), i = 1, NATOM)
IGRAPH:the user atoms names

FORMAT(5E16.8) (CHRG(i), i = 1,NATOM)
CHRG:the atom charges

FORMAT(5E16.8) (AMASS(i), i = 1,NATOM)
AMASS:the atom masses

FORMAT(12I6) (IAC(i), i = 1,NATOM)
IAC:index for the atom types involved in Lennard Jones (6–12) interactions. See ICO below.

FORMAT(12I6) (NUMEX(i), i = 1,NATOM)
NUMEX:total number of excluded atoms for atom "i". See NATEX below.

FORMAT(12I6) (NNO(i), i = 1,NTYPES * NTYPES)
ICO:Lennard Jones (6–12) nonbond type index into CN1 and CN2 for atoms of each particular type. For example, idx = ICO(NTYPES * IAC(i) − 1 + IAC(j).See CN1, CN2 below.

FORMAT(20A4) (LABRES(i), i = 1,NRES)
LABRES:the residue labels

FORMAT(12I6) (IPRES(i), i = 1,NRES)
IPRES:atoms inech residue are listed for atom "i" in IPRES(i) to IPRES(i + 1) − 1

FORMAT(5E16.8) (RK(i), i = 1, NUMBND)
RK:force constant for the bonds of each type, kcal/mol

FORMAT(5E16.8) (RK(i), i = 1,NUMBND)
REQ:the equilibrium bond length for the bonds of each type, angstroms

FORMAT(5E16.8) (RK(i), i = 1,NUMANG)
TK:force constant for the angles of each type, kcal/mol A** 2

FORMAT(5E16.8) (RK(i), i = 1,NUMANG)
TEQ:the equilibrium angle for the angles of each type, degrees

FORMAT(5E16.8) (RK(i), i = 1,NPTRA)
PK:force constant for the dihedrals of each type, kcal/mol

FORMAT(5E16.8) (RK(i), i = 1,NPTRA)
PN:periodicity of the dihedral of a given type

FORMAT(5E16.8) (RK(i), i = 1,NPTRA)
PHASE:phase of the dihedral of a given type

FORMAT(5E16.8) (SOLTY(i), i = 1,NATYP)
SOLTY:currently unused (reserved for future use)

FORMAT(5E16.8) (CN1(i), i = 1,NTYPES*(NTYPES + 1)/2)
CN1:Lennard Jones r**12 terms for all possible atom type interactions, indexed by ICO and IAC; for atom i and j where i < j, the index intothis array is as follows: CN1(ICO(NTYPES*IAC(i) – 1 + IAC(j))).

FORMAT(5E16.8) (CN2(i), i = 1,NTYPES*(NTYPES + 1)/2)
CN2:Lennard Jones r**6 terms for all possible atom type interactions. Indexed like CN1 above

FORMAT(12I6) (IBH(i),JBH(i)ICBH(i), i = 1,NBONH)
IBH:atom involved in bond "i", bond contains hydrogen
JBH:atom involved in bond "i", bond contains hydrogen
ICBH:index into parameter arrays RK and REQ

FORMAT(12I6) (IB(i),JB(i),ICB(i), i = 1,NBONA)
IB:atom involved in bond "i",bond does not contain hydrogen
JB:atom involved in bond "i",bond does not contain hydrogen
ICB:index into parameter arrays RK and REQ

FORMAT(12I6) (ITH(i),JTH(i),KTH(i),ICTH(i), i = 1,NTHETH)
ITH:atom involved in angle "i", angle contains hydrogen
JTH:atom involved in angle "i", angle contains hydrogen
KTH:atom involved in angle "i", angle contains hydrogen
ICTH:index into parameter arrays TK and TEQ for angle ITH(i)-JTH(i)-KTH(i)

FORMAT(12I6) (IT(i),JT(I),KT(i),ICT(i), i = 1,NTHETA)
IT:atom involved in angle "i", angle does not contain hydrogen
JT:atom involved in angle "i", angle does not contain hydrogen
KT:atom involved in angle "i", angle does not contain hydrogen
ICT:index into parameter arrays TK and TEQ for angle IT(i)-JT(i)-KT(i)

