CONGEN

CONformation GENerator program
Version 2.2.1, December 2009
Processed by TEX on 17 December 2009

Robert E. Bruccoleri
Malcolm E. Davis
Department of Macromolecular Modeling
Bristol-Myers Squibb Pharmaceutical Research Institute
P.O. Box 4000
Princeton, N.J. 08543-4000

This is the documentation for Congen, a conformational search program.

Congen is derived from CHARMM version 16, B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, David J. States, S. Swaminathan, and Martin Karplus. *J. Comput. Chem.* **4**, 187-217 (1983).

If you use CONGEN for research which is published, please cite the following papers:

1. R. E. Bruccoleri, "Application of Systematic Conformational Search to Protein Modeling", *Molecular Simulations* **10**, 151-174 (1993).
2. R. Bruccoleri and M. Karplus, "Prediction of the Folding of Short Polypeptide Segments by Uniform Conformational Sampling", *Biopolymers* **26**, 137-168 (1987).

If you use CONGEN's molecular mechanics features like minimization and dynamics, please cite the reference above for CHARMM as well.

Legalities

Short Contents

Table of Contents

Acknowledgements

There were many contributors in writing the CONGEN documentation. Bernie Brooks wrote the sections on Vibrational Analysis, Images, and Coordinate Manipulations. Dave States wrote the sections on the Non-bonded Interactions and Parameter File Development. S. Swaminathan wrote the section on Correlation Functions. I also thank Donna Bassolino for reviewing the entire document and for contributing the code for NMR constraints and all hydrogen topology file searching. Susan Cottingham worked on improvements to the coordinate I/O code, specifically on the Brookhaven Protein Data Bank section. A special thanks is extended to Drs. Juan Luis Pascual-Ahuir, Estanislao Silla and Iñaki Tuñon for making the GEPOL algorithms available for use within CONGEN.

Besides the direct contributors, CONGEN has had many supporters over the years. Dr. Edgar Haber provided the research environment where most of the work was done. His interest in macromolecular structure was essential. When support was difficult to obtain, Dr. Haber was always there. Dr. Jiri Novotny is my closest scientific colleague, and together, we have applied CONGEN to many different modeling problems. Many of the program's capabilities resulted from questions he wanted to answer. Jiri has bestowed me with his considerable knowledge about protein structure as well as his experience as a scientist. Professor Guy Montelione was a key collaborator in the development of the NMR constraint code, and Dr. Keith Constantine conceived of the ensemble averaging concept. Professor Martin Karplus was my Ph.D. advisor, and the laboratory that he established was the womb where CONGEN was created. I thank Professor Karplus, Mick Savage from Molecular Solutions Inc., Joyce Brinton and Bristol-Myers Squibb for permitting me to make CONGEN freely available and redistributable. Special gratitude goes to John Mesrobian, Annette Tobia, and Ed Penick for all their legal help over the years with CONGEN. The command processing language was inspired by the GRIPHOS package developed by Professor Jack Heller at the State University of New York at Stony Brook. Finally, Richard M. Stallman gave me the inspiration that CONGEN should be free.

This manual was formatted using the Texinfo macros running with TeX.

1 Introduction to Congen

CONGEN (**CON**formation **GEN**erator) is a program performing conformational searches on segments of proteins. It is most suited to problems where one needs to construct underdetermined loops or segments in a known structure, i.e. homology modeling. The program is a modification of CHARMM version 16, and has most of the capabilities of that version of CHARMM. Good references for CONGEN are R. E. Bruccoleri "Application of Systematic Conformational Search to Protein Modeling", *Molecular Simulations* **10**, 151-174 (1993); R. E. Bruccoleri, E. Haber, J. Novotny, "Structure of Antibody Hypervariable Loops Reproduced by a Conformational Search Algorithm", *Nature* **335**, 564-568 (1988); *Nature* **336**, 1266 (1988); R. Bruccoleri and M. Karplus. *Biopolymers* **26**, 137-168 (1987); and B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, David J. States, S. Swaminathan, and Martin Karplus. *J. Comput. Chem.* **4**, 187-217 (1983).

This documentation can be easily perused on-line using the INFO facility under GNU Emacs or using an HTML browser pointed to the file, '`$CGD/congen_toc.html`'. INFO provides a text oriented hypertext system where one can navigate throughout the documentation either by the hierarchy of information (chapters, sections, subsections, etc.) or by the cross references.

CONGEN is an non-interactive program. You specify commands in a command file, and they are executed by the program. Error messages and essential results are written to a log file, and other information can be written to files of your own choosing. Commands to CONGEN are generally written in free field format; exceptions are noted in the appropriate manual pages. CONGEN can generate graphics for both non-interactive and interactive use.

Despite the complexities in CONGEN's implementation, CONGEN is designed to be portable. At present, the code has been ported to VAXen running VMS, Silicon Graphics Iris 4D's, Convex C2's, Sun's, Hewlett-Packard Unix, Fujitsu VP series, DEC Alpha Unix, and Cray's running Unix.[1] There are a small number of machine dependent features which must be accommodated when porting. In particular, the program is written using both C and FLECS (a Fortran preprocessor), and the interface between languages requires the use of a special program, `wrapgen`, see Section 30.4 [Wrapgen], page 261.

CONGEN is provided free of charge. It is my intention that the program be used as widely as possible so that all scientists can benefit from the work that many people have put into it. I do not believe that anyone should reinvent the wheel. If you find problems or make improvements, please let me know about them and make them available. My email address is '`bruc@acm.org`'.

Although I try to keep CONGEN bug free, it is not. *Don't trust the program too much.* If you don't understand what it is doing from this manual or its operation, read the source code and run the program with a debugger. The source code and all of the tools necessary to build CONGEN are provided to help you understand it. I especially welcome comments from those who explore its internals.

Note added in 2009: I have not worked on Congen actively for over a decade, having taken my career in new directions. However, I thought it was important to revisit the program, and bring it up to date so that it could compiled on a modern Linux machine and pass its test cases again. This current release does that, but unfortunately, it is now somewhat dated. However, many of the algorithms are still useful, so please make use of it.

[1] A disclaimer is necessary here. Since CONGEN is under continuous development, and because we do not have continuous access to all these machines, "ported" means that it was successfully compiled and executed on those machines at some point prior to the current release. Changes made to the code after the test may not compile, but it should be a simple matter to correct any problems.

2 How to Use CONGEN

This chapter contains an overview of how to use CONGEN. CONGEN executes commands as they are read sequentially from a command file. In general the ordering of these commands is limited only by the requirement that each command must have all of its prerequisite data. For example, the energy cannot be calculated unless the arrays holding the coordinates, the parameters, etc., have already been filled. It is up to the user of program to ensure that everything is present, so it is vital to understand how the program works in order to use it correctly.

Although this manual is extensive, it does not provide much tutorial help in resolving questions about the program's operations. Also, there is not much guidance on how to deal with errors. The full source code for CONGEN is provided with the program, and it should be consulted if this manual does not provide answers to questions that arise with using the program.

This manual is available in on-line form. If you use GNUemacs and if the INFO form of this documentation has been installed, then the manual can be perused while you edit CONGEN input files. See Section 30.14 [UNIX Installation], page 273, or Section 30.13 [VMS Installation], page 272, for more information about the installing the on-line documentation.

2.1 Controlling a CONGEN Run

A CONGEN run is controlled by a command file. The user specifies operations to be performed via these commands, and CONGEN executes them sequentially. There are commands for changing the file from which input is taken, see Section 3.7 [Stream Command], page 41.

A legal command file for CONGEN begins with a specification of the title of the run. See Section 2.12 [Syntactic Glossary], page 12, for the syntax of a title. Then, any number of commands may be specified.

Each command consists of a command line possibly followed by other data. The command line is scanned free field. This command line may be longer than one line in the file; to do this, one must place a hyphen, '`-`', at the end of line which is to be continued on the next line. Comments may be placed on a command line by preceding the comments by exclamation points. Note that comments may be placed after a hyphen — the comment is removed before the hyphen is checked for. All lower case characters are converted to upper case. This format is identical to that used by the VAX command language interpreter. In addition, blank lines are permitted to separate blocks of commands for increased readability.

Generally, when a free field command line is read in, it is echoed onto Fortran unit 6. Each such echo will be prepended by a short marker, eg. `CONGEN>`, which identifies the line of input as well as the command processor which is interpreting it.

The command line is scanned in units of words and delimited strings. A word is defined by a sequence of non-blank characters, A delimited string consists of a keyword followed by a string of characters of variable length followed by a delimiter string. The delimiter string is either a single delimiter character, two such characters concatenated together with no space in between, or the keyword, `END`. The initial value for the delimiter is a '`$`'. It may be changed with the `DELIm` command, see Section 27.3 [Delim Command], page 228.

