AMBER

Version 4.1

Amber 4.1 is a collaborative effort of the research groups of Peter Kollman (UCSF) and David Case (Scripps). The authors of Amber 4 are:

David A. Pearlman (UCSF; currently Vertex Pharmaceuticals)

David A. Case (The Scripps Research Institute)

James W. Caldwell (UCSF)

Wilson S. Ross (UCSF)

Thomas E. Cheatham, III (UCSF)

David M. Ferguson (University of Minnesota)

George L. Seibel (for contributions to Amber version 3A while at UCSF)

U. Chandra Singh (for contributions to Amber versions 2 and 3 while at UCSF)

Paul Weiner (for contributions to Amber version 1 while at UCSF)

and

Peter A. Kollman (UCSF)

Acknowledgements

We acknowledge the generous cooperation of Wilfred van Gunsteren, whose molecular dynamics code was used as the basis of the md modules in version 2.0. We are also pleased to acknowledge Rad Olson and Bill Swope at IBM Almaden Center, whose contributions were instrumental in developing the better vector optimized non-bonded routines first released in version 3, revision A. Research support from DARPA, the NIH, and the NSF for Peter Kollman is gratefully acknowledged, as is support from the NIH for David Case. Use of the facilities of the UCSF Computer Graphics Laboratory (Thomas Ferrin, PI) is appreciated. We thank Nelson H.F. Beebe of the University of Utah for permission to include his ''portable namelist'' code. Wendy Cornell contributed a discussion of charge derivation to the manual and added demos and documentation for the RESP program. We also thank Allison Howard and Valerie Daggett for various helpful discussions and suggestions. Many people helped add features to various codes in 4.1; these contributions are described in the documentation for the individual programs.

Recommended Citation:

When citing Amber Version 4.1 in the literature, the following citation should always be used:

> David A. Pearlman, David A. Case, James W. Caldwell, Wilson S. Ross, Thomas E. Cheatham III, David M. Ferguson, George L. Seibel, U. Chandra Singh, Paul K. Weiner and Peter A. Kollman (1995), AMBER 4.1, University of California, San Francisco.

In addition, you may wish to also cite the author(s) of the individual energy program used, if significant features new to version 4.1 are used from that module. See the Release Notes section for new features; see documentation for the individual modules for authors.

Table of Contents

Foreword 4
Introduction 6
Installation 11
Database 16
Tutorial 20

Main AMBER programs:

resp 41
prep 47
link 56
edit 65
parm 82
sander 95
gibbs 155
anal 212
mdanal 223
carnal 233
nmode 253
nmanal 258
lmanal 261

Utility programs:

nucgen 263
ambpdb 267
protonate 268
gwh and pol_h 270
mdovrly 272
mdextract 274
mdcorrp2 275
intense 276
spectrum 278
rdis 279
curvop 280
curvemax 280

Appendices:

Namelist format 281
Group input 283
Parameter development 288
Charge fitting philosophy 297
File Formats 303
Release Notes 312

Foreword

Amber is the collective name for a suite of programs that allow users to carry out molecular dynamics simulations, particularly on biomolecules. None of the individual programs carries this name, but the various parts work reasonably well together, and provide a powerful framework for many common calculations. The term *amber* is also sometimes used to refer to the empirical force field that is implemented here. It should be recognized however, that the code and force field are separate: several other computer packages have implemented the *amber* force field, and other force fields can be implemented with the *amber* programs. Further, the force field is in the public domain, whereas the codes are distributed under a license agreement.

Amber 4.1 (1995) represents a significant change from the most recent previous version, *4.0*, which was released in 1991. A detailed discussion of the changes from earlier versions is contained in the release notes. Briefly, the major differences include:

(1) a completely updated force field for proteins and nucleic acids;

(2) faster algorithms for simulations with water;

(3) parallelized dynamics codes;

(4) new algorithms and procedures for free energy simulations, including support for force fields with polarization terms;

(5) Ewald sum periodicity in the dynamics program Sander;

(6) new graphical and text-based tools for building molecules and preparing input to the dynamics programs;

(7) faster and more powerful tools for NMR spectral simulations;

(8) a new program for fitting point electrostatic charges from quantum data;

(9) a new dynamics and free energy program.

In spite of the changes, this is still recognizably Amber: it still has *prep, link, edit* and *parm* modules, as described in the original paper, [1] and both the input files and much of the code will be familiar to those who have used earlier versions.

What to read next.

If you are installing this package or want to redimension the code, see *installation* section of this manual. If you are familiar with a previous release of this software, you should read the release notes, which describe recent changes. New users should begin with the *introduction* section, and will also find the *examples* section useful. The directories under amber41/demo contain a number of systems that may serve as examples. You should familiarize yourself with the files in the database directory, amber41/dat, by looking over the *database* section of the manual. Although Amber may appear dauntingly complex at first, it has become easier to use over the past few years, and overall is reasonably

[1] P. Weiner and P.A. Kollman, *J. Comput. Chem.* **1981,** *2*, 287.

straightforward once you understand the basic architecture and option choices. Hundreds of people have learned to use Amber; don't be easily discouraged.

An Introduction to Amber 4

Understanding where to begin in Amber is primarily a problem of managing the flow of information in this package. You first need to understand what information is needed by the energy programs (*gibbs, sander, spasms and nmode*). You need to know where it comes from, and how it gets into the form that the energy programs need. This section is meant to orient the new user, and is not a substitute for the individual program documentation.

Information all the energy programs need:

* Cartesian coordinates for each atom in the system.

* "Topology": connectivity, atom names, atom types, residue names, and charges.

* Force field: Parameters for all of the bonds, angles, dihedrals, and atom types in the system.

* Commands: The user specifies the procedural options and state parameters desired.

This information is provided to the energy programs in three files: One contains the coordinates; the second contains the topology and parameters, and called the "topology file"; the command or "input" file is the third file. Additional files may needed for special options specified in the command file. Files are specified in several ways, such as by command line arguments, by unit assignment in the operating system, or by naming them in the input file.

Where is the information found?

Cartesian coordinates usually come from Xray crystallography, NMR spectroscopy, or model-building. They should be in Brookhaven Protein Databank (PDB) format. Somewhat arbitrary cartesian coordinates for some or even all of the atoms can also be obtained from the database described below. Although "raw" Amber has some rudimentary model-building features, it is not terribly useful in this regard, and most people will probably want to use some interactive program to build initial coordinates if they are not otherwise available. The program *LEaP*, distributed for the first time in this release, provides a platform for carrying out many of these modeling tasks, but users may wish to consider other programs as well.

Topology comes from the database: The database is found in the *amber41/dat* directory. It is called *db94.dat*. It contains topology for the standard amino acids as well as N and C-terminal charged amino acids, DNA, and RNA. The database also contains internal coordinates for these monomer units, but coordinate information is usually obtained from PDB files. Topology information for other molecules (not found in the standard database) is kept in user-generated "residue files". These are constructed in the same manner as the database using the program *prep*.

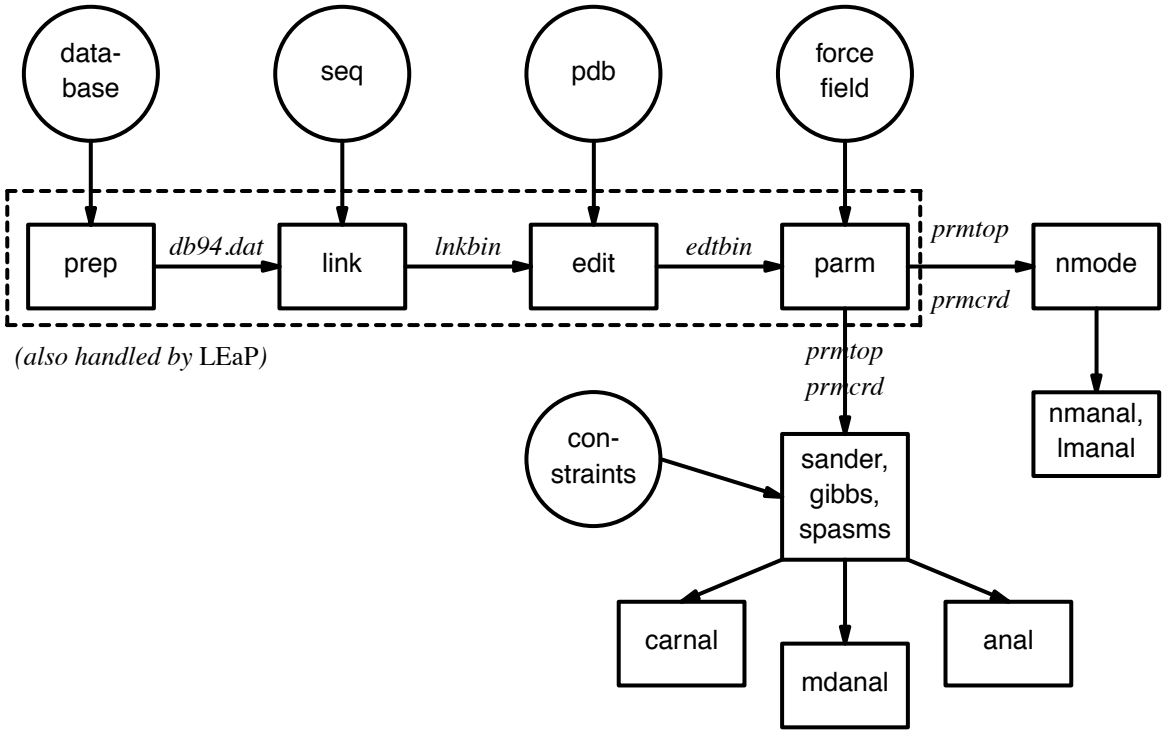
The basic *force field* parameters are also found in the *amber41/dat* directory; the *database* section of the manual contains some detailed descriptions of various force field options. The force field file is read as input by the PARM program. This file may be used "as is" for proteins and nucleic

acids, or users may prepare their own files that contain modifications to the "standard" force fields. Note: energies in *amber* are always in kcal/mol, distances in Ångstroms, and angles in degrees. In the code, calculations are often carried out in different units, but conversion of these units is always performed on input and output.

How do you convert this information to the form needed by the energy programs?

Structures consisting entirely of "standard" residues (proteins, nucleic acids, water, etc.): In order to generate the topology and coordinate files needed by the energy programs, you must run three programs in the sequence *link, edit, parm*. After this, energy or analysis programs may be run, using the *prmtop* file (an output of the *parm* program) plus a coordinate file (which comes either from the *parm* program or from previous runs of the energy programs.)

Structures containing non-standard residues: In addition to the above, the PREP program must be run to generate a database for the non-standard residues.



Basic information flow in AMBER

PREPARATORY PROGRAMS

PREP

creates or adds to a residue database from the appropriate topology/parameter information. Required for residues not already defined in the standard AMBER database. (As supplied, the standard AMBER database contains definitions for the 20 standard amino acids, nucleic acids with the five standard bases, and a few other units). An alternative form of output is the "residue file" which contains the same information as would be found in a database entry, and is simply an alternative way of providing this information to LINK.

LINK

deals only with topology. You tell LINK the residue sequence of your molecule (even if there is only one residue). LINK will extract the topology information for each residue from the standard AMBER database or, optionally, from the residue database files created with PREP. The topology for each residue will be linked together to form the topology for the system. This is written to a binary file (default name = *lnkbin*) that is read by EDIT.

EDIT

deals mainly with coordinates. One of the primary purposes of EDIT is to read PDB coordinates and apply them to the system generated by LINK. Coordinates for atoms that are missing from the PDB file (usually hydrogens) will in most cases be generated automatically by EDIT, using the stored internal coordinates in the link binary file link.bin. Although the usual case is to read PDB coordinates, it is not essential. Edit can act as a simple filter, converting the internal coordinates in the link binary file to cartesian coordinates. EDIT is also a model-building program. It can be used to solvate a molecule in water, to add counterions to nucleic acid systems, or to alter coordinates in specific ways. EDIT creates a binary file (default name = *edtbin*) containing both topology and cartesian coordinates. *edtbin* is read by PARM.

PARM

will determine which bonds, angles, dihedrals, and atom types exist in the system, and extract the appropriate parameters for them from the force field file. PARM then writes the final coordinate and topology files needed by ANAL, MIN, MD, and NMODE. Parm can also be used to add simple non-varying internal coordinate restraints to the system, and to create a two-state topology file for use in GIBBS free energy perturbation calculations. PARM outputs two files which are used subsequently: The topology file (default name = *prmtop*); and the coordinate file (default name = *prmcrd*).

After successfully running PARM, you are ready to begin energy calculations using ANAL, GIBBS, SANDER, or NMODE. Coordinate files that are output from any of these programs may be read by any other.

PROTONATE

This program will add hydrogens in appropriate locations to peptides and proteins that lack them. It can also check the suitability of protons that are already present, and convert from one naming system to another (*e.g.* from IUPAC-IUB recommendations to Brookhaven format.)

ENERGY PROGRAMS

SANDER

is the basic energy minimizer and molecular dynamics program. This program relaxes the

structure by iteratively moving the atoms down the energy gradient until a sufficiently low average gradient is obtained. Structures should usually be minimized before molecular dynamics simulation. The molecular dynamics portion generates configurations of the system by integrating Newtonian equations of motion. MD will sample more configurational space than MIN, and will allow the structure to cross over small potential energy barriers. For complicated systems MD is usually able to locate lower energy conformations than simple energy minimization. Configurations may be saved at regular intervals during the simulation for later analysis.

More elaborate conformational searching and modeling MD studies can also be carried out using the SANDER module. This allows a variety of constraints to be added to the basic force field, and has been designed especially for the types of calculations involved in NMR structure refinement.

GIBBS

is the Free Energy Perturbation Program. It is similar to MD, but uses the ensemble of generated configurations to calculate the free energy difference between two similar states through either a perturbation or thermodynamic integration approach. The two states are defined by the user in PARM.

NMODE

is both a quasi-Newton Raphson second derivative energy minimizer and vibrational analysis program. The NMODE minimizer is capable of obtaining extremely low energy gradients. NMODE can calculate the normal modes of the system as well as numerous thermochemical properties. New features for this revision include the ability to compute "Langevin modes" (normal modes including viscous coupling to a continuum solvent,) and techniques to find transitions states as well as minima.

SOME OTHER PROGRAMS

ANAL

is for the analysis of structure and especially molecular mechanical energy of a single configuration of a system. It can be run on structures both before and after modification by the energy programs. Running ANAL on the initial configuration of your system is a good way to locate errors in the structure that result in large energies. Anal can also be used for more sophisticated analyses of energy and structure.

CARNAL

is a molecular dynamics analysis program. It is used for geometrical measurements, root mean square coordinate fitting, trajectory averaging, and other structural analyses of MD trajectories. CARNAL executes a programming language for filtering, measuring and comparing multiple streams of coordinate files (the language contains 44 keywords and uses 10 punctuation/logical characters). As an example, one can use it to build a trajectory in which the solute is positioned for minimum root mean square fit of residues in the active site and only the first shell of waters is included.

NMANAL

computes atomic fluctuations and various correlation functions from normal modes.

LMANAL

does the same things as *nmanal*, but for Langevin modes.

NUCGEN

is a nucleic acid model-building program. It is used to generate arbitrary sequences of DNA or RNA in a variety of conformations. Cartesian coordinates are output in PDB format.

See also the "Utility Programs" section of the manual.

Installation of Amber 4

These instructions describe the Unix release from UCSF. VMS-specific instructions are in the file [amber41.src.vms]vms.notes. Separate instructions will be provided with the Oxford Molecular CD distribution.

The AMBER 4 distribution is supplied on magnetic tape in Unix tar format or in compressed tar file format. Load the distribution tape on your machine and read it as you would other tar tapes, or uncompress and untar the compressed tar file. (More detailed instructions are not provided here because of system differences.) The release consists of the directory amber41, with a number of subdirectories. In total there are approximately 30 megabytes. You may want to check your disk quota if applicable before reading the tape.

Once the tape has been read, cd to amber41/src. The `0README` file there describes installation and resizing of code. Note that the most up-to-date information on operational details is *always* in the `0README` files which are found throughout the distribution. Create the configuration file `MACHINE` by copying the `src/Machine/Machine.xxx` for your machine. This file is sourced by all compilation scripts, and sets environmental variables, which include system-dependent fortran compiler directives, switches used in preprocessing the code, and the location of a system-dependent library. The distribution tape includes configuration files for a number of different machines. When the `src/MACHINE` file has been created by copying or modifying the template, then the `Makeall` script can be run to make and install all the executables, and then the validation tests can be run. For example, if you are using a Convex, the appropriate commands are:

```
cd amber41/src
cp Machine/Machine.convex MACHINE
Makeall >& mk.out &
```

and when compilation is complete,

```
cd amber41/test
Run.tests >& tst.out &
```

You may inspect the contents of mk.out while compilation is in progress, and tst.out while the tests are running, but avoid writing to these files. After compilation, examine any error messages. If warnings or errors do occur during compilation (messages about unused variables can be ignored), consult the *Compiler Warnings* section below. If there are no compilation problems, run the tests and go to the *Testing* section, below.

Installing by hand

If your system is not included among the configuration files supplied, or if you want to alter the existing file or are curious how these files hide machine dependencies, see the file `amber41/src/Machine/0README`. If you are developing or changing the `MACHINE` file, you may want to go more slowly, compiling and testing only prep first as described below.

Compiling and testing a single program. This section describes how to do manually what the Makeall and Run.tests scripts automate. You can type `make` or `make install` in any src/ directory to make the program(s) in that directory, e.g. when redimensioning arrays or otherwise modifying the code. `make clean` will cause all .o files to be removed; otherwise they stay around using significant space to conserve recompilation time.

To compile PREP (not necessary if Makeall is used):

```
cd amber41/src/prep
make install
```

This will compile prep. If the compilation is successful, the *prep* executable will be placed in *amber41/exe*.

After prep has been compiled (either alone or by Makeall), cd to amber41/test/prep, and execute Run.crown and Run.tri interactively. These will run prep using input from the demo directories, generating a number of output files. These are automatically diffed with the appropriate files in the demo directories. In general there will be few if any diffs. Occasional differences of one in the last decimal place may be seen due to precision differences on different machines. We have tried to minimize these as much as possible, but maintaining data-file compatibility with previous versions of Amber precluded making all of the software full double precision. When examining the output diffs of prep and other programs, note that a dihedral angle of 0.0 and 360.0 are actually identical.

The next thing to do is build the database, *db4.dat*. This is a binary direct access file of approximately half a Megabyte on 32 bit machines, or double that size on 64 bit machines like the Cray.

```
cd amber41/dat
make db4
```

After the database is built, if you didn't use Makeall, you need to compile the rest of the programs. (You could use Makeall now to save typing, at the expense of recompiling prep.) The source to *link, edit, parm, anal, sander, gibbs, mdanal, nmode, nucgen*, and *etc* (utilities) will be found in directories with those names, all under the *src* directory. The procedure will be the same as with *prep*. You compile with `make install`, then go to the appropriate *amber41/test/** directory and execute command files to run the validation tests. The tests take input from and compare output with files in the demo directories, which contain explanatory `0README` files. In some cases you will need to manually compare your output with the sample outputs given there, being intelligent about deciding if differences reflect round-off errors only, or are symptomatic of other problems. A few of the test cases are rather lengthy (in order to be realistic) and you may wish to skip them. Clearly, if you are not interested in normal modes, you need not compile or test the nmode program; ditto for other modules. You may want to conserve disk space by running the Makeclean script in the src directory; this will remove the object files in the src tree. Note that there is a `test/Run.tests` script that will run all the tests. See `test/0README` for details.

Testing

We have installed and tested AMBER 4 on a number of machines, including Cray, IBM, Sun, Hewlett-Packard, DEC, Convex, and Silicon Graphics hardware. However, owing to time and access limitations, not all machines for which Machine.xxx files are supplied are tested with the current code, compilers, or operating systems. Therefore we recommend running the test suites.

The distribution contains a validation suite that can be used to help verify correctness. The nature of molecular dynamics, and to a lesser extent molecular mechanics, is such that the course of

the calculation is very dependent on the order of arithmetical operations and the machine arithmetic implementation, *i.e*. the method used for roundoff. Because each step of the calculation depends on the results of the previous step, the slightest difference will eventually lead to a divergence in trajectories. As an initially identical dynamics run progresses on two different machines, the trajectories will eventually become completely uncorrelated. Neither of them are "wrong;" they are just exploring different regions of phase space. The main point of this is that states at the end of long simulations are not very useful for verifying correctness. Averages are meaningful, provided that normal statistical fluctuations are taken into account. "Different machines" in this context means any difference in floating point hardware, word size, or rounding modes, as well as any differences in compilers or libraries. Differences in the order of arithmetic operations will affect roundoff behavior; (a + b) + c is not the same as a + (b + c). Different optimization levels will among other things affect operation order, and may therefore affect the course of the calculations.

When comparing the output from two different machines for purposes of verification, it is very important that identical input files be used to generate both sets of output. The validation suite uses matched inputs and outputs in the amber41/demo/ tree, which is set to read-only to help you avoid overwriting them with files created on your machine. Testing takes place in the amber41/test/ tree.

The single-precision setup programs *prep, link, edit,* and *parm* will occasionally show differences of +/– 1 in the last decimal place of floating point values. The double precision version of sander should be used for verification purposes. *Gibbs* and *nmode* are only provided in full double precision versions. All initial values reported as integers should be identical. The energies and temperatures on the first cycle should be identical. The RMS and MAX gradients reported in sander are often more precision sensitive than the energies, and may vary by 1 in the last figure on some machines. As is the case with sander, the trajectory in a Gibbs simulation will diverge, but the resulting free energy should not if the simulation is run to convergence (this is not done because of the time involved). In minimization and dynamics calculations, it is not unusual to to see small divergences in behavior after as little as 1-200 cycles, if the two machines being compared have very different numerical behavior.

Precision. In the *sander* program it is possible to generate either single or double precision versions of the executables. (On Cray machines, the precision is always single word, 64-bit). The double precision version is the "default", with the executable name *sander*. A single precision version is also occasionally useful, and is called *spsander*. The double precision version is numerically more stable, and gives more consistent results between different machines. It can be slower than the single precision version on many machines, typically by 25-100%. The single precision minimizer is good for quick minimizations prior to MD equilibration, but may not be capable of achieving a very low energy gradient.

In general, compiler and optimizer errors are fairly obvious, and result in rather large changes in the output, if you get any output at all. See `test/0README` for examples of acceptable output differences and discussion of peculiarities of various machines.

Compiler Warnings

Some compilers (e.g. bsd f77) will generate warnings because of initializations of Hollerith data. They will say "integer variable initialized with character datum" or something similar. These may be ignored. If your compiler actually can not deal with Hollerith data, you are in trouble, but this is not likely. Smart compilers may find some unreachable code in parm and mdanal, and possibly other places. In general these are code stubs that have been hardwired off, or a check on improper parameter

values, and are not a problem. If any other warnings or errors are produced during compilation, they should be taken seriously. See the section below on Assumptions.

Assumptions

This installation guide assumes the existence of a 'normal' Unix system. This should include the following software: /bin/csh, /lib/cpp, egrep, and sed. With the exception of subroutine *putres* in *prep* and *transf* in *link*, and various parts of *anal* and *mdanal*, the fortran code does not assume that sizeof(int) = sizeof(real), so it can be safely run in double precision on 32 bit machines. The code deviates from the ANSI fortran 77 standard only in ways which are widely supported. These programs still use Hollerith data, but do not do any bitwise manipulation of it, and limit it to 32 bit integers. Files are assumed to be positioned at the top on an OPEN. If your compiler is pathological, you might need to add a rewind statement in subroutine amopen. Some Amber programs (e.g. *nmode*) declare scratch arrays of one type, then pass them to subroutines where they are declared and used as a different type. It is being used only as a crude memory management technique, and does not rely on specific bit patterns in any way. If your compiler won't buy it, you'll have to fix it by adding the appropriate equivalences in the calling routines.

Memory Requirements

The AMBER 4 programs as distributed are dimensioned for a fairly large system (about 10K atoms), and you may want to change their dimensions to be more appropriate for the machine you are using if you are running in a tight memory environment. See `src/0README` for information on resizing. Some programs use local scripts called resize.csh; this script uses the stream editor sed, and employs regular expression matching to correctly redimension the code regardless of what its dimensions are currently set to, even if the same parameter has been inadvertently set to different values in different modules. Here we describe dimensioning of sander as an example.

In src/sander/sizes.h, you will find the following parameters:

```
parameter (MAXINT=1300000)
parameter (MAXPR=2500000)
parameter (MAXREA=340000)
parameter (MAXHOL=200000)
parameter (MAXDUP=5000)
```

The actual memory requirements for a particular job can be determined from the output of SANDER or GIBBS. An annotated example follows:

```
1.  R E S O U R C E   U S E:


  Memory Use       Allocated      Used
  Real              340000        19109  <-- minimum MAXREA
  Integer          1300000        29904  <-- minimum MAXINT

  Max Nonbonded Pairs: 1270096 packed  1 to a machine word
                                       ^
                                    "NWDVAR"
  Duplicated   26 dihedrals

  Duplicated   94 dihedrals  <-- minimum MAXDUP
      .
      .
      .

   NB-UPDATE: NPAIRS =  150395  HBPAIR =    2804
```

As is shown above, MAXREA must be at least as large as the number of Real words used. It can be read directly off the output. MAXDUP must be at least as large as the larger of the two 'duplicated' values given, in this case 94. The actual amount of memory controlled by MAXDUP is 10 times its value. MAXINT is slightly more complicated, since it depends on the type of pairlist packing used. The number of NB pair pointers packed in a native integer word, NWDVAR, is printed in the output as shown. It will be 1 or 2 on byte-oriented machines, 1, 2, or 4 on 64 bit machines like Cray or FPS264. The minimum value of MAXINT is determined by the sum of the static integer requirement given in the output. For gibbs, MAXINT also includes the requirement for the pairlist, while in sander the pairlist size is determined by MAXPR. The pairlist requirement is the total number of nonbonded pairs, NPAIRS, divided by NWDVAR. Because the number of pairs may grow or shrink during a run, you should include a safety factor of 5-10% extra for NPAIRS. The algorithm to determine the MAXPR (sander) or MAXINT pairlist component is thus:

(NPAIRS/NWDVAR) * 1.1

In our Cray and FPS264 implementations, we use explicit packing functions that require the use of an extra NATOM words of memory. For those machines, the formula is:

(NPAIRS/NWDVAR) * 1.1 + NATOM

More detailed documentation on memory use and packing configuration is found in sander/sizes.h.

NOTE: For gibbs, the variables which define memory allocation are MAXREA, MAXINT, and MAXCHR. MAXREA and MAXINT can be set as described above. MAXCHR allocates character storage, is typically small, and scales linearly with the number of atoms. These parameters are scaled to a single parameter, *memmax*.

The database directory

There are two main types of force field file in the amber41/dat/ directory: residue descriptions for building the PREP database, and force field files for PARM. The residue descriptions include topologies, atom types and charges and have `.in` extensions. The PARM force field files contains parameters mapped to the atom types: mass, Van der Waals, bond, angle, torsional and hydrogen bonding terms. These files have names matching the pattern, `parm*.dat`.

Files. The following files are found in the database directory `amber41/dat/`:

DATABASE AND DATABASE INPUT FILES:

```
db94.dat            Residue database for the 1994 force field.
all_nuc94.in        Nucleic acid input for building database.
all_amino94.in      Amino acid input for building database.
all_aminoct94.in    COO- amino acid input for database.
all_aminont94.in    NH3+ amino acid input for database.

db4.dat             Residue database for the 1991 force field.
uni.in              United atom database input.
unict.in            United atom database input, COO- Amino acids.
unint.in            United atom database input, NH3+ Amino acids.
all.in              All atom database input.
allct.in            All atom database input, COO- Amino acids.
allnt.in            All atom database input, NH3+ Amino acids.

opls_uni.in         Normal OPLS residues.
opls_unict.in       OPLS COO- Amino acids.
opls_unint.in       OPLS NH3+ Amino acids.
```

FORCE FIELD PARAMETER FILES:

```
parm94.dat          1994 force field file.
parm91.dat          1991 force field file.
opls_parm.dat       OPLS force field file.
```

STANDARD PROGRAM INPUTS:

```
wat216.dat          Cube of 216 TIP4P waters, MC liquid.
nucgen.dat          Nucgen nucleic acid conformations.
```

OTHER PREP FILES:

```
nacl.in             Ion prep file
```

FORCE FIELD DOCUMENTATION:

`0README` `Documentation for Force Field files`

(1) **1994 parameters.** These parameters are especially derived for solvated systems, and when used with an appropriate 1-4 electrostatic scale factor, have been shown to perform well at modelling the small molecules examined to date. The parameters in *parm94.dat* omit the hydrogen bonding terms of earlier force fields.

The main files in the amber41/dat/ directory that users normally need are `db94.dat` (for link) and `parm94.dat` (for parm). This is an all-atom force field; no united-atom counterpart is provided. 1-4 electrostatic interactions are scaled by 1.2 instead of 2.0; *users must make this adjustment in their input files for sander, gibbs etc. when using this force field.*

Nucleic acid residue definitions are different - see the LINK documentation for the new residue names. Note that old Amber PDB format files need to be massaged for use with this force field, *i.e.* terminal hydrogens and phosphates are no longer separate residues, so the names and numbers of the residues need to be changed. (NUCGEN has been modified to accept the new names.)

Charges for the old AMBER (Weiner *et al.*) force field were derived using the STO-3G basis set. The 6-31G* basis set was used for the new charges because it exaggerates the dipole moment of most residues by 10-20%. It thus ''builds in'' the amount of polarization which would be expected in aqueous solution. This is necessary for carrying out condensed phase simulations with an effective two-body force field which does not include explicit polarization.

The charge-fitting procedure is described at length in the Appendix.

The alkali ions with explicit waters adapted from the parameters of Åqvist (see 1991 notes below) have been renamed:

Li: Li+ IP: Na+ K: K+ Rb: Rb+ Cs: Cs+

(2) **1991 parameters.** These parameters may still be useful for vacuum simulations of nucleic acids and proteins using a distance-dependent dielectric. The material in *parm91.dat* is the parameter set distributed with Amber 4.0. It is derived from *oldff3a/parmallhb.dat* with the following changes: The mass of all hydrogens has been reduced from 3.0 to 1.008 and the lone pair mass has been reduced from 12.0 to 3.0. TIP3P atom type HW has been corrected to have 6-12 coefficients of zero. The bending parameters for the lone pairs around sulfur have been changed from their uniform value of 600 kcal/mole rad**2 to 150 kcal/mole rad**2. The previous extremely stiff force constants led to erratic and sometimes catastrophic behaviour in minimization and dynamics. The softer force constants behave much better without changing significantly any structural properties. (G. Seibel, Ph.D. thesis, UCSF, 1990). Finally, the *STUB* nonbonded set has been copied from *parmuni.dat*; these sets of parameters are appropriate for united atom calculations using the "larger" carbon radii referred to in the "note added in proof" of the 1984 JACS paper. If these values are used for a united atom calculation, the parameter *scnb* should be set to 8.0, not its default value of 2.0.

A number of terms in the non-bonded list of parm91.dat should be noted. The non-bonded terms for I(iodine),CU(copper) and MG(magnesium) have not been carefully calibrated, but are given as approximate values. In the STUB set of non-bonded parameters, we have included parameters for a large hydrated monovalent cation (IP) that represent work by Singh et al 1985 on large hydrated counterions for DNA. Similar values are included for a hydrated anion (IM).

For alkali ions with explicit waters, we have provided the latest values of Åqvist (J. Phys. Chem., 1990, 94: 8021-8024) which are adjusted for Amber's nonbonded atom pair combining rules to give the same ion-OW potentials, in order to reproduce the first peak of the radial distribution for ion-OW and the relative free energies of solvation in water of the various ions. These are included in the standard (STDA) parameter file. The atom types are:

QC: Cs+ QK: K+ QL: Li+ QN: Na+ QR: Rb+

The file *opls_parm.dat* is a parameter set appropriate for use with the AMBER/OPLS parameter set as described by Tirado-Rives and Jorgensen. [2]

(3) **1984 and 1986 parameters.** Parameters from the 3.0 and 3.0 Rev A releases are in the dat/oldff3a/ directory. These files should only be consulted if you need compatibility with that version. They are *not* recommended for current use. The file *oldff3a/parmuni.dat* is the 1984 united atom parameter set. [3] The file *oldff3a/parmall.dat* is the 1986 all atom parameter set. [4] The *parmall.dat* also contains the united atom parameters, and can be used to run a mixed united atom / all atom system.

The two files in the oldff3a/ directory, *parmunihb.dat* and *parmallhb.dat*, have been modified in the following way: Parameters for TIP3P water have been included, i.e., atom types OW and HW for the oxygen and hydrogen, a bond length parameter, OW-HW and HW-HW, the latter to be used when using SHAKE, i.e., rigid water molecules, as was the spirit of the TIP3P potential, and a reasonable bond angle parameter for allowing water flexibility. The 10-12 list in these two parm files has been set up in the following way: All the possible H-bond combinations not discussed and considered by Weiner *et al*. have been filled in according to the following rules: For non-sulfur H-bonds, we have three classes: large (C=10238, D=3071), medium (7557,2385) and small (4019,1409). For sulfur as a proton acceptor, the default values are (265720,35429) and for a hydrogen covalently bonded to sulfur the default value is (14184,3082). For ionic H-bonds (those involving H3 or O2), the small parameters are used, except for the sulfur proton acceptor, where the medium is used. For NC in the nucleic acids in neutral H-bonds the large is used, except for HS as proton donor, where the sulfur value is used. For every other neutral H-bond, the medium value is used, except for H2--O, which corresponds to the Watson-Crick H-bonds and for which the larger value is used.

Note: There are two possible models for handling TIP3P water - solute H-bonds: One is to set all 10-12 parameters involving HW or OW to zero, which would be the "philosophy" inherent in the TIP3P and OPLS models, where there are no non-bonded terms involving hydrogen bonding hydrogens. The other is to include non-zero interactions between HW and solute atoms, consistent with the Weiner *et al*. results on protein-protein H-bonds. It is the latter which has been implemented in parmunihb and parmallhb. It should be noted that preliminary molecular mechanics calculations on N-methyl acetamide/water interactions using the two models give very similar results for interaction energies, but the model with 10-12 parameters gives distances in better agreement with the best available ab initio calculations. Free energy simulations using the two models can yield significantly different results. [5] To remove the 10-12 interactions between water and solute, set the all 10-12 parameters involving atom types HW and OW to

---

[2] W.L. Jorgensen and J. Tirado-Rives, *J. Am. Chem. Soc.* **1988,** *110,* 1657.

[3] S. J. Weiner, P. A. Kollman, D. A. Case, U. C. Singh, C. Ghio, G. Alagona and P. Weiner, *J. Am Chem. Soc.* **106,** 765-784 (1984).

[4] S. J. Weiner, P. A. Kollman, D. T. Nguyen and D. A. Case. *J. Comput. Chem.* **7,** 230-252 (1986).

[5] D.A. Pearlman and P.A. Kollman *J. Chem. Phys.***94,** 4532-4545 (1991).

zero. In addition to the use of 10-12 parameters for water-solute hydrogen bonds, these two parameter files have a 6-12 term for HW. We have removed this in the Rev A parameter file described below.

Finally, the *oldff3a* directory contains *parm89a.dat*, which is the force field distributed with Amber 3A. This is identical to the current force field, with two exceptions: *parm89a.dat* has a zero force constant for LP-S-LP angles; this caused some problems, and has been set to 150.0 in the current force field. Second, the "STUB" non-bonded was not in *parm89a.dat* (see below).

OLD FORCE FIELD FILES (in oldff3a):

```
db3a.dat          The Rev A database--has non-IUPAC names
makedb3a.com      Script to build the Rev A database.
parmall.dat       1986 force field
parmallhb.dat     1986 force field + water, more Hbonds
parmuni.dat       1984 force field
parmunihb.dat     1984 force field + water, more Hbonds
parm89a.dat       Force field distributed with Amber 3A
```

An Amber Tutorial

AMBER is a suite of programs for use in molecular modeling and molecular simulations. It consists of a substructure database, a force field parameter file, and a variety of useful programs. Here we give some commented sample runs to provide an overview of how things are carried out. The examples do not use the *LEaP* or *interface* programs, and only a cover a fraction of the things that it is possible to do with AMBER. The formats of the example files shown are described in detail later in the manual, in the chapters pertaining to the programs.

Example 1. Minimization of BPTI in vacuum

Step 1. Generate some starting coordinates.

The first step is to obtain starting coordinates. We begin with the file *6pti.pdb*, exactly as distributed by the Protein Data Bank and Brookhaven. This file (as with most Brookhaven files) needs some editing before it can be used by Amber. First, alternate conformations are provided for residue 50, so we need to figure out which one we want. For this example, we choose the "A" conformation, and manually edit the file to remove the alternate conformers. Second, coordinates are provided for a phosphate group and a variety of water molecules. These are not needed for the calculation we are pursuing here, so we also edit the file to remove these. Let's call this modified file *6pti.mod.pdb*. Third, hydrogen positions are not included, so we run the Amber program *protonate* to provide these:

```
protonate -d amber41/dat/PROTON_INFO < 6pti.mod.pdb > 6pti.H.pdb
```

In other situations, many different programs could be used to generate starting coordinates, but the basic ideas are the same: somehow generate what you want in a "pdb" format, then run the result through *protonate*. We recommend doing the last step even if protons are present, since protonate performs a number of checks on the correctness and naming of hydrogen atoms.

Step 2. Run LINK to establish the topology.

The following script will accomplish this by creating an input file and running LINK with a prep database:

Running link for BPTI

```
cat <<eof >lnkin
 bpti


DU
    0    0    0    0    0
 bpti
P   1    0    1    3    1
ARG 2PRO  ASP  PHE  CYX  LEU  GLU  PRO  PRO  TYR  THR  GLY
PRO  CYX  LYS  ALA  ARG  ILE  ILE  ARG  TYR  PHE  TYR  ASN
ALA  LYS  ALA  GLY  LEU  CYX  GLN  THR  PHE  VAL  TYR  GLY
GLY  CYX  ARG  ALA  LYS  ARG  ASN  ASN  PHE  LYS  SER  ALA
GLU  ASP  CYX  MET  ARG  THR  CYX  GLY  GLY  ALA

    5   55SG  SG       0
   14   38SG  SG       0
   30   51SG  SG       0

QUIT
eof
#
link -i lnkin -o lnkout -p  $AMHOME/dat/db4.dat
/bin/rm lnkin
```

You should interpret the file given above using the input description for *link*. Basically, the first seven lines contain operation flags, many of which are almost always the same. The next four lines give the amino acid sequence, then come lines that establish cross-links (disulfide bonds) between residues 5-55, 14-38 and 30-51. The UNIX `AMHOME` variable should be set to the location of Amber on your system, and your `PATH` variable should allow the program *link* to be found. The above script will create a text output file *lnkout* (which you should read), and a binary file *lnkbin*, which will be used as input to the next step.

This step informs AMBER of the "topology" of the system: what all the atoms are called, what their "types" are (needed to set up a force field), and where all the bonds are. All this information was assembled from the sequence and the information about amino acids that is contained in the `db4.dat` file.

Step 3. Run EDIT to insert the starting coordinates.

The following script will accomplish this:

Running edit for BPTI

```
cat <<eof > edtin
bpti, 5pti structure
    0    0    0    0
XYZ
OMIT
XRAY
    0    0    0    0    0
QUIT
eof
#
edit -i edtin -o edtout -pi 6pti.H.pdb
/bin/rm edtin
```

The XRAY option reads in a Brookhaven format file and compares the atoms in that file to what Amber expects to see; when it finds matches it inserts the proper coordinates into the system, and it reports errors when it fails to find matches. In this case, all the atoms are present, and no warning messages should be obtained.

The output from the above script will be a text file called *edtout* and a binary file *edtbin*, which will be used in the next step.

Step 4. Run PARM to connect the force field to the protein.

This is done by this simple script:

Running parm for BPTI

```
cat <<eof >prmin
 name of system
BIN FOR STDA

    0

eof
#
parm -i prmin -o prmout  -f $AMHOME/dat/parm91.dat
/bin/rm prmin
```

In this step, the *parm* program looks through the molecular information in *edtbin* and determines all the types of "parameters" (force constants, bond lengths, non-bonded sizes, etc.) that are necessary to calculate the energy of BPTI. It then searches through the *parm91.dat* file to find the parameters. The program will complain if something is missing, but this is just a standard protein, and everything is in place. The output is a text file *prmout*, which you should read, and data files *prmtop* and *prmcrd* that will be used in the next step. The *prmtop* and *prmcrd* files are ascii files, so can be moved easily from one machine to another. (It is common to run *link*, *edit*, and *parm* on a workstation, then transfer the *prmtop* and *prmcrd* files to a more powerful computer for minimization and dynamics.) The *prmtop* file contains all the information needed to compute the energy of a molecule, and *prmcrd* contains the coordinates (in this case, the starting coordinates.) This division makes sense since minimization or dynamics will change the coordinates but not the make-up of the molecule.

Step 5. Perform some minimization.

Use this script:

Running minimization for BPTI

```
cat << eof > min.in
#  do 200 steps of minimization:
 &cntrl
    maxcyc=200, imin=1, cut=12.0, nsnb=20, idiel=0, scee=2.0, ntpr=10,
 &end
eof
sander -i min.in -o 6pti.min1.out -c prmcrd -r 6pti.min1.xyz
/bin/rm min.in
```

This will perform minimization (`imin`) for 200 steps (`maxcyc`), using a nonbonded cutoff of 12 'angstroms' (`cut`) and a distance-dependent dielectric constant (`idiel`). The list of non-bonded pairs will be updated every 20 steps (`nsnb`), and intermediate results will be printed every 10 steps (`ntpr`). Text output will go to file *6pti.min1.out*, and the final coordinates to file *6pti.min1.xyz*.

Example 2. Peptide with a non-standard residue.

As an example, suppose you want to minimize an enzyme - substrate complex, and that you have a standard PDB file with coordinates for the enzyme and substrate, which we will call 'model.pdb'. Such a file might come from X-ray crystallography or model building. PDB files generally don't contain connectivity information, so this must be provided. In addition, each atom of the system must be assigned the appropriate AMBER atom type so the correct force field parameters will be applied. For the amino acid residues of the protein, this connectivity and atom type information already exists, keyed by residue name in the AMBER database, and need not be specified. However, the substrate will require that you input connectivity and atom type information. This is done using the program PREP. See PREP.DOC for details. The input for prep consists of only one file, in this case subprep.in. Running PREP will give you two output files. One of the files, sub.res, is a residue topology file for your substrate. It will have the same format as the amino acid residues in the standard data- base. The other file, prep.out, is a list of diagnostic information.

Amber programs are usually run through the use of command files. In Unix environments, the command file contains the name of the executable and its command-line arguments. Under VMS the command file contains operating system commands for setting the default directory, assigning files to fortran logical units, and running the executable program. A command file named filename.com is executed on VMS by entering @filename. In the following examples, one will see that the control file for each program is named *filename.in*, and is always assigned to unit 5. The output file containing user information and diagnostics is called *filename.out* and is assigned to unit 6. The binary topology files that are passed from module to module are called filename.bin, where filename is the name of the module that created it. The logical unit assignment of these files varies from program to program. Residue files from prep have names ending in ".res", pdb files have names ending in ".pdb", and coordinate files created by AMBER are usually named *name.crd*. It is not essential that these naming conventions be adhered to but it will facilitate communication with other AMBER users. (See `demo/0README` for more naming conventions.) The following two files are the command and input files that create the substrate residue file using PREP.

Running PREP

Unix:

```
~amber41/exe/prep -i subprep.in -o prep.out
```

VMS:

```
$ set default [yourdir.tet]
$ assign subprep.in for005
$ assign prep.out for006
$ run [amber41.exe]prep
```

Here is the "subprep.in" file; strip comments before using.

```
0 0 1                !control for database generation
                     !blank card
substrate            !title
sub.res              !name of output file
SUB INT   0          !control parameters - see PREP.DOC
CORRECT OMIT DU  BEG

 1 DUMM DU M 0 0 0 0.   0.  0.  0.
 2 DUMM DU M 1 0 0 1.449  0.  0.  0.
 3 DUMM DU M 2 1 0 1.522 111.1  0.  0.
 4 N  N M 3 2 1 1.335 116.6 180. -0.5200
 5 HN  H E 4 3 2 1.01 119.8  0. 0.2480
 6 CA  CH M 4 3 2 1.449 121.9 180. 0.2270
 7 CB  C2 S 6 4 3 1.525 111.1  60. 0.0390
 8 CG  C2 S 7 6 4 1.525 109.47 180. 0.0530
 9 CD  C2 S 8 7 6 1.525 109.47 180. 0.0480
10 CE  C2 S 9 8 7 1.525 109.47 180. 0.2180
11 NZ  N3 3 10 9 8 1.47 109.47 180. -0.2720
12 HNZ1 H3 E 11 10 9 1.01 109.47 60. 0.3110
13 HNZ2 H3 E 11 10 9 1.01 109.47 180. 0.3110
14 HNZ3 H3 E 11 10 9 1.01 109.47 300. 0.3110
15 C  JJ M 6 4 3 1.522 111.1 180. 0.5260
16 O  O2 E 15 6 4 1.229 120.5  0. -0.5000



IMPROPER
-M  CA  N  H
CA  +M  C  O
CB  CA  N  C

DONE
STOP
```

The next step is to link the appropriate residues from the standard database, along with the residue file you created with PREP into a macromolecule. This is done using the program LINK. Note that if you were only interested in the enzyme and not the substrate, you would start at this point. The

third line of link.in tells LINK that the topological information for residue "SUB" is in the file sub.res (the "standard" residues are retrieved from the prep database file specified with the '–p' argument). The residues of the enzyme are listed sequentially in the order that they are to be bonded. The substrate residue is put at the end, separated by the spacer residue "***" indicating that it is not covalently attached. Alternatively it could be specified as a separate molecule. After the residue sequence, disulfide crosslinks are input. Any desired covalent attachment can be input as a crosslink. See LINK.DOC for details. Running LINK again produces two files: link.bin, the molecular topology file, and link.out, which contains diagnostic information.

Running LINK

Unix:

```
~amber41/exe/link -i link.in -o link.out -l link.bin -p db4.dat
```

VMS:

```
$ SET DEFAULT [YOURDIR.TET]
$ ASSIGN [AMBER.DAT]DB4.DAT FOR001
$ ASSIGN LINK.IN FOR005
$ ASSIGN LINK.OUT FOR006
$ ASSIGN LINK.BIN FOR010
$ RUN [AMBER41.EXE]LINK
```

LINK.IN:

```
TACK'S PROTEIN       !title
                     !blank line
SUB   0sub.res       !filename for residue SUB
                     !blank card
DU                   !symbol for dummy atoms
  0  0  0  0  0      !print controls
tacks protein        !subtitle for first molecule
P  1  0  1  3  1     !control parameters for first molecule
ASP 1SER  CYX  GLU  ALA  ILE  ILE  HIP  GLU  LEU  HID  SER
ARG  HID  PRO  GLY  ASP  PHE  GLY  ALA  ASP  ALA  GLN  GLY
ALA  MET  ASN  LYS  ALA  CYX  GLU  SER  ***  SUB   !residue list
                     !blank card
  3  30SG SG   0     !crosslink info
                     !blank card
QUIT                 !exit control
```

The binary file link.bin contains your system, but at this point it lacks the correct atomic coordinates. It does contain the internal coordinates for each residue, but the residues are linked with arbitrary dihedral angles. The file also contains some pseudo atoms called "dummy" atoms. They are there to define the space axes for the internal coordinate system and must be removed. The addition of correct coordinates and removal of dummy atoms is accomplished with the program EDIT. Input for EDIT consists of a small control file, edit.in; the topology file from LINK, link.bin; and your PDB file. Two files are output: edit.bin, the molecular topology file (now with correct coordinates and dummy atoms removed), and edit.out containing user information and diagnostics. A look at edit.out generally reveals some frightening diagnostics stating that input for some atoms was not found. What this actually means is that the PDB file was missing some atoms present in the database residues, or had some

extra atoms not present in the database (sometimes these are the same atoms, with different names). If atoms in the PDB file are missing, edit will add them using the stored internal coordinates of the residues. In the event that this can't be done (notably for the very first atoms in a molecule), the correct orientation of the atoms can be specified on edit.in using the "ABC" option of EDIT. Extra atoms in the PDB file are ignored. Some important notes: EDIT expects the residue sequence of the pdb file to match the link input file. If any residues are missing or extra ones are present, the program will stop with an error message. The ordering of atoms within a residue does not matter, nor do the atom sequence numbers, however, all atom records for a given residue should have the same residue sequence number. Since Rev A, EDIT reads real PDB files, unlike earlier versions of EDIT.

Running EDIT

Unix:

```
~amber41/exe/edit -i edit.in -o edit.out -l link.bin
      -e edit.bin -pi model.pdb -po edit.pdb
```

VMS:

```
$ set default [yourdir.tet]
$ assign edit.in for005
$ assign edit.out for006
$ assign link.bin for010
$ assign edit.bin for012
$ assign model.pdb for015
$ run [amber41.exe]edit
```

EDIT.IN:

```
tacks protein          !title
  0  0  0  0           !print controls
XYZ                    !select xyz option
OMIT                   !xyz input - omit dummy atoms
XRAY                   !select xray option
  0  0  0  0           !xray input
ABC                    !select abc option
  1    0               !abc input
  1  1.01 109.5 60.0  2  5  6   !abc input
  3  1.01 109.5 180.0  2  5  6  !abc input
  4  1.01 109.5 300.0  2  5  6  !abc input
                       !blank card
   QUI                 !terminate abc option
QUIT                   !terminate edit
```

All that remains to be done is add force field parameters to the molecular topology file, and you will be ready to run either molecular mechanics or molecular dynamics. Force field parameters are added with the program PARM. Input for PARM consists of a control file; parm.in, a parameter file; parm91.dat, and the topology file from EDIT; edit.bin. parm91.dat is part of the AMBER 4 distribution. Output from PARM consists of the completed topology file, parm.top, a coordinate file, parm.crd, and the diagnostics file parm.out. The finished topology and coordinate files can be written either in binary form or formatted form. In general we now use only the formatted form for all files, so they can be used on various machines regardless of the underlying representation of data. It is

important to look at parm.out to make sure that all the needed parameters were found in parm91.dat. If you are only working with amino acids, nucleic acids, or water, they should all be there. However, it is very easy to construct a molecule for which parameters do not exist in parm91.dat. In that event you will have to create some on your own. Often parameters for similar bonding situations can be found in parm91.dat, and simply duplicated in that file with the appropriate atom types. Otherwise see Hopfinger and Pearlstein, J. Comp. Chem., 5, p486, 1985. This article tells how to generate any needed force field parameters for general molecular mechanics use.

Running PARM

Unix:

```
~amber41/exe/parm -i parm.in -o parm.out
     -e edit.bin -f ~amber41/dat/parm91.dat
     -c parm.crd -p parm.top
```

VMS:

```
$ SET DEFAULT [YOURDIR.TET]
$ ASSIGN PARM.IN FOR005
$ ASSIGN PARM.OUT FOR006
$ ASSIGN [AMBER41.DAT]PARM91.DAT FOR010
$ ASSIGN PARM.TOP FOR012
$ ASSIGN EDIT.BIN FOR015
$ ASSIGN PARM.CRD FOR018
$ RUN [AMBER41.EXE]PARM
```

PARM.IN:

```
tack helix      !title
BIN FOR STDA    !format controls + nonbon param set name
  0   0   0     !print flags
  1   1   1     !print flags
```

Now you are finally ready to run SANDER, the molecular mechanics/dynamics module. In honor of its predecessor, minmd, and because the name is more self-explanatory, the input and output files are here named minmd.xxx rather than sander.xxx. Input for this program consists of a control file, minmd.in, the topology file from PARM; parm.top, and the coordinate file from PARM; parm.crd. Output from SANDER consists of the final coordinates, minmd.crd, and a record of the molecular mechanical energies of the system as the minimization/dynamics proceeds, minmd.out. Output from a dynamics run may optionally include files containing the dynamics trajectory and velocities of all the atoms of the system over the course of the simulation.

SANDER.COM: This file will typically be submitted to a batch queue, or run in the background at reduced priority.

Running SANDER

Unix:

```
~amber41/exe/sander -i minmd.in -o minmd.out -p parm.top
     -c parm.crd -r minmd.crd -inf minmd.inf
```

VMS:

```
$ SET DEFAULT [YOURDIR.TET]
$ ASS MINMD.IN FOR005
$ ASS MINMD.OUT FOR006
$ ASS PARM.BIN FOR020
$ ASS COORD.DAT FOR021
$ ASS COORD.OUT FOR033
$ RUN [AMBER41.EXE]SANDER.EXE
```

MINMD.IN:

This file uses the new *namelist* style of input.

```
Tack Helix: 500 steps min, constant dielectric
 &cntrl imin = 1, maxcyc = 500, nrun = 0 nsnb = 50,
  idiel = 1, cut = 8.0, scee = 2.0,
 &end
```

Example 3. A more complicated protein example.

This section works through in some detail setting up a protein simulation in AMBER. The example is for plastocyanin in water, and contains a number of things that experienced AMBER users know how to do, but which may be far from obvious for others. In particular, there are a couple of items that go beyond a simple protein:

(1) Plastocyanin contains a metal ion bound to four amino acids, and I also want to modify a methionine residue that is bound to the copper in such a way that it has a different type of sulfur than is found in the standard database.

(2) The Brookhaven crystallographic file (1PLC) contains crystallographic waters, which I want to keep. Only the oxygen positions are provided, so I will need to try to figure out where to put protons.

(3) Somewhat unusually, this PDB file has proton positions for the protein, which I would like to keep. However, Brookhaven uses proton names that are different than what NMR spectroscopists use, and I would like to be able to use the latter to make easy contact with NMR results.

(4) Using the most probable ionization states of the protein (at neutral pH) results in a protein with a net charge of -8, so I would like to include mobile counterions in the solution to create an overall neutral system.

This will be a lot of work, but it's infinitely easier now in AMBER than it used to be.

Step 1: Make database file for the modified residues.

For plastocyanin, I will define two new types of residues: HIC, which will be a histidine coupled to a copper ion, and which will take the place of HIS 37 in the "real" sequence, and MEM, which is a modified methionine where the sulfur atom is of type "SM" rather than type "S". "SM" is a type I made up, and will use to create special force field parameters for MET 94, which is bonded to the copper ion with a fairly long bond.

Here are the input files for these two residues:

hicu_all.in

```
    0    0    2

HISTIDINE epsilon-H, with copper attached to N-delta
hicu_all.db4
HIC   INT     1
 CORRECT OMIT DU   BEG
   0.00000
   1   DUMM  DU    M      0  -1  -2       0.0000      0.0000      0.0000
   2   DUMM  DU    M      1   0  -1       1.4490      0.0000      0.0000
   3   DUMM  DU    M      2   1   0       1.5220    111.1000      0.0000
   4   N     N     M      3   2   1       1.3350    116.6000    180.0000
   5   H     H     E      4   3   2       1.0100    119.8000      0.0000
   6   CA    CT    M      4   3   2       1.4490    121.9000    180.0000
   7   HA    H1    E      6   4   3       1.0900    109.5000    300.0000
   8   CB    CT    3      6   4   3       1.5250    111.1000     60.0000
   9   HB2   HC    E      8   6   4       1.0900    109.5000     60.0000
  10   HB3   HC    E      8   6   4       1.0900    109.5000    300.0000
  11   CG    CC    S      8   6   4       1.5100    115.0000    180.0000
  12   ND1   NB    B     11   8   6       1.3900    122.0000    180.0000
  13   CU    CU    E     12  11   8       2.05      109.0      180.
  14   CE1   CR    B     12  11   8       1.3200    108.0000    180.0000
  15   HE1   H5    E     14  12  11       1.0900    120.0000    180.0000
  16   NE2   NA    B     14  12  11       1.3100    109.0000      0.0000
  17   HE2   H     E     16  14  12       1.0100    126.0000      0.0000
  18   CD2   CW    S     16  14  12       1.3600    110.0000      0.0000
  19   HD2   H4    E     18  16  14       1.0900    120.0000    180.0000
  20   C     C     M      6   4   3       1.5220    111.1000    180.0000
  21   O     O     E     20   6   4       1.2290    120.5000      0.0000

CHARGE
  -0.41570   0.27190  -0.05810   0.13600  -0.00740
   0.03670   0.03670   0.18680  -0.54320   1.00000
   0.16350   0.14350  -0.27950   0.33390  -0.22070
   0.18620   0.59730  -0.56790

LOOP CLOSING EXPLICIT
 CG   CD2

IMPROPER
 -M   CA   N    HN
 CA   +M   C    O
 CE1  CD2  NE2  HE2
 CG   NE2  CD2  HD2
 ND1  NE2  CE1  HE1
 ND1  CD2  CG   CB

DONE
STOP
```

met.in

```
    0    0    2

METHIONINE, with SM atom type for the sulfur
met.db4
 MEM  INT     1
 CORR OMIT DU   BEG
   0.00000
   1  DUMM  DU    M    0  -1  -2     0.000     0.000     0.000   0.00000
   2  DUMM  DU    M    1   0  -1     1.449     0.000     0.000   0.00000
   3  DUMM  DU    M    2   1   0     1.522   111.100     0.000   0.00000
   4  N     N     M    3   2   1     1.335   116.600   180.000  -0.41570
   5  H     H     E    4   3   2     1.010   119.800     0.000   0.27190
   6  CA    CT    M    4   3   2     1.449   121.900   180.000  -0.02370
   7  HA    H1    E    6   4   3     1.090   109.500   300.000   0.08800
   8  CB    CT    3    6   4   3     1.525   111.100    60.000   0.03420
   9  HB2   HC    E    8   6   4     1.090   109.500   300.000   0.02410
  10  HB3   HC    E    8   6   4     1.090   109.500    60.000   0.02410
  11  CG    CT    3    8   6   4     1.525   109.470   180.000   0.00180
  12  HG2   H1    E   11   8   6     1.090   109.500   300.000   0.04400
  13  HG3   H1    E   11   8   6     1.090   109.500    60.000   0.04400
  14  SD    SM    S   11   8   6     1.810   110.000   180.000  -0.27370
  15  CE    CT    3   14  11   8     1.780   100.000   180.000  -0.05360
  16  HE1   H1    E   15  14  11     1.090   109.500    60.000   0.06840
  17  HE2   H1    E   15  14  11     1.090   109.500   180.000   0.06840
  18  HE3   H1    E   15  14  11     1.090   109.500   300.000   0.06840
  19  C     C     M    6   4   3     1.522   111.100   180.000   0.59730
  20  O     O     E   19   6   4     1.229   120.500     0.000  -0.56790

IMPROPER
 -M   CA   N    H
 CA   +M   C    O

DONE
STOP
```

I made *met.in* just by copying the relevant portions of the methionine entry from *all_amino94.in* in the database directory, changing the atom type of the sulfur, and adding appropriate first and last lines. Similar things were done for the histidine residue, except that I added a copper atom bonded to ND1. It is a good idea to read these files alongside the PREP documentation.

Then, these files were used as input to PREP:

```
prep -i hicu_all.in -o hicu_all.prpout
prep -i met.in -o met.prpout
```

These input and output files may be found in the %%% directory.

Step 2: Do some editing of the PDB file.

Several small changes need to be made to the input PDB file to make it work with Amber:

(1) First, we need to split of the HOH water residues into a separate file, say *watpdb*, and remove them from the main PDB file (call this modified file *1plc.nowat.pdb*). Further, the remarks in this pdb file indicate that waters #183 and #187 are a disordered pair, and should not both be present. So, I arbitrarily choose to delete #187 and to keep #183. [Note that AMBER by default will also choose only the principal position for disordered side chains, *i.e.* the "A" conformation if there is more than one. But this is done automatically, and does not require editing. If you want to start a simulation from the "B" conformation of a side chain, you need to manually edit the PDB file to remove the "A" conformation and blank-out the alternate conformation flag for the atoms you want AMBER to use.] For some reason, these two disordered waters were both put in the PDB file, and not assigned as alternate conformers. Generally speaking, you have to look carefully at a Brookhaven file before really using it.
(An alternative is to simply strip out the "crystallographic" waters and not use them at all. This is most appropriate if you are planning an MD or free energy simulation that will go through an extensive equilibration period before the "real" simulation begins. One goal of equilibration is to minimize dependence upon the starting conditions, and certainly the individual water molecules will move around a lot during any decent equilibration. At that point, the fact that you went to some trouble to originally place the waters in some nice positions may be irrelevant. Or maybe not; opinions differ on this matter, which is why we try to provide flexible tools in Amber.)

(2) Next, we need to work on the proton names in the main protein file. Most Brookhaven crystallographic files do not have protons, so the *protonate* program is used to add them. Even here, we want to change the names Brookhaven uses to NMR conventions as described above, so we will still use *protonate*. This program also does sanity and chirality checking, so it is generally a good idea to use it prior to putting any pdb file into Amber.
At this point, you need to decide about protonation states of various residues, especially histidines. This is pretty easy for plastocyanin, since the two histidine side chains are both bound to copper through the ND1 nitrogen. So we initially change both HIS residues to HIE, in order to tell AMBER to put the protons on the NE2 nitrogen. (Note that in many other proteins, it will often be reasonable to assign surface-accessible histidines to be protonated, residue name HIP.) Now run:

```
( protonate -k < 1plc.nowat.pdb > 1plc.nowat.H.pdb) >& protonate.out
```

The *-k* option changes the names but "keeps" the positions of the protons in the input pdb file.

(3) Next, I moved the copper ATOM card from the end of the pdb file into residue 37, changing its residue name to "HIC" and its residue number to 37. I also changed the residue name for the rest of atoms of residue 37 from "HIS" to "HIC", and changed the residue name for residue 92

from "MET" to "MEM" as described above.

Step 3: Create AMBER files for the basic protein.

AMBER gets the sequence information, plus information about how the copper ion is bound to its ligands, from the input files to LINK:

lnkin.nowat

```
PLASTOCYANIN

HIC      2hicu_all.db4
MEM      2met.db4

DU
    0    0    0    0    0
Plastocyanin (poplar)
P   1    0    1    3    1
ILE 2ASP  VAL  LEU  LEU  GLY  ALA  ASP  ASP  GLY  SER  LEU  ALA  PHE  VAL  PRO
SER  GLU  PHE  SER  ILE  SER  PRO  GLY  GLU  LYS  ILE  VAL  PHE  LYS  ASN  ASN
ALA  GLY  PHE  PRO  HIC  ASN  ILE  VAL  PHE  ASP  GLU  ASP  SER  ILE  PRO  SER
GLY  VAL  ASP  ALA  SER  LYS  ILE  SER  MET  SER  GLU  GLU  ASP  LEU  LEU  ASN
ALA  LYS  GLY  GLU  THR  PHE  GLU  VAL  ALA  LEU  SER  ASN  LYS  GLY  GLU  TYR
SER  PHE  TYR  CYX  SER  PRO  HIE  GLN  GLY  ALA  GLY  MEM  VAL  GLY  LYS  VAL
THR  VAL  ASN

   37   87CU  ND1      0
   37   84CU  SG       0
   37   92CU  SD       0

QUIT
```

Again, it is a good idea to compare this input to the descriptions in the manual. Note the the copper atom is already bonded to the ND1 atom of HIS37, and that crosslink commands to used to add three other ligands to it. The "***" in the waters is a special residue name used to indicate that no chemical bond between residues is to be added. Then run:

```
link -i lnkin.nowat -o lnkout.nowat -p /home/case/dat/db94.dat
```

where you must substitute the location of *db94.dat* on your machine for the file listed above. Next create a standard input for for edit:

edtin.nowat

```
poplar plastocyanin
    0    0    0    0
XYZ
OMIT
XRAY
    0    0    0    0    0
QUIT
```

and run:

```
edit -i edtin.nowat -o edtout.nowat -pi 1plc.nowat.H.pdb
```

and look carefully at the output file. It is very common to find warning messages at this point, and they need to be cleared up, usually by minor re-editing of the input PDB file. Finally, create a standard input file for *parm*:

prmin

```
  standard parm using parm94.dat
BIN FOR MOD4
    0    0    0
    1
```

We also need to make modifications to the AMBER force field to recognize the copper atom and the modified methionine residue. The best way to do this is not to edit the main force field file, but to create a *frcmod* file with changes specific to each project. Here is the one I created for plastocyanin:

frcmod.pcy

```
#
#  this file has force constants needed for plastocyanin
#     refinements, in which methionine S is type SM
#
MASS
SM 32.06

BOND
NB-CU  70.000     2.05000       kludge by JRS
CU-S   70.000     2.10000       kludge by JRS
CU-SM  70.000     2.90000       for pcy
CT-SM       222.000     1.81000       met(aa)
LP-SM       600.000     0.67900

ANGLE
CU-NB-CV   50.000     126.700    JRS estimate
CU-NB-CR   50.000     126.700    JRS estimate
CU-NB-CP   50.000     126.700    JRS estimate
CU-NB-CC   50.000     126.700    JRS estimate
CU-SM-CT   50.000     120.000    JRS estimate
CU-S -CT   50.000     120.000    JRS estimate
CU-S -C2   50.000     120.000    JRS estimate
CU-S -C3   50.000     120.000    JRS estimate
CU-SM-LP   00.000      96.700    dac estimate
CU-S -LP   00.000      96.700    dac estimate
NB-CU-NB   10.000     110.000    dac estimate
NB-CU-SM   10.000     110.000    dac estimate
NB-CU-S    10.000     110.000    dac estimate
SM-CU-S    10.000     110.000    dac estimate
CU-SM-CT   50.000     120.000    JRS estimate
CT-CT-SM   50.000     114.700     met(aa)
HC-CT-SM   35.000     109.500
H1-CT-SM   35.000     109.500
CT-SM-CT   62.000      98.900    MET(OL)
CT-SM-LP  600.000      96.700
LP-SM-LP  600.000     160.000

DIHE
X -NB-CU-X    1       0.000     180.000       3.000
X -CU-SM-X    1       0.000     180.000       3.000
X -CU-S -X    1       0.000     180.000       3.000
X -CT-SM-X    3       1.000       0.000       3.000

NONBON
  CU       2.20      0.200
  SM       2.00      0.200
```

Crating a *frcmod* file is a bit of an art, since one is often faced with unknown parameters (such as force constants from copper to its ligands in the above example.) In this tutorial, I am just going over the mechanics of running AMBER, and make no claims about the validity or appropriateness of the above parameters. There is a big literature on parameter estimation, and users are encouraged to consult this when faced with unusal chemical environments.

Now, run *parm* with the above inputs:

```
parm -i prmin -o prmout.nowat -f /home/case/dat/parm94.dat
    -f frcmod.pcy
```

You might also create a pdb file at this point with the new coordinates:

```
ambpdb -wrap < prmcrd > 1plc.nowat.amber.pdb
```

The *-wrap* flag will make the output proton names more like those Brookhaven uses. Leave this flag off if you want the names in the output PDB file to be the internal AMBER proton names.