FORMAT(12I6) (IPH(i)JPH(i),KPH(i),LPH(i),ICPH(i), i = 1,NPHIH)
IPH:atom involved in dihedral "i", dihedral contains hydrogen
JPH:atom involved in dihedral "i", dihedral contains hydrogen
KPH:atom involved in dihedral "i", dihedral contains hydrogen
LPH:atom involved in dihedral "i", dihedral contains hydrogen
ICPH:index into parameter arrays PK, PN, and PHASE for dihedral IPH(i)-JPH(i)-KPH(i)-LPH(i)

FORMAT(12I6) (IP(i),JP(i),KP(i),LP(i),ICP(i), i = 1,NPHIA)
IP:atom involved in dihedral "i", dihedral contains hydrogen
JP:atom involved in dihedral "i", dihedral contains hydrogen

KP:atom involved in dihedral "i", dihedral contains hydrogen
LP:atom involved in dihedral "i", dihedral contains hydrogen
ICP:index into parameter arrays PK,PN, and PHASE for dihedral IPH(i)-JPH(i)-KPH-LPH(i)

FORMAT(12I6) (NATEX(i), i = 1,NEXT)
NATEX:the excluded atom list. To get the excluded list for atom "i" you need to traverse the NUMEX list, adding up all the previous NUMEX values, since NUMEX(i) holds the number of excluded atoms for atom "i", not the index into the NATEX list. Let IEXCL = SUM(NUMEX(j),j = 1,i − 1), then excluded atoms are NATEX (IEXCL) to NATEX(IEXCL + NUMEX(i)).

FORMAT(5E16.8) (ASOL(i), i = 1,NPHB)
ASOL:the value for the r**12 term for hydrogen bonds of all possible types. Index into these arrays is equivalent to the CN1 and CN2 arrays, however the index is negative. For example, for atoms i and j, with i < j, the index is −(NTYPES*IAC(i) − 1 + IAC(j)).

FORMAT(5E16.8) (BSOL(i), i = 1,NPHB)
BSOL:the value for the r**10 term for hydrogen bonds of all possible types. Indexed like ASOL.

FORMAT(5E16.8) (HBCUT(i), i = 1,NPHB)
HBCUT:no longer in use

FORMAT(20A4) (ISYMBL(i), i = 1,NATOM)
ISYMBL:the AMBER atom types for each atom

FORMAT(20A4) (ITREE(i), i = 1,NATOM)
ITREE:the list of tree joining information, classified into five types. M -- main chain, S -- side chain, B -- branch point, 3 -- branch into three chains, E -- end of the chain

FORMAT(12I6) (JOIN(i), i = 1,NATOM)
JOIN:tree joining information, potentially used in ancient analysis programs. Currently unused in sander or gibbs.

FORMAT(12I6) (IROTAT(i), i = 1,NATOM)
IROTAT:apparently the last atom that would move if atom i was rotated, however the meaning has been lost over time. Currently unused in sander or gibbs.

**** The following are only present if IFBOX.g.t. 0 ****

FORMAT(12I6) IPTRES, NSPN, NSPSOL
IPTRES:final residue that is considered part of the solute, reset in sander and gibbs

NSPM:total number of molecules
NSPSOL:the first solvent 'molecule'

FORMAT(12I6) (NSP(i), i = 1,NSPM)
NSP:the total number of atoms in each molecule,
necessary to correly determine the pressure scaling

FORMAT(5E16.8) BETA,BOX(1),BOX(2),BOX(3)
BETA:periodic box, angle between the XY and YZ planes in degrees.
BOX:the periodic box lengths in the X, Y, and Z directions

**** The following are only present if IFCAP.gt.0 ****

FORMAT(12I6) NATCAP
NATCAP:number of waters placed in the cap by edit

FORMAT(5E16.8) CUTCAP,XCAP,YCAP,ZCAP
CUTCAP:the distance from the center of the cap to the outside
XCAP:X coordinate for the center of the cap
YCAP:Y coordinate for the center of the cap
ZCAP:Z coordinate for the center of the cap

**** The following is only present if IFPERT.gt.0 ****

FORMAT(12I6) (IBPER(i),JBPER(i), i = 1,NBPER)
IBPER:atoms involved in perturbed bonds
JBPER:atoms involved in perturbed bonds

FORMAT(12I6) (ICBPER(i), i = 1,2 * NBPER)
ICBPER:pointer into the bond parameter arrays RK and REO for the perturbed bonds. ICBPER(i) represents lambda = 0 and ICBPER(i + NBPER) represents lambda = 1.