The first word of every command line specifies the command. Generally, required operands of a command must follow in a precise order. On the other hand, options may generally be specified in any order. Further, any number is always preceded by a key word so that any numeric operands, can be placed in arbitrary order.

Abbreviations are permitted in various contexts. The first word may be abbreviated to four characters except for commands to the analysis facility. Numerous options and operands may also be abbreviated to four characters. However, key words which are used to mark numbers may never be abbreviated. See the description for individual commands to see what can and cannot be abbreviated.

In general, as each of the command is interpreted, it is deleted from the command line. When command processing is finished, a check is made to see that nothing is left over. The presence of extraneous junk indicates that something was mistyped.

Some of the options and numeric values are maintained from one invocation of a command to the next. This is done with the energy manipulation options, see Chapter 7 [Energy Manipulations], page 55. Other options have no memory and must be respecified each time they are changed from their default values. Thus the safest approach is to respecify all options whenever they are used.

Although the command line of every command is processed free field, there are commands (especially ones implemented in the early days of CHARMM) which are followed by data in a fixed format. In general (and there are *many* exceptions), such commands expect integers as `I5` and floating point numbers as `F10.n`. In the event that the documentation is not clear or non-existent on such matters, the source code can be consulted.

2.2 Rules for Describing the Syntax (The Meta-Syntax)

The syntax of commands is described using the following rules: Capitalized words are keywords that must be specified as is. However, if the word is partially capitalized, it may be abbreviated to the capitalized part. Lower case words are to be replaced by a corresponding data entry. The symbol '`::=`' means "has the following syntactic form:". Anything enclosed in square brackets, '`[]`', is optional. If several things are stacked in square brackets, one may choose one optionally. Anything enclosed in curly brackets, '`{}`', specifies that a selection must be made of the choices stacked vertically inside. The syntactic entities which appear as an argument to '`repeat`' may be repeated any number (including zero) times. Defaults for optional parameters may be enclosed in apostrophes and placed under the entity they stand for. However, defaults are not specified in this manner if the rules for the default are complex.

The syntactic glossary, see Section 2.12 [Syntactic Glossary], page 12, contains further syntactic entities which are used in the command descriptions. Finally, the options and operands in each command can usually be specified in any order except if otherwise noted.

As an illustration of the syntax notation, the plot command in the analysis facility has following syntax:

```
PLOT [titspec] [HORIZ real real integer]

titspec ::= TITLE string del
```

Assuming the standard definition of the del, a syntactically acceptable command is

```
PLOT HORIZ 1.0 10.3 60 TITLE BOND ENERGY PER RESIDUE $
```

2.3 Fortran I/O Units Usage by CONGEN

In order to keep CONGEN as machine independent as possible, all specification of files is done through Fortran unit numbers. Two unit numbers have special significance, 5 and 6. Unit 5 is the command file interpreted by CONGEN. Unit 6 is the output file for all printed messages. As commands are read from unit 5, they are echoed on unit 6. On Unix machines, unit 5 is usually the standard input, and unit 6 is the standard output. On VMS machines, unit 5 is assigned to the logical name, `FOR005` which defaults to `SYS$INPUT`, and unit 6 is assigned to the logical name, `FOR006` which defaults to `SYS$OUTPUT`. All other unit numbers have no predefined meaning. The

`OPEN` command, see Section 3.4 [Open Command], page 40, can be used within CONGEN to open a file on a particular Fortran unit. The DCL command, `ASSIGN`, may be used on VMS systems to assign files to units. Unix systems can use whatever mechanism is provided with their Fortran run-time system to achieve this effect. E.g. on the Iris, one can make a symbolic link between '`fort.`*n*' and a file in order to associate unit *n* with the file.

On Unix systems, all file names in `OPEN` commands are translated to lower case. This permits CONGEN input files to be portable from VMS to Unix.

When CONGEN is about to read a file on a VAX/VMS machine, it opens the file as a shared, readonly file. This is done to allow users to share commonly useful files, such as topology files. One should be aware that this feature prevents a user from overwriting any file which he is reading. Any file which is written is not opened shared, so it cannot be used by any other process until CONGEN completes execution.

On most Unix machines, the normal I/O operation opens all files shared, so it is possible to read any file being read or written by CONGEN. When a file is opened for writing, any previous file is overwritten. If two CONGEN processes attempt to open the same file for writing, the result is unpredictable, and it is unlikely that the user will receive a warning message indicating he has made a mistake.

2.4 The CONGEN System of Units: AKMA.

CONGEN uses a distinct system of units, the AKMA system. I.e. **A**ngstroms, **K**ilocalories / **M**ole, **A**tomic mass units. All distances are measured in Angstroms, energies in kcal/mole, mass in atomic mass units, and charge is in units of electron charge. Using this system, the AMKA unit of time is 4.888821×10^{-14} seconds (based on the constants tabulated in Abramowitz and Stegun (1970)). 20 AKMA time units is .978 picoseconds and is sometimes referred to as a `"ps"`.

Angles are given in degrees for the analysis and constraint sections. In parameter files, the minimum positions of angles are specified in degrees, but the force constants for angles, dihedrals, and dihedral constraints are specified in kcal/mole/radian^2. Any numbers used in the documentation may be assumed to be in AKMA units unless otherwise noted.

2.5 Data Structures

There are a number of data structures that CONGEN manipulates. Many of these data structures are important for most operations; others which are less important, are described with the commands that use them. Much more specific information is available in the various common blocks whose extension is '`.FCM`' in the source directories.

The important data structures are given below: Each data structure name is followed by its abbreviation which is used as its name in commands.

1. Residue Topology File (RTF). The residue topology file stores the definitions of all residues. The atoms, atomic properties, bonds, bond angles, torsion angles, improper torsion angles, hydrogen bond donors and acceptors and antecedents, and non-bonded exclusions are all specified on a per residue basis. The RTF also specifies the list of chemical type which are used in the parameter file. This file is required for any calculations.
2. The Parameters (PARA or PARM). The parameters specify the force constants, equilibrium geometries, Van der Waals radii, and other such data needed for calculating the energy. The list of atom type codes comes from the RTF. The parameters are required for any calculation, and they depend on the list of chemical types provided in the RTF. The parameters must be consistent with the topology file in that they must designed together. In addition, there must be one and only one non-bonded parameter for each atom type specified in the topology file.

3. Protein Structure File (PSF). The protein structure file is the concatenation of information in the RTF. It specifies the information for the entire structure. It has a hierarchical organization wherein atoms are grouped into residues which are grouped into segments which comprise the structure. Each atom is uniquely identified within a residue by its IUPAC name; each residue is uniquely identified in the segment by a residue identifier which is the character form of the residue's position in the segment; and each segment is identified by a segment identifier specified by the user. This information is required for any calculations.
4. The Coordinates (COOR). The coordinates are the Cartesian coordinates for all the atoms in the PSF. There are two sets of coordinates provided. The main set is the default used for all operations involving the positions of the atoms. A comparison set (also called the reference set) is provided for a variety of purposes, such as a reference for rotation or operations which involve differences between coordinates for a particular molecule.
5. The Non-bonded List (NBON). The non-bonded list contains the list of non-bonded interactions to be used in calculating the energies as well as optional information about the charge, dipole moment, and quadrupole moments of the residues. This data structure depends on the coordinates for its construction and must be periodically updated if the coordinates are being modified.
6. The Hydrogen Bond List (HBON). The hydrogen bond list contains the list of hydrogen bonds. Like the non-bonded list, this data structure depends on the coordinates and must be periodically updated.
7. The Constraints (CONS). The constraints are harmonic potentials placed on selected atomic positions or on dihedral angles. The purpose of these constraints is to limit motion of those atoms or torsions or to force the molecule to assume a particular conformation. Normally, there are no constraints on the molecule. One should note that this data structure does not hold either constraints related to SHAKE or motion constraints where atoms are prevented from moving entirely.
8. The Internal Coordinates (IC). The internal coordinates data structure contains information concerning the relative positions of atoms within a structure. This data structure is most commonly used to build or modify cartesian coordinates from known or desired internal coordinate values. It is also used in conjunction with the analysis of normal modes. Since there are complete editing facilities, it can be used as a simple but powerful method of examining or analyzing structures.
9. The Images data structure (IMAGES). The images data structure determines and defines the relative positions and orientations of any symmetric image of the primary molecule(s). The purpose of this data structure is to allow the simulation of crystal symmetry or the use of periodic boundary conditions. Also contained in this data structure is information concerning all nonbonded, H-bonds, and ST2 interactions between primary and image atoms.