Step 4: Work on positioning counterions and crystallographic waters

The question of how or whether to include solvent counterions in protein and DNA simulations is a difficult one. Generally speaking, DNA simulations have often used counterions and many existing protein simulations have not. In terms just of "mechanics" and not science, AMBER will suggest counterion positions for you, by using the *cion* program:

```
( cion -elstat < 1plc.nowat.amber.pdb > cion.pdb) >& cion.out
```

This routine places counterions at the most favorable electrostatic positions, until it achieves a neutral overall system. Note, however, that this may end up corresponding to a fairly high salt concentration, and may not be at all what you want. At this stage, the user's judgement is required, which is why a lot of this stuff is not yet automated. This tutorial can't go over all of the pros and cons of various choices, and in any event, different users will have different preferences. Let's suppose that you choose a few chloride counterions to add to the simulation. The positions that *cion* suggests would get added to the bottom of the *1plc.nowat.amber.pdb* file we created above. Then, you need to run LINK again, using an input something like the following:

input to link with counterions

```
PLASTOCYANIN

HIC      2hicu_all.db4
MEM      2met.db4

DU
    0    0    0    0    0
Plastocyanin (poplar)
P   1    0    1    3    1
ILE 2ASP  VAL  LEU  LEU  GLY  ALA  ASP  ASP  GLY  SER  LEU  ALA  PHE  VAL  PRO
SER  GLU  PHE  SER  ILE  SER  PRO  GLY  GLU  LYS  ILE  VAL  PHE  LYS  ASN  ASN
ALA  GLY  PHE  PRO  HIC  ASN  ILE  VAL  PHE  ASP  GLU  ASP  SER  ILE  PRO  SER
GLY  VAL  ASP  ALA  SER  LYS  ILE  SER  MET  SER  GLU  GLU  ASP  LEU  LEU  ASN
ALA  LYS  GLY  GLU  THR  PHE  GLU  VAL  ALA  LEU  SER  ASN  LYS  GLY  GLU  TYR
SER  PHE  TYR  CYX  SER  PRO  HIE  GLN  GLY  ALA  GLY  MEM  VAL  GLY  LYS  VAL
THR  VAL  ASN

   37   87CU  ND1      0
   37   84CU  SG       0
   37   92CU  SD       0

A few chloride counterions
P   0    0    1    3    0
CL  2***  CL   ***  CL   ***  CL   ***  CL   ***  CL   ***  CL   ***  CL   ***
CL  2

QUIT
```

Note the use of the "***" residue to indicate that the counterions are not chemically bonded to each other.

Step 5: Run EDIT and PARM to add a box of waters around the system.

Actually, the hard part is mostly done. At this point we might run edit to read in the PDB file that has counterions, and to add a box of water molecules around the solute. Here is a sample input file for that:

Input to edit to add a box of waters

```
poplar plastocyanin
    0    0    0    0
XYZ
OMIT
XRAY
    0    0    0    0    0
BOX
HW   OW        4
  0.417      2.8        2.3
  14.0       14.0         14.0
QUIT
```

This creates a pretty big box, with a minimum of 14 Å between the protein and the edge of the box; most simulations would use a smaller value to save on computer time. With counterions, though, you need to be sure that there is enough room for them to move around if they need to.

Running PARM with counterions and water is no different than without, so at this point you need to repeat the PARM step outlined above. If everything went well, you will have a parameter file (default name is *prmtop*), and a coordinate file (default name *prmcrd*). These would be used to start an equilibration procedure, followed by an MD or free energy simulation. Future revisions to this tutorial will outline some of the details of that step.

A word about FORTRAN....

Amber is written entirely in FORTRAN. It is not absolutely necessary that one know FORTRAN to run AMBER, but it is important to know a little bit about fortran format statements in order to understand the documentation and prepare your input files correctly. Each line (often called a "card") of input has a FORMAT statement associated with it. The format statement will contain a number of terms separated by commas. Each term represents a value that is being read, and dictates where on the line the characters that comprise that value should lie, and how they will be interpreted. The following terms will be seen:

NAn Alphabetic format. N times n characters will be read as a literal string. If N is missing it is equal to one. 'A' alone means as many characters as the variable being read will hold. An important thing to remember is that character data should be left justified. If a residue name is being read in A4 format, it should appear on the input as GLY(sp) and NOT as (sp)GLY. As a general rule, all character input in AMBER should be upper case (except in titles).

NIn Integer format. N Integers (no decimal pt) occurring in fields n characters wide will be read. These numbers must be RIGHT justified. In some BUT NOT ALL cases a blank will be interpreted as a zero when read in integer format. To be on the safe side, always include zeroes in your input explicitly.

NFm.n
Floating point (real) format. N real numbers (with decimal pt) will be read in fields m characters wide, with n of those characters allocated to the part of the number to the right of the decimal point. It is only necessary to have these numbers within the boundaries of the m character width field. They do not need to be justified.

NX Spaces. N characters of any kind may be present. They will be ignored.

The following example may be helpful:

```
FORMAT(I5,2A4,4X,3F8.3)

    0ALA GLY     12.345  12.345  12.345
IIIIIaaaaAAAAXXXXFFFFFFFFffffffffFFFFFFFF
12345678901234567890123456789012345678901
```

Note that the integer field is right justified, and the two alpha fields are left justified in uppercase.

You will occasionally see constructs such as the following in the documentation:

```
(KVAR(I), I = 1, NVAR)

 FORMAT(16I5)
```

This means that an array (KVAR(1), KVAR(2), ... KVAR(NVAR)) is being read in I5 (or any other) format. You can have up to but not more than 16 values on a line, in whatever format is specified. It is not necessary to use up all the fields in the format statement if less than that number of variables is being read. If NVAR is greater than 16, the input is continued on the next line in the same format.

Some of the routines allow the use of NAMELIST-type input. A description of this more intuitive input scheme is given later and in an Appendix.

RESP

Usage:

```
resp [-O] -i input -o output -p punch -q qin -t qout
                -e espot -w qwts -s esout
```

–O Overwrite output files if they exist.

RESP (Restrained ElectroStatic Potential) fits the quantum mechanically calculated electrostatic potential at molecular surfaces using an atom-centered point charge model. This method was developed by Christopher Bayly. [6] [7] See Appendix D for background on charge fitting.

file name	flag	fortran unit		purpose
input	-i	5	required	input options
output	-o	6	a/p	output of results
punch	-p	7	a/p	synopsis of results
qin	-q	3	optional	replacement charges
qout	-t	19	a/p	ouput of current charges
espot	-e	10	required	input of ESP's and coordinates
qwts	-w	4	optional	input of new weight factors
esout	-s	20	optional	generated esp values for new charges

a/p = always produced

Input included in the "-i" file

-1st line-

TITLE

input: a character string

[6] "A Well-Behaved Electrostatic Potential Based Method Using Charge Restraints For Determining Atom-Centered Charges: The RESP Model," C.I. Bayly, P. Cieplak, W.D. Cornell, and P.A. Kollman, *J. Phys. Chem.* **1993,** *97,* 10269.

[7] "Application of RESP Charges to Calculate Conformational Energies, Hydrogen Bond Energies, and Free Energies of Solvation," W.D. Cornell, P. Cieplak, C.I. Bayly, and P.A. Kollman, *J. Am. Chem. Soc.* **1993,** *115,* 9620.

```
-2nd section-

    Begin with namelist " &cntrl"
    (see example at end)

       inopt   =  0  normal run
               =  1  cycle through a list of different qwt
                        read from -w unit

      ioutopt  =  0  normal run
               =  1  write restart info of new esp etc to
                     unit -es (esout unit)

        iqopt  =  1  reset all initial charges to zero (default)
               =  2  read in new initial charges from -q (qwt)

        (normally not used:

               =  3  read in new initial charges from  -q (qwt)
                     and perform averaging of those new
                     initial charges according to ivary values
        )

         nmol  =  n  the number of molecules in a multiple molecule
                     fit (default 1)

       ihfree  =  0  all atoms are restrained
               =  1  hydrogens not restrained (default)

      irstrnt  =  0  harmonic restraints (old style)
               =  1  hyperbolic restraint to charge of zero (default)
               =  2  only analysis of input charges; no
                     charge fitting is carried out

       iunits  =  0  atom coordinates in angstroms (default)
               =  1   "     "          "  bohrs


         qwt   =  normally use 0.0005 for Stage 1 (default)
                     "      "  0.001  for Stage 2

    end namelist " &cntrl" with " &end"

-3rd "line"-

    wtmol .... relative weight for the molecule if
                multiple molecule fit (1.0 otherwise)

     input: real number
```

```
-4th "line"-

    subtitle for molecule

     input: a character string

-5th "line"-

    charge, iuniq ( the number of atoms)

     input: 2I5

-6th "area"-

     one line for each atom

     input: 2I5

     Atomic number, ivary

            ivary
                  =  0  charge varied independently of previous
                        centers
                  =  n  current charge fitted together with
                        center "n"
                  = -99 charge frozen at "initial charge" value
                        typically read in unit "qin"

-7th- "area"

    charge constraints...  blank line if no constraints


    input:   I5,F10.5

    ngrp = number of centers in the group associated with this
           constraint (i.e. the number of centers to be read in)

    grpchg(i) = charge to which the associated group of atoms
                (given on the next card) is to be constrained


-7.1th-

    imol, iatom

    format(16I5)

      the list (ngrp long) of the atom indices of those atoms to be
```

```
        constrained to the charge specified on the previous line.

       *blank to end

  -8th "area"-

      intermolecular charge constraints
      same format as indvidual molecule constraints

       *blank to end

  -9th "area"-

      Multiple molecule constraints
      same format as -7,8-

       *blank to end
```

```
Other file formats

   -q    input of replacement charges if requested
         (note: same format as produced by -q)

   input: 8f10.6
```

```
   -w  input of new weight factors if requested

     input: i5     nqwt  number of new weights to cycle thru

     input: f10.5  new weights
                   nqwt lines
```

```
  -e   input of ESP's and coordinates

    -1st line-

       n_atoms n_esp_points

       input: 2i5
```

```
    -2nd- 2->natoms+1 lines-

       atom coordinates
       x,y,z (in Bohrs)

       input 17x,3e16.7


    -3rd natoms+2->natoms+2+nesp lines-

       potential and coordinate
       qpot,x,y,z  (in a.u.,bohrs)

       input 1X,4E16.7
```

```
       sample input for a single water molecule force H's equivalent:
       NOTE: the blanks before &cntrl and &end are necessary

test for water
 &cntrl nmol=1
 &end
    1.0
water
    0    3
    8    0
    1    0
    1    2
```

```
       sample input for methanol two configurations;
       hold methyls equal:


test for meoh
 &cntrl nmol=2
 &end
    1.0
m3oh trans
    0    6
    6    0
    8    0
    1    0
    1    0
```

```
    1     3
    1     3

    1.0
meoh cis
    0     6
    6     0
    8     0
    1     0
    1     0
    1     0
    1     0


    2
    1     1     2     1
    8
    1     3     1     5     1     6     2     3     2     5     2     6
```

PREP

Usage:

```
prep [-O] -i input -o output -p params
```

–O Overwrite output files if they exist.

New to 4.1: atom types "4", "5" and "6" to facilitate setting up special residues for GIBBS.

The purpose of this module is to add new residues to the standard AMBER residue database, create new databases, or to create new residues as individual LINK-readable files. It is not necessary to run PREP if all residues needed for a simulation are already present in the standard AMBER database, described in the LINK documentation. A residue is the basic molecular unit of the AMBER simulation package. It is typically an amino acid or nucleic acid unit, but could be a prosthetic group, a small molecule, or a single ion.

Tree Structure: The geometry of the residue is described by a "tree" structure to enable the LINK module to successfully connect it to a larger structure. The atoms in a residue are classified into five topological types: "Main", "Side", "Branch", "3", "4", "5" "6" and "End" types. They are denoted as M, S, B, 3 4 5 6 and E respectively.

Main atoms describe the principal "path" through the residue, starting at the connection to the previous residue and ending at the connection to the next residue. The LINK module will connect the last main atom of a residue to the first main atom of the next residue in the molecule. If there is only one residue in a molecule, the main atoms are typically the longest continuous non-intersecting chain. The main type atoms can have 1, 2, 3, or 4 atoms connected to them.

Any atom that is not a main atom is described by one of the other topological types: "E", "S", "B", "3", "4", "5", or "6". An "E" atom has only one connection to other atoms, thus is a "dead end" for any branch from any other atom type. An "S" atom must have a total of two connections to other atoms, a "B" atom must have a total of three connections, and a "3" atom actually has a total of four connections; the same applies for "4" "5" and "6". The topological types described here can only describe acyclic systems. In order to describe the topology of cyclic systems, explicit loop closing bonds are specified using the LOOP command described below. Loop closing bonds are not counted as connections when assigning M, E, S, B, 3, 4, 5, 6 topological types. If an atom has more than four connections, it is not defined in the present tree structure.

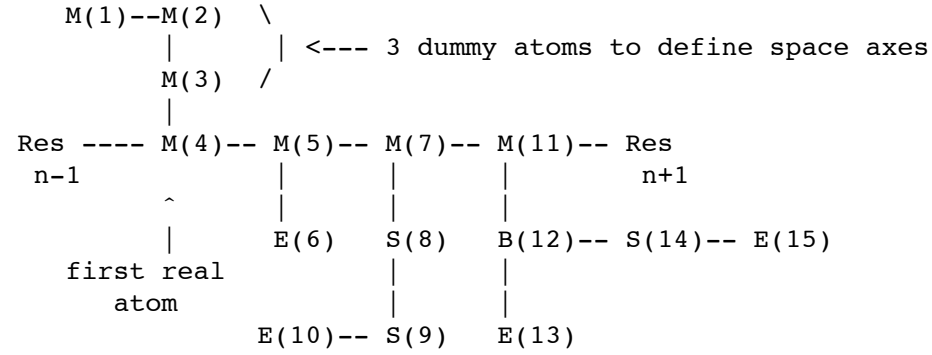
Dummy atoms: PREP requires that three dummy atoms precede the actual atoms of the residue. These atoms are simply used to define the space axes for the residue. The three dummy atoms must be given the topological type "M", and they must be assigned a force field atom type that defines them as dummy atoms. The symbol "DU" is recommended to be consistent with the standard database. It is necessary to have the three initial dummy atoms whether internal or cartesian coordinates are given as input.

It is important for the proper functioning of the EDIT module that dummy atoms be left in the first residue of the system, but that they be removed in any subsequent residues. Therefore you

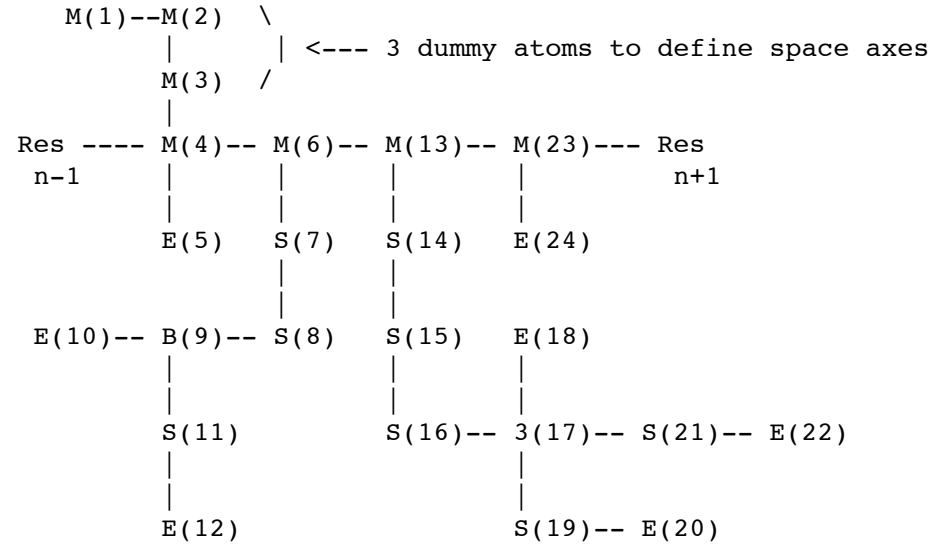
should specify the "NOMIT" flag for any initial residue, and the "OMIT" flag for all others. In typical use of AMBER, peptide systems are either started with the acetyl residue ACE, which carries the dummy atoms with it in the standard database, or they are started with a charged N-terminal residue, which also has dummy atoms. Likewise, nucleic acid systems are generally started with the HB residue, which also has dummy atoms. The other residues in the database have had their dummy atoms stripped at the end of PREP through the use of the "NOMIT" option.

In the examples below, topological types are assigned and the atoms are numbered in correct tree structure order. An actual PREP input file appears at the end of this document.

Example 1:



Example 2:



Note on Tree Ordering: The tree structure begins at the first dummy atom, and traverses the main chain until a branch point (node) is reached. That branch is traversed until its end or until the next node is reached. When you come to a node with more than one branch (topological type "B" or "3"), it doesn't matter which branch is traversed first as long as you return to the next higher node when an end is reached.

PREP input files for standard peptide and nucleic acid residues are typically maintained in several large files for generation of the standard database for the LINK module. Note that it is not necessary to run the PREP module unless non-standard residues are needed. Non-standard residue data may be output as individual files or appended to the standard database if desired.

The LINK module is currently dimensioned to handle a maximum of 150 atoms per residue.

Note that smaller, neutral residues are most appropriate unless an infinite cutoff is desired, because the first atoms in each residue are used in applying the cutoff. The larger the residue, the more unbalanced the cutoff, i.e. the greater difference between head-to-head and tail-to-tail orientations.

This module was originally written by P. K. Weiner at UCSF and overhauled by U. C. Singh in 1984. The data base structure was completely modified. Prep was revised for Rev A by George Seibel in 1989.

Input description: This section describes the residue(s) input file which is read through unit 5. The input is free format and it is assumed that the different fields are separated by at least one space (including character fields). The character variables are always left justified. If a character field contains more than four characters the rest are ignored. If it contains less extra blanks are added to it. Since blanks are separators between fields signs have to immediately precede numbers.

```
- 1 -       CONTROL FOR DATA BASE GENERATION

      The data base is a direct access file containing the
      standard residues and a directory of their names.  It
      is named DB4.DAT in the version 4 AMBER distribution,
      and is found in the DAT directory.  The LINK module
      will search this file for a residue before searching
      the external files for it.  The LINK module can only
      access one database per run.  Thus if any user supplied
      residues are needed, they can be accessed by LINK as
      individual files.  The data base can also be appended
      with user supplied residues if desired.

      IDBGEN , IREST , ITYPF

          FORMAT(3I)

IDBGEN      Flag for data base generation
 = 0  No database generation.  Output will be individual files.
      This is the standard procedure if you want to create a
      single small molecule.
 = 1  A new data base will be generated or the existing database
      will be appended.

IREST       Flag for the type of generation (assuming IDBGEN = 1)
 = 0  New data base
 = 1  Appending an existing data base

ITYPF       Force field type code (used in LINK stage)
      Ignored if IDBGEN = 0   The following codes are used in
      the standard database:
```

```
    = 1  United atom model
    = 2  All atom model
  = 100  United atom charged N-terminal amino acid residues
  = 101  United atom charged C-terminal amino acid residues
  = 200  All atom charged N-terminal amino acid residues
  = 201  All atom charged C-terminal amino acid residues

         Note:  This variable allows you to have several different
         models for the same residue name stored in one database.
         These models could differ in topology, charge, or other
         factors.  The charged terminal residues are selected
         internally by LINK if the IFTPRO flag is set.  The
         database can hold up to 510 residues.
```

```
   - 2 -       NAMDBF

              FORMAT(A80)

   NAMDBF      Name of the data base file (maximum 80 characters)
         if NOT data base generation leave a BLANK CARD
```

```
   - 3 -      TITLE

              FORMAT(20A4)

   TITLE       Descriptive header for the residue
```

```
   - 4 -      NAMF

             FORMAT(A80)

   NAMF       Name of the output file if an individual residue file is
        being generated.  If database is being generated or
        appended this card IS read but ignored.
```

```
   - 5 -      NAMRES , INTX , KFORM

             FORMAT(2A,I)

   NAMRES     A unique name for the residue of maximum 4 characters

   INTX       Flag for the type of coordinates to be saved for the
        LINK module
 'INT'  internal coordinates will be output (preferable)
 'XYZ'  cartesian coordinates will be output
```

```
   KFORM      Format of output for individual residue files
   = 0  formatted output (recommended for debugging)
   = 1  binary output
```

```
   - 6 -      IFIXC , IOMIT , ISYMDU , IPOS

              FORMAT(4A)

   IFIXC      Flag for the type of input geometry of the residue(s)

 'CORRECT' The geometry is input as internal coordinates with
           correct order according to the tree structure.
           NOTE: the tree structure types ('M', 'S', etc) and order
           must be defined correctly: NA(I), NB(I), and NC(I) on card
           8 are always ignored.
 'CHANGE'  It is input as cartesian coordinates or part cartesian
           and part internal.  Cartesians should precede internals
           to ensure that the resulting coordinates are correct.
           Coordinates need not be in correct order, since each
           is labeled with its atom number. NOTE: NA(I), NB(I), and
           NC(I) on card 8 must be omitted for cartesian coordinates
           with this option.

   IOMIT      Flag for the omission of dummy atoms

 'OMIT'    dummy atoms will be deleted after generating all the
           information (this is used for all but the first residue
           in the system)
 'NOMIT'   they will not be deleted (dummy atoms are retained for
           the first residue of the system.  others are omitted)

   ISYMDU     Symbol for the dummy atoms.  The symbol must be
        be unique.  It is preferable to use 'DU' for it

   IPOS       Flag for the position of dummy atoms to be deleted

 'ALL'     all the dummy atoms will be deleted
 'BEG'     only the beginning dummy atoms will be deleted
```

```
   - 7 -      CUT

              FORMAT(F)

   CUT        The cutoff distance for loop closing bonds which
         cannot be defined by the tree structure.  Any pair of
         atoms within this distance is assumed to be bonded.
         We recommend that CUT be set to 0.0 and explicit loop
         closing bonds be defined below.
```

```
- 8 -      I , IGRAPH(I) , ISYMBL(I) , ITREE(I) , NA(I) , NB(I) ,
     NC(I) , R(I) , THETA(I) , PHI(I) , CHG(I) , I = 1, NATOM

         FORMAT(I,3A,3I,4F)

I          The actual number of the atom in the tree.
     If IFIXC .eq. 'CHANGE' then this number is important
     since the corresponding coordinates are stored at that
     location.  If IFIXC .eq. 'CORRECT' then atoms are in
     the correct order according to the tree structure.

NOTE:  PREP always expects three dummy atoms for the beginning.

IGRAPH(I)  A unique atom name for the atom I. If coordinates are
     read in at the EDIT stage, this name will be used for
     matching atoms.  Maximum 4 characters.

ISYMBL(I)  A symbol for the atom I which defines its force field
     atom type and is used in the module PARM for assigning
     the force field parameters.

ITREE(I)   The topological type (tree symbol) for atom I
     (M, S, B, E, or 3)

NA(I)      The atom number to which atom I is connected.
     Read but ignored for internal coordinates; If cartesian
     coordinates are used, this must be omitted.

NB(I)      The atom number to which atom I makes an angle along
     with NA(I).
     Read but ignored for internal coordinates; If cartesian
     coordinates are used, this must be omitted.

NC(I)      The atom number to which atom I makes a dihedral along
     with NA(I) and NB(I).
     Read but ignored for internal coordinates; If cartesian
     coordinates are used, this must be omitted.

R(I)       If IFIXC .eq. 'CORRECT' then this is the bond length
     between atoms I and NA(I)
     If IFIXC .eq. 'CHANGE' then this is the X coordinate
     of atom I

THETA(I)   If IFIXC .eq. 'CORRECT' then it is the bond angle
     between atom NB(I), NA(I) and I
     If IFIXC .eq. 'CHANGE' then it is the Y coordinate of
     atom I

PHI(I)     If IFIXC .eq. 'CORRECT' then it is the dihedral angle
     between NC(I), NB(I), NA(I) and I
     If IFIXC .eq. 'CHANGE' then it is the Z coordinate of
     atom I

CHRG(I)    The partial atomic charge on atom I
```

This section is terminated by one BLANK CARD if IFIXC = 'CORRECT'.
This section is terminated by TWO BLANK CARDS if IFIXC = 'CHANGE'.

- 9 - IOPR

FORMAT(A4)

IOPR Flag to read additional information about the residue. There are four options available. The order in which they are specified is not important. Format is keyword on its own line, followed by data on succeeding lines, terminated by a BLANK CARD.

'CHARGE' Control to read additional partial atomic charges. These will override charges specified above in section 8. The charges are read in format(5F) for the non-dummy atoms. A BLANK CARD terminates this section. It is less error-prone to specify charges as in section 8.

'LOOP' Control to read explicit loop closing bonds (in addition to the loops generated based on the cutoff criterion). If this option is used it is preferable to set the cutoff criterion to zero. The loop closing atoms are read in format(2A) as their atom (IGRAPH) names. A BLANK CARD terminates this section.

'IMPROPER' Control for reading the improper torsion angles. A proper torsion I - J - K - L has I bonded to J bonded to K bonded to L. An IMPROPER torsion is any torsion in which this is not the case. Improper torsions are used to keep the asymmetric centers from racemizing in the united atom model where all the C-H hydrogens are omitted. They can also be used to enforce planarity. The improper torsions should be defined between 4 atoms in such a way that the proper torsions are not duplicated. The atoms making the improper torsions are read as their atom (IGRAPH) names. A BLANK CARD terminates this section.

'DONE' Control to exit from this section.

NOTE: If extra blank cards are found between different options they are ignored. Control will exit only when the 'DONE' option is found. If it is desired to process another residue place the appropriate information after the 'DONE' card.

-10 - KSTOP

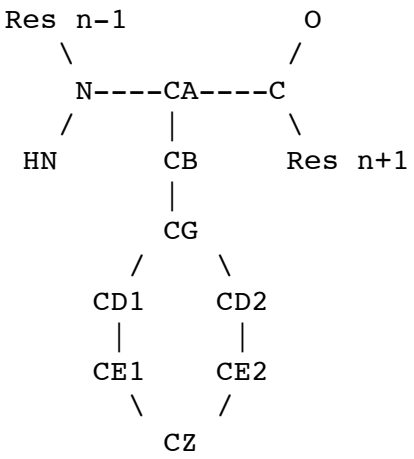
FORMAT(A4)

KSTOP Control to exit from the program

'STOP' Exit from the program. It has to be placed immediately following the 'DONE' card.

The program can never make a graceful exit if this card is missing since it is working inside an infinite loop.

Example input for phenylalanine (united atom)




```
    0    0    1
<blank line>
     PHENYLALANINE PREP INPUT EXAMPLE (title)
PHE
PHE  INT    1
CORRECT  OMIT DU   BEG
  0.0
    1 DUMM   DU    M    0   -1   -2   0.0000    0.0000    0.0000  0.000
    2 DUMM   DU    M    1    0   -1   1.4490    0.0000    0.0000  0.000
    3 DUMM   DU    M    2    1    0   1.5220  111.1000    0.0000  0.000
    4 N      N     M    3    2    1   1.3350  116.6000  180.0000 -0.5200
    5 HN     H     E    4    3    2   1.0100  119.8000    0.0000  0.2480
    6 CA     CH    M    4    3    2   1.4490  121.9000  180.0000  0.2140
    7 CB     C2    S    6    4    3   1.5250  111.1000   60.0000  0.0380
    8 CG     CA    S    7    6    4   1.5100  115.0000  180.0000  0.0110
    9 CD1    CD    S    8    7    6   1.4000  120.0000  180.0000 -0.0110
   10 CE1    CD    S    9    8    7   1.4000  120.0000  180.0000  0.0040
   11 CZ     CD    S   10    9    8   1.4000  120.0000    0.0000 -0.0030
   12 CE2    CD    S   11   10    9   1.4000  120.0000    0.0000  0.0040
   13 CD2    CD    E   12   11   10   1.4000  120.0000    0.0000 -0.0110
   14 C      C     M    6    4    3   1.5220  111.1000  180.0000  0.5260
   15 O      O     E   14    6    4   1.2290  120.5000    0.0000 -0.5000

IMPROPER
-M  CA  N   HN
CA  +M  C   O
CB  CA  N   C
```

```
    LOOP
    CG  CD2

    DONE
    STOP
```

Rev A Revision: George Seibel
PREP Authors: P. K. Weiner and U. C. Singh
Director: P.A. Kollman
Department of Pharmaceutical Chemistry
School of Pharmacy
University of California
San Francisco CA 94143
Phone (415) 476 4637

LINK

Usage:

```
link [-O] -i input -o output -l lnkbin -p db4.dat
```

–O Overwrite output files if they exist.

The purpose of this module is to create the molecular topology file read by EDIT. It reads the topology of individual residues from one of the standard databases and/or from individual files, and links them together to create the topology of the final system. It uses the tree convention of the PREP module and always connects the first main type atom of the current residue to the last main type atom of the previous residue. In addition to this standard linking process, it can cross link specified atoms in a molecule or different molecules to form a covalent bond. The macromolecule can be specified as a single molecule or set of molecules. For example, double stranded DNA is normally defined as two molecules, where each strand constitutes a molecule. Separate molecules are not linked by a covalent bond unless explicitly specified by the cross linking information.

The standard database contains topological information for nucleic acid and peptide residues. If non-standard residues are required, the PREP module must first be used to create this nonstandard residue. It may be appended to the database or kept as an individual file. See PREP.DOC for details.

It is important to note that this module will generate the connectivity and the internal parameter lists such as bond angle, dihedral and excluded atom pointers correctly. However, the coordinates are generally not meaningful except for smaller molecules because residues are linked with the dihedral angles stored in the data base. Side chain conformation will also be that found in the database entry. Hence the user will generally want to read in a new coordinates at the next stage using the EDIT module and a coordinate file.

This module was originally written by P. K. Weiner at UCSF and overhauled by U. C. Singh in Feb. 1984. LINK 3.0 Rev A is a revision for portability and reliability by George Seibel, 1989.

NOTE: the utility program NUKIT will generate link (and nucgen) input files for nucleic acids interactively.

Files:

file	unit	description
db94.dat	15	Prep database
input	5	Program control input

```
output     6      User information and diagnostics
lnkbin     10     Output topology file (binary)
```

NOTE: LINK has formatted input, so pay careful attention to the fields specified.

Nucleic Acids

The nucleic acid residues have different naming conventions in the 1994 and 1991 databases (`db94.dat` and `db4.dat`, respectively). *Nucgen* can generate PDB files using the 1994 residue names.

1994 Nucleic Acids

strand position	termini 5'	termini 3'	residue names
beginning	OH	O	G5, C5, A5, T5, U5
middle	phosphate	O	G, C, A, T, U
end	phosphate	OH	G3, C3, A3, T3, U3
single residue	OH	OH	GN, CN, AN, TN, UN

1991 Nucleic Acids

In the 1991 database, phosphates and terminal hydrogens are treated as separate residues, so the bases have only one version: GUA, CYT, ADE, THY and URA. The phosphate group is represented by the residue name POM. The terminal hydrogen atoms are represented by residues HB (at the 5' end) and HE (at the 3' end).

At the 5' end of a nucleic acid chain, the atom H in HB is connected to the O5' atom of the first nucleoside. The nucleic acid chain grows in the 5' to 3' direction. At the 3' end the H atom in HE is connected to the O3' atom of the last nucleoside residue. Two nucleoside residues are connected by a phosphate group (POM). The 5' and 3' ends of POM are linked to O3' and O5' atoms of the preceding and following nucleoside residue.

In the case of a double-helix, the complementary chain also is represented from the 5' end to the 3' end. Each chain is described as a separate molecule. Thus, the double helix will be represented by two molecules, the triple helix by three molecules.. etc..

For example, the sequence d(ATATAT).d(ATATAT) is represented by two molecules as:

```
A5E T   A   T   A   T3E   (MOL 1)
A5E T   A   T   A   T3E   (MOL 2)
```

in the 1994 convention, and

```
HB  ADE POM THY POM ADE POM THY POM ADE POM THY HE   (MOL 1)
HB  ADE POM THY POM ADE POM THY POM ADE POM THY HE   (MOL 2)
```

in the 1991 convention. The sequence d(CGATG).d(CATCG) is represented by

```
C5E G   A   T   G3E   (MOL 1)
C5E A   T   C   G3E   (MOL 2)
```
in the 1994 convention and

```
HB  CYT POM GUA POM ADE POM THY POM GUA HE   (MOL 1)
HB  CYT POM ADE POM THY POM CYT POM GUA HE   (MOL 2)
```
in the 1991 convention.

Peptides and Proteins

Proteins are assumed to begin from the N-terminus and end at the C-terminus. There are three ways to handle these termini:

(1) specify only the 'normal' residues in the chain with IFTPRO=0 for uncharged ends:
(N--CA--...--CA--C)
which will leave 'unbalanced' charges

(2) specify only 'normal' residues in the chain with IFTPRO=1 for charged ends:
(NH3+ --CA--...--CA-- COO-)

(3) specify the 'normal' residues *with* terminal residues ACE and NME and IFTPRO=0 for neutral ends:
(ACE--CA--...--CA--NME).

IFTPRO is on card 6B.

The program does not recognize the disulfide S-S bridge in proteins so this must be input as cross links for each bridge (cards 6B and 6E).

name	residue
Alanine	ALA
Arginine	ARG
Asparagine	ASN
Aspartic acid	ASP
Cysteine	CYS
Cystine (S-S bridge)	CYX
Glutamine	GLN
Glutamic acid	GLU
Glycine	GLY
Histidine delta H	HID
Histidine epsilon H	HIE
Histidine +	HIP
Isoleucine	ILE
Leucine	LEU
Lysine	LYS
Methionine	MET
Phenylalanine	PHE
Proline	PRO

```
            Serine                          SER
            Threonine                       THR
            Tryptophan                      TRP
            Tyrosine                        TYR
            Valine                          VAL
            Acetyl group                    ACE  (beginning residue)
            N-Methyl                        NME  (end residue)
```

Input description

The following section describes the input data necessary for this module, which is read from unit 5. IMPORTANT: Character data should be left-justified.

```
------------------------------------------------------------------------

   - 1a -       TITLE FOR THE RUN

             FORMAT(20A4)

   TITLE       Title for identification

------------------------------------------------------------------------

   - 1b -   blank card (read but ignored)

------------------------------------------------------------------------

   - 2 -      This section is used to inform LINK of nonstandard
        residues and where these files can be located.  If no
        nonstandard residues are required one blank card (card 3) is
        still necessary to terminate this section.  If a nonstandard
        residue has the same name as a standard residue found in the
        database, set ITYPEF = 9 in card 6C for the nonstandard
        residue.  This will prevent the standard residue from being
        substituted for the nonstandard residue.  The actual value of
        ITYPEF in the nonstandard residue file doesn't matter.

        IERES(I) , JERES(I) , KERES(I) , I = 1,NERES

            FORMAT(A4,1X,I5,A40)

   IERES(I)   name of the residue  (PREP input card 5)

   JERES(I)   flag for the type of topology file
        (PREP input card 5 (KFORM))
   = 0  formatted file
   = 1  binary file

   KERES(I)   Name of the residue topology file (PREP input card 4)
```

Note: The external source for nonstandard residue(s) is read until a blank card is encountered. As many as 200 external residues can be read.

- 3 - one blank card

- 4 - ISYMDU

FORMAT(A4)

ISYMDU Symbol for the dummy atoms.
It is advisable to use 'DU' as the symbol for dummy atoms as in the data base. The symbol for the dummy has to be unique for a given system. It is not permitted to have more the one dummy atom symbol. Do NOT confuse these dummy atoms with the dummy atoms used in Perturbation.

- 5 - IWO , IWI , IWN , IWA

Output information written to file 'output' (unit 6)

FORMAT(10I5)

IWO Flag to output the coordinates for each atom
= 0 coordinates will be output
= 1 output will be suppressed

IWI Flag to output the residue information for all residues
= 0 none
= 1 output the information

IWN Flag to output non-bonded excluded lists for all atoms
= 0 none
= 1 output

IWA Flag to output bond, angle and dihedral pointers
= 0 none
= 1 output

- 6 - Individual molecule information. The program will continue to read groups of cards described in this section until a 'QUIT' card is encountered. Each group of cards represents one molecule. Repeat cards 6A - 6F for each molecule of the system.

- 6A - SUBTITLE FOR THE MOLECULE TO BE READ

```
            FORMAT(20A4)

   TITLE      Subtitle for the molecule

-----------------------------------------------------------------------

   - 6B -     LBMOL , ICROSL , ICONN , NA0 , NM0 , IFTPRO

            FORMAT(A1,I4,4I5)

   LBMOL      Label for the type of molecule.
        This is necessary since some adjustments of the end
        residues of nucleotides and peptides are made so
        that the charge of the system is an integral value.
        Consult the subroutine BLDIT in the LINK source code
        for details.

   'D'  the molecule is a deoxy nucleotide
   'R'  the molecule is a ribonucleotide
   'P'  the molecule is a peptide or protein
   'O'  the molecule is anything else ('O' = other)

   ICROSL     Flag for the presence of cross links within the
        molecule or between molecules.

   = 0  none
   = 1  cross link is present and its information will be read
        after the residue sequence.

   ICONN      Read but not used.

   NM0        The molecule number to which the first main atom of
        the present molecule is attached either by a covalent or
        non-covalent bond. If there is no covalent attachment,
        NM0 should be set equal to 1, as the first molecule defines
        the space axes for all subsequent molecules.
       *** This is usually set equal to 1 ***

   NA0        The relative atom number in molecule NM0 to which the
        current molecule's first main type atom is connected.
        If there is no covalent connection NA0 should be set
        equal to 3 as the space axes are defined by the first
        three atoms of the first molecule.  If this is not done
        there will be an error in converting from internal
        coordinates to cartesian coordinates in the EDIT module.
       *** This is usually set equal to 3 ***

   IFTPRO     Flag for the type of protein terminal residues

   = 0  Standard (uncharged) terminal residues
   = 1  Charged terminal residues ( NH3+, COO-)

-----------------------------------------------------------------------
```

- 6C - Residue information for the current molecule. It is read in the following format until a blank card is encountered.

LBRES(I) , ITYPF(I) , I = 1, NRESM

FORMAT(16(A4,I1))

LBRES(I) Residue name

ITYPF(I) The type of force field for the current residue.
This option is included so that different types of residues may be kept in the data base. Currently three types are available; the united atom type, the all atom type, and Jorgensen's OPLS model.
Additional models could be put into the data base and retrieved using this option.

= 1 united atom model
= 2 all atom model
= 3 OPLS united atom model (requires use of OPLS force field file)

If this is zero then the previous non zero value is carried over until a non zero option is specified.

NOTE: The program assumes it is done reading the residue information when it encounters a blank card. It is always assumed that the first main type atom of the current residue is connected to the last main type atom of the preceding residue by a covalent bond. If this covalent linkage is not desired the two residues should be separated by the spacer residue, '***'. When two residues are separated by the spacer residue they are connected without a covalent linkage.

For example the sequence ALA 1GLY will be connected by a covalent bond while ALA 1*** GLY will be topologically connected but no bonding parameters will be considered. Both systems in this example use the united atom force field.

- 6D - Blank card to terminate residue input

- 6E - This section is used to create cross linkages within a molecule or between different molecules. A normal use of this is to create disulfide bonds in proteins.

***** ONLY IF ICROSL.GT.0 *****
*** ICROSL is set in card 6B ***

ICROS , JCROS , IACROS , JACROS , MOLNM

FORMAT(2I5,2A4,I5)

```
   ICROS , JCROS

        The residue numbers, as they are listed in 6C, of the two
        residues to be cross linked.  ICROS specifies the relative
        residue number in molecule MOLNM, (MOLNM is assumed to be
        the current molecule number unless otherwise specified at
        the end of this card.), and the residue JCROS is assumed
        to be in the current molecule.

   IACROS     The graph name, the name assigned to the atom in PREP, of
        the atom in residue ICROS which is involved in the cross link.

   JACROS     The graph name of the atom in residue JCROS which is
        involved in the cross link.

   MOLNM      The molecule number to which the residue ICROS belongs.
        If MOLNM = 0 then it is assumed to be in the current
        molecule.

-------------------------------------------------------------------------
   - 6F -     Blank card if ICROSL .GT. 0
-------------------------------------------------------------------------

   - 7 -      KSTOP

            FORMAT(A4)

   KSTOP      Control to exit from the program

        immediately following the last blank card.  If additional
        molecules are to be processed the cards in group 6 are
        repeated before the 'QUIT' card.  The program will never
        make a graceful exit if this card is missing since it
        is working inside an infinite loop.

-------------------------------------------------------------------------

              ++++++ END OF INPUT ++++++
```

Rev A Revision: George Seibel
LINK 3.0 Authors: U.C. Singh and P.K. Weiner
Director: P.A. Kollman
Department of Pharmaceutical Chemistry
School of Pharmacy
University of California
San Francisco CA 94143
Phone (415) 476 4637

EDIT

Usage:

```
edit [-O] -i input -o output -l lnkbin -e edtbin
         [-pi pdbin -po pdbout -a addwat -b boxfil -z zmat]
```

–O Overwrite output files if they exist.

The main purpose of this module is to provide correct coordinates for the LINK generated structure, and to remove dummy atoms. The topology of each residue in the LINK file is correct, but inter-residue dihedral angles and side chain conformations are mostly arbitrary (i.e. a peptide generated simply using LINK will be linear). Thus it is necessary to provide coordinates for some (usually most) of the atoms of the system.

Unlike previous versions of EDIT, this revision will now read a standard Protein Data Bank (PDB) format coordinate file as supplied by Brookhaven National Laboratory. It is no longer necessary to strip out non-'ATOM' records from your PDB files.

For each residue in the LINK topology file, EDIT will attempt to match atoms from the PDB file by atom name on a residue by residue basis. If atoms in the PDB file are missing (e.g. hydrogens from an x-ray data file) relative to the LINK topology file, EDIT will try to generate their coordinates using the internal coordinates from the LINK file. If this is not possible, the missing atoms can be assigned explicitly through the use of the 'ABC' option. Any extra atoms in the PDB file will be ignored and a diagnostic issued.

In addition, various EDIT options allow the user to solvate the molecule (either with cubes of standard "Monte Carlo" water or water from a PDB file), to add counterions to nucleic acids automatically, and to modify the sugar pucker of nucleic acids.

This module was originally written by P. K. Weiner at UCSF and was updated by U. C. Singh 1986. Revision A constitutes a further update by G.L. Seibel, 1989. 4.0/4.1 updates/additions by J. W. Caldwell 1992/1994 and Bill Ross and Thomas Huber 1994.

On VMS systems files are assigned by Fortran unit number. These are given below along with a description of each file. Files shown in [] above are optional, others are mandatory.

```
input      5    Program control input

output     6    User information and diagnostics

lnkbin    10    Input topology file from LINK (binary)

edtbin    12    Output topology file to be read by PARM
                (binary)
```

```
          pdbin     15    PDB input (X-ray or model-built)
                          coordinates.  See the "XRAY" option.

          pdbout    18    PDB output. See the "XRAY" option.

          addwat    25    Water molecules in PDB format.  Used for
                          input of crystallographic waters.  See the
                          "ADD" option.

          boxfil    35    Monte Carlo water for solvating the system.
                          See the "BOX" ,"SOL" etc options.

          zmat       7    Unit to write the coordinates in a z-matrix

          solv      55    Unit to read the topology for the "general"
                          solvent
```

Input description:

```
   - 1 -      TITLE FOR THE EDIT INPUT

             FORMAT(20A4)

   TITLE      title for identification purposes
```

```
   - 2 -      IWO , IWN , IWA, ILINK

            FORMAT(4I5)

   IWO        Flag to output the coordinates for each atom

   = 0  coordinates will be output to unit 6.
   = 1  output will be suppressed

   IWN        Flag to output non-bonded excluded lists for all atoms

   = 0  none (recommended)
   = 1  output

   IWA        Flag to output bond, angle, and dihedral pointers

   = 0  none (recommended)
   = 1  output

   ILINK      Flag for reading solute from lnkbin

   = 0  normal
   = 1  don't read lnkbin - pure water with no solute
```

EDIT options are invoked by keywords read from unit 5. Each keyword is processed as it is encountered in the file. Many of the options require further input which is read in card image format directly following the keyword. The following section describes each keyword briefly. Any extra input needed by the various options is found in the section after the following.

Note that the sequence of options has to follow certain logic to get meaningful output for other modules. For example, the placement of counterions or use of any of the solvation options should not be attempted until after the xray coordinates have been read. Coordinates assigned by ABC for atoms that could not be matched in XRAY must be put on AFTER the PDB input is read. One should not attempt to write PDB output until all options that add to or change the coordinates have been completed.

- 4 - CONTROL WORDS FOR INVOKING EDIT OPTIONS

IFORM

FORMAT(A4)

IFORM The control word for the option

'XYZ' Option to convert the coordinates from internal to cartesian coordinates. This is necessary because the LINK module builds the macromolecule from the residue topology files, most of which are in internal coordinates. The current minimizer works only in cartesian space, therefore it is necessary to convert the internal coordinates to cartesian coordinates before proceeding further.

'INT' Option to convert the coordinates from cartesian to internal. This option is suitable for internal coordinates minimization (not currently available). After reading the correct coordinates from the external source in cartesian form they can be converted to internal form.

'XRAY' Option to read and/or write the desired coordinates for the system in PDB format. The program reads the external source residue by residue and assigns coordinates for each residue in the topology file from LINK based on matching of atom names. Note that the residues must have the same names and order in the PDB file as in the LINK input for warningless matching to occur. A Brookhaven Protein Data Bank (PDB) 'ATOM' record looks like this:

ATOM 1234 N ILE 116 18.896 65.826 12.793

The fields shown are:
record_type atom# atom_name residue_name residue# x y z

Other fields are used in standard PDB files but not read by EDIT. It is necessary that the atom names in the PDB file be identical to the atom names in the standard database or any user prepared residue files. Any atoms present in the LINK topology file but missing from the PDB file will be added automatically using the stored internal coordinates from the LINK topology file. In the event that the stored internal coordinates are insufficient to assign missing atoms, they may be assigned explicitly through the use of the 'ABC' option. Extra atoms in the PDB file are ignored and a diagnostic is issued.

'ABC' Option to fix the coordinates of specified atoms by means of internal coordinates. This is useful to fix beginning hydrogen atoms for which there is insufficient main chain "TREE" information. This option can also be used to override the default PREP coordinates in the case PDB coordinates are not available.

'PUCKER' Special option to change the sugar puckering values in DNA or RNA. It is useful in forcing some of the sugar puckers to different conformations for minimization.

'CION' Option to place counter ions around any charged groups such as the phosphate residue in DNA with the atom type CI (The option is designed primarily for nucleic acids and may place the counter ions incorrectly for other type of molecules).

'SOL' Option to solvate the system. The solute molecule is placed in a cube of "Monte Carlo water". Water molecules are removed based on input steric and distance criteria. The SOL option can be used to create a spherical "CAP" of water molecules centered on a key part of the solute or can be used to create an asymmetric "BLOB" of water surrounding the entire solute.

'OCT' Option to make any subsequent periodic solvation (e.g. 'BOX') assume truncated octahedral rather than rectilinear shape. In general, this means that fewer waters will be required to solvate to a given distance from the solute. [Information on the box shape is passed through PARM and is used if periodic conditions are specified in dynamics (NTB not equal to 0).]

'BOX' Option to place the solute into a large volume of TIP3P "Monte Carlo water" which is then truncated to a "box" around the solute based on user cutoff criteria. The box is rectilinear by default, or truncated octahedral if the 'OCT' keyword precedes this option. Water molecules are characterized by three bonds rather than two bonds and an

angle. This results in a rigid, TIP3P water model if SHAKE is used on covalent bonds involving Hydrogens during dynamics.

'BX4' Same as 'BOX', except the four-point TIP4P water model is used. Water molecules are characterized by six bonds rather than two bonds and an angle. The 4th point is on the C2v axis between the Oxygen and Hydrogens and carries the negative charge. In dynamics, SHAKE must be used on covalent bonds involving Hydrogens to achieve a rigid TIP4P model.

'WAT' Option to make a box of NCUBE^3 * 216 water molecules. No solute is involved, and ILINK on Line 2 must be set to 1. The box is rectilinear by default, or truncated octahedral if the 'OCT' keyword precedes this option. The water molecules are TIP3P as described in 'BOX', above.

'TP4' Same as 'WAT', but the water molecules are TIP4P as described in 'BX4', above.

'FLO' Same as 'WAT', but the water molecules are 'floppy', i.e. "normal" two bonds and an angle water.

'ADD' Option to add an arbitrary amount of water to the solute for partial hydration. The coordinates of the water molecules can be obtained either from x-ray or from model building and are read in PDB format from unit 25. Atom names must begin with 'O' or 'H' for Oxygen and Hydrogens, respectively. The atoms can be in any order within the residue, but each water must have one Oxygen and two Hydrogens. Atom numbers, residue names, and residue numbers in the file will be ignored. The file must NOT contain any non-ATOM format records, i.e. no TER, HEADER, etc.

'GEN' Option to make a box of solvent in a quasi crystal around a solute. Additional information must be read from files specified by the -s and -b flags. The -s file contains the PREP topology data (the output file as specified on line 4 of the prep file) for the new solvent, the -b file contains the pdb coordinates of a representative solvent residue.

'DIHED' Option to calculate bond, angle and dihedral values only for specified atoms.

'TELL' Option to calculate and print the bond, bond angle, and dihedral angle values for ALL atoms.

'GAUSS' Option to punch out the coordinates in a z-matrix.

'QUIT' Option to terminate the run. Necessary!

ADDITIONAL INPUT FOR EACH OPTION -- UNIT 5 --

XYZ OR INT

XYZ/INT 1: IOMIT

FORMAT(A4)

IOMIT Flag to omit the dummy atoms from the system. When the LINK module links different residues into a macromolecule an unspecified number of dummy atoms will be present for defining the coordinate axes. These dummy atoms are essential in the case of internal coordinate minimization (not implemented) but should be removed for minimization in cartesian space.

'OMIT' Dummy atoms are omitted.
' ' Dummy atoms are not omitted.

This flag is maintained for future development of minimization in internal coordinate space.

XRAY

XRAY 1: IOPT , IOPTF , MOLNU , IDBL

FORMAT(4I5)

IOPT Flag for reading or writing the coordinates.

= 0 Read in new coordinates in PDB format for replacing the LINK coordinates. Coords are read from unit 15.
= 1 Read in new coordinates and output the replaced coordinates in PDB format. Units 15 and 18 are used. This is the same as calling XRAY twice, first with IOPT=0, then 2. It is often necessary to use IOPT=0, call other EDIT options, then call XRAY again with IOPT=2 in order to get all the coordinates out.
= 2 Write out the coordinates in PDB format to unit 18.

IOPTF Read but not used.

MOLNU The molecule number for which the coordinates are to be read and replaced.

= 0 All the molecules coordinates are replaced.
> 0 The specific molecule whose coordinates are to be replaced.
NOTE: It is advisable to read in the new coordinates for

all the molecules to avoid an error in generating the coordinates for minimization.

IDBL Flag to write out the coordinates in extended precision.

= 0 Normal PDB format.
= 1 Coordinates are written in 3F10.5 instead of 3F8.3.

CION

CION 1: CMIN , SDIST , DISCK , CUTSB

FORMAT(4F10.3)

CMIN The program sums all the charges in a sphere of radius "SDIST" around the specified atom(s) (see next card) and places a counter ion only if the sum is greater than or equal to CMIN.

SDIST The sphere radius around the specified atom(s) for calculating the charge.

DISCK The cutoff distance between any atom in the molecule and the counter ion to be placed. If the distance is less than or equal to DISCK then a counter ion is not placed.

CUTSB The cutoff distance between two specified atom centers to be assumed to have a salt bridge. If the distance between any two atom centers around which counter ions have to be placed is .le. CUTSB and the counter ions are opposite in sign then a salt bridge is assumed and the counter ions are not placed.

CION 2: JRESN , JATN , CION , DISTI

FORMAT(A4,1X,A4,1X,2F10.2)

JRESN Residue name to which atom "JATN" belongs. All the atoms with this residue name will be taken into account for placing the counter ion.

JATN The atom name around which counter ion is to be placed.

CION The charge on the counter ion.

DISTI The distance between the counter ion and the specified atom.

Note: A maximum of 50 atom centers can be read for placing

counter ions.

CION 3: Blank card to terminate CION input

ABC

ABC 1: MOLNU , IPART , IDIREC

FORMAT(I5,A3,I5)

MOLNU The molecule number in which the following specified atoms are located which need to have coordinates fixed. If MOLNU.le.0 then it is assumed that the atom numbers defined are the absolute numbers. Otherwise, it refers to the relative number with respect to the molecule specified.

IPART Flag for ending this option.

'QUI' Termination of the option.

IDIREC Flag for searching for the connecting atoms to the specified atom (if those atoms are not given explicitly). In order to fix a specified atom, three other atoms are necessary. If those atoms are not given, they are determined by the tree structure. The three atoms chosen are three consecutive main type atoms which are closest along the tree. The search direction along the tree to get those atoms is controlled by this flag.

= 0 Search direction is "up the tree", or toward higher atom numbers. If the specified atom is connected to an atom other than a main type then the first main atom is located along the downward tree and the other two main atoms are taken along the upward tree from that point.

= -1 Search direction is downward tree (toward beginning).

ABC 2: NA , R , THETA , PHI , NB , NC , ND

FORMAT(I,3F,3I)

NA The atom number whose coordinates are to be fixed. If MOLNU.eq.0 then it refers to the absolute number.

R The bond length between "NA" and "NB".

```
   THETA      The bond angle between "NA" , "NB" and "NC".

   PHI        The dihedral angle between "NA" , "NB" , "NC" and "ND".

   NB         Number of the atom bonded to NA.

   NC         Number of the atom bonded to NB.

   ND         Number of the atom bonded to NC.

        NOTE: If NB, NC, and ND are not given, then they will be
              generated from the tree structure.

              The input for atom specification is terminated by
              a blank card.

  NOTE:       This option is terminated when IPART .eq. 'QUI'.
```

DIHED

```
   DIHED 1:    IAT , JAT , KAT , LAT

             FORMAT(4I)

   IAT ...     Atom numbers for which bond length, bond angle and
         dihedral angle are required.
         If IAT = 0 then this option is terminated.
         If KAT = 0 and LAT = 0 then only bond length between
         IAT and JAT is calculated.  If LAT = 0 then bond length
         and bond angle are calculated.

         There is no limit for the number of sets of
         atoms for calculating the internal parameters.

    NOTE:    The option is terminated by a blank card.
```

PUCKER

```
   PUCKER 1:  ITITL

            FORMAT(20A4)

   ITITL      A sub title for this option.
```

```
   PUCKER 2:  TMAX , PHASE

              FORMAT(2F10.2)

   TMAX       Amplitude for the pseudo rotation.

   PHASE      Phase angle of the pseudo rotation of the five membered
        ring.
```

```
   PUCKER 3:  KRANAM(I) , I = 1 , 5

              FORMAT(5(A4,1X))

   KRANAM(I)  The atom name of the sugar ring atoms whose pucker
        is to be changed.
```

```
   PUCKER 4:  IRES

              FORMAT(I5)

   IRES       The number of the residue to which the sugar ring belongs.
        This card can be repeated for all the residues whose
        sugar ring pucker is to be changed to the above value.

        This option is terminated when IRES = 0.
```

TELL

```
   TELL 1:    IOPTB , IOPTA , IOPTD

              FORMAT(3I5)

   IOPTB      Flag to print bond lengths.

   = 0  none
   = 1  bond lengths will be printed

   IOPTA      Flag for printing bond angles.

   = 0  none
   = 1  bond angles will be printed

   IOPTD      Flag to print dihedral angles.

   = 0  none
   = 1  dihedral angles will be printed
```

TELL 2: MOL1 , MOL2

FORMAT(2I5)

MOL1,MOL2 The range of molecules for which the internal coordinates are to be printed. If MOL2 is greater than the total number of molecules in the system it is set to the total number of molecules.

This card can be repeated for different ranges of molecules.

TELL 3: Blank card to terminate TELL.

SOL

SOL 1: KH , KO, NCUBE

FORMAT(A4,1X,A4,1X,I5)

KH The atom type for the HYDROGEN atom in water
The default atom type is 'HW'

KO The atom type for the OXYGEN atom in water
The default atom type is 'OW'

NCUBE This indicates the NCUBE**3 boxes of 216 Monte Carlo waters. The limit is determined by parameter MAXWAT in the Rev A EDIT code. It is usually set for NCUBE = 4, which would be 64 216 water boxes creating a 75A cube.

SOL 2: QH , DISO , DISH

FORMAT(3F10.5)

QH Charge on the water hydrogen atoms.

DISO Waters whose oxygen atom is closer than DISO Angstroms to any solute atom are discarded.

DISH Waters whose nearest hydrogen atom is closer than DISO Angstroms to any solute atom are discarded.

SOL 3: IORINT , IPRINT , NOWALL , IGROUP

FORMAT(10I5)

IORINT Flag to orient the water molecule along the electric field produced by the rest of the system.

= 0 No orientation.
= 1 The water molecule is reoriented keeping the oxygen position fixed.

IPRINT Flag to print the electric field vector when the water molecule is reoriented.

= 0 Electric field will not be printed.
= 1 Electric field will be printed.

NOWALL (unused)

IGROUP Controls whether CAP or BLOB solvation is produced.

= 0 BLOB option: All the atoms in the solute molecule will be taken into account and the center of mass of the solvent will be moved to that of the solute. Waters within 'CUT' (next card) of any atom in the solute will be retained.

= 1 CAP option: A spherical cap of waters of radius 'CUT' will be centered on the center of mass of the atoms specified in group format below. CAP information will be written to the topology file to be passed through PARM to the simulation programs.

= -1 CAP option as above, but the cap will be centered on the cartesian coordinates XMOVE, YMOVE and ZMOVE to be read below.

SOL 4: CUT , XMOVE , YMOVE , ZMOVE

FORMAT(10F10.5)

CUT Waters whose oxygen atom is within "CUT" Angstroms of the solute atoms chosen by IGROUP above will be retained; others are discarded.

XMOVE The X coordinate to which the center of mass of the solvent bath has to be moved if IGROUP is NEGATIVE.

YMOVE The Y coordinate to which the center of mass of the solvent bath has to be moved if IGROUP is NEGATIVE.

ZMOVE The Z coordinate to which the center of mass of the solvent bath has to be moved if IGROUP is NEGATIVE

SOL 5: ***** ONLY IF IGROUP .GT. 0 *****

The atoms of the solute to be taken for the center of mass calculation to which the CAP will be centered are read in GROUP format. See the GROUP section of the Appendices.

ADD

ADD 1: QH , KH , KO

FORMAT(F10.5,2A4)

QH The charge on the water hydrogen.
If QH .le. 0.0 then QH = 0.417.

KH The atom type for the HYDROGEN atom in water
If KH is blank the atom type defaults to 'HW'

KO The atom type for the OXYGEN atom in water
If KO is blank the atom type defaults to 'OW'

BOX

BOX 1: KH , KO, NCUBE

FORMAT(4A,1X,4A,1X,I5)

KH The atom type for the HYDROGEN atom in water
The default atom type is 'HW'

KO The atom type for the OXYGEN atom in water
The default atom type is 'OW'

NCUBE This indicates the NCUBE**3 boxes of 216 Monte Carlo waters. The limit is determined by parameter MAXWAT in the Rev A EDIT code. It is usually set for NCUBE = 4, which would be 64 216 water boxes creating a 75A cube.

BOX 2: QH , DISO , DISH

FORMAT(3F10.5)

QH Charge on the water hydrogen atoms.

DISO Minimum distance of the water oxygen atom from any of solute atoms.

DISH Minimum distance of the water hydrogen from any of the solute atoms.

BOX 3: CUTX , CUTY , CUTZ

FORMAT(10F10.5)

CUTX Any water molecule which is farther than "CUTX" along the X-axis from any atom of solute will be discarded for the Calculation. Essentially CUTX defines the minimum thickness of the solvent shell along the X-axis.

CUTY Same as CUTX but for Y-axis

CUTZ Same as CUTX but for Z-axis

BX4

Same as BOX except KO ignored and KEP = =2.*KH

WAT

WAT 1: KH , KO, NCUBE, ITRIM

FORMAT(4A,1X,4A,1X,I5)

KH The atom type for the HYDROGEN atom in water
The default atom type is 'HW'

KO The atom type for the OXYGEN atom in water
The default atom type is 'OW'

NCUBE This indicates the NCUBE**3 boxes of 216 Monte Carlo waters. The limit is determined by parameter MAXWAT in the Rev A EDIT code. It is usually set for NCUBE = 4,

which would be 64 216 water boxes creating a 75A cube.

ITRIM = 1 read in the final box dimensions below (WAT 3)

WAT 2: QH

FORMAT(3F10.5)

QH Charge on the water hydrogen atoms.

WAT 3: (only if ITRIM = 1)

FORMAT(3F10.5)

XNEW, YNEW, ZNEW (box edges in angstroms).

T4P

Same as WAT execpt KO is ignored (set to zero actually)
and KEP = -2 * KH.

FLO

FLO 1: KH , KO, NCUBE

FORMAT(4A,1X,4A,1X,I5)

KH The atom type for the HYDROGEN atom in water.
The default atom type is 'HW'.

KO The atom type for the OXYGEN atom in water.
The default atom type is 'OW'.

NCUBE This indicates the NCUBE**3 boxes of 216 Monte Carlo waters. The limit is determined by parameter MAXWAT in the Rev A EDIT code. It is usually set for NCUBE = 4, which would be 64 216 water boxes creating a 75A cube.

FLO 2: QH , DISO , DISH

```
           FORMAT(3F10.5)

   QH         Charge on the water hydrogen atoms.
```

GEN

```
   GEN 1:     DISS, SCALE, KFORM  (defaults: 2.0, 1.0, 0)

           FORMAT(2F10.5, I5)

   DISS:      Minimum contact distance for solvent-solute

   SCALE:     Scale factor for expansion/compression of
        solvent to enable the user to get the approximately
        correct density for the starting coordinates.

   KFORM:     Form of `-s´ file.

   = 0  `-s´ is formatted
   = 1  `-s´ is binary


   GEN 2:    CUTX, CUTY, CUTZ, ITYP

           FORMAT(3F10.5, I5)

   CUTX, CUTY, CUTZ:
        Distance the solvent is to extend from the solute.

   ITYP:      Generate array of solvent molecules or use box.

   = 0  Generate array from prep solvent residue only.
   = 1  Use pdb box of solvent residues.
```

++++++ END OF INPUT ++++++

Rev 4.1 Additions: Jim Caldwell, Thomas Huber, and Bill Ross
Rev 4.0 Additions: Jim Caldwell
Rev 3A Revision: George Seibel
EDIT Authors: U.C. Singh, P.K. Weiner, and S.J. Weiner
Director: P.A. Kollman
Department of Pharmaceutical Chemistry
School of Pharmacy
University of California
San Francisco CA 94143

Phone (415) 476 4637

PARM

Usage:

```
parm [-O] -i input -o output -e edtbin -f frcfld
     -c prmcrd -p prmtop -m frcmod
```

–O Overwrite output files if they exist.

The purpose of this module is to assign the force field parameters for all the constants in the potential energy function to the topology file created by EDIT. The constants such as equilibrium bond lengths, angles, dihedral angles and their force constants and the non-bonded 6-12 parameters and H-bond 10-12 parameters are read in from a parameter file. Parameters are allocated based on Amber atom types. Internal coordinate constraints can be placed on any of the atoms in the system by the use of the CONS option. Note: force constants are in terms of kcal/mol, distances in Ångstroms, bond angles in radians for standard parameters and degrees for internal constraints, and dihedral angles in terms of degrees. Note also that both internal and cartesian constraints can be applied directly in the simulation programs, which are also capable of holding groups of atoms fixed with the 'belly' option.

PARM creates both a molecular topology file and a coordinate file that can be read by the minimizer, the dynamics module, the normal modes module, and the static and dynamics analysis modules. This version of PARM also handles input for Free Energy Perturbation (GIBBS) calculations.

Files:

input	5	read control data
output	6	output user information and diagnostics
frcfld	10	read the force field parameters
frcmod	16	read modified force field parameters
prmtop	12	output the molecular topology for SANDER, GIBBS, NMODE, ANAL, MDANAL and CARNAL
edtbin	15	read the molecular topology generated by EDIT
prmcrd	18	output the coordinates

Input parameters:

```
- 1 -       TITLE FOR THE RUN.

                FORMAT(20A4)
```

```
   TITLE       Title for identification purposes only.

-------------------------------------------------------------------------

   - 2 -      JFORI , KFORM , NAMNB , IFCON , KPERT ,IPOLA

                  FORMAT(20A4)

 2.1   JFORI      Flag for the type of format for the molecular topology
         file from the EDIT module.

        'BIN'     The file is in binary form.
        'FOR'     The file is in formatted form.
                  Only the binary form is currently output from the EDIT.

 2.2   KFORM      Flag for the type of format for the molecular topology
         file output from this module.

        'BIN'     The parameter and coordinate files will be written in
                  binary form.
        'FOR'     The files will be in formatted form.
                  The formatted form is necessary when the output topology
                  file will be used on a different type of machine than
                  the one on which it was created.

 2.3   NAMNB      The name of the non-bonded parameters.  This name
         must match with the label on one of the nonbonded sets in the
         parameter file.  It is used to enable one to keep several
         different sets of non-bonded parameters in the same file
         and use only the one desired for a given calculation.

 2.4   IFCON      Flag for reading internal parameter constraints.

        'CONS'    Internal coordinate constraints will be read.
        '    '    No internal constraints.

 2.5   IPERT      Flag for reading Perturbation Information

        'PERT'    Perturbation Information will be input
        '    '    No Perturbation Calculation

 2.6   IPOLA      Flag for adding polarizabilites to the topology file

        'POLA'    atomic polariziabilites will be added to the topology
        '    '    No polarizability info

-------------------------------------------------------------------------

   - 3 -      ILBL , ILTP , ILPP

                  FORMAT(3I5)

 3.1   ILBL       Flag for printing all the bonds in the system with the
         parameters assigned to them.
```

= 0 None (recommended)
= 1 All the bonds will be output.

3.2 ILTP Flag for printing all the angles.

= 0 None (recommended)
= 1 All the angles will be output.

3.3 ILPP Flag for printing all the dihedrals.

= 0 none (recommended)
= 1 All the dihedrals will be output.