FORMAT(12I6) (ITPER(i),JTPER(i), KTPER(i), i = 1,NGPER)
IPTER:atoms involved in perturbed angles
JTPER:atoms involved in perturbed angles
KTPER:atoms involved in perturbed angles

FORMAT(12I6) (ICTPER(i), i = 1,2 * NGPER)
ICTPER:pointer into the angle parameter arrays TK and TEQ for the perturbed angles. ICTPER(i) represents lambda = 0 and ICTPEr(i + NGPER) represents lambda = 1.

FORMAT(12I6) (IPPER(i),JPPER(i),KPPER(i),LPPER(i), i = 1,NDPER)
IPTER:atoms involved in perturbed dihedrals
JPPER:atoms involved in perturbed dihedrals

KPPER:atoms involved in perturbed dihedrals
LPPER:atoms involved in perturbed dihedrals

FORMAT(12I6) (ICPPER(i), i = 1,2 * NDPER)
ICPPER:pointer into the dihedral parameter arrays PK,PN and PHASE for the perturbed dihedrals. ICPPER(i) represents lambda = 0 and ICCPER(i + NGPER) represents lambda = 1.

FORMAT(20A4) (LABRES(i), i = 1,NRES)
LABRES:residue names at lambda = 1

FORMAT(20A4) (IGRPER(i), i = 1,NATOM)
IGRPER:atomic names at lambda = 1

FORMAT(20A4) (ISMPER(i), i = 1,NATOM)
ISMPER:atomic symbols at lambda = 1

FORMAT(5E16.8) (ALMPER(i), i = 1,NATOM)
ALMPER:unused currently in gibbs

FORMAT(12I6) (IAPER(i), i = 1,NATOM)
LAPER:IAPER(i) = 1 if the atom is being perturbed

FORMAT(12I6) (IAPER(i), i = 1,NATOM)
IACPER:index for the atom types involved in Lennard Jones interaction at lambda = 1. Similar to IAC above. See ICO above.

FORMAT(5E16.8) (CGPER(i), i = 1,NATOM)
CGPER:atomic charges at lambda = 1

* * * * The following is only present if IPOL.eq. 1* * * *

FORMAT(5E18.8) (ATPOL(i), i = 1,NATOM)
ATPOL:atomic polarizabilities

* * * * The following is only present if IPOL.eq. 1. and. IFPERT. eq. 1* * * *

FORMAT(5E18.8) (APTOL1(i), i = 1,NATOM)
ATPOL1:atomic polarizabilities at lambda = 1 (above is at lambda = 0)

AMBER 4.0 RESTART FILE SPECIFICATION

FORMAT(20A4) ITITL

ITITL:the titel of the current run, from the AMBER
parameter/topology file

FORMAT(15,5E15.7) NATOM,TIME
NATOM:total number of atoms in coordinate file
TIME:option, current time in the simulation (picoseconds)

FORMAT(6F12.7) (X*i*), Y(i), Z(i), i = 1, NATOM)
X,Y,Z,:coordinates

IF dynamics:
FORMAT(6F12.7) (VX(i), VY(i), VZ(i), i = 1, NATOM)
VX,VY,VZ:velocities

IF constant pressure:FORMAT(6F12.7)BOX(1),BOX(2),BOX(3)
BOX:size of the periodic box

GIBBS will print extra information.