2.6 Directory Names

In order to simplify the use of CONGEN, all the directories used for storing CONGEN executables, sources, parameter sets, test inputs, etc., are defined with either environment variables (on UNIX) or logical names (on VAX/VMS).

On Unix machines, the CONGEN directories have separate trees for each machine on which CONGEN can be run. These trees begin under the root CONGEN directory. This structure allows one directory tree to be shared using the Network File System (NFS) by multiple hardware architectures. The program, '`$CGROOT/update_tree`', is used to update different machine trees from the master copy. See Section 30.14 [UNIX Installation], page 273, for more information about the machine designations.

The `OPEN` command, see Section 3.4 [Open Command], page 40, is designed to allow the usage of identical syntax on either VMS or UNIX for using these directory names. The directory is placed before the file name with a colon, `:`, separating the two. E.g. '`CGDATA:RTOPH8.MOD`'. Also, all file names are converted to lower case.

When referring to the directory names in operating system commands, it is necessary to keep the operating system in mind. On VMS, one uses a colon to separate the two as above. On UNIX running the C-shell, one uses a dollar sign prefix and a slash. For example, '`$CGDATA/rtoph8.mod`'.

The directory names are given below. Unix syntax is used for the directory names, but a similar structure is also used under VMS.

'`MM`'	Specifies the disk where CONGEN is kept. This is used only on VMS.
'`CGROOT`'	Specifies the root directory for CONGEN. This is used only on Unix.
'`CG`'	'`$CGROOT/<machine>/v<version-number>/`' Top level for a particular implementation of CONGEN.
'`CGBIN`'	'`$CG/bin/`' Location of executables. This directory will be put into your path by initialization file such as '`cgdefs`', see Section 30.14 [UNIX Installation], page 273.
'`CGD`'	'`$CGROOT/doc/`' Documentation and INFO.
'`CGDATA`'	'`$CGROOT/data/`' Data files for normal execution.
'`CGLIB`'	'`$CG/lib/`' Object libraries and files for building user versions of CONGEN and for the support programs.
'`CGP`'	'`$CG/support/`' Support programs for use with CONGEN. See Chapter 29 [Support], page 249, for details.
'`CGPS`'	'`$CG/support/src/`' Source code for support programs for use with CONGEN. There are several subdirectories under this one which contain larger support programs, too bulky to keep in this directory.
'`CGS`'	'`$CG/source/`' Source code for CONGEN.
'`CGT`'	'`$CG/test/`' Test cases.
'`CGTD`'	'`$CG/testdat/`' Data files for testing the recent production version.

2.7 Files Available for General Use

There are number of residue topology files, parameter files, coordinates files and files of other data structures available. The most important files generally available are residue topology and parameter files. Both such classes of files are stored for general use in the directory, '`CGDATA:`'. The file names used for both these files consists of an alphabetic part followed by a number, e.g. '`PARAM5`'. There are two copies of each file; one with extension, '`.INP`', which is a character files used as an command file to generate the binary file, with extension, '`.MOD`'. The '`.INP`' is meant for human eyes; the '`.MOD`' files is meant for CONGEN to read.

The numeric part of each name is its version number. In general, one should use the highest version number of a file.

The current list of files which are in the best condition are as follows:

‘`PARAM5`’	Parameters for all protein topology files.
‘`RTOP8`’	Extended atom (no hydrogen) topology file. This topology has not been tested with conformational search.
‘`RTOPH8`’	Explicit hydrogen (polar hydrogens only) topology file.
‘`RTOPALLH7`’	All hydrogen topology file.

The DNA files are not usable with extensive review. The chemical type codes are not consistent with proteins, and the linkage scheme or the patching code has not been tested in years.

The directory, ‘`CGTD`’, contains a few protein coordinates assembled from a variety of sources.

For information on the general use of directories, and the files they contain, see Section 2.6 [Directory Names], page 8.

2.8 Size Limits

At present, the description of the molecule is stored in fixed arrays in Fortran common blocks. As a result, there are limits to the size of various entities used to describe a molecule. The limits are given in the file `$CGS/aaalimits.fcm`.

2.9 Sample CONGEN Runs

There are several examples of CONGEN runs within this manual, e.g. Section 12.6 [Examples of Conformational Search], page 137. Also, the input files found in the test data directory, ‘`CGTD`’, give many examples of CONGEN input. The test case directory, ‘`CGT`’, has many CONGEN runs, but these are less useful to the novice because they are designed to test program features, not get any useful work done.

2.10 DCL or Shell CONGEN Commands

There are several shell or DCL commands for running CONGEN. In addition to these commands, on Unix systems, CONGEN can be executed directly by typing `congen`.

Syntax: `runcg [-i] [-n` ***`integer`***`]` *filespec*

Function: This command will make logical name assignments for CONGEN normal I/O units and run CONGEN. The filespec must not specify any file type (file extension) or version. On VMS, `RUNCGn` will assign *filespec*.`INP` to `FOR005` and *filespec*.`OUT` to `FOR006` and then run CONGEN. The `-i` and `-n` ***`integer`*** options are not permitted on VMS. On UNIX, it will run CONGEN with standard input and output being set to *filespec*.`inp` and *filespec*.`out`. The `nice` command will be used with an argument of 4 to lower the runtime priority. The option, `-i`, specifies that interactive priority should be used. The option, `-n` ***`integer`***, specifies an alternate nice value.

Syntax: `rc`

Function: Run CONGEN. All logical name assignments for Fortran units must be set beforehand.

Syntax: `ruc [-i] [-n` ***`integer`***`]` *filespec* [*directory*]

Function: Same as RUNCG except the directory from which CONGEN is taken is specified in the command. If no directory is specified, then the current default directory is used.

2.11 Interfacing to CONGEN

A mechanism has been provided to allow casual users of the CONGEN to write their own special purpose subroutines which can be incorporated into the system without threatening its integrity.

There are two "hooks" into the protein system which have been specially provided for casual modifiers. The first is the `USER` command which invokes the subroutine, `USERSB`, and performs no other action. `USERSB` is a subroutine with no arguments. However, parameters may be passed to this subroutine via the system's `COMMON` blocks. These `COMMON` blocks store nearly all of the systems data. These common blocks may be obtained by including them from the directory containing the sources for the version of the program you are using.

The second hook is the user energy function `USERE`. `USERE` is a subroutine that is called whenever the total energy and its derivatives are calculated. `USERE` has ten parameters (in order):

`EU`	to be returned with the user energy

`X`
`Y`
`Z`
	the current coordinates

`DX`
`DY`
`DZ`
	to be returned with the derivatives if `ANALYS` is zero

`ANALYS`	a mode flag set to 0 if the total user energy and its derivatives are needed, and 1 if the atom energies are desired (in analysis)
`DATA`	to be returned with the atom energies if ANALYS is set to one.
`NATOM`	the current number of atoms.

Thus the call should be:

```
CALL USERE(EU,X,Y,Z,DX,DY,DZ,ANALYS,DATA,NATOM)
```

`EU` should be set to the value of the user energy upon return. If this value is non-zero, it will be automatically printed each time the energy is evaluated. The system supplied version of `USERE` does nothing except set `EU` to zero. The coordinates are supplied via `X`, `Y`, and `Z`. Derivatives of the user energy must be added to `DX`, `DY`, and `DZ`. All other information used by `USERE` must be obtained through the system's common blocks as is the case with `USERSB`. Older versions of `USERE` that used only the first seven variables of the call can still be linked and run in the main section of the program, but will fail in the analysis section.

When using `USERE` fill only the arrays that are being requested by `ANALYS` (otherwise you will get access violations). In the analysis section of the program *do not assume* that the common block information will be correct for a comparison data structure, it may not be.

To simplify the use of these hooks and to allow users to replace subprograms in CONGEN with their own versions of said subprograms, the makefile, '`$CGS/usermake`', has been provided for Unix, and the equivalent MMS descriptor file, '`CGS:USERMAKE.MMS`' has been provided on VMS. These makefiles will produce a private version of CONGEN in your default directory using your version of '`usersb.flx`'. If you need to change more files, then copy '`usermake`' into your working directory, and modify it accordingly.

Before attempting to write your own user functions, you should familiarize yourself with the information available on the implementation of CONGEN, see Chapter 30 [Implementation], page 259.

There are several utility routines available to a user routine. Some of them are listed below.