The list of torsional parameters (force field terms) is itemized by GROUP of parameters that refer to one underlying torsional sequence, although internally the program keeps a separate sequence for each parameter, i.e. separately labeling elements of groups. These internal sequence numbers (ICP indices) are used if you request that a list of topological torsions and the assigned parameters be output. E.g. in the force field listing:

```
   [...]

   6   CT-CT-C -O   1    0.07  180.0    3.
   7   N -CT-C -O   1    0.07  180.0    3.
   8   H -N -C -O   1    2.50  180.0   -2.
                        0.65    0.0    1.

  Improper Dihedrals:

   9   X -X -N -H        1.00  180.0    2.
  10   X -X -C -O       10.50  180.0    2.
  11   X -CT-N -CT       1.00  180.0    2.
```

where the "8" refers to two torsional parameters here, that is, two terms that are both used (a Fourier series) for a single torsion type (H-N-C-O). But for bookeeping, those are two separate torsional parameters, each with its own ICP number. In terms of ICP numbers, they're 8 and 9, and the next item "9" is ICP 10, etc. In the topological listing:

```
   [...]

   All Dihedrals in System:

     I     I1    J1    K1     L1              ICP
    1.      1    2     5    6   HC-CT-C -O       5
    2.      2    7     5    6   CT-N -C -O      11
```

Here, ICP 11 corresponds to term 10, above.

```
   - 4 -      IPARML , IOHB , IONB

                    FORMAT(3I5)

 4.1   IPARML     Flag to list all the parameters read.

        = 0   None
        = 1   All the modified parameters will be output. (recommended)
        = 2   All parameters printed (modified + standard)
              Note:  some parameters that are not used in the
              system may appear in the output.

 4.2   IOHB       Flag to list all the H-bond pair types.

        = 0   None
        = 1   All the H-bond pair types will be output.

 4.3   IONB       Flag to list all the non-bonded pair types.

        = 0   None
        = 1   All the non-bonded pair types will be output.

-------------------------------------------------------------------------

   - 5 -      ***** ONLY IF IPERT .EQ. 'PERT' *****

              ITITL

                 FORMAT(20A4)

 5.1   ITITL      Control word for reading the perturbation input

        'PERTURBATION'
              Perturbation input begins.  This line is mandatory
              if IPERT .eq. 'PERT'

-------------------------------------------------------------------------

   - 5A -     A TITLE CARD FOR EACH TYPE OF INPUT TO BE GIVEN BELOW

              ITITL

                FORMAT(20A4)

 5a.1  ITITL      This card is used as a title for each perturbed
         residue.  It may also be a control variable.  If the first
         four characters match any of the keywords below, it is taken
         as a control variable and the action indicated below is taken.
         Note: each NEW (not repeat) residue requires a descriptive
         title card.

        'END '    Perturbation input ends
        'REPEAT'  Flag for repeating the previous residue input for
                  the following residues. This is useful for identical
```

residues.

- 5B - CONTROL CARD FOR THE TYPE OF INPUT
ITYPE , LABEL , INUMR

FORMAT(2A,I)

5b.1 ITYPE Flag for the type of Input

'RES ' The input will be for a whole residue. End the residue input with 'END' on the following line, in addition to the 'END' that ends perturbation input.

'ATOM' The input will be given for individual atoms. Atom input is ended with a blank line.

5b.2 LABEL The name of the residue you are perturbing TO if ITYPE .EQ. 'RES '.

5b.3 INUMR The perturbed residue number if ITYPE .EQ. 'RES '

NOTES:

The INITIAL state of the perturbed system is that which is defined in the topology file from the EDIT module. The FINAL state of the perturbed system is input here.

The perturbed residue must have the same topology and number of atoms as the original residue. Thus if you want to perturb alanine into phenylalanine you must include the proper number of "dummy" atoms in the initial alanine residue to compensate for the extra atoms in the phenyl ring. These "dummy" atoms are not the same as the dummy atoms used in PREP for definition of the coordinate space, and MUST NOT be given the same atom type symbol as those. The Perturbation dummy atoms are not removed by EDIT. They remain in the system throughout all calculations.

The 'ATOM' option should ONLY be used under two circumstances: 1) if the entire residue consists of only one atom, and 2) if one wants to perturb ONLY the VDW parameters of an atom. Otherwise, the 'RES' option should be used. This ensures that proper charge relationships will be maintained after perturbing an atom or several atoms. For example, if benzene is to be perturbed into pyridine, all the atoms in the benzene residue must be treated as perturbed atoms. This allows one to account for the polarity which develops during the introduction of the nitrogen atom. For more information on the general theory see:
U. C. Singh, F. K. Brown, P. A. Bash, P. A. Kollman,
J. Am. Chem. Soc. 109, 1607, 1987.

```
   - 5C -     INUM , IGRPER , ISMPER , CGPER

                 FORMAT(I,2A,F)

 5c.1  INUM       The perturbed atom number if ITYPE .EQ. 'ATOM'
         It is ignored in RES input.

 5c.2  IGRPER     The atom name for the perturbed atom
                  (final state)

 5c.3  ISMPER     The atom type symbol for the perturbed atom
                  (final state)

 5c.4  CGPER      The partial charge for the perturbed atom
                  (final state)

 5c.5  POLPER     The atomic polarizability for the perturbed atom
                  (final state)

              NOTE: In the case of residues the total number of
                    input atoms should match the number of atoms
                    for that residue in the EDIT topology file.
                    If you are growing atoms where there were none
                    before, your EDIT topology file must have been
                    built with appropriate dummy atoms in their place.
```

```
   - 5D -     INPUT FOR REPEATING RESIDUES

              ITRES , IRES, [IRES, ...]

                FORMAT(2A,20I)

 5d.1  ITRES      Control flag for the residue repeat input
         It must be 'RES ' for repeating the immediately preceding
         residue information

 5d.2  IRES       The residue numbers for which the previous input
         is to be repeated.  Max 10 per card.

         ** The REPEAT group is terminated with an 'END ' card, in
            addition to the 'END ' that terminates perturbation input.
```

```
   - 6 -      ***** ONLY IF IFCON .EQ. 'CONS' *****

              IAT , JAT , KAT , LAT , REQ , CONK

                  FORMAT(4I5,2F10.2)

       We recommend applying internal constraints via the simulation
       programs' input rather than using this option. These programs
```

```
      also have options for harmonic cartesian constraints and
      freezing of groups of atoms (with the belly option).

6.    IAT, JAT, KAT, LAT
        Atom numbers for which internal constraints are to be applied.

       IAT .le. 0 : reading of constraint cards is terminated.
       KAT .le. 0 : a bond constraint is assumed between the atoms
                    IAT and JAT.
       LAT .le. 0 : an angle constraint is assumed between the atoms
                    IAT, JAT, and KAT.
       IJKL > 0   : If all four are non zero then a dihedral angle
                    constraint is assumed between them.

6.5   REQ        The constrained value for either bondlength, bondangle
        or dihedral angle. Bondlengths are in angstroms and angles are
        in degrees.

6.6   CONK       The force constant for the constrained value
        (kcal/mol/Angstrom or kcal/mol/degree).


      The input for constraints is terminated by a blank card.
```

Force field information on the file *frcfld*: The following section of this document describes the format of the AMBER Force Field Parameter File. It is not expected that the user will ordinarily modify this file; rather modifications should ordinarily be entered through the *frcmod* file described further below. Of course, major changes, such as using the AMBER/OPLS force field rather than the AMBER one, would best be made by changing this file. WARNING: multiple entries for the same atom symbols within a single *frcfld* or *frcmod* file can lead to undefined results, e.g. if there are two definitions of angle energies between atom types A, B and C one of them is picked arbitrarily.

```
  - 1  -     ITITL

                 FORMAT(20A4)

  ITITL      A title for identification of the parameter set.
```

```
  - 2 -      ***** INPUT FOR ATOM SYMBOLS AND MASSES *****

             KNDSYM , AMASS, ATPOL

                 FORMAT(A2,2X,F10.2x,f10.2)

  KNDSYM     The unique atom symbol used in the system.

  AMASS      Atomic mass of the center having the symbol "KNDSYM".

  ATPOL      The atomic polarizability for each atom (in A**3)
```

This is the type of polarizability used in sander
and gibbs. No parameters are supplied for this since
the feature is still in development (Amber 4.1).

NOTE: All the unique atomic symbols and their masses must
be read. The input is terminated by a blank card.

- 3 - ***** INPUT FOR ATOM SYMBOLS THAT ARE HYDROPHILIC *****

JSOLTY(I)

FORMAT(20(A2,2X))

JSOLTY(I) The atom symbols which are hydrophilic in solution.
This information is read but not used.

The input is terminated when a blank value is read for
the atom symbol.

- 4 - ***** INPUT FOR BOND LENGTH PARAMETERS *****

IBT , JBT , RK , REQ

FORMAT(A2,1X,A2,2F10.2)

IBT,JBT Atom symbols for the two bonded atoms.

RK The harmonic force constant for the bond "IBT"-"JBT".
The unit is kcal/mol/(A**2).

REQ The equilibrium bond length for the above bond in angstroms

The input is terminated by a blank card.

- 5 - ***** INPUT FOR BOND ANGLE PARAMETERS *****

ITT , JTT , KTT , TK , TEQ

FORMAT(A2,1X,A2,1X,A2,2F10.2)

ITT,... The atom symbols for the atoms making an angle.

TK The harmonic force constants for the angle "ITT"-"JTT"-
"KTT" in units of kcal/mol/(rad**2) (radians are the
traditional unit for angle parameters in force fields).

TEQ The equilibrium bond angle for the above angle in degrees.

The input is terminated by a blank card.

- 6 - ***** INPUT FOR DIHEDRAL PARAMETERS *****

IPT , JPT , KPT , LPT , IDIVF , PK , PHASE , PN

FORMAT(A2,1X,A2,1X,A2,1X,A2,I4,3F15.2)

IPT, ... The atom symbols for the atoms forming a dihedral angle. If IPT .eq. 'X ' .and. LPT .eq. 'X ' then any dihedrals in the system involving the atoms "JPT" and and "KPT" are assigned the same parameters. This is called the general dihedral type and is of the form "X "-"JPT"-"KPT"-"X ".

IDIVF The factor by which the torsional barrier is divided. Consult Weiner, et al., JACS 106:765 (1984) p. 769 for details. Basically, the actual torsional potential is

(PK/IDIVF) * (1 + cos(PN*phi - PHASE))

PK The barrier height divided by a factor of 2.

PHASE The phase shift angle in the torsional function. The unit is degrees.

PN The periodicity of the torsional barrier.
NOTE: If PN .lt. 0.0 then the torsional potential is assumed to have more than one term, and the values of the rest of the terms are read from the next cards until a positive PN is encountered. The negative value of pn is used only for identifying the existence of the next term and only the absolute value of PN is kept.

The input is terminated by a blank card.

- 7 - ***** INPUT FOR IMPROPER DIHEDRAL PARAMETERS *****

IPT , JPT , KPT , LPT , IDIVF , PK , PHASE , PN

FORMAT(A2,1X,A2,1X,A2,1X,A2,I4,3F15.2)

The input is the same as in for the dihedrals except that the torsional barrier height is NOT divided by the factor idivf. The improper torsions are defined between any four atoms not bonded (in a successive fashion) with each other as in the case of "regular" or "proper" dihedrals. Improper dihedrals are used to keep certain groups planar and to prevent the racemization of certain centers in the united

atom model. Consult the above reference for details.

Important note: all general type improper dihedrals (e.g. x -x -ct-hc) should appear before all specifics (ct-ct-ct-hc) in the parm list. Otherwise the generals will override the specific with no warning.

The input is terminated by a blank card.

- 8 - ***** INPUT FOR H-BOND 10-12 POTENTIAL PARAMETERS *****

KT1 , KT2 , A , B , ASOLN , BSOLN , HCUT , IC

FORMAT(2X,A2,2X,A2,2x,5F10.2,I2)

KT1,KT2 The atom symbols for the atom pairs for which the parameters are defined.

A The coefficient of the 12th power term (A/(r**12)).

B The coefficient of the 10th power term (-B/(r**10)).

ASOLN Not used

BSOLN Not used

HCUT Not used

IC Not used

- 9 - ***** INPUT FOR EQUIVALENCING ATOM SYMBOLS FOR THE NON-BONDED 6-12 POTENTIAL PARAMETERS *****

IORG , IEQV(I) , I = 1 , 19

FORMAT(20(A2,2X))

IORG The atom symbols to which other atom symbols are to be equivalenced in generating the 6-12 potential parameters.

IEQV(I) The atoms symbols which are to be equivalenced to the atom symbol "IORG". If more than 19 atom symbols have to be equivalenced to a given atom symbol they can be included as extra cards.

It is advisable not to equivalence any hydrogen bond atom type atoms with any other atom types.

NOTE: The input is terminated by a blank card.

- 10 - ***** INPUT FOR THE 6-12 POTENTIAL PARAMETERS *****

LABEL , KINDNB

FORMAT(A4,6X,A2)

LABEL The name of the non-bonded input parameter to be used. It has to be matched with "NAMNB" read through unit 5. The program searches the file to load the the required non-bonded parameters. If that name is not found the run will be terminated.

KINDNB Flag for the type of 6-12 parameters.

'SK' Slater-Kirkwood parameters are input. see "caution" below.

'RE' van der Waals radius and the potential well depth parameters are read.

'AC' The 6-12 potential coefficients are read.

NOTE: All the non equivalenced atoms' parameters have to be given.

The input is terminated when label .eq. 'END'

CAUTION: the polarizabilities mentioned below are NOT the polarizabilities used in the sander (min/md) code. KINDNB 'SK' parameters are not currently part of the AMBER force field. See card 2, ATPOL for sander polarizability.

- 10A - ***** ONLY IF KINDNB .EQ. 'SK' *****

LTYNB , POL , XNEFF , RMIN

FORMAT(2X,A2,6X,3F10.6)

LTYNB Atom symbol.

POL Atomic polarizability for the atom centers having the the above symbol.

XNEFF Effective number of electrons on the atom centers having the above symbol.

RMIN van der Waals radius of the atom center having the above symbol.

```
------------------------------------------------------------------------

   - 10B -     ***** ONLY IF KINDNB .EQ. 'RE' *****

               LTYNB , R , EDEP

   LTYNB       Atom symbol.

   R           The van der Waals radius of the atoms having the symbol
               "LTYNB"  (Angstoms)

   EDEP        The 6-12 potential well depth. (kcal/mol)

------------------------------------------------------------------------

   - 10C -     ***** ONLY IF KINDNB .EQ. 'AC' *****

               LTYNB , A , C

   LTYNB       Atom symbol.

   A           The coefficient of the 12th power term (A/r**12).

   C           The coefficient of the 6th power term (-C/r**6).

------------------------------------------------------------------------
```

Modified force field parameters in file *frcmod*: This file is normally the one that will be changed by the user. It consists of a 1-card title, followed by a blank line, then keyword sections. The allowed keywords (appearing in columns 1-4) are:

```
MASS           follow this card by card of type - 2 - listed in the
               unit 10 instructions above.  End with a blank line.

BOND           follow this card by card of type - 4 - listed in the
               unit 10 instructions above.  End with a blank line.

ANGL           follow this card by card of type - 5 - listed in the
               unit 10 instructions above.  End with a blank line.

DIHE           follow this card by card of type - 6 - listed in the
               unit 10 instructions above.  End with a blank line.

IMPR           follow this card by card of type - 7 - listed in the
               unit 10 instructions above.  End with a blank line.

HBON           follow this card by card of type - 8 - listed in the
               unit 10 instructions above.  End with a blank line.

NONB           follow this card by card of type - 10A, B or C - listed
               in the unit 10 instructions above.  E.g. if you specify
               STDA in parm.in for the "regular" parm.dat file, this is
```

```
the convention that will be used when reading frcmod.
End with a blank line.
```

Any or all of the keywords may be missing, if you have no changes for that section. The entire file can be missing if you have no changes at all to make to the standard force field. Restrictions: note that you cannot modify the equivalence pattern set up in the standard force field.

If you have parameters in the *frcmod* file that modify values in the standard parameter file, and if *iparml* is set to 2, then both the original and the modified parameters will be printed to the *output* file. The modified parameters will be marked with asterisks, and it is these values that will be used in subsequent calculations.

SANDER

Usage: `sander [-O] -i mdin -o mdout -p prmtop -c inpcrd -r restrt -ref refc -x mdcrd -v mdvel -e mden -inf mdinfo`

`-O` Overwrite output files if they exist.

This is a guide to *sander*, the AMBER module which carries out energy minimization, molecular dynamics, and NMR refinements. The acronym stands for **S**imulated **A**nnealing with **N**MR-**D**erived **E**nergy **R**estraints, but in fact the NMR-related functionality of *sander* will perhaps most often be used for things not related to NMR. *All of the features of the minmd program from AMBER version 4.0 are now incorporated here*.

The annealing, ''weight change,'' ''restraints'' and NMR-specific portions of *sander* were primarily written by David Pearlman and David Case. All of the AMBER crew listed on the title page contributed to the general portions; the polarization implementation is that of Jim Caldwell and Liem Dang.

Parallelism. This is the first version of sander to fully integrate parallel code. In 4.0, an SGI shared memory version by Roberto Gomperts and Michael Schlenkrich of SGI with assistance from Thomas Cheatham was distributed on request, as were a message-passing version for the SP1 by Steven Chin of IBM and a KSR version by David Zirl and Nick Camp. A general PVM version by Terry Lybrand and Eric Swanson of the University of Washington was included as a separate source tree in the later 4.0 release. We also had an unreleased version of Steve Debolt's AMBERCUBE [8] (based on release 3A) that ran on nCube and had been ported to 4.0 on the Intel Paragon by David Case of Scripps with help from Jerry Greenberg and Jack Rodgers of San Diego Supercomputer Center. (George Seibel of UCSF and Tom Darden of NIEHS also had done shared memory versions of previous releases for Cray and SGI, respectively.)

In 4.1, all message-passing parallel code falls under a generalized MPI interface developed by Jim Vincent and Ken Merz of Pennsylvania State University, who provided PVM, SPx and T3D versions. [9] (The PVM, *etc*. message-passing libraries are only used for systems that do not have MPI implemented.) Thomas Cheatham and David Case helped to integrate, extend, and optimize this work. The SGI shared memory version from 4.0 was improved by Gomperts and Schlenkrich of SGI, and has been reorganized and incorporated into the release by Thomas Cheatham. Other credits: Steve Chin (IBM, SPx optimization), Jeyapandian Kottalam, Mike Page and Asiri Nanayakkara (Cray optimization), Michael Crowley (Pittburgh Supercomputer Center, T3D portable namelist port, PME development), and Thomas Huber (Ludwig Maximilian Universitaet, TCGMSG library). Thomas Huber also

[8] DeBolt, S.E. and Kollman, P.A. (1993) *J. Comput. Chem.* **14**, 312.

[9] "A Highly Portable Parallel Implementation of AMBER 4 Using the Message Passing Interface Standard," Vincent, J. and Merz, K.M. (submitted) *J.Comput.Chem*.

added truncated octahedral periodicity to sander.

Particle Mesh Ewald. The Particle Mesh Ewald (PME) method, implemented originally in AMBER 3a and contributed by Tom Darden [10] of the NIEHS, is included in this release as an *experimental* option. The PME method not only provides a better treatment of long range electrostatics (at a modest computational cost), but can be applied in both rectangular and non-rectangular periodic boundary simulations.

We have divided this manual into the six sections listed below.

Purpose	*Sections involved*
Simple min/md	1
varying parameters over time (simulated annealing)	1,2
using internal restraints (including NMR distance & angle constraints)	3,4
nmr refinement using NOESY volume restraints	5
nmr refinement using chemical shift restraints	6

If you are just doing "standard" minimization or dynamics, read section *one*, and ignore the rest. If you want to carry out simulated annealing, consult section *two*. Those who wish to carry out simulations while imposing internal coordinate restraints should also read sections *three* and *four*. Sections *five* and *six* allow you to add sophisticated penalty functions during NMR refinement. Summarizing:

Sander provides standard protocols for minimization and molecular dynamics, and we use it for just about everything except free energy calculations. Some of the features are outlined in the following paragraphs:

(1) *Sander* provides direct support for the AMBER and AMBER/OPLS force fields for proteins and nucleic acids, and for the TIP3 and TIP4 models for water. Other types of restraints can be applied, 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 can be supported. In addition, "non-additive" force fields based on atom-centered dipole polarizabilities can be invoked.

(2) The relative weights of various terms in the force field can be varied over time. It is also straightforward to choose a constant weighting that implements a "geometric" force field, in which bonds and angles are kept fixed, torsions are free and non-bonded interactions consist solely of chargeless Van der Waals 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.

[10] Ported to AMBER 4.1 by Tom Darden with the assistance of Thomas Cheatham. For citation information and more information about the method, see the input description in section one of this manual.

(3) Two periodic imaging geometries are included: rectangular parallelopiped and truncated octahedron (box with corners chopped off). The size of the repeating unit can be coupled to a given external pressure, and velocities can be coupled to a given external temperature by several schemes. The external conditions and coupling constants can be varied over time, so various simulated annealing protocols can be specified in a simple and flexible manner.

(4) 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. The penalty function can consist of as many as three types of region: it can be flat between an 'inner' 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 r_1 and r_4). 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 is available.

(5) 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.

(6) Restraints can be directly defined in terms of NOESY intensities (calculated with a relaxation matrix technique), scalar coupling constants and proton chemical shifts. There are provisions for handling overlapping peaks or ambiguous assignments. In conjunction with distance and angle constraints, this provides a powerful and flexible approach to NMR structural refinements.

In addition to the descriptions of options given below, the user may inspect the files in the *amber41/demo* directory for concrete examples and look in the *amber41/Questions* directory for discussions of issues that arise in practice.

We now turn to a description of the files used by sander.

Files used

Usage: `sander [-O] -i mdin -o mdout -p prmtop -c inpcrd -r restrt -ref refc -x mdcrd -v mdvel -e mden -inf mdinfo`

`-O` Overwrite output files if they exist.

On VMS systems, files are assigned by Fortran unit number. These unit numbers are also sometimes useful to reference when viewing i/o-related operating system error messages, and are given below along with a description of each file.

file	unit	in/out	purpose
mdin	5	input	control data for the min/md run
prmtop	8	input	molecular topology, force field, periodic box type, atom and residue names
inpcrd	9	input	initial coordinates and (optionally) velocities and periodic box size
refc	10	input	(optional) reference coords for position constraint
mdout	6	output	user readable state info and diagnostics
mdinfo	7	output	latest mdout-format energy info
restrt	16	output	final coordinates, velocity, and box dimensions if any - for restarting run
mdcrd	12	output	coordinate sets saved over trajectory
mdvel	13	output	velocity sets saved over trajectory
mden	15	output	extensive energy data over trajectory

Overview of the contents of *mdin*

Section	*Comments*	*Format*
ONE	Standard minimization and dynamics input	`&cntrl` namelist
TWO	Varying conditions	Parameters for changing temperature, restraint weights, etc. during the MD run. Sometimes called "weight change lines". Each weight change line is specified by a separate `&wt` namelist specifier, ending with `&wt type='END', &end.` *NOTE:* the termination of this section is `&rst iat=0, &end` at the end of section FOUR. *I.e*. without an explicit section THREE or section FOUR, you must use this line if you want a group specification following this one to be read properly.
THREE	I/O redirection	TYPE=*filename* lines. Optional. Section ends with the first non-blank line which does not correspond to a recognized redirection.
FOUR	Distance and angle restraints	Multiple `&rst` namelists; if a *DISANG=filename* redirection was given in TWO, these are read from *filename* instead of *mdin*. One `&rst` definition is given per restraint. Section FOUR is terminated by `&rst iat=0, &end`.
FIVE	NOESY volume restraints	Read only if NMRMAX= 2 or 3. If a *NOEEXP=filename* was given in TWO, these are read from *filename* instead of *mdin*. Defines molecular subgroups. Each definition consists of one `&noeexp` namelist followed by the group cards defining the subgroup. Ended by `&noeexp npeak(1)=-1, &end`.
SIX	Chemical shifts restraints	Read only if NMRMAX= 3 or 4. If a *SHIFTS=filename* was given in TWO, these are read from *filename* instead of *mdin*. Exactly one `&shf` namelist must be provided for this section.

SECTION ONE: General minimization and dynamics parameters

Each of the variables listed below is input in a namelist statement with the namelist identifier `&cntrl`. You can enter the parameters in any order, using keyword identifiers. Variables that are not explicitly listed retain their default values. Support for namelist input is included in almost all current Fortran compilers, and is a standard feature of Fortran 90. In addition, a "portable" namelist implementation, written in Fortran by N.H.F. Beebe, is included, to allow namelist input on (almost) all machines. This "portable" version is actually an improvement over most native implementations, because it gives better error messages in case of problems. A detailed description of the namelist convention is given in Appendix B.

In general, namelist input consists of an arbitrary number of comment cards, followed by a record whose first 7 characters after a '&' (e.g. " `&cntrl` ") name a group of variables that can be set by name. This is followed by statements of the form " `maxcyc=500, diel=2.0, ...` ", and is concluded by an " `&end` " token. The files in the demo directory contain examples of this format. The first 'card' or line of input contains a title, which is then followed by the `&cntrl` namelist. Note that the first character on each line of a namelist block must be a blank.

A simple input file

```
Sample input file : just a few steps of minimization.
    [minimize for 50 cycles, print results every 10 steps]
 &cntrl
     imin=1, maxcyc=50, ntpr=10, scee=2.0,
 &end
```

Input parameters

General flags describing the calculation

TIMLIM Time limit, in seconds, for the job. Default 999999.

IMIN Flag to run minimization

= 0 No minimization (only do molecular dynamics; default)

= 1 Perform minimization (and no molecular dynamics)

NMRMAX

= 0	no nmr-type analysis will be done; default
> 0	NMR restraints/weighting changes will be read
= 2 or 3	NOESY volume restraints will be read as well
= 3 or 4	chemical shift restraints will also be read

Nature and format of the input

NTX

Option to read the initial coordinates, velocities and box size from the "inpcrd" file (also see INIT). The options 1-2 must be used when one is starting from minimized or model-built coordinates. If an MD restrt file is used as inpcrd, then options 4-7 may be used. Note: BOX (the periodic box lengths) is written to the restrt file in periodic boundary runs. If NTB.gt.0 (a periodic boundary run) and NTX.lt.6, the box sizes in the prmtop are used; otherwise, the box sizes from the inpcrd (MD restrt) file will be used. This enables one to use the last box from the constant pressure regime when switching to constant volume runs.

= 1	X is read formatted with no initial velocity information (default)
= 2	X is read unformatted with no initial velocity information
= 4	X and V are read unformatted.
= 5	X and V are read formatted.
= 6	X, V and BOX(1..3) are read unformatted.
= 7	X, V and BOX(1..3) are read formatted.

IREST

Flag to restart the run.

= 0	No effect (default)
= 1	restart calculation (i.e. read restart time and set INIT= 4. Requires velocities in coordinate input file, so you also may need to reset NTX if restarting MD)

NTRX

Format of the cartesian coordinates for restraint from file "refc". Note: the program expects file "refc" to contain coordinates for all the atoms in the system. A subset for the actual restraints is selected by the GROUP input which follows.

= 0	Unformatted (binary) form
= 1	Formatted (ascii, default) form

Nature and format of the output

NTXO Format of the final coordinates, velocities, and box size (if constant volume or pressure run) written to file "restrt".

= 0 Unformatted

= 1 Formatted (default).

NTPR Every NTPR steps energy information will be printed in human-readable form to files "mdout" and "mdinfo". "mdinfo" is closed and reopened each time, so it always contains the most recent energy and temperature. Default 50.

NTWX Every NTWX steps the coordinates will be written to file "mdcrd". NTWX=0 inhibits all output. Default 0.

NTWV Every NTWV steps the velocities will be written to file "mdvel". NTWV=0 inhibits all output. Default 0.

NTWE Every NTWE steps the energies and temperatures will be written to file "mden" in compact form. NTWE=0 inhibits all output. Default 0.

NTWXM The maximum number of steps that NTWX is active. At this number of steps no more trajectories will be written to file "mdcrd". Set this to 0 to disable the limit.

NTWVM Analogous to NTWXM for velocities. 0 to disable.

NTWEM Analogous to NTWXM for energies. 0 to disable.

IOUTFM Format of velocity, coordinate, and energy sets

= 0 Formatted (default)

= 1 Binary

NTWPRT Coordinate/velocity archive limit flag. This flag can be used to decrease the size of the coordinate / velocity archive files, by only including that portion of the system of greatest interest. (E.g. one can print only the solute and not the solvent, if so desired).

Coord/velocity archives will include:

= 0 all atoms of the system (default).

< 0 only the solute atoms.

> 0 only atoms 1->NTWPRT.

Potential function

NTF Force evaluation. Note: If SHAKE is used (see NTC), it is not necessary to calculate forces for the constrained bonds.

= 1 complete interaction is calculated (default)

= 2 bond interactions involving H-atoms omitted (use with NTC=2)

= 3 all the bond interactions are omitted (use with NTC=3)

= 4 angle involving H-atoms and all bonds are omitted

= 5 all bond and angle interactions are omitted

= 6 dihedrals involving H-atoms and all bonds and all angle interactions are omitted

= 7 all bond, angle and dihedral interactions are omitted

= 8 all bond, angle, dihedral and non-bonded interactions are omitted

NTB Periodic boundary. If NTB .EQ. 0 then a boundary is NOT applied regardless of any boundary condition information in the topology file. The value of NTB specifies whether constant volume or constant pressure dynamics will be used. Options for constant pressure are described in a separate section below.

= 0 no periodicity is applied (default)

= 1 constant volume

= 2 constant pressure

If NTB .NE. 0, there must be a periodic boundary in the topology file. Constant pressure is not used in minimization (IMIN=1, above).

For a periodic system, constant pressure is the only way to equilibrate density if the starting state is not correct. For example, the solvent packing scheme used in EDIT can result in a net void when solvent molecules are subtracted which can aggregate into ‘vacuum bubbles’ in a constant volume run. Another consideration is box shrinkage under constant pressure, which if the solute clearance has been chosen too close to the cutoff distance can result in solvent molecules ‘seeing’ parts of the solute in opposite directions (not desirable if one believes that less correlated interactions are significantly more like simulations in free solution). The remedy for this is to allow enough margin when building the box.

For a 7682-atom system, [11] constant volume uses about 45% more computing time than having no periodicity (because of the additional atom pairs across the periodic boundary). For this DNA system, constant pressure uses about 30% more time than constant volume (because of the work involved in scaling the box).

IDIEL Type of dielectric function to be used in calculating the electrostatic energy.

= 0 distance dependent dielectric function. This is used to mimic the presence of a high dielectric solvent, typically for simulating water when no explicit water is present.

= 1 constant dielectric function. This is used when there is explicit solvent (e.g. water) in the calculation, or for a true gas phase calculation. Default.

DIELC Dielectric multiplicative constant for the electrostatic interactions. If DIELC .le. 0.0 then DIELC = 1.0. DIELC and IDIEL are coupled. For example to obtain a dielectric constant of 4rij set DIELC=4 and IDIEL=0. Default 1.0.

CUT The primary cutoff distance for non-bonded interactions. CUT should be no more than half the shortest BOX dimension in order to maintain spherical symmetry in the nonbonded potential. Note that AMBER only uses a residue-based cutoff. This means that if any atom of one residue is within CUT of any atom of another residue, every atom in each residue will see every atom of the other residue. This is done to avoid splitting the residue dipoles. The average effective cutoff is thus increased by the average residue diameter. Default 8.0.

NTNB Non-bonded pair list.

= 0 no pair list will be generated and no nonbonded interactions are calculated.

= 1 Normal behavior (default; recommended).

NSNB After NSNB steps the non-bonded pair list will be updated. It is recommended that the pairlist be updated every 25fs, but for very mobile systems or when short cutoffs are used it may be necessary to update the pairlist more frequently. If the nonbonded cutoff is larger than the system size (ie, no cutoff), you should set NSNB to a large value so that the pairlist is only constructed once. Default 25.

NTID Water pairlist method.

= 0 In periodic systems the water pairlist is generated from oxygen coordinates. Default and recommended.

[11] A 274-atom DNA hexamer with counterions in a 45 Å box of water (built with 10.5 Å solute clearance) using an 8 Å cutoff.

= 86 pairlist generated on residue basis from atom coordinates. I.e. if any pair of atoms in different waters is within the cutoff, all the interactions between the 2 waters are used. This yields an effective cutoff distance that is somewhat longer than that specified in CUT. This option may be extremely slow and is provided only for comparison to old runs.

SCNB 1-4 vdw interactions are divided by SCNB. Default 2.0.

SCEE 1-4 electrostatic interactions are divided by SCEE; the 1991 and previous force fields used 2.0, while the 1994 force field uses 1.2. *No default; must be set*.

CUT2ND An (optional) secondary cutoff. If CUT2ND > 0.0, then at every nonbonded update (every NSNB steps), the energies and forces due to interactions in the range CUT< Rij <= CUT2ND will be determined. These energies and forces will be added to the non-bonded interactions within CUT distance at every timestep. The idea is that long-range interactions change more slowly than short range interactions, and thus this dual cutoff method allows one to include longer-range information at only a moderate additional cost. Default 0.0.

ICHDNA Option to modify the charge of end hydrogens. This is useful for "in vacuo" simulations of RNA and DNA. Without this option, energy minimization calculations on nucleotides will result in bonding between the 5' and 3' hydrogens and the corresponding phosphate groups. This option transfers the charge from H5' to O5' so that the hydrogen on the end is neutral.

= 0 no charge modification (default)

= 1 modify charge

The "soft repulsion" option

ISFTRP

= 0 No "soft repulsions" (default).

If ISFTRP > 0, a "soft" repulsion-only potential term will be used in place of the standard 6-12 potential. This term has the form

$$E = K_{rep} (r_o^2 - r^2)^2 \text{ for } r < r_o$$
$$E = 0 \text{ for } r > r_o$$

where r_o is the sum of the van der Waals radii of the interacting atoms, r is their interatomic distance, and K_{rep} is a force constant. This type of potential has shown some usefulness in improving the efficiency of restrained refinement using MD.

= 1 The standard 10-12 potential will still be used for interactions between hydrogen bonding atoms, rather than the "soft" repulsion-only term.

>=2 The soft-repulsion term will replace the 10-12 term for hydrogen bonds, as well.

RWELL Default 0.0. If ISFTRP > 0, RWELL gives the initial value of K_{rep}. All interactions use the same value of K_{rep}, which can be changed using the SOFTR option in the NMR control file (see below).

[Note: If, in the force field, either epsilon or r_o for an atom is specified to be zero, that atom will not contribute to the vdw potential energy. This is always true, regardless of the values of ISFTRP or RWELL.]

Caution: Note that the van der Waals radii in the "standard" force field may not be what you want for soft repulsion. In particular, the atom type *HC* (hydrogen bonded to carbon) has a large value for r* (1.54 A) and a very small value for the well depth (0.01 kcal/mol). This results in a relatively weak repulsive wall, but will not translate well into a soft repulsion. You will probably want to use a *frcmod* file to reduce this radius to something more like 1.0 A. You may wish to modify other radii as well. Some useful information about this (for proteins) is in "Calibration of effective van der Waals atomic contact radii for proteins and peptides", by Iijima, Dunbar and Marshall, *Proteins: Str. Funct. Gen.* **2,** 330-339 (1987).

Note also that the *RSTAR* weight function (described below) can be used to modify *all* of the radii by a constant amount. For example, if you want the repulsive force to begin where the 6-12 potential crosses zero (rather than at the minimum), set *RSTAR* to (1/2)**(1/6), or about 0.8909.

Polarizable potentials

IPOL Inclusion of polarizabilities in the force field.

= 0 non polar calc (default).

= 1 turn on polarization calculation. Polarizabilities must be present in prmtop; see PARM.

= 2 Polarization calculation + read and use 3-body interaction definitions

Note: polarization is expensive and is currently recommended ONLY for investigation of polarization parameters.

N3B Number of three-body interactions to be defined; current maximum is 5.

NION Number of ions in the system.

AT1(I) The second atom in this 3-body interaction.

AT2(I) The third atom in this 3-body interaction.

ACON(I) The pre-exponential factor for this 3-body interaction.

BETA3(I) The beta value for this 3-body interaction.

GAMMA3(I) The gamma value for this 3-body interaction.

Frozen or restrained atoms

IBELLY Flag for belly type dynamics.

= 0 No belly run (default).

= 1 Belly run. A subset of the atoms in the system will be allowed to move, and the coordinates of the rest will be frozen. The *moving* atoms are specified in Group format at the end of all other input from file "mdin". Group input is described in the Appendix.

NTR Flag for restraining specified atoms in Cartesian space using a harmonic potential. Note: the restrained atoms are read in GROUP format after the numeric input from file "mdin" - see Appendices for GROUP. The coordinates are read in "restrt" format from the "refc" file (see NTRX, above).

= 0 No position restraints (default)

= 1 MD with restraint of specified atoms

Energy minimization

MAXCYC Maximum number of cycles of minimization. Default 1.

NCYC After NCYC cycles the method of minimization would be switched from steepest descent to conjugate gradient method. Default 10.

NTMIN Flag for the method of minimization.

= 0 Full conjugate gradient minimization. The first 10 cycles are steepest descent at the start of the run and after every nonbonded pairlist update.

= 1 For NCYC cycles the steepest descent method is used then conjugate gradient is switched on (default).

= 2 Only steepest descent method is used.

DX0 The initial step length. If the initial step length is big then the minimizer will try to leap the energy surface and sometimes the first few cycles will give a huge energy, however the minimizer is smart enough to adjust itself. Default 0.01.

DXM The maximum step length allowed. Default 0.5.

DRMS Convergence criterion for norm of the gradient of the energy. If the difference in the norm of the gradient for successive steps is within DRMS then optimization is complete. Default 1.0E-4 kcal/mole Å.

Molecular dynamics

NRUN Number of MD-runs of NSTLIM steps to be performed. Since the restart coordinates are written only at the end of each "run", it is sometimes advisable to break a long MD calculations into several "runs". The number of picoseconds of molecular dynamics is equal to the product of NRUN x NSTLIM x DT. Default 1.

NSTLIM Number of MD-steps per NRUN to be performed. Default 1.

NDFMIN Number of degrees of freedom that will be subtracted from the total number of degrees of freedom. If either NTCM or NSCM .NE. 0 then this option should be set equal to 6. Otherwise, NDFMIN should be 0. NDFMIN, NTCM, and NSCM are ignored for belly dynamics. Default 0.

NTCM Flag for the removal of translational and rotational motion at the beginning of the simulation.

= 0 The translational and rotational motion about the center of mass is not removed (default)

= 1 The above motion is removed one time at the beginning of the simulation.

NSCM Flag for the removal of translational and rotational motion at regular intervals. After every NSCM steps, translational and rotational motion will be removed. This flag is ignored for both belly and periodic simulations. Default 0.

INIT Flag for different starting procedures. If option NTX is less than 4, INIT should be equal to 3. If option NTX is greater than or equal to 4, this option should be equal to 4.

= 3 Generate starting velocities (NTX = 1 or 2). V(T-DT/2) is obtained by calculating force(T) unless TEMPI .gt. 1e-6, in which case the velocities are assigned from a Maxwellian at TEMPI K. Default.

= 4 Use input velocities (NTX >= 4). V(T-DT/2) is read from the input file "inpcrd".

T The time at the start (psec) this is for your own reference and is not critical. Start time is taken from the coordinate input file if IREST=1. Default 0.0.

DT The time step (psec). Recommended MAXIMUM is .002 if SHAKE is used, or .001 if it isn't. Note that for temperatures above 300K, the step size should be reduced since greater temperatures mean increased velocities and longer distance travelled between each force evaluation, which can lead to anomalously high energies and system blowup. Default 0.001.

Temperature regulation

TEMP0 Reference temperature at which the system is to be kept. Note that for temperatures above 300K, the step size should be reduced since increased distance travelled between evaluations can lead to SHAKE and other problems. Default 300.

TEMPI Initial temperature. For the initial dynamics run, (NTX .lt. 3) the velocities are assigned from a Maxwellian distribution at TEMPI K. If TEMPI = 0.0, the velocities will be calculated from the forces instead. TEMPI has no effect if NTX .gt. 3. Default 0.0.

IG The seed for the random number generator. The MD starting velocity is dependent on the random number generator seed if NTX .lt. 3 .and. TEMPI .ne. 0.0. Default

71277.

HEAT If ABS(HEAT) .GE. 1.0E-06, all the velocities are multiplied by HEAT. This only affects the initial velocities assigned at TEMPI. Default 0.0.

NTT Switch for temperature scaling.

Note that some of the following options are rather ad-hoc, and may not result in a thermodynamically relevant ensemble. However, they may be useful when using MD strictly to sample conformational space, such as with simulated annealing and nmr refinement – cases where the temperature of the system may be too unstable to use standard coupling scheme. In particular, option NTT=4 may be useful in such cases. Coupling schemes NTT=0,1 or 5 should be used when generating a thermodynamic ensemble is crucial.

= 0 Constant total energy classical dynamics. Velocities are never rescaled after the start of the simulation *except* at the end of every NSTLIM steps, when they will be rescaled to the target temperature if the current temperature deviates from TEMP0 by more than DTEMP. Default, but owing to the "hard" cutoff that lacks a switching function, energy will not be conserved unless the cutoff includes the whole system.

= 1 Constant temperature, using the Berendsen coupling algorithm (Berendsen et al. J. Chem. Phys., 81, 3684 (1984)). A single scaling factor is used for all atoms. This is good for small solutes, e.g. methane, but can result in cold solute for larger ones. See NTT=5.

= 2 Constant temperature, using Berendsen coupling algorithm. But only consider the solute temperature in determining the velocity scaling factor on each step. Could result in solvent atoms having very high temperature, and is not recommended for most cases.

= 3 Constant temperature, using Berendsen algorithm. But only rescale when the temperature deviates from TEMP0 by more than DTEMP. Single scaling factor.

= 4 Any time temperature deviates from TEMP0 by more than DTEMP, do one quick scale of the velocities to bring them back to TEMP0. Otherwise, do not scale.

= 5 Berendsen algorithm, use separate scaling factors for atoms of the solute and atoms of the solvent. This option is recommended as a replacement for NTT=1, and can help alleviate the "cold solute/hot solvent" problem.

<–1 Re-assign random velocities for all atoms whenever the current temperature deviates by more than DTEMP from TEMP0 (target temperature), and every ABS(NTT) steps. Velocities are assigned in a Maxwellian distribution.

=–1 Re-assign random velocities for all atoms whenever the current temperature deviates by more than DTEMP from TEMP0. Velocities are assigned in a Maxwellian distribution.

NOTE 1: When option (5) is chosen, both the solute and solvent coupling constants are used (TAUTP and TAUTS, respectively). In cases (1), (2), and (3), the single temperature coupling constant TAUTP is used for all atoms.

NOTE 2: If you are using NTT=2 or NTT=5, you can specify the variable ISOLVP to redefine the last_solute_atom pointer. See below.

ISOLVP Last-solute-atom pointer to be used with temperature scaling, when separate scaling of solute and solvent atoms has been requested (NTT=2 or NTT=5).

By default (ISOLVP=0), the last non-TIP3P water molecule in the system is generally taken as the last_solute_atom. For example, the counterions are considered part of the "solute" by default. You could re-define the counterions to be part of the solvent by setting ISOLVP.

DTEMP For NTT = 0, if the difference between the system temperature and TEMP0 is more than DTEMP at the end of each run, the velocities will be linearly scaled to TEMP0. Default 0.0.

TAUTP Time constant for heat bath coupling for the SOLUTE. Default 0.2.

NOTE: Users familiar with AMBER 3.0 or 3.0a will find a significant difference in the behavior of these options since the coupling is now 20.455 times looser than amber3a and previous versions.

Generally, values for TAUTP should be in the range of 0.1-0.4ps, with a smaller value providing tighter coupling to the heat bath, therefore a less natural trajectory. Values of TAUTP less than 0.1 result in smaller fluctuation in kinetic energy, but larger fluctuation in the total energy. Values much larger than the length of the simulation result in a return to constant energy conditions.

TAUTS Time constant for the heat bath coupling for the SOLVENT (NTT=5). Default 0.2.

VLIMIT If .ne. 0.0, then any component of the velocity that is greater than abs(VLIMIT) will be reduced to VLIMIT (preserving the sign). This can be used to avoid occasional instabilities in molecular dynamics runs. VLIMIT should be set (if at all) to a value like 20., which is well above the most probable velocity in a Maxwell-Boltzmann distribution at room temperature. A warning message will be printed whenever the velocities are modified. Runs that have more than a few such warnings should be carefully examined. Default 0.0.

PEACS temperature algorithm

PEACS is a searching technique wherein the trajectory follows a constant energy contour. See Schaik et al, J. Comp. Aided Mol. Design, 6, 97 (1992). PEACS can only be carried out using MD (not minimization).

TAUV0 Defines the rate constant for lowering the temperature if a PEACS constant potential energy search is to be carried out.

- = 0.0 No PEACS search will be carried out (default).
- > 0.0 TAUV0 defines the rate at which the target energy contour is annealed down.

TAUV When a PEACS search is being carried out (TAUV0 > 0.0), TAUV defines the coupling constant between the target energy contour and the contour being followed. Default 0.1.

VZERO Defines the value of the initial potential energy contour to be followed if a PEACS search is being carried out. If VZERO is specified as 0.0 (default), the initial energy of the system will be used.

Pressure regulation

Pressure regulation only applies when Constant Pressure periodic boundary conditions are used (NTB = 2).

NTP Flag for constant pressure dynamics. This option MUST be set to 1 or 2 when Constant Pressure periodic boundary conditions are used (NTB = 2).

- = 0 Used with NTB not = 2 (default)
- = 1 md with isotropic position scaling
- = 2 md with anisotropic diagonal (x-,y-,z-) position scaling

PRES0 Reference pressure (in units of bars, where 1 bar ˜ 1 atm) at which the system is maintained (when NTP > 0). Default 1.0.

COMP compressibility of the system when NTP > 0. The units are in 1.0E-06/bar; a value of 44.6 (default) is appropriate for water.

TAUP Pressure relaxation time when NTP > 0. The recommended value is between 0.1 and 1.0 psec**-1. Default 0.2.

NPSCAL Method for pressure scaling of the atomic coordinates. Pressure scaling means changing the size of the box to match the target pressure, which involves scaling the positions of the contents of the box so that they are proportionally distributed within the new box size. For example if the box contracts, the system as a whole must be contracted to avoid overlaps at the periodic boundary, and if the box expands, the system is expanded uniformly to fill the vacuum at the edge(s).

= 0 Atom scaling. All atoms are independently moved according to the scale factor. This causes some degree of compression or stretching of bonds. Default.

= 1 Molecule scaling. Bonded groups of atoms (molecules) are moved as units. This is intended to avoid changing bond lengths as a side effect of pressure scaling, but may disrupt coordination of extended contacting molecules such as nucleic acid strands.

SHAKE bond length constraints

NTC Flag for SHAKE to perform bond length constraints. SHAKE is used in the TIP water potentials and keeps waters rigid so that the hydrogens (which have 0 vdw radius) do not extend beyond the vdw sphere defined for the oxygens. For energy minimization, no SHAKE should be necessary, unless electrostatic energies blow up (negatively) in a water bath. If both problems affect a minimization with periodic boundary conditions, increasing the box dimension at the end of the prmtop file by 0.2-0.8 Angstroms may help. The SHAKE option should be used for most MD calculations. The size of the MD timestep is determined by the fastest motions in the system. SHAKE removes the bond stretching freedom, which is the fastest motion, and consequently allows a larger timestep to be used.

= 1 SHAKE is not performed (default)

= 2 bonds involving hydrogen are constrained

= 3 all bonds are constrained

TOL Relative geometrical tolerance for coordinate resetting in shake. Recommended maximum: <0.0005 Angstrom Default 0.0005.

Special water treatment

IMGSLT Controls Solute-Solvent imaging in periodic boundary calculations.

= 0 Solute is imaged with solvent. Solute is allowed to interact with solvent images (if they are within CUT). Default.

= 1 No Solute-Solvent imaging. Solute does not see image solvent. This assumes that the solute is centered in the periodic system, and is not free to migrate. Do not use this with mobile solutes. This option is mainly useful for large solutes.

IFTRES Flag to remove the nonbonded cutoff from the solute in periodic boundary simulations.

= 0 ALL intramolecular solute - solute nonbonded interactions are calculated regardless of whether the interatomic distance is greater than the nonbonded cutoff. Solute-solute imaging is turned off.

= 1 Nonbondeds are evaluated normally. Default.

NOTE: For simulations of highly charged solutes in a water bath, it can be useful to calculate ALL solute - solute nonbonded interactions in order to reduce electrostatic problems. This is especially important for highly charged systems like nucleic acids. Note that this option is intended for small solutes, and will generate many more nonbonded pairs than the normal method if the solute is large. Counterions added in EDIT are considered part of the solute.

JFASTW Fast water definition flag. By default, the system is searched for TIP3P waters, and special fast routines are used for these molecules. There are two types of fast routines specific to TIP3P water: 1) A faster, analytic SHAKE algorithm for 3-point water; 2) A faster routine to calculate non-bonded TIP3P-TIP3P water interactions. In normal operation, the program defaults will be acceptable. However, in rare instances (e.g. for debugging purposes, or when the user has redefined the definition of a TIP3P water), one may wish to inhibit the use of these fast routines and/or redefine the default definition used in Amber to define TIP3P waters. This option makes this possible.

= 0 Normal operation. The default AMBER definition of TIP3P water is used, and the fast water routines are used where appropriate.

= 1 Use the fast water routines for SHAKE and non-bonds, but redefine the names the program uses to recognize TIP3P waters. The redefinition names are provided below.

= 2 Use the fast water routine for SHAKE. Do not use the fast water routine for non-bonds.

= 3 Use the fast water routine for SHAKE. Do not use the fast water routine for non-bonds. Redefine the names the program uses to recognize TIP3P waters. The redefinition names are provided below, after the normal

= 4 Do not use fast water routines for either SHAKE or non-bonds.

The following variables allow redefinition of the default residue and atom names used by the program to determine which residues are TIP3P waters. Except in unusual circumstances, the default water names should be acceptable.

WATNAM The residue name the program expects for TIP3P waters. Default 'WAT '.

OWTNM The atom name the program expects for the oxygen of TIP3P wat. Default 'O '.

HWTNM1 The atom name the program expects for the 1st H of TIP3P wat. Default 'H1 '.

HWTNM2 The atom name the program expects for the 2nd H of TIP3P wat. Default 'H2 '.

Water "cap"

IVCAP Flag to control Cap Option. The Cap refers to a spherical portion of water centered on a point in the solute and restrained by a soft half-harmonic potential. Caps are constructed using EDIT.

= 0 Cap will be in effect if it is passed from the the parm module (default)

= 1 Cap will be activated except that the Cap atom pointer will be modified

= 2 Cap will be inactivated

MATCAP The Cap atom pointer. This is the last Non-Cap atom number. If IVCAP = 1 then the pointer passed from the PARM module will be overwritten by this number. PARM passes the NATCAP parameter which is replaced by the value in MATCAP. Default 0.

FCAP The Force Constant for the Cap restraint potential. A value of 0.0 for FCAP will result in the default force constant of 1.5.

NMR refinement options

ISCALE Number of additional variables to optimize beyond the 3N structural parameters. (Default = 0). At present, no options other than ISCALE = 0 are supported (see code).

NOESKP The NOESY volumes will only be evaluated if mod(nstep, noeskp) = 0; otherwise the last computed values for intensities and derivatives will be used. (default = 1, i.e. evaluate volumes at every step)

IPNLTY

= 1 the program will minimize the sum of the absolute values of the errors; this is akin to minimizing the crystallographic R-factor (default).

= 2 the program will optimize the sum of the squares of the errors.

= 3 For NOESY intensities, the penalty will be of the form

$awt\ [I_c^{(1/6)} - I_o^{(1/6)}]^2$.

Chemical shift penalties will be as for *ipnlty=1*.

MXSUB Maximum number of submolecules that will be used. This is used to determine how much space to allocate for the NOESY calculations. Default 1.

SCALM "Mass" for the additional scaling parameters. Right now they are restricted to all have the same value. The larger this value, the slower these extra variables will respond to their environment. Default 100 amu.

PENCUT In the summaries of the constraint deviations, entries will only be made if the penalty for that term is greater than PENCUT. Default 0.1.

TAUSW For noesy volume calculations (*NMRMAX = 2 or 3*), intensities with mixing times less that TAUSW (in seconds) will be computed using perturbation theory, whereas those greater than TAUSW will use a more exact theory. See the theory section (below) for details. To always use the "exact" intensities and derivatives, set TAUSW = 0.0; to always use perturbation theory, set TAUSW to a value larger than the largest mixing time in the input. Default is TAUSW of 0.1 second, which should work pretty well for most systems.

Particle Mesh Ewald

This is an experimental option and is not fully tested!

IEWALD Turns on the Particle Mesh Ewald (PME) method. PME [12] is a fast implementation of the Ewald summation method [13] for calculating the full electrostatic energy of a unit cell (periodic box) in a macroscopic lattice of repeating images. As implemented, the PME in AMBER bypasses the standard pairlist creation and nonbonded energy and force evaluation, calling special PME functions to calculate the Lennard-Jones and electrostatic interactions. The PME method is fast since the reciprocal space Ewald sums are B-spline interpolated on a grid and since the convolutions necessary to evaluate the sums are calculated via fast Fourier transforms. Note that the accuracy of the PME is related to the density of the charge grid (NFFTX, NFFTY, and NFFTZ), the spline interpolation order (SPLINE_ORDER), and the direct sum tolerance (DSUM_TOL); see the descriptions below for more information. This is an *experimental* option since the standard AMBER routines are bypassed, the code has not been verified on all architectures, the code has not been verfied to work properly under all possible sander input options, and finally since the method is still under active development.

= 0 PME is turned off. This is the default option.

= 1 PME is turned on. This requires extra input in order to control the calculation. This input, consisting of three lines of free format numerical input (*not* namelist input!), is described below and *must* be placed in the input file just *after* the end of this namelist (`&cntrl &end`) [or after the formatted input] and *before* any weight change information (`&wt &end`) described in the next section and/or before the group input information.

Special input: only processed when IEWALD = 1.

Line 1 The unit cell parameters: BOXX, BOXY, BOXZ, ALPHA, BETA and GAMMA. All are double precision free format input.

BOXX, BOXY, BOXZ
The PME unit cell (periodic box) lengths (Å) in each dimension. This information must be specified and *overrides* the box information specified in the parm file. When NTX = 7 (used to read in the velocity and box information upon restart) this information is read, but ignored, and the unit cell information is obtained from the restart file.

ALPHA, BETA, GAMMA
The PME unit cell angles (in degrees). Unlike standard AMBER, PME allows non-rectangular boxes. [A rectangular box has angles of 90.0.]

---

[12] (a) Darden, T.A.; York, D. and Pedersen L. "Particle Mesh Ewald: An N log(N) method for Ewald sums in large systems." *J. Chem. Phys.* **98**, 10089 (1993). (b) Darden, T.A. (under preparation).

[13] Ewald, P. (1921) *Ann. Phys. (Leipzig)* **64**, 253.

When NTX = 7, ALPHA, BETA and GAMMA are obtained from the restart file. If the restart file doesn't contain these values (for example if restarting from standard, non PME, periodic boundary conditions), ALPHA, BETA and GAMMA will default to 90.0 degrees.

Line 2 Interpolation and control information: NFFT1, NFFT2, NFFT3, SPLINE_ORDER, ISCHARGED, VERBOSE, EXACT_EWALD. All are integer free format input.

NFFTX, NFFTY, NFFTZ

These give the size of the charge grid (upon which the reciprocal sums are interpolated) in each dimension. Higher values lead to higher accuracy (when the DSUM_TOL is also lowered) but considerably slow the calculation. Generally it has been found that reasonable results are obtained when NFFTX, NFFTY and NFFTZ are approximately equal to BOXX, BOXY, and BOXZ respectively, leading to a grid spacing (BOXX/NFFTX, etc) of 1.0 Å. Significant performance enhancement in the calculation of the fast Fourier transform is obtained by having each of the integer NFFTX, NFFTY and NFFTZ values be a *product of powers* of 2, 3, and 5.

SPLINE_ORDER

The order of the B-spline interpolation. The higher the order, the better the accuracy (unless the charge grid is too coarse). The minimum order is 3. An order of 4 implies a cubic spline approximation which is a good standard value. Note that the cost of the PME goes as roughly the order to the third power.

ISCHARGED

Standard use is to have ISCHARGED = 0 which forces neutralization of the unit cell by removal of the average charge over the system at the beginning of the run. [This is necessary due to the roundoff error associated with the parm derived charges (upon reading in a parameter file, the sum of the charges for a neutral system does not sum to zero).] When ISCHARGED = 1, the unit cell is not neutralized. Technically, the Ewald summation method is not correct when a non-neutral system is used (energy will change, independent of the direct sum tolerance, but the forces are still correctly determined). However, the method has been applied for non-neutral systems and may be useful for equilibrating systems in the absence of counterions, for example.

VERBOSE

Standard use is to have VERBOSE = 0. Turning VERBOSE = 1 leads to voluminous output of information about the PME run.

EXACT_EWALD

Standard use is to have EXACT_EWALD = 0 which turns on the particle mesh ewald (PME) method. When EXACT_EWALD = 1, instead of the

approximate, interpolated PME, an *exact* Ewald calculation is run. The exact Ewald summation is present to serve as an accuaracy check allowing users to determine if the PME grid spacing, order and direct sum tolerance lead to acceptable results. Although the cost of the exact Ewald method formally increases with system size at a much higher rate than the PME, it is faster for small numbers of atoms (< 500). For larger, macromolecular systems, with > 500 atoms, the PME method is significantly faster.

Line 3 The direct sum tolerance: DSUM_TOL. This is a double precision free format value.

DSUM_TOL

This relates to the width of the direct sum part of the Ewald sum, requiring that the value of the direct sum at the Lennard-Jones cutoff value (specified in CUT as during standard dynamics) be less that DSUM_TOL. In practice it has been found that the relative error in the Ewald forces (RMS) due to cutting off the direct sum at CUT is between 10.0 and 50.0 times DSUM_TOL. Standard values for DSUM_TOL are in the range of 0.000001 to 0.00001 leading to estimated RMS deviation force errors of 0.00001 to 0.0005.

Special notes about the PME (when IEWALD = 1*):*

(1) *Imaging:* The PME method, as implemented, does not image residues in the same manner as standard AMBER. [14] Standard AMBER, when periodic boundary conditions are applied, will translate all the atoms in a given residue back into the box if the first atom in that residue is outside the box. [An exception to this case – when running dynamics and running without a belly (IBELLY = 0) and without position constraints (NTR = 0) – is that translation of the entire solute will occur if the center of geometry of the solute is outside the box.] Technically, the PME does not need to explicitly image (translate) the atoms since in the calculations imaging to the unit cell is done implicitly.

(2) *Pressure:* Correct calculation of the pressure requires that each solvent molecule be represented as a separate molecule in the topology file. This is the default behavior when the BOX or SOL options are used in EDIT. A routine is called to check if this is indeed the case when using standard water residues (named WAT) and will print a warning message if inconsistencies are detected.

(3) *Quick Error estimate:* During a PME run, whenever the energy summary is printed, an estimate of the RMS force error is also printed.

(4) *Pairlist:* The Lennard-Jones interactions (stored in a pairlist updated every NSNB steps) are calculated using an atom-based cutoff when using PME rather than the residue (charge group) based cutoff applied in standard AMBER. This is appropriate since residue-based pairlisting is only relevant to avoid splitting the dipole when electrostatic interactions are involved.

[14] A supplementary program called rdparm has been provided which can convert trajectory (mdcrd) files from PME imaging to standard imaging.

(5) *Forces:* Although the PME method does rigourously conserve energies-- assuming a high enough level of accuracy obtained through small charge grids (NFFT1, NFFT2, NFFT3), low DSUM_TOL, and high level interpolation (ORDER)-- the forces are not conserved. In test cases, this led to problems whereby a directed force component appeared when the pairlist was not updated very frequently. In order to circumvent this, any net force component is now zeroed every step (in subroutine accumforce). This does not seem to effect energy conservation. In the future, this will be an input option (e.g. to turn on or off net force zeroing).

(6) *Compatibility:* The following options are not compatible with the Ewald method:

CUT2ND: cannot have secondary cutoffs on the van der Waals terms.
IPOL = 1: polarization is not supported
IPRR and IPWR: cannot read/write pairlists to a file for later reuse.
IFTRES = 1: cannot calculate all solute – solute interactions.
IMGSLT = 1: not compatible
IFBOX = 2: cannot work with truncated octahedral boundary conditions.

For more information about the application of the PME method, please refer to the following references:

York, D.M.; Darden, T.A. and Pedersen, L.G. "The effect of long-range electrostatic interactions in simulations of macromolecular crystals: A comparison of the Ewald and truncated list methods." *J. Chem. Phys.* **99**(10), 8345 (1993).

York, D.M.; Wlodawer, A.; Pedersen, L.G. and Darden, T.A. "Atomic-level accuracy in simulations of large protein crystals." *Proc. Natl. Acad. Sci.* **91**, 8715 (1994).

For more general information about the Ewald method, please see:

Allen, M.P. and Tildesley, D.J. *Computer Simulation of Liquids* Oxford (1987).

Valleau, J.P.; Whittington, S.G. "A Guide to Monte Carlo for Statistical Mechanics: 1. Highways." in *Statistical Mechanics. A. A Modern Theoretical Chemistry.* B.J. Berne: New York, 1977; pp 137-168.

SECTION TWO: Weight change information

This section of information is read (*if NMRMAX > 0*) as a series of namelist specifications, with name "`&wt`". This namelist is read repeatedly until a namelist `&wt` statement is found with TYPE=END.

Overview of weight change variables

variable	*description*
TYPE	Defines quantity being varied; valid options are listed below.
ISTEP1,ISTEP2	This change is applied over steps/iterations ISTEP1 through ISTEP2. If ISTEP2 = 0, this change will remain in effect from step ISTEP1 to the end of the run at a value of VALUE1 (VALUE2 is ignored in this case). *(default= both 0)*
VALUE1,VALUE2	Values of the change corresponding to ISTEP1 and ISTEP2, respectively. If ISTEP2=0, the change is fixed at VALUE1 for the remainder of the run, once step ISTEP1 is reached.
IINC IMULT	If IINC > 0, then the change is applied as a step function, with IINC steps/iterations between each change in the target VALUE (ignored if ISTEP2=0). If IINC =0, the change is done continuously. *(default=0)* If IMULT=0, then the change will linearly interpolated from VALUE1 to VALUE2 as the step number increases from ISTEP1 to ISTEP2. *(default)* If IMULT=1, then the change will be effected by a series of multiplicative scalings, using a single factor, R, for all scalings. i.e. VALUE2 = (R**INCREMENTS) * VALUE1. INCREMENTS is the number of times the target value changes, which is determined by ISTEP1, ISTEP2, and IINC.

The remainder of this section describes the options for the TYPE parameter. For a few types of cards, the meanings of the other variables differ from that described above; such differences are noted below. Valid Options for TYPE (you must use uppercase) are:

BOND	Varies the relative weighting of bond energy terms.
ANGLE	Varies the relative weighting of valence angle energy terms.
TORSION	Varies the relative weighting of torsion (and J-coupling) energy terms. Note that any restraints defined in the input to the PARM program are included in the above. Improper torsions are handled separately (IMPROP).
IMPROP	Varies the relative weighting of the "improper" torsional terms. These are not included in TORSION.
VDW	Varies the relative weighting of van der Waals energy terms. This is equivalent to changing the well depth (epsilon) by the given factor.
HB	Varies the relative weighting of hydrogen-bonding energy terms.
ELEC	Varies the relative weighting of electrostatic energy terms.
NB	Varies the relative weights of the non-bonded (VDW, HB, and ELEC) terms.
ATTRACT	Varies the relative weights of the attractive parts of the van der waals and h-bond terms.
REPULSE	Varies the relative weights of the repulsive parts of the van der waals and h-bond terms.
RSTAR	Varies the effective van der Waals radii for the van der Waals (VDW) interactions by the given factor. Note that this is done by changing the relative attractive and repulsive coefficients, so ATTRACT/REPULSE should not be used over the same step range as RSTAR.
SOFTR	Varies the soft-repulsion non-bond force constant. Has no effect if ISFTRP.LE.0.
INTERN	Varies the relative weights of the BOND, ANGLE and TORSION terms. "Improper" torsions (IMPROP) must be varied separately.
ALL	Varies the relative weights of all the energy terms above (BOND, ANGLE, TORSION, IMPROP, VDW, HB, and ELEC; does not affect RSTAR).
REST	Varies the relative weights of *all* the NMR restraint energy terms.
RESTS	Varies the weights of the "short-range" NMR restraints. Short- range restraints are defined by the SHORT instruction (see below).
RESTL	Varies the weights of any NMR restraints which are not defined as "short range" by the SHORT instruction (see below). When no SHORT instruction is given, RESTL is equivalent to REST.
NOESY	Varies the overall weight for NOESY volume restraints. Note that this value multiplies the individual weights read into the "awt" array. (Only if NMRMAX=2 or 3; see Section 4 below).
SHIFTS	Varies the overall weight for chemical shift restraints. Note that this value multiplies the individual weights read into the "wt" array. (Only if NMRMAX=2 or 3; see section 4 below).

SHORT

Defines the short-range restraints. For this instruction, ISTEP1, ISTEP2, VALUE1, and VALUE2 have different meanings. A short-range restraint can be defined in two ways.

(1) If the residues containing each pair of bonded atoms comprising the restraint are close enough in the primary sequence:

$$\text{ISTEP1} \le \text{ABS(delta\_residue)} \le \text{ISTEP2},$$

where delta_residue is the difference in the numbers of the residues containing the pair of bonded atoms.

(2) If the distances between each pair of bonded atoms in the restraint fall within a prescribed range:

$$\text{VALUE1} \le \text{distance} \le \text{VALUE2}.$$

Only one SHORT command can be issued, and the values of ISTEP1, ISTEP2, VALUE1, and VALUE2 remain fixed throughout the run. However, if IINC>0, then the short-range interaction list will be re-evaluated every IINC steps.

TEMP0

Varies the target temperature TEMP0.

TAUTP

Varies the coupling parameter, TAUTP, used in temperature scaling when temperature coupling options NTT=1,2 or 3 are used.

CUT

Varies the non-bonded cutoff distance.

NSTEP0

If present, this instruction will reset the initial value of the step counter (against which ISTEP1/ISTEP2 and NSTEP1/NSTEP2 are compared) to the value ISTEP1. An NSTEP0 instruction only has an effect at the beginning of a run. For this card (only) ISTEP2, VALUE1, VALUE2 and IINC are ignored. If this card is omitted, NSTEP0 = 0. This card can be useful for simulation restarts, where NSTEP0 is set to the final step on the previous run.

STPMLT

If present, the NMR step counter will be changed in increments of STPMLT for each actual dynamics step. For this card, only VALUE1 is read. ISTEP1, ISTEP2, VALUE2, IINC, and IMULT are ignored. Default = 1.0.

DISAVE

ANGAVE

TORAVE

If present, then by default time-averaged values (rather than instantaneous values) for the appropriate set of restraints will be used. DISAVE controls distance data, ANGAVE controls angle data, TORAVE controls torsion data.

See below for the functional form used in generating time-averaged data.

For these cards: VALUE1 = τ (characteristic time for exponential decay)
VALUE2 = POWER (power used in averaging; the nearest integer of value2 is used)

Note that the range (ISTEP1→ISTEP2) applies only to TAU; The value of POWER is not changed by subsequent cards with the same ITYPE field, and time-averaging will always be turned on for the entire run if one of these cards appears.

Note also that, due to the way that the time averaged internals are calculated, changing τ at any time after the start of the run will only affect the relative weighting of steps occurring after the change in τ.

Separate values for τ and POWER are used for bond, angle, and torsion averaging.