AMBER TRAJECTORY (COORDINATES OR VELOCITY) FILE SPECIFICATION

FORMAT(20A4)ITITL
ITITL:the title of the current run, from the AMBER
parameter/topology file

The followingis sequentially dropped for each snapshot of the
trajectory:

FORMAT(10F8.3) (X(i), Y(i), Z(i), i = 1,NATOM)
X,Y,Z:coordinates or velocities

IF constant pressure
FORMAT(10F8,3) BOX(1), BOX(2), BOX(3)
BOX:size of periodic box.

References

[1] S. Lifson and A. Warshel, J. Chem. Phys. 49 (1968) 5116–5124.
A. Warshel and S. Lifson, J. Chem. Phys. 53 (1968) 582–594.
[2] B. Gelin, Ph.D. thesis, Harvard University (1976)
[3] P.K. Weiner and P.A. Kollman, J. Comput. Chem 2 (1981) 287–303.
[4] R. Langridge, T.E. Ferrin, I.D. Kuntz and M.L. Connolly, Science 211 (1981) 661–666.
[5] A. Dearing, CHEM, developed at University of California, San Francisco, 1980–1981.
[6] T.E., Ferrin, C.C. Huang, L.E. Jarvis and R. Langridge, MIDAS: Molecular Interactive Display And Simulation, University of California, San Francisco (1985).