`CALL GETE(X,Y,Z)` will cause the energy and forces to be computed and values are saved in the appropriate common blocks. For this to work properly, `NBONDS`, `HBONDS`, and `CODES` must have been called. This can be done by executing both the `NBONds` and `HBONds` command, or by having previously found the energy (minimization, dynamics, etc.).

`CALL PRINTE(IUNIT,ICYCLE,LHDR)` will write the current energy values (from common block values) to the specified unit, `IUNIT`. It will also write out the cycle or iteration number, `ICYCLE` and optionally write out the standard header if `LHDR` is `TRUE`.

2.12 Glossary of Syntactic Terms

`atom-selection`	A description of a set of atoms. See Section 11.7 [Atom Selection], page 108, for the complete syntax.
`char`	A character
`del`	The delimiter - a single character which is used to mark the end of a portion of a command. Initially, it is a dollar sign but can be changed using the `DELIM` command, see Section 27.3 [Delim Command], page 228. It should be noted that the delimiter cannot be a character within any string it is supposed to delimit.
`deldel`	Two delimiters concatenated together with no space in between.
`integer`	An integer.
`iupac`	IUPAC name for an atom. Initially specified in the residue topology file.
`keyword`	A word, see below, serving to identify some option.
`name`	A word, see below, serving to identify an object.
`property`	A table property which is syntactically a string. See Section 17.1 [Building Tables], page 159, and a description of dynamical properties, see Section 19.4 [Dynamical Table Properties], page 184.
`range`	equivalent to real real integer. The first real is the minimum value in the range, the second number is the maximum value in the range, and the third number gives the number of interval, i.e. lines or columns.
`real`	A real number. No decimal point is required for the number to be interpreted correctly.
`resid`	Residue identifier (a string of up to 4 characters).
`resname`	Residue name.
`segid`	Segment identifier (a string of up to 4 characters).
`string`	An ordered set of characters.
`tag`	A string which is a tag, i.e. no embedded spaces.
`title`	A series of 1 to 10 lines of text (max 80 characters per line) terminated by a line which has an asterisk '`*`' as the first character. Used for commenting files.
`word`	A string with no blanks
`unit-number`	An integer which is a Fortran unit number.

2.13 General Glossary

data structure	A collection of arrays, scalars, and possibly other data structures which are related by part of a larger entity. For example, a coordinate set is a data structure which hold the three dimensional positions of atoms. This data structure consists of 1 scalar and three arrays. The scalar is the number of coordinates; the three arrays are the X, Y, and Z components of the coordinates.
Internal Coordinates	Bonds, angles, torsions, improper torsions. Also, a data structure used for constructing coordinates.
Iupac Names	The name of an atom with a residue. This name should be within a residue and should conform to the IUPAC nomenclature. See *Biochemistry* **9**:3471 (1970).
Hbonds	Hydrogen bonds.
Parameters	Constants in the energy expression (force constants, minima of energy surfaces, charges, Lennard-Jones parameters, van der Waals radii, etc.)
PSF	Protein Structure File a list of the internal coordinates and related information
Residue Identifier	A string of four characters or less which uniquely specifies residue with in a segment. This value is currently set by CONGEN to be the character representation of the residue number in the segment starting from the first real monomer unit in it. Special terminating residues gets their names as identifiers. For example, if we build a tripeptide `LYS ARG ASP` using an explicit hydrogen topology file, we get five residues in the segment, `NTER LYS ARG ASP CTER`, and the residue identifiers are then '`NTER`', '`1`', '`2`', '`3`', and '`CTER`'.
RTF	residue topology file : a list of standard internal coordinates, atom charges, atom types, excluded non-bonded interactions, etc.
Segment Identifier	A string of up to four characters uniquely designating a segment. Specified in the GENErate command, see Section 4.1 [Generate Command], page 43.
Sequence	list of residues.

3 Input-Output Commands

The commands described here are used for reading and writing data structures used in the main part of CONGEN. Some of data structures used in the analysis facility may also be read and written, see Chapter 21 [I/O in Analysis], page 197.

3.1 `READ` — Reads Data from External Sources

This command reads data into the data structures from external sources. The external sources can be either text files (card images for the ancient among us) or binary files. The fortran unit number from which the information is read, is specified with the unit-spec. Specifying `UNIT 5` indicates that the data to be read following the command in the input stream.

The precise format of all these files is described only in the source code as that serves as the only definitive, accurate, and up to date description of these formats. The description of the data structures provides pointers to the subroutines which should be consulted, see Section 2.5 [Data Structures], page 7.

3.1.1 Syntax of READ Command

```
READ { {RTF [PRINT]         } { FILE unit-spec   } }
     { {PARAmeter parm-opts } { CARD [unit-spec] } }
     { {IC  [APPEnd]        } {      'UNIT 5'    } }
     {                                             }
     { {SEQUence} seq-spec                         }
     { {RESIdue }                                  }
     {                                             }
     { {HBONd     } [FILE] unit-spec               }
     { {PSF       }                                }
     { {CONStraint}                                }
     { {NBONd     }                                }
     {                                             }
     { { IMAGes     } [CARD] unit-spec             }
     {                                             }
     {                      [ MAIN ]               }
     { COORdinate coor-spec [ COMP ]               }
     {                      [ DIFF ]               }

unit-spec ::= UNIT unit-number

parm-opts ::= [NOFAIL] [VERSION int]

             [ST2  int                                          ]
seq-spec ::= [WATEr int                                         ]
             [[src-spec] [unit-spec] [rtf-type] [abbrev-spec] seq-opts]


seq-opts ::= [BYATom] [IDREad] [MODEl modelid]


             { CARDs                         }
src-spec ::= { brookhaven-name [CHAIN segid]}
```

```
                     { BROOkhaven }
brookhaven-name ::= { BRKHvn     }
                     { TAPE       }

              { PROT }
              { HPRO }
              { ALLH }
rtf-type ::= { DNA  }
              { A94N }
              { A94P }

                         {AA }
abbrev-spec ::= [ABBREV {DNA}]
                         {RNA}

coor-spec ::= { FILE unit-spec [IFILE int]       } coor-option
               { CARD [unit-spec] [OFFS  int ]    }
               { IGNO [unit-spec]                 }
               { KONN [unit-spec]                 }
               {                                  }
               { { BROOkhaven } unit-spec brk-opt }
               { { BRKHvn     }                   }

coor-option ::= [APPEnd] [INITial] [EXPAnd]

                [abbrev-spec] atom-selection

brk-opt ::=  [CHAIN segid] [IDREad] [MODEl modelid]

             [ALTErnate identifier] [  SEQUence]
                                    [NOSEQUence]
```

Syntactic ordering: The second field must be specified as shown.

3.1.2 Specifying a Sequence of Residues for a Segment

The specification of `SEQUence` or `RESIdue` causes the program to accept a sequence of residue names to be used to generate the next segment in the molecule. There are four sources of sequence information. The first source is a CONGEN format sequence file which has the following syntax:

```
title
number-of-residues repeat(residue-names)
```

The form of the title is defined in the syntactic glossary, see Section 2.12 [Syntactic Glossary], page 12. The number of residues is specified on the line following the title in free field format. If the number of residues you specify is less than zero, CONGEN will read residues until it encounters a blank line or end of file. If the number is greater than zero, it will also stop once it has read at least as many residues as you've specified. If the number you specify is zero, you will get a warning message as one common error is to forget the number entirely. In this case, the first residue name will be consumed as the number and converted to zero.

The residue names are specified as separate words, each no longer than 4 characters, on as many lines as are required for all the residues. This sequence may be placed immediately following the `READ` command if the unit number is 5 or may be placed in a separate file.

The second source of sequences is a CONGEN coordinate file in CARD format. Currently, the `BYATom` option reads all residues within the file for inclusion in the sequence.

The third source of sequence information is a Brookhaven Protein Data Bank file. The `BROOKHAVEN`, `BRKHVN`, and `TAPE` options allow the sequence to be read from the `SEQRES` records in a Brookhaven protein data bank coordinate file. (`TAPE` is used because the Brookhaven protein data bank used to come on a tape.) If the `CHAIN` option is specified, then only the sequence of chain with the specified `segid` is read. Otherwise, the sequence of all the chains will be read together. Note that the Brookhaven format only allows single letter chain names, so your `segid` should only have one character.

Alternatively, the sequence may be read directly from the `ATOM` records by using the `BYATOM` option. Under the `BYATOM` option, if there are insertions or deletions in the within the list of residue idenitifiers, the `IDREAD` option will read the sequence identifiers, including insertion codes, directly from the Brookhaven file, rather than automatically generating a residue number based on sequential order. It should be noted that currently, the `IDREAD` option conflicts with the `DISULPHIDE` command, since this command assumes that the residue identifiers are those generated automatically. The `MODEL` option may also be used in conjunction with `BYATOM` to read the sequence from a particular model number in the file. If not specified, the first model in the brookhaven file is used.