The default value of τ (if it is 0.0 here) is 1.0D+6, which results in no exponential decay weighting. Any value of $\tau \geq$ 1.D+6 will result in no exponential decay.
If DISAVE,ANGAVE, or TORAVE is chosen, one can still force use of an instantaneous value for specific restraints of the particular type (bond, angle, or torsion) by setting the IFNTYP field to "1" when the restraint is defined (IFNTYP is defined in section 3 below).
If time-averaging for a particular class of restraints is being performed, all restraints of that class that are being averaged (that is, all restraints of that class except those for which IFNTYP=1) *must* have the same values of NSTEP1 and NSTEP2 (NSTEP1 and NSTEP2 are defined below).

(For these cards, IINC and IMULT are ignored)

See the discussion of time-averaged restraints following the input descriptions.

DISAVI

ANGAVI

TORAVI

ISTEP1: Ignored.

ISTEP2: Sets IDMPAV. If IDMPAV > 0, *and* a dump file has been specified (DUMPAVE is set in the file redirection section below), then the time-averaged values of the restraints will be written every IDMPAV steps. Only one value of IDMPAV can be set (corresponding to the first DISAVI/ANGAVI/TORAVI card with ISTEP2 > 0), and *all* restraints (even those with IFNTYP=1) will be "dumped" to this file every IDMPAV steps. The values reported reflect the current value of τ.

VALUE1: The integral which gives the time-averaged values is undefined for the first step. By default, for each time-averaged internal, the integral is assigned the current value of the internal on the first step. If VALUE1≠0, this initial value of internal r is reset as follows:

```
-1000. <  VALUE1 <    1000.:  Initial value = r_initial + VALUE
          VALUE1 <= -1000.:  Initial value = r_target + 1000.
 1000. <= VALUE1           :  Initial value = r_target - 1000.
```

r_target is the target value of the internal, given by R2+R3 (or just R3, if R2 is 0). VALUE1 is in angstroms for bonds, in degrees for angles.

VALUE2: This field can be used to set the value of τ used in calculating the time-averaged values of the internal restraints reported at the end of a simulation (if LISTOUT is specified in the redirection section below). By default, no exponential decay weighting is used in calculating the final reported values, regardless of what value of τ was used during the simulation. If VALUE2>0, then τ = VALUE2 will be used in calculating these final reported averages. Note that the value of VALUE2 = τ specified here only affects the reported averaged values in at the end of a simulation. It does not affect the time-averaged values used during the simulation (those are changed by the VALUE1 field of DISAVE, ANGAVE and TORAVE instructions).

IINC: If IINC = 0, then forces for the class of time-averaged restraints will be calculated exactly as (dE/dr_ave) (dr_ave/dx). If IINC = 1, then then forces for the class of time-averaged restraints will be calculated as (dE/dr_ave) (dr(t)/dx). Note that this latter method results in a non-conservative force, and does not integrate to

	a standard form. But this latter formulation helps avoid the large forces due to the (1+IPOWER) term in the exact derivative calculation--and may avert instabilities in the molecular dynamics trajectory for some systems. See the discussion of time-averaged restraints following the input description. Note that the DISAVI, ANGAVI, and TORAVI instructions will have no affect unless the corresponding time average request card (DISAVE, ANGAVE or TORAVE, respectively) is also present. (For these cards, ISTEP1 and IMULT are ignored).
#	If formatted input is being read (&formwt was specified), any line which starts with a pound symbol (#) is considered a comment line, and will be skipped.
END	END of this section.

NOTES:

(1) All weights are relative to a default of 1.0 in the standard force field.

(2) Weights are not cumulative.

(3) For any range where the weight of a term is not modified by the above, the weight reverts to 1.0. For any range where TEMP0, SOFTR or CUTOFF is not specified, the value of the relevant constant is set to that specified in the input file.

(4) If a weight is set to 0.0, it is set internally to 1.0D-7. This can be overridden by setting the weight to a negative number. In this case, a weight of exactly 0.0 will be used. *However,* if any weight is set to exactly 0.0, it cannot be changed again during this run of the program.

(5) If two (or more) cards change a particular weight over the same range, the weight given on the last applicable card will be the one used.

(6) Once any weight change for which NSTEP2=0 becomes active (i.e. one which will be effective for the remainder of the run), the weight of this term cannot be further modified by other instructions.

(7) Changes to RSTAR result in exponential weighting changes to the attractive and repulsive terms (proportional to the scale factor**6 and **12, respectively). For this reason, scaling RSTAR to a very small value (e.g. ≤0.1) may result in a zeroing-out of the vdw term.

SECTION THREE: File redirection commands

Input/output redirection information can be read as described here. The inclusion of these cards is optional. By default (if not redirected here), all input is taken from the standard input file. Redirection cards, if provided, must follow the end of the SECTION TWO input. Redirection card input is terminated by the first non-blank line which does not start with a recognized redirection TYPE (e.g. LISTIN, LISTOUT, etc.).

The format of the redirection cards is

TYPE = filename

where TYPE is any valid redirection keyword (see below), and filename is any character string. The equals sign ("=") is required, and TYPE must be given in *uppercase* letters.

Valid redirection keywords are:

LISTIN	An output listing of the restraints which have been read, and their deviations from the target distances *before* the simulation has been run. By default, this listing is not printed. If POUT is used for the filename, these deviations will be printed in the normal output file.
LISTOUT	An output listing of the restraints which have been read, and their deviations from the target distances *after* the simulation has finished. By default, this listing is not printed. If POUT is used for the filename, these deviations will be printed in the normal output file.
DISANG	The file from which the distance and angle restraint information described below (Section 3) will be read.
NOESY	File from which NOESY volume information (Section 4), if any, will be read.
SHIFTS	File from which chemical shift information (Section 5), if any, will be read.
DUMPAVE	File to which the time-averaged values of all restraints will be written, if DISAVI / ANGAVI / TORAVI has been used to set IDMPAV≠0. If either IDMPAV has not been set, or DUMPAVE is not specified, this file will not be written.

SECTION FOUR: Distance, angle and torsional (and J-coupling) restraints

The input/output redirection cards (if any) are followed by the distance and angle restraints, which are read if *nmrmax* > 0. Namelist rst ("`&rst`") contains the following variables; it is read repeatedly until a namelist `&rst` statement is found with IAT(1)=0.

In many cases, the user will not prepare this section of the input by hand, but will use the auxiliary programs *makeRST* and *makeCHIR_CONS* to prepare input from simpler files. See the **Auxiliary programs** section below for details.

Variables in the `&rst` namelist:

IAT(1)→IAT(4) *If IRESID = 0 (normal operation):*

The atoms defining the restraint. If IAT(3) ≤0, this is a distance restraint. If IAT(4) ≤0, this is an angle restraint. Otherwise, this is a torsional (or J-coupling, if desired) restraint.

If this is a distance restraint, and IAT1 <0, then a group of atoms is defined below, and the coordinate-averaged position of this group will be used in place of the coordinates of atom 1 [IAT(1)]. Similarly, if IAT(2) < 0, a group of atoms will be defined below whose coordinate-averaged position will be used in place of the coordinates for atom 2 [IAT(2)].

If IRESID=1:

IAT(1) → IAT(4) point to the numbers of the *residues* containing the atoms comprising the internal. Residue numbers are the absolute numbers in the entire system. In this case, the variables ATNAM(1) → ATNAM(4) must be specified, and give the character names of the desired atoms within the respective residues.

If IAT(1) < 0 or IAT(2) < 0, then group input will still be read in place of the corresponding atom, as described below.

Defaults for IAT(1)→IAT(4) are 0.

ATNAM If IRESID = 1, then the character names of the atoms defining the internal are contained in ATNAM(1)→ATNAM(4). Residue IAT(1) is searched for atom name ATNAM(1); residue IAT(2) is searched for atom name ATNAM(2); etc. On machines using the portable namelist code, the form is atnam(1)='AT1',atnam(2)='AT2' etc, otherwise the form atnam='AT1','AT2' etc can be used.

Defaults for ATNAM(1)→ATNAM(4) are ' '.

IRESID — Indicates whether IAT(I) points to an atom # or a residue #. See descriptions of IAT() and ATNAM() above.

Default = 0.

NSTEP1

NSTEP2 — This restraint is applied for steps/iterations NSTEP1 through NSTEP2. If NSTEP2 = 0, the restraint will be applied from NSTEP1 through the end of the run. Note that the first step/iteration is considered step zero (0).

Defaults for NSTEP1, NSTEP2 are both 0.

IRSTYP — Normally, the restraint target values defined below (R1→R4) are used directly. If IRSTYP = 1, the values given for R1→R4 define relative displacements from the current value (value determined from the starting coordinates) of the restrained internal. For example, if IRSTYP=1, the current value of a restrained distance is 1.25, and R1 (below) is -0.20, then a value of R1=1.05 will be used.

Default is IRSTYP=0.

IFVARI — If IFVARI > 0, then the force constants/positions of the restraint will vary with step number. Otherwise, they are constant throughout the run. If IFVARI >0, then the values R1A→R4A, RK2A, and RK3A must be specified (see below).

Default is IFVARI=0.

NINC — If IFVARI > and NINC > 0, then the change in the target values of of R1→R4 and K2,K3 is applied as a step function, with NINC steps/ iterations between each change in the target values. If NINC = 0, the change is effected continuously (at every step).

Default for NINC is the value assigned to NINC in the most recent namelist where NINC was specified. If NINC has not been specified in any namelist, it defaults to 0.

IMULT — If IMULT=0, and the values of force constants RK2 and RK3 are changing with step number, then the changes in the force constants will be linearly interpolated from rk2→rk2a and rk3→rk3a as the step number changes.

If IMULT=1 and the force constants are changing with step number, then the changes in the force constants will be effected by a series of multiplicative scalings, using a single factor, R, for all scalings. *i.e.*

rk2a = R**INCREMENTS * rk2
rk3a = R**INCREMENTS * rk3.

INCREMENTS is the number of times the target value changes, which is determined by NSTEP1, NSTEP2, and NINC.

Default for IMULT is the value assigned to IMULT in the most recent namelist where IMULT was specified. If IMULT has not been specified in any namelist, it defaults to 0.

R1→R4

RK2,RK3

R1A→R4A

RK2A,RK3A — The restraint is a well with a square bottom with parabolic sides out to a defined distance, and then linear sides beyond that. Force constants are in units of

kcal/mol. If R is the value of the restraint in question:

R < r1	Linear, with the slope of the "left-hand" parabola at the point R=r1.
r1 <= R < r2	Parabolic, with force constant k2. E=0 at R=r2.
r2 <= R < r3	E = 0.
r3 <= R < r4	Parabolic, with force constant K3. E=0 at R=r3.
r4 <= R	Linear, with the slope of the "right-hand" parabola at the point R=r4.

For torsional restraints, the value of the torsion is translated by +-n*360, if necessary, so that it falls closest to the mean of r2 and r3.

Specified distances are in Angstroms. Specified angles are in degrees. Force constants for distances are in kcal/mol-A^2. Force constants for angles are in kcal/mol-rad^2. (Note that angle positions are specified in degrees, but force constants are in radians, consistent with typical reporting procedures in the literature).

IFVARI = 0 The values of r1→r4, rk2, and rk3 will remain constant throughout the run.

IFVARI > 0 The values r1a, r2a, r3a, r4a, r2ka and r3ka are also used. These variables are defined as for r1→r4 and rk2, rk3, but correspond to the values appropriate for NSTEP = NSTEP2: e.g., if IVARI >0, then the value of r1 will vary between NSTEP1 and NSTEP2, so that, e.g. r1(NSTEP1) = r1 and r1(NSTEP2) = r1a. Note that your *must* specify an explicit value for *nstep1* and *nstep2* if you use this option.

Defaults for r1→r4,rk2,rk3,r1a→r4a,rk2a and rk3a are the values assigned to them in the most recent namelist where they were specified. They should always be specified in the first `&rst` *namelist.*

(IGR1(i),i=1→200)

If IAT(1) < 0 and IAT(3)=IAT(4)=0, then IGR1() gives the atoms defining the group whose coordinate averaged position is used to define "atom 1" in a distance restraint. If IRESID = 0, absolute atom numbers are specified by the elements of IGR1(). If IRESID = 1, then IGR1(I) specifies the number of the residue containing atom I, and the name of atom I must be specified using GRNAM1(I). A maximum of 200 atoms are allowed in any group. Only specify those atoms which are needed.

RJCOEF(1)→RJCOEF(3)

By default, 4-atom sequences specify torsional restraints. It is also possible to impose restraints on the vicinal ^{3}J-coupling value related to the underlying torsion. J is related to the torsion τ by the approximate Karplus relationship: $J = A\cos^2(\tau) + B\cos(\tau) + C$. If you specify a non-zero value for either RJCOEF(1) or RJCOEF(2), then a J-coupling restraint, rather than a torsional restraint, will be imposed. At every MD step, J will be calculated from the Karplus relationship with A = RJCOEF(1), B = RJCOEF(2) and C = RJCOEF(3). In this case, the target values (R1->R4, R1A->R4A) and force constants (RK2, RK3, RK2A, RK3A) refer to J-values for this restraint. RJCOEF(1)->RJCOEF(3) must be set individually for each torsion for which you wish to apply a J-coupling restraint, and RJCOEF(1)->RJCOEF(3) may be different for each J-coupling restraint.

With respect to other options and reporting, J-coupling restraints are treated identically to torsional restraints. This means that if time-averaging is requested for torsional restraints, it will apply to J-coupling restraints as well. The J-coupling restraint contribution to the energy is included in the "torsional" total. And changes in the relative weights of the torsional force constants also change the relative weights of the J-coupling restraint terms.

Setting RJCOEF has no effect for distance and angle restraints.

Defaults for RJCOEF(1)->RJCOEF(3) are 0.0.

(IGR2(i),i=1→200)

If IAT(2) < 0 and IAT(3)=IAT(4)=0, then IGR1 gives the atoms defining the group whose coordinate averaged position is used to define "atom 2" in a distance restraint. If IRESID = 0, absolute atom numbers are specified by the elements of IGR2(). If IRESID = 1, then IGR2(I) specifies the number of the residue containing atom I, and the name of atom I must be specified using GRNAM1(I). A maximum of 200 atoms are allowed in any group. Only specify those atoms which are needed.

Default value for any unspecified element of IGR1 or IGR2 is 0.

(GRNAM1(i),i=1→200)

(GRNAM2(i),i=1→200)

If group input is being specified (IAT(1) or IAT(2) < 0 and IAT(3)=IAT(4)=0), *and* IRESID = 1, then the character names of the atoms defining the group are contained in GRNAM1(i) or GRNAM2(i)), as described above. In the case IAT(1) < 0, each residue IGR1(i) is searched for an atom name GRNAM1(i) and added to the first group list. In the case IAT(2) < 0, each residue IGR2(i) is searched for an atom name GRNAM2(i) and added to the second group list.

Defaults for GRNAM1(i) and GRNAM2(i) are ' '.

IR6 If a group coordinate-averaged position is being used (see IGR1 and IGR2 above), the average position can be calculated in either of two manners: If IR6 = 0, center-of-mass averaging will be used. If IR6=1, the $< r^{-6} >^{-1/6}$ average of all interaction distances to atoms of the group will be used.

Default for IR6 is the value assigned to IR6 in the most recent namelist where IR6 was specified. If IR6 has not been specified in any namelist, it defaults to 0.

IFNTYP If time-averaged restraints have been requested (see DISAVE/ANGAVE/TORAVE above), they are, by default, applied to all restraints of the class specified. Time-averaging can be overridden for specific internals of that class by setting IFNTYP for that internal to 1. IFNTYP has no effect if time-averaged restraint are not being used.

Default value is IFNTYP=0.

Namelist `&rst` is read for each restraint. Restraint input ends when a namelist statement with iat(1) = 0 (or iat(1) not specified) is found. Note that comments can precede or follow any namelist statment, allowing comments and restraint definitions to be freely mixed.

SECTION FIVE: NOESY volume restraints

After the previous section, NOESY volume restraints may be read. This data described in this section is only read if NMRMAX = 2 or 3. The molecule may be broken in overlapping submolecules, in order to reduce time and space requirements. Input *for each submolecule* consists of namelist "`&noeexp`", followed *immediately* by standard AMBER "group" cards defining the atoms in the submolecule. In addition to the submolecule input ("`&noeexp`"), you may also need to specify some additional variables in the `cntrl` namelist in section ONE; see the "NMR variables" description in that section.

In many cases, the user will not prepare this section of the input by hand, but will use the auxiliary program *makeNOESY* to prepare input from simpler files. See the **Auxiliary programs** section below for details.

Variables in the `&noeexp` namelist:

For each submolecule, the namelist "`&noeexp`" is read (either from *stdin* or from the NOESY redirection file) which contains the following variables. There are no effective defaults for *npeak, emix, ihp, jhp,* and *aexp*: you must specify these.

NPEAK*(imix)* Number of peaks for each of the "imix" mixing times; if the last mixing time is *mxmix*, set NPEAK(*mxmix+1*) = -1. End the input when NPEAK(1) < 0.

EMIX*(imix)* Mixing times (in seconds) for each mixing time.

IHP*(imix,ipeak)*

JHP*(imix,ipeak)* Atom numbers for the atoms involved in cross-peak "ipeak" at mixing time "imix"

AEXP*(imix,ipeak)* Experimental or target integrated intensity for this cross peak.

ARANGE*(imix,ipeak)*
"Uncertainty" range for this peak: if the calculated value is within ±ARANGE of AEXP, then no penalty will be assessed. Default uncertainties are all zero.

AWT*(imix,ipeak)* Relative weight for this cross peak. Note that this will be multiplied by the overall weight given by the NOESY weight change cards in the weight changes section (Section 1). Default values are 1.0.

If AWT is negative, this cross peak is part of a set of overlapped peaks. The computed intensity is added to the peak that follows; the next time a peak with AWT > 0 is encountered, the running sum for the calculated peaks will be compared to the value of AEXP for that last peak in the list. Hence, when AWT < 0, its magnitude is ignored, and the corresponding entry in the AEXP array is also ignored. In other words, a set of overlapping peaks is represented by one or more peaks with AWT < 0 followed by a peak with AWT > 0. The computed total intensity for these peaks will be compared to the value of AEXP for the final peak, making use of the value given for AWT in the final peak.

OMEGA Spectrometer frequency, in Mhz. Default is 500. It is possible for different sub-molecules to have different frequencies, but omega will only change when it is explicitly re-set. Hence, if all of your data is at 600 Mhz, you need only set *omega* to 600. in the first submolecule.

TAUROT Rotational tumbling time of the molecule, in nsec. Default is 1.0 nsec. Like *omega*, this value is "sticky", so that a value set in one submolecule will remain until it is explicitly reset.

TAUMET Correlation time for methyl jump motion, in ns. This is only used in computing the intra-methyl contribution to the rate matrix. The ideas of Woessner are used, specifically as recommended by Kalk & Berendsen, *J. Magn. Res.* **24,** 343 (1976). Default is 0.0001 ns, which is effectively the fast motion limit. The default is consistent with the way the rest of the rate matrix elements are determined (also in the fast motion limit,) but probably is not the best value to use, since methyl groups appear to have T1 values that are systematically shorter than other protons, and this is likely to arise from the fact that the methyl correlation time can be near to the inverse of the spectrometer frequency. A value of 0.02 - 0.05 ns is probably better than 0.0001, but this is still an active research area, and you are on your own here! A couple of recent papers that deal with this subject are Olejniczak & Weiss, *J. Magn. Res.* **86**, 148-155 (1990) and Ishima, Shibata & Akasas, *ibid.* **91,** 455-465 (1991); these papers also provide references to earlier work. As with *omega*, *taumet* can be different for different sub-molecules, but will only change when it is explicitly re-set.

ID2O Flag for determining if exchangeable protons are to be included in the spin-diffusion calculation. If ID2O=0 (default) then all protons are included. If ID2O=1, then all protons bonded to nitrogen or oxygen are assumed to not be present for the purposes of computing the relaxation matrix. No other options exist at present, but they could easily be added to the subroutine *indexn*. Alternatively, you can manually rename hydrogens in the *prmtop* file so that they do not begin with "H": such protons will not be included in the relaxation matrix. (*Note:* for technical reasons, the HOH proton of tyrosine must always be present, so setting ID2O=1 will not remove it; we hope that this limitation will be of minor importance to most users.) The *id2o* variable retains its value across namelist reads, *i.e.* its value will only change if it is explicitly reset.

OSCALE overall scaling factor between experimental and computed volume units. The experimental intensities are multiplied by *oscale* before being compared to calculated intensities. This means that the weights WNOESY and AWT always refer to "theoretical" intensity scales rather than to the (arbitrary) experimental units. The *oscale* variable retains its value across namelist reads, *i.e.* its value will only change if it is explicitly reset. The initial (default) value is 1.0.

The atom numbers *ihp* and *jhp* are the absolute atom numbers assigned in the EDIT module of AMBER. For methyl groups, use the number of the last proton of the group; for the delta and epsilon protons of aromatic rings, use the delta-2 or epsilon-2 atom numbers. Since this input requires you to know the absolute atom numbers assigned by AMBER to each of the protons, you may wish to use the separate *getsub* program which provides some facility for turning human-readable names into atom numbers, and also assists in dividing a large molecule into submolecules.

Following namelist "`&noeexp`", give the AMBER "group" cards that identify this submolecule. This combination of "`&noeexp`" and "group" cards can be repeated as often as needed for many

submolecules, subject to the limits described in the "resize.com" shell script. As mentioned above, this input section ends when NPEAK(1) < 0, or when and end-of-file is reached.

SECTION SIX: Chemical shift restraints

After reading NOESY restraints above (if any), read the chemical shift restraints in namelist *&shf*. In many cases, the user will not prepare this section of the input by hand, but will use the auxiliary program *makeSHF* to prepare input from simpler files. See the **Auxiliary programs** section below for details.

Variables in the `&shf` namelist. (Defaults are only available for *shrang, wt*, *nter*, and *shcut*; you must specify the rest.)

NRING — Number of rings in the system.

NATR*(i)* — Number of atoms in the *i-th* ring.

IATR*(j,i)* — Absolute atom number for the *j-th* atom of the *i-th* ring.

NAMR*(i)* — Eight-character string that labels the *i-th* ring. The first three characters give the residue name (in caps); the next three characters contain the residue number (right justified); column 7 is blank; column 8 may optionally contain an extra letter to distinguish the two rings of trp, or the 5 or 8 rings of the heme group.

STR*(i)* — Ring current intensity factor for the *i-th* ring. Older values are summarized by Cross and Wright, J. Magn. Res. 64:220-231 (1985); more recent empirical parametrizations based on a larger database give improved results (K. Osapay and D.A. Case, *J. Am. Chem. Soc.***113,** 9436-9444 (1991).

NPROT — Number of protons for which penalty functions are to be set up.

IPROT*(i)* — Absolute atom number of the *i-th* proton whose shifts are to be evaluated. For equivalent protons, such as methyl groups or rapidly flipping phenylalanine rings, enter all two or three atom numbers in sequence; averaging will be controlled by the *wt* parameter, described below.

OBS*(i)* — Observed secondary shift for the *i-th* proton. This is typically calculated as the observed value minus a random coil reference value.

SHRANG*(i)* — "Uncertainty" range for the observed shift: if the calculated shift is within ±SHRANG of the observed shift, then no penalty will be imposed. The default value is zero for all shifts.

WT*(i)* — Weight to be assigned to this penalty function. Note that this value will be multiplied by the overall weight (if any) given by the SHIFTS command in the assignment of weights (above). Default values are 1.0. For sets of equivalent protons, give a negative weight for all but the last proton in the group; the last proton gets a normal, positive value. The average computed shift of the group will be compared to *obs* entered for the last proton.

SHCUT — Values of calculated shifts will be printed only if the absolute error between calculated and observed shifts is greater than this value. *Default = 0.3 ppm.*

NTER — Resiude number of the N-terminus, for protein shift calculations; *default = 1.*

CTER Residue number of the C-terminus, for protein shift calculations. Believe it or not, the current code cannot figure this out for itself.

Example input files

In this section, we give some commented examples of files for various common tasks. We hope these represent "good AMBER practice," but please recognize that there are many ways to use this program. Comments in parentheses should not be placed in the input file; comments following a "#" sign may be placed in the input files.

1. Simple restrained minimization

```
Minimization with cartesian restraints
 &cntrl
    imin=1, maxcyc=200,           (invoke minimization)
    scee=2.0, idiel=0, cut=12.0, (force field options)
    nsnb=20,                      (update non-bonded list)
    ntpr=5,                       (print frequency)
    ntr=1,                        (turn on cartesian restraints)
 &end
Group input for restrained atoms
 1.0                              (force constant for restraint)
RES 1 58                          (all atoms in residues 1-58)
END
END
```

2. "Plain" molecular dynamics run

```
 molecular dynamics run
 &cntrl
    imin=0, irest=1, ntx=7,                 (restart MD)
    scnb=8.0, scee=1.2, idiel=1, cut=9.0, (force field options)
    ntt=1, temp0=300.0, tautp=0.2,          (temperature control)
    ntp=2, taup=0.2,                        (pressure control)
    ntb=2, ntc=4, ntf=2, nsnb=25,           (SHAKE, periodic bc.)
    nstlim=500000,                          (run for 0.5 nsec)
    ntwe=100, ntwx=100, ntpr=200,           (output frequency)
 &end
```

3. Simulated annealing NMR refinement

```
 15ps simulated annealing protocol
 &cntrl
    nstlim=15000, ntt=1,          (time limit, temp. control)
    scee=1.2,                     (scee must be set - 1-4 scale factor)
    ntpr=500, pencut=0.1,         (control of printout)
    ipnlty=1, nmrmax=1,           (NMR penalty function options)
    vlimit=10,                    (prevent bad temp. jumps)
 &end
#
# Simple simulated annealing algorithm:
#
#  from steps      0 to  1000: raise target temperature 10->1200K
#  from steps  1000 to  3000: leave at 1200K
#  from steps  3000 to 15000: re-cool to low temperatures
#
 &wt type='TEMP0', istep1=0,istep2=1000,value1=10.,
            value2=1200.,    &end
 &wt type='TEMP0', istep1=1001, istep2=3000, value1=1200.,
            value2=1200.0,     &end
 &wt type='TEMP0', istep1=3001, istep2=15000, value1=0.,
            value2=0.0,     &end
#
# Strength of temperature coupling:
#   steps      0 to  3000: tight coupling for heating and equilibration
#   steps  3000 to 11000: slow cooling phase
#   steps 11000 to 13000: somewhat faster cooling
#   steps 13000 to 15000: fast cooling, like a minimization
#
 &wt type='TAUTP', istep1=0,istep2=3000,value1=0.2,
            value2=0.2,     &end
 &wt type='TAUTP', istep1=3001,istep2=11000,value1=4.0,
            value2=2.0,     &end
 &wt type='TAUTP', istep1=11001,istep2=13000,value1=1.0,
            value2=1.0,     &end
 &wt type='TAUTP', istep1=13001,istep2=14000,value1=0.5,
            value2=0.5,    &end
 &wt type='TAUTP', istep1=14001,istep2=15000,value1=0.05,
            value2=0.05,    &end
```

(continued on next page)

3. Simulated annealing NMR refinement *(continued)*

```
#
#  "Ramp up" the restraints over the first 3000 steps:
#
 &wt type='REST', istep1=0,istep2=3000,value1=0.1,
            value2=1.0,  &end
 &wt type='REST', istep1=3001,istep2=15000,value1=1.0,
            value2=1.0,  &end

 &wt type='END'  &end
LISTOUT=POUT                    (get restraint violation list)
DISANG=RST.f                    (file containing NMR restraints)
```

4. Part of the RST.f file referred to above

```
# first, some distance constraints prepared by makeRST:
#      (comment line is input to makeRST, &rst namelist is output)
#
#(   proton 1          proton 2          upper bound)
#--------------------------------------------
#
# 2 ILE HA       3 ALA HN         4.00
#
 &rst iat=  23,  40, r3= 4.00, r4= 4.50,
      r1 = 1.3, r2 = 1.8, rk2=0.0, rk3=32.0, ir6=1, &end
#
# 3 ALA HA       4 GLU HN         4.00
#
 &rst iat=  42,  50, r3= 4.00, r4= 4.50,  &end
#
# 3 ALA HN       3 ALA MB         5.50
#
 &rst iat=  40,  -1, r3= 6.22, r4= 6.72,
      igr1=   0,   0,   0,   0, igr2=  44,  45,  46,   0,   &end
#
# .....etc.....
#
#  next, some dihedral angle constraints, currently prepared "by hand":
#
 &rst iat= 213, 215, 217, 233, r1=-190.0,
      r2=-160.0, r3= -80.0, r4= -50.0, &end

 &rst iat= 233, 235, 237, 249, r1=-190.0,
      r2=-160.0, r3= -80.0, r4= -50.0, &end

# .....etc.....
```

4. Part of the RST.f file referred to above *(continued)*

```
#
#
#   next, chirality and omega constraints prepared by makeCHIR_RST:
#
#
#  chirality for residue  1  atoms:   CA CG HB2 HB3
 &rst iat= 3 , 8 , 6 , 7 ,
   r1=10., r2=60., r3=80., r4=130., rk2 = 10., rk3=10.,  &end
#
#  chirality for residue  1  atoms:   CB SD HG2 HG3
 &rst iat= 5 , 11 , 9 , 10 , &end
#
#  chirality for residue  1  atoms:   N C HA CB
 &rst iat= 1 , 18 , 4 , 5 , &end
#
#  chirality for residue  2  atoms:   CA CG2 CG1 HB
 &rst iat= 22 , 26 , 30 , 25 , &end#
#
  .....etc.....

# trans-omega constraint for residue  2
 &rst iat= 22 , 20 , 18 , 3 ,
   r1=155., r2=175., r3=185., r4=205., rk2 = 80., rk3=80.,  &end
#
# trans-omega constraint for residue  3
 &rst iat= 41 , 39 , 37 , 22 , &end
#
# trans-omega constraint for residue  4
 &rst iat= 51 , 49 , 47 , 41 , &end
#
# .....etc.....
#
```

5. Sample NOESY intensity input file

```
#
A part of the NOESY intensity file we use for plastocyanin:
 &noeexp
  id2o=1,                  (exchangeable protons removed)
  oscale=6.21e-4,          (scale between exp. and calc. intensity units)
  taumet=0.04,             (correlation time for methyl rotation, in ns.)
  taurot=4.2,              (protein tumbling time, in ns.)
  NPEAK = 13*3,            (three peaks, each with 13 mixing times)
  EMIX = 2.0E-02,  3.0E-02,  4.0E-02,  5.0E-02,  6.0E-02,
     8.0E-02,  0.1,  0.126,  0.175,  0.2,  0.25,  0.3,  0.35,
                           (mixing times, in sec.)
  IHP(1,1) = 13*423,  IHP(1,2) = 13*1029,  IHP(1,3) = 13*421,
                           (number of the first proton)
  JHP(1,1) = 78*568,  JHP(1,2) = 65*1057,  JHP(1,3) = 13*421,
                           (number of the second proton)
  AEXP(1,1) = 5.7244,  7.6276,  7.7677,  9.3519,
              10.733,  15.348,  18.601,
              21.314,  26.999,  30.579,
              33.57,  37.23,  40.011,
                           (intensities for the first cross-peak)
  AEXP(1,2) =  8.067,  11.095,  13.127,  18.316,
              22.19,   26.514,  30.748,
              39.438,  44.065,  47.336,
              54.467,  56.06,   60.113,
  AEXP(1,3) =  7.708,  13.019,  15.943,  19.374,
              25.322,  28.118,  35.118,
              40.581,  49.054,  53.083,
              56.297,  59.326,  62.174,
 &end
SUBMOL1
RES 27 27 29 29 39 41 57 57 70 70 72 72 82 82     (residues in this submol)
END
END
```

6. A more complicated constraint

```
# 1)   Define two centers of mass. COM1 is defined by
#      {C1 in residue 1; C1 in residue 2; N2 in residue 3; C1 in residue 4}.
#      COM2 is defined by {C4 in residue 1; O4 in residue 1;  N* in residue 1}.
#      (These definitions are effected by the igr1/igr2 and grnam1/grnam2
#      variables; You can use up to 200 atoms to define a center-of-mass
#      group)
#
# 2)   Set up a distance restraint between COM1 and COM2 which goes from a
#      target value of 5.0A to 2.5A, with a force constant of 1.0, over steps 1-5000.
#
# 3)   Set up a distance restraint between COM1 and COM2 which remains fixed
#      at the value of 2.5A as the force slowly constant decreases from
#      1.0 to 0.01 over steps 5001-10000.
#
# 4)   Sets up no distance restraint past step 10000, so that free (unrestrained)
#      dynamics takes place past this step.
#

 &rst iat=-1,-1, nstep1=1,nstep2=5000,
    iresid=1,irstyp=0,ifvari=1,ninc=0,imult=0,ir6=0,ifntyp=0,
    r1=0.00000E+00,r2=5.0000,r3=5.0000,
       r4=99.000,rk2=1.0000,rk3=1.0000,
    r1a=0.00000E+00,r2a=2.5000,r3a=2.5000,
       r4a=99.000,rk2a=1.0000,rk3a=1.0000,
    igr1 = 2,3,4,5,0,
    grnam1(1)='C1',grnam1(2)='C1',grnam1(3)='N2',grnam1(4)='C1',
    igr2 = 1,1,1,0,
    grnam2(1)='C4',grnam2(2)='O4',grnam2(3)='N*',
 &end
 &rst iat=-1,-1, nstep1=5001,nstep2=10000,
    iresid=1,irstyp=0,ifvari=1,ninc=0,imult=0,ir6=0,ifntyp=0,
    r1=0.00000E+00,r2=2.5000,r3=2.5000,
       r4=99.000,rk2=1.0000,rk3=1.0000,
    r1a=0.00000E+00,r2a=2.5000,r3a=2.5000,
       r4a=99.000,rk2a=1.0000,rk3a=0.0100,
    igr1 = 2,3,4,5,0,
    grnam1(1)='C1',grnam1(2)='C1',grnam1(3)='N2',grnam1(4)='C1',
    igr2 = 1,1,1,0,
    grnam2(1)='C4',grnam2(2)='O4',grnam2(3)='N*',
 &end
```

Auxiliary programs

Here we describe some additional programs that may help out with preparing input for SANDER. These programs are found in the *src/nmr_aux* directory. We are working on improving documentation, so please be patient. An overview of the programs and scripts follows; some of the programs are further described in their own 'chapters' later in the manual.

These programs are grouped for convenience: *prepare-input*; *spectrum* (programs that simulate NOESY spectra from structures); *shifts* (programs that compute chemical shifts from a structure in pdb file format); and *correlation_functions* (programs that compute time-correlation functions of interest in NMR relaxation).

(1) **makeCHIR_RST:** A shell script that reads in a pdb file, and outputs sander- style constraints to maintain chirality about tetrahedral carbons, and to keep peptide bonds trans. See comments at the beginning of the script for usage. NOTE: you must edit the output of this file if you have cis peptide bonds. Generally, this program would be used with an AMBER-created pdb file to create extra constraints that would be used when high-temperature annealing runs are planned.

(2) **makeDIST_RST:** Takes a "7-column" input file giving atom pairs and distance bounds and produces input suitable for SANDER. The input has this format:

```
3  GLY  HN        4 LEU HA      3.5
```

where the first three columns identify proton #1, the second three identify proton #2, and the final value is a distance upper bound. Common pseudo- atoms are defined in a "map" file (like map.DG-AMBER, which translates pseudoatoms from disgeo into AMBER nomenclature). The output can be used for input to sander, but you might want/need to hand-edit the output to fine-tune the constraint weights and force constants, etc. You should certainly check the AMBER output carefully (try setting LISTIN=POUT) in order to be sure that AMBER is interpreting the restraints the way you want it to.

makeDIST_RST.c can just be compiled with a "C" compiler. Then cd to the test directory and run "testit". Compare your output with the *.scripps files. These files will also give you an idea of what "real" input looks like.

The input to *makeDIST_RST* probably doesn't match exactly anything you have. The intent is that this "7-column" format is probably easier for you to create than raw SANDER input is, so that this program might help.

(3) **makeANG_RST:** A similar script to take torsion angle upper and lower bounds, as determined by coupling constant measurements, and convert to SANDER input. This program takes as input a five-column dihedral angle and J coupling constant file along with a pdb file of the DNA molecule that these angles refer to. It creates as output (to standard out) a list of constraints that is readable by AMBER.

The angle file should look something like this:

```
GUA 1   PPA 111 144
CYT 2   EPSILN  20  100
CYT 2   PPA 115 134
THY 3   ALPHA   20  35
ADE 4   GAMMA   54  78
GUA 5   J1P2P   2   3
CYT 6   J2P3P   0   4
THY 7   J3P4P   4   0
```

The first column is the residue name (three letter code). The second is the residue number. Third is the angle (or coupling constant) name. Fourth is the lower bound. Last is the upper bound. The angles that are currently supported follow:

ALPHA BETA GAMMA DELTA EPSILN ZETA CHI PPA

PPA stands for Pseudorotation Phase Angle. When a constraint of this type is encountered, it is expanded into the dihedral angles NU0-NU4. This is a redundant constraint if you are using coupling constant constraints for the ribose ring.

For coupling constants which are not precisely known, a zero can be entered as the upper or lower bound (see example above for CYT 6 and THY 7) to specify a restraint that enforces J coupling to be less than the upper bound (for lower bound=0) or greater than the lower bound (for upper bound=0). The coupling constants that are currently supported are:

J1P2P J1P2D J2P3P J2D3P J3P4P

Where a P refers to ' and a D refers to '' (single and double prime). These are redundant constraints if you are using PPA angle constraints. The pdb file is assumed to be AMBER pdb since the program outputs constraints in AMBER format. Written primarily by Jarrod Smith at Scripps; extensions of the definition files to proteins are planned.

(4) **makeNOEEXP:** Helps prepare NOESY volume input files for sander. For now, documentation is at the top of the makeNOEEXP.f file. The input is in a "7-column" format similar to that used for makeRST, but the last column contains NOESY intensities rather than distance bounds.

(5) **makeSHF:** A shell script that takes chemical shift information in human-readable format, plus a pdb file, and produces sander input for chemical shifts. For now, the user instructions are at the top of this file.

(6) **mdovrly:** Routine that takes coordinates from an SANDER molecular dynamics run and overlays them by doing translations and rotations to provide a best overall fit. This program generally then feeds into the next one:

(7) **mdextract:** This takes an (overlaid) coordinate stream and extracts inter-atomic vectors as a function of time. This program in turns leads to:

(8) **mdcorrp2:** This routine computes time-correlation functions of interest to fluorescence depolarization and spin relaxation.

(9) **intense:** This is a stand-alone program that computes intensities of a NOESY spectrum given an input pdb-file.

(10) **spectrum:** This program takes the output of *intense* and creates an "smx"-file in the format used by the *ftnmr* program from Hare Research. This, in turn, can easily be converted to *Felix* "mat" format using programs that come with that package.

(11) **shifts:** This program essentially is abstracted from sander, and allows chemical shifts to be calculated from a pdb-file, without having to set everything up and run AMBER. It is intended just as a simple "one-step" program to compute chemical shifts for a given structure. If the pdb-file was not generated from AMBER, (e.g., it was taken from Brookhaven, or made in some other way,) it should be run through `protonate` *(q.v.)* first; this of course now makes it a "two-step" procedure(!). Instructions for *shifts* are currently at the top of this shell script.

Overview of NMR refinement using SANDER

We find the SANDER module to be a flexible way of incorporating a variety of restraints into a optimization procedure that includes energy minimization and dynamical simulated annealing. However, there is not, as yet, a generally-accepted and complete "recipe" for obtaining solution structures from NMR data. The comments below are intended to provide a guide to some commonly-used procedures.

Sander is part of a general environment for performing molecular refinements using nmr data as input. Generally speaking, the programs required to do this can be divided into three parts: 1) *front-end* modules, which interact with nmr databases that provide information about assignments, chemical shifts, coupling constants, NOESY intensities, etc.; 2) *restrained molecular dynamics,* which is at the heart of the conformational searching procedures; and 3) *back-end* routines that do things like compare families of structures, generate statistics, simulate spectra, and the like.

Sander provides facilities for carrying out the molecular dynamics part of this scheme. Some of the front-end and back-end programs that we use in conjuction with *sander* are provided in the *nmr_aux* subdirectory. The basic front-end program is *makeDIST_RST*, which converts information in assignment databases into *sander* format. Different NMR-processing programs will clearly require somewhat different input processing. We mostly use *Felix*, marketed by Biosym, and have developed macros for that package that will automatically create input files for *makeDIST_RST*. If you are interested in this option, please contact Dave Case (case@scripps.edu) for up-to-date information. We are happy to provide these to anyone who wants them, but we have not included them in the standard AMBER distribution, since they are still in a state of development.

The principal back-end programs we use are *intense* and *spectrum*, which compute NOESY or ROESY spectra, and the correlation function analysis programs *mdcorrp2* (and their companions). Details about these programs are given below. The *superpose* program is also useful in making multiple structure comparisons among a family of NMR-derived structures. Many other programs are available elsewhere for graphical and tabular examination for bioplymer structures.

Refinements using distance and angle restraints.

The most common approach at present is to interpret NOESY intensities as distance constraints and coupling constants in terms of dihedral angle constraints. To implement this, set *nmrmax = 1* and provide input for sections *one* to *four*. The *plastocyanin* subdirectory in the demo files provides some examples from our work, including a demonstration of how to set up a simulated annealing protocol.

In carrying out such simulations, it is common to modify the standard force field. The *ifstrp* variable allows one to treat nonbonded interactions as soft repulsions with no electrostatic contributions. Since bonds and angles are kept close to ideal values by the force constants inherent in the standard force field, and since the intrinsic dihedral barriers for single bonds are also quite small, this provides a "generic" or simplified representation of the allowed conformational space that may appeal to some users.

Other users will wish to use force fields that incorporate more of what we know about relative conformational energies, i.e. a more elaborate force field. Even here, though, some modifications may be advisable. For example, modifications we have found useful for peptides/proteins include increasing the torsional force constant for the peptide bond (to reduce the tendency of restained simulations to produce badly distorted bonds) and a reduction in the net charge of charged side chains (to compensate in part for neglect of explicit solvent). These changes are incorporated into a database and a *frcmod* file in the *src/nmr_aux/forcefield* subdirectory; see the README file in that directory.

Also in the *dat/nmr* directory is a script, *makeCHIR_RST*, which creates constraints relating to chirality and peptide bonds. Use of these constraints will ensure that high-temperature annealing runs do not destroy chirality or flip peptide bonds. Instructions are in the comments at the beginning of that shell script.

The basic ideas of this scheme owe a lot to the general experience of the nmr community over the past decade. Some good papers to look at are:

(1) "Determination of the three-dimensional structures of proteins and nucleic acids in solution by nuclear magnetic resonance spectroscopy", by G.M. Clore and A.M. Groenenborn, *Crit. Rev. Biochem. Mol. Biol.* **24,** 479-564 (1989). A fairly comprehensive review of the field through 1988.

(2) "Computational methods for determining protein structures from NMR data," by G.P. Gippert, P.F. Yip, P.E. Wright and D.A. Case, *Biochem. Pharm.* **40,** 15-22 (1990).

(3) "Determination of high resolution NMR structures of proteins", by D.A. Case and P.E. Wright, in *NMR in Proteins*, G.M. Clore and A.M. Gronenborn, eds. (New York: McMillan, 1993), pp. 53-91.

(4) D.A. Case, H.J. Dyson and P.E. Wright. Use of chemical shifts and coupling constants in nuclear magnetic resonance structural studies on peptides and proteins. *Methods in Enzymology* **239,** 392-416 (1994).

These last three papers outline procedures in the Scripps group, from which a lot of the NMR parts of SANDER are derived. They are by no means the only way to proceed. We hope that the flexibility incorporated into SANDER will encourage folks to experiment with refinement protocols.

Time-averaged restraints

Time-averaged bonds and angles are calculated as

$$\bar{r} = (1/C) \left\{ \int_0^t e^{(t'-t)/\tau} r(t')^{-ipower} dt' \right\}^{-1/ipower} \quad (1)$$

where

$\bar{r}$ = time-averaged value of the internal

t = the current time

τ = the exponential decay constant

r(t') = the value of the internal at time t'

ipower = average is over internals to the inverse of *ipower*. Usually *ipower* = 3 or 6 for NOE distances, and −1 for angles and torsions (linear averaging).

C = a normalization integral.

Time-averaged torsions are calculated as

$$< \phi > = atan(< \sin(\phi) > / < \cos(\phi) >) \quad (2)$$

where ϕ is the torsion, and $< \sin(\phi) >$ and $< \cos(\phi) >$ 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. This option is chosen with the DISAVI / ANGAVI / TORAVI commands (Section 1). In the first (the default),

$$\partial E/\partial x = (\partial E/\partial \bar{r}) (\partial \bar{r}/\partial r(t)) (\partial r(t)/\partial x) \quad , \quad (3)$$

(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. For example, when $r_3 < \bar{r} < r_4$,

$$E = k_3(\bar{r} - r_3)^2 \quad (4)$$

and similarly for other ranges of $\bar{r}$.

When the second option for calculating forces is chosen (IINC = 1 on a DISAVE, ANGAVI or TORAVI card), forces are calculated as

$$\partial E/\partial x = (\partial E/\partial \bar{r}) (\partial r(t)/\partial x) \quad . \quad (5)$$

For example, when $r_3 < \bar{r} < r_4$,

$$\partial E/\partial x = 2\ k_3\ (\bar{r} - r_3)\ (\partial r(t)/\partial x) \quad . \quad (6)$$

Integration of this equation does not give Equation (4), but rather 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 this second option is that the term $\partial \bar{r}/\partial r(t)$, which occurs in the exact expression [Eq. (3)], varies as $(\bar{r}/r(t))^{1+ipower}$. When *ipower*=3, this means the forces can be varying with the fourth power the distance, which can possibly lead to very large transient forces and instabilities in the molecular dynamics trajectory. [Note that this will not be the case when linear scaling is performed, i.e. when *ipower*=−1, as is generally the case for valence and torsion angles. Thus, for linear scaling,

the default (exact) force calculation should be used].

It should be noted that forces calculated using Equation (5) are not conservative forces, and would cause the system to gradually heat up, if no velocity rescaling were performed. The temperature coupling algorithm should act to maintain the average temperature near the target value. At any rate, this heating tendency should not be a problem in simulations, such as fitting NMR data, where MD is being used to sample conformational space rather than to extract thermodynamic data.

This section has described the methods of time-averaged restraints. For more discussion, the interested user is strongly urged to consult recent studies:

(1) "Time Averaged Nuclear Overhauser Effect Distance Restraints Applied to Tendamistat" by A.E. Torda, R.M. Scheek & W.F. van Gunsteren (1989) *J. Mol. Biol.* **214,** 223-235; and

(2) "Are Time-Averaged Restraints Necessary for NMR Refinement: A Model Study for DNA" by D.A. Pearlman and P.A. Kollman (1991) *J. Mol. Biol.* **220,** 457-479.

(3) "Structure refinement using time-averaged J-coupling constant restraints", by A.E. Torda, R.M. Brunne, T. Huber, H. Kessler and W.F. van Gunsteren, (1993) *J. Biomol. NMR* **3,** 55-66.

Refinements using NOESY and chemical shift restraints.

Refinement directly against measured NOESY and chemical shift restraints is the newest and most specialized functionality of the SANDER module. These can be used in either of two modes:

(1) "Single-point" mode (*imin=1, maxcyc=1, nrun=0*). In this case, the NOESY intensities or ring current contributions can be calculated for a given structure. Allows "back calculation" of the spectrum corresponding to a putative conformation.

(2) Dynamic refinement mode. In this case, measured NOESY and chemical shift data are included in penalty functions that depend upon $(I - I_0)$ where I_0 is the experimentally measured value, and I is the value corresponding the current conformation; the functional form of the penalty depends upon the *ipnlty* variable. Careful experimentation will undoubtedly be required for each data set to define a reasonable penalty function. Simply weighting each observed peak equally (with the default values of *awt* and *arange*) is almost certainly a bad idea, since this effectively gives too much influence to the strong peaks at the expense of longer-range information. Several groups are trying calculations such as these, and some general guidelines about penalty functions should emerge soon.

Users of direct NOESY and chemical shift restraints are encouraged to contact Dave Case (*case@scripps.edu*) to enquire about updates to these sections, and for information about auxiliary programs that help in the preparation of the input files that SANDER needs. Some good references to look at are:

(1) "Characterization of biomolecular structure and dynamics by NMR cross relaxation," by R. Brüschweiler and D.A. Case. *Progress in NMR Spectroscopy,* **26,** 27-58 (1994). Detailed exposition of most of the theory behind NMR dipolar relaxation simulations.

(2) "A new analysis of proton chemical shifts in proteins,", by K. sapay and D.A. Case. *J. Am. Chem. Soc.* **113,** 9436-9444 (1991). Presents the chemical shift algorithm used in SANDER.

(3) R. Brüschweiler and D.A. Case. A collective NMR relaxation model applied to protein dynamics. *Physical Review Letters* **72,** 940-943 (1994). Discusses ways in which normal modes can be used to compute motional correction factors ("order parameters"); this facility is built into SANDER.

Card image input

This section is provided for those who still wish to use the older formatted input for section ONE. Since *interface* also uses this format, this section will also be of interest to those who wish to modify the *interface* definition for *sander*. Descriptions of all of the variables are given above.

```
- 1 -       TITLE

           FORMAT(20A4)

TITLE       Title of the md-run for identification.
```

```
- 2 -      1)TIMLIM  2)IREST  3)IBELLY  4)KFORM  5)ICHDNA  6)IMIN
           7)IPOL    8)IEWALD

           FORMAT(F10.0,10I5)
           Namelist defaults: 999999.,  0,  0,  1,  0,  0,  0,  0
```

```
- 3 -     1)NTX  2)NTXO  3)NTCX  4)IG  5)TEMPI  6)HEAT

           FORMAT(3I5,I10,2F10.5)
           Namelist defaults: 1,  1,  0,  71277,  0.0,  0.0

      NTCX is read but not used.
```

```
- 4 -      1)NTB  2)IFTRES

          FORMAT(2I5)
          Namelist defaults: 0,  1
```

```
- 5 -     1)NRUN  2)NTT  3)TEMP0  4)DTEMP  5)TAUTP  6)TAUTS  7)ISOLVP
           8)VLIMIT

          FORMAT(2I5,4F10.5,I5,F10.5)
          Namelist defaults: 1,  0,  300., 0.0, 0.2, 0.2, 0, 0.0
```

```
  - 6 -      1)NTP  2)PRES0  3)COMP  4)TAUP  5)NPSCAL

           FORMAT(I5,3F10.5,I5)
           Namelist defaults: 0,  1.0,  44.6,  0.2,  0
```

```
  - 7 -      1)NDFMIN  2)NTCM  3)NSCM

           FORMAT(3I5)
           Namelist defaults: 0,  0,  0
```

```
  - 8 -      1)NSTLIM  2)INIT  3)NTU  4)T  5)DT  6)TAUV0  7)TAUV  8) VZERO

           FORMAT(3I5,5F10.5)
           Namelist defaults: 1,  3,  1,  0.0,  0.001, 0.0, 0.1, 0.0
```

```
  - 9 -      1)NTC  2)NTCC  3)NCONP  4)TOL  5) JFASTW

           FORMAT(3I5,F10.5,I5)
           Namelist defaults: 1,  0,  0,  0.0005, 0

      NTCC and NCONP are read but not used.
```

```
 - 10 -     1)NTF  2)NTID  3)NTN  4)NTNB  5)NSNB  6)IDIEL  7)IMGSLT
            8)IPRR 9)IPRW

           FORMAT(9I5)
           Namelist defaults: 1,  0,  1,  1,  25,  1,  0,  0,  0

      NTN is read but not used.
```

```
 - 11 -     1)CUT  2)SCNB  3)SCEE  4)DIELC, 5) CUT2ND

           FORMAT(5F10.5)
           Namelist defaults: 8.0,  none, none,  1.0, 0.0
```

```
 - 12 -     1)NTPR  2)NTWX    3)NTWV   4)NTWE  5)NTWXM  6)NTWVM  7)NTWEM
            8)NTPP  9)IOUTFM 10)NTWPRT
```

```
            FORMAT(10I5)
            Namelist defaults: 50,  0,  0,  0,  999999,  999999,  999999,

      NTPP is read but not used.

------------------------------------------------------------------------

  - 13 -     1)NTR  2)NRC  3)NTRX  4)TAUR  5)NMRMAX  6)ISFTRP
             7)RWELL  8)PENCUT

            FORMAT(3I5,F10.5,2I5,2F10.5)
            Namelist defaults: 0, 0, 1, 0.0, 0, 0, 0.0, 0.1

      NRC and TAUR are read but not used.

------------------------------------------------------------------------

  - 14 -     1)IVCAP  2)MATCAP  3)FCAP

           FORMAT(2I5,F10.5)
            Namelist defaults: 0,  0,  0.0

------------------------------------------------------------------------


   -15 -      1)MAXCYC  2)NCYC  3)NTMIN  4)DX0  5)DXM  6)DELE  7)DRMS

            FORMAT(3I5,4F10.5)
            Namelist defaults: 1,  10,  1,  0.01,  0.5,  0.001,  0.001

      DELE is read but not used.
      Note: this line must be present even if no minimization is
      requested (blank line ok). Minimization is requested using IMIN
      (line 2).

------------------------------------------------------------------------

         -- This card is read only if JFASTW (9.5) = 2 or 3 --

  - 16 -    1)WATNAM  2)OWTNM  3)HWTNM1  4)HWTNM2

            FORMAT(4A4)
            Namelist defaults: "WAT ", "O   ", "H1  ", "H2  "

------------------------------------------------------------------------

         -- This card is read only if NMRMAX (13.5)  >2 --

  - 17 -    1)ISCALE  2)NOESKP  3)IPNLTY  4)MXSUB  5)SCALM  6)TAUSW
```

FORMAT(4I5,2F10.5)
Namelist defaults: 0, 1, 1, 1, 100., 0.1

-- These card is read only if IPOL (2.5) > 1 --

This information must be provided in the formatted form given, even if namelist format input is used above.

- 18- 1) N3B, NION

FORMAT(2I5)

- 19 - IDENTIFICATION OF ATOMS WITH POSITION CONSTRAINTS
*** ONLY IF NTR = 1 ***

See the GROUP section in the Appendices for format.

- 20 - IDENTIFICATION OF ATOMS WHICH MOVE IN A BELLY SIMULATION
*** ONLY IF IBELLY.GT.0 ***

See the GROUP section in the Appendices for format.

GIBBS

Usage: `gibbs [gibfile] [-O] -i gibin -o gibout`
`-p prmtop -c inpcrd -r restrt`
`-ref refc -x mdcrd -v mdvel -e mden`
`-inf mdinfo -ms micstat`
`-cm constmat -cs cnstscrt -a patnrg`

–O: Overwrite output files.

GIBBS/AMBER 4.x is a major functionality revision of AMBER/GIBBS by

David A. Pearlman *
Dept. of Pharmaceutical Chemistry
University of California, San Francisco
San Francisco, CA 94143-0446 (415) 476-4637

* Current Address:
Vertex Pharmaceuticals
40 Allston Street
Cambridge, MA 02139-4211 (415) 576-3111
dap@vpharm.com

The 4.1 release incoporates the ability to carry out polarizability calculations. The polarizabilty code is by Jim Caldwell and Liam Dang, and was ported to Gibbs/Amber by Jim Caldwell.

Other features new to the 4.1 release include:

(1) The ability to carry out Potential of Mean Force calculations when using the Thermodynamic Integration method. (dap, 8/92)

(2) The introduction of code that allows TIP3P water-water non-bonded interactions to be calculated MUCH faster than before (dap,dac, 4/93)

(3) The ability to calculate free energy derivatives. (dap, 8/91)

(4) Implementation of a much faster analytic version of SHAKE for 3-point waters (Using an algorithm and routine written by Shuichi Miyamoto, J. Comp. Chem., 13, 952 (1992)). (dap, 12/92)

(5) The ability to determine free energy contributions on a per-atom basis. (dap, 6/91)

(6) The introduction of an optional secondary cutoff. Interactions in the range between the primary and secondary cutoffs are only calculated every non-bonded update. (dap, 12/92)

(7) The introduction of an optional different cutoff for interactions between the perturbed group and the remainder of the system. (dap, 12/92)

(8) Additional consistency checking when control parameters are read. (dap, Bill Ross)

(9) Various minor improvements.

(10) Incorporation of published bugfixes (as posted on the Amber mail list)

GIBBS is built upon the original (3.0) version of this code, which was written by U.C. Singh and P.A. Kollman (UCSF), using numerous MD routines adapted from GROMOS83 by W.F. van Gunsteren. Speedups to non-bonded pairlist generation and residue-based imaging adopted from code written by George Seibel for AMBER revision 3A.

Background

This module of the AMBER suite of programs calculates the free energy difference, ΔG, between two states "0" and "1":

$$\Delta G = G_1 - G_0 \quad . \tag{1}$$

State 1 is defined in the AMBER PREP module. State 0 is defined in the PARM module. The free energy difference is calculated in a series of incremental steps which connect physical states 1 and 0 through a series of not-necessarily-physical intermediates. The character of the system at each of these intermediate steps is related to a parameter λ.

Free Energy Techniques Available in GIBBS Version 4

There are several techniques available in GIBBS/AMBER 4.0 for evaluating the free energy difference between two states, all based on various statistical mechanical relationships. These include:

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

$$G_{\lambda(i+1)} - G_{\lambda(i)} = -RT \ln < \exp -[(V_{\lambda(i+1)} - V_{\lambda(i)})/RT] >_{\lambda(i)} \tag{2}$$

$$\Delta G = G_1 - G_0 = \sum_i G_{\lambda(i+1)} - G_{\lambda(i)} \tag{3}$$

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 $<>_{\lambda(i)}$ 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 equilibration (NSTPE or NSTMEQ) and data collection (NSTPA or NSTMUL) steps for each $\lambda(i) \rightarrow \lambda(i+1)$ "window".

(2) Slow growth – the same as window growth, except lambda changes by a small amount at every step. Lambda changes slowly enough that it is assumed the system remains in equilibrium at every step (i.e. NSTPE=0, NSTPA=1). Thus the ensemble average in Equation (2) is replaced by its instantaneous value at each step.

(3) Thermodynamic integration – instead of Equations (2) and (3), we use

$$G_1 - G_0 = \int_0^1 < \partial V/\partial \lambda >_\lambda \, d\lambda \tag{4}$$

to calculate the free energy difference. In practice, the integral is approximated by a summation over discrete intervals in λ.

(4) Dynamically Modified Windows – the equations of FEP (2 and 3) 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 Equation (4) 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 pathway. By selecting the appropriate option (NCORC=1), the contribution to the free energy from such constraints will be calculated. This constitutes a PMF calculation. PMF calculations can be carried out as part of either a FEP Window Growth or Dynamically Modified Windows run (1 and 3 above).

Understanding the Output

(a) WINDOW GROWTH, SLOW GROWTH, DYNAMICALLY MODIFIED WINDOWS: At specified intervals during the simulation, the energies calculated up to that point will be reported in the format:

```
Current Lambda =   0.850000
Last F.E. update: Lambda =   0.800000  Step =    4000  Method = F.E.P.
Accumulated "forward" quantities (Nonbond change)
   Lam+d_lam = 0.850000    F_energy  =   +0.64300
   ELEC =      0.000    NONB =     +0.643    14NB =      0.000
   14EL =      0.000    BADH =      0.000
Accumulated "reverse" quantities (Nonbond change)
   Lam-d_lam = 0.750000    F_energy  =   -0.62130
   ELEC =      0.000    NONB =     -0.621    14NB =      0.000
   14EL =      0.000    BADH =      0.000
```

When the free energies reported were last updated, the values of lambda and step number were as given on the second line. Note that the *current* values of λ and Step may be different, if the free energies have not yet been updated to reflect the ensemble now being generated. Also reported on the second line is the method being used to calculate free energy differences: F.E.P. is Free Energy Perturbation (standard or Dynamically Modified Windows); T.I. is Thermodynamic Integration (standard or Dynamically Modified Windows); Slow Growth is self explanatory.

Both "forward" and "reverse" accumulated free energies are reported. By default, GIBBS carries out "double-wide sampling", which means that at every value of λ we calculate the free energies both for going $\lambda \rightarrow \lambda + \delta\lambda$ and for going $\lambda \rightarrow \lambda - \delta\lambda$. The values "Lam+d_lam" and "Lam-d_lam" which are reported were the values at the last free energy update. If there were no sampling errors in our calculations, the independent sums of the "forward" and "reverse" values over the entire simulation would be the same, except for sign. Their actual difference gives us a *lower bound* on the error. By convention, the "forward" energy always corresponds to the energy for the process represented by λ increasing $0 \rightarrow 1$. Similarly, the "reverse" energy corresponds to the process represented by λ decreasing $1 \rightarrow 0$. *This is true regardless of the direction in which the actual simulation was run.*. Note that this differs from the confusing convention used in older versions of GIBBS.

Along with the total accumulated free energies in the "forward" and "reverse" directions, a component breakdown of the energies is given. Components listed include: ELEC (electrostatics, except 1-4's); NONB (non-bonds, except 1-4's); 14NB (1-4 nonbonds); 14EL (1-4 electrostatics) and BADH (bonds, valence angles and torsion angles). *Note that for Windows and Dynamically Modified*

Windows, these components are only estimates. For slow growth and thermodynamic integration, they are exact.

If PMF calculations are performed, a sixth component will be listed, CORC. The procedure used to perform a PMF makes it difficult to separate contributions due to the constraints themselves from those due to non-bonded/electrostatic interactions. For this reason, in these cases CORC will reflect the sum total of all three types of contributions and the individual non-bonded/electrostatic contributions will be reported as 0's.

(b) THERMODYNAMIC INTEGRATION: The output is similar to that described above, except that, because of the integral which must be evaluated in thermodynamic integration (TI) (Equation 4), double-wide sampling is not possible. Thus, only a "forward" set of energies is reported. Again, by convention, these value have the sign appropriate for the 0→1 conversion, regardless of the direction in which the simulation was actually run.

If the calculation of individual entropy/enthalpy contributions is requested, these will also be included in the output, following the same forward/reverse conventions as above.

Defining States and Obtaining Appropriate Starting Coordinates

The state defined in PREP is the λ=1 state and the state given in the PARM is the λ=0 state. The default state from which to start the perturbation is usually λ=1, because the coordinates which are carried from EDIT to PARM to MIN to GIBBS will corresponds to the PREP state. However, you can equilibrate at either λ=1 or λ=0 (or any arbitrary value of λ) as follows:

Set ISLDYN (line 14) to +-2 or +-3;
Set NRUN (line 5) to 1;
Set NSTLIM (line 8) to the number of steps of equilibration desired;
Set ALMDA (line 14) to the value of λ at which equilibration is to take place;
And set NSTMEQ (line 14) to any value greater than NSTLIM.

The program is capable of handling periodic boundary conditions with the solute in a solvent bath either with constant volume or constant pressure. All the data required for boundary conditions is passed from the EDIT and PARM modules. Additionally, it is possible to decouple the free energy into electrostatic and van der Waals contributions, if desired.

SUGGESTED INTRODUCTORY REFERENCES

General Review:

(1) D.L. Beveridge and F.M. Di Capua (1989) "Free Energy Via Molecular Simulation: Applications to Chemical and Biomolecular Systems." Annu. Rev. Biophys. Biophys. 18, 431-492.

Discussion of issues pertinent to free energy perturbation:

(2) D.A. Pearlman and P.A. Kollman (1989) "Free Energy Perturbation Calculations: Problems and Pitfalls Along the Gilded Road." In: Computer Simulation of Biomolecular Systems: Theoretical and Experimental Applications (W. van Gunsteren and P.K. Weiner, eds.), pp. 101-119, Escom Science Publishers, Netherlands.

(3) W.F. van Gunsteren, "Methods for Calculation of Free Energies and Binding Constants: Successes and Problems," ibid, pp.27-59.

(4) D.A. Pearlman and P.A. Kollman (1989) "The Lag Between the Hamiltonian and the System Configuration in Free Energy Perturbation Calculations." J. Chem. Phys. 91, 7831-7839.

(5) D.A. Pearlman and P.A. Kollman (1991) "The Overlooked Bond-Stretching Contribution in Free Energy Perturbation Calculations." J. Chem. Phys. 94, 4532-4545.

Description and characterization of dynamically modified windows:

(6) D.A. Pearlman and P.A. Kollman (1989) "A New Method for Carrying Out Free Energy Perturbation Calculations: Dynamically Modified Windows." J. Chem. Phys. 90, 2460-2470.

Descriptions of general holonomic internal constraints:

(7) D.J. Tobias and C.L. Brooks, III (1988) "Molecular Dynamics with Internal Coordinate Constraints." J. Chem. Phys. 89, 5115-5127.

Description of an application of internal-internal free energy map generation:

(8) D.A. Pearlman and P.A. Kollman (1991) "Evaluating the Assumptions Underlying Force Field Development and Application, Using Free Energy Conformational Maps for Nucleosides." J. Am. Chem. Soc. 113, 7167-7177.

In addition, it is strongly suggested that the user read the discussion which follows the description of the input variables before using GIBBS.

Features new to Version 4.0 of GIBBS (6/91):

-> Ability to use dynamically modified windows
-> Ability to perform thermodynamic integration calculations.
-> The ability to carry out potential of mean force (PMF) calculations about any chosen set of internal coordinates
-> The ability to define lambda-dependent restraints for any chosen set of internal coordinates.
-> Can easily perform the PMF bond length change correction.
-> Improved parameter mixing for dummy/real atom interactions
-> The ability to use both the standard "amber" type of mixing for FEP, and the dual topologies method used by e.g. CHARMM.
-> Enthalpies/entropies can be calculated.
-> Several new temperature coupling options have been introduced, including separate solute/solvent coupling.
-> Perturbation parameters can be different over different ranges of lambda
-> Non-bonded perturbation pairs of atoms which are represented as a hydrogen bond (10-12) in the initial and/or final states are now handled correctly
-> A MICSTAT file is output with dynamically modified windows, which contains a concise history of the free energy change per window.
-> Double-wide sampling can be turned off
-> Ability to use analytic derivatives in slow growth simulations.
-> Periodic imaging can now be done on a residue basis. Imaging is general, and will work for any type of solvent. This allows the cutoff to be >= 1/2 the box width.
-> Intra-perturbed group contributions to the free energy can be calculated, at the user's discretion.
-> User can request pre-SHAKE coordinates to be written to a restart file when SHAKE fails. User can also request program attempt to continue upon a SHAKE failure.
-> A standard method for assigning file names which works on all computers is available.
-> Namelist-type input is optionally available.
-> Numerous bug fixes have been incorporated.
-> Output has been improved and clarified.
-> And more!

Assigning files in version 4

GIBBS, version 4, incorporates a new file assignment protocol which is easy to use, and which will work on all computers. In addition, on Unix machines, file assignments can optionally be specified using flags on the command line, as in version 3A.

For Unix machines (only), the program is invoked:

```
gibbs [gibfile] [-O] [-i PIN] [-p PPARM] [-c PINCRD]
          [-o POUT] [-r PREST] [-inf PINFO] [-ms MICSTAT]
          [-cm CONSTMAT] [-cs CNSTSCRT]
          [-x PCOORD] [-v PVEL] [-e PEN] [-ref PREFC]
```

where PIN, PPARM, etc. are replaced by the appropriate filenames to be assigned. The meanings of the various files are given below.