[7] S.J. Weiner, P.A. Kollman, D.A. Case, U.C. Singh, C. Ghio, G. Alagona, S. Profeta Jr., and P. Weiner, J. Amer. Chem. Soc. 106 (1984) 765–784.
[8] S.J. Weiner, P.A. Kollman, D.T. Nguyen and D.A. Case, J. Comput. Chem. 7 (1986) 230–252.
[9] U.C. Singh and P.A. Kollman, J. Comput. Chem. 5 (1984) 129–145.
[10] P. Weiner, D.A. Case and P.A. Kollman, AMBER 1.1, University of California, San Francisco (1981).
[11] H.J.C. Berendsen, J.P.M. Postma, W.F. van Gunsteren, A. DiNola and J.R. Haak, J. Chem. Phys. 81 (1984) 3684–3690.
[12] U.C. Singh, S.J. Weiner and P.A. Kollman, Proc. Nat. Acad. Sci. USA 82 (1985) 755–759.
[13] G.L. Seibel, U.C. Singh, and P.A. Kollman, Proc. Natl. Acad. Sci. USA 82 (1985) 6537–6540.
[14] U.C. Singh, P.K. Weiner, S.J. Weiner and P.A. Kollman, AMBER 2.0, University of California, San Francisco (1984).
[15] W. Jorgensen and C. Ravimohan J. Chem. Phys. 83 (1985) 3050–3054.
[16] P.A. Bash, U.C. Singh, F. Brown, R. Langridge and P.A. Kollman, Science 235 (1987) 574–576.
[17] P.A. Bash, U.C. Singh, R. Langridge and P.A. Kollman, Science 236 (1987) 564–568.
[18] U.C. Singh, F.K. Brown, P.A. Bash and P.A. Kollman, J. Amer. Chem. Soc. 109 (1987) 1607–1614.
[19] A.E. Howard, U.C. Singh, M. Billeter and P.A. Kollman, J. Amer. Chem Soc. 110 (1988) 6984–6991.
[20] U.C. Singh and P.A. Kollman, J. Comput. Chem. 7 (1986) 718–730.
[21] U.C. Singh, P.K. Weiner, J.W. Caldwell and P.A. Kollman, AMBER 3.0, University of California, San Francisco (1986).
[22] G.L. Seibel, U.C. Singh, P.K. Weiner, J.A. Caldwell and P.A. Kollman AMBER 3A, University of California, San Francisco (1989).
[23] G.L. Seibel, Ph.D. thesis, University of California, San Francisco (1991).
[24] D.C. Spellmeyer, W.C. Swope, E.-R. Evensen and D.M. Ferguson, SPASMS, University of California, San Francisco (1990—1994).
[25] D.A. Pearlman, D.A. Case, J.W. Caldwell, G.L. Seibel, U.C. Singh, P.K. Weiner and P.A. Kollman, AMBER 4.0, University of California, San Francisco (1991).
[26] D.A. Pearlman, D.A. Case, J.W. Caldwell, W.S. Ross, T.C. Cheatham, D.M. Ferguson, G.L. Seibel, U.C. Singh, P.K. Weiner and P.A. Kollman, AMBER 4.1, University of California, San Francisco, (1994).
[27] C. Schafmeister, LEaP, University of California, San Francisco, (1992–1994).
[28] D.A. Pearlman, Interface: An interface design system (1994); The AMBER/Interface Front End.
[29] N. Pavitt and D. Hall J. Comput. Chem 5 (1984) 441.