The final source of sequences are the two water options. The `WATEr` option allows a sequence of water molecules to be specified. The integer which follows the keyword gives the number of waters. Likewise, the `ST2` option allows ST2 waters to be specified. Obviously, no sequence on separate lines need be given. For CONGEN topology files, a residue named `OH2` (or `ST2`) must be present. For AMBER94 topology file, a residue named `HOH` must be present. If these residues are missing, the `GENErate` command called afterwards will fail.

When reading is complete, CONGEN will list all the residues it has read.

The options; `PROT`, `HPRO`, `ALLH`, and `DNA`; specify what type of CHARMM potential file is being read. They are very important because they specify which patching operations are to take place on the segment once it is generated. The patching operations involve correcting the linkage of prolines, and correcting the charges and chemical types of the ends of the segment. `PROT` signifies that we are using an extended atom residue topology file as the source of residues. `HPRO` signifies that we have an explicit hydrogen topology file being used. `ALLH` signifies that we have an all hydrogen topology file. `DNA` specifies that we are working with the DNA topology file.

In addition, these options may cause additional residues to be added to the sequence. These additional residues serve to terminate the segment. However, if the segment is generated cyclically (see Section 4.1 [Generate Command], page 43), then no termini will be added. In particular, `PROT` will add a CTER residue that has the C-terminal oxygen. `HPRO` and `ALLH` will add a CTER residue along with an NTER residue that holds two additional hydrogens for the N-terminus. `DNA` will add a 5TER to the beginning and a 3TER residue to the end of the segment.

If an AMBER94 topology file is being used, then the keywords, `A94P` or `A94N`, should be specified to indicate whether a protein or nucleic sequence is being read. Use of these keywords will then result the correct terminal residues being used at the ends of the segment. See Section 4.1 [Generate Command], page 43, for more information about this process.

The `ABBREV` option allows the specification of residues using one letter abbreviations. When the `AA` keyword is specified, one letter amino acid codes can be used. For `RNA` and `DNA`, one letter nucleotide names will be translated into the appropriate two letter AMBER94 residue names.

3.1.3 Reading coordinates

The reading of coordinates is done with the `READ COOR` command, and there are several options (which may change over in future versions).

There are four possible file formats that can be used to read in coordinates. They are coordinate binary files, dynamics coordinate trajectories, coordinate card images, and Brookhaven Protein Data Bank files.

For all formats, a subset of the atoms in the PSF may be selected using the standard atom selection syntax. For binary files, this is a risky maneuver, and warning messages are given when this is attempted. Only coordinates of selected atoms may be modified. When reading binary files, or using the `IGNORE` keyword, coordinate values are mapped into the selected atoms sequentially (NO checking is done!).

The reading of the first two file formats is specified with the `FILE` option. The program reads the file header to tell which format it is dealing with. The coordinate binary files have a file header of `COOR` and contain only one set of coordinates. These are created with a `WRIT COOR FILE` command. The dynamics coordinate trajectories have a file header of `CORD` and have multiple coordinate sets. These files are created by the dynamics function of the program. To specify which coordinate set in the trajectory to be read, the `IFILE` option is provided. One specifies the coordinates position within the file. The default value for this option will cause the first coordinate set to be read.

For binary files, the `APPEnd` command will 'deselect' all atoms up to the highest one with a known position. This is done in addition to the normal atom selection. This is useful for structures with several distinct segments where it is desirable to keep separate coordinate modules.

The `CARD` file format is the standard means in CONGEN for providing a human readable and writable coordinate file. The format is as follows:

```
title
NATOM (I5)
ATOMNO RESNO   RES  TYPE  X      Y      Z  (repeated NATOM times)
  I5    I5  1X A4 1X A4 F10.5 F10.5 F10.5
```

The `title` is a title for the coordinates, see Section 2.12 [Syntactic Glossary], page 12. Next comes the number of coordinates. If this number is zero or too large, the entire file will be read. Finally, there is one line for each coordinate. The coordinates, but not the initial lines, may contain blank lines for readability

`ATOMNO` gives the number of the atom in the file. It is ignored on reading. `RESNO` gives the residue number of the atom. It must be specified relative to the first residue in the PSF. The `OFFSet` option should be specified if one wishes to read coordinates into other positions. The `APPEnd` option adds an additional offset which points to the the residue just beyond the highest one with known positions. This option also 'deselects' all atoms below this residue (inclusive). For example, if one is reading in coordinates for the second segment of a two chain protein using two card files, and the `APPEnd` option is used, `RESNO` must start at 1 in both files for the file reading to work correctly.

It should also be remembered that for card images, residues are identified by *residue number*. This will change someday. What this implies, is that if one wishes to read coordinates from an extended atom (`PROT`) RTF into a structure using an explicit hydrogen (`HPRO`) RTF, the `OFFSet` keyword *MUST* be used to shift the residue numbers by one, (to make room for the NTER) so that the residues will line up. If the reverse process is required, an `OFFSet` value of -1 is called for.

`RES` gives the residue name of the atom. `RES` is checked against the residue name in the PSF for consistency. `TYPE` gives the IUPAC name of the atom. The coordinates of an atom within a residue need not be specified in any particular order. A search is made within each residue in the PSF for an atom whose IUPAC name is given in the coordinate file.

The `MAXERR` option controls how many error messages are printed. Its default value is 10. Normally, the coordinate reader will scan the entire file, and it will list errors as it encounters them, until to the `MAXERR` limit. At the end of reading, it will terminate execution if any fatal errors were encountered.

The `KONN` option allows the reading of Konnert Hendrickson format files. The file consists of just atom records where each atom coordinate has the following format:

```
         Res Segid Resid Iupac X Y Z
      3X,A4,  A1,   A3,   A4,  3F10.5
```

The four alphabetic fields are left justified by the program so they can be placed anywhere within their columns. If the `Segid` is not specified, the program will attempt to place the atoms within a segment which is determined by the `APPEnd` option (above). If `APPEND` is not specified, then the first segment in the structure will be used. If `APPEND` is specified, then the first segment which has a residue with all undefined atoms will be used. Blank lines may be specified between coordinates.

Note that the `Segid` and `Resid` fields are too small to hold the maximum length values. Truncations will cause unavoidable problems. However, residue identifiers NTE and CTE are extended to NTER and CTER.

The `BROOKHAVEN` option (or its synonyms, `TAPE` or `BRKHVN`) specify that the coordinate file is in the Brookhaven Data Bank format. CONGEN can read the `ATOM` records for coordinates. However, because the Brookhaven format uses slightly different naming conventions, there are a number of inconsistencies you should be aware of when using this option:

1. Chain identifiers in Brookhaven are only one character long. In CONGEN, the corresponding segment identifiers are currently four characters. Thus, if you read a Brookhaven file with chain identifiers, you must generate your segments with one character identifiers (see Section 4.1 [Generate Command], page 43). If no chain identifiers are present in the Brookhaven file, then CONGEN will search the coordinates for the first residue which has all undefined atoms. Then, it will add the value you specify for `OFFSET` to this number, and it will read coordinates into the segment which contains the offsetted residue. Be careful in the case where the terminal atom is undefined because in the protein and DNA cases, that atom is in a residue all by itself.
2. The sequence number in the Brookhaven data bank is an amalgam of a residue number and an insertion code, whereas in CONGEN, it is a four letter identifier which is usually just the text representation of the sequence number (except for the terminating residues). There are two ways that you can handle this number in CONGEN. The `SEQUENCE` option, which is the default, causes CONGEN to assume that the atoms are provided in sequence order, and that every change of sequence number or insertion code in the file implies that the next residue is being specified. When insertions and deletions are present, the `IDREad` option is used to read the both the residue number and the insertion code. Note that this option must used in conjunction with reading the sequence with the `IDREad` option. If `NOSEQUENCE` is specified, then the residue number is used, but the insertion code is ignored.
3. Hydrogen atoms have different specifications. In some cases the final digit of the hydrogen name is placed before the '`H`'. In others it is placed after. CONGEN will move any digit found before the '`H`' after the other atoms in the name.
4. Sometimes, the hydrogen atom attached to the peptide nitrogen is labeled '`HN`'. If so, it is renamed to '`H`'.
5. Atoms at the terminii, such as '`NT`' and '`OT1`' are renamed.