If "gibfile" is present, it must be the first option given, and this file will be read to make the file assignments. In this case, any remaining flags are ignored. Otherwise, all assignments are made using command-line flags. Any flags not specified default to the given name (e.g. if -o is not specified, output would be in file POUT).

For other machine types (and if gibfile is given on a Unix machine), file assignments are read at run-time from a file named "GIB.FILE" (non-Unix machines) or the file specified as "gibfile" (Unix machines). GIB.FILE contains file assignments, one per line, in the following format:

```
Filetype = Filename
```

"Filetype" is the type of file, from the list of GIBBS I/O file assignments listed below. *It must be given in upper case letters*. Filename is the actual name to be used in opening that file. E.g.

```
POUT = test.out
```

would place the results and diagnostics in a file named test.out. The order in which files are defined is not important. Any line that does not contain the "=" character will be considered a blank line. The GIB.FILE file is opened and read when the run is commenced, and then closed. Once the file definitions have been read, the user is free to discard or change the GIB.FILE file (to e.g. start up a second Gibbs run).

GIBBS I/O FILE ASSIGNMENTS

file	unit	purpose
INPUT:		
PIN	5	Control data for the run (described below).
PPARM	8	Topology file (created by PARM)
PINCRD	9	Initial positions and (optionally) velocities.
PREFC	10	Reference coordinates for optional position restraints (only if NTR = 1)
OUTPUT:		
POUT	6	Formatted results and diagnostics
PREST	16	Restart coordinates and velocities. For restarts, this file should be assigned to PINCRD.
PINFO	7	Short file containing a summary of current energies. For monitoring runs which are executing.
MICSTAT	27	A concise summary of important energy information for each window/interval.
CONSTMAT	28	Contains data related to the matrix of free energy data generated. Only used when IPER>0 for one or more of the constraints/restraints defined with INTR > 0 (see line 13).
CNSTSCRT	42	Contains data required when generating a matrix of free energies corresponding to two independent sets of constraints (IPER>0 and INTR>0; see line 13).
PCOORD	12	Archived coordinate sets (if NTWX > 0)
PVEL	13	Archived velocity sets (if NTWV > 0)
PEN	15	Archived energy related data (if NTWE > 0)

CONTROL PARAMETERS – READ FROM FILE 'PIN'

The title (line 1) must be the first line in PIN. All remaining standard flags may be specified either in the formatted form described below, or using the "namelist" convention. The namelist name is *&cntrl*. The namelist convention is described in Appendix B of the manual.

Note that for each input field, the namelist default is given. This default is used *only* if namelist-form input is used. When formatted input is provided, the default value for every field is 0 or 0.0, except where explicitly noted below.

- 1 - TITLE

FORMAT(A80)

TITLE Title of the md-run for identification.

- 2 - TIMLIM, IREST, IBELLY, IDUM, ICHDNA, IPOL, I3BOD

FORMAT(F10.0,10I5)
Namelist defaults: 999999., 0, 0, 1, 0, 0, 0

2.1 TIMLIM Time limit for the job (in seconds).

2.2 IREST Flag to restart the run.

= 0 Normal start
= 1 Job to be restarted. The accumulated free energies, current value of lambda, and other required quantities are read from the end of the input coordinate file (PINCRD). This file should be the PREST file written by the simulation being restarted.

2.3 IBELLY Flag for belly type dynamics.

= 0 No belly run (allow all atoms to move).
= 1 Belly run. The subgroups of atoms which are allowed to move are read as groups from file PIN. See the section on GROUP in the Appendices.

2.4 IDUM Read but not used.

2.5 ICHDNA Option to modify the charge of end hydrogens during

in vacuo simulations. Without this option, molecular dynamics calculations on nucleotides will result in bonding between the 5′ and 3′ hydrogens and the corresponding phosphate groups.

= 0 no charge modification
= 1 modify charge

2.6 IPOL for inclusion of polarizabilities in the force field.

= 0 non polar calc (no polarizabilities read from "prmtop").
= 1 turn on polarization calculation.

Note: polarization is expensive and is currently recommended ONLY for investigation of polarization parameters.

2.7 I3BOD For 3-body terms with a polarization calc.
= 0 No 3-body terms to be defined.
= 1 Read and use 3-body interaction definitions (see card 18).
3-Body terms only have an effect when polarization is turned on (IPOL=1).

- 3 - NTX, NTXO, IDUM, IG, TEMPI, HEAT

FORMAT(3I5,I10,2F10.5)
Namelist defaults: 1, 1, 0, 71277, 0.0, 0.0

3.1 NTX Option to read the initial coordinates and velocities (also see 'INIT': card #8).

Options 1-3 are used when no set of starting velocities is available (e.g. when starting from a set of minimized coordinates).

Options 4-5 are used when: 1) a starting set of velocities is available (e.g. after MD equilibration or on an MD RESTART); and 2) The coordinates/velocities were generated with MD run either without periodic boundary conditions, or with constant VOLUME periodic boundary conditions. (Box dimensions, if any, are taken from the PARM file).

Options 6,7 are used when both a starting set of velocities are available and the coordinates/velocities were generated with MD run using constant PRESSURE periodic boundary conditions.

= 1 X is read; no velocity information read (Amber format)

= 2 X is read; no velocity information read (unformatted)
= 3 No longer an option

Do not use values of NTX >=4 when PINCRD was not generated with a previous MD simulation.

= 4 X and V are read (unformatted)
= 5 X and V are read (Amber format)

Note: box dimensions only appear in coordinate files written (as PREST) after simulations using periodic boundary conditions (constant volume or constant pressure).

= 6 X, V and BOX are read (unformatted)
= 7 X, V and BOX are read (formatted)

3.2 NTXO Option to write the final coordinates and velocities.

= 0 X, V and BOX are written to file 'PREST' (unformatted)
= 1 X, V and BOX are written to file 'PREST' (Amber format)

3.3 IDUM read, but not used.

3.4 IG The seed for the random number generator. The MD starting velocity is dependent on the random number generator seed. The generator works most effectively when the seed is large and an odd or a prime number (e.g. 71277).

3.5 TEMPI Initial temperature. If TEMPI > 1.0e-06, the velocities are taken from a maxwellian distribution with TEMPI (K). Choosing a low initial temperature (e.g. 10K) allows the calculation to reach the equilibrium conditions with the residual forces in the system during the initial steps.

TEMPI is read but ignored if NTX > 3.

3.6 HEAT If ABS(HEAT) .GE. 1.0E-06, all the velocities are multiplied by HEAT.

- 4 - NTB, IFTRES, BOXX(1), BOXX(2), BOXX(3), BETA, IBXRD

FORMAT(2I5,4F10.5,I5)
Namelist defaults: 0, 1, 0.0, 0.0, 0.0, 90.0, 0

4.1 NTB Flag for periodic boundary conditions.
If NTB .EQ. 0 then the boundary conditions are NOT applied. The periodic box may be rectangular or monoclinic depending on the value of BETA.

= 0 no periodicity is applied
= 1 constant volume
= 2 constant pressure.

4.2 IFTRES Flag to remove the nonbonded cutoff from the solute.

= 0 ALL solute - solute nonbonded interactions are calculated, and the boundary conditions are not applied to the solute. For simulations of highly charged solutes in a water bath, it can be useful to calculate ALL solute - solute nonbonded interactions in order to reduce electrostatic problems. Note that this option is intended for small solutes, and will generate many more nonbonded pairs than the normal method if the solute is large. This option is useful for DNA and counterions. Note: if counterions are added in edit, then they are considered part of the solute.

= 1 Nonbondeds are evaluated normally.

Note: IFTRES will only have an effect when periodic boundary conditions are employed (NTB > 0). When NTB=0, IFTRES=1 behavior (normal nonbond generation) always occurs.

4.3 BOXX(1..3) Lengths of the edges of the periodic box.
If IBXRD > 0, then the values specified here will be used. Otherwise, the values specified here are ignored and the values in the PARM output file (if NTX < 7) or the values in PINCRD (if NTX >= 7) will be used.

4.4 BETA Angle between the x- and z- axes of the box in degrees. The y- axis is assumed to be orthogonal to the other axes. (0 < BETA < 180). The information given for BOX(1..3) above applies to BETA as well.

Non-orthogonal systems do not currently work correctly. Therefore, if IBXRD > 1, BETA must be set to 90.0.

4.5 IBXRD If IBXRD > 0, then the values of BOX(1..3) and BETA specified here will be used. Otherwise, the values in the PPARM or PINCRD file will be used (see above).

- 5 - NRUN, NTT, TEMP0, DTEMP, TAUTP, TAUTS, ISOLVP, NSEL, DTUSE

FORMAT(2I5,4F10.5,2I5,F10.5)

Namelist defaults:

1, 1, 298.0, 10.0, 0.1, 0.1, 0, 0, 1.0

5.1 NRUN Number of MD-runs of NSTLIM steps to be performed. Since the restart coordinates are written only at the end of each run, it is sometimes desirable to break a long run into a series of shorter steps.

If NRUN is set > 1, one should ensure that the number of equilibration+data_collection steps (if performing windows/TI) divides evenly into NSTLIM (line 8).

The number of picoseconds of molecular dynamics is equal to the product of NRUN X NSTLIM X DT.

NSTLIM and DT are found on line 8

5.2 NTT Switch for temperature scaling. Note that several of the temperature coupling options available here are new to version 4 of GIBBS. Several of these are rather ad-hoc, and may not result in a thermodynamically relevant ensemble. (They may be useful when using MD strictly to sample conformational space).

For free energy calculations, it is recommended you stick with NTT = 0 (constant energy), NTT = 1 (constant temperature) or NTT = 5 (constant temperature, separate solute/solvent temperature scaling).

< 0 Re-assign random velocities whenever the current temperature deviates by more than DTEMP from DTEMP0 (target temperature), and every ABS(NTT) steps. Velocities are assigned in a Maxwellian distribution. By default, velocities are are reset for all atoms. If NSEL > 0 (see below), NSEL atoms are selected at random each time a velocity reassignment is to take place, and only those

atoms have their velocities reassigned. (Be sure to set DTEMP0 to a very large value if you wish to disable its action with this option).

Note that the procedure which assigns velocities makes the assignments as if all particles possessed three independent degrees of translational freedom. If SHAKE is used, this will not strictly be the case, and the effective temperature immediately after velocity assignment will be higher than the target temperature. As velocity contributions along the constrained directions are dissipated, the temperature will rapidly adjust towards the target.

= 0 Classical dynamics. Never rescale/reassign velocities after the start. [The total energy (kinetic + potential) is conserved; same as in older versions of GIBBS.]

= 1 Constant temperature, using the Berendsen coupling algorithm. A single scaling factor for velocities is used (same as in older versions of GIBBS).

= 2 Constant temperature, using the Berendsen coupling algorithm. But only consider the solute temperature in determining the velocity scaling on each step. Could result in solvent atoms having very high temperature, and not generally recommended.

= 3 Constant temperature, using Berendsen algorithm. But only rescale when temperature deviates from TEMP0 by more than TEMP0. Single scaling factor.

= 4 When temperature deviates from TEMP0 by more than DTEMP, do one quick scale of the velocities to bring them back to TEMP0. Otherwise, do not scale.

= 5 Constant temperature, using the Berendsen coupling algorithm, and with separate solute/solvent velocity scaling factors. This option is recommended as a replacement for NTT=1, and can help alleviate the "cold solute/hot solvent" problem.

5.3 TEMP0 Reference temperature at which the system is to be kept if NTT not = 0.

5.4 DTEMP The deviation allowed in the constant temperature MD-runs (read but ignored if NTT=0,1,2 or 5).

5.5 TAUTP Temperature relaxation time when NTT .gt. 0.

This is a damping factor which prevents abrupt changes in the system, if the temperature exceed specified deviations. Generally, values for TAUTP should be in the range of 0.1-0.4. Smaller values of TAUTP result in "tighter" coupling.

5.6 TAUTS If NTT=5, then TAUTP is the temperature relaxation time for the solute, while TAUTS is the relaxation time for the solvent. If is specified as 0.0, TAUTS is set equal to TAUTP. Generally, TAUTS should be in the range of 0.1-0.4, with smaller values resulting in "tighter" coupling.

If NTT.NE.5, TAUTS is read but ignored.

5.7 ISOLVP Only used if NTT = 2 or 5 (sep. solute/solvent temp coupling)

= 0 default solvent atom pointer is used. If periodic boundary conditions are being used, this is the last solute atom. Otherwise, it will be the last atom of the system (which results in no separate solute/solvent coupling). Note that counterions are by default considered part of the _solute_.

> 0 Gives the number of the last atom to be considered part of the "solute". ISOLVP should generally be specified if NTT = 5 and NTB = 0.

ISOLVP only affects temperature scaling.

5.8 NSEL Only used if NTT < 0 (random velocity reassignments)

= 0 When velocity reassignment takes place, velocities for all atoms are reassigned.

> 0 When velocity reassignment takes place, NSEL atoms are randomly selected, and only the velocities for those atoms are reassigned.

5.9 DTUSE The value of d_TEMP used in approximating the temperature derivatives by finite differences. DTUSE is only used when individual enthalpy/entropy values are being calculated (ISANDE = 1, line 12). DTUSE should generally be <= 1.0 (larger values often cause floating overflows/ underflows).

- 6 - NTP, NPSCAL, PRES0, COMP, TAUP

FORMAT(2I5,3F10.5)
Namelist defaults: 0, 0, 1.0, 44.6, 0.4

6.1 NTP Flag for constant pressure dynamics. This option MUST be set to 1 or 2 when the MD calculation is done with constant pressure periodic boundary conditions (NTB=2, line 4).

= 0 Classical dynamics without any Pressure Monitoring

= 1 MD with isotropic position scaling

= 2 MD with anisotropic diagonal (x-,y-,z-) position scaling

6.2 NPSCAL Flag for the type of scaling in case of constant pressure run.

= 0 Uniform coordinate Scaling
= 1 Sub molecules Center of mass Scaling

6.3 PRES0 Reference pressure at which the system is maintained (when NTP > 0) in units of bars, where 1 bar ~ 1 atm.

6.4 COMP Inverse compressibility of the system when NTP > 0. The unit is in 1.0E-06/bar (a value of 44.6 is recommended).

6.5 TAUP Pressure relaxation time when NTP .gt. 0 The recommended value is between 0.1 and 1.0 pSEC-1

- 7 - NDFMIN, NTCM, NSCM, ISTAY, NSTAY, NATRCM, ISVAT

FORMAT(7I5)
Namelist defaults: 0, 0, -1, 0, 0, 0, 1

7.1 NDFMIN Number of degrees of freedom that will be subtracted from the total number of degrees of freedom to account for center of mass removal, belly runs, etc. (This will be a value between 0 and 6).

By default (if NDFMIN.GE.0), this value will be set automatically.

-- For nearly all simulations, you should accept the
-- default calculated when NDFMIN = 0.

If you set NDFMIN<0, then ABS(NDFMIN) additional degrees of freedom will be subtracted *in addition to* the number calculated automatically. This option is provided so that you can account for systems containing extended linear moities that reduce the true number of degrees of freedom from that which would be calculated by a simple 3N-6 determination. For example, if you used a linear triatomic molecule for your solvent, you would need to set NDFMIN = -(number of solvent molecules).

7.2 NTCM Flag for the removal of translational and rotational motion from the initial velocities.
NOTE: this flag is automatically set to 0 if belly option is used.

= 0 The translational and rotational motion about the center of mass is not removed
= 1 The above motion is removed and NTCM is reset to 0.

If velocities are being periodically reassigned according to a Boltzmann distribution (NTT < 0) and NTCM = 1, then center of mass motion will be removed after each reassignment.

7.3 NSCM After NSCM steps the above motion will be removed again if NTB .EQ. 0. This flag should be set to -1 if the belly option is used. This results in NSCM .EQ. 90 000 000 steps.

7.4 ISTAY Read, but not used.

7.5 NSTAY Read, but not used.

7.6 NATRCM Read, but not used.

7.7 ISVAT Residue-based periodic imaging flag

= 1 Residue-based periodic boundary conditions are used. For each residue, imaging is determined based on the position of the atom in the residue which is closest to the residue's initial center of mass. Both solute and solvent atoms are imaged on a residue basis. Each atom of any solute or solvent residue "sees" the same image of any interacting residue. This is the default.

= 2 Same as 1, except that for each atom of the _solute_, different whole-residue images on interacting residues may be used. Can be useful when a solute residue is fairly long in one or more dimensions.

The code required to implement ISVAT=2 does not vectorize, and may result in a substantial hit to performance on vector machines. For this reason, ISVAT=1 should be used except where ISVAT=2 is clearly required.

= 3 No residue-based periodic imaging. Separate imaging is done for each atom-atom pair. This is the way imaging was done in versions ≤ 3 of GIBBS (and MD). In typical operation, you would NOT want to use this option.

Setting ISVAT<3 allows a cutoff of as large as ~ 1/2 the smallest box dimension to be used. When ISVAT=3 with periodic boundary conditions, a much smaller cutoff/box ratio must be used.

ISVAT is ignored when periodic boundary conditions are not used.

- 8 - NSTLIM, INIT, NTU, T, DT, VLIMIT, IVEMAX

FORMAT(3I5,3F10.5,I5)
Namelist defaults: 1, 3, 1, 0.0, 0.001, 0.0, 0

8.1 NSTLIM

> 0 Number of MD-steps per run to be performed. NRUN (line 5) such runs will be carried out.

= -1 Continue simulation until done, or until TIMLIN (line 2) is exceeded. This option is often used with dynamically modified procedures (since we don't know at the outset how many total steps will be required).

8.2 INIT Flag for different starting procedures.

If option NTX is less than 5, INIT should be equal to 3. If option NTX is greater than or equal to 5, this option should be equal to 4.

= 3 V(T-DT/2) is obtained by calculating force(T)
= 4 Input V(T-DT/2) is used for the starting velocity

8.3 NTU Read but ignored.

8.4 T The time at the start (psec). Only for your own use. Not important for the simulation.

8.5 DT The time step (psec).

(Note that in the special case where window growth is requested by using the unrecommended flag combination (IFTIME = 0 and ISLDYN = 0; line 14), DT is replaced by the value of DTA on line 15).

8.6 VLIMIT Limiting velocity

If .ne. 0.0, then any component of the velocity that is greater than abs(VLIMIT) will be reduced to VLIMIT (preserving the sign), and a warning message will be printed. This can be used to avoid occasional instabilities in molecular dynamics runs. VLIMIT should generally be set (if at all) to a value like 20., which is well above the most probable velocity in a Maxwell-Boltzmann distribution at room temperature. Note that although it is anticipated that use of a liberal (large) value of vlimit should not adversely affect the statistics accumulated during a free energy simulation, this has not yet been definitively demonstrated.

8.7 IVEMAX Maximum times VLIMIT may be exceeded.

If IVEMAX >0, then IVEMAX specifies the number of times the limiting velocity VLIMIT can be exceeded in a simulation. If VLIMIT is exceeded >= IVEMAX times, the simulation will stop. If IVEMAX =0, there is no limit on the number of times VLIMIT can be exceeded.

- 9 - NTC, NTCC, NCONP, TOL, TOLR2, NCORC, ISHKFL, ITIMTH, JFASTW

FORMAT(3I5,2F10.5,4I5)
Namelist defaults: 1, 0, 0, 0.0005, 0.0001, 0, 1, 0, 0

9.1 NTC Flag for SHAKE to perform bond length constraints. Constraining the bond lengths removes the highest frequency motions from the system and usually allows somewhat larger timesteps to be used.

= 1 SHAKE is not performed
= 2 bonds involving hydrogen are constrained. No bonds which are part of the pert group are constrained.
= 3 all bonds are constrained

9.2 NTCC (Not used)

9.3 NCONP (Not used)

9.4 TOL Relative geometrical tolerance for bond constraints in SHAKE. Smaller values give tighter tolerances. The recommended value is <= 0.0005 Angstrom

9.5 TOLR2 Relative geometrical tolerance for angle and torsion constraints (radians). Smaller values give tighter tolerances. The recommended value is <= 0.0001 rad.

9.6 NCORC Constraint energy flag.

= 0 No constraint contributions to the free energy are calculated.

= 1 The contributions to the free energy from any constraint whose equilibrium value changes with lambda will be calculated. This includes: A) Any constrained internals defined at the end of the input (see flag INTR, line 13); and B) any SHAKE-en bonds (see NTC).

If NCORC=1 is specified, the program will determine which atoms of the system have positions which are dependent on the constraints, and all of these will effectively be included in the "perturbed group". This forces some time-consuming calculations. If no constraints are changing with lambda, be sure to set NCORC=0.

The procedure used to perform a PMF makes it difficult to separate contributions due to the constraints themselves from those due to non-bonded/electrostatic interactions. For this reason, in these cases CORC will reflect the sum total of all three types of contributions and the individual non-bonded/ electrostatic contributions will be reported as 0's.

Note: If you are using a "belly" with NCORC=1, you must ensure that all residues of the pert group are part of the moving belly, and that, additionally, any residues sharing constrained bonds with the pert group (if any) are part of the moving belly.

9.7 ISHKFL Flag which determines what the program will do in the event of a SHAKE/internal constraint failure.

= 0 Program halts immediately. This is what the old versions of Amber did.

= 1 Program will write a restart file containing the coordinates before the failed call to the constraint routine (+ velocities, if applicable). The program will then halt.

> 1 The coordinates will not be constrained on any iteration for which the constraint routine fails. If constraint failure occurs on more than ISHKFL-1 contiguous steps, the program will stop as described for ISHKFL=1.

9.8 ITIMTH Defines which method should be used to calculate constraint free energy contributions when NCORC=1 and the Thermodynamic Integration method (IDIFRG=1) approach is being used.

= 0 Use the Potential Forces (PF) method.

= 1 Use the Constraint Forces (CF) method.

=-1 Use the PF method, override program warnings about constraints within closed rings.

Two methods for determining the constraint free energy contributions during TI have been derived in the literature. The PF method appears to be more efficient, and so is the default. However, PF method cannot be used when any constraints of the system which are changing with lambda (and hence contribute to the free energy) are part of a closed ring. In this case, the CF method must be used.

The program will flag any constraints of the perturbed group which are part of a closed ring, and will stop with a warning if TI is used with PF in such a case. If none of these constrained bonds change with lambda, you can still use the PF method, but must specify ITIMTH=-1 here to ensure you have considered whether this will be appropriate. It is suggested you NOT set ITIMTH=-1 automatically, but only after ensuring that it will be appropriate.

9.9 JFASTW Fast water definition flag. By default, the system is searched for TIP3P waters, and special fast routines are used for these molecules. There are two types of fast routines specific to TIP3P water: 1) A faster, analytic SHAKE algorithm for 3-point water; 2) A faster routine to calculate non-bonded TIP3P-TIP3P water interactions.

In normal operation, the program defaults will be acceptable. However, in rare instances (e.g. for debugging purposes, or

when the user has redefined the definition of a TIP3P water), one may wish to inhibit the use of these fast routines and/or redefine the default definition used in Amber to define TIP3P waters. This option makes this possible.

= 0 Normal operation. The default AMBER definition of TIP3P water is used, and the fast water routines are used where appropriate.

= 1 Use the fast routines for water SHAKE and non-bonds, but redefine the names the program uses to recognize TIP3P waters. The redefinition names are provided below (line 17).

= 2 Use the fast water routine for SHAKE. Do not use the fast water routine for non-bonds.

= 3 Use the fast water routine for SHAKE. Do not use the fast water routine for non-bonds. Redefine the names the program uses to recognize TIP3P waters. The redefinition names are provided below (line 17).

= 4 Do not use fast water routines for either SHAKE or non-bonds.

- 10 - NTF, NTID, NTN, NTNB, NSNB, IDIEL, INBPER, IELPER, IMGSLT, IDSX0, ITRSLU, IOLEPS, INTPRT, ITIP

FORMAT(14I5)

Namelist defaults:

1, 0, 3, 1, 50, 0, 0, 0, 0, 0, 1, 0, 0, 0

10.1 NTF Flag for force evaluation. Typically set to the same value as NTC (line 9).

= 1 complete interaction is calculated
= 2 bond interactions involving H-atoms omitted, except bonds in the perturbed group (use with NTC = 2, see above SHAKE options)
= 3 all the bond interactions are omitted (use with NTC = 3)
= 4 angle involving H-atoms and all bonds are omitted
= 5 all bond and angle interactions are omitted
= 6 dihedrals involving H-atoms and all bonds and all angle interactions are omitted
= 7 all bond, angle and dihedral interactions are omitted
= 8 all bond, angle, dihedral and non-bonded interactions are omitted

10.2 NTID Flag for solvent pairlist behavior.

= 0 only the first atom of each solvent molecule is used when generating the non-bonded pairlist for a periodic system (for water, this is the oxygen). If this atom lies within the specified cutoff, the entire solvent molecule is included in the non-bonded pairlist. This can result in a substantial speedup in non-bonded pairlist generation, and is recommended when using water as the solvent.

=86 all atoms in a solvent molecule are considered when generating the non-bonded pairlist for a periodic system. If any atom of the solvent molecule lies within the specified cutoff, all atoms of the solvent molecule will be included in the non-bonded list. This is the behavior of versions of AMBER <= 3.0.

A value of NTID=0 is suggested for calculations using water as a solvent. For calculations using larger solvent molecules, one should carefully consider whether using only the first atom is appropriate.

Regardless of the value of NTID, all atoms of the *solute* are considered when deciding whether to include a second residue in the interacting non-bonded list for the solute residue.

NTID will have no affect for non-periodic systems.

10.3 NTN Read, but not used.
(This was formerly the flag for generating the non-bonded pairlist. Now non-bonded interactions are always calculated based on a residue basis and stored as atom pairs. If any pair of atoms from different residues are within the cutoff, all atoms pairs across the two residues are included.)

10.4 NTNB Flag for non-bonded pair list generation.

= 0 no pair list will be generated (unlikely you would choose this).
= 1 pair list will be generated

10.5 NSNB After NSNB steps the non-bonded pair list will be updated.

10.6 IDIEL Type of dielectric function to be used.

= 0 distance dependent dielectric function (for in vacuo

simulations of "aqueous" systems).
= 1 constant dielectric function (always use with explicit solvent, e.g. water)

10.7 INBPER Read but ignored.

10.8 IELPER Flag to control the "electrostatic decoupling" of the perturbation energy

= 0 Regular run; no electrostatic decoupling.

= 1 Only the electrostatic contribution to the free energy is calculated keeping the geometry and the VDW parameters pertaining to LAMBDA = 1.

=-1 Only the non-electrostatic (VDW, etc.) contributions to the free energy are calculated and the system changes from that characteristic of LAMBDA = 1 to 0 (or from that characteristic of LAMBDA = 0 to LAMBDA = 1 depending on the signs of IFTIME or ALMDEL).

In electrostatic decoupling, two runs have to be performed, one for electrostatic and the other for VDW etc. contributions. This is useful when a polar or charged group is being established or removed. However, the LAMBDA = 1 state must pertain to the established group (the residue generated by PREP) and the LAMBDA = 0 to the removal of the group (as designated in the PARM input).
The decoupling MUST go through the following perturbation cycle: electrostatic LAMBDA = 1 -> 0 with LAMBDA(vdw) = 1, followed by van der Waals LAMBDA = 1 -> 0. If the simulation is started at LAMBDA = 0, then reverse the above procedure. In this way, charges never appear on atoms which do not possess a vdw radius which avoids very close contacts due to charge-charge attractions.

Notes:
1) Two separate runs are needed to fully carry out the decoupling calculation.

2) In the IELPER=+1 phase, any added restraints/constraints (if INTR > 0) will be fixed at the values they have when lambda=1. (They will still only be applied, however, over the ranges specified).

3) The free energy contribution from internal constraints is never calculated during the IELPER=+1 phase (it is calculated during the IELPER=-1 phase).

```
          ------------------------------------------------------------
          To summarize:

          IELPER          internals/vdw      electrostatics
            +1         fixed @ lambda=1          vary
                       (non-pert) values

            -1            vary                fixed @ lambda=0
                                              (pert) values
          ------------------------------------------------------------
```

10.9 IMGSLT Flag to control the Solute-Solvent interaction in the case of PB simulation

= 0 The Boundary condition is applied to solute-solvent interactions

= 1 No Solute-Solvent imaging. Solute does not see image solvent. This assumes that the solute is centered in the periodic system, and is not free to migrate. Do not use this with mobile solutes. This option is mainly useful for large solutes.

10.10 IDSX0 Flag which controls how the mixed van der Waals parameters are calculated for atom pairs where one atom vanishes (at either lambda=1 or lambda=0). (See Ref. 6).

= 0 r*(state where one atom vanishes) = r*(non-vanishing atom)
(This is the way AMBER has done this in the past)

> 0 r*(lambda) will be calculated so that
r*(state where one atom vanishes) = IDSX0/1000
r*(state where both atoms exist) = r*(A) + r*(B)

= -1: results in r*(state where one atom vanished) = 0.0

10.11 ITRSLU During a periodic boundary conditions simulation, controls whether SOLUTE molecules which exit the primary image box will be translated back into the central box. SOLVENT molecules which exit the central image box are always translated back into the box. A molecule is considered to have floated out of the central box if the first atom of the molecule exits the box.

= 1 Both SOLUTE and SOLVENT molecules which exit the primary image box will be translated back into the box. The system will be translated every 500 steps so that the center of geometry of the solute is centered in the primary image box. (Recommended for most systems).

= 2 Same as 1, except that the system as a whole is not periodically translated to keep the solute centered in the primary image box.

= 0 Only SOLVENT molecules will be translated back into the primary image box. SOLUTE molecules are not translated.

10.12 IOLEPS Controls how parameter mixing is performed for non-bonded interactions.

= 0 Mixing of epsilon (well-depth) van der Waals parameters done as

$$\varepsilon(\lambda) = \lambda * \varepsilon(mixed, \lambda = 1) + (1 - \lambda) * \varepsilon(mixed, \lambda = 0)$$

Mixing of electrostatic interactions done as

$$q_1 q_2(\lambda) = \lambda * q_1 q_2(\lambda = 1) + (1 - \lambda) * q_1 q_2(\lambda = 0)$$

= 1 Mixing of epsilon done as

$$\varepsilon(\lambda) = \sqrt{(\varepsilon_i(\lambda)\varepsilon_j(\lambda))}$$

Mixing of electrostatics done as

$$q_1 q_2(\lambda) = q_1(\lambda) q_2(\lambda)$$

Setting IOLEPS=1 forces mixing to be done as in older versions (e.g. 3.0, 3A) of AMBER. The "new" mixing scheme (IOLEPS=0) has several advantages, including A) a finite derivative for van der Waals interactions involving an atom which "disappears" at one end point; and B) Interaction between pairs of atoms where one/both atoms "disappear" at both end points never contribute to the energy. [One side-benefit of this is that it allows duplicate topologies; thus one can perform perturbations using the "CHARMM" methodology, if desired].

Note that if IDIFRG = 1 (thermodynamic integration), the epsilon parameters are always mixed as described for IOLEPS = 0.

10.13 INTPRT Determines which energies contribute to the calculation of the free energy change.

= 0 No intra-perturbed group energies are accumulated (Same as pre-4.0 versions of AMBER)

= 1 intra-pert. group non-bond energies accumulated as well (but no 1-4's).

= 2 intra-pert. group non-bond energies accumulated
(including 1-4's).

= 3 intra-pert group internal energies accumulated
(bonds, angles, torsions)

= 4 intra-pert group non-bond and internal energies accumulated

= 5 intra-pert group non-bond, 1-4, and internal

Note: If any PMF contributions are being calculated (NCORC = 1, line 9), *all* intra-perturbed group non-bonded contributions will be calculated if INTPRT = 1,2,4 or 5 (when NCORC=1, 1-4's are not broken out separately).

10.14 ITIP By default (ITIP=0), GIBBS assumes that if you are running a periodic boundary conditions (PBC) simulation with solvent, the solvent is TIPNP water. A special characteristic of this solvent model is that there are no h-bond (10-12) interactions between any pair of solvent molecules. A potential speedup is thus obtained by skipping all such h-bond interactions.

If you choose to use a solvent model where there should be h-bond (10-12) interactions calculated between pairs of solvent molecules, set ITIP to any value other than 0.

Note that in either case, all 10-12 interactions between solvent and solute molecules will still be determined normally.

- 11 - CUT, SCNB, SCEE, DIELC, CUT2ND, CUTPRT

FORMAT(6F10.5)
Namelist defaults: 8.0, 2.0, *none*, 1.0, 0.0, 0.0

11.1 CUT The primary cutoff distance for the non-bonded pairs.

11.2 SCNB The scale factor for 1-4 vdw interactions
if (SCNB .EQ. 0.0) then SCNB = 2.0

11.3 SCEE The scale factor for 1-4 electrostatic interactions
There is no namelist default, since the 1991 and previous force fields used 2.0, while the 1994 force field uses 1.2.

11.4 DIELC Dielectric constant for the electrostatic interactions
if (DIELC .LE. 0.0) then DIELC = 1.0

11.5 CUT2ND An (optional) secondary cutoff. If CUT2ND > 0.0, then at every nonbonded update (every NSNB steps), the energies and forces due to interactions in the range CUT< Rij <= CUT2ND will be determined. These energies and forces will be added to the non-bonded interactions within CUT distance at every timestep.

The idea is that long-range interactions change more slowly than short range interactions, and thus this dual cutoff method allows one to include longer-range information at only a moderate additional cost.

11.6 CUTPRT An (optional) alternative cutoff to be used for interactions with the perturbed group. If CUTPRT and CUT2ND are both defined, interactions in the range

CUTPRT < Rij <= CUT2ND

will constitute the secondary cutoff range for interactions with the perturbed group.

- 12 - NTPR, NTWX, NTWV, NTWE, NTWXM, NTWVM, NTWEM, NTPP, IOUTFM, ISANDE, IPERAT, IATCMP, NTATDP, ICMPDR, NCMPDR, NTWPRT

FORMAT(16I5)
Namelist defaults:
100, -1, -1, -1,
999999, 999999, 999999, 0, 0, 0

12.1 NTPR Flag for printing energy related quantities. for every NTPR steps these quantities will be output.

12.2 NTWX Flag for packing the coordinates. For every NTWX steps the coordinates will be dumped through file 'PCOORD' in format (10F8.3). If NTWX=-1, no dumping will be performed.

12.3 NTWV For every NTWV steps the velocities will be written in file 'PVEL' in format (8F8.4). If NTWV=-1, no dumping will be performed.

12.4 NTWE Every NTWE steps energy info is written in file 'PEN' in formatted form. If NTWE=-1, no dumping will be performed.

12.5 NTWXM After NTWXM steps the NTWX switch will be inactive. WARNING: set the following three flags to 0 if long run ... this results in NTWXM= 999 999

12.6 NTWVM After NTWVM steps the NTWV switch will be inactive.

12.7 NTWEM After NTWEM steps the NTWE switch will be inactive.

12.8 NTPP (not used).

12.9 IOUTFM Flag for format of velocity and coordinate sets

= 0 Formatted
= 1 Binary

12.10 ISANDE Flag to output enthalpies and entropies, as well as free energies. Note that these quantities are typically an order of magnitude or more less precise than free energy values, and will be much more sensitive than free energies to the completeness of the ensemble statistics collected. See the discussion following the input description for more information.

Setting ISANDE = 1 will also force the printing of the integrand quantity $< \partial V/\partial \lambda >$
when Thermodynamic Integration is being performed (see the IDIFRG flag, 14.6). This can be useful if the user wishes to apply an alternative integration algorithm.

12.11 IPERAT Request that free energy components or derivatives be calculated. Note that free energy components can be determined during any standard free energy simulation. Free energy derivatives can only be calculated in a special simulation where lambda does not change.

= 0 No free energy components or derivatives will be calculated.

= 1 Report free energy components. Components will be be reported in file PATNRG on a per-atom basis.

= 2 Report free energy components. Components will be be reported in file PATNRG on a per-residue basis.

= 3 Report free energy components. Components will be be reported in file PATNRG on a per-molecule basis.

= 4 Calculate/report free energy components or derivatives (depending on the flag ICMPDR). Values will be reported in file PATNRG for the atoms/groups defined at the end of input using GROUP input.

For free energy components, free energies will be logged as defined by the GROUP definition, subject to the condition that only those atoms which are part of the perturbed group or which move with an added CONstraint will ultimately

be included. All atoms not explicitly included in a group will be put in a final single group.

For free energy derivatives, derivatives will be logged only for those atoms included in a group definition. Any atom of the system may be designated as part of any group (but each atom will be a member of at most one group). Typically, you will place individual atoms in their own groups when calculating derivatives.

12.12 IATCMP If free energy components are being reported, by default only the total free energy per atom/residue/molecule/group is reported. By setting IATCMP > 0, one can force the components to be broken down into electrostatic, non-bonded and internal contributions. IATCMP has no affect when free energy derivatives are being calculated.

= 0 Do not break free energy components into contributions.

= 1 break free energy componenets into contributions.

12.13 NTATDP Free energy components/derivatives will only be reported every NTATDP steps. Note that if free energy components are being logged, a free energy report will occur at a particular multiple of NTATDP steps only if the free energy accumulators have been updated since the last report. For free energy derivatives, energies will be reported every NTATDP steps in all cases.

= -1 NTATDP set to NTPR

12.14 ICMPDR

= 0 no free energy derivatives.

= 1 If IPERAT=4, log the free energy derivatives with respect to charge and the non-bonded parameters epsilon and r*. If the contributions of constraints to the free energy are being calculated (NCORC = 1), then derivatives with respect to constraints in the perturbed group (and added constraints) will also be calculated.

Free energy derivatives can only be calculated for lambda = 0 or lambda = 1.

It is sufficient to define a "null" perturbed group in PARM if you simply wish to determine the non-bonded free energy derivatives of specified atoms.

12.15 NCMPDR IF free energy derivatives are being calculated (IPERAT=4 and ICMPDR=1), NCMPDR gives the number of steps of effective "equilibration." After the first NCMPDR steps, the accumulators for the free energy derivatives are cleared and reset. Free energy derivatives reported from this point forward will only reflect averaging since the accumulators were cleared.

Some people prefer to use a post-processing program to analyze free energy derivatives. Such programs can usually "remove" a given initial portion of the free energy derivative information from subsequent totals. In such a case, you may wish to set NCMPDR=0 here (no "equilibration" phase), and pick the amount of data to discard in the post-processing program.

12.16 NTWPRT Coordinate/velocity archive limit flag. This flag can be used to decrease the size of the coordinate / velocity archive files, by only including that portion of the system of greatest interest. (E.g. one can print only the solute and not the solvent, if so desired).

= 0 Coord/velocity archives will include all atoms of the system.
< 0 Coord/velocity archives will include only the solute atoms.
> 0 Coord/velocity archives will include only atoms 1->NTWPRT.

- 13 - NTR, NRC, NTRX, TAUR, INTR, IBIGM, IDUM, NMRMAX, IWTMAX, ISFTRP, RWELL

FORMAT(3I5,F10.5,6I5,F10.5)
Namelist defaults:
0, 0, 1, 1.0, 0, 1, 0, 0, 0, 0, 5.0

13.1 NTR Flag for restraining specified atoms.

= 0 Classical MD
= 1 MD with restraint of specified atoms

13.2 NRC The number of atoms whose positions to be restrained. (now calculated from group input - cards 17 to end)

13.3 NTRX Flag for reading the cartesian coordinates for restraint from unit PREFC.
Note: the program expects coordinates for all atoms from which a subset is selected by the GROUP input which follows.

= 0 binary form
= 1 formatted form

13.4 TAUR The relaxation time for restraint.

13.5 INTR

= 0 No additional internal restraints or constraints will be read.

> 0 Additional internal restraints/constraints will be read following the normal input. Storage will be allocated for a maximum of INTR added restraints/constraints.

These restraints/constraints can be used for e.g. a PMF calculation.

13.6 IBIGM

To calculate the free energy contributions of a constraint (if NCORC=1, line 9), the free energy at lambda±d_lambda is evaluated by shifting the value of the constraint to its value at lambda±d_lambda. This change in the value of the constraint can be effected either by performing half of the shift at each end/side of the internal, or by performing the entire shift at one end.

= 0 Half of the shift is performed at each end of the internal.

= 1 The entire shift occurs at the end/side of the internal which results in fewer atoms being moved.

The number of atoms whose positions change with shifting the constraint affects how quickly the calculation can be performed. Setting IBIGM = 1 can significantly speed up some calculations (e.g. when rotating a ring about a constrained torsion which joins it to a protein), and IBIGM should typically be set to 1 for *in vacuo* simulations.

In all cases, GIBBS determines which interatomic nonbonded distances depend on constraint values, and only these are recalculated when NCORC=1.

13.7 IDUM Read, but not used.

13.8 IDUM Read, but not used.

13.9 IDUM Read, but not used.

13.10 ISFTRP Causes the 6-12/10-12 functions used for non-bonded interactions to be replaced by "soft repulsion" terms of the form

$$RWELL * (r^2 - r^{*2})^2$$

where r* is the optimal interaction distance between a pair of atoms, calculated from their respective van der Waals radii. This function is sometimes useful in structure refinement, but should *not* typically be used in free energy calculations. Atoms in the perturbed group are always treated by normal (6-12 or 10-12) non-bonded forces, regardless of the value of ISFTRP.

= 0 regular 6-12/10-12's. No soft repulsion.
= 1 replace 6-12's by soft repulsion.
= 2 replace 10-12's by soft repulsion, as well.

13.11 RWELL Force constant (in kcal/mol) used for soft repulsion interactions.

- 14 - IFTIME, CTIMT, ALMDA, ALMDEL, ISLDYN, IDIFRG, NSTMEQ, NSTMUL, NDMPMC, IDUM, IDWIDE, IBNDLM

FORMAT(I5,3F10.5,8I5)
Namelist defaults:
0, 0.0, 1.0, 0.1, -3, 0, 2, 2, 0, 0, 0, 0

14.1 IFTIME Mutation flag.

If ISLDYN=0, then if IFTIME = 0 a standard Window Free Energy Perturbation will be carried out. The perturbation will start at lambda = ALMDA, and proceed in equally spaced intervals of delta(lambda) = ALMDEL until 1 (ALMDEL > 0) or 0 (ALMDEL < 0) is reached. At each value of lambda, NSTPE steps of equilibration and NSTPA steps of data collection (see line 15) will be performed, and energy evaluated using Equation 2.

=±1 A "Slow Growth" perturbation will be carried out. The simulation will start at lambda = ALMDA, and will be run in either the 0->1 direction (IFTIME = +1) or 1->0 direction (IFTIME = -1). CTIMT gives the number of psec of dynamics which would be used to perform the complete change 0->1 (or 1->0). The actual length of the simulation will depend on the starting value ALMDA.

NOTE IFTIME is included for backwards compatibility with

input files created for previous versions (< 4) of AMBER. However, it is strongly recommended that you use the ISLDYN flag to specify the type of simulation desired.

If ISLDYN.NE.0, IFTIME is ignored.

14.2 CTIMT The total length of the MD simulation (in psec) to be carried out in performing a slow growth simulation which transforms state lambda = 0 into lambda = 1 (or vice-versa).

Note that this variable does not control the number of steps which will actually be run. For example, if CTIMT = 10psec, ALMDA = 0.0, ISLDYN = +1, and NRUN*NSTLIM*DT = 5psec, only half of the desired simulation would be carried out. The remainder would have to be carried out by a restart.

CTIMT is only used when ISLDYN = ±1 or (IFTIME=±1 and ISLDYN = 0).

14.3 ALMDA The starting value of lambda for this simulation. The value can be on the inclusive interval 0.0-> 1.0.

ALMDA = 1 corresponds to the "initial" state defined by the structure described in PREP. ALMDA = 0 corresponds to the "final" state defined by the structure described in PARM.

Intermediate "states" are defined by a linear combination of the parameters representative of (lambda = 0) and (lambda = 1).

For restart simulations (IREST=1, line 2), ALMDA is read directly from the restart file, and the value specified here is ignored.

14.4 ALMDEL For _Standard_ (fixed width) Window and TI simulations, ABS(ALMDEL) gives the width of each window or integration interval.

If double-wide sampling is used with Window Growth (default), at each value of lambda, the free energies to both +ALMDEL and -ALMDEL are evaluated. This results in "double wide sampling" (see the introductory text).

If (IFTIME=0 and ISLDYN=0), the sign of ALMDEL determines the direction of the change. If ISLDYN=±3, the sign of ISLDYN determines the direction of the change.

ALMDEL should be chosen so that the free energy change over any interval is not too large. It has been suggested (somewhat arbitrarily) that as a rule the free energy change/window should not exceed 2RT.

ALMDEL is only used when ISLDYN = ±3 or (IFTIME=0 and ISLDYN = 0)

14.5 ISLDYN Free Energy Method flag.

It is recommended that you use this flag exclusively, and ignore IFTIME.

= ±1 Perform a Slow Growth simulation. The simulation will be started at ALMDA, and CTIMT psec will be required to complete the conversion to the end (0 or 1). If ISLDYN = +1, the simulation will be carried out in the direction 0-> 1. If ISLDYN = -1, the simulation will be carried out in the direction 1-> 0.

= ±2 Perform a Dynamically Modified Window simulation. The simulation will be started at ALMDA and progress either in the direction 0-> 1 (if ISLDYN = +2) or 1-> 0 (if ISLDYN = -2). The numbers of equilibration and data collection steps performed at each window are given by NSTMEQ and NSTMUL (on this line).

If IDIFRG = 0, the energy will be evaluated at each interval using Equation 2 (FEP). If IDIFRG = 1, thermodynamic integration will be carried out using Equation (4).

= ±3 Perform a "standard" Window Growth simulation (with fixed width lambda intervals). The perturbation will start at lambda = ALMDA, and proceed in equally spaced intervals of delta(lambda) = abs(ALMDEL) until 1 (ISLDYN > 0) or 0 (ISLDYN < 0) is reached. At each value of lambda, NSTMEQ steps of equilibration and NSTMUL steps of data collection (see this line) will be performed.

If IDIFRG = 0, the energy will be evaluated at each interval using Equation 2 (FEP). If IDIFRG = 1, thermodynamic integration will be carried out, using Equation (4).

14.6 IDIFRG Thermodynamic integration flag.

= 0 No thermodynamic integration.

= 1 If windows or dynamically modified windows have been specified, the energy will be calculated using thermodynamic integration (TI) (Equation 4). The integrand will be evaluated at the endpoints of each "window", and the integral will be approximated using the trapezoidal rule (see the discussion following the input description).

In addition to the integrated free energy, if ISANDE is set = 1 (see flag 12.10), the value of $< \partial V/\partial \lambda >$ will be output at every energy update, so a different integration algorithm can be applied by the user, if desired.

If slow growth has been requested, setting IDIFRG=1 has the effect of performing the slow growth summation using the non-averaging equivalent of the TI equation (4), rather than the FEP equation (2).

14.7 NSTMEQ # of steps of equilibration to be used for each window if ISLDYN = ±2 or +-3.

(Note that if windows are instead requested using the flag combination IFTIME = 0 and ISLDYN = 0, NSTPE [line 15] is used).

14.8 NSTMUL # of steps of data collection to be used for each window if ISLDYN = ±2 or +-3.

(Note that if windows are instead requested using the flag combination IFTIME = 0 and ISLDYN = 0, NSTPA [line 14] is used).

14.9 NDMPMC Every NDMPMC windows, statistics will be dumped to the statistics file (MICSTAT). The statistics file contains a condensed format record of the free energy for each window interval.

The MICSTAT file is not written with slow growth, or if NDMPMC is set < 0.

By default NDMPMC=100. NDMPMC cannot exceed 100.

14.10 IDUM Not currently used.

14.11 IDWIDE Allows double-wide sampling to be turned off with FEP.

= 0 Double-wide sampling performed when FEP windows are being calculated.

= 1 Double-wide sampling turned off when FEP windows are being

```
            calculated.

          Double wide sampling means at each value we calculate the free
          energy in both the "forward" and "reverse" direction. This
          gives an intra-run consistency check (lower bound on the
          error), but requires we calculate every interval twice. The
          simulation can be run in roughly half the time, without the
          forward/reverse consistency check, by setting IDWIDE=1.

          The nature of thermodynamic integration (IDIFRG=1) is such that
          double wide sampling is never carried out. IDWIDE has no effect
          for such calculations.

14.12  IBNDLM     By default (IBNDLM), lambda±d_lambda is
          constrained to the range 0<lambda±d_lambda<1.

          If IBNDLM=1, then lambda±d_lambda can exceed the range 0->1.
          Useful when doing PMF-type calculations.

          Ignored for regular slow growth.

-----------------------------------------------------------------

NOTE: The following three cards are only read if ISLDYN = ±2.

Dynamically Modified Windows options (only if ISLDYN = ±2):

  - 14a -     IAVSLP, IAVSLM, ISLP, CORRSL, AMXMOV

             FORMAT(3I5,2F12.7)
             Namelist defaults:  8,  2,  0,  0.8,  0.1

14a.1  IAVSLP     The current dG/dLAMBDA slope will be approximated by a
          linear fit to the Accumulated G vs. LAMBDA data for the
          previous IAVSLP windows. Maximum value = 1000.

14a.2  IAVSLM     Until IAVSLM windows have been collected, the window
          spacings will be fixed at ALMDL0 (line 14c). When IAVSLM
          windows have been collected, the slope will be calculated
          over all available windows, until IAVSLP windows are available.

          i.e. #_windows < IAVSLM : dLAMBDA = ALMDL0
              IAVSLM <= #_windows < IAVSLP :
                    dLAMBDA calculated from slope over #_windows
              #_windows >= IAVSLP :
                    dLAMBDA calculated from slope over previous IAVSLP
                    windows

          If IAVSLM=-1, window widths will be fixed at ALMDL0 until
```

IAVSLP windows are available.

14a.3 ISLP Determines the direction in which the slope is calculated.

= 0 (default) use the appropriate value of ISLP (-1 or 1) to calculate the slope from energies calculated in the same direction as the simulation (recommended).
= 1 the slope is calculated from the forward (0->1) energy at each step.
=-1 the slope is calculated from the reverse (1->0) energy at each step.
= 2 the slope is calculated using the average of the redundant free energy values (from double wide sampling) over the interval in the direction opposite to the simulation, i.e.
G(reverse[curr window] - G(forware[prev window])/2 or
G(forward[curr window] - G(reverse[prev window])/2
for simulations run 0->1 and 1->0, respectively.
This option can be useful when very few points are used to evaluate each slope (e.g. IAVSLP = 2).
= 3 the slope is calculated using the average of the forward and reverse energies at each lambda.

For best results in most cases, the slope should be calculated in the same direction as the simulation. This is the default behavior (ISLP=0).

With thermodynamic integration, or when double-wide sampling is defeated, ISLP has no effect.

Only options ISLP=0 or ISLP=3 should typically be used when AMXRST > 0.

14a.4 CORRSL If the correlation coefficient for a linear fit to the previous IAVSLM windows is < CORRSL, the number of windows over which the slope is calculated will be halved (for this determination of the slope only), and the slope calculated again. This process continues until the correlation coefficient is > CORRSL.

14a.5 AMXMOV The target free energy change per window. If M is the slope over the previous IAVSLP windows, the next value of dLAMBDA is chosen as dLAMBDA = AMXMOV/M

Note that when double wide sampling is defeated (IDWIDE=1) while using a window FEP technique (IDIFRG=0), the free energy change at a window is defined as the total ("forward" + "reverse") energy change. This differs from the definition when double wide sampling is used, where the free energy change

at a window is approximately 1/2 * ("forward" + "reverse").

Thus, AMXMOV should be suitably increased when IDWIDE = 1.

Dynamically Modified Windows options (only if ISLDYN = ±2):

- 14b - IAVDEL, IAVDEM, AMXDEL

FORMAT(2I5,F12.7)
Namelist defaults: -1, 2, 1.0

14b.1 IAVDEL Number of windows over which the forward and reverse energies will be compared. If IAVDEL<0, no comparisons will be carried out. IAVDEL should always be set <0 when thermodynamic integration is used (IDIFRG = 1). Maximum value = 1000.

14b.2 IAVDEM The relationship between IAVDEL and IAVDEM is analogous to that between IAVSLP and IAVSLM (see line 14a).

14b.3 AMXDEL If < ABS (DA(for)-DA(rev)) > .GT. ABS(AMXDEL) then the next dLAMBDA will be scaled as

[< ABS (DA(for) - DA(rev)) > / AMXDEL] **2 * dLAMBDA

If AMXDEL < 0, then scaling occurs in all cases.

Dynamically Modified Windows options (only if ISLDYN = ±2):

- 14c - ALMDL0, DLMIN, DLMAX, AMXRST, NORSTS, NTSD, ALMSTP(1)

FORMAT(4F14.9, 2I5, F14.9)
Namelist defaults:
0.0001, 1.0D-6, 0.1, 0.5, 0, 0, -1.0

14c.1 ALMDL0 Until enough intervals have been calculated to allow determination of dG/d_lambda and d_lambda consistent with IAVSLP and IAVSLM (see line 14a), an interval width of ALMDL0 will be used.

14c.2 DLMIN The minimum allowable window width.

14c.3 DLMAX The maximum allowable window width

14c.4 AMXRST If the free energy change, dG, over any window is greater than AMXRST, then the data collection phase for that

window will be re-performed using a reduced value of dLAMBDA. The new value of dLAMBDA is determined as

dLAMBDA(new) = (dLAMBDA(old)/dG) * AMXMOV

AMXRST should not be set too close to AMXMOV, or too many windows will be recalculated (which is inefficient). By default, AMXRST=5.*ABS(AMXMOV).

14c.5 NORSTS If this is a restart run, and NORSTS=1, then the restart information relating to dynamically modified windows is not read (cold start for the dynamically modified windows). NORSTS is ignored if this is not a restart run.

Normally, NORSTS should be set to 0.

14c.6 NTSD The statistics relating to dynamically modified windows are written to file POUT every NTSD. If NTSD=0, then NTSD is set equal to NTPR (line 12), and these statistics will be output every time the standard energy information is printed.

14c.7 ALMSTP(1) Allows the values of AMXMOV, DLMIN, DLMAX, AMXRST, and NTSD to be different for different ranges in LAMBDA.

> 1 or
< 0

the values defined in lines 14a-14c will remain in effect for the whole run.

> 0 and
< 1

the values defined in lines 14b-14d will remain in effect for the range of LAMBDA

ALMDA-> ALMSTP(1) (ALMDA is defined on line 14)

In this case, _additional line(s)_ are read with the values of the above variables over various ranges of LAMBDA. Each line has the format

AMXMOV, DLMIN, DLMAX, AMXRST, NTSD, ALMSTP(I)

FORMAT(4F14.9,I5,F14.9)

These lines are read until ALMSTP(I) > 1 or ALMSTP(I) < 0. Each set of values applies to the range in LAMBDA

ALMSTP(I-1) -> ALMSTP(I)

Note that the for the last line, ALMSTP(I) must be greater than 1, or less than 0 (not equal to). This is avoid machine precision problems.

Note also that, at present, "namelist"-format input always assumes ALMSTP(1) < 0 (i.e. AMXMOV, DLMIN, etc. remain fixed over the entire run). If you wish to use the functionality described above for ALMSTP(), you must use formatted input.

- 15 - NSTPE, NSTPA, DTE, DTA

FORMAT(2I5,2F10.5)
Namelist defaults: 2, 2, 0.001, 0.001

Additional input in the case where IFTIME = 0 and ISLDYN = 0 (Window Growth specified using the "old" amber syntax).
In all other cases this information is read but ignored.

15.1 NSTPE The number of steps of Equilibration before collecting the Free Energy Statistics. For each window the system is equilibrated for NSTPE steps. (When ISDYN=±2 or ±3, NSTMEQ serves the same purpose).

15.2 NSTPA The number of steps for data collection. The averaging is performed over this number of steps. (When ISLDYN=±2 or ±3, NSTMUL serves the same purpose).

15.3 DTE Read, but not used.
(Note that in earlier <4 versions of Amber, this variable could be used to set the time-step for the equilibration phase of window runs).

15.4 DTA The time-step used for window runs specified by IFTIME=0 and ISLDYN=0. All other runs use the time-step specified on line 8.

- 16 - IVCAP, NATCAP, FCAP

FORMAT(2I5,F10.5)
Namelist defaults: 0, 0, 0.0

16.1 IVCAP Flag to control Cap Option
The Cap option is to solvate a spherical portion of a solute and to hold the solvent from evaporating through a half-harmonic potential.

= 0 Cap will be in effect if it is passed from the the parm module
= 1 Cap will be activated except that the Cap atom pointer would be modified
= 2 Cap will be inactivated

16.2 NATCAP The Cap atom pointer
It is the last Non-Cap atom number. If IVCAP.EQ.1 then the pointer passed from the PARM Module will be overwritten by this number.

16.3 FCAP The Force Constants for the Cap Atoms

-- This card is read *only* if JFASTW (9.9) = 2 or 3 --

- 17 - 1)WATNAM 2)OWTNM 3)HWTNM1 4)HWTNM2

FORMAT(4A4)
Namelist defaults: "WAT ", "O ", "H1 ", "H2 "

This line allows redefinition of the default residue and atom names used by the program to determine which residues are TIP3P waters. Except in unusual circumstances, the default water names should be acceptable.

17.1 WATNAM The residue name the program expects for TIP3P waters.

17.2 OWTNM The atom name program expects for the TIP3P oxygen.

17.3 HWTNM1 The atom name program expects for the TIP3P 1st H.

17.4 HWTNM2 The atom name program expects for the TIP3P 2nd H.

-- These card is read *only* if I3BOD (2.7).NE.0 --

This information must be provided in the formatted form given, even if namelist format input is used above.

- 18A- 1) N3B, NION

FORMAT(2I5)

The number of 3body interactions to be defined, and the number of ions in the system.

== Include N3N cards 18B to define all 3-body interactions ==.

- 18B- 1)AT1(I) 2)AT2(I) 3)ACON1(I) 4)BETA31(I) 5)GAMMA31(I)
6)ACON0(I) 7)BETA30(I) 8)GAMMA30(I)

FORMAT(A4,A4,2X,6E10.3)

AT1(I): The second atom in this 3-body interaction.
AT2(I): The third atom in this 3-body interaction.
ACON1(I): The pre-exponential factor for this 3-body interaction for the lambda = 1 state.
BETA31(I): The beta value for this 3-body interaction, for the lambda = 1 state.
GAMMA31(I): The gamma value for this 3-body interaction, for the lambda = 1 state.
ACON0(I): The pre-exponential factor for this 3-body interaction for the lambda = 0 state.
BETA30(I): The beta value for this 3-body interaction, for the lambda = 0 state.
GAMMA30(I): The gamma value for this 3-body interaction, for the lambda = 0 state.

- 19 - IDENTIFICATION OF ATOMS WITH POSITION CONSTRAINTS
*** ONLY IF NTR = 1 ***

Constraint reference atoms are obtained by first reading coordinates for the entire structure through file 'PINCRD' or 'PREFC', then specific constraint atoms are selected by group. See the section on GROUP in the Appendices for format. Does not support a namelist convention.

- 20 - IDENTIFICATION OF ATOMS FOR BELLY RUN
***** ONLY IF IBELLY .GT. 0 *****

The belly atoms are loaded as groups. Consult the GROUP section in the Appendices for a description of how to define a group. The group definition immediately follows the end of the &cntrl namelist. *The GROUP input does not support a namelist convention.*

- 21 - DEFINITION OF GROUP INPUT FOR FREE ENERGY COMPONENTS OR DERIVATIVES
***** (ONLY IF IPERAT = 4) *****

For free energy components, free energies will be logged as defined by the GROUP definition, subject to the condition that only those atoms which are part of the perturbed group or which move with an added CONstraint will ultimately be included. All atoms not explicitly included in a group will be put in a final single group.

For free energy derivatives, derivatives will be logged only for those atoms included in a group definition. Any atom of the system may be designated as part of any group (but each atom will be a member of at most one group). Typically, you will place individual atoms in their own groups when calculating derivatives.

Note that in GIBBS, GROUP input supports two new features that can be helpful in defining the input for free energy components or derivatives. Both allow the creation of multiple single-atom groups:

ATOM -IAT1 IAT2

(1st atom number negative) will place each atom from IAT1 to IAT2 in its own group.

RES -IRES1 -IRES2

(both residue numbers negative) will place each atom of every residue in the range IRES1->IRES2 in a separate group.

Group definition syntax is otherwise the same as described in the manual.

- 22 - DEFINITIONS OF INTERNAL RESTRAINTS/CONSTRAINTS
*** ONLY IF INTR > 0 (line 13) ***

BRIEF DESCRIPTION:

Setting INTR > 0 allows the user to define here a series of internal restraints and constraints whose force constants and equilibrium values are a function of lambda.

Restraint/constraint definitions must be entered in the formatted form shown below, not in a namelist.

Restraints/constraints are read in as pairs of lines:

```
line A: IAT1,IAT2,IAT3,IAT4,IUMB,IZE,ITOR,RLMDA1,RLMDA2
        FORMAT (7(I5,1X),2F10.5)
line B: RKEQ1,REQ1,RKEQ2,REQ2,IPER,IPER2
        FORMAT (4F10.5,2I5)
```

As many restraints/constraints may be defined as are desired. A blank record signals the end of the input. This data must be entered in the formatted form shown.
It does not support a namelist convention.

INPUT VARIABLES
----- ---------
IAT1-->IAT4:

The absolute atom numbers for the atoms defining the restraint. If an atom number is <0, the absolute value of the atom number is used (additional behavior for <0 values is defined when IZE=1; see below).

```
IAT3 = IAT4 = 0    :  Bond restraints/constraints
IAT4 = 0           :  Angle restraints/constraints
IAT1->IAT4 non-zero:  Torsion restraints/constraints
```

RLMDA1
RLMDA2:The restraint/constraint will be applied only over the range in lambda (RLMDA1, RLMDA2).

RKEQ1
REQ1 :The force constant in kcal/mol and equilibrium value, respectively, for the restraint/constraint at lambda = RLMDA1.
RKEQ2
REQ2 :The force constant in kcal/mol and equilibrium value, respectively, for the restraint/constraint at lambda = RLMDA2.

If RLMDA1=RLMDA2, the force constant and eq. value are fixed at RKEQ1 and REQ1 (RKEQ2 and REQ2 are ignored).

RKEQ1 and RKEQ2 are ignored for constraints (ITOR=2).

If REQ1=999. or REQ2=999., the corresponding equilibrium value is set to the current value of the internal coordinate (as determined from the input set of coordinates PINCRD).

If ABS(REQ1) > 1000, the corresponding equilibrium value is set

```
REQ1 < 0:    REQ1 = current_value - [ABS(REQ1)-1000.]
REQ1 > 0:    REQ1 = current_value + [ABS(REQ1)-1000.]
```

If ABS(REQ2) > 1000, REQ2 is analogously reset.

Intermediate $K_{eq}(\lambda)$ and $R_{eq}(\lambda)$
are determined by linear interpolation between the force constants and equilibrium values at RLMDA1 and RLMDA2. No restraint/constraint is applied outside the range (RLMDA1,RLMDA2).

IZE:

= 0 The restraint/constraint defined here is used _in addition to_ other parameters corresponding to this atom sequence from parm (if any).

= 1 The restraint parameters defined here _replace_ overlapping parameters from parm (if any) for this atom sequence.

When IZE=1, any atom number IAT1->IAT4 which was specified as < 0 has a special meaning: It allows a "wildcard" match to the corresponding atom number when replacing parameters from parm. For example, the sequence -1 3 8 -14 would result in a torsional restraint which would replace parameters for all torsions centered on the bond between atoms 3 and 8.

IZE is read but ignored when ITOR=2 (constraints).

IUMB: Determines the type of restraint.

= 0 The restraint is to be considered part of the molecular force field. The free energy contribution from the restraint is calculated by the standard formula (c.f. Equation 2).

= 1 The restraint is considered to be an "umbrella" term. The effects on the ensemble of the restraint are evaluated using the following function in place of Equation 2:

$$\Delta G = -RT \ln(< e^{-\Delta V/RT} e^{\phi/RT} >_{V+\phi} / < e^{\phi/RT} >_{V+\phi}) \quad ,$$

where ϕ is the sum of all umbrella restraint terms and ΔV is as described for Equation 2.

IUMB is ignored for constraints (ITOR=2).
IUMB = 1 will not work correctly with slow growth or thermodynamic integration.

ITOR: Functional form/constraint flag

= 0 If this is a torsional restraint, a potential of the form

$$K_{tor} (\tau - \tau_0)^2$$

is used. This functional form is always used for bonds and angles (ITOR = 0 has no effect for bonds/angles).

= 1 If this is a torsional restraint, a potential of the form

$$K_{tor} (1 - \cos(\tau - \tau_0))$$

is used. (ITOR = 1 has no effect for bonds/angles).

= 2 Then a constraint, rather than restraint, is applied to the corresponding internal coordinate. This is applicable to all types of internal coordinates (distances, angles, torsions). If NCORC = 1 (line 9), then an effective "potential of mean force" (PMF) contribution to the free energy will be calculated for this internal coordinate.
General "holonomic" internal constraints are used, as described in Reference 7.

When ITOR = 2 (internal is being constrained), IZE is ignored, and the following occurs:

For bonds and angles, if the constrained internal matches an internal in the topology file, force constant parameters for matching internal will be set to 0.

For torsions, if the constrained internal matches an internal in the topology file, A) forces for all torsions centered on the same bond will be omitted B) The contributions to the free energy of all torsions centered on the same bond as the constraint will be calculated. This is necessary because several torsions can be centered on a central bond, and there is no fixed relative arrangement for these torsions.

IPER: IPER can be used to define a simulation where two internal coordinates will be varied with two independent values of lambda. Such a simulation can be used to generate a free energy

internal-internal map (sort of a free energy equivalent to a Ramachandran map) to be generated.

NOTE

The output of this option is somewhat complex, and is intended for post-processing by a separate program. Any 2-D run of value will necessarily be very compute-intensive, and a number of issues must be considered before undertaking such a simulation. This option should generally be avoided by the novice user. If you are considering performing such a simulation, you are *urged* to read Reference 8 (see above) first.

For use with the IPER flag, we define:

"primary" lambda: the "normal" lambda; that is, the lambda used in standard GIBBS runs to describe how the system varies between the initial and final states.

"secondary" lambda: a second lambda, which is translated from 0->1 at each value of the "primary" lambda.

= 0 This restraint will vary with the primary lambda; i.e. the equilibrium value and force constant will be a function only of the primary lambda. This is standard behavior.

> 0 This restraint will vary with the secondary lambda; i.e. the equilibrium value and force constant will be a function only of the secondary lambda. Lambda will be varied from 0->1 for this restraint in a series of IPER equally-spaced intervals (windows).

The "secondary" lambda is not used unless one or more restraints are defined with IPER > 0.

The number of windows used for each "primary" restraint will be the same, and the number used for each "secondary" restraint will be the same. The first IPER(I) > 0 sets the number of windows used for _all_ secondary restraints.

If secondary restraint(s) are requested, the value of IPER2 (see below) corresponding to the first value of IPER(I) > 0 defines the number of windows used for every primary restraint. Note that any dynamically modified window or slow growth flags (card 14) will be defeated in this case.