[30] N.L. Allinger, Y.H. Yuh and J.H. Lii, J. Amer. Chem Soc. 111 (1989) 8551–8566.
[31] M.J. Hwang, T.P. StockFish and A.T. Hagler, J. Amer. Chem. Soc. 116 (1994) 2525–2535.
[32] W.N. Cornell et al., J. Amer. Chem. Soc., 117 (1995) 5199–5197.
[33] Y.X. Sun and P.A. Kollman J. Comput. Chem. 13 (1992) 33–40.
[34] Y.X. Sun and P.A. Kollman J. Chem. Phys. 97 (1992) 5108–5112.
[35] R.V. Stanton, D.S. Hartsough and K.M. Merz, J. Phys. Chem. 97 (1993) 11868–11870.
[36] J. Applequist, J.R. Carl and K.-K. Fung, J. Amer, Chem. Soc. 94 (1971) 2952–2960.
[37] J.B. Applequist, J. Chem. Phys. 58 (1973) 4251–4259; Accts, Chem. Res. 10 (1977) 79–85.
[38] J.B. Applequist, K.R. Sundberg, M.L. Olson and L.C. Weiss, J. Chem. Phys. 70 (1979) 1240–1246;
J. Applequist, J. Chem. Phys. 71 (1979) 1983–1984; 4324–4331; 4332–4338.
Biopolymers 20 (1981) 387–397; 2311–2322; 21 (1982) 779–795;
J. W. Caldwell and J.B. Applequist, Biopolymers 23 (1984) 1891–1904.
[39] T.P. Lybrand and P.A. Kollman J. Chem. Phys. 83 (1985) 2923;
P. Cieplak, T.P. Lybrand and P.A. Kollman, J. Chem. Phys 87 (1987) 6393.
[40] P. Ahlström Theoretical Studies of Protein Solutions, Ph. D. Thesis, University of Lund, 1988 (published as : P. Ahlström, A. Wallquist, S. Engström, B. Jonsson, Mol. Phys. 68 (1989) 459).
[41] M. Sprik and M.L. Klein, J. Chem. Phys. 89 (1988) 7556–7560.
[42] T.P. Straatsma, J.A. McCammon Mol. Simul. 5 (1990) 181; Chem. Phys. Lett. 167 (1990) 252.
[43] J.B. Applequist, private communication.
[44] J.W. Caldwell, L.X. Dang and P.A. Kollman, J. Amer. Chem. Soc. 112 (1990) 9144–9147.
L.X. Dang, J.E. Rice, J.W. Caldwell and P.A. Kollman, J. Amer Chem. Soc. 113 (1991) 2482–2486.
[45] W.L. Jorgensen and J. Tirado-Rives, J. Amer. Chem. Soc. 110 (1988) 1657.
[46] W.L. Jorgensen J. Chandrasekhar, J.D. Madura, R.W. Impey, M.L. Klein, J. Chem. Phys. 79 (1983) 926–935.
[47] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan and M. Karplus, J. Comput. Chem. 4 (1983) 187–217.
[48] R. Fletcher and C.M. Reeves, Computer J. 7 (1964) 149–153.
[49] T. Schlick and M. Overton J. Comput. Chem. 8 (1987) 1025–1039.
[50] W.F. van Gunsteren, in: Computer simulation of biomolecular systems: Theoretical and experimental applications, Vol. 2. W.F. van Gunsteren, P.K. Weiner and A.J. Wilkinson, eds. (ESCOM, Leiden, 1993) pp. 3–36.
[51] J.-P. Ryckaert, G. Ciccotti and H.J.C. Berendsen J. Comput. Phys. 23 (1977) 327–341.