6. Atoms can be defined at different positions in the Brookhaven Data Bank. CONGEN will use the last value found in the file.
7. If you wish to select a particular alternate location identifier for a set of coordinates, use the `ALTERNATE` option along with the one-letter identifier desired.
8. Multiple models may be handled by using the `MODEL` option. In these cases, it is necessary to specify the model number you wish to use.

Reading Brookhaven file format is not straightforward, so check the coordinates after they are read to see if there are correct. Energy evaluations (see Chapter 7 [Energy Manipulations], page 55 followed by analysis of the geometric terms (see Chapter 16 [Analysis], page 157) are a useful way to do this. Also, the `brkchm` command (see Section 29.1.4 [Brkchm], page 250) is an alternate way of converting Brookhaven files into a form that can be edited.

The `IGNORE` option allows one to read in a card coordinate file while bypassing the normal tests of the residue name, number, and atom name. When `IGNORE` is specified in place of card, the identifying information is ignored completely. Starting from the first selected atom, the coordinates are copied sequentially from the file.

Normally, the coordinates are not reinitialized before new values are read, but if this is desired, the `INITIALIZE` keyword, will cause the coordinate values for all selected atoms to be initialized. Note that only atoms that have been selected, will be initialized. The `COOR INIT` command provides a more general way to initialize coordinates.

The `EXPAnd` option should be specified if the following conditions apply:

1. An explicit hydrogen topology file is being used, and the coordinates we are reading do not have hydrogens in them.
2. The coordinates were read using the `IGNORE` option or were read from a binary file.

In this case, the coordinates will be shuffled in order to leave room for the hydrogens. The hydrogen bond generation routine, Section 15.2.17 [HBUILD Command], page 155, or the builder routines, Chapter 14 [Internal Coordinates], page 147, must be called to construct the positions of these hydrogens.

It is also possible to read coordinates into the comparison (or reference) set using the `COMP` keyword. The `DIFF` keyword will read coordinates into the coordinate differences (also referred to as the normal mode arrays). It expected that these “coordinates” are really displacements that will be processed by the vibrational analysis command, see Chapter 10 [Vibrational Analysis], page 89.

Currently, CONGEN will perform a limited set of name translations on any formatted coordinate reading operation. The isoleucine translations are not needed for the AMBER 94 topology file, see Section A.6.5 [AMBER94RTF], page 297. represent common differences in nomenclature:

Residue	Input ⇒ Final
ILE	CD1 ⇒ CD
ILE	HD11 ⇒ HD1
ILE	HD12 ⇒ HD2
ILE	HD13 ⇒ HD3
SER	HG1 ⇒ HG
OH2	O ⇒ OH2

The `ABBREV` option allows the specification of residue names using one letter abbreviations. When the `AA` keyword is specified, one letter amino acid codes can be used. For `RNA` and `DNA`, one letter nucleotide names will be translated into the appropriate two letter AMBER94 residue names.

Finally, the reading of coordinates is always a tricky business. Although standards exist for naming conventions, there are enough minor variations to make the situation difficult. Always check the structure after reading coordinates to ensure that the geometries and energies are reasonable.

3.1.4 The Format of Parameter Files

In 1995, the parameter file format was completely redone to accomodate the addition of the AMBER potential, see Section B.1.12 [AMBERPARM], page 306. The new format is free field and allows patterns to be supplied for parameters in order to reduce the size of the file and to allow for default parameters to handle molecules which have not been seen before.

For the bond length, bond angle, torsion angle, and improper torsion parameters, CONGEN stores a patterns to match the atom types along with the relevant force field parameters. When the programs needs to calculate the energy of any internal coordinate, it goes sequentially through the patterns, and upon finding the first pattern which matches the atom types of the internal coordinate with the highest "specificity", it uses the corresponding parameters. In this context, "specificity" means the sum of the specificities of each atom type pattern. The specificity of an atom type pattern is 0 if it's a complete wildcard, "*"; 0.5 if any wildcards are present, and 1.0 if there are no wildcards at all. This scheme allows parameters to be specified in different levels of generality, with specific parameters taking precedence over general ones.

In the case of hydrogen bond and non-bonded interactions, all the possible combinations of atom types are computed, and tables for the parameters are constructed. Patterns are used to match against the atom types when the tables are computed.

Parameters are stored as strings with the first character being either "S" or "P", which means string or pattern, respectively. If a pattern is a string, then the program does a simple string comparison; otherwise, a wildcard match is used, see Section 11.7.2 [Interpretation of Atom Selection Tokens], page 109. CONGEN checks the patterns you specify to see whether a pattern or string has been specified.

Parameter files can be read in either text or binary format. For text files, the version can be set using the `VERSION` keyword on the `READ PARAMETER` command. The default value of 2 specifies the old format. The new version is specified by using a value of 3. For binary files, the header record indicates which version of parameter file is being read.

3.1.4.1 Parameter File Format

The text format for the parameter file begins with a title, see Section 2.12 [Syntactic Glossary], page 12, followed by a set of free field commands, and terminating with the end of the file or an `END` statement. The purpose of the commands is to fill the various parameter arrays. The commands are described below:

`BOND` command

Syntax

```
BOND repeat(word word) FORCe real DISTance real
```

Function

The `BOND` command adds bond parameters. The bond energy term for one bond is given by

$$E_b = k_b(b - b_0)^2$$

where k_b is the force constant and b_0 is the equilibrium bond length.

The force constant is given by the `FORCE` keyword and the equilibrium bond length is given by the `DISTANCE` keyword. Each pair of `words` is treated as a separate entry in the bond parameter arrays, so it is possible to specify the same parameters for many bonds.

`ANGLE` command

Syntax

```
{ANGLE} repeat(word word word) FORCe real ANGLE real
{THETA}
```

Function

The `ANGLE` command adds angle parameters. The angle energy term is given by

$$E_\theta = k_\theta(\theta - \theta_0)^2$$

where k_θ is the force constant and θ_0 is the equilibrium bond angle. Bond angles are defined over triplets of atoms. The force constant is given by the `FORCE` keyword and the equilibrium angle is given by the `ANGLE` keyword. Each triplet of `words` is treated as a separate entry in the angle parameter arrays, so it is possible to specify the same parameters for many angles.

`TORSION` command

Syntax

```
{TORSION} repeat(word word word word) repeat(torsion-term)
{PHI    }

torsion-term ::= TERM FORCe real PHASe real PERIod real

                                   MULTiplicity int END
```

Function

The `TORSION` command adds torsion angle parameters. The torsion angle term has the following form

$$E_\phi = \sum_i \frac{k_{\phi_i}}{m_i}(1 + \cos(n_i\phi + \delta_i))$$

where i is over all the terms for one torsion angle, k_{ϕ_i} is the force constant for the ith term, m_i is the multiplicity for the ith term, n_i is the periodicity of the ith term, δ_i is the phase for the ith term, and ϕ is the current value of the torsion angle.

Torsion angles are defined over quadrulets of atoms, and there can be multiple terms per torsion angle so that complex torsions can be established. Each term is specified by strings beginning with `TERM` and ending with `END`. The force constant for each term is given by the `FORCE` keyword. The phase is given by the `PHASE` keyword. The periodicity is given by the `PERIOD` keyword, and limited to values of 1, 2, 3, 4, and 6. The multiplicity is given by the `MULTIPLICITY` keyword, and is most useful in using the AMBER force field, see Section B.1.12 [AMBERPARM], page 306. At least one term must be specified for a torsion angle. Each quadruplet of `words` is treated as a separate entry in the torsion parameter arrays, so it is possible to specify the same parameters for many torsions.

`IMPROPER` command

Syntax

```
{IMPROPER} repeat(word word word word) FORCe real improper-term
{IMPHI   }

improper-term ::= {PHASe real PERIod real}
                  {MIN real              }
```

Function

The `IMPROPER` command adds improper torsion parameters. If the dihedral form of improper torsion is selected, the improper torsion term use the torsion angle term given above. If the harmonic form of the improper torsion is selected, then the improper torsion energy term is given by

$$E_i = k_i(\phi_i - \phi_{i_0})^2$$

where k_i is the force constant and ϕ_{i_0} is the equilibrium improper torsion. Improper torsions are defined over quadruplets of atoms. The force constant is given by the `FORCE` keyword. If the dihedral form of the energy is used, then the phase and period are given by the `PHASE` and `PERIOD` keywords, respectively. The multiplicity is set to 1. If the harmonic form is used, then the equilibrium improper torsion angle is given by the `MIN` keyword. Each quadruplet of `words` is treated as a separate entry in the improper torsion parameter arrays, so it is possible to specify the same parameters for many improper torsions.