When calculating PMF-type energies (if NCORC=1), constraints will be applied in two cycles. First, dG will be calculated for +-d(internal) for only those internals for which

IPER=0. Then a dG will be calculated +-d(internal) for only those internals for which IPER>0.

Any parameters (other than constraints) that vary with lambda will only change when lambda for the primary constraints changes. You will typically define the "perturbed group" (see the PARM module) to contain no atoms, when using "secondary" restraints/ constraints.

If IPER > 0, window or dynamically-modified window growth must have been requested (line 14). IPER cannot be set > 0 with slow growth or with thermodynamic integration (IDIFRG > 0).

The matrix of energies from a 2-D run is contained in file CONSTMAT. A matrix can be generated with either IDWIDE = 0 or IDWIDE = 1, but it is strongly recommended that IDWIDE = 1 (no double-wide sampling) be used. In this case, five free energy difference are evaluated from each ensemble, corresponding to moves from (lam1, lam2) to (lam1, lam2+d_lam2), (lam1, lam2-d_lam2), (lam1+d_lam1, lam2), (lam1-d_lam1, lam2), (lam1-d_lam1, lam2-d_lam2). This set allows the whole free energy map to be evaluated most efficiently (see the Pearlman and Kollman reference [8] noted above).

The "secondary" lambda always changes in the "forward" direction, always starts at 0.0, and always ends at 1.0. After lambda has gone from 0->1. The primary lambda is incremented one step, the secondary lambda is reset to 0, and another cycle of secondary lambda changes occurs. At the start of each cycle of changes in the "secondary" lambda, the current coordinates are stored in file CNSTSCRT.

IPER2: If IPER > 0 for a particular restraint/constraint ("secondary" restraints defined), IPER2 gives the number of "windows" used in translating the "primary" lambda from 0 to 1. See the description of IPER above. If IPER > 0, IPER2 fixed-width windows will be used for the "primary" restraints, regardless of the behavior requested by ISLDYN, etc. (lines 14-ff).

++++++ END OF INPUT ++++++

Variables printed in the unit 6 output file but not defined in the user input file:

NPM, NSM, NRAM, IPTRES, IPTATM, NSPSOL, SSPSTR

Choices Affecting Free Energy Calculations

David A. Pearlman
Dept. of Pharmaceutical Chemistry
University of California, San Francisco, CA 94143-0446
1/91

The development of ever-more-powerful computers, combined with the wide dissemination of modeling packages like AMBER, puts the power to perform valuable calculations in the hands of an increasingly large number of scientists. It is tempting to say that, given the increasing sophistication of such programs, all one needs is the appropriate hardware and software to perform good experiments.

But this is not the case. As modeling programs have grown more sophisticated, they have sprouted an ever-increasing array of options–options which must be properly chosen, if worthwhile results are to be obtained. And even if the options are appropriately set, one must ensure that the program is properly suited for the chosen application. Nowhere in AMBER is this more true than the GIBBS free energy module.

Here we discuss several issues which impinge on developing an appropriate GIBBS input file, and on interpreting the results produced. One is also strongly encouraged to review the literature referenced here and in the preface to the GIBBS program.

I. What method should be used to calculate the free energy?

GIBBS version 4.0 offers five choices of method for calculating the free energy difference between two states. These include the general classes slow growth, free energy perturbation, and thermodynamic integration, as well as dynamically modified variants of the latter two. These were described in the introduction to GIBBS. As yet, it has not been shown conclusively what method is "best" for any particular type of problem.

(1) Slow growth: Some early studies indicated that slow growth might be a more efficient technique for free energy calculation than fixed-width windows[1]. More recent work[2] 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 quantified.

(2) Window Growth: The equations of 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 Equation (2) 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 difficult to obtain reliable statistics. 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[3].

(3) Thermodynamic Integration (TI): TI is appealing because it avoids the problem in sampling V($\lambda(i+1)$)-V($\lambda(i)$) described for FEP above. But TI has its own problem: The driving equation of TI is an integral (Equation 4), which in practice must be calculated approximately by evaluating the integrand at finite intervals of λ. 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.

At this time, the relative rates of convergence of the averaged quantities required by FEP and TI, which will directly impinge on the reliabilities of the two techniques, have not been determined.

Note that we approximate the integral using the trapezoidal algorithm, i.e.

$$\Delta G_i \;=\; G(\lambda(i+1)) + G(\lambda(i)) \;=\; (<\partial V/\partial\lambda>_{\lambda(i+1)} - <\partial V/\partial\lambda>_{\lambda(i)})\;\;(\lambda(i+1)-\lambda(i))/2 \quad . \qquad (5)$$

This integration method should be reasonably accurate in most cases. But in case the user wishes to try their own integration scheme, setting ISANDE = 1 with TI will also force reporting of the values of $<\partial V/\partial\lambda>_{\lambda(i)}$ and several other averages at every evaluation point (the other values reported relate to calculating the enthalpy/entropy, as described below).

(4) Dynamically Modified Windows (DMW): In dynamically modified windows[3], the $\delta\lambda$ spacing between consecutive windows in FEP or TI is continually changing, to achieve a relatively constant free energy change per window. This should improve the efficiency of the calculation, by focusing proportionately more simulation time on those ranges of λ where the free energy is changing more rapidly. We have, in fact, shown that dynamically modified windows significantly improve the sampling efficiency of FEP simulations for model compounds[3]. The biggest drawback to DMW is that, because we do not know *a priori* the exact shape of the free energy versus λ curve when we start a simulation, we cannot predict with certainty how long the simulation will take to go to completion. This caveat noted, it appears that DMW would be beneficial to most FEP and TI simulations.

II. Enthalpies and entropies

GIBBS Version 4 allows the user to request that the enthalpy and entropy changes be reported in addition to the free energy (which is always reported). Two different schemes are used to calculate these quantities, depending on the free energy calculation method. Note that in either case, the enthalpy and entropy are necessarily dependent on being able to reliably extract small differences between averages of (often large) total system energies. In the case of free energy, on the other hand, we need only measure the average of a potential difference or a derivative. For this reason, enthalpy/entropy estimates are typically more than an order of magnitude less accurate than their free energy counterpart. One should be very cautious when interpreting them.

For FEP, the approximate equations derived in Ref. 4 are used. These approximate the required temperature derivatives by a finite difference. The equations used are derived from the FEP expression, and the sum of the resulting (enthalpy - T*entropy) will equal the reported free energy.

For TI, the enthalpy and entropy are evaluated using exact-form integral relationships presented in Ref. 5. The (enthalpy - T*entropy) calculated by this method will not necessarily equal the reported total free energy; the difference between the two quantities can be taken as a crude indication of the reliability of the enthalpy/entropy values. The integrals are approximated by the trapezoidal rule, as

described above (Equation (5)).

III. Mixing rules for "vanishing" atoms

By default (and without exception in older versions of Gibbs), the optimal interaction r^*_{ij} between two atoms i and j is given by

$$r^*_{ij}(\lambda) = r^*_i(\lambda) + r^*_j(\lambda) \quad (6)$$

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

$$r^*_{ij}(0) = r^*_i(0) + r^*_j(0) = r^*_j(0) \quad . \quad (7)$$

Thus, r^*_{ij} never gets smaller than $r^*_j(0)$. At λ=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". This is the effect of variable IDSX0 (line 10).

IV. Using Dynamically Modified Windows

The theory of DMW, and some exploratory applications, are described in Ref. (3). A sample input for GIBBS is shown below, follow by a few important explanatory notes.

```
line
14      0  40.00000  0.00000  -0.02500  +2  0  100  100  0  0  0  0
14a     8     2    0    0.8000000    0.0100000
14b  -10    20    0.0001000
14c   1.0D-5  1.D-10   1.0D-2         0.10000000     0     0  -1.00

(format compressed to fit page)
```

Line 14
We set ALMDEL = 0, ISLDYN=+2, IDIFRG=0, NSTMEQ=100, and NSTMUL=200. This results in dynamically modified window FEP, with 100 steps of equilibration and 100 steps of data collection per window.

Line 14a:
On the next line, we set IAVSLP = 8, IAVSLM=2, and CORRSL=0.8. This means that, at most, the 8 most recently calculated (λ, accumulated_free_energy) points will be used in approximating the $\partial G/\partial\lambda$ slope. IAVSLM=2 means that as soon as 2 points are available, we will calculate the slope from all available points, until the maximum of 8 is reached. If the best-fit line through the points fits the data with a correlation coefficient (CC) < 0.8, then the number of points used in the current slope determination will be halved, the slope and CC will be recalculated, and the comparison against CC will be performed again. A minimum of two points are always used to calculate the slope.

AMXMOV, which is set to 0.01 here, is the target change in free energy per window we are aiming for. The $\delta\lambda$ change on the next step is calculated as

$$\delta\lambda = \frac{AMXMOV}{(\partial G/\partial\lambda)} \tag{8}$$

Note that since we don't know *a priori* what the free energy versus λ curve will look like, we do not know exactly how many steps will be required to complete the simulation. The total number of MD steps required will depend both on AMXMOV and on NSTMEQ and NSTMUL (line 14). NSTLIM can be set to -1 on line 8 to force the program to continue until the total required number of steps have been performed. Also note that the value of AMXMOV used will often depend on the magnitude of the total anticipated free energy change. For example, one would not typically want to use AMXMOV=0.01 and NSTMEQ=NSTMUL=100 if the total energy change is 50 or 100 kcal/mole, as it can be for certain electrostatic changes.

line 14b:
IAVDEL < 0, which means that the $\Delta G_{forward} - \Delta G_{reverse}$ comparisons will not be used in scaling the widths of λ windows. The viability and reliability of changes made using these types of comparisons has not yet been established.

line 14c:
ALMDL0 is set to 1.0D-5. This means that the first IAVSLM window steps (before we have enough points to calculate a slope) will be made with this small step size. This step is chosen to be small in case the energy is changing quickly in this region.

DLMIN is set to 1.0D-10. Typically, a value of DLMIN such as this would have no effect, since it is unlikely that the slope and AMXMOV would be such to require a step this small in the first place. At any rate, steps calculated to be smaller than DLMIN are reset to DLMIN. DLMIN can be valuable in some cases when one wishes to limit how slowly a simulation can progress.

DLMAX is set to 1.0D-2. Setting an appropriate value for DLMAX is important. If the G versus λ curve has any points of inflection, we might calculate a slope of approximately 0 at one or more points. In this case, the simple formula used to determine the next step size would indicate a very large step (as large as 1.0, the whole simulation length). This would be incorrect, as the slope could clearly turn significantly non-0 in a future range of λ. DLMAX bounds the change in such cases.

AMXRST is set to 0.10. The slope we calculate is only an approximation of the "true" instantaneous slope, and the current slope is only an estimate of the slope over the next λ interval (window). Thus, it is possible that when we calculate the actual free energy change over the next window, it will be an unacceptable amount larger than the target value. In such a case, we may want to decrease the λ step size for this window and re-evaluate the energy. AMXRST is the largest allowable energy for a step. If the energy is > AMXRST, the $\delta\lambda$ stepsize is reduced, and the energy for the window recalculated. Note that setting AMXRST too close to AMXMOV will result not only in too many windows being reevaluated (inefficient), but can also lead to biased sampling.

ALMSTP(1) is set to -1.0. If 0 < ALMSTP(1) < 1.0, one can prescribe that the values of AMXMOV, DLMIN, DLMAX and AMXRST vary over different ranges in λ, as described in the input discussion.

V. Potential of Mean Force (PMF) calculations

It is often of interest to determine the free energy difference between two states which differ in conformation, rather than in composition. For example, one might be interested in the free energy profile for rotation about a ring in a protein. Such a profile can be determined by performing a PMF

simulation. To perform such a simulation, one must be able to define conformation as a function of lambda within the context of an otherwise free MD simulation. Fortunately, methods have been developed which allow selected internal coordinates to be constrained to chosen values, while otherwise affecting the MD trajectory only minimally. The best known of these is the SHAKE method for bond constraints. The methods of SHAKE have recently been extended to be generally applicable to angles and torsions[6].

GIBBS version 4.0 allows the user to define any chosen set of internal coordinates to be λ-dependent constraints. By setting NCORC=1 (line 9), one can calculate the free energy changes that accumulate as the internal constraints are translated from those of the initial state to those of the final state. If one graphs the free energy changes as a function of the restraint target values (themselves a function of λ), one gets the free energy profile for conformational changes.

Any constraint with a target value which is itself a function of λ will contribute to the free energy as lambda changes. This means that if SHAKE is used to constrain bonds of the perturbed group, and any of those bonds "grow" or "shrink" during the simulation, there will be a corresponding contribution to the free energy. In earlier work, this contribution has been overlooked, but we have shown that it must be included to reliably calculate free energies using the FEP method[7]. The contribution in such a case can be calculated simply by setting NCORC=1.

Constraints other than SHAKE-en bonds can be defined by setting INTR > 0 (line 14) and providing the definitions after the standard input (see above). Any internal coordinate can be used; Be aware, however, that any internal coordinate which is part of a closed ring will present a special set of (often tricky) considerations (see below). In typical use, no compositional (or topological) change is performed when a PMF simulation is being carried out. A GIBBS-format topology file is still required from PARM, though. An appropriate topology file with no atoms in the "perturbed group" can be generated by using the PERT option in PARM, but with no atoms defined in the pert group; i.e.

```
Title: Generic PERT topology with no atoms changing
BIN FOR STDA    PERT
    0    0    0
    0    0    0    0
PERTURBATION
No atoms change
END
END
```

In general, PMF calculations within GIBBS may be performed with any method – FEP, TI or slow growth. (Before version 4.1, only FEP could be used for PMF calculations.) Note that there is one scenario where *only* the TI (with "constraint forces") method may be used: when any constrained internals whose target values change with lambda lie within a closed loop. The loop can either be part of the molecular topology, *or as a result of the added topology of the constraint(s)*. To understand why neither FEP nor TI with "potential forces" can be used in such a case, you must recognize that for these latter methods, part of the procedure for calculating constraint contributions requires that we determine which atoms of the system are affected by a rigid body translation/rotation about the constrained internals. But the requisite set of atoms is not unambiguously defined when the constraint lies within a closed loop. Fortunately, the "constraint force" implementation of TI doesn't require that we make such a determination.

It is important to note that PMF calculations are typically very compute-intensive. For FEP, Gibbs will determine which non-bonded pairs have an interatomic distance which varies with one or more constraints, and only these are re-evaluated in determining $V_{\lambda(i+1)}$. This helps reduce the amount

of computer time required for a FEP simulation, although the total amount of time can remain high. The additional cpu overhead for calculating constraint energies with TI is negligible in all cases.

While we have a good error check for some torsional PMF's (the free energy values after rotating 360° should be the same), we typically have no reliable way of determining that for other simulations enough sampling has carried out to determine a converged PMF curve. Our best guard against spurious results is careful consideration of the specific problem and the inherent relaxation timescales of the surroundings.

VI. Error estimates and convergence

One of the thorniest issues related to free energy calculations is estimating the error in the results[7–9]. At present, this error is typically estimated in one of four ways:

(1) Two separate free energy simulations can be run, one with λ progressing from $0 \rightarrow 1$, the second with λ progressing from $1 \rightarrow 0$. These two calculations should yield the same free energy value, and the difference between them (the "hysteresis") gives a lower bound on the estimate in the calculation. Errors derived in this way often underestimate the actual error[7].

(2) The difference between "forward" and "backward" values for a single run. As described in the introduction, when FEP or slow growth is performed, double-wide sampling can be carried out. This ultimately results in two pseudo-independent values for the free energy, one calculated from the sum of all the $\lambda(i) \rightarrow \lambda(i+1)$ energies, and the other calculated from the sum of all the $\lambda(i) \rightarrow \lambda(i-1)$. If the results were exact, these values would be the same. In practice, they will not be, and their difference gives a crude lower bound on the inherent error. Error estimates derived in this manner tend to be even less reliable than those estimated using method (1), and are usually worthless for slow growth type runs[8].

(3) Two or more simulations are run under equivalent but different conditions. For example, staring with different randomly assigned sets of velocities. The difference between the free energies provide an estimate to inherent errors. These estimates are subject to the same problems as (1) above.

(4) A series of simulations is run which differ in the respective amounts of sampling done. For example, simulation 1 might use 100 steps of equilibration and 100 steps of data collection at each window, while simulation 2 used 200 steps of each. If the value from the shorter simulation was accurate, the value from the second simulation should be acceptably close to it. If it is not, the simulation must be run even longer to confirm convergence. This method probably provides the best insurance that convergence has been reached, but it is not definitive, and it is also the most costly.

It must be understood that none of the above methods allows a completely reliable error estimate. At best, they provide a *lower bound* on the error. A large apparent error is a good indication that the results obtained are not appropriately converged. But a low apparent error does not necessarily indicate a converged and accurate simulation. This is clearly shown in Reference (7).

VII. 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.

The improved mixing rules which can be used in GIBBS Version 4 (IOLEPS = 0, line 10) 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 not 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 corresponding to the $\lambda = 0$ endpoint, and the other corresponding to the $\lambda = 1$ endpoint. For the former 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 topologies would then never "see" each other at intermediate values of λ. Defining dual topologies can aid in performing free energy calculations where bond connectivities must change. Dual topologies is the method incorporated into the "CHARMM" program.

On an efficiency basis, the relative merits of the two methods have not been established. Additional discussion of the two methods can be found in Ref. (7).

References

(1) Straatsma, T.P., Berendsen, H.J.C. & Postma, J.P.M. (1986) *J. Chem. Phys.* **85**, 6720.

(2) Pearlman, D.A. & Kollman, P.A. (1989) *J. Chem. Phys.* **91**, 7831

(3) Pearlman, D.A. & Kollman, P.A. (1989) *J. Chem. Phys.* **90**, 2460.

(4) Fleischman, S.H. & Brooks, C.L. (1987) *J. Chem. Phys.* **87**, 3029.

(5) Yu, H.-A. & Karplus, M. (1988) *J. Chem. Phys.* **89**, 2366.

(6) Tobias D.J. & Brooks, C.L. III (1988) *J. Chem. Phys.* **89**, 5115.

(7) Pearlman, D.A. & Kollman, P.A. (1991) *J. Chem. Phys.* **94**, 4532.

(8) Pearlman, D.A. & Kollman, P.A. (1989) In: *Computer Simulation of Bimolecular Systems: Theoretical and Experimental Applications* (van Gunsteren, W. and Weiner, P.K, eds.), p. 101, Escom Science Publishers, Netherlands; van Gunsteren

(9) van Gunsteren, W., *ibid*, p 27.

ANAL

Usage:

```
anal [-O] -i analin -o analout -p prmtop -c inpcrd [-ref refc
                -r rmscrd -s compac -p1 pdb1 -p2 pdb2 -z zmat]
```

–O Overwrite output files if they exist.

This is the static analysis module of AMBER. Its purpose is to do energetic and structural analyses of static structures. This program should be run before MIN, in order to detect problems in model-built or xray derived structures. Typical problems that may be found this way are close nonbonded contacts and grossly distorted structure. A generic anal.in input file for this purpose is provided in the *templates* directory of the distribution.

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. The program puts those atoms which are not in explicitly defined groups into a separate group. In the case of a belly or partial minimization the unfrozen part of the system can be defined as the desired groups and the frozen part will be automatically taken by the program as an additional group.

The following options are available:

a) energy decomposition analysis between groups of atoms

b) sugar puckering analysis for DNA or RNA

c) helical parameter analysis for DNA or RNA

d) RMS fit between two structures of identical topology

e) compaction analysis between two structures

f) hydrogen bond analysis

g) output pdb coordinates

h) generate z-matrix for gaussian 80

i) calculate internal coordinates for the entire structure

j) calculate internal coordinates for specific dihedrals

On VMS systems files are assigned by Fortran unit number. These are given below along with a description of each file. Files shown in [] above are optional, others are mandatory.

Files:

analin	5	Input control data for the analysis run
analout	6	Output results
prmtop	20	Input Molecular topology file from PARM
inpcrd	21	Input Coordinates to be analyzed
refc	24	Input positional constraint coordinates
rmscrd	26	Input reference set of coordinates for RMS fit
compac	28	Input reference set of coordinates for compaction analysis
pdb1	33	Output pdb coordinates; these correspond to the coordinates in the inpcrd file, but will have been translated so that the origin is at the center of mass.
pdb2	35	Output pdb coordinates; these correspond to the coordinates in the rmscrd file, but will have been rotated and translated for best fit to the inpcrd coordinates. Hence, for a "typical" use of the RMS option, one would use inpcrd for the "original" (e.g. X-ray) coordinates, and rmscrd for the "final" coordinates. Then, if several simulations were compared to the same "original" structure, all of the "pdb1" files would be identical, and the various "pdb2" files would be rotated and translated more maximum fit to these original coordinates.
zmat	7	Output GAUSSIAN 80 zmatrix

Input description:

- 1 - TITLE

FORMAT(20A4)

TITLE Title for identification.

- 2 - NTX , NTXO , NRC , NRCX , NGRPX , KFORM

FORMAT(5I)

```
   NTX        Format of coordinates.

    = 1  Formatted (inpcrd, unit 21)
    = 2  Unformatted (inpcrd, unit 21)

   NTXO       Read but not used

   NRC        Option to read position constraints.

    = 0  no constraints
    = 1  constrained minimization

         The atoms to be constrained are read as groups with
         different harmonic force constants for each group.
         Consult the section on GROUP in the Appendices for
         group specification format.
         When using positional constraints, the constrained
         groups are given first in the group input followed
         by the groups for energy analysis.

   NRCX       Format of constraint coordinates.   The constraint
         coordinate file has the same organization as the structure
         coordinates.

    = 0  Formatted (refc, unit 24)
    = 1  Binary (refc, Unit 24)

   NGRPX      Maximum number of groups that the structure can
         be divided into for analysis

    = 0  Default = 70
    = N  Structure may be partitioned into N different groups

   KFORM      The Flag for the type of Topology File
    = 0  Binary    (prmtop, unit 20)
    = 1  Formatted (prmtop, unit 20)
```

```
   - 3 -      NTB , BOX(1) , BOX(2) , BOX(3) , BETA

              FORMAT(I,4F)

   NTB        Flag for periodic boundary conditions.
         ( not yet operational )

    =-n  Periodicity is applied.  Box is truncated octahedron
         (BETA = 90)
    = 0  No periodicity is applied
```

=+n Periodicity is applied. Box is rectangular or monoclinic depending on the value of BETA

BOX(1..3) Lengths of the edges of the periodic box

BETA Angle between the X- and Z- axes of the box in degrees. the Y- axis is assumed to be orthogonal to the other axes. (0 < BETA < 180)

- 4 - NTF , NTID , NTN , NTNB , NSNB , IDIEL

FORMAT(6I)

NTF Flag for force evaluation.

= 1 complete interaction is calculated

= 2 bond interactions involving H-atoms omitted

= 3 all the bond interactions are omitted

= 4 angle involving H-atoms and all bonds are omitted

= 5 all bond and angle interactions are omitted

= 6 dihedrals involving H-atoms and all bonds and all angle interactions are omitted

= 7 all bond, angle and dihedral interactions are omitted

= 8 all bond, angle, dihedral and non-bonded interactions are omitted

NTID Flag for improper dihedral contribution (read but not used).

NTN Read but not used

Note: non-bonded interactions are now always calculated using a residue based cutoff. The nb pairs are stored as residue pairs. This uses substantially less memory than the atom pairlist in the minimizer.

NTNB Read but not used

NSNB Read but not used

IDIEL Flag for the type of dielectric function to be used in

```
          calculating the electrostatic energy.

     = 0  distance dependent dielectric function
     = 1  constant dielectric function

-------------------------------------------------------------------------

   - 5 -      CUT , SCNB , SCEE , DIELC

              FORMAT(4F)

   CUT        The cutoff distance for the non-bonded pairs.


   SCNB       1-4 vdw interactions are divided by SCNB.
         if SCNB .le. 0.0 then SCNB = 2.0

   SCEE       1-4 electrostatic interactions are divided by SCEE
         if SCEE .le. 0.0 then SCEE = 2.0

   DIELC      Dielectric multiplicative constant for the electrostatic
         interactions.  If DIELC .le. 0.0 then DIELC = 1.0.  DIELC
         and IDIEL are coupled.  For example to obtain a dielectric
         'constant' of 4rij set DIELC=4 and IDIEL=0.

-------------------------------------------------------------------------

   - 6 -      Printout of energies beyond specified values.  You
         must use the ENERGY keyword to obtain output.

         IMAX , EMAX(I) , I = 1, 9

              FORMAT(I,9F)

   IMAX       Flag for printing the energy contributions.

     = 0  no printing
     = 1  energy contributions will be printed

   EMAX(1)    All the bonds whose energy contribution is greater
         than EMAX(1) will be printed.

   EMAX(2)    All the angles whose energy contribution is greater
         than EMAX(2) will be printed.

   EMAX(3)    All the dihedrals whose energy contribution is
         greater than EMAX(3) will be printed.

   EMAX(4)    All the 1-4 vdw whose energy contribution is greater
```

greater than EMAX(4) will be printed.

EMAX(5) All the 1-4 eel whose energy contribution is greater than EMAX(5) will be printed.

EMAX(6) All the vdw nb pairs whose energy contribution is greater than EMAX(6) will be printed.

EMAX(7) All the eel nb-pairs with absolute value of energy greater than EMAX(7) will be printed.

EMAX(8) All the H-bond pairs whose energy contribution is greater than EMAX(8) will be printed.

EMAX(9) All the constrained atoms whose energy contribution is greater than EMAX(9) will be printed.

INPUT FOR PROGRAM OPTIONS

- 7 - The control for doing the desired options. When a control word is encountered the program reads the needed input for that option, then looks for the next control word. This process continues until the word STOP is read.

IOPT

FORMAT(A)

IOPT The control word for the option.

'ENERGY' Energy decomposition into groups

'PUCKER' Sugar puckering analysis

'HELIX' DNA/RNA helical parameters analysis

'RMS' RMS fit between two structures of identical topology

'VOLUME' Compaction analysis

'HBOND' Hydrogen bond analysis

'GAUSS' Punch the Z-matrix for GAUSSIAN 80 (to unit 7)
Note: be careful to check the connectivity especially

for the first dihedral angle.

'PDB' Write the coordinates in pdb format (to unit 33)

'TELL' Output of internal coordinates for the entire system. (TELL in the edit module gives a more complete description)

'TORSION' Output of torsion angles only for specified torsions. (avoids the voluminous output of TELL)

'STOP' Control to terminate the run

GAUSS, PDB, and **TELL** require no further input. The following section describes additional input for those options that require it.

ENERGY

Parts of the molecule for which interaction energies are to be calculated are entered in GROUP format. See the section on GROUP in the Appendices for details. Groups are read sequentially in any order. Each group is terminated by an "END" card.

The ENERGY option is terminated by another "END" card.

PUCKER

PUCKER 1: NMIS

FORMAT(I)

NMIS The number of unique non-standard or modified bases in the DNA molecule. "Standard" means DNA from the Amber United Atom database. Anything else (including residues from the Amber all-atom database) is considered nonstandard. Important: If any of the standard names ADE, CYT, GUA, THY, or URA are used with other than standard united atom topology, make NMIS NEGATIVE. If any of the standard names are redefined, they all must be.

PUCKER 2: NAMBAS(I), NRA(I), (KRA(J,I), J = 1, 20), I = 1, abs(NMIS)

***** THIS CARD READ ONLY IF abs(NMIS) .GT. 0 *****

FORMAT(A,21I)

NAMBAS Residue name of the non-standard base.

NRA The number of sugar atoms in base (I) to be used for the puckering analysis.

KRA The atom numbers to be used, relative to the first atom in the base residue.

HELIX

HELIX 1: IBPGEN , NMIS , NBASP , NMISF , NFOSP

FORMAT(5I)

IBPGEN Flag for the base pair generation.

= 0 DNA is standard. This means that the DNA bases are from the AMBER united atom data base. The base pairing will be generated automatically

= 1 Non-standard DNA. The base pairing must be read explicitly from the following four parameters and lines below.

NMIS The number of types of non-standard or modified bases in the DNA molecule.
Important: If any of the standard names ADE, CYT, GUA, THY, or URA are used with other than standard united atom topology, make NMIS NEGATIVE. If any of the standard names are redefined, they all must be.

NBASP The total number of base pairs in the molecule if any nonstandard residues are present.

NMISF The number of types of non-standard or modified phosphate residues in the DNA molecule.

NFOSP The total number of phosphate pairs in the DNA molecule if any non-standard residues (either base or phosphate) are present.

HELIX 2: NAMBAS(I), NRA(I), (KRA(J,I), J = 1, 20), I = 1, abs(NMIS)

***** THIS CARD READ ONLY IF abs(NMIS) .GT. 0 *****

FORMAT(A,21I)

NAMBAS Residue name of the non-standard base.

NRA The number of atoms in base (I) to be used for calculating the mean plane of the base.

KRA The atom numbers to be used, relative to the first atom in the base residue.

HELIX 3: NAMF(I) , NFRA(I) , KFRA(I) , I = 1, NMISF

***** THIS CARD READ ONLY IF NMISF .GT. 0 *****

FORMAT(A,2I)

NAMF Residue name of the non-standard phosphate.

NFRA The number of atoms to be considered for finding the helical twist. (it is always 1 since the twist is taken as the angle between two phosphate pairs)

KFRA The relative position of the P atoms in the residue.

HELIX 4: KBASA(I) , KBASB(I) , I = 1, NBASP

***** THIS CARD READ ONLY IF NBASP .GT. 0 *****

FORMAT(2I)

Residue numbering for DNA analysis is sequential from the 5′ end of the first chain to the 3′ end, continuing from 5′ end of the second chain and ending at the 3′ end of the second chain. All bases must be paired in this way if NBASP = 0.

KBASA The residue number of the first base in a pair.

KBASB The residue number of the second base in a pair.

HELIX 5: KFA(I) , KFB(I) , I = 1, NFOSP

***** THIS CARD READ ONLY IF NFOSP .GT. 0 *****

FORMAT(2I)

See note on residue numbering above.

KFA The residue number of the first phosphate in a pair.

KFB The residue number of the second phosphate in a pair.

RMS

RMS 1: NTXP , FACT , JGROUP, IPDB , IMOVE

FORMAT(I,F,3I)

NTXP Format of the reference set of conformer coordinates to be read for the rms fit.

= 0 Formatted input
= 1 Unformatted input (same as the initial binary coordinates)

FACT The threshold for printing the deviation of individual atoms. The default is 0.02.

JGROUP
= 0 Compare all atoms for rms fit
= 1 Compare only those atoms read in GROUP format from unit 5

IPDB
= 0 No output of the rotated coordinate sets.
= 1 Output the rotated coordinate sets.

IMOVE
= 0 The molecules are NOT rotated to the principal axes
= 1 The molecules ARE oriented along the principal axes
NOTE: principal axis transformation is not required for RMS fitting.

VOLUME

```
   VOLUME 1:     NTXP

              FORMAT(I)

 NTXP       Flag for type of format of the reference coordinates
            of the conformer whose compaction is to be calculated.

       = 0  Formatted input
       = 1  Binary input (same as the initial binary coordinates).
```

HBOND

```
   HBOND 1:   CUTHB

             FORMAT(F)

   CUTHB      The cut off distance for chosing the hydrogen bond
         pairs.  The default is 4.0 Angstroms
```

TORSION

```
  TORSION 1:  Four atom names, free format.  All torsion angles between
         atoms with these names will be reported.  This card may
         be repeated.
```

```
  TORSION 2:  'END' to end TORSION input.
```

++++++ END OF INPUT ++++++

Rev A Revision by: George Seibel
ANAL 3.0 Authors: U.C. Singh, P.K. Weiner and S.J. Weiner
Director: P.A. Kollman
Department of Pharmaceutical Chemistry
School of Pharmacy
University of California
San Francisco CA 94143
Phone (415) 476 4637

MDANAL

Usage:

```
mdanal -i mdain  -o mdaout -p prmtop -a avgpdb
                    -g pltfil -c movcon -m movcrd
```

This molecular dynamics analysis program was written in Sept 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 is unchanged for version 4.0. In version 4.1 much of its functionality is duplicated by CARNAL. MDANAL is not supported and will be removed in a later release, so use of CARNAL is recommended.

In addition to the command-line file assignments, the trajectory files are assigned by Fortran unit number on Unix as well as other systems. On VMS systems all files are assigned by Fortran unit number. These are given below along with a description of each file. Assignment of Fortran units on Unix is described below.

```
file    unit        purpose

mdain     5   Input:   Control data for the run

mdaout    6   Output:  Results and diagnostics

prmtop   10   Input:   Molecular topology file from PARM

avgpdb    8   Output:  Averaged coordinates in PDB format

pltfil    7   Output:  correlation plots in "Curvy" format

movcon    2   Output:  Connectivity info for movies

movcrd    3   Output:  Coordinates for normal mode movies

          9   unit to store intermediate results for calculating
              different correlation time functions.

         11   unit to store subaverages.

     12 - N   files from which the coordinates of the dynamics
              run are to be read for averaging.  (the number of
              coordinate sets in each file is arbitrary).
```

The following csh script demonstrates the assignment of Fortran logical units on a Unix system. "ln –s" is analogous to "assign" on VMS. The trajectory files mdeq.trj and mdeq2.trj are linked to

fort.12 and fort.13, respectively. At the end of the run the links are removed with rm.

```
#!/bin/csh -f
#
#  mdanal demo:   cart. coord averaging
#
ln -s mdeq.trj fort.12
ln -s mdeq2.trj fort.13
mdanal -i mdan.in -o mdan.out -p crown.top -a crown.avg
/bin/rm fort.12 fort.13
```

CONTROL DATA -- UNIT 5 -- FREE FORMAT --

```
   - 1 -       TITLE

               FORMAT(20A4)

   TITLE       Title of the mdanal run for identification.
```

```
    - 2 -      INTX , ICOR , IDIM , ISUB , NFI , NFF , KFORM

               FORMAT(10I)

   INTX        Flag for the type of averaging.

    = 0  internal coordinates averaging
    = 1  position coordinates averaging
    = 2  special dna helical parameter averaging
    = 3  special dna sugar puckering averaging
    = 4  md movie generation
    = 5  normal mode movie generation (modes file is read on
         unit 12)
         Note:  movie display software is not part of the
                Amber package.

   ICOR        Flag for the calculation of correlation functions.

    = 0  no correlation functions
    = 1  correlation functions are calculated
    = 2  only correlation functions are calculated.  The output
         from averaging is suppressed
    = 3  only simple PAK cross correlation coefficient is
         calculated
    NOTE: for INTX.EQ.1 .OR. INTX.GT.3 ICOR is ignored

   IDIM      Flag to monitor dihedral transitions in case of
         dihedral averaging.
```

```
     = 0  no monitoring of dihedral transitions
     = 1  dihedral transitions are monitored

   ISUB       Flag to store sub averages.

     = 0  the sub-averages are not stored
     = 1  the sub-averages are stored

   NFI        Starting read unit for coordinates.
         The default is 12.

   NFF        Final read unit for coordinates.  If it is zero
         the default is 12.  The read unit is incremented
         by one from nfi to nff and the coordinates are read
         from different files.
     NOTE: if intx.eq.5 nfi and nff would be the same

   KFORM      The Flag for the type of Molecular Topology input
     = 0  BINARY FORM
     = 1  FORMATTED FORM

-----------------------------------------------------------------

   - 3 -      NTR , NRIS , NRRC , NSKP

              FORMAT(10I)

   NTR        Flag to read either coordinates or sub-averages for
         averaging.

     = 1  coordinates will be read through units NFI to NFF
     = 2  sub-averages are read through units NFI to NFF

   NRIS       The number of coordinate sets to be skipped for
         every set to be used for averaging.  The default is one.

   NRRC       not used

   NSKP       not used

-----------------------------------------------------------------

   - 4 -      NTB , BOX(1) , BOX(2) , BOX(3) , BETA

             FORMAT(I,4F)

   NTB        Flag for periodic boundary conditions.

     =-n  periodicity is applied.  box is truncated octahedron
          (BETA = 90)
     = 0  no periodicity is applied
     =+n  periodicity is applied.  box is rectangular or monoclinic
          depending on the value of BETA
```

```
   BOX(1..3)  Lengths of the edges of the periodic box.

   BETA       Angle between the x- and z- axes of the box in degrees.
         The y- axis is assumed to be orthogonal to the other
         axes. ( 0 < BETA < 180 )

-----------------------------------------------------------------

   - 5 -      NTQ , NTPI , NTPR , NTPL(1) , NTPL(2) , NLIS

              FORMAT(10I)

   NTQ        Flag for selecting an internal coordinate quantity Q
         for averaging.  If INTX .GT. 0 it is ignored.

    = 1  bond lengths involving hydrogen atoms
    = 2  bond lengths involving non-hydrogen atoms
    = 3  bond angles involving hydrogen atoms
    = 4  bond angle involving non-hydrogen atoms
    = 5  hydrogen bonds within 2.5 angstrom
    = 6  non-bonded distances involving given atom pairs
    = 7  dihedral angles involving at least one hydrogen atoms
    = 8  dihedral angles involving non-hydrogen atoms

   NTPI       Flag for the type of input coordinates.

    = 0  the coordinates are in binary form
    = 1  the coordinates are in formatted form


   NTPR       Flag for printing the averaged quantity.
         If INTX .eq. 1 the term "q values" means atom
         in the following.

    = 0  no results per q values are printed
    = 1  results per q values are printed according to the
         sequence
    = 2  in addition ordered per q value name and residue
         name are printed
    = 3  ordered per q value name and residue name following
         the list of quantities given below

   NTPL(1)    Flag for plotting the averages.

    = 0  no plotting
    = 1  the means of q values are plotted
    = 2  rms fluctuations per q value are plotted
         if INTX .eq. 1 rms fluctuation per atom is plotted
    = 3  in addition the third root of the third moments of
         the q values are plotted
         if INTX .eq. 1 the rms fluctuations per atom having
         atom name given in the next card are plotted
    = 4  in addition the fourth root of the fourth moments of
         the q values are plotted
```

NTPL(2) Same as NTPL(1) for the energies.
If INTX .eq. 1 it is ignored.

NLIS The number of q values or atoms to be read below over which the averages to be performed. If NLIS .eq. 0 all are taken.

- 6 - CUT , NAMA , NAMB

***** ONLY IF NTQ.EQ.6 .AND. INTX.EQ.0 *****

FORMAT(F,2A)

CUT The cut off distance for choosing pair distances.

NAMA The graph name of atom a.

NAMB The graph name of atom b. If NAMB is blank then NAMB = NAMA. The pair distances ab would be chosen for averaging.

- 7 - NTTR

***** ONLY IF INTX .EQ. 1 *****

FORMAT(I)

NTTR Flag for least square fit of the coordinates.

= 0 The least square fit are not made for the coordinate sets for performing the deviation analysis
= 1 The coordinate sets are translated and rotated to make a best fit with respect to the first set. The atoms to be included for the least square fit are input in GROUP format. See group.doc.

- 8 - LIS(I) , I = 1 , NLIS

***** ONLY IF NLIS .GT. 0 *****

FORMAT(10I)

LIS The list of q value or atom numbers over which the average is to be performed.

- 9 - DT , ITCOR , IFCOR , NCOR , NTPRC , IFFT

```
      ***** ONLY IF ICOR .GT. 0 *****

          FORMAT(F,12I)

   DT         The time interval between successive coordinate sets.

   ITCOR      Flag for the type of derived quantity F(Q) to be used
      for calculating the correlation functions.

    = 0  F(Q) = Q
    = 1  F(Q) = Q*Q
    = 2  F(Q) = COS(Q)
    = 3  F(Q) = (3*COS(Q)**2-1)/2

   IFCOR      Flag for the type of correlation functions to be used.

    = 0  <F(QI(0).F(QJ(T)>
    = 1  <COS(F(QI(0)-F(QJ(T))>

   NCOR       The number of q values for which correlation functions
      are to be calculated.

   NTPRC      Flag to plot the cf related functions.

    = 0  no plotting
    = 1  the normalized correlation function(nfc) is plotted
    = 2  in addition, the spectral density of nfc is plotted
    = 3  in addition, log(abs(nfc)) functions are plotted
    = 4  in addition, the integral of the nfc is plotted

   IFFT       Flag for the method of correlation function evaluation

    = 0  by direct multiplication
    = 1  by fast fourier transform

-------------------------------------------------------------------

   -10 -       IFAV , NPR , NPR2 , NINT , NSDM

      ***** ONLY IF ICOR .GT. 0 *****

            FORMAT(5I)

   IFAV       Flag for subtracting the mean of q from the q values.

    = 0  F(Q) = F(Q)
    = 1  F(Q) = F(Q-<Q>)

   NPR        The number of time intervals over which the nfc
      functions are to be plotted.  The default is the total
      number of points (NS).

   NPR2       The number of points of the spectral density to be
      plotted.  The default is NPR/NSDM.
```

NINT Number of points over which the nfc is integrated
The default is NPR.

NSDM The number of points of the spectral density that
are plotted as one point. The default is 1.

-11 - (IACOR(I) , JACOR(I)) , I = 1 , NCOR

***** ONLY IF ICOR .GT. 0 *****

FORMAT(10I)

IACOR(I) The q value sequence number for which correlation
function is to be calculated.

JACOR(I) If JACOR(I) .eq. IACOR(I)
auto correlation function is calculated.

If JACOR(I) .ne. IACOR(I)
cross correlation function is calculated.

NOTE: Five pairs are read per card

***** ONLY IF INTX .EQ. 2 *****

- 12A - IBPGEN , NMIS , NBASP , NMISF , NFOSP

FORMAT(5I)

IBPGEN Flag for the base pair generation.

= 0 DNA is standard. This means that the DNA
bases are from the AMBER data base. The base
pairing will be generated automatically.

= 1 Non-standard DNA. The base pairing must be read
explicitly from the following four parameters:

NMIS The number of types of non-standard or modified
bases in the DNA molecule. If NMIS.LT.0 the program
will assume that you wish to replace the hard-wired
values for ADE,THY,CYT,GUA,URA and will specify them
in the input on line 11B. In this case the absolute
value of NMIS is assumed to be the total number of base
types in the molecule. This option is useful for
analysis of all atom runs since the hard-wired base
types are united atom.

NBASP The total number of base pairs in the DNA molecule

if any nonstandard residues are present.

NMISF The number of types of non-standard or modified phosphate residues in the DNA molecule.

NFOSP The total number of phosphate pairs in the DNA molecule if any nonstandard residues are present.

- 12B - NAMBAS(I), NRA(I), (KRA(J,I), J = 1, 20), I = 1, NMIS

***** ONLY IF ABS(NMIS).GT.0 *****

FORMAT(A,21I)

NAMBAS Residue name of the non-standard base.

NRA The number of atoms to be considered for finding the mean plane of the base. If INTX .eq. 3, it is the number of sugar atoms to be considered for puckering analysis.

KRA The relative position of the atoms in the residue.

The relative position of the C1′ atom in the residue.

The relative position of the nitrogen in the base that is bonded to the C1′ atom specified above.

(Note that KRA is a list of all of the above so that the number of elements that you must specify for KRA is NRA+2 with the last two elements being the relative positions of the C1′ and the base nitrogen that it is bound to.)

- 12C - NAMF(I) , NFRA(I) , KFRA(I) , I = 1, NMISF

***** ONLY IF NMISF.GT.0 *****

FORMAT(A,2I)

NAMF Residue name of the non-standard phosphate.

NFRA The number of atoms to be considered for finding the helical twist. (it is always 1 since the twist is taken as the angle between two phosphate pairs)

KFRA The relative position of the P atoms in the residue.

- 12D - KBASA(I) , KBASB(I) , I = 1, NBASP

***** ONLY NBASP.GT.0 *****

FORMAT(2I)

Residues in nucleotide molecules are numbered sequentially starting at the 5′ end of the first chain to the 3′ end, continuing from the 5′ end of the second chain and ending at the 3′ end of the second chain. This numbering scheme is used regardless of the number of molecules in the original model. All bases must be paired in this way if nbasp .gt. 0.

KBASA The residue number of the first base in a pair.

KBASB The residue number of the second base in a pair.

- 12E - KFA(I) , KFB(I) , I = 1,NFOSP

***** ONLY IF NFOSP.GT.0 *****

FORMAT(2I)

see note on residue numbering above.

KFA The residue number of the first phosphate in a pair.

KFB The residue number of the second phosphate in a pair

***** ONLY IF INTX .EQ. 3 *****

- 13A - NMIS

FORMAT(I)

NMIS The number of types of non-standard or modified bases in the DNA molecule.

- 13B - NAMBAS(I) , NRA(I) , (KRA(J,I),J=1,20) , I = 1, NMIS

***** ONLY IF NMIS.GT.0 *****

FORMAT(A,22I)

NAMBAS Residue name of the non-standard base.

NRA The number of atoms to be considered for finding the mean plane of the base. If INTX .eq. 3, it is the number

```
            of sugar atoms to be considered for puckering analysis.

   KRA      The relative position of the atoms in the residue.

-----------------------------------------------------------------
```

+++++ END OF INPUT +++++

CARNAL

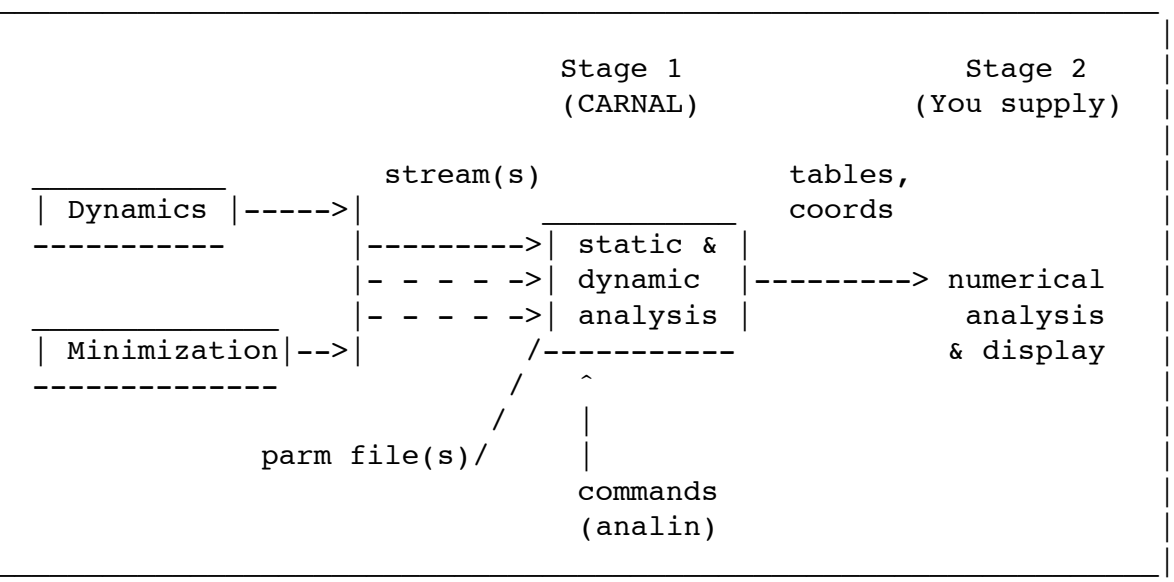
Wilson S. Ross
Department of Pharmaceutical Chemistry
University of California, San Francisco
ross@cgl.ucsf.edu
ross@ucsfcgl.bitnet

Usage:

```
carnal [-O] < analin > analout
carnal [-O] -i analin -o analout -p parm
```

–O Overwrite output files.

CARNAL is a new coordinate analysis program that allows comparisons between multiple streams of coordinates using a flexible command language. It has many of the capabilities of ANAL and MDANAL, which it is intended to eventually replace. It also provides set-theoretic group specification, cartesian vector oriented measurements, hbond analysis, output of distributions (including radial), selection of coordinate sets from streams, interpretations of md streams in terms of windows and format conversion.



CONTENTS

Introduction
Input
Output
Analin intro
Overview of analin sections
Simple analin example
Analin Syntax Specification
More complex capabilities
HBOND

DISTRIBUTION
Examples

INTRODUCTION

CARNAL input is essentially a programming language that lets one specify variables and perform operations on them. The control input file for CARNAL is referred to as *analin*. The commands in this file name the other input and output files as well as the measurements to be performed. These commands are described in detail in the syntax specification and examples below. The –o file or standard output contains carnal's interpretation of the analin input and summary data for the run.

Note that periodic boundary conditions are only applied to DISTRIBUTION DIST measurements. {CHANGE this eventually} This is because of difficulties in measuring across 2 streams, but could be enabled in the normal case of measuring within a stream.

Input

CARNAL takes as input one or more ''streams'' of input coordinates, which are lists of restart, mdcrd, or Amber pdb format files. The formats can be mixed in a stream, but the files must have the same number of atoms in the same order defined by the parm file for that stream. Each stream may have a different parm file. The –p argument or the first parm file defined in analin is used as the default parm file if no parm file is specified for a stream. At least one stream must be specified; the first one is used as the default when a stream reference is expected. CARNAL will also load static coordinate sets for comparison with the individual sets in the streams. It detects formats transparently. NOTE: periodic boundary conditions are only applied to DISTRIBUTION DIST measurements; this is because of difficulties in measuring across 2 streams, but could (and at some point will) be enabled in the normal case of measuring within a stream.

Output

CARNAL writes as output tables of measurements (scalar measurements or 0/1 hydrogen bond occupancies), distributions (including radial) and coordinate sets in mdcrd, restrt or pdb format. Summary data is also written to the file named with the –o argument on the command line or standard output if no –o argument is given.

ANALIN INTRODUCTION

This introduction is intended to give the feel of the analin language via an overview of the syntax and a simple example. A complete syntax definition and more complex examples are given below.

Summary of Analin Sections

These are the required sections in the analin input file syntax. Comments follow '!'s. In actual analin files, a '#' at the beginning of a line turns it into a comment. There are 4 main sections, each begun by a keyword in this order:

```
FILES_IN          ! name parm, coord/restrt/pdb files
FILES_OUT         ! name output tables, coord/restrt/pdb dumps
DECLARE           ! describe items to be measured
OUTPUT            ! direct declared stuff to output files
END               ! end input, start execution: STOP may be
                  ! substituted for debugging: program stops
```

Things are declared in the first 3 sections and referenced in the last 2 sections. When something is declared it is given an id for referencing it later.

A Simple Analin Example

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 /tmp/ke.p PDB;   ! keyword, id, filename, output format
DECLARE
                             ! no declarations 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
```

In this case, coord sets 1, 3, 5, and 200 from the concatenated stream of mdcoord files kecrd and kfcrd will be output to pdb files /tmp/ke.p.1 /tmp/ke.p.3 /tmp/ke.p.5 /tmp/ke.p.200.

Analin Syntax Specification

Notes

Aspects to be changed are indicated by 'CHANGE:'. Definitions must precede references: you cannot refer to something defined later in the file. Characters reserved for explicit purposes are:

```
# - . % ( ) & | , ! ?
```

A '#' as the first character of a line makes the line a comment. The format is entirely free, i.e. statements can be spread across multiple lines with any indentation and with comment lines embedded. Lines may not exceed 80 characters. In the syntax below, items in [] brackets are optional and items within {} braces are descriptions rather than token-by-token matchings.

---FIRST SECTION = "FILES_IN"

Input coordinates may be MD crd dump, inpcrd/restrt or Amber output pdb format.

FILES_IN

PARM id filename;

Amber Parm file. Multiple parms can be defined; the 1st one (or one defined by the [-p parm] argument) becomes the default for STREAMs that don't specify a parm.

STREAM strid [parmid]
[NOBOX] [ATOM n] [WIN x y] file1 file2 ... ;

At least 1 STREAM must be specified. STREAM files are read sequentially at each step. If > 1 STREAMs are named, they can be compared at each step. If no parmid is given, the first one defined is used by default. The NOBOX and ATOM options allow CARNAL to handle certain discrepancies between the parm topology and the input stream. If these options are inaccurate, synchronization may be lost, resulting in garbage. ATOM is for reading in a stream that has *fewer atoms that the prmtop* - such a stream might have been created earlier using the ATOM option in the COORD section. All coordinate sets in a stream must have the same number of atoms.

NOBOX
Mdcrd files for periodic simulations have box coordinates after each coordinate set. Carnal automatically detects the presence of periodic conditions from the parm topology and allows for reading the box coordinates in mdcrd. However, previous to 4.1, constant volume runs did not include the box coordinates. The NOBOX option allows carnal to read old constant volume mdcrd files correctly.

ATOM n
Read only n atoms (more may be defined in the parm file). I.e. coordinates for only n atoms are in the stream.

WIN
Means, "skip x sets, use y sets" repeatedly. This is for analysis of periodic equilibration / data collection runs such as gibbs.

STATIC statid [parmid] file1 file2 ... ;
STATIC files are read at the beginning and remain in memory for comparison with STREAM coordinates. Each static set in an id can be referenced by 'id%1',

---

---SECOND SECTION = "FILES_OUT"

FILES_OUT

TABLE tabid filename ;
In the tables, there is one "logical row" per coordinate set measured, so a given measurement over a trajectory occupies a column. For example, the Nth item directed to the table (in the OUTPUT section, below) would form the Nth numerical column and the Ith measurement of that value would be in the Ith logical row. The logical rows are wrapped so that a row continues through a series of lines in a single file beginning with key characters 'A' 'B' 'C' ... Users will extract a page of columns by grepping for the key letter. If there is demand, rows can instead be spread across multiple files (filename.0 filename.1 ...) or just tabbed continuously within a file (harder to read visually). Thus, the format is:

```
A    m1     m2     m3     m4     ...    |
```

```
B   m11   m12   m13   m14   ...   |   1st coord set
C   m21   m22                     |
A   m1    m2    m3    m4    ...    |
B   m11   m12   m13   m14   ...    |  2nd coord set
C   m21   m22                     |
```

where the measurements are m1, m2, ... m22 extending over a logical line consisting of lines 'A', 'B', and 'C'.

COORD crdid file [APPEND] [BLANK] [format] ;

APPEND
Add to end of named file if it exists.

BLANK
Write a blank line after each set.

Format symbols are 'PDB' 'RST' and 'CRD'. The default format is CRD.

HBOND hbid base_file [TABLE] [LIST];

TABLE
Output table of occupancies in base_file.tab. This table has two sections. The first part is a key that lists the possible hbonds in order. The format is:

```
# 1  (ADE  2 O5' )--(HB   1 H   ) .. (ADE  2 O1' )
# 2  (ADE  2 O5' )--(HB   1 H   ) .. (ADE  2 N7  )
```

The second part consists of a matrix of 0's and 1's. Each column is for a given hbond pair according to the numbering in the key section, and each row (line) is for a coordinate set. The format is '0' if no hbond is happening, '1' if it is. The Unix 'awk' utility can be used to extract column(s) of interest for further occupancy analysis or plotting, e.g. ''egrep -v '^#' | awk '{print $2, $5, $8}' base_file.tab'' where the 'egrep' command strips out the key section and the 'awk' command selects the columns of interest. Note that if there are too many columns for awk to handle, the 'perl' utility may be needed.

LIST
Output list of per-hbond-per-set to base_file.lis for extensive analysis. The format is:

```
1 ( ADE  2 N6   )( THY  5 O4   ) 2.930768 9.125721
2 ( ADE  2 N6   )( THY  5 O4   ) 2.957820 3.151730
```

where the 1st number is the number of the coordinate set (starting with 1), followed by donor, acceptor, distance and angle (in radians). Atoms are specified by residue name, residue number, and atom name. (See OUTPUT HBOND STAT for summary hbond info, including fraction of occupancy.)

HBOND specifications are given in the OUTPUT section.
Summary info is printed to standard output.
CHANGE someday: at least one of {TABLE, LIST} must be given, and the

OUTPUT HBOND statement is req'd to do any hbond analysis.

DISTRIBUTION dbid filename [DAP][MIN];

DAP

Put number of intervals on 1st line.

MIN

For DIST option, below. For each 'solvent' atom, write out min distance to 'solute' for the run (multiple lists separated by a'%' are output if WIN is chosen with DIST). This is to figure out which waters to keep in a second pass dumping COORDs. List output goes to filename.min. See Example #.

The definition of contents of the file is described in the OUTPUT section.

---THIRD SECTION = "DECLARE"

Each object is bound to a crd set; if not bound explicitly, the default is stream 0. References to that object inherit the binding of the object, except for within a GROUP statement (GROUP (GROUP id)...). I.e. in general an optional [crdset] is not allowed after a declared id is used (binding at reference rather than creation), for now.

'Points' can be atoms (specified by 'number [crdset]' or 'atom_name residue_number [crdset]') or centers of geometry or mass of groups of atoms (see GROUP definition below). For example, "PLANE id 12 34 58;" specifies the plane formed by atoms 12, 34 and 58 in the default stream, and "PLANE id OD1 2 ND1 4 OD1 6;" specifies the plane formed by atom name OD1 in residue 2, etc. "PLANE id gid1 gid2 gid3;" specifies the plane formed by the centers of geometry of three groups.

DECLARE

----Group is defined by set theoretic operations. Group attributes include center of geometry or mass, moment of inertia, and radius of gyration.

GROUP id (((set OP set) OP set) ...) [crdset];

Where the nesting in parentheses determines the order of evaluation.

OP can be either '&' or 'I'

where '&' = intersection, 'I' = union

and set can be any one of: { (ATOM numlist),
(ATOM [NAME|TYPE] namelist),
(RES numlist),
(RES NAME namelist),
(SOLUTE)
(GROUP namelist_of_groupids),
!set }

In the (GROUP) set, the groups are OR'd together. The NAME and TYPE options allow the use of '?' as a wild card matching any single character.

Allowing expressions:

groupid%center

center of geometry - default if groupid is used as a point

groupid%cmass

center of mass
groupid%momin
moment of inertia
groupid%radgyr
radius of gyration

AXIS id {2 points} ;
AXIS axid1 1 2 ;
Atoms 1 -> 2 in stream 0 by default.
AXIS axid1 1 st1id 2 st2id ;
Atom 1 in stream/static st1 -> atom 2 in stream/static st2.
AXIS axid2 grp1id grp2id%cmass;
grp1 center of geometry to grp2 center of mass: note that the groupids may be the same or groups may be defined on different streams.

PLANE plid { 3 points or 2 axes } ;
A plane is treated as its normal vector where appropriate.

ANGLE angid { 3 points or 2 axes/planes };
Planes are interpreted as normal vectors.

TORSION tid { 4 points or 3 axes/planes };
Planes are interpreted as normal vectors.

TORSION tid BACKBONE [residue1 [residue2]] [crdset] ;
Find all torsions involving backbone bonds (between Amber main chain atoms), starting with residue1 (default: 1st residue) and ending with residue2 (default: last residue). If first and last residues' terminal backbone atoms are bonded to each other, torsions involving them are included.

DIST dsid { 2 of: points, axes, planes };
Select 2 points, 2 axes, or point and axis or plane. [planes and axes are not supported yet]

RMS rmsid [FIT] groupid ;
RMS rmsid [FIT] groupid streamid ;
RMS rmsid [FIT] groupid streamid refcrdid ;
RMS rmsid groupid prevrmsid ;
Using atoms in groupid, measure rms of one coordinate set to another. If FIT is selected, position for minimum mass-weighted rms of the group, allowing rmsid to be used for RMS measurement of other groups and to be used like a stream for other measurements as well as to be output via a COORD statement. The first and simplest case above uses groupid to compare the default stream to its first set. The second case compares a named stream (rather than the default) to its first set. The third case specifies both the stream and a reference set for comparison; this reference set could be a static set or another stream. The fourth case measures the rms of a second group on a pair of sets that were positioned by a previous RMS FIT statement. See OUTPUT TABLE for instructions on obtaining per-residue and per-atom rms. Any number of any of these types of RMS measurements can be used.

PUCKER pukid NUCLEIC [streamid] [residue_names|residue_numbers] ;
PUCKER pukid number_of_points points ;
Measure pucker using algorithm of D. Cremer and J. A. Pople (JACS 96:6 pp

1354-1358, 1975). For the NUCLEIC options, the Altona and Sundaralingham convention (JACS 94 pp 8205-8212, 1972 or p. 20 of Saenger's "Principles of Nucleic Acid Structure", Springer-Verlag, 1983) is approximated by adding 90 degrees to the phase angle, the puckers are always ordered according to residue order in the parm file, and the standard atom names (O4'/O1', C1', C2', C3', C4') are used to determine the points. (For a comparison of different nucleic acid pucker conventions, see S. C. Harvey and M. Prabhakaran, JACS 108:20, pp 6128-6136, 1986.) In the general case (specifying points explicitly), a ring of N points can be parameterized by N-3 alternating amplitudes and phase angles. Note that the Cremer/Pople algorithm finds a mean plane based on the assumption that successive points in the ring have the same angle between them with respect to the center of geometry, so for kinky rings this may not work.

PUCKER pukid NUCLEIC;
: Measure pucker of all standard residues ('94 force field: G5, G, G3, GN, *etc*.; '91 force field: GUA, *etc*.) in default stream using standard atom names (O4'/O1', C1', C2', C3', C4').

PUCKER pukid NUCLEIC streamid;
: Same as above, but uses specific stream rather than default.

PUCKER pukid NUCLEIC GUA;
: Measure pucker of all residues named 'GUA' using standard atom names.

PUCKER pukid NUCLEIC 2,4,6,8 ;
: Measure pucker of residues 2,4,6, and 8 using standard atom names.

PUCKER pukid 5 O1' 2 C1' 2 C2' 2 C3' 2 C4' 2 ;
: Measure pucker of 5 points: O1' (residue 2), C1' (residue 2), etc.

---FOURTH SECTION = "OUTPUT"

Define what declares go to things defined in FILES_OUT.

OUTPUT

TABLE tbid { column_list } ;
: At least one column must be specified. Columns are printed in their order in the list. Column_list may include ids, classes of measurement (e.g. DIST) which print in the order declared, MEAS which prints all scalar measurements, or ALL which prints everything. AXIS ids result in vectors, PLANE ids in normals, and GROUP ids default to center of geometry unless attributes are specified, such as grpid%cmass. RMS ids default to the rms of the group, while rmsid%residues and rmsid%atoms give per-residue and per-atom rms respectively. For per-residue rms, the group must not have any partially-included residues. If either per-residue or per-atom options are used, the statistics are printed in the summary with the residue and atom names.

COORD crdid streamid
: [SELECT (-i j k,l m-p q-)] [MOD h]

[AVERAGE] [ATOM n] [EXH2O m [GROUP gid]] [INH2O gid] ;

SELECT (-5 7 8,10 100-105 200-)

Select certain sets from the stream by order. Numbers are separated by spaces or commas, and '-' is used to indicate ranges. In this example, select sets 1 through 5, then sets 7, 8, and 10, then sets 100 through 105, then sets 200 through the end. Note that this option selects files for output only, and does not affect measurements on the stream, as opposed to the STREAM WIN option, which pre-selects sets for all the other commands.

MOD h

Select every h'th set from the stream. Note that this option selects files for output only, and does not affect measurements on the stream, as opposed to the STREAM WIN option, which pre-selects sets for all the other commands.

AVERAGE

Average the coordinates. Not compatible with EXH2O or INH2O, but ok with ATOM. Produces a single set, so PDB or RST format is advised for the corresponding FILES_OUT COORD declaration. Suggested that this be applied to an RMS FIT streamid so that the area of interest has minimal distortion from drift or pressure scaling of the box.

ATOM n

Output only the first n atoms. ATOM, INH2O and EXH2O are mutually exclusive options.

EXH2O m [GROUP gid]

Omit all but m waters from the set, either those closest to the non-waters, or those closest to the atoms specified by GROUP. Distance is measured from water oxygen. Waters are printed in order of closeness to the solute, i.e. the order varies from set to set, so identity-based dynamic graphics smoothing schemes will fail. Cannot be used with INH2O.

INH2O gid

Omit all but waters with OW in group gid from the set, where gid contains only OWs. This group is intended to be built with the output of a previous pass using DISTRIBUTION MIN which informs the user how close each water came to the area of interest during the run. See Example #.

HBOND hbid [DONOR g1] [ACCEPTOR g2]

[DISTANCE x] [ANGLE y] [STATS];

DONOR and ACCEPTOR indicate group ids for searching for the appropriate atoms. The default for either is all atoms. A single group id may be given in place of separate definitions.

DISTANCE

Cutoff distance in angstroms between the heavy atoms: default is 4.0.

ANGLE

Cutoff H-donor-acceptor angle in degrees (0 is linear): default 1 radian ~= 60 degrees.

STATS

Directs printing of per-hbond summaries to standard output. The format is:

HBOND h1 stats:
19 (ADE 2 N6)_(ADE 2 HN6A)..(THY 5 O4) % 64.400000
distance avg: 2.909575 max 2.961379 min 0.000000
angle(deg) avg: 7.241544 max 15.219878 min 0.000000

where the # refers to the column of file.tab and the '64.400000' gives the percentage of occupancy. The other statistics are only for the ''occupied'' states.

DISTRIBUTION dsid
{ RAW | min max nboxes [WIN] }
{ measid | DIST group1 [SELF] [group2 [ALL]] [NORM] } ;

Distribution output can be either RAW (a long line of ascii floating point numbers per coordinate set) or binned and normalized. If the latter, the WINdow option causes the distribution for each n sets supplied by the STREAM to be written, with a '%' line to separate each window. RAW is the recommended option for large data sets, since the proper range and number of bins are hard to guess at: the rdis program can be used on the raw output to quickly try various min/max/nboxes numbers on the raw file. Note, however, that measurements including many terms can generate files larger than the original trajectory, so the RAW option may not be appropriate in such cases. For example, when measuring O-O distributions in a system of N waters, there are 9N numbers per coordinate set, but N(N-1)/2 distances to write out if RAW is used. For N=1000 this amounts to a RAW file 55 times larger than the trajectory.

Either an id for a scalar measurement or a radial distance distribution (DIST) may be specified. In the latter case, one or two groups can be specified. A single group may be given followed by the SELF option, in which case all intra-group distances are used (this would be appropriate for e.g. water O-O); otherwise, two groups are required. When just two groups are given (without SELF or ALL options), the groups are treated as ''solute'' and ''solvent'' respectively: for each ''solvent'' atom, the distance to the closest ''solute'' atom is applied. The SELF option includes the intra distances of the first group, and the ALL option includes all group1-group2 distances rather than the ''solvent'' to closest ''solute'' atom. Note that when two groups overlap, distances of 0 would be obtained for the atoms that are in both groups, so groups should be disjunct. SELF and ALL should make it possible to measure various forms of intra-solvent distributions (see examples, below). The MIN option from the FILES_OUT section above is only valid for the plain, two-group mode.

Volumetric NORMalization is optional when radial DISTribution is selected. This only makes sense when the solute group is unconnected, since the normalization is done by dividing the count for each interval by the volume of a spherical shell having radii equal to the shell boundaries.

The output format is: ''bin_center value smoothed_value integral.''

Examples

======Simple Coordinate Averaging

```
#Simple Coordinate Averaging
FILES_IN
  PARM   p1 ketop;           ! keyword, id, filename
  STREAM s1 kecrd kfcrd;     ! keyword, id, 2 filenames
FILES_OUT
  COORD  c1 /tmp/ke.p PDB;   ! keyword, id, filename, output format
DECLARE
                             ! no declarations for this simple case
OUTPUT
  COORD  c1 s1 AVERAGE;
                             ! keyword, files_out id, files_in id:
                             ! command to average sets
END

======Simple Distance, Angle, Torsion Measurements

Note the semicolons terminating each statement!