[52] S. Miyamoto and P.A. Kollman, J. Comput. Chem. 13 (1992) 952–962.
[53] R.C. van Schaik, W.F. van Gunstern and H.J.C. Berendsen, J. Comp.-Aided Mol. Design 6 (1992) 97–112.
[54] G.M. Clore and A. Gronenborn, eds., NMR in Proteins (MacMillan, New York, 1993).
[55] K. Wüthrich, Science 243 (1989) 45–50.
[56] A.M.J.J. Bonvin, R. Boelens and R. Kaptein, in: Computer simulation of biomolecular systems: Theoretical and experimenal applications, Vol. 2, W.F. van Gunsteren, P.K. Weiner and A.J. Wilkinson, eds. (ESCOM, Leiden, 1993) pp. 407–440.
[57] G.P. Gippert, P.F. Yip, P.E. Wright and D.A. Case, Biochem. Pharm. 40 (1990) 15–22.
[58] B.A. Borgias M. Gochin, D.J. Kerwood and T.L. James, Prog. NMR Spect. 22 (1990) 83–100.
[59] D.A. Case and P.E. Wright, in: NMR in Proteins, G.M. Clore and A. Gronenborn, eds. (MacMillan, New York, 1993) pp. 53–91.
[60] R. Brüschweiler and D.A. Case, J Prog. NMR Spectr. 26 (1994) 27–58.
[61] D.A. Case, H.J. Dyson. and P.E. Wright, Meth. Enzymol. 239 (1994) 392–416.
[62] D.A. Pearlman, J. Biomol. NMR 4, (1994) 279–299.
[63] W. Braun and N. Go, J. Mol. Biol 186 (1985) 611–626.
[64] A.E. Torda, R.M. Scheek and W.F. Van Gunsteren, Chem. Phys. Lett. 157 (1989) 289–294.
[65] A.E. Torda, R.M. Scheek and W.F. Van Gunsteren, J. Mol. Biol. 214 (1990) 223–235.
[66] A.E. Torda, R.M. Brunne, T. Huber, H. Kessler and W.E. van Gunsteren, J. Biomol. NMR 3 (1993) 55–66.
[67] D.A. Pearlman and P.A. Kollman, J. Mol. Biol. 20 (1991) 457–479.
[68] D.A. Pearlman, J. Biomol. NMR 4 (1994) 1–16.
[69] A.G. Palmer and D.A. Case, J. Amer. Chem. Soc. 114 (1992) 9050–9067.
[70] R. Brüschweiler and D.A. Case, Phys. Rev. Lett. 72 (1994) 940–943.
[71] P. Yip and D.A. Case, in: Computational Aspects of the Study of Biological Macromolecules by NMR Spectroscopy, J. Hoch, F.M. Poulsen and C. Redfield, eds. (Plenum, New York, 1991) pp. 317–330.
[72] K. Ösapay and D.A. Case, J. Amer. Chem. Soc. 113 (1991) 9436–9444.
[73] K. Ösapay and D.A. Case, J. Biomol. NMR 4 (1994) 215–230.
[74] K. Ösapay, Y. Theriault, P.E. Wright and D.A. Case, J. Mol. Biol. 244 (1994) 183–197.
[75] D.L. Beveridge and F.M. Di Capua, Annu. Rev. Biophys. Biophys. 18 (1989) 431–492.
[76] D.A. Pearlman and P.A. Kollman, J. Chem. Phys. 90 (1989) 2460–2470.
[77] T.P. Straatsma and J.A. McCammon, J. Chem. Phys 95 (1991) 1175–1188.
[78] D.A. Pearlman, J. Phys. Chem. 98 (1994) 1487–1493.
[79] T.P. Straatsma, H.J.C. Berendsen and J.P. M. Postma, J. Chem. Phys. 85 (1986) 6720–6727.
[80] D.A. Pearlman and P.A. Kollman, J. Chem. Phys. 91 (1989) 7831–7839.
[81] T.P. Straatsma, T.J.C. Berendsen and A.J. Stam, Mol. Phys. 57 (1986) 89–95.
[82] D.A. Pearlman, J. Comput. Chem. 15 (1994) 105–123.
[83] D.A. Pearlman and P.A. Kollman, J. Chem. Phys. 94 (1991) 4532–4545.
[84] T.P. Straatsma, M. Zacharias and J.A. McCammon, Chem. Phys. Lett. 196 (1992) 297–302.
[85] D.A. Pearlman, J. Chem. Phys. 98 (1993) 8946–8957.
[86] S. Yun-yu, A.E. Mark, W. Cun-xin, H. Fuhua, H.J.C. Berendsen and W.F. van Gunsteren, Protein Engineering 6 (1993) 289–295.
[87] J. Gao, K. Kuczera, B. Tidor and M. Karplus, Science 244 (1989) 1069–1072.
[88] S. Miyamoto and P.A. Kollman, Proteins: Struct. Funct. Genet. 16 (1993) 226–245.
[89] A.E. Mark and W.F. van Gunsteren, J. Mol. Biol. 240 (1994) 167–176.
[90] D.A. Pearlman and P.R. Connelly, J. Mol. Biol. 248 (1995) 696–717.
[91] P. Cieplak, D.A. Pearlman and P.A. Kollman, J. Chem. Phys. 101 (1994) 627–633.
[92] P.R. Gerber, A.E. Mark and W.F. van Gunsteren, J. Comp. Ass. Mol. Des., 7 (1993) 305–323.
[93] S.H. Fleischman and C.L. Brooks, J. Chem. Phys. 87 (1987) 3029–3037.
[94] H.-A. Yu and M. Karplus, J. Chem. Phys. 89 (1988) 5115–7177.
[95] D.J. Tobias and C.L. Brooks, J. Chem. Phys. 89 (1988) 5115–5127.
[96] D.A. Pearlman and P.A. Kollman, J. Am. Chem. Soc. 113 (1991) 7167–7177.
[97] M.P. Allen and D.J. Tildesley, Computer Simulation of Liquids (Clarendon Press, Oxford, 1987).
[98] W.C. Swope, H.C. Andersen, P.H. Berens and K.A. Wilson, J. Chem. Phys. 76 (1982) 637.
[99] H.J.C. Berendsen, J.P.M. Postma, W.F. van Gunsteren, A. DiNola and J.R. Haak, J. Chem. Phys. 81 (1984) 3684–3690.
[100] J. Guenot and P.A. Kollman, J. Comput. Chem. 14 (1993) 295-311.
[101] D.M. Ferguson, J. Comput. Chem., in press (1995).
[102] H. Andersen, J. Chem. Phys. 72 (1980) 2384–2393.
[103] D.M. Ferguson, Isothermal-Isobaric Molecular Dynamics Simulations with Monte Carlo Volume Sampling, Comp. Phys. Lett, in press.