`HBOND` command

Syntax

```
HBOND repeat(word word) {EMIN real RMIN real             }
                        {CREPulsive real CATTractive real}
```

Function

The `HBOND` command adds hydrogen bond parameters. The form of the hydrogen bond term is given by

$$E_{hb} = \left(\frac{C_r}{r^{12}} - \frac{C_A}{r^{10}}\right) cos^4(\theta_{HB})$$

There are two different ways to calculate hydrogen bond energies. The form in the old CHARMM potential uses the distance between the heavy atom attached to the donor hydrogen and the acceptor, and angular term based on the heavy atom donor, donor hydrogen, acceptor angle. The form used by the AMBER potential uses the distance between the hydrogen and the acceptor, and no angle term. The `DEFAULT` command described below allows you to switch from one form to the other.

There are two ways to specify the two coefficients. They may be specified directly using `CREPULSIVE` to specify the first coefficient, and `CATTRACTIVE` for the second. The second way is to specify the minimum energy, keyword `EMIN`, and minimum energy distance, keyword `RMIN`, and CONGEN will compute the coefficients for you.

The pairs of `words` in each command specifies pairs of atom type patterns to be used for setting the coefficients. The first pattern in the pair gives the atoms types for the donor, being heavy atom or hydrogen. The second pattern gives the acceptor.

The actual process of setting hydrogen bond parameters is complicated by the requirement for constructing a table of hydrogen bond codes so that hydrogen bond codes can be looked up rapidly. Pseudocode for the operation is as follows:

```
For Ih = 1 to Number of Hydrogen bond patterns
   For I = 1 to Number of Atom Types (NATC)
      If atom_type(I) matches pattern(1,Ih)
         For J = 1 to NATC
            If atom_type(J) matches pattern(2,Ih)
               HBCODE = I*NATC+J-1
               if (HBCODE is not in current list of HB codes)
                  add new HBCODE and coefficients.
               fi
            fi
         done
      fi
   done
done
```

`NBOND` command

Syntax

```
{NBOND    } repeat(word) [EMIN real       ]
{NONBONDED}              [RADIUS real     ]
                         [ALPHa real      ]
                         [NEFF real       ]
                         [CREPulsive real ]
                         [CATTractive real]
```

Function

The `NBOND` command adds non-bonded energy parameters. The nonbonded energy function is

$$E_{nb} = \frac{A}{r^{12}} - \frac{B}{r^{6}}$$

Non-bonded energy parameters are specified only by atom types, and mixed parameters are specified using the combination rules in the CHARMM paper, see Chapter 1 [Introduction], page 3, for the reference.

In each `NBOND` command, the `word`s are atom type codes. The options have the following meanings:

`EMIN`	Minimum van der Waals energy (kcal/mole).
`RADIUS`	Van der Waals radius (Angstroms).
`ALPHA`	Atomic Polarizability (cubic Angstroms)
`NEFF`	Number of effective electroncs (dimensionless)
`CREPulsive`	Coefficient A above.
`CATTractive`	Coefficient B above.

The parameters can be specified in three different ways; by the 6-12 coefficients (`CREPULSIVE` and `CATTRACTIVE`, by minimum energy (`EMIN`) and radius (`RADIUS`), or by radius (`RADIUS`), number of effective electrons (`NEFF`), and polarizabilities (`ALPHA`). The program does not check if you overspecify options, so pick one method and use it consistently.

DEFAULT command

Syntax

```
DEFAULT [IMPRoper [COSIne  ] [NOSYmmetry] END]
                  [HARMonic] [SYMMetry  ]

        [HBOND [H-A] END]
               [D-A]

        [NBOND [VDW14 real] [EL14 real] [HBEXclude] END]
                                        [HBINclude]
```

Function

The `DEFAULT` command is used to set defaults which pertain how some of energy terms are calculated. These defaults are set in the parameter file because the parameters are developed as an integrated whole. Settings in the `DEFAULT` command are an integral part of any parameter file.

From the syntax, it can be seen that there are three different energy terms to which these defaults can apply. The `IMPROPER` options control the following aspects of the improper torsion energy:

`COSINE` The improper torsion term is calculated using the trigonometric function used for the torsion angle term. This form was developed for the implementation of the AMBER potential.

`HARMONIC` The improper torsion term is calculated using the harmonic version of the potential. This form was developed for the CHARMM potential. This is the default.

`SYMMETRY`
`NOSYMMETRY`
These keywords control the matching of improper torsions against the parameters. If `SYMMETRY` is selected, then matching of the four atoms in an improper torsion is attempted in the original order, with the first and fourth atoms swapped, with the second and third atoms swapped, and with all atoms reversed in position. `SYMMETRY` is used for the CHARMM potential. If `NOSYMMETRY` is selected, no reorderings are done. `NOSYMMETRY` is used for the AMBER potential. `SYMMETRY` is the default.

The `HBOND` default options control which distance is used in the hydrogen bond energy. If `D-A` is specified, the distance is calculated between the heavy atom donor and the acceptor, and the angular term is included. In addition, the parameterization is done based on the heavy atom donor and acceptor. This is the CHARMM form. If `H-A` is specified, the distance is calculated between the hydrogen and the acceptor, and no angular term is included. The parameterization is done based on the hydrogen and the acceptor. The default is `D-A`

The `NBOND` default options control scaling for 1-4 interactions and the inclusion of van der Waals energies for hydrogen bond pairs. 1-4 interactions are non-bonded interactions of atoms connected by three bonds (see Section 4.2 [Nbxmod], page 45, for more information). The `VDW14` keyword sets the scale factor for the van der Waals energy of 1-4 interactions. The `EL14` keyword sets the scale factor for 1-4 electrostatic interactions. The default is 1.0 for both of these scale factors. In the AMBER potential, they are set to 0.5. The `HBINCLUDE` keyword specifies that van der Waals interactions will be calculated for atoms involved in hydrogen bonds. This is the default. The `HBEXLCUDE` keyword specifies that van der Waals interactions will be turned off for all possible atom pairs specified as possible hydrogen bonds. This is the default for the AMBER potential. *Warning:* you must ensure that the hydrogen bond distance cutoff is positive when this option is in use. Otherwise, it is possible to generate infinite energies if a charged hydrogen and its acceptor get too close together.

`PRINT` command

Syntax

```
PRINT [ON ]
      [OFF]
```

Function

The `PRINT` command turns on the echoing of commands in the parameter file and the display of all non-bonded parameters. It is useful for debugging. It is off by default.

`END` command

Syntax

```
END
```

Function

The `END` statement terminates the parameter file.

3.1.4.2 Old Parameter File Format

In the format for text parameter file, the data is divided into sections beginning with a keyword line and followed by data lines. The sections may be arranged in any order, and may divided up as well. Just prefix each set of data with the appropriate keywords. The format for each data section follow along with the necessary keywords. Please look at the parameter input files in the '`CGDATA`' directory for examples.

```
Keyword   Format

BOND  -   atom atom force_constant distance
          (2(A4,1X),2F10.0)

THETA -   atom atom atom force_constant theta_min
          (3(A4,1X),2F10.0)

PHI   -   atom atom atom atom force_constant periodicity phi_max
          (4(A4,1X),3F10.0)

IMPHI -   atom atom atom atom force_constant i_phi_min
          (4(A4,1X),2F10.0)

NBOND -   atom polarizability n_effective_electrons vdW_radius
          (A4,1X,3F10.0)

HBOND -   atom atom well_depth distance
          (2(A4,1X),2F10.0)
```

Note that the data lines are NOT free field. However, you can add comments using the exclamation point, see Section 2.1 [Run Control], page 5. Sections end with the occurrence of another keyword or a line with the word END in columns 1-3, the latter terminating parameter reading. Blank lines are allowed in all the sections.

Some errors in the input file will result in warning messages but not termination of the run.

CONGEN will check for duplicate parameters. If all the corresponding values for a duplicate parameter are the same, then only a warning message is issued. Otherwise, an error message will be issued.

Any errors detected in the reading of the formatted parameter file will result in termination of the run, unless `NOFAIL` is specified on the `READ` command.

`phi_max` is either 0.0 or 180.0 for dihedrals with the minimum staggered or eclipsed respectively.

If successive torsion angle or improper torsion angle parameters are specified with all four atoms and have the same atoms, this is a flag that the energy is to be computed as a sum of these multiple terms. For this special processing to be done, the PSF (or topology file used to generate the PSF) must have successively equal torsion or improper torsion angles which correspond to the parameters. In order to use this option, you must specify `NOFAIL` on the command line.

NBOND parameters must be present for all of the atom types. The program attempts to check this when reading either card image or binary parameter files.