# Plain measurements involving points (atoms, centers of mass)
FILES_IN
  PARM p1 prm.top;
  STREAM s1 a1.trj a2.trj a3.trj;
FILES_OUT
  TABLE tab1 meas.tab;
DECLARE
#
#    First, some measurements using atoms only: format is:
#       FUNC_NAME ID atom_specs ;
#    each atom_spec can be either:
#       ATNAME RESNUM
#    or
#       ATNUM
#
  DIST dist1 O1' 2 O1' 7;
  ANGLE ang1 2 12 13;
  TORSION tor1 C1 4 C2 4 C3 4 C4 4;
#
#    A special case for torsions:
#
  TORSION tor2 BACKBONE;
#              ^ find all torsions consisting completely of
#                main chain atoms
#
#    Now some more geometrical stuff: the angle between
#    the normal vectors of two planes:
#
  PLANE pla1  C1 4 C2 4 C3 4;
  PLANE pla2  C1 5 C2 5 C3 5;
  ANGLE ang2  pla1 pla2;
```

```
#
#    Now some measurements using composite points:
#    format is the same, except for atom_specs.
#    First we'll define a group consisting of the non-waters,
#    and a group consisting of atoms in 3 numerically adjacent
#    residues:
#
  GROUP g1 (SOLUTE);
  GROUP g2 (RES 5,6,7);
#
#    And now to measure the distance between the centers of
#    mass of each group to see how that pair of residues
#    fluctuates from the center:
#
  DIST dist2 g1%cmass g2%cmass ;
#
#    Now let's define 2 more residue-based groups:
#
  GROUP g3 (RES 21,22,23);
  GROUP g4 (RES 44,45,46);
#
#    And we'll measure the angular fluctuations of the 3
#    residue-based centers of mass:
  ANGLE ang3 g2%cmass g3%cmass g4%cmass ;

OUTPUT
#    Now direct all the measurements to the table defined above
#    MEAS refers to all scalar measurements; each 1 will be a
#    column in the order defined (dist1 ang1 tor1 tor2 dist2 ang2).
#    Alternatively, the ids could be given explicitly in any order.
#
  TABLE tab1 MEAS;
END
```

======RMS deviation

You want a measurement of the minimum RMS deviation of a group of atoms as a measure of how disordered some structures are relative to another structure. Given the best fit on that group of atoms, you also want to know how much another subgroup differs and how much all the atoms outside the fit group differ. You also want to save the fit structures for viewing.

```
#
# RMS example: fit central 8 bases of a G4 DNA quadruplex
#
FILES_IN
  PARM p1 p524.top;
  STREAM s1
      sm4.pdb
```

```
      sm9.pdb
      sm17.pdb;
  STATIC ref_set  sm3.rst;
#                 ^^^^^^^ this file will be used as one
#                         reference set for the comparison;
#                         no pdb file around, but that's ok
FILES_OUT
  TABLE tbl  sm.rms;
#           ^^^^^^ this is a table for the per-set rms values
  COORD crd  fit.p  PDB;
#                   ^^^ save some structure(s) in PDB format
#           ^^^^^ name of file
DECLARE
#
#    Now let's get down to business.
#    Declare a group of atoms to fit on - all the
#    non-sugars in the central 8 GUAs:
#
  GROUP grp1 (  (ATOM NAME N9 C8 H8 N7 C5 C6 O6 N1 H1
                          C2 N2 HN2A HN2B N3 C4)
              & (RES 4,6, 13,15, 22,24, 31,33)  );
#             ^ boolean for "all of the atom names in
#               the 1st part that occur in the following
#               residue numbers"
#
#    RMS fit the structures in the stream to the reference
#    set using the group of atoms just defined (fitting is
#    mass-weighted). This creates a new thing that can be
#    treated as a stream.
#
  RMS fit1  FIT  grp1  s1 ref_set;
#                        ^^^^^^^ reference structure id -
#                                if not given, the first
#                                structure in the stream
#                                would be used; or this id
#                                could be for a different
#                                stream instead of the static
#                                coord set used here
#                      ^^ stream id - what to fit
#                ^^^ group of atoms to use for fitting
#           ^^^ fit (position) the stream set to minimize rms
#     ^^^ 'fit1' is the new, streamlike thing
#
#    Specify the group of atoms to measure a secondary
#    deviation - the terminal bases on each strand, w/out
#    the sugars:
#
  GROUP g2 ((RES 2,8, 11,17, 20,26, 29,35) &
          (ATOM NAME N9 C8 H8 N7 C5 C6 O6 N1 H1
```

```
                C2 N2 HN2A HN2B N3 C4));
#
#    Measure the RMS on that group resulting from the fit
#    of the other bases, i.e. between the target structure
#    and the new, fitted structure. Just measuring this
#    time, not creating a new set.
#
  RMS fit2  g2  fit1  ref_set;
#
#    Let's see what that fit does for _all_ the atoms outside
#    of the fit, not just the end bases.
#    Specify the group of all atoms not in the original group
#    used for the fitting:
#
  GROUP g3 (!grp1);
#
#    Measure the RMS of that group on the fitted structures
#    as before.
#
  RMS fit3  g3  fit1  ref_set;
#
#    Just for fun, create a new fitted set using the 1st group
#    but using the 1st set in the stream as reference.
#
  RMS fit4  FIT  grp1  s1;
#                      ^^ just specifying the stream with no
#                         reference defaults the reference to
#                         the 1st set in the stream
#               ^^^^ use our "central bases" group again
#          ^^^ FIT the thing
#     ^^^^ name of a new, stream-like thing starting with the
#          second crd set in the stream
#
OUTPUT
#
#    Write the RMS values to the table:
#
  TABLE tbl  fit1  fit2  fit3  fit4;
#            ^^^^  ^^^^  ^^^^  ^^^^ output the measured rms
#                                   values as columns in the
#                                   table declared as a file
#                                   above
#       ^^^ to table 'tbl'
#
#    Average the the structures fitted using the 1st group
#    on the reference set and print them to the coordinate
#    file defined above. Perhaps we will then energy min
#    this structure and claim it means something.
#
```

```
  COORD c1  fit1  AVERAGE;
#
END
```

======Coordinate Selection: waters: use of INH2O with DISTRIBUTION MIN

You want coordinate dump with selected waters. Not the closest waters at each step: you _know_exactly_ which waters you want: the same ones in every set, maybe so that when you smooth the trajectory, the Nth water won't change its identity at each step. This is a 2-pass procedure: first you need to get a list of the waters: you need the atom number of the OW in each water. If you want those waters to be all those that came within a given distance of the solute, have I got an option for you. The thing to have is, for each water, the closest it came to whatever you want to call the solute. DISTRIBUTION MIN will give you a list of atom number, distance pairs that you can sort to generate the list of favored waters you need. With this list of OW atom numbers, you can define a group which, in another pass, can be used to filter waters.

```
# use of INH2O with DISTRIBUTION MIN - 1st pass
FILES_IN
  PARM p1 prm.top;
  STREAM s1 a1.trj a2.trj a3.trj;
FILES_OUT
  DISTRIBUTION d1 file MIN;
#                       ^this is the key: creates "file.min"
DECLARE
  GROUP g1 (SOLUTE);
  GROUP g2 (ATOM TYPE OW);
#                     ^^ have to use OW
OUTPUT
  DISTRIBUTION d1 0.0 10.0 10 DIST g1 g2;
#                 ^^^^^^^^^^^             you'll also get the
#                                           net curve; RAW ok
#                            ^^^^         this is the 2nd key
#                                 ^^^^    order is solute then
#                                           solvent
END
```

--Now the critical step: filtering the water. First we use 'awk' to see what waters we want to keep, e.g.:

```
% awk '$2 < 3.0 {print $1}' file.min | wc -l
```

This tells you how many water oxygens came within the cutoff (3.0 in this example). Choose a cutoff such that the resulting list of OW atoms is the right size for you and:

```
% awk '$2 < 3.0 {print $1}' file.min > temp
```

Now you need to take the list of OWs in the temp file and include it in a GROUP

ATOM statement as in the following example in order to select the waters into a coordinate dump.

```
# use of INH2O with DISTRIBUTION MIN - 2nd pass
FILES_IN
  PARM p1 prm.top;
  STREAM s1 a1.trj a2.trj a3.trj;
FILES_OUT
  COORD c1 filtered.trj;
DECLARE
  GROUP g1 (ATOM 2,11,26,29);
#                ^^^^^^^^^^ the OW atom numbers of your choice
OUTPUT
  COORD c1 INH2O g1;
END
```

======DISTRIBUTION DIST (radial distance distributions) examples

```
# DISTRIBUTION EXAMPLE: ONE GROUP to itSELF (OW-OW)
FILES_IN
  PARM p1 prmtop;
  STREAM s1 crd;
FILES_OUT
  DISTRIBUTION d1 xdb ;
DECLARE
  GROUP g1 (ATOM TYPE OW);
OUTPUT
  DISTRIBUTION d1 RAW DIST g1 SELF;
#                              ^ use all intra-group distances
#                            ^ group id from DECLARE GROUP
#                       ^ distance macro
#                  ^ dump all distances to file for use w/ rdis
#                    program (i.e. don't bin measurements or
#                    output bins)
#              ^ id
#       For each 'solvent' group atom, the nearest 'solute' atom
#       is found and binned if it satisfies the min, max criterion.
END


# DISTRIBUTION EXAMPLE: TWO GROUPS using closest atom in 1st to
#                       each of 2nd
FILES_IN
  PARM p1 prmtop;
  STREAM s1 crd;
FILES_OUT
  DISTRIBUTION d1 xdb ;
DECLARE
  GROUP g1 (RES NAME ADE);
```

```
  GROUP g2 (RES NAME THY);
OUTPUT
  DISTRIBUTION d1 RAW DIST g1 g2 ;
#                              ^ 2nd group is the 'solvent'
#                            ^ 1st group is the 'solute'
#                       ^ distance macro
#                  ^ dump all distances to file (don't bin)
#              ^ id
#       For each 'solvent' group atom, the nearest 'solute' atom
#       is found and binned if it satisfies the min, max criterion.
END

# DISTRIBUTION EXAMPLE: TWO GROUPS using all inter-group pairs
FILES_IN
  PARM p1 prmtop;
  STREAM s1 crd;
FILES_OUT
  DISTRIBUTION d1 xdb ;
DECLARE
  GROUP g1 (RES NAME ADE);
  GROUP g2 (RES NAME THY);
OUTPUT
  DISTRIBUTION d1 RAW DIST g1 g2 ALL;
#                                  ^ consider all group1-group2
#                                    interactions
#                               ^ 2nd group id from DECLARE GROUP
#                             ^ 1st group id from DECLARE GROUP
#                       ^ distance macro
#                  ^ dump all distances to file (don't bin)
#              ^ id
END

# DISTRIBUTION EXAMPLE: TWO GROUPS using all group1 intra pairs and
#                       all inter group1-group2 pairs.
#                       This case: all distances between water O and
#                       other O (including water) in a water/octanol
#                       solution.
FILES_IN
  PARM p1 prmtop;
  STREAM s1 crd;
FILES_OUT
  DISTRIBUTION d1 xdb ;
DECLARE
  GROUP g1 (ATOM NAME OW);
#           ^^^^^^^^^^^^ all water oxygens
  GROUP g2 (ATOM TYPE OH);
#           ^^^^^^^^^^^^ all octanol oxygens
OUTPUT
  DISTRIBUTION d1 RAW DIST g1 SELF g2 ALL;
```

```
#                                        ^ consider all octanol-water
#                                          distances too
#                                     ^ group id from DECLARE GROUP
#                                       (octanol O)
#                                  ^ consider all water-water distances
#                              ^ group id from DECLARE GROUP (water O)
#                         ^ distance macro
#                     ^ dump all distances to file (don't bin)
#                ^ id
END
```

======HBOND examples

You want a occupancies for all possible hbonds at each step. This file will consist of a line for each coordinate set in the stream with a '0' or '1' followed by a space for each possible hbond. You also want the percentage occupancy of each hbond over the run, and the average distance and angle when occupied. And while you're at it, you want to print the distance and angle of each possible hbond.

You also want to specify the maximum distance and angle that qualify an hbond.

The percentages and averages are written to the main output at the end of the run.

```
FILES_IN
        PARM p1 hbtop;
        STREAM s1 hbmd;
FILES_OUT
        HBOND h1 xhb TABLE LIST;
#                           ^ write a list of distances/angles
#                             to "xhb.lis"
#                       ^ write occupancies to "xhb.tab"
#                  ^ use "xhb" as the basis for filenames
#
#
DECLARE
OUTPUT
        HBOND h1 DISTANCE 3.3 ANGLE 20.0 STATS;
#                                          ^ print the averages
#                                            of the occupied cases
#                               ^ limit the angle;
#                                 default 1 radian ~= 60 degrees
#                  ^ limit the distance between heavy atoms;
#                    default 4 Angstroms
END
```

Perhaps you want to specify the donor and acceptor groups, if only to limit the number of columns in the table. This time, we'll also just use the default criteria for hbonds.

```
# HBOND ANALYSIS EXAMPLE: USING GROUPS FOR DONOR/ACCEPTOR
FILES_IN
  PARM p1 hbtop;
  STREAM s1 hbmd;
FILES_OUT
  HBOND h1 xhb TABLE LIST;
DECLARE
DECLARE
  GROUP g1 (ATOM TYPE N2 NA);
  GROUP g2 (ATOM TYPE NC O );
OUTPUT
  HBOND h1 DONOR g1 ACCEPTOR g2 STATS;
END
```

====== Case history 1: a geometrical example

A long, relatively stiff molecule was bending: how to characterize it? One approach short of measuring the curvature (which carnal doesn't do yet) would be to define groups for different segments, then measure angles between vectors involving centers of mass or geometry; e.g. if this is the molecule:

```
        ======================================
        ======================================
        ======================================
        ^^^^^^          ^^^^^^          ^^^^^^
        grp1            grp2            grp3


  ANGLE a1 grp1%cmass grp2%cmass grp3%cmass;

or

        ======================================
        ======================================
        ======================================
        ^^^^^^ ^^^^^^             ^^^^^^ ^^^^^
        grp1   grp2               grp3   grp4

  AXIS ax1 grp1%cmass grp2%cmass;
  AXIS ax2 grp3%cmass grp4%cmass;
  ANGLE a1 ax1 ax2;
```

==== references

Batschelet, Edward. Circular statistics in Biology (1981) Academic Press Inc., New York, NY
Kabsch, (1976) *Acta Cryst*. **A32**, 922-923 and (1978) *Acta Cryst*. **A34**, 827-828.

NMODE

Usage:

```
nmode [-O] -i nmdin -o nmdout -c inpcrd -p prmtop -r restrt
       -ref refc -v vecs -l lmode -t tstate -e expfile
```

–O: Overwrite output files if they exist.

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. It is designed to read the *prmtop* and *inpcrd* files from the Amber suite of programs. There are accompanying programs *nmanal* (normal mode analysis) and *lmanal* (Langevin mode analysis) that use the output of these programs to compute molecular fluctuations and time correlation functions. *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 Research Institute of Scripps Clinic by J. Kottalam and D.A. Case. M. Pique has provided valuable advice and help in porting it to many different machines.

References. The second derivative routines are based on expressions used in the Consistent Force Field programs; [15] similar information is given by K.J. Miller, *et al*., [16] although these expressions were not actually used in writing this code. The code also contains routines to search for transition state, starting (generally) from a minimum. This procedure uses a modification of the procedure of Cerjan and Miller [17] as described elsewhere. [18] 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, [19] and were implemented in the Amber environment by us. [20]

General description: This program performs five tasks, depending on the value of the input variable ntrun (see below):

(1) Perform a normal mode analysis from starting coordinates. 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, thermodynamic parameters are calculated.

(2) Search for transition state, starting (generally) from a minimum. See the references above for a detailed description of the method.

---

[15] S.R. Niketic and K. Rasmussen, *The Consistent Force Field: A Documentation*, Springer-Verlag, 1977.

[16] R.J. Hinde and J. Anderson, *J. Comput. Chem.* **1989,** *10,* 63.

[17] C. Cerjan and W.H. Miller, *J. Chem. Phys.* **1981,** *75,* 2800.

[18] D.T. Nguyen and D.A. Case, *J. Phys. Chem.,* **1985,** *89,* 4020.

[19] G. Lamm and A. Szabo, *J. Chem. Phys.* **1986,** *85,* 7334.

[20] J. Kottalam and D.A. Case, *Biopolymers* **1990,** *29,* 1409-1421.

(3) Perform a conjugate gradient minimization from the starting coordinates. This routine uses an IMSL library routine for this purpose, which is not supplied with this program. Persons who do not have access to the IMSL library should probably use the AMBER "min" program to carry out conjugate gradient minimizations. (Compile min in the double precision version for best convergence.)

(4) Does a Newton-Raphson minimization from starting coordinates. A constant (tlamba) is added to the diagonal elements of the Hessian matrix to make it positive definite. Tlamba is chosen in a manner such that the step is always downhill in all directions. Whenever the change in energy is > emx or the rms of step length is > smx, the step length is scaled back repeatedly until the above two conditions are satisfied. Note that this routine will not converge to a transition state.

(5) 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.

Input files for this program are the same as for the regular AMBER minimization and molecular dynamics programs, with the exception of File 5, whose parameters are given below. The defaults have been carefully selected, so that for most purposes, few of them need to be changed. See the sample runs for more information.

Files used in the program:

```
nmdin  : control input for the run
nmdout : standard output file for print and error messages
prmtop : parameter file as output by the AMBER program parm
inpcrd : starting coordinates
refc   : input coordinates for constraints
restrt : output coordinates at end of minimization
prlist : file for reading or storing the non-bonded pair list
vecs   : file containing output normal mode frequencies and eigenvectors
tstate : output coordinates at a transition state
expfile: file to read exposed surface area for atoms
lmode  : file to write Langevin modes
```

Input found on *nmdin:* You can use as many title cards as you want, followed by the namelist `&data`, which contains the following variables.

General flags describing the calculation

---

ntrun 1: do normal mode analysis *(default)*
2: search for transition states
3: do conjugate gradient minimization
4: do Newton-Raphson minimization
5: do Langevin mode analysis

ibelly 1: some atoms are to be held fixed (default=0)

icons	1: do constrained minimization to initial coordinates specified in *refc*. (default=0)
maxcyc	max. number of cycles for minimization (default=100)
drms	rms gradient to stop minimization (default=1.e-5)
nvect	number of vectors for normal mode analysis (default=0)
nsave	for every nsave steps the coordinates are saved. (default= 20)
nprint	every nprint-th step the energy will be printed
ilevel	if .ne. 0, then adjust second derivative matrix to put rotation and translation vectors to a high frequency; this can be useful if you want to perform a normal mode analysis from a not-completely-minimized structure, so that rotations and translations don't mix with the low-lying modes (default=0).
ivform	0 if the normal mode eigenvectors are to be written out in unformattted form; 1 to use the formatted option (default).
ntx	0 if the input coordinates are to be read in unformattted form; 1 to use the formatted option (default).
ntxo	0 if the output (restart) coordinates are to be written out in unformattted form; 1 to use the formatted option (default).

Control of certain force field parameters

---

cut	radius for non-bonded cutoff (default=99.)
scnb	1-4 nonbonded scale factor (default=2.0)
scee	1-4 electrostatic scale factor (default=2.0)
dielc	dielectric constant (default=1.0)
idiel	0 for r**2 dielectric dependence (default); 1 for constant dielectric.
iprr	1: read in a non-bonded pair list from *prlist*; (default = 0)
iprw	1: write out non-bonded pair list to *prlist*; (default = 0)

---

control of Newton-Raphson and transition-state searches

---

smx	maximum rms step length (default = 0.08)
emx	maximum energy change per step (default =0.3)
alpha	scale factor for step length (default = 0.8) (See Nguyen and Case paper for description of smx, emx, and alpha.)
bdwnhl	constant to determine tlamba, the value to be subtracted from the diagonal elements of Hessian matrix for a downhill step. tlamba is chosen as min ((lowest eigenvalue - bdwnhl) , 0.00d0). (default bdwnhl = 0.25)

ndiag for every ndiag steps, the matrix is diagonalized to calculate tlamba, when ntrun=4

dfpred a rough estimate of the expected reduction in energy for the initial step (only for ntrun = 3). (default = 0.01 kcal/mol)

---

parameters for running Langevin modes (set ntrun = 5)

---

eta viscosity in centipoise

ioseen 0: Stokes Law used for hydrodynamic interaction
1: Oseen interaction included
2: Rotne-Prager correction included

hrmax hydrodynamic radius for the atom with largest area exposed to solvent. If a file named 'expfile' is present, then the relative exposed areas are read from that file as a namelist

```
namelist /exposure/ expr(natom)
```

If 'expfile' does not exist, then all atoms are assigned a hydrodynamic radius of hrmax.

---

parameters for transition state search (when ntrun = 2)

---

istart 0: new calc. (default)
1: restart calc.

iflag 0: search for transition state then minimum (default)
1: search for minimum from a transition state
-1: search for a transition state, then stop

ivect no. of eigenvectors wanted (default=2) (ivect has to be >=isdir)

isdir eigenvector along which search for transition state is to be made. Note that translations and rotations are removed from the Hessian, so this number refers to the ordering of the "true" vibrational normal modes. (default=1)

idir search direction: = 1 along isdir direction (default); = -1 opposite isdir direction

isw no. of steps before switching to the lowest mode (default=40)

hnot initial step length (default=0.1 Ang.)

buphl switch to Newton-Raphson step when lowest eigenvalue is less than this for uphill walk. (default=-0.1)

Cards 3 group cards for the parts of the molecule that move, if ibelly.ne.0. See group documentation for format.

Cards 4 group cards for the part of the molecule to be constrained, along with the constraint weights, if icons.ne.0. See group documentation of format.

NMANAL

Usage:

```
nmanal [-O] -i nmdin -o nmdout -p prmtop -v vecs -r rvecs
```

–O Overwrite output files if they exist.

This is a general routine to do vibrational analysis by projecting cartesian normal mode eigenvectors (generated by the nmode program) 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 that internal coordinate (the "potential energy distribution" for each mode.) You can also sum over all modes to obtain the rms thermal fluctuations for any particular internal coordinate.

The 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.

The original code was written by D. T. Nguyen and D. A. Case at U. C. Davis, 1985. The RMS analysis added by Barbara Rudolph. Capabilities for time correlation functions were added by J. Kottalam at Scripps Clinic. Responsibility for the final versions of the codes (and for any bugs) rests with Dave Case.

Files used in the program:

```
nmdin  : control input for the run, described below.
nmdout : standard output file for print and error messages
prmtop : formatted parameter file
vecs   : formatted file containing output normal mode
         frequencies and eigenvectors. This file is generated
         by the program nmode.
rvecs  : formatted file containing the reference eigenvectors
         and associated frequencies
```

Input found on nmdin:

Card 1: Title of the run

Cards 2: namelist /data/, which contains the following parameters:

ntrun Values of *ntrun* from –1 to 3 are used to analyze modes in terms of internal coordinates (if *ipro = 1*), to compute thermal fluctuations in internal coordinates (if *ifluc = 1*), or to compute time correlation and cross-correlation functions (if *ntrun = -1*). Options 4 and 5 are present only for historical reasons (although the code might be a good starting point for some interesting calculations), and options 6 to 8 carry out some specialized tasks: *(default=1)*

=-1 project eigenvectors onto those internal coordinates read in on subsequent cards (labelled "3c", below).
= 0 project eigenvectors onto bonds only
= 1 project eigenvectors onto all internal coordinates
= 2 " " " angles and dihedral angles
= 3 " " " dihedral angles only
= 4 project eigenvectors onto "dynamics groups"
= 5 project eigenvectors of the system('system vectors') onto reference eigenvectors. (This is a fairly specialized option, and you will probably have to look at the code to see what you really get. It has rarely been used.)
= 6 just calculate rms fluctuations in cartesian coordinates.
= 7 compute dipole-dipole correlation functions. In this case, prmtop is not read, and the &data namelist must be followed by cards that have two integers per card (free format), giving the atom numbers for each pair for which the correlation functions are desired. See subroutine "corf" for details of the calculational procedure.
= 8 project MD snapshots onto normal mode directions

nvect = number of eigenvectors in file vecs to be read in (default=50)

ivform = 0 if the input vectors are in unformatted form
= 1 for input vectors in formatted form (default)

ieff = 0 use true frequencies (default)
= 1 use effective frequencies (not implemented!)

pcut cutoff value for printing out projections: print will occur if the estimated contribution of an internal coordinate to the total potential energy distribution along this mode exceed pcut. (default = 0.02)

ibelly = 0 (default) no belly
= 1 belly calculation

ibeg first eigenvector to be analyzed (default = 7)

iend last eigenvector to be analyzed (default = 50)

ifluc = 0 don't do the rms internal crds. fluctuation(default)
= 1 do the rms internal crds. fluctuation for the internal coordinates selected by the "ntrun" variable

ipro = 0 don't print out the projections onto internal crds.
= 1 print out the projections onto int. crds (default)

bose true. if quantum (Bose) statistics are to be used in populating the modes; .false. (default) if classical (Boltzmann) statistics are to be used.

natom number of atoms; only needed if ntrun=7 or 8

ihsful = 0 if dipole-dipole correlations do not include contributions from distance fluctuations
=1 (default) if both distance and angle fluctuations are to be included in dipole-dipole correlations.

tmax maximum value for time correlation functions if *ntrun* = 7. Default is 0.0, i.e. no time correlations will be carried out.

tintvl interval for time correlation functions (default = 1.0).

The following are only used if ntrun=8:

first — first snapshot from MD simulation be be projected onto normal mode directions, when *ntrun* = 8. Default = 1.

last — last snapshot to be projected. Default = 9999.

iskip — every *iksip*-th snapshot will be projected. Default = 1.

The following are only used if ntrun=5:

nrgrp — number of rigid groups (default = 0)

nrvec — number of reference eigenvectors to be read in from file "rvecs" (only if ntrun=5; default=0)

nrat — number of atoms in reference system (default=0)

iat — first atom number of the part of the system to be excised and compared to reference system (only if ntrun = 5; default=1)

jat — last atom number of the part of the system to be excised (only if ntrun = 5; default = natom)

imov — flag to rotate/translate eigenvectors of system and reference to principal axes. This essentially decouples translation and rotation of the excised part of the system as a rigid body from the rest of the vibrational motion (only if ntrun=5, default=0).

Cards 3a — group cards for the parts of the molecule that move, only if ibelly.ne.0. See group documentation for format. This card is not needed when ibelly = 0.

Cards 3b — group cards for subdividing the molecule into rigid groups (if ntrun=4). See group documentation for format. Each rigid group will have its own set of cards 3b.

Cards 3c — *(if ntrun .eq. -1)* Quantities for which time correlation functions are to be calculated. These quantities are of the form of internal coordinates. All input is free format, which means that you must enter all numbers -- blanks are ignored.

```
TYPE, IAT, JAT, KAT, LAT
  INTNAME = identifier for this internal coordinate (character*8)
  IAT, JAT, KAT and LAT  are atom numbers.  Set LAT to
    zero for bonds and angle, KAT to zero for bonds.  Repeat this card
    for as many internal coordinates as you are interested in, up the
    the value of MAXINT specified in the "sizes.h" header file.
```

Input is terminated when the end-of-file is reached.

LMANAL

Usage:

```
lmanal [-O] -i lmdin -o lmdout -c inpcrd -l lmode
```

–O Overwrite output files if they exist.

This program will compute time correlation functions from Langevin modes. Note that since the time-independent aspects of the molecular normal mode description are independent 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 vacuum calculation. Hence the companion program *nmanal* should be used to compute those.

Input description for the *lmdin* file:

```
namelist   default    meaning
&data
 ntrun       1        'type of run' flag
                      1: correlation function calculated is
                         for the deviation of the length of
                         the vector from the reference value
                         in the minimum energy structure.
                         i.e., <dr(t)dr(0)> / <dr(0)dr(0)>
                      2: for the orientation of the vector
                         i.e., <P2[r(t).r(0)]>
                      3: the frequency distribution is plotted
                         i.e., the imaginary parts of the
                         eigenvalues are counted at every interval
                         of 10 wavenumbers. Other input parameters
                         are irrelevant.
 kup         1
 lup         2        atom numbers defining a position vector
 nvect      12        number of langevin modes to be used
 tf        2.0        final time for correlation functions
                      i.e., t ranges from 0.0 to tf picoseconds
 np       1000        number of points at which to calculate
                      correlation function between t = 0.0 and
                      t = tf ps.
bose      .false.    .true. if quantum (Bose) statistics are to be
                      used in populating the modes; =.false. (default)
                      if classical (Boltzmann) statistics are to be
                      used.
&end
```

The input file *inpcrd* is a standard Amber coordinate file; the file *lmode* is that created by the *nmode* program with ntrun = 5. Output files are CORF (if ntrun = 1 or 2) and DENS, CUMU (if ntrun = 3.) All of the output files (except *lmdout*) are input files for the <plot79> package, which will create plots of the correlation functions. You should(?) have little trouble converting them to some other plotting package in order to see the correlation functions.

Sample input file for nmode with ntrun=1

```
 get vibrational modes for staph nuclease ternary complex
 &data
     ntrun = 1,           do vibrational calculation
     cut=10.0,            cutoff; use same value in minimization
     idiel=0,             distance-dependent dielectric
     nvect=6753,          write out all 3*N modes...
     ivform=0,            ...in unformatted form...
     ilevel=0,            ...with no removal of trans. & rotation
     drms = 0.0001,       will complain if rms gradient is not
                             less than this
 &end
```

Sample input file for nmanal with ntrun=7

```
#
 Get N-H S**2 values from quasi-harmonic modes
 &data
  ntrun=7,                compute dipole-dipole correlation fns.
  nvect=4496,             this many modes in input file
  ibeg=1, iend=4496,      use all of the modes for the calculation
  ivform=0                unformatted modes files
  natom=1529,             molecule has this many atoms
  ihsful=0,               do no include distance fluctuations
 &end
3 4                       atom numbers for N and H of residue 1
11 12
20 21
28 29
38 39
```

NUCGEN

Usage: `nucgen [-O] -i ngin -o ngout -d ngdat -p pdbout`

–O Overwrite output files.

Purpose: This program generates cartesian coordinate models for either double helical DNA or RNA with a number of possible conformations. The helical topology of the double helix is stored in a file for individual types in terms of cylindrical coordinates. The program loads the required topoplogy and applies two fold symmetry with necessary helical repeat and height values. The cartesian coordinates are output in PDB format. The residue information is read as in the link module either for DNA or RNA. The input is described below.

NUCGEN requires specification of two strands: if only one is given, it will wrap it into two with highly stretched base-phosphate bonds across the end, so for single strands, specify a dummy strand and edit it out of the resulting PDB file. NUCGEN only generates reasonable geometries for complementary base pairs.

NUCGEN can generate PDB files using the 1994 Amber force field convention, which does not have explicit terminal hydrogen or phosphate residues. For the new residue names, only the bases need to be specified, while for the old convention, terminal hydrogen residues (HB and HE) and phosphates (POM) must also be specified. In the 1994 convention, residues are indicated by the first letter (A, G, C, T) and terminal residues have an additional `5' or `3' appended (*e.g.* A5, A3). See the LINK documentation for a table of these names. *The residue names in the input file must all be of either the old or the new convention – mixed conventions will not work.*

NOTE: the utility program NUKIT will generate NUCGEN and LINK input files for nucleic acids interactively;
Usage: ``% nukit''

On VMS systems files are assigned by Fortran unit number. These are given below along with a description of each file.

```
file    unit        purpose

ngin      5   Input:   Control and sequence data for the run

ngout     6   Output:  Diagnostics

ngdat     7   Input:   Monomer geometry file, found in amber41/dat
```

```
pdbout   10   Output:  PDB output coordinates
```

```
  Input data:   Unit 5  File  'ngin'
```

Nucleic Acid sequence information is given as described here for each strand. Both strands are entered in the 5' to 3' direction. This input is similar to LINK.

```
-------------------------------------------------------------

  - 1A -     A TITLE FOR EACH STRAND

                FORMAT(20A4)

  TITLE      A title for the molecule.

-------------------------------------------------------------

  - 1B -     ILBMOL

                FORMAT(A4)

  ILBMOL     Label for the type of molecule.

        'D'  DNA
        'R'  RNA

-------------------------------------------------------------

  - 1C -     RESIDUE INFORMATION FOR EACH STRAND
             it is read in the following format until a blank
             card is encounterd (card 1D).

             LBRES(I) , I = 1,NRESM

                FORMAT(16(A4,1X))

  LBRES(I)   Residue name.

-------------------------------------------------------------
  - 1D -     Blank Card to terminate residue input
-------------------------------------------------------------
  NOTE:  Cards 1A-1D are repeated for the second strand.
-------------------------------------------------------------
```

```
  - 2 -      KEND

                FORMAT(A4)

  KEND       Control to stop reading the nucleotide strands.

     'END '  end of reading the sequence information

-------------------------------------------------------------

  - 3 -      CONTROL FOR THE TYPE OF DNA OR RNA CONFORMATION

             TYPM

                 FORMAT(A8)

  TYPM       Name of the type of conformation to be generated.

   '$ARNA'   right handed a-rna (arnott)
   '$APRNA'  right handed a-prime rna (arnott)
   '$LBDNA'  right handed bdna (langridge)
   '$ABDNA'  right handed bdna (arnott)
   '$SBDNA'  left handed bdna (sasisekharan)
   '$ADNA'   right handed a-dna (arnott)
 '$SPECIAL'  special type by the user

-------------------------------------------------------------

  - 4 -       special helical parameter

              ***** only if typm .eq. '$SPECIAL' *****

              hxrep , hxht

                    format(2f10.5)

  hxrep       Helical repeat angle in degrees for the special
              type of conformation.

  hxht        Helical height.

              NOTE: If you use '$SPECIAL', you will have to
                    add the appropriate data to file ngdat (found
                    in the database directory).  Consult subroutine
                    gennuc for details.

-------------------------------------------------------------
```

++++++ END OF INPUT ++++++

Rev A Revision by: George Seibel
Authors: U.C. Singh, N. Pattabiraman, S.N. Rao
Director: P.A. Kollman
Department of Pharmaceutical Chemistry
School of Pharmacy
University of California
San Francisco CA 94143
Phone (415) 476 4637

ambpdb

NAME

ambpdb – convert amber-format coordinate files to pdb format

SYNOPSIS

ambpdb [-tit title] [-pqrl-bndl-atm] < *restrt* > *pdb*

DESCRIPTION

ambpdb is a filter to take a coordinate "restart" file from an AMBER dynamics or minimization run and prepare a pdb-format file. The program assumes that a *prmtop* file is available, from which it gets atom and residue names. The source code is in the `src/etc/` directory.

OPTIONS

–tit The title, if given, will be output as a REMARK at the top of the file. It should be protected by quotes or double quotes if it contains spaces or special characters.

-pqr Output will be in the format needed for the MEAD suite of programs created by Don Bashford.

–atm Output will be in the format used by Mike Connolly's surface area/volume programs.

–bnd Output is a file that lists the bonds in the molecule, one per line; this file is used by the "flex" program created by Mike Pique.

Users should consult the code and the input instructions for these other programs to see how these options are to be used.

FILES

There must be a *prmtop* file (with that name) in the current directory.

BUGS

Inevitably, various niceties of the Brookhaven format are not as well supported as they should be. The *protonate* program can be used to fix up hydrogen atom names.

protonate

NAME

protonate – add protons to a heavy-atom protein or DNA PDB file; convert proton names between various conventions; check (pro)-chirality.

SYNOPSIS

protonate [-k] [-b] [-d info_file] < *pdb* > *prot.pdb*

DESCRIPTION

Protonate combines a program originally written by K. Cross to add protons to a heavy-atom pdb file with many extensions by G.P. Gippert & D.A. Case. Names and descriptions of the output protons are contained in the info-file (see below.) *Protonate* can be used to add protons that don't exist, to change the names of existing protons to some new convention, and to check pro-chirality of protons in an input pdb file. The source code is in the `src/protonate/` directory.

OPTIONS

−k The output pdb file will ''keep'' the proton coordinates of the input file, to the extent consistent with how well it can identify what names they should ''really'' have. Otherwise it will replace input protons with ones it builds.

−b The program will insert a space before the name of each heavy atom in the output file. This is most often used to convert input files whose atom names begin in column 13 to the Brookhaven format where most heavy atom names begin in column 14. NOTE: ''two-letter'' heavy atom names (like FE or CA [calcium]) will not be correct; the resulting output file must be hand-edited to check for this.

−d info_file Specifies the file containing information on how to build and name protons. The default name is PROTON_INFO. This information used to determine where on the amino acids the protons should be placed. The file provided handles funny Amber residue names like HIE, HIP and HID and HEM. Other files provided include PROTON_INFO.Brook, which uses Brookhaven proton naming convention (such as 1HB, etc.), and PROTON_INFO.oldnames, which uses ''old'' amber names. For example, to take an Amber pdb file and convert to the Brookhaven naming convention, set ''-d PROTON_INFO.Brook''.

Output to STDERR includes ''matches'' of protons the program builds with any found in the input file, plus a list of any input protons that could not be matched. Questionable ''matches'' are flagged and should be checked manually.

BUGS

Format of the PROTON_INFO file is not obvious unless you have read the code.

Methyl protons are built in a staggered conformation; hydroxyl protons in a arbitrary (and generally sub-optimal) conformation. A program like *pol_h* or its equivalent should be used (if desired) to place polar hydrogens on LYS, SER, THR, and SER residues.

HIS in the input file is assumed to be HID. Users should generally explicitly figure out the desired protonation state for histidines.

No attempt is made to identify heavy atoms in the input file that have two-letter element names; this means that ''Brookhaven-style'' output may require some hand-

editing if atoms like calcium or iron are present.

It is assumed that the ‘‘alternate conformer’’ flag in column 17 of the PDB file is either blank, or ‘A’. The program needs to be recompiled to change this; perhaps this should become an input option.

gwh, cion and pol_h

NAME

gwh – Guess Water H: set positions of polar hydrogens onto water oxygen positions
pol_h – set positions of polar hydrogens in proteins
cion – suggest possible counterion positions

SYNOPSIS

gwh [-p *prmtop*] [-w *wat.pdb*] [-c] [-e] < *in.pdb* > *out.pdb*
cion [-p *prmtop*] [-w *wat.pdb*] [-c] [-e] < *in.pdb* > *out.pdb*
pol_h [-p *prmtop*] [-w *wat.pdb*] < *in.pdb* > *out.pdb*

DESCRIPTION

Gwh sets positions of water hydrogens onto water oxygen positions that may be present in PDB files, by optimizing simple electrostatic interactions. *Pol_H* resets positions of polar hydrogens of protein residues (Lys, Ser, Tyr and Thr), by optimizing simple electrostatic interactions. *Cion* suggests possible counterion positions. The source code is in the `src/protonate/` directory.

OPTIONS

−p prmtop The Parm topology file; default is ‘‘prmtop’’.

−w wat.pdb Read water oxygen positions from the file *wat.pdb*, rather than the default name *wat-pdb*.

−c (Gwh and cion only) A constant dielectric will be used to construct potentials, otherwise the (default) distant-dependent dielectric will be used.

−e (Gwh and cion only) The electrostatic potential will be used to determine which hydrogens are placed first; otherwise, a distance criterion will be used.

Accuracy of Pol_H & GuessWatH:

In the following the results for BPTI and RSA (ribosuclease A) are given together with those of Karplus [21] and Ornstein [22] groups. In the case of Ornstein’s method, it handles only some of hydrogens in question and therfore I normalized (scaled) their results using expected values for random generation. The rms deviation from the experimental positions (neutron difraction) and the number of hydrogens are shown below.

The accuracies seem to be similar among three approaches if scaled values of Ornstein’s data are considered.

BPTI	Lys	Ser	Tyr	Thr	Wat
# of H	12	1	4	3	112 (4˜)
Pol_H	0.39	0.36	1.08	0.20	0.98(0.38)

[21] A. T. Brunger and M. Karplus, *Proteins*, **4,** 148 (1988).

[22] M. B. Bass,,, R. L. Ornstein, *Proteins*, **12**, 266 (1992).

Karplus	0.25	0.71	0.81	0.19	– (0.35)
Ornstein	0.22	0.96	0.00	0.07	–
Ornstn(scaled)	0.51	0.96	1.28	0.07	(1.17)^

~internal waters. ^by random generation.

RSA	Lys	Ser	Tyr	Thr	Wat
# of H	30	15	6	10	256
GuesWatH	0.61	0.96	1.22	0.96	0.98
Karplus	0.60	0.98	0.60	1.12	1.20
Ornstein	0.20	0.61	0.60	0.30	–
Ornstn(scaled)	0.49	0.89	0.76	0.93	(1.14)^
-					

^by random generation.

FILES

in.pdb must have been generated by AMBER, e.g. through LEaP, EDIT, ambpdb, or the like: it must have exactly the same atoms (in the same order) as the prmtop file.

mdovrly

NAME

mdovrly – optimally overlay all coordinate sets in an mdcrd stream

SYNOPSIS

mdovrly [-c control_file] [-m mass_file] < mdcrdin > mdcrdout

DESCRIPTION

Note that carnal also provides this function; mdovrly may be deleted from future releases.

mdovrly is the first step of a three-step process to compute correlation functions from coordinate streams such as those created by AMBER dynamics simulations. [The subsequent steps are *mdextract* and *mdcorrp2*.] If you don't want or don't need to remove overall translation and rotation, you can skip this step of the sequence. The program basically acts as a filter, with some diagnostic information going to *stderr.* The source code is in the `src/nmr_aux/` directory.

OPTIONS

The *control_file* has one line, with five integers in free format: *natom, ntot, ncrd, ftyp,* and *irot*. The first two, *natom* and *ntot* should be the same, the number of atoms. [They are only distinct for historical reasons.] The maximum number of coordinate sets to use in given by *ncrd*; end-of-file on the input stream can also be used to end input. The options for *ftyp*, the file-type indicator are as follows:

2 – Amber coordinate stream, created with the NTWX variable
4 – Amber coordinate, or ".xyz" file; not usually used.
5 – "binpos" file, a binary standard used at Scripps

Other numbers have only historical uses. All of these i/o options are collected in the subroutine *fileio*, which you can examine for details, or to support other coordinate stream formats. Currently, the output file format is "binpos" format, which we find very useful: the graphics program *flex* reads this format, and they can be concatenated together. Finally, *irot* is 0 if no translation/rotation is to be done, is 1 if this is to be carried out by rotating the molecule so that its principal moments of inertia are along the coordinate axes, and is 2 if all coordinate sets are to be fit (in a least-squares sense) to the first coordinate set. Option "1" is probably the least desirable, albeit in some ways natural. Users should bear in mind that for flexible molecules, there is no rigorous way to separate overall and internal motions, and some fiddling with the code may be required. In many cases, (especially for relatively short trajectories) it may make little difference what you do here.

The *mass_file* contains *natom* floating-point values, one "mass" for each atom. Actually, these are really weights for overlays, and not necessarily masses. For example, if you wanted to fit on backbone heavy atoms, you could set their "mass" to 1.0, and everything else to 0.0. The simple scripts *pdb_to_mass* and *pdb_to_mass.bb* provide a convenient way to make this file.

FILES

Reads a control and a mass file, as described above; defaults are *ctrlin* and *mass*. Acts

as a filter on a coordinate stream.

SEE ALSO

mdextract, mdcorrp2

DIAGNOSTICS

Messages about what has been done are sent to *stderr*. It is typical to divert this to a log file to have a record of what sort of coordiante manipulations were done.

BUGS

Send bug-reports and comments to `case@scripps.edu`. Program is based on codes originally written by Art Palmer and Dave Case. The use of *control_file* might seem strange here, but the remaining programs in the series read an expanded version of this, so that a single control file can actually drive all steps of the process.

mdextract

NAME

mdextract – extract a set of interatomic vectors for correlation analysis from an mdcrd stream

SYNOPSIS

mdextract [-c control-file] < mdcrd > vector_file

DESCRIPTION

mdextract is the second step of a three-step process to compute correlation functions from coordinate streams such as those created by AMBER dynamics simulations. [The other steps are *mdovrly* and *mdcorrp2*.] The source code is in the `src/nmr_aux/` directory.

OPTIONS

The first line of the *control_file* has four integers in free format: *natom, ntot, ncrd,* and *ftyp*. The first two, *natom* and *ntot* should be the same, the number of atoms. [They are only distinct for historical reasons.] The maximum number of coordinate sets to use in given by *ncrd*; end-of-file on the input stream can also be used to end input. The file type indicator *ftyp* is defined as follows:

2 – Amber coordinate stream, created with the NTWX variable
4 – Amber coordinate, or ".xyz" file; not usually used.
5 – "binpos" file, a binary standard used at Scripps

Other numbers have only historical uses. All of these i/o options are collected in the subroutine *fileio*, which you can examine for details, or to support other coordinate stream formats.

The second card of *control_file* has a single integer, *nch*, which is the number of vectors to be computed. The third card contains three free-format floating point values: *tstep*, the time between snapshots; *tcorr* and *tmax*, which are ignored in this program.

The next *nch* cards contain a string and two integers, giving a name for the vector (maximum 10 characters) and the atom numbers of the two atoms involved.

FILES

Reads a control file, as described above; default if *ctrlin*. An input coordinate stream is on *stdin*, and the output vectors are placed into *stdout*. coordinate stream.

SEE ALSO

mdovrly, mdcorrp2

BUGS

Send bug-reports and comments to `case@scripps.edu`. Program is based on codes originally written by Art Palmer and Dave Case. The use of *control_file* might seem strange here, but the other programs in the series read an expanded version of this, so that a single control file can actually drive all steps of the process.

mdcorrp2

NAME

mdcorrp2 – take a set of vectors prepared by *mdextract*, and output correlation functions for NMR analysis.

SYNOPSIS

mdcorrp2 [-c control-file] < vector_file

DESCRIPTION

mdcorrp2 is the final step of a three-step process to compute correlation functions from coordinate streams such as those created by AMBER dynamics simulations. [The other steps are *mdovrly* and *mdextract*.] The source code is in the `src/nmr_aux/` directory.

OPTIONS

The first line of the *control_file* has four integers in free format: *natom, ntot, ncrd,* and *ftyp*. The first two, *natom* and *ntot* should be the same, the number of atoms. [They are only distinct for historical reasons.] The maximum number of coordinate sets to use in given by *ncrd*; end-of-file on the input stream can also be used to end input. The file type indicator *ftyp* is read but not used here.

The second card of *control_file* has a single integer, *nch*, which is the number of vectors in the input file. The third card contains three free-format floating point values: *tstep*, the time between snapshots; *tcorr*, the maximum time for which correlation functions are to be computed (should be at most 1/3 to 1/2 of the simulation length), and *tmax*, the maximum time in the simulation to be used. *Ncrd* and *tmax* can be set to large values for the usual case where the entire trajectory is to be analyzed.

The next *nch* cards contain a string and two integers, giving a name for the vector (maximum 10 characters) and the atom numbers of the two atoms involved.

FILES

Reads a control file, as described above; default is *ctrlin*. An input set of vectors is on *stdin*, and the output correlations are placed in separate files, using the name of each vector, followed by *.ocf*. These output files have comment lines (which begin with "#") that give overall statistics, then lines with four floating-point values, the first two of which are time and C(t), defined by <P2(0).P2(t)/r**3(0).r**3(t)>, normalized to unity at t=0, as described in the reference below. This file can be directly sent to many plotting programs such as *gnuplot* or *xvgr*.

SEE ALSO

mdovrly, mdextract. Algorithms are discussed in detail in Palmer & Case, *J. Am. Chem. Soc*. **114,** 9059-9067 (1992).

BUGS

Send bug-reports and comments to `case@scripps.edu`. Program is based on codes originally written by Art Palmer and Dave Case. The use of *control_file* might seem strange here, but the other programs in the series read an expanded version of this, so that a single control file can actually drive all steps of the process.

intense

NAME

intense – compute NOESY intensities from a structure

SYNOPSIS

intense -tauc *rot._corr._time,ns* -taum *mixing_time,sec*.
[-p *pdb_file* -i *output_intensity_file* -c *output_shift_skeleton*
-cutoff *cutoff* -s *smatrix_file*]

DESCRIPTION

Intense takes a structure from a pdb file as input, and outputs a list of NOESY intensities, suitable for input to other programs such as *spectrum*. The source code is in the `src/nmr_aux/` directory.

The input pdb file must include all hydrogens that you want to include in the spin system. If a phe or tyr residue exists, the order of the hydrogen names must be HD1, HE1, HZ (or HOH), HE2, HD2. IUPAC-IUB names for methyls are required for them to be properly identified. The command line also must include rotational correlation time and a mixing time.

All intensities greater the *cutoff* will be printed in the output intensity file. Default for *cutoff* is 0.0005.

If present, the S-matrix file will be read and used instead of computing distances from the pdb file; any matrix elements not present in the *smatfile* will be estimated from the distances in the pdb file. The format of the *smatfile* is a namelist "smat", containing the two-dimensional variable "s". Indices of s are the absolute proton numbers, i.e. those in the pdb file.

The output *cshfile* contains the atom names in the proper order for providing chemical shift information to the next program, *spectrum*. Edit this file to put the chemical shifts in the first 15 columns. If you have already put your chemical shift information into a database of the format support by Garry Gippert, then the script */case/nmr/spectrum/shiftconv* will take the skeleton file that *intense* makes and insert the proper shifts for you. See that shell script for instructions on using it.

Default file names are *pdbfile*, *intfile*, *smatfile*, and *cshfile*.

SEE ALSO

These programs are based on the "remarc" codes in Amber 4.0.

DIAGNOSTICS

File names, correlation and mixing times and number of protons are output to *stderr*. An error message is generated if the input pdb file has more than 750 protons or more than 1500 total atoms; the program needs to be recompiled after changing the appropriate variables in the "nmr.h" file.

BUGS

Format of the output file should be expanded to allow the parameters used to be embedded as comments.

If a tyrosine is present, the proton HOH must be present, even if this is a D20 simulation in which that proton has been exchanged away. The work-around is to include HOH, but with very large coordinates (e.g. 999.,999.,999) so that it won't contribute

to the spin systems. Other exchanged protons can be left out, or entered as "D...".

spectrum

NAME

spectrum – compute smx format file from the output of *intense*

SYNOPSIS

spectrum [-c *chemical-shift-file* -i *intense-file* -s *smx-file* -s1min *omega1-min* -s1max *omega1-max* -s2min *omega2-min* -s2max *omega2-max* -hwidth *half-width*]

DESCRIPTION

Spectrum takes the output of the *intense* program (*q.v.*) plus information on chemical shifts and produces an output "smx" format spectrum that ranges from s1min to s1max and from s2min to s2max. The source code is in the `src/nmr_aux/` directory.

Peaks are assigned a half-width given by hwidth. The defaults are 0 to 10 ppm in each direction, with a half width of 0.05 ppm. Default filenames are *cshfile*, *intfile* and *smx-file.smx*. The output file is a 512 x 512 real smx file that should be acceptable for viewing or processing by *ftnmr*. Since a square matrix is first set up, and then converted to the funny block smx format, it should be relatively easy to modify this program to accommodate other nmr analysis packages, or other plotting programs, etc. To accommodate the default contour levels in *ftnmr*, the peaks are multiplied by 10**7.

The program is currently configured for a maximum of 900 protons. This can easily be changed by modifying parameter statements at the beginning of the program.

SEE ALSO

intense
Sample calculation is in */case/nmr/spectrum/example*.

BUGS

Only 512 x 512 spectra can be output. This should not be too hard to fix with a code hack.

The width of the peaks must be the same in each direction, and the same for all peaks.

The header information that *ftnmr* uses is not fully documented, but spectrum will set up some of the important ones: it assumes a spectral frequency of 500.00 MHz in each dimension, sets up the proper spectral width and reference points (referenced at the edge of the spectrum) and sets the axis type to "ppm". A simple revision could make the spectrometer frequency an input variable.

rdis

NAME

rdis – calculate distribution curve of a series of numbers

SYNOPSIS

rdis min_val max_val n_points [radial] < input > output

DESCRIPTION

Rdis takes the *input* values in the range {min_val ... max_val} and counts the frequency of occurrence of values in each of n_points intervals (bins) in the range. The output is n_points rows of four columns of numbers: the (x-value) of the center of the bin; the normalized frequency of occurrence of input values in the bin; the smoothed frequency; and the integral of the frequency. In the default, the frequency is the number of cases in the bin divided by the total number of cases in the range. The source code is in the `src/carnal/etc/` directory.

With the *radial* option, a volumetric normalization is also applied that is only appropriate for radial distributions around a point. This hinges on the fact that equal increments of radius give unequal volumes (shells) at different distances from the center of a sphere. In this case, the normalization is such that the value at the average point density is 1. That is, the value for each bin in the measured range is

$$\frac{\text{(cases[bin] / Volume[bin])}}{\text{(total\_cases\_in\_range / Volume[range])}}$$

curvop

NAME

curvop – mathematical operations with curves

SYNOPSIS

curvop –{s|a|v|m|d} file1 file2 > output

DESCRIPTION

Curvop subtracts, adds, averages, divides or multiplies two curves represented by *x, y* values in *file1* and *file2*. The resulting curve is defined at the x-values in file1; if the x-values in file2 do not correspond, linear interpolation is performed between neighboring points in file2 to obtain corresponding values for the operation. Naturally, the curve resulting from the operation is only defined for the region of file1 that is covered by file2. The source code is in the `src/carnal/etc/` directory.

OPTIONS

–s	Subtract file1.y – file2.y.
–a	Add file1.y + file2.y.
–v	Average file1.y and file2.y.
–m	Multiply file1.y * file2.y.
–d	Divide file1.y / file2.y. If the Y value for file2 is 0, the program will stop with an error message.

curvemax

NAME

curvemax – print x, y of cumulative maximum Y-value

SYNOPSIS

curvemax < input > output

DESCRIPTION

Curvemax reads a file of X, Y values and prints the X, Y values at the cumulative maximum. *I.e.* printed Y value *i + 1* will always be greater than or equal to value *i*.

APPENDICES

Appendix A: Namelist Input Syntax

NAMELIST input is available on many host computers, including all of the machines on which AMBER is supported. It dates back to the earlier 1960's on the IBM 709, but is regrettably not part of Standard FORTRAN (either 1966 or 1977). It is a part of the proposed Fortran 8.X standard. NAMELIST input groups take the form:

```
&name
 var1=value, var2=value, var3(sub)=value,
 var4(sub,sub,sub)=value,value,
 var5=repeat*value,value,
&end
```

where the variables must be one of the names in the NAMELIST variable list. The order of the variables in the input list is of no significance, except that if a variable is specified more than once, later assignments may overwrite earlier ones. Blanks may occur anywhere in the input, except embedded in constants (other than string constants, where they count as ordinary characters). A comma (or the terminal &END) must follow each constant; end-of-line does NOT constitute a valid constant separator.

The NAMELIST name &name must ALWAYS begin in column 2 of an input record. The ampersand sign in &name and &end may optionally be replaced by a dollar sign, and the word "end" after the dollar sign may optionally be omitted. Column 1 of all input records is ignored, and only columns 2..80 are examined (in some implementations, a non-blank in the first column comments out the whole line). The terminal &end may occur anywhere in the input character stream (ignoring column 1 of course); it need not begin in column 2.

Letter case is ignored in all character comparisons, but case is preserved in string constants. An exception is that the namelist name itself must appear in lower case, e.g. *&cntrl*, not *&CNTRL*. String constants must be enclosed by single quotes ('). If the text string itself contains single quotes, indicate them by two consecutive single quotes, e.g. C1' becomes 'C1'" as a character string constant.

Scalar variables may NOT be subscripted, and must be followed by 0 or 1 constant.

Array variables may be subscripted or unsubscripted. An unsubscripted array variable is the same as if the subscript (1) had been specified. If a subscript list is given, it must have either one constant, or exactly as many as the number in the declared dimension of the array. Bounds checking is performed for ALL subscript positions, although if only one is given for a multi-dimension array, the check is against the entire array size, not against the first dimension. If more than one constant appears after an array assignment, the values go into successive locations of the array. It is NOT necessary to input all elements of an array.

There is one exception to the array specification syntax described above. For the most part, this only occurs in the case where the host machine does not adequately support namelist input, and the "portable" namelist routines supplied with AMBER are used. In this case, character arrays *must* be specified by explicit definition of each element of the array. E.g. ATNAME(1) = 'H', ATNAME(2) = 'O' will work, but ATNAME = 'H', 'O' will not. This exception to array specification should only occur for the "portable" namelist code, and only for *character* arrays (integer and real arrays can be

specified as described above). To determine whether the ‘‘portable’’ namelist code was used, check the value assigned to MACHINEFLAGS in the file amber41/src/MACHINE. If -DREGNML is not specified (e.g. if -DnoREGNML is specified instead), the ‘‘portable’’ namelist is being used.

Any constant may optionally be preceded by a positive (1,2,3,..) integer repeat factor, so that, for example, 25*3.1415 is equivalent to twenty-five successive values 3.1415. The repeat count separator, *, may be preceded and followed by 0 or more blanks. Valid LOGICAL constants are 0, F, .F., .FALSE., 1, T, .T., and .TRUE.; lower case versions of these also work.

Appendix B: GROUP Specification

Entering Group Information

This section describes the format used to define groups of atoms in various AMBER programs. In *sander*, a group can be specified as a movable "belly" while the other atoms are fixed absolutely in space, and/or a group of movable atoms can independently restrained (held by a potential) at their positions. In *anal*, groups can be defined for energy analysis.

Except in the analysis module where different groups of atoms are considered with different group numbers for energy decomposition, in all other places the groups of atoms defined are considered as marked atoms to be included for certain types of calculations. In the case of constrained minimization or dynamics, the atoms to be constrained are read as groups with a different weight for each group.

Reading of groups is performed by the routine RGROUP and you are advised to consult it if there is still some ambiguity in the documentation.

Input description:

```
 - 1 -       Title

           format(20a4)

 ITITL       Group title for identification.