[104] C.H. Bennett, J. Comput. Phys. 22 (1976) 245.
[105] D.M. Ferguson, J. Chem. Phys. 99 (1993) 10086-10087.
[106] U. Burkert and N.L. Alligner, Molecular Mechanics (Am. Chem. Soc., Washington, D.C., 1982).
[107] D.M. Ferguson, A. Marsh, T. Metzger, D.G. Garrett and K. Kastella, J. Glob. Opt. 4 (1994) 209-227.
[108] S.R. Niketic and K. Rasmussen, The Consistent Force Field: A Documentation (Springer, New York, 1977).
[109] D.A.Case, J. Curr. Opin. Struct. Biol. 4 (1994) 285-290.
[110] R. Brüschweiler and D.A. Case, J Phys. Rev. Lett. 72 (1994) 940-943.
[111] C. Cerjan and W.H. Miller, J. Chem. Phys. 75 (1981) 2800.
[112] J. Simons, P. Jorgenson, H. Taylor and J. Ozment, J. Phys. Chem. 87 (1983) 2745.
[113] D.T. Nguyen and D.A. Case, J. Phys. Chem. 89 (1985) 4020-4026.
[114] G. Lamm and A. Szabo, J. Chem. Phys 85 (1986) 7335-7348.
[115] J. Kottalam and D.A. Case, Biopolymers 29 (1990) 1409-1421.
[116] M. Karplus and J. N. Kushick, Macromolecules 14 (1981) 325-332.
[117] R.M. Levy, M. Karplus, J. Kushick and D. Perahia, Macromolecules 17 (1984) 1370-1374.
[118] A. Kitao, F. Hirata and N. G'obar, J. Chem. Phys. 158 (1991) 447-472.
[119] M.M. Teeter and D.A. Case, J. Phys. Chem. 94 (1990) 8091-8097.
[120] S.E. DeBolt and P.A. Kollman, J. Comput. Chem. 14 (1993) 312.
[121] S.E. DeBolt, D.A. Pearlman and P.A. Kollman, J. Comput. Chem. 15 (1994) 351-373.
[122] We would like to acknowledge SGI for their support and help, and in particular Roberto Gomperts (Boston) for the initial implementation and for working closely with Thomas Cheatham (UCSF) to fit the code into the distribution, Dr. Michael Sclenkrich (Switzerland) for providing better scratch memory handling, parallelizing resnba and particularly for parallelizing SHAKE, and Marc Berger (the Chemistry Applications Specialist at SGI) for coordinating and supporting the active collaboration.
[123] W.F. van Gunsteren and H.J.C. Berendsen, Mol. Phys. 34 (1977) 1311.
J.-P. Ryckaert, G., Ciccotti, and H.J.C. Berendsen, J. Comput. Phys. 34 (1977) 327-341.
G. Ciccotti, M. Ferrario and J.P. Ryckaert, Mol. Phys. 47 (1982) 1253.
[124] Message Passing Interface Forum MPI: A Message Passing Interface Standard, University of Tennessee (5 May 1994). (Anonymous FTP from netlib, mail netlib@ornl.gov with message body send index from mpi).
[125] J. Vincent and K. Merz., MPI implementation of parallel AMBER, Pennsylvania State University (1994).
[126] In addition to those primarily responsible for the AMBER/MPI distribution, we would also like to acknowledge the following individuals for help:
(1)Michael Crowley/David Deerfield at Pittsburgh Supercomputing Center (PSC) for fixing a bug in the portable namelist code for the T3D and T3D resources and for continuing a collaboration to help port Tom Darden's EWALD code to the T3D.
(2) Jeyapandian Kottalam (Cray) and Asiri Nanayakkara for work at fixing the Cray T3D getarg() calls and evaluation of the T3D implementation.
(3) Thomas Huber (Ludwig Maximilian Universitaet, Muenchen) for evaluating the MPI version and writing a TCGMSG wrapper for it.
(4) Steve Chin (IBM) for development of an IBM SP2 version similar in spirit to the current MPI version and SP1/2 adaptation of the latter.
(5) Wilson Ross (UCSF) for discussions of the various parallel versions and help in coordinating the MPI collaboration.
[127] A.L. Beguelin, J.J. Dongarra, G.A. Geist, W.C. Jiang, R.J. Manchek, B.K. Moore and V.S. Sunderam, PVM Version 3.3: Parallel Virtual Machine System University of Tennessee, Knoxville TN; Oak Ridge National Laboratory, Oak Ridge TN; and Emory University, Altanta, Georgia.
[128] Kabsch, Acta Cryst. A 32 (1976) 922-923, A 34 (1976) 827-828.
[129] E. Batschelet, Circular Statistics in Biology, (Academic Press, Inc, New York, NY, 1981).
[130] D. Cremer and J.A. Pople, J. Amer. Chem. Soc. 96 (1975) 1354-1358.
[131] C. Altona and M. Sundaralingham, J. Amer. Chem. Soc. 94 (1972) 8205-8212.
[132] D.M. York, A. Wlodawer, L.G. Pedersen and T.A. Darden, Proc. Nat. Acad. Sci. 91 (1994) 8715-8718.