3.1.5 The Format of a Residue Topology File

Here is a description of what is in residue topology files (as they are stored in text files). You may use this format if you specify the `CARD` option in the `READ` command. The format of binary files depends on the current implementation of the RTF data structure. See the file '`RTF.FCM`' in the CONGEN source directory for more details. These files are read by `RTFRDR`, a subroutine in `RTFIO` which should be be consulted for formats and the final word on what is actually done with these files.

The purpose of residue topology files is to store the information for generating a representation of macromolecule from its sequence. For each residue, CONGEN requires a description of all bonds, bond angles, dihedral angles, improper torsion angles, partial charges, chemical types, and hydrogen bond donors and acceptors. By linking residues in the sequence together, segments in the molecule are constructed.

The linkage between successive residues is determined when the segments are generated (see Section 4.1 [Generate Command], page 43). It is specified in the residue by using special prefixes on the atom names which refer to residues either ahead of or behind the current residue. In the case of cyclic segments, the program will wrap references around the cycle.

The residue topology files begin with '`rtop`'. There are two forms, binary module ('`.mod`') and card format (usually '`.inp`'). The card format files are used only for creating binary modules and therefore are structured as input files for CONGEN, beginning with a run title and the command `READ RTF CARD`, followed by the actual topology file.

The first section of the topology files is a title section in the usual format of up to ten lines delimited by a line containing only a '`*`' in column 1.

The next line is a set of up to 20 numbers of which the first number gives the topology file format version number. This number be set to 200 for CONGEN to read the remaining file correcting in free field format. If some other number is present or the number is missing, the program will attempt to read the topology file in the current format.

The remaining information is read in free field format as commands to define the RTF. The ordering of the commands is important in that some information is needed to define others (i.e. the atoms of a residue should be defined before the bonds between them). The recommended structure of this file is:

```
     Initial setup:
             TYPE declaration
             MASS specification for each atom type (also hydrogen
                  bond donor and acceptor classifications)
             DECLarations of out of residue definitions
             ORDEr specification for atom order.
             SET  command for charge patching
     For each residue:
             RESIdue name and total charge specification
             ATTRibute option to specify
             ATOM definitions within this residue
             BOND specifications
             ANGLe specifications
             DIHEdral angle specifications
             IMPRoper dihedral angle specifications
             DONOr specifications
             ACCEptor specifications
             BUILD information
             GROUPing definitions
             GENErate options
             COPY option to copy information from other residues
     Closing:
             END statement
     Display control:
             PRINT option
```

The format above is not rigid.

There exists the facility to automatically generate the bond angles, torsion angles, hydrogen bond donors and acceptors, and the BUILD information. It is also possible to delete terms that are generated automatically, and therefore, it is possible to correct any deficiencies in the automatic schemes.

3.1.5.1 Linkage Atom Naming

It is important to understand how to make references to adjacent residues when constructing a topology file. The following table lists the possible prefixes which may used. Note that the actual atoms referenced are not determined until the generation of segments, see Section 4.1 [Generate Command], page 43.

`+`	The next residue in the sequence.
`-`	The previous residue in the sequence.
`#`	Two residues forward in the sequence.
`=`	Two residues backward in the sequence.
`+n`	`n` residues forward in the sequence. Note that '`+`' sign is mandatory.
`-n`	`n` residue backward in the sequence.

No reference will be made beyond the end of a segment. In the case of cyclic peptide, CONGEN will wrap around the sequence in the appropriate direction.

3.1.5.2 RTF Type Command

Syntax

```
                { PROT }
                { HPRO }
     TYPE       { ALLH }
                { DNA  }
                { UNKN }
```

Function

This option sets the type of the residue topology file. When a segment is generated, the program will check to see that the sequence type matches the RTF type, provided that a sequence type is specified.

3.1.5.3 RTF Mass Command

Syntax

```
     MASS int word real [ACCEptor]
                        [DONOr   ]
```

The MASS command specifies the chemical types of atoms, their names, their masses, and optionally whether they are hydrogen bond donors or acceptors. This command is one of the most important in the topology file because is specifies all the permissible atoms in any system.

The `int` is the numerical chemical type code as used in the parameter file, see Section 3.1.4 [Param Files], page 21. Its value may not exceed the parameter `MAXATC`, which is currently 100. The `word` is the chemical type name, and this symbol is used in the parameter file. It can also be referenced when analysis tables are built, see Section 17.1.1 [Build Syntax], page 159. The `real` number specifies the mass of each atom type in Atomic Mass Units. Finally, the optional keywords, `ACCEPTOR` and `DONOR`, indicate when the atom can participate in a hydrogen bond as an acceptor or donor, respectively. These finally keywords are used only by the RTF `GENERATE` command, see Section 3.1.5.19 [RTF Generate Command], page 35.

3.1.5.4 RTF Declare Command

Syntax

```
     DECLARE word
```

Function

When a formatted RTF file is read, CONGEN will check to see that all components of a residue refer to atoms within that residue, and will issue an informational message if they are not. However, since all polymeric structures will have linkages between residues, there will be atoms which refer outside of each residue.

The `DECLARE` command informs the program that atoms whose name is `word` is a linkage atom, and that CONGEN should not issue a message for such messages. Aside from these messages, the `DECLARE` command is optional.

3.1.5.5 RTF Set Command

Syntax

```
     SET word real
```

Function

When residues in the topology file are patched when a segment is made from them, some atomic charges must be adjusted. Currently, the program calculates the correct charges for the explicit hydrogen and extended atom topology files, see Appendix A [Residue Topology Files], page 289. However, with other topology files, it does not.

The SET command provides a mechanism for assigning these charges in the topology file. The user specifies a variable name as the first operand in the command, and a charge value or adjustment in the second. These variable name, charge value pairs are stored with the topology file. If no variable name is available, then a default value is used which will give the correct values for the explicit hydrogen topology files.

The following table gives the currently used variables, default values, and their meaning.

Variable	Default	Meaning
CG_C3_PRIME	0.084	Charge increment for the 3' carbon in DNA.
CG_C5_PRIME	0.092	Charge increment for the 5' carbon in DNA.
CG_DISU_CB	0.022	Charge for the beta carbon in a disulphide linkage.
CG_DISU_SG	-0.032	Charge for the gamma sulfur in a disulphide linkage.
CG_FIRSTCA	0.020	Charge increment for the amino terminal alpha carbon.
CG_FIRSTN	0.710	Charge increment for the amino terminal nitrogen in a protein.
CG_LASTC	0.030	Charge increment for the carboxy terminal carbonyl carbon.
CG_LASTO	-0.200	Charge increment for the carboxy terminal carbonyl oxygen.
CG_O3_PRIME	0.163	Charge increment for the 3' oxygen in DNA.
CG_O5_PRIME	0.147	Charge increment for the 5' oxygen in DNA.
CG_PRO_FIRSTN	0.085	Charge of the amino nitrogen in an amino-terminal proline. NOTE: the presence of this variables signifies that prolines get special treatment. If you omit it, then none of the CG_PRO variables will have any effect.
CG_PRO_FIRSTCD	0.079	Charge of the delta carbon in an amino-terminal proline.
CG_PRO_FIRSTCA	0.095	Charge of the alpha carbon in an amino-terminal proline.
CG_PRO_IHT1	0.225	Charge of the first amino terminal peptide hydrogen in an amino-terminal proline.
CG_PRO_IHT2	0.225	Charge of the second amino terminal peptide hydrogen in an amino-terminal proline.

This particular mechanism used for charged should be viewed a temporary measure. There are plans to replace it with a more robust patching scheme.

3.1.5.6 RTF Order Command

Syntax

```
ORDER repeat(word)
```

Function

Certain operations within CONGEN expect the atoms in a residue to appear in a given order. Examples of such commands are the `RANGE` options in an atom selection, see Section 11.7 [Atom Selection], page 108, or the reading of binary coordinate files, see Section 3.1.3 [Read Coordinates], page 18. The `ORDER` command permits you to specify the atom order for all current and succeeding residues in the topology file.

Each word in the `ORDER` command is interpreted as a wild card atom selection token, see Section 11.7 [Atom Selection], page 108. When each residue is completed, all the atoms in the residue are matched against the words in the `ORDER` command. An exact match takes precedence over a wildcard match, and the last match takes precedence over earlier matches. The order of the atoms in the residue is then rearranged based on the matches into these words. If a set of atoms match the same word in the `ORDER` command, then these will be ordered according to their original order in the topology file.

As an example, the command

```
ORDER N H CA * C O
```

will put the amino nitrogen