      Setting ITITL = 'END' ends group input.
```

```
 - 1A -      Weight

      This line is only provided/read when using GROUP input to
      define restrained atoms.

           format(f)

 WT          The harmonic force constants in kcal/mol for the group
      of atoms for restraining to a reference position.
```

```
 - 1B -     Control to define the group

      KTYPG , (IGRP(I) , JGRP(I) , I = 1,7)

           format(a,14i)

 KTYPG      Type of atom selection performed.  A molecule can be
      defined by using only 'ATOM' or 'RES', or part of the
      molecule can be defined by 'ATOM' and part by 'RES'.
```

'ATOM' The group is defined in terms of atom numbers. The atom number list is given in igrp and jgrp.

'RES' The group is defined in terms of residue numbers. The residue number list is given in igrp and jgrp.

'FIND' This control is used to make additional conditions (apart from the 'ATOM' and 'RES' controls) which a given atom must satisfy to be included in the current group. The conditions are read in the next section (1C) and are terminated by a SEARCH card.

Note that the conditions defined by FIND filter any set(s) of atoms defined by the following ATOM/RES instructions. For example,

```
-- group input: select main chain atoms --
FIND
* * M *
SEARCH
RES 1 999
END
END
```

'END' End input for the current group. Followed by either another group definition (starting again with line 1 above), or by a second 'END' ''card'', which terminates all group input.

IGRP(I) , JGRP(I)

The atom or residue pointers. If ktypg .eq. 'ATOM' all atoms numbered from igrp(i) to jgrp(i) will be put into the current group. If ktypg .eq. 'RES' all atoms in the residues numbered from igrp(i) to jgrp(i) will be put into the current group. If igrp(i) = 0 the next control card is read.

It is not necessary to fill groups according to the numerical order of the residues. In other words, Group 1 could contain residues 40-95 of a protein, Group 2 could contain residues 1-40 and Group 3 could contain residues 96-105.

If ktypg .eq. 'RES', then associating a minus sign with igrp(i) will cause all residues igrp(i) through jgrp(i) to be placed in separate groups.

In the analysis modules, all atoms not explicitly defined as members of a group will be combined as a unit in the (n + 1) group, where the (n) group in the last defined group.

- 1C - Section to read atom characteristics

***** Read only if KTYPG = 'FIND' *****

JGRAPH(I) , JSYMBL(I) , JTREE(I) , JRESNM(I)

format(4a)

A series of ''filter'' specifications are read. Each filter consists of four fields (JGRAPH,JSYMBL,JTREE,JRESNM), and each filter is placed on a separate line. Filter specification is terminated by a line with JGRAPH = 'SEARCH'. A maximum of 10 filters may be specified for a single 'FIND' command.

The union of the ''filter'' specifications is applied to the atoms defined by the following ATOM/RES cards. I.e. if an atom satisfies any of the filters, it will be included in the current group. Otherwise, it is not included. For example, to select all non main chain atoms from residues 1 through 999:

```
-- group input: select non main chain atoms --
FIND
* * S *
* * B *
* * 3 *
* * E *
SEARCH
RES 1 999
END
END
```

'END' End input for the current group. Followed by either another
The four fields for each filter line are:

JGRAPH(I) The atom name of atom to be included. If this and the following three characteristics are satisfied the atom is included in the group. The wild card '*' may be used to to indicate that any atom name will satisfy the search.

JSYMBL(I) Amber atom type of atom to be included. The wild card '*' may be used to indicate that any atom type will satisfy the search.

JTREE(I) The tree name (M, S, B, 3, E) of the atom to be included. The wild card '*' may be used to indicate that any tree name will satisify the search.

JRESNM(I) The residue name to which the atom has to belong to be included in the group. The wild card '*' may be used to indicate that any residue name will satisify the search.

Examples:

The molecule 18-crown-6 will be used to illustrate the group options. This molecule is composed of six repeating (-CH2-O-CH2-) units. Let us suppose that one created three residues in the PREP unit: CRA, CRB, CRC. Each of these is a (-CH2-O-CH2-) moiety and they differ by their dihedral angles. In order to construct 18-crown-6, the residues CRA, CRB, CRC, CRB, CRC, CRB are linked together during the LINK module with the ring closure being between CRA(residue 1) and CRB(residue 6).

Input 1:

```
Title one
RES   1  5
END
Title two
RES   6
END
END
```

Output 1: Group 1 will contain residues 1 through 5 (CRA, CRB, CRC, CRB, CRC) and Group 2 will contain residue 6 (CRB).

Input 2:

```
Title one
RES   1  5
END
Title two
ATOM 36 42
END
END
```

Output 2: Group 1 will contain residues 1 through 5 (CRA, CRB, CRC, CRB, CRC) and Group 2 will contain atoms 36 through 42. Coincidentally, atoms 36 through 42 are also all the atoms in residue 6.

Input 3:

```
Title one
RES  -1  6
END
END
```

Output 3: Six groups will be created; Group 1: CRA, Group 2: CRB,..., Group 6: CRB.

Input 4:

```
Title one
FIND
O2 OS M CRA
SEARCH
RES   1  6
END
END
```

Output 4: Group 1 will contain those atoms with the atom name 'O2', atom type 'OS', tree name 'M' and residue name 'CRA'.

Input 5:

```
Title one
FIND
O2 OS * *
SEARCH
RES   1  6
END
END
```

Output 5: Group 1 will contain those atoms with the atom name 'O2', atom type 'OS', any tree name and any residue name.

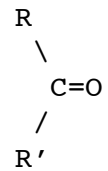
Appendix C: Parameter Development

How should one proceed to develop parameters for new molecules or fragments? The general principle is to use analogy as much as possible. The amount of effort that should be expended is related to the scientific question being asked. To accurately calculate thermodynamic interactions with water or a macromolecule, one needs the best parameters that can be obtained. If only qualitatively reasonable geometries are needed, less work may be required. Van der Waals, bond, angle, torsion and improper torsion parameters are discussed; the philosophy of derivation of specific atomic charges for a new residue is given in Appendix D.

Atom types. The first step in parameter development is to make a two-dimensional sketch of the fragment for which parameters are needed, and then to assign atom types to the atoms. The comments in the first section of the `parm94.dat` file describe the hybridization and other attributes of the atom types for the 1994 force field; [23] for the older force field, one may consult the Weiner *et al*. force field papers. [24] [25] This approach may be augmented by looking at the atom types in the existing residues in the files `all_*94.in`. For example, in pyridine the nitrogen would be assigned the same type (NC) as N1 and N3 in adenine. Note that an atom type is intrinsic to an array of distance, angle, and dihedral parameters involving the types of the neighboring atoms, as well as having its own van der Waals (VDW) parameters, and, of course, atomic mass (charge is not fixed per atom type). Therefore, if a new atom type is required, the first step is to attempt to reason by analogy and clone as many of the pre-existing parameters as possible to account for the environment of the new atom. Here it is instructive to consider the variability of the existing parameters, which tend to be duplicated over various combinations of atoms. This step may also be required if old atom types are used in a new topological relation.

For example, consider the oxygen of a sulfoxide or a sulfone:

We would expect the van der Waals parameters of this oxygen to be similar to those of a carbonyl oxygen of the force field (type 'O'):



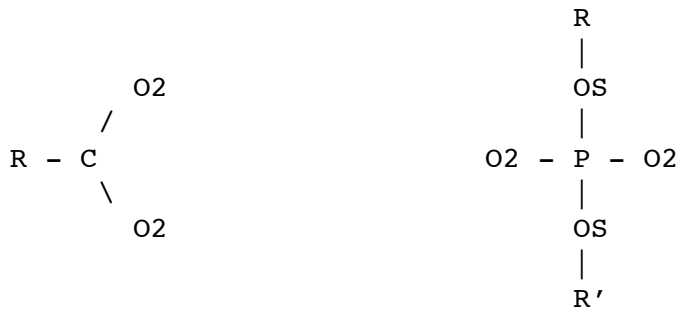
or to those of carboxyl or phosphate oxygens (type 'O2' in both examples)

---

[23] Cornell, W.D., Cieplak, P., Bayly, C.I., Gould, I.R., Merz, Jr. K.M., Ferguson, D.M., Spellmeyer, D.C., Fox, T., Caldwell, J.W., Kollman, P.A. *J. Am. Chem. Soc*. 117, 5179-5197 (1995).

[24] Weiner, S.J., Kollman, P.A., Case, D.A., Singh, U.C., Ghio, C., Alagona, G., Profeta, S., Jr., Weiner, P. *J. Am. Chem. Soc*. 106, 765-784 (1984).

[25] Weiner, S.J., Kollman, P.A., Nguyen, D.T., Case, D.A. *J. Comput. Chem*. 7, 230-252 (1986).



because VDW radii are dominated by the number of electrons in an atom and are not very sensitive to chemical environment. In fact, the VDW parameters for types 'O' and 'O2' are identical. Can one of these types be used for the oxygen in sulfoxide/sulfone? The environment must now be considered. If no suitable analogy can be found, a new atom type must be created and a complete set of bond, angle, and dihedral parameters for its neighbors added to the force field. In this case, both sulfoxide and sulfone oxygens are substituents of tetrahedral sulfurs rather than trigonal planar carbons, so the phosphate oxygen case makes an appealing analogy. We then check whether any existing bond, angle or dihedral parameters involve a S='O2' bond, and if they do, we check that those parameters are appropriate for *this* case of an S=O bond. But there are no such parameters – it is therefore reasonable to extend the use of type 'O2' for this case. (Had there been inappropriate S='O2' parameters, we might want to either make a new sulfur type, or make a new oxygen type starting with the VDW parameters of 'O' and 'O2'.) We continue discussion of bond and angle parameters for these example fragments below.

van der Waals parameters. What if no atom type lends itself to adaptation? When creating a new type, the first thing one must consider is VDW parameters. As illustrated above, for organic compounds these parameters may be straightforward to find by analogy based on element and bond order alone. Monoatomic ions, however, do not present such analogies in the AMBER force field and are discussed in more detail as an example.

The shape of the VDW potential for a given atom type is specified in terms of the distance between two atoms of the same type at the minimum energy point. Half the interatomic distance at that point is treated as the basic radius, or R*, parameter for that type. The form for the radial potential for two atoms is the sum of the R* values of their types. The potential well depth ('e') of the minimum energy point between two atoms of the same type is combined with the potential of another atom type by taking the root of the product. (Other parametric forms can be used which tend to have different type-type 'combining rules'.)

The simplest approach to deriving VDW parameters is to match a relevant experimental determination of the size of the atom in question. One source of such measurements is diffraction data. The sum of metal and oxygen Pauling radii [26] tends to be 3% smaller than indicated by water-ion neutron and X-ray diffraction data for Li+ and Na+ ions, 2% smaller than for K+, and 1% smaller than for Rb+ and Cs+. [27] Another source of ion 'size' information is crystallographic studies of ion complexes. [28] Since van der Waals parameters consist of two terms, the parameters that *e.g.* yield a given first peak of

[26] Pauling, L. The Nature of the Chemical Bond and the Structure of Molecules and Crystals; Cornell University Press: Ithaca, New York, 1960.

[27] Ross, W.S. and Hardin, C.C., *J. Am. Chem. Soc.* 116, 6070-6080 (1994). (This discussion of VDW parameters is based on that work.)

[28] Vedani, A., Huhta, D. W., *J. Am. Chem. Soc.* 112, 4759-4767 (1990) and references therein.

the radial distribution of the distance between two types of atoms are not unique. Another variety of experimental data that can contribute to parameterization is the free energy of solvation in water or another relevant solvent. However, it is still not clear whether the combination of experimental size and solvation free energy is sufficient to determine unique R* and 'e' parameters for an atom in relation to an existing type. A further complication arises because an atom type may come into contact with more than one other type, and nothing in principle guarantees that VDW parameters for a group of types can be fitted to yield uniformly correct pairwise potentials. Therefore it is important to choose parameters consistent with the most significant atom types that the new type will come in contact with. To a first approximation, atom types that tend to be oppositely charged, if present, are of most interest. In a particularly important case for ions, the TIP [29] water models (as well as some other waters) have a spherical van der Waals potential centered on the water oxygen (type 'OW'), which is somewhat inflated to enclose the hydrogen atoms in the molecule. Thus a cation that has been parameterized to give a correct ion-'OW' radial distance distribution function in such a water model will be ''smaller'' and come in closer contact with neighboring atoms if it is bound in a molecule consisting of AMBER atoms.

Moreover, remembering that different pairwise combining rules are in use in the modeling community, parameters from one convention must be adapted to yield the same results for a given pair of types in another scheme. Thus it was necessary to adapt the monovalent cation parameters of Åqvist [30] (found in `parm94.dat` and `parm91.dat`) for AMBER so that the ion-water *combined* potential gave the same optimal distance as with the combining rules used by Åqvist. Matching the ion size in the environment seems to be sufficient in the case with small monoatomic *monocations*; the default has traditionally [31] been to use a somewhat arbitrary well depth (epsilon) of 0.1 kcal/mol, characteristic of a rather nonpolarizable atom, and fit an R* parameter (see the Ross and Hardin reference). For the multivalent ions, different further approaches may be considered to capture the quasi-bonding electron mobility, including the use of explicit bonds or hydrogen bonding terms (see the Vedani and Huhta reference).

We have also discussed the derivation of van der Waals parameters for hydrogen in different bonding environments. [32] Using ab-initio calculations to study the interaction between water and various hydrogens, we found that a reduction in R* was required for hydrogens attached to carbons with adjacent electronegative atoms. This trend is nicely paralleled in the progessively smaller R* for hydrogens attached to carbon (HC), nitrogen (H), and water oxygen (HW).

Bond and angle parameters. Having chosen or created one or more atom types and sets of van der Waals parameters, the bond, angle and dihedral parameters must be created. Equilibrium bond lengths and angles may be obtained from tabulations of experimental data in the literature. [33] [34] Initial bond and angle force constants may be chosen based upon analogy to similar parameters in the force field or using the method of Hopfinger and Pearlstein. [35] See [36] for an example of extending the Weiner *et al*. force field to guanosine triphosphate and analogs. The AMBER-1994 force field

---

[29] Jorgensen, W.L., Chandreskhar, J., Madura, J.D., Impey, R.W., and Klein, M.L. *J. Chem. Phys.* 79, 926-935 (1982).

[30] Åqvist, J., *J. Phys. Chem.* 94, 8021-8024 (1990).

[31] Wipff, S. J., Weiner, P., Kollman, P. A. *J. Am. Chem. Soc.* 104, 3249 (1982).

[32] Veenstra, D. L., Ferguson, D. M., and Kollman, P. A. *J. Comput. Chem.* 13, 971-978 (1992).

[33] Allen, F. H., Kennard, O., Watson, D. G., Brammer, L., Orpen, A. G., and Taylor, R. *J. Chem. Soc. Perkin Trans. II*, S1-S19 (1987).

[34] Harmony, M. D., Laurie, R. W., Kuczkowski, R. L., Schwendemann, R. H., Ramsay, D. A., Lovas, F. J., Lafferty, W. J., and Maki, A. G. *J. Phys. Chem. Ref. Data*, 8, 619 (1979).

[35] A.J. Hopfinger and R.A. Pearlstein, *J. Comp. Chem.*, 5, 486 (1985).

[36] J.F. Cannon, *J. Comp. Chem.*, 14, 995-1005 (1993).

contains a limited number of unique bond and angle force constants and therefore selection by analogy is a feasible starting point. Returning to our sulfoxide/sulfone example, we find that the only existing bonds involving O2 are:

```
        Kbond     Rbond
C -O2   656.0     1.250       JCC,7,(1986),230; GLU,ASP
O2-P    525.0     1.480       JCC,7,(1986),230; NA PHOSPHATES
```

and it would be reasonable to use the O2-P force constant with a bond distance from the literature. Similarly, it would be reasonable to use the same angle bending parameters as for phosphates:

```
  Ktheta(O2-P-O2) = Ktheta(O-S-O)
  Ktheta(O2-P-OS) = Ktheta(R-S=O)
  Ktheta(OS-P-OS) = Ktheta(R-S-R)
```

Unless only crude parameters are desired, one should check the force constants by means of normal mode calculations if spectroscopic measurements are available for comparison; such calculations on N-methylacetamide are described in the Weiner *et al*. JACS 1984 paper. The bond and angle parameters are the primary determinants of the high and middle frequency vibrational modes of a molecule. For applications which require the highly accurate reproduction of vibrational frequencies, it is necessary to use a force field which includes higher order terms (anharmonicity) and cross-terms. For modeling the structures and interactions of molecules which are not highly strained, however, the simple harmonic approximation used in AMBER appears to be quite adequate.

Dihedral parameters. The dihedral parameters, in conjunction with the atomic charges and van der Waals parameters, are the primary determinants of the relative conformational energies of a molecule. The AMBER parameters IDIVF, PK, PN, and PHASE are used to define the torsional potential energy function. Each bonded series of atoms I-J-K-L must have at least one set of these dihedral parameters in the force field (just as every bonded pair I-J or triplet I-J-K must have bond or angle parameters, except that for dihedrals multiple terms may be used). The torsional energy function formula is:

$$Etors = (PK / IDIVF) * (1 + \cos(PN * phi - PHASE))$$

Let us look at a few examples in order to illustrate the nature of the dihedral parameters. For our first example (Figure 1), if atoms J and K are sp3 carbons (type CT) as in the molecule ethane (H3C-CH3), then the intrinsic barrier to rotation about the J-K bond is on the order of 3 kcal/mol. PK is equal to one-half of the barrier magnitude [37] and would therefore be equal to 1.5 kcal/mol. The topology about the dihedral of interest has a three-fold periodicity (PN), that is, there are three potential barriers as the C-C bond is rotated -180 to 180 degrees. These barriers occur when the methyl hydrogens eclipse each other: at 0, -120, and 120 degrees. Since the dihedral formula is a Fourier series truncated to a single cosine term, no phase shift would be needed to reproduce the potential energy barriers and PHASE = 0 degrees. (PHASE = 0 degrees if an energy *maximum* is at 0 degrees; PHASE = 180 degrees if an energy *minimum* is at 0 degrees.) The final dihedral parameter that must be specified is the number of torsions associated with the central bond (IDIVF), which is the product of the number of substituents on the two central atoms. For ethane, this is 3x3 = 9. So we have:

```
        PK    = 3.0 kcal/mol / 2.0 = 1.5 kcal/mol
        PN    = 3
        PHASE = 0.0 degrees
```

[37] Because the (1 + cos()) term ranges from 0..2, it is scaled to 0..1 by including the divisor of 2 in the energy term, PK.

Figure 1

```
IDIVF = 9
```

These same torsional parameters can be used for n-butane, and the results are in good agreement with experiment and higher-level calculations for the relative energy of *trans* and *gauche* minima and *cis* and *skew* energy barriers.

Consider now the molecule ethylene, H2C=CH2, whose dihedral potential energy is shown in Figure 2.

Figure 2

The lowest-energy conformation of this molecule is planar with a two-fold (PN = 2), 60 kcal/mol (PK = 30.0 kcal/mol) barrier to rotation about the C=C bond. The barriers are found at dihedral angles of -90 and 90 degrees (energy minimum at 0 degrees), and can be reproduced by the truncated Fourier series only if a phase shift of 180 degrees (PHASE = 180.0 degrees) is used.

```
PK    = 60.0 kcal/mol / 2.0 = 30.0 kcal/mol
IDIVF = 4
PN    = 2
PHASE = 180.0
```

Finally, we examine a hypothetical molecule ZH2C-CH2Z, where Z represents an electronegative functional group. Let us imagine that we either have experimental data on the relative conformational energies or we have simulated the rotational potential of this molecule with a series of quantum mechanical calculations. In practice, this is only done for minimum and maximum energy conformations – *trans*, *gauche+*, *gauche-*, *eclipsed*, *skew*, etc. In our example, the energy profile shows that the *trans* conformation (Z-C-C-Z = 180 degrees) is about 0.5 kcal/mol less stable than the *gauche*.

Before fitting the torsional parameters, we must generate the energy profile for the molecular mechanical nonbonded potential as was done for the quantum potential, subtract this curve from the quantum curve, and fit the torsional potential to the difference potential.

Before these calculations can be done, atomic charges need to be calculated, also by fitting to quantum mechanical results. The difference potential is then deconvoluted into Fourier series terms (Figure 3) which give the force field parameters:

	IDIVF	PK	PHASE	PN
Z-CT-CT-Z	1	0.260	0	-3
Z-CT-CT-Z	1	0.384	0	-2
Z-CT-CT-Z	1	0.241	0	1

which result in the total torsional potential shown in Figure 4. (In AMBER, PN is set to less than zero when additional terms remain to be read.)

Care must be taken when deconvoluting the torsional potential not to introduce spurious minima or maxima into the rotational energy profile. The combined potential of the deconvoluted parameters can be plotted directly by a graphing program, or the torsional energy profile can be 'empirically' generated at 20-30 degree intervals in AMBER.

In practice, such an elaborate Fourier series treatment may not be appropriate because (a) the quantum mechanical treatment may not be accurate enough to warrant it, (b) one would rather have a simpler torsional potential that is more consistent with the existing force field and (c) the electrostatic potential fitting procedure may capture the torsional energy profile well enough so that many terms are not needed. For example, in the case of 1,2-difluoroethane, the known gauche tendency of the fluorines can be simulated by adding a twofold torsion

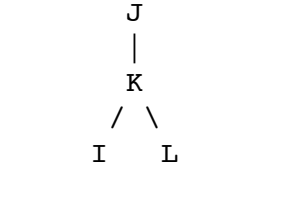
	IDIVF	PK	PHASE	PN
F-CT-CT-F	1	X	0	2

with 'X' adjusted to make the total molecular mechanical energy of the *gauche* conformation 1 kcal/mol lower than the *trans* conformation.

Improper torsions. Improper torsions are so named because the atoms involved are not serially bonded; rather they are branched:

Figure 3

Figure 4



Improper I-J-**K**-L

The convention is that the central atom is listed in the third position of the dihedral ('K' in the figure). Improper dihedral potentials are sometimes necessary to reproduce out-of-plane bending frequencies, *i.e*. they keep four atoms properly trigonal planar for a two-fold torsional potential (PN=2). They are additionally used in the united-atom force field model when a carbon with an implicit hydrogen is a chiral center; in effect they keep the position from inverting (PN=3).

The PHASE for improper torsions is always 180 degrees. Improper torsional parameters listed in the force field file they can use ''wild-card'' specifications ('X') for the non-central atoms (note that wild-card impropers must follow the explicit ones in the parm.dat force field file). When using the Prep-Link-Edit-Parm ('PLEP') programs to create coordinate and parameter input files, the improper torsions must also be specified by atom name in the Prep residue input file. In LEaP, every atom with three substituents is matched against the impropers in the force field file, and all matches are applied (discarding any wild-card terms if an explicit match is found). Thus care must be taken that a new improper does not inadvertently match other cases. In both PLEP and LEaP, an improper with no wild cards causes all wild-card-containing impropers to be ignored. Except for not mixing wild-card with explicit cases, all improper terms that match a given central atom are applied. In LEaP, if no match is found, no improper term is applied (unlike bonds, angles and 'proper' torsions, for which parameters *must* exist). In PLEP, a parameter must be in parm.dat if there is an improper in the Prep input.

Hydrogen bonding parameters. Unlike the previous AMBER force fields, the 1994 force field does not include a 10-12 hydrogen bonding function. This function, however, is still supported by the software. When using the hydrogen bonding function, all relevant pairs of atom types need to have parameters. Note that if a pair of atom types has H-bond (10-12) parameters, these will override any van der Waals (6-12) parameters for that pair.

Atomic charges. To generate atomic ('point' or 'partial') charges for a new fragment, the electrostatic potential around the fragment is first calculated at a grid of points using quantum or semiempirical methods, then charges at the atom centers are fit to reproduce the potential using the RESP (Restrained ElectroStatic Potential) program. The amount of effort applied can vary considerably; multiple conformations and even families of analogs can be considered, as has been done with the proteins and nucleic acids in the AMBER-1994 [38] force field. The charges for the AMBER-1994 force field have been calculated at the 6-31G* basis set level of *ab initio* theory and to maintain consistency, this basis set should be used when calculating charges for a new molecule. (See Appendix D regarding the charge-fitting philosophy.)

It is important to energy minimize the fragment at a suitable level of theory before calculating the electrostatic potential around it. Quantum mechanical geometry optimizations were considered to be too expensive for the amino acid charges in the AMBER-1994 force field. Rather, the RESP charges were calculated after first minimizing the geometries using the Weiner *et al*. force field. MM2/MM3-minimized geometries would also be reasonable for examples when a quantum mechanical optimization is deemed too expensive. In general, the charges are likely to vary more as a function of conformation than degree of optimization. This is illustrated by the results obtained for the alanyl and glycyl dipeptides (Cornell *et al*. paper).

[38] Cornell, W.D., Cieplak, P., Bayly, C.I., Gould, I.R., Merz, Jr. K.M., Ferguson, D.M., Spellmeyer, D.C., Fox, T., Caldwell, J.W., Kollman, P.A. *J. Am. Chem. Soc*. 117, 5179-5197 (1995).

Appendix D: Charge fitting philosophy

Wendy Cornell

The philosophy of the Kollman group (AMBER) has been that the accurate representation of electrostatic interactions is crucial for a force field intended for application to biological molecules. [39]

We note that the choice of a particular force field should depend on the system properties one is interested in. Some applications require more refined force fields than others. Moreover, there should be a balance between the levels of accuracy or refinement of different parts of a molecular model. Otherwise the computing effort put into a very detailed and accurate part of the calculations may easily be wasted due to the distorting effect of the cruder parts of the model. [40]

In other words, a force field which has a complicated potential form for representing bonds and angles and is very precise in terms of reproducing geometries and vibrational frequencies will not accurately model complex intermolecular interactions if the charge model is not also of high quality.

The new charges which were developed for the 1994 force field are called RESP charges, for Restrained ElectroStatic Potential fit. This modification of the original ESP method was developed by Christopher Bayly, who was a postdoc in the group. [41]

The basic idea with electrostatic potential fit charges is that a least squares fitting algorithm is used to derive a set of atom-centered point charges which best reproduce the electrostatic potential of the molecule. In the AMBER charge fitting programs, the potential is evaluated at a large number of points defined by 4 shells of surfaces at 1.4, 1.6, 1.8, and 2.0 times the VDW radii. These distances have been shown to be appropriate for deriving charges which reproduce typical intermolecular interactions (energies and distances). The dipole moment of the molecule is well reproduced.

Other programs have embedded the molecule in a cubic grid of points to evaluate the potential. We believe that assigning the points along the contours of the molecule provides a reasonable sampling of the esp around each atom.

The value of the electrostatic potential at each grid point is calculated from the quantum mechanical wavefunction. The charges derived using this procedure are basis set dependent. For example, the Weiner *et al*. force field employs STO-3G based charges, whereas the new Cornell *et al*. 1994 force field uses charges derived using the 6-31G* basis set. The 6-31G* basis set is bigger and, for the most part, ''better.'' Because quantum mechanics calculations scale as the number of basis functions to about the 2.7 power (HF as implemented in Gaussian92), the bigger 6-31G* basis set was prohibitively large for use in developing the earlier 1984/1986 force field.

The 6-31G* basis set tends to result in dipole moments which are 10-20% larger than gas phase. This behavior is desirable for deriving charges to be used for condensed phase simulations within an effective two-body additive model, where polarization is being represented implicitly. In other words a molecule is expected to be more polarized in condensed phase vs. gas phase due to many body

---

[39] Cieplak, P., Cornell, W.D., Bayly, C., and Kollman, P., ''Application of the Multimolecule and Multiconformation RESP Methodology to Biopolymers: Charge Derivation for DNA, RNA, and Proteins'' J Comp Chem, in press.

[40] van Gunsteren, W.F and Berendsen, H.J.C., *Angew. Chem. Int. Ed. Engl*. 29, 992 (1990) – a lucid and succinct review of MD applications to chemistry.

[41] ''A Well-Behaved Electrostatic Potential Based Method Using Charge Restraints For Determining Atom-Centered Charges: The RESP Model,'' C.I. Bayly, P. Cieplak, W.D. Cornell, and P.A. Kollman, *J. Phys. Chem*. **1993,** *97,* 10269.
''Application of RESP Charges to Calculate Conformational Energies, Hydrogen Bond Energies, and Free Energies of Solvation,'' W.D. Cornell, P. Cieplak, C.I. Bayly, and P.A. Kollman, *J. Am. Chem. Soc*. **1993,** *115,* 9620.

interactions, so we ''pre-polarize'' the charges.

A study by St-Amant *et al*. [42] calculated DFT charges for a number of small molecules and found them to be smaller than HF/6-31G* derived ones. DFT charges for methanol did not reproduce the relative free energy of solvation of methanol. Such charges may be more appropriate for use with a non-additive model, since the DFT model reproduced the gas phase dipole moments very well.

ESP fit charges have many advantages. They reproduce interaction energies well. They can be calculated in a straightforward fashion. They have been shown to perform well at reproducing conformational energies when used with an appropriate 1-4 electrostatic scale factor. The Cornell *et al*. JACS paper provides much of the validation of our new charge model. A study by Howard, Cieplak, and Kollman [43] showed how ESP and RESP charges performed quite well at modeling the conformational energies of a series of 1,3-dioxanes. In addition, a more thorough study of the performance of RESP charges at calculating small molecule conformational energies is currently underway in the group.

It should be noted that Mulliken charges do NOT reproduce the electrostatic potential of a molecule very well. Mulliken charges are calculated by determining the electron population of each atom as defined by the basis functions. When the density is associated with the square of a single basis function, that density is assigned to the atom associated with that basis function. Similarly, if the density is associated with 2 basis functions which are on a common atom, the density is assigned to that atom. The ambiguity arises when the density is associated with 2 basis functions lying on different atoms. In that case the density is partitioned equally onto each atom.

Another charge model is that of Gasteiger-Marsili. [44] This approach involves the partial equalization of electronegativity between bonded atoms.

A description of RESP. The basic idea behind the new charge fitting algorithm is that the charges on non-hydrogen atoms are restrained to an ''optimal'' value of zero. This model evolved from work carried out by Christopher Bayly, which showed that charges on buried atoms (such as alkyl carbons) were not well determined by the electrostatic potential points. Such buried charges often assumed large values during the fitting process and the values of these charges showed great conformational variability. The restraints are hyperbolic in nature, so approximately the same restraining force is felt by charges of all magnitudes. This reduces the magnitude of charges which can be reduced (typically the ones buried within the residue) without affecting the fit to the potential by much. For example, Christopher Bayly shows in the JPC paper that the charges on the methyl atoms in methanol can be significantly reduced without impacting the fit, while the charges on the hydroxyl O and H have well defined values. An earlier model employed harmonic restraints, but they reduced the values of the heteroatom charges too drastically since those values (typically +/- 0.6 or higher) fell in the steep part of the function. The details of the derivation of this method are given in the Bayly *et al*. JPC paper.

The RESP charges show less conformational variability than the standard ESP charges. They result in very good conformational energies for the small molecules studied to date using only a very simple torsional potential. The RESP charges also reproduce the important interaction energies and free energies of solvation. Furthermore, fitting can be carried out using multiple conformations and/or multiple molecules.

---

[42] ''Calculation of Molecular Geometries, Conformational Energies, Dipole Moments, and Molecular Electrostatic Potential Fitted Charges of Small Organic Molecules of Biochemical Interest Using Density Functional Theory,'' St-Amant, A., Cornell, W.D., Halgren, T.A., and Kollman, P.A., *J. Comp. Chem*., in press.

[43] ''A Molecular Mechanical Model that Reproduces the Relative Energies for Chair and Twist-Boat Conformations of 1,3-Dioxanes,'' Howard, A.E., Cieplak, P., and Kollman, P.A., *J. Comp. Chem*., **1995**, 16:2, 243-261.

[44] Gasteiger and Marsili, *Tet. Lett*. **1980**, *36,* 3219.

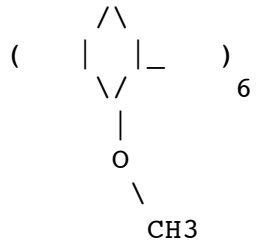
Charges for the old AMBER (Weiner *et al*.) force field were derived using the STO-3G basis set. The 6-31G* basis set was used for the new charges because it exaggerates the dipole moment of most residues by 10-20%. It thus "builds in" the amount of polarization which would be expected in aqueous solution. This is necessary when using an effective two-body force field which does not include explicit polarization. The 6-31G* charges are then critical for the side of the free energy cycle where one solute is being mutated into another in solution. Charges derived using a higher level of theory (either in terms of a bigger basis set or through the inclusion of correlation) won't necessarily be better for such applications if they do not result in dipole moments which are enhanced over the gas phase values.

The RESP method involves a two-stage approach where charges on atoms such as methyl hydrogens are not forced to be equivalent until the second stage. At that point they are refit while charges on the other atoms are constrained to their values from stage one. Forcing methyl hydrogens to have equivalent charges during the first stage can adversely affect the values of the heteroatom charges, because such hydrogens are not equivalent in a static conformation. In the standard ESP model, methyl hydrogen charges were typically averaged after the fit, but this averaging often changed the value of the dipole moment as well as the fit to the potential.

One problem with electrostatic potential fit charges in general is that they reproduce the molecular potential and the dipole moment very well *for the conformation of the molecule employed in the fit*. However, when those charges are applied to other conformations, the agreement is not as good. Chris Reynolds proposed using multiple conformations of a molecule in the charge fitting process. [45] In fitting the amino acid charges for our new force field, we used 2 conformations for each amino acid – the first conformation had the backbone in an extended conformation and the second had it in an alpha-helical conformation. Each of those 2 conformations then had different values for chi1, chi2, etc.

We have found that multiple conformation RESP fitting results in fairly robust charges. The RESP program also allows for specifying additional lagrange constraints so that (e.g.) blocking groups can be forced to have net neutral charges and molecules can be spliced together more algorithmically.

Pitfalls of ESP-fit charges. In general, standard esp-fit charges perform well at reproducing desired properties such as DNA base pair, NMA dimer, and methanol-water interaction energies. This is not always the case, however. For example, Yax Sun in our group calculated ESP charges for a spherand for the purpose of carrying out free energy perturbation studies to compare the binding of Li+ and Na+ to the spherand. The spherand looked like this:



Standard ESP charges were calculated from a 3-unit, non-cyclic, methyl-blocked analog. The charges were then taken from the central residue. The standard ESP fit charges underestimated the interaction energies between the spherand and the ions. As the charge on the oxygen was on the low side, this

[45] "Atomic Charges for Variable Molecular Conformations," Reynolds, C. A.; Essex, J. W.; Richards, W. G. *J. Am. Chem. Soc*., 1992, 114:23, 9075-9079.

was thought to be the source of the error. Sun and Kollman then refit the charges, this time with electrostatic potential points within 5 A of the oxygen weighted more heavily than those around the rest of the molecule. The resulting charges resulted in a relative free energy of binding in much better agreement with experiment.

There also cases where polarization or lone pairs are required to reproduce interaction energies.

Electrostatic potential fit charges also do not always result in good conformational energies. For example, multiple conformation (C5/aR) RESP charges calculated for glycine and alanine dipeptides do not result in conformational energies which are in good agreement with the quantum mechanically calculated values. These charges were derived using the 6-31G* basis set and were applied with a 1-4 electrostatic scale factor of 1/1.2 (Cornell *et al*. JACS). The reason for this performance is unclear. It is possible that the 6-31G* charges overstabilize the C7 (7-membered H-bonded ring) conformations. When these charges are scaled back to gas-phase-like values (q*0.88), the conformational energies show good agreement with the QM data.

I've given 2 examples where esp-fit charges were not able to be applied in a straighforward fashion. Overall, however, we have found these charge models (ESP and RESP) to be quite useful for modeling biomolecular systems. The alternatives usually involve either (1) models such as Mulliken charges which do not necessarily reproduce the molecular electrostatic potential or (2) empirically derived charges such as those fit to reproduce interaction energies and distances (CHARMM) or liquid properties (OPLS). Because electrostatic potential fit charges can be calculated fairly easily, they allow the force field to be extended to other molecules. Overall we find them to be a very useful and relatively general model.

Further Notes on Charge Derivation for the 1994 Force Field

I. Introduction

The 4.1 release of AMBER includes a new parameter file (parm94.dat) and new residue files (all_nuc94.in, all_amino94.in, all_aminont94.in, and all_aminoct94.in) which contain the parameters developed for our second generation force field. The new force field employs an approach to atomic charge fitting which is a modification of the original "standard ESP" (ElectroStatic Potential fit) approach. The new approach still involves a least squares fit of the atom centered charges so that the classical potential reproduces the quantum mechanical potential. With the new approach, however, hyperbolic restraints are applied during the fit to the charges in order to attenuate the charges on buried atoms which are statistically ill-determined. The new approach is called "RESP," for "Restrained ElectroStatic Potential fitting."

II. Two-Stage Restrained ESP (RESP) Fitting

The new charge model includes another major modification – a two-stage fit. This approach allows for the fitting of equivalent charges on certain atoms which are equivalent when the molecule is freely rotating, but are not equivalent within a static conformation. These equivalent charges are fit during the second stage of the fit, in the presence of the charges which were determined for the other atoms during the first stage.

With the previously used standard ESP method, such equivalent atoms were typically averaged "a posteriori," or after the fit. This approach often resulted in a significant change in the molecular dipole moment.

Methyl hydrogens are the atoms most typically included in the second stage fit. The methyl carbon is included as well in order to provide a sufficient number of degrees of freedom (in this case "1" since if q(C)= x then q(H)= -x/3).

The need for refitting methyl hydrogens during a second stage fit was made obvious by the example of methanol. When the three methyl hydrogens were constrained to have the same charge during a one stage fit, the charge on the oxygen was significantly reduced over its value in an unconstrained fit.

The restraint applied during the second stage of the fitting is twice as strong as that applied during the first stage (0.0010 vs. 0.0005). The motivation behind this choice was that nonpolar groups were being refit in the second stage and those atoms should have small charges in order to decrease their conformational dependence. The more the charges vary with conformation, the more the resulting conformational energies are subject to variation. Such behavior is not only undesirable from the standpoint of general reproducibility of results, but also from the standpoint that any dihedral parameters which are optimized are coupled to the charge set used.

Because the stronger restraint was applied in the second stage, the decision was also made to refit methylene groups during that stage. Second stage refitting has not been applied to polar atoms, such as the two oxygens in 1,2-ethane diol which are inequivalent when that molecule is in a conformation other than the one with highest symmetry (tTt). Similarly, second stage refitting has not been applied to amino hydrogens. Tests showed that constraining them to have the same charge during the first stage did not change their value much when compared to an unconstrained fit.

It should be emphasized that the two-stage model as described above is based on refitting only nonpolar groups during the second stage. In cases where a second stage refitting of polar atoms is thought to be necessary, then the second stage should also employ the weaker restraint and a third stage refitting employed with the stronger restraint for nonpolar groups. The best way to determine the ideal values of nonpolar charges is to carry out a fit with no restraint or with the weak restraint and with *no* constraints of equivalent atoms.

This new approach is clearly more complicated than the previous charge model and involves more subjective decisions. We have therefore provided a number of demos which we hope will serve as useful guides.

III. Multiple Conformation Fitting

Ideally, the new charge model also includes the use of multiple conformations of a given molecule in the charge fit. The amino acid charges which were derived for the new force field were fit to two conformations of each amino acid. Specifically, the extended (C5/beta-sheet) and alpha-helical conformations were applied to the backbones. The side chain torsions were then assigned so that a given torsion had a different orientation in the extended and in the alpha-helical conformation. This strategy was based on earlier results obtained with propylamine, which showed that a two-conformation fit was nearly as good as a five-conformation fit, as long as all of the primary dihedrals were varied between the two conformations. Primary dihedrals can be defined as ones where both terminal atoms are either heavy atoms or polar hydrogens.

The combined use of two-stage fitting and multiple conformation fitting requires even more decisions in terms of which atoms should be constrained to be equivalent at what times. For the example of propylamine, the best approach was found to be to constrain all corresponding heavy atoms and the amino hydrogens to be equivalent between the different conformations in the first stage. The corresponding methyl and methylene atoms were constrained to be equivalent between conformations (for the carbons and hydrogens) and within conformations (only for the hydrogens) during the second stage of the fit.

The first stage of the fit then resulted in three different charges for the carbons (C-alpha, C-beta, and C-gamma) and 7 times the number of different conformations charges for the hydrogens. That is, every hydrogen was allowed to have a different charge. One common mistake would be simply to

make each atom be equivalent between all conformations, e.g. the first methyl hydrogen across all conformations and the second methyl hydrogen across all conformations, etc. However, because the conformations are different, there is no reason that individual hydrogens in a methyl group should correspond to individual hydrogens in that same methyl group in a different conformation of the molecule, based on the numbering.

IV. Conclusion

In summary, the new Two-Stage Multiple conformation RESP charge model has been shown to perform quite well in the calculation of interaction energies, free energies of solvation, and conformational energies. The new method involves more subjective decisions than the previously used standard ESP method, however. When in doubt, it is useful to carry out the charge fit in two or more different ways, in order to compare the effects of different constraints.

AMBER 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, NEXT, 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 constraint 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 with atoms completely in perturbed group
MGPER : number of angles with atoms completely in perturbed group
MDPER : number of dihedrals with atoms completely in perturbed groups
IFBOX : set to 1 if standard periodic box, 2 when truncated octahedral
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. (Divide by 18.2223 to convert to kcals/mol)

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) (ICO(i), i=1,NTYPES*NTYPES)

ICO : provides the index to the nonbon parameter arrays CN1, CN2 and ASOL, BSOL. All possible 6-12 or 10-12 atoms type interactions are represented. NOTE: A particular atom type can have either a 10-12 or a 6-12 interaction, but not both. The index is calculated as follows:
index = ICO(NTYPES*IAC(i)-1+IAC(j)
If index is positive, this is an index into the 6-12 parameter arrays (CN1 and CN2) otherwise it is an index into the 10-12 parameter arrays (ASOL and BSOL).

FORMAT(20A4) (LABRES(i), i=1,NRES)

LABRES : the residue labels

FORMAT(12I6) (IPRES(i), i=1,NRES)

IPRES : atoms in each 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 : 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 into this array is as follows (assuming in index is positive): 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.

NOTE: the atom numbers in the arrays which follow that describe bonds, angles, and dihedrals are obfuscated by the following formula (for runtime speed in indexing arrays). The true atom number equals the absolute value of the number divided by three, plus one. In the case of the dihedrals, if the third atom is negative, this implies an improper torsion and if the fourth atom is negative, this implies that end group interactions are to be ignored. End group interactions are ignored, for example, in dihedrals of various ring systems (to prevent double counting) and in multiterm dihedrals.

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(i)-LPH(i). Note, if the
periodicity is negative, this implies the following entry
in the PK, PN, and PHASE arrays is another term in a
multitermed dihedral.

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 .gt. 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 correctly 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 : last atom before the start of the cap of waters
placed 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 are only present if IFPERT .gt. 0 ****

Note that the initial state, or equivalently the prep/link/edit state,
is represented by lambda=1 and the perturbed state, or final
state specified in parm, is the lambda=0 state.

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 REQ for the
perturbed bonds. ICBPER(i) represents lambda=1 and
ICBPER(i+NBPER) represents lambda=0.

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 pertrubed 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=1 and ICPPER(i+NGPER) represents lambda=0.

FORMAT(20A4) (LABRES(i), i=1,NRES)

LABRES : residue names at lambda=0

FORMAT(20A4) (IGRPER(i), i=1,NATOM)

IGRPER : atomic names at lambda=0

FORMAT(20A4) (ISMPER(i), i=1,NATOM)

ISMPER : atomic symbols at lambda=0

FORMAT(5E16.8) (ALMPER(i), i=1,NATOM)

ALMPER : unused currently in gibbs

FORMAT(12I6) (IAPER(i), i=1,NATOM)

IAPER : IAPER(i) = 1 if the atom is being perturbed

FORMAT(12I6) (IACPER(i), i=1,NATOM)

IACPER : index for the atom types involved in Lennard Jones
interactions at lambda=0. Similar to IAC above.
See ICO above.

FORMAT(5E16.8) (CGPER(i), i=1,NATOM)

CGPER : atomic charges at lambda=0

**** 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) (ATPOL1(i), i=1,NATOM)

ATPOL1 : atomic polarizabilities at lambda = 1 (above is at lambda = 0)

AMBER RESTART FILE SPECIFICATION

FORMAT(20A4) ITITL

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

FORMAT(I5,5E15.7) 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 periodic box [4.0 and previous: only if constant pressure]:

FORMAT(6F12.7) BOX(1), BOX(2), BOX(3)

BOX : size of the periodic box

Note: in AMBER 4.1 if the ewald option is turned on, the box angles will also be written out in the same format.

GIBBS will print extra information.

AMBER AMBER TRAJECTORY (COORDINATES OR VELOCITY) FILE SPECIFICATION

FORMAT(20A4) ITITL

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

The following is 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 periodic box [4.0 and previous: only if constant pressure]:

FORMAT(10F8.3) BOX(1), BOX(2), BOX(3)

BOX : size of periodic box

Amber 4.1 Release Notes

(Differences Between Amber 4.1 and Amber 4.0)

The information in this section is mainly of interest to those who have used previous versions of Amber. The current section describes changes between Amber 4.0 and Amber 4.1, and the following sections contain similar descriptions of changes introduced with earlier releases.

World Wide Web Page:

A series of web pages has been created for Amber: this contains bugfixes, discussions of issues relevant to molecular mechanics, documentation, information on obtaining Amber, and more. See

http://www.amber.ucsf.edu/amber/amber.html

Unix Release:

The user's installation procedure has been simplified slightly by the elimination of the src/make-make script; when building in the src/ tree, Makefiles now figure out the system-specific directory to use rather than needing makemake to edit them. See src/0README for more detailed notes of changes in code and compilation.

The tests can be run using 'make', thanks to a Makefile in the test/ directory.

Documentation:

The manual has been reorganized somewhat, and the readability of the ascii .doc files has been improved with the help of GNU utilities. A new amber41/Questions subtree has been added, which is a somewhat informal compendium of answers to users' questions and (in amber41/Questions/0Net/) general discussions of modeling issues from a variety of sources. These are also to be found under the Web page.

Manuals have been added for the new programs SPASMS, LEaP and Interface, and are included in two new bound volumes. The LEaP and Interface manuals are in amber41/leap/doc/ and amber41/interface/doc/. The SPASMS manual is in .doc form in amber41/doc/ and in Rich Text Format (RTF) form (which should be accessible to newer word processors such as Word) in amber41/doc/spasms/. Postscript files will be included if a way of generating them can be found (MS Word generates huge ones that don't always print).

Benchmarks:

A special bench/ directory has been added with scripts for running benchmarks and collating the results.

Database:

The C-shell files for building the residue database under Unix have been replaced by a Makefile.

There is a new force field which is particularly adapted for solvated systems, primarily involving a new charge-calculation scheme in conjunction with the 6-31G* quantum mechanical basis set. This is described in detail in the **Database** section of the manual. Operational differences include the use of a factor of 1.2 for scaling electrostatic interactions (SCEE; 2.0 is used with the old force field) and new names for the nucleic acid residues, eliminating separate HB, HE and POM (terminal hydrogen and phosphate) residues. The new residues are described in the LINK section of the manual. The files for the new database are `db94.dat` (for link) and `parm94.dat` (for parm).

Tutorials:

A fairly extensive tutorial on plastocyanin has been added.

LEaP:

A new package called LEaP has been added, written by Christian Schafmeister and fixed and extended by Bill Ross and Vladimir Romanovski. It provides an alternative to the combination of prep, link, edit and parm; *xleap* includes a graphical molecular editor that runs on any X-windows platform, and *tleap* provides a simple command-line interface to most of the functionality. LEaP has its own special installation procedure and manuals, and is found in amber41/leap/.

Interface:

A new program called Interface has been added, written by David Pearlman. It provides a front-end scripting language to simplify decisions on sander and gibbs input and to replace shell scripts for running the programs. It has an independent tree under amber41/interface/ with its own special installation procedure and manuals.

Resp:

This is the new program for fitting atomic charges to reproduce an electrostatic potential. It is derived from a program ESPFIT originally written by U. Chandra Singh and Peter Kollman. The current version adds restraint methods for holding the point charges relatively neutral and fitting across multiple conformations, and was written by Christopher Bayly (now at Merck Frosst Canada).

Link:

The ICONN option for linking molecules has been disabled since it was discovered that torsions were not placed on the linking bond. The crosslink mechanism can be used for this function.

Edit:

When adding a box trimmed to a certain distance from a molecule, 0.4 Å clearance is now added at the box boundary to prevent close water-water contacts, since Edit does not check for them. This decreases the possibility of shake failure at minimization or dynamics startup.

User-defined solvents may now be added. The OCT option has been added to generate water boxes with truncated octahedral boundaries, courtesy of Thomas Huber of Ludwig Maximilian Universitaet, Muenchen. If periodic boundaries are used in sander, *etc*., the `oct box' is automatically

detected and used. Input checking has been added so bad input in edit.in is no longer skipped when reading options.

Parm:

Input checking has been added so that keyword fields in parm.in must be all blank if no keyword is used.

Minmd:

This program has been eliminated and its polarization force field option moved to Sander.

Sander:

Some defaults have changed: IDIEL = 1 (no longer distance dependent dielectric). CUT is now 8.0 Å instead of 9.0 Å. SCEE no longer has a default value since the new force field requires 1.2 while the old one used 2.0.

The mden file is now in table format, with all the numbers for each step in a single row and some number of spaces between each number. The first line no longer has the title of the run; instead it is a list of names for the columns.

An optional secondary cutoff has been added. A fast Ewald sum method has been contributed. Fast routines for intra- and inter-water potentials have been introduced, for a speedup of *e.g.* 40% on 2466 waters and 284 solute atoms in a constant volume box with 8 Å cutoff. Sander has been parallelized for shared memory and message-passing architectures (see src/0README.parallel).

The box coordinates are now output in the mdcrd and restrt coordinate files when using constant volume, as has always been done for constant pressure. Handling of boxes with truncated octahedral boundaries has been added. If periodic boundaries are used, the box type from EDIT is automatically detected and used. Polarization has been added. The input description in the sander manual has been rearranged from the original arbitrary, somewhat unordered grouping of variables in IBM `card' fashion to a more logical grouping by function.

Spasms:

This is a completely new molecular mechanics/dynamics program contributed by David C. Spellmeyer (Chiron Corporation), William C. Swope (IBM Research), Erik-Robert Evensen (Dept. Chemistry, Harvard University), and David M. Ferguson (University of Minnesota), with major contributions from Shuichi Miyamoto, Thomas Cheatham and Randall Radmer (code development) and Allison Howard (documentation). Principal investigators from UCSF are Peter A. Kollman and Robert Langridge.

The spasms directory is organized differently from the other Amber source directories, in that the `real' source is kept in files with the .msk extension, which are designed to be preprocessed by the spasms/util/unmask program into machine-specific .f files, which can then be compiled. This enables source for different machines to be easily generated from a non-Unix platform. (The other Amber programs use the more flexible but Unix-only cpp program for generating source for arbitrary platforms.) See src/spasms/0README for further details.

The spasms manual is provided in electronic form in RTF format, which is suitable for viewing and printing with editors such as Word.

Gibbs:

Some defaults have changed: IDIEL = 1 (no longer distance dependent dielectric). SCEE no longer has a default since the new force field requires 1.2 (the old one uses 2.0).

A full implementation of Thermodynamic Integration is provided, a polarization potential is included, and free energy derivatives and components may be calculated. An optional secondary cut-off has been added, as well as fast routines for intra- and inter-water potentials. Gibbs has been parallelized for shared memory and message-passing architectures (see src/0README.parallel).

The mden file is now in table format, with all the numbers for each step in a single row and some number of spaces between each number. The first line no longer has the title of the run; instead it is a list of names for the columns.

Carnal:

Carnal is a new coordinate/trajectory analysis program, written by Bill Ross. It will eventually replace mdanal (and possibly anal), so we recommend using it instead of mdanal whenever possible. Bugs in mdanal will not be fixed, while bugs in carnal will. Carnal is the first Amber program to be written in C, and as such has the distinction of not being statically dimensioned, so recompilation for larger problems is not required.

Since in 4.1 periodic box coordinates are written to mdcrd files during constant volume simulations, the STREAM BOX option has been deleted and the assumption is made that a box is present in the mdcrd files if one is indicated in the prmtop file; if using an old 4.0 constant volume trajectory, use NOBOX to force carnal to use the box from prmtop. (When the box is read from an mdcrd file, the line is checked to verify that only 3 numbers are on it.)

Rdis:

Rdis calculates distributions of measurements, including density-normalized distributions around a point.

Curvop:

Curvop performs pointwise subtraction, averaging, addition, multiplication and division of pairs of curves.

Curvemax:

Curvemax prints the x,y values at the cumulative maximum of a table of x,y values.

Protonate:

Protonate adds protons to protein or DNA heavy atoms, converts proton names between various conventions, and checks (pro)-chirality.

Gwh:

Gwh (Generate Water Hydrogens) sets positions of water hydrogens onto water oxygen positions that may be present in PDB files by optimizing simple electrostatic interactions.

Pol_H:

Pol_H resets positions of polar hydrogens of protein residues (Lys, Ser, Tyr and Thr) by optimizing simple electrostatic interactions.

Ambpdb:

Ambpdb converts restrt files to various formats: PDB, Don Bashford's MEAD program, Mike Connolly's surface area/volume programs, and Mike Pique's FLEX program.

Rdparm:

Rdparm displays and modifies the contents of parm topology files.

Pdbconvert:

Pdbconvert changes PDB residue naming conventions for nucleic acids from the old force field format to the new one.

Other programs:

Pdbgen has been modified to allow command-line arguments. Geom has corrected dihedral measurements.

Benchmarks:

The DNA cpu performance benchmarks have been moved from demo/dna/ to amber41/bench/ and a plastocyanin benchmark has been added.

Amber 4.0 Release Notes

(Specific Differences Between Amber 4.0 and Amber 3.0 Rev A)

Unix Release:

More extensive use is made of the Unix environment, including the use of *make*(1) and shell scripts. The .com extension for scripts has been replaced by .csh for C-shell scripts and .com is now used only for VMS scripts. 0README files have been put in various directories explaining the contents. More extensive facilities are provided for generating source code for different target machines, with single- and multiple-file options. All the programs except those in the etc directory now return exit status of 1 in case of error and 0 otherwise. The demo and test scripts take advantage of this feature, which should aid in writing batch job scripts and in integrating Amber into larger systems. The demo directories have been reorganized, and demo file names have been changed to simplify porting them to a flat file system. The scripts for running the tests now print fewer extraneous non-numeric diffs.

Database:

The all-atom protein data base now uses IUPAC-IUB names for hydrogens (and, indeed, for all atoms.) One exception is for the 2' hydroxyl proton on ribose: the IUPAC-IUB document says that this should be named "O2'H", making it unique in being a proton whose name does not begin with "H". Since many parts of AMBER identify protons on the basis of the first letter of the name, we have deviated from the IUPAC-IUB standard, and called this proton "HO2'". All we can say is that we hope this doesn't cause too much trouble.

Different people interpret the IUPAC-IUB rules for proton names in amino acids in different ways; in our convention, the beta-methylene protons on most residues are labeled "HB2" and "HB3", with equivalent rules for other methylenes. Edit the *all.in, allnt.in* and *allct.in* files in the */dat* directory if you want to change this convention. Note in particular that Brookhaven labels beta-methylene protons (for example) as "1HB" and "2HB", with the comment that these names are to be interpreted as "HB1" and "HB2". Names in pdb files like "1HB" will be converted by Amber into "HB1" for internal use, but the "1" and "2" will *not* be translated to "2" and "3". [In particular, Amber 4.0 reads in pdb files with the following conventions: spaces in Col. 13 are stripped away, in effect left-justifying atom names; if a "1", "2" or "3" appears in column 13, it is rotated around to the end of the name. If this process does not do what you want it to do, you could modify the "pdbrd.f" file in *../src/edit*, or manually edit your pdb file.]

When outputting pdb files, Amber4 re-wraps the proton atom names in a way that is consistent with Brookhaven usage. Again, however, if the default Amber proton names (like "HB2" and HB3") are used, they will be converted to "2HB" and "3HB", but the "2" and "3" will *not* be translated into "1" and "2".

In addition to the standard charges, an "alternate" data base is supplied in which the charges on lys, arg, asp and glu are reduced to +/- 0.2; such reductions are often used in carrying out nmr refinements in vacuum, or for other vacuum calculations where a simple way of screening charged side chains may be desirable.

The *parm89a.dat* file the Amber3a version has been changed slightly by making the force constant for the LP-S-LP angle be 150. rather than zero. Problems were occasionally found with lone-pair geometries using the earlier value. The new force field is called *parm91.dat*. The older version is also

on the distribution tape for backwards compatibility.

For alkali ions with explicit waters, we have provided the latest values of Åqvist [46] which are adjusted to reproduce the ion-OW potential using the Amber combining rules in order to reproduce the first peaks of the radial distribution and the relative free energies of solvation in water of the various ions. See dat/0README for more details.

Source code:

All common blocks have been aligned on double word boundaries to avoid compiler warnings and to accommodate compilers that do not provide padding.

Link:

The specification of the database has been moved from inside the link.in file to a '−p' argument (Unix) or unit 1 file assignment (VMS). The old database specification in line 2 of link.in is read but ignored.

Edit:

The *pdbrd* subroutine has been modified to read in the funny Brookhaven proton names as described above.

Parm:

In a major change, *parm* can now read a second force field file, *frcmod*, which is used to specify substitute or additional parameters for a particular project. The standard Amber parameters are intended to reside in the old *frcfld* file, which can be treated as a shared, "read-only" standard database. This is backward compatible since the feature need not be used.

The *parm* program can also now read in polarizability data. There is a possibility of non-backwards compatibility here: sometimes comments have been placed in the section of old parm files where the new polarizabilities are now input; these should be very easy to fix. Note that no polarizability parameters are supplied, since this feature is still in development.

Min/md:

The functionality of the old *min* and *md* programs has now been merged into a single program, minmd. Force fields that make use of dipole polarizabilities may now be used. For more details, see the appropriate input section, below.

A long-standing scaling bug in the use of the TAUTP and TAUP parameters in the *md* module has been fixed, so that the new versions work as described by Berendsen *et al*.. NOTE: to compare 4.0 MD runs to amber3A runs you must multiply TAUTP, and TAUP by 20.455 in the amber3A run. In addition, new temperature scaling options are available which allow the solvent and solute to be separately coupled to the temperature bath (i.e. the TAUTS switch is now active).

[46] J. Åqvist, *J. Phys. Chem.* **1990,** *94*, 8021-8024.

Gibbs:

Major changes include: the incorporation of dynamically modified windows; [47] the ability to carry out Potential of Mean Force (PMF) simulations using holonomic constraints; the ability to include the "PMF bond correction" [48] when calculating free energies; the ability to carry out thermodynamic integration (TI) calculations; improved initial state/final state mixing rules; the inclusion of several new temperature coupling options; perturbations where non-bonded interactions are represented as a hydrogen bond (10-12) in the initial and/or final state are now handled correctly; double-wide sampling can be turned off; a new succinct free energy vs. window summary file can be written; periodic imaging can be performed on a residue basis; intra-perturbed group contributions can optionally be calculated; if SHAKE fails, a restart file and an attempted continuation can be requested; some of the formatted output has been clarified and reformatted; several bug fixes (including the TAUTP/TAUTS/TAUP bug mentioned above) have been incorporated; code clean-up and documentation.

This program is a major change from the previous versions of Gibbs. Gibbs version 4.0 is a descendent of Gibbs version 3.0 (not version 3A), so some of the comments in the 3A release notes (in particular regarding common libraries, 16 bit integers, and some vectorization) do not apply to version 4.0.

Sander:

This is a completely new module for carrying out minimization and dynamics, which makes available many interesting features for carrying out refinements of solution structures of macromolecules based on magnetic resonance data, and for general conformational searching. In addition to allowing a flexible description of distance, bond angle, and torsional restraints, the module gives the user control over the weights associated with these penalty functions, and over the van der Waals portion of the underlying force field. The user can straightforwardly "program" a simulation to change the nature of the restraints, the target temperature, etc., as many times as is desired, all within a single run. In addition, calculations based on relaxation matrix estimates of NOESY intensities, [49] and chemical shifts, [50] are available. And time-averaged restraints, [51] can be easily applied.

Nmode:

This is a thorough revision of the previous *nmode* module. Major additions include the ability to compute "Langevin modes" (normal modes in the presence of viscous damping from a continuum solvent), [52] and to find transition states as well as minima. [53] The associated modules *nmanal* and *lmanal* (new in this release) are analysis programs for normal modes and Langevin modes, respectively. They calculate atomic fluctuations in Cartesian and internal coordinates, plus a variety of time-correlation

---

[47] D.A. Pearlman and P.A. Kollman, *J. Chem. Phys.* **1989,** *90,* 2460.

[48] D.A. Pearlman and P.A. Kollman, *J. Chem. Phys.* **1991,** *94,* 4532.

[49] P. Yip and D.A. Case, *J. Magn. Res.* **1989,** *83,* 643.

[50] G.P. Gippert, P.F. Yip, P.E. Wright and D.A. Case, *Biochem. Pharm.* **1990,** *40,* 15; K.J. Cross and P.E. Wright, *J. Magn. Res.* **1985,** *64,* 220.

[51] A.E. Torda, R.M. Scheek and W.F. van Gunsteren, *J. Mol. Biol.* **1990,** *214,* 223; D.A. Pearlman and P.A. Kollman, *J. Mol. Biol.* **1991,***220,*457.

[52] G. Lamm and A. Szabo, *J. Chem. Phys.* **1986,** *85,* 7334; J. Kottalam and D.A. Case, *Biopolymers* **1990,** *29,* 1409.

[53] D.T. Nguyen and D.A. Case, *J. Phys. Chem.* **1985,** *89,* 4020.

functions of relevance to magnetic resonance relaxation and fluorescence anisotropy decay.

Amber 3.0 Rev A Release Notes

(Differences Between Amber 3.0 and Rev A)
by George Seibel

Database:

The entire database has been examined. The error-prone CHARGE keyword method of assigning charges was abandoned in favor of a more reliable method. Atom names of C-terminal charged amino acids have been changed to the PDB standard 'O' and 'OXT' in both the united and all atom files. Previously these were 'O1' and 'O2' in uni.in, and 'OA' and 'OB' in all.in. Inconsistent SG-LP bond lengths were fixed in all files. Other than this, geometries are unchanged, although some residues are far from their minimum energy conformations.

A number of errors in the 3.0 database have been fixed:

all.in: RPOM charges, RURA connectivity.
allct.in: GLN, HIP, and GLU charges, planarity constraints on carbonyls, all residues.
allnt.in: no errors.
uni.in: no errors.
unict.in: Carboxy oxygen atom types changed to O2, all residues
unint.in: PRO charges.
opls_uni.in: no errors.
opls_unict.in: Carbonyl planarity constraints, all residues. Charges and Carboxy oxygen atom types: ARG, HID, HIE, TRP.
opls_unint.in: no errors.

Force Field Parameter Files

All parameter files distributed with Amber 3.0 are provided unchanged. In addition, a new parameter file, parm89a.dat, is provided. This is a modification of the 3.0 parmallhb.dat file, with the following changes: TIP3P atom type HW has been corrected to have 6-12 coefficients of zero. Masses of hydrogens have been changed from 3.0 to 1.008, and the mass of the LP (sulfur lone pair) has been reduced from 12.0 to 3.0. Angle bending force constants for some angles involving the LP (lone pair) atom type have been modified. The previous force constants were an order of magnitude larger than normal bending force constants, and resulted in erratic (sometimes catastrophic) behavior in minimization and md. The new force constants have been softened, and behave much better. Energetic and structural results related to the LP force constants are not significantly changed. All parameter changes are documented in the file readme.parm in the database directory.

Prep:

Installed portable file handling scheme with Unix command line interface, configurable for JCL file assignment. Rewrote subroutine OPENDA for portability. Added partial bounds violation protection. Fixed bug that resulted in garbage and/or wrong dihedrals on unit 6 output. Changed internal variables in TORS and BANGLE to double precision for consistent output on different machines. Removed bogus args from call to BONDFM. Fixed bound violation problem in BONDFM. Removed illegal status keyword in CLOSDA. Fixed incorrect args to COORD in GETMOL. Removed dead

code, cleaned up some questionable syntax. Fixed array arg dimensions so bounds checker could be used, ran tests with compiler bounds checking enabled.

Link:

Parameterization of all arrays. Added array bounds protection on all arrays. These changes allow Link to be easily and safely redimensioned to run on small memory machines. File handling largely replaced by a more portable scheme with Unix command-line interface, configurable for JCL file assignment. Rewrote OPENDA and APPEND in standard Fortran 77 for machine independence. Fixed illegal initializations of data in commons. Fixed uninitialized INDX in WRITEO and IOUT in FINDRS. Fixed error in common/IOFILE/, daf routines. Removed unused common blocks. Cleaned up output a bit. Removed cryptic messages for dangling improper dihedral removal that looked like error output, but were in fact normal behavior. Removed dead code, cleaned up some questionable syntax. Fixed array arg dimensions so bounds checker could be used, ran tests with compiler bounds checking enabled.

Edit:

Parameterization and bounds protection for all arrays. Replaced hardcoded limits with appropriate parameters. Removed machine dependencies and installed a portable file handling scheme with Unix command line interface, configurable for JCL file assignment. Removed broken pdb matching options, fixed iopt=1 XRAY option to work as it has been documented for years. (ie, read AND write a PDB file) Wrote new pdb reading routine to read real PDB files, as well as maintaining backwards compatibility with old "Amber" PDB files. Note that we now use a 3 character residue name, as per PDB standard. Fixed XRAY to stop if PDB sequence doesn't match LINK sequence, instead of generating garbage. Added user warnings if generic PDB residues matched to CYX/HIP/HIE/HID. Cleaned up and improved output. Removed undocumented hack that turned off H addition if extended precision XRAY output used. Replaced PGFIND and RSFIND with library routines FINDP and FINDRS. Removed unused 42000 word array in common/corsol. Fixed uninitialized variables in WRITEO, WSOL and BSOL. Added new routine to print the five longest bonds in the system at the end of each run. This will reveal the most common errors that are typically made in Edit, but not revealed until energy calculations are started. Removed dead code, cleaned up some questionable syntax. Changed array dimensioning conventions so bounds checker could be used, and ran tests with compiler bounds checking enabled.

Parm:

Parameterization and bounds protection for all arrays. Replaced hardcoded limits with appropriate parameters. Removed machine dependencies in BONSEP, ANGSEP, DIHSEP, and RFREE, and installed a portable file handling scheme with Unix command line interface, configurable for JCL file assignment. Major improvement in user output. Made all internal parameter manipulations double precision to help get the same answers on all machines. Fixed equivalencing scheme for H-valence terms to make safe redimensioning possible. Made RCONST more robust. Fixed divide-by-zero bugs encountered with dummy R* or AC parameters. Removed dead code, cleaned up some questionable syntax. Array dimensioning conventions have been fixed so bounds checker could be used, and tests have been run with compiler bounds checking enabled. Rev A PARM generates a "scalar" topology file, ie it does not duplicate dihedral pointers for dihedrals with multiple fourier terms. This topology file is appropriate for all Rev A programs.

Min:

Amber 3.0 contained two energy minimizers, BORN for Cray machines and MIN for the FPS264 and VAX/VMS machines. These programs differed slightly in input and in the handling of solvated systems, and BORN required a special "vector" topology file in which multi-term dihedrals had duplicated atom pointers. Rev A MIN is machine independent, thus there is now only one minimizer for all machines. The pairlist generator and all energy routines are vectorized, but also run very efficiently on scalar machines. There is now only one kind of topology file for all Rev A software. It is identical to the Ver. 3.0 "scalar" topology file. Multi-term dihedrals now have their pointers duplicated in MIN. In Rev A MIN all assumptions that integers and floating point words are the same size have been removed. This allows a full double precision version to be created on machines with native 32 bit wordsize. The single and double versions are now created automatically from the same code through the use of the cpp code preprocessor. A check for 16 bit overflow has been installed. This is important in very large systems when using 16 bit integers for the pairlist. The 16 bit pairlist is a configurable option on byte oriented machines, and is the default for Cray hardware. A check for pairlist overwrites has been installed to alleviate this common source of crashes. Memory use has been reduced. All single precision constants have been changed to double precision, improving consistency of results between different machines. The unreliable atom-based pairlist option has been removed. The pairlist generator now uses the oxygen atom of solvent waters in periodic systems. This behavior was formerly only found in BORN. An input switch is provided to revert to the old MIN behavior in which all atoms are used. A new highly vectorized solute-solvent routine which is much faster for large solvated systems has been added to the pairlist generator. Periodic systems are now imaged on a residue by residue basis, which prevents the serious imaging errors common to the earlier code. Installed portable file handling scheme with Unix command line interface, configurable for JCL file assignment. Rev A MIN is compatible with all data files from Ver. 3.0, with the exception of the old "vector" topology file. Binary data files are only supported in the single precision version. There is a check in MNREAD on the use of binary i/o. User-level output has been cleaned up, and some error messages improved. Redimensioning has been simplified by the use of parameter statements which only need to be set in main. Documentation on memory requirements is found in install.doc and in the Main module of Rev A MD.

Md:

Amber 3.0 contained two molecular dynamics programs, NEWTON for Cray machines and MD for the FPS264 and VAX/VMS machines. These programs differed slightly in input and in the handling of solvated systems, and NEWTON required a special "vector" topology file in which multi-term dihedrals had duplicated atom pointers. Rev A MD is machine independent, thus there is now only one program for all machines. The pairlist generator and all energy routines are vectorized, but also run very efficiently on scalar machines. There is now only one kind of topology file for all Rev A software. It is identical to the Ver. 3.0 "scalar" topology file. Multi-term dihedrals now have their pointers duplicated in MD.

In Rev A MD all assumptions that integers and floating point words are the same size have been removed. This allows a full double precision version to be created on machines with native 32 bit wordsize. The single and double versions are now created automatically from the same code through the use of the cpp code preprocessor. A check for 16 bit overflow has been installed. This is important in very large systems when using 16 bit integers for the pairlist. The 16 bit pairlist is a configurable option on byte oriented machines, and is the default for Cray hardware. A check for pairlist overwrites has been installed to alleviate this common source of crashes. Memory use has been reduced. All single precision constants have been changed to double precision, improving consistency of results

between different machines.

The unreliable atom-based pairlist option has been removed. The pairlist generator now uses the oxygen atom of solvent waters in periodic systems. This behavior was formerly only found in NEWTON. An input switch is provided to revert to the old MD behavior in which all atoms are used. A new highly vectorized solute-solvent routine which is much faster for large solvated systems has been added to the pairlist generator. Periodic systems are now imaged on a residue by residue basis, which prevents the serious imaging errors common to the earlier code. Support has been added for center-of-mass scaling in constant pressure systems.

Installed portable file handling scheme with Unix command line interface, configurable for JCL file assignment. Rev A MD is compatible with all data files from Ver. 3.0, with the exception of the old "vector" topology file. Binary data files are only supported in the single precision version. There is a check in MDREAD on the use of binary i/o. User-level output has been cleaned up, and some error messages improved. Redimensioning has been simplified by the use of parameter statements which only need to be set in main. See Code Configuration Section in MD main.

Rev A MD uses the exact same energy routines as MIN, thus assuring identical results and reducing the software maintenance burden. In addition to the portable file handling and removal of wordlength dependencies, a portable random number generator that gives bit-identical results on all machines has been implemented.

SHIAG now translates ALL molecules in periodic systems, so solute will not leave the box. Output intervals NTWX,V,E and their limit values NTWX(V,E)M now may be set to 0 to obtain sensible defaults. The new default behavior is no output, and dump files will not be opened. The default limit value is unlimited. I/O buffers are now flushed at each energy output, on systems where this is useful (ie, most Unix systems). First cycle energies are now always printed in order to check energy consistency between runs. End of File conditions in GETCOR are now handled cleanly, giving a diagnostic message instead of a crash.

Fixed long-standing bug in temperature evaluation for systems with non-zero NDFMIN. This bug resulted in temperatures that were too high, especially for small systems. The ISTAY option, which removed center of mass motion from solute only has been removed, as it was not correct to do this. Removal of center of mass motion is now correctly inhibited for belly systems regardless of the value of NTCM. Fixed bug that caused unwanted center of mass motion removal at start of multiple runs in certain cases. The cartesian restraint routine RESTX that used relaxation time TAUR has been removed. Its functionality is provided by XCONST, the harmonic restraint routine. The RESTX restraints were interacting incorrectly with the XCONST restraints in the 3.0 implementation. Fixed factor of ten error in total mass output from CENMAS. (this affected unit 6 output only, actual mass was correct) Removed constraint coords from restart file.

Gibbs:

All code changes, configuration options, and removal of wordlength dependence in Rev A MD also apply to GIBBS. See above or Rev A MD main for documentation. In addition to those changes, the following modifications apply to GIBBS only: All dynamics code shared with MD was put in library, so there would be only one version of each routine. In some cases this required small modifications, such as in PSCALE, where the sense of a flag argument needed to be reversed, and in a number of common blocks that had to be modified. Program now stops with error message if no restart free energy is read on a slow growth restart, instead of assuming it to be zero. NNBOND was rewritten for improved vectorization and residue based imaging. The dihedral, angle, and bond evaluation for non-perturbed atoms was vectorized, using the same code as MIN and MD. Consistency of results

on different machines has been improved. Fixed bug in GETALM; affected any machines that do not auto-save. [Note that some of the notes here regarding e.g. common libraries and vectorization of internal coordinate energy routines) do not apply to Gibbs version 4.0; see the 4.0 Gibbs release notes].

Anal:

Added overwrite protection, parameterized arrays. Portable file handling with Unix command-line interface, configurable for JCL file assignment. Removed other machine dependencies in GAUSS80, HANGL, and RFREE. Fixed bug in CALHB that overwrote dihedral pointers. Standardized PDB output. Removed all pairlist options except res-based/res pair memory efficient model. Cleaned up user output. Fixed uninitialized variables and questionable syntax. Converted local variables and constants in energy routines to double precision.

Mdanal:

This program did not receive the same level of attention as other Rev A software. Although I believe the changes made for Rev A were an improvement, I cannot guarantee the correctness or robustness of this program. The following changes were made: Installed portable file handling with Unix command-line interface, configurable for JCL file assignment. Parameterized the large scratch array, fixed broken or missing overwrite protection in some places. Set up code to allow compiler bounds checking. There are still unprotected arrays in this program, so use a bounds checker for debugging if you run into any. Fixed uninitialized MQ in RSUBI. Removed machine dependencies in HANGL, OPENB, NFORM, and WCONN. Fixed wrong unit number in PAKCM. Turned off movie coord generation in MDSHOW; no longer needed. Standardized PDB output. Made all common blocks consistent.

Nmode:

Installed portable file handling with Unix command-line interface, configurable for JCL file assignment. Parameterized the large scratch array, fixed broken or missing overwrite protection. Removed hardwired limit on max number of pairs, changed to use all allocated memory. Set up code to allow compiler bounds checking. Converted all single precision constants to double precision form. Added stop after minimizer error. Removed hbond cutoff in nonbon. Changed unit number of modes file from 50 to 11. Made all values of MACHEP consistent. Cleaned up output. Made all common blocks consistent.

New tools:

Two new features have been provided with Revision A running under Unix operating systems; an automated validation suite and tools for automatic redimensioning of the Fortran code. The validation suite is found in the amber41/test directory. There is a subdirectory for each program. Each subdirectory contains a number of command files that run the program using inputs from the demo directory, then diff the appropriate outputs. If the files have no differences, the test and diff outputs are removed. If differences are found, the outputs and diffs are left for evaluation. The validation tests should be run at installation, and after any change in the code or in system hardware, compilers, or libraries. Most of the programs have been modified to make safe redimensioning possible, through the use of Fortran parameters and any other necessary alteration of the code. In order to simplify

redimensioning, each source directory contains an executable csh script called 'resize.com'. These scripts use the stream editor sed to perform regular expression matching in order to change all instances of parameter values consistently, regardless of the value they were previously set to. These scripts will work correctly even if the same parameter was inadvertently set to different values in the same or different files. Each resize.com script contains documentation on appropriate ratios for parameter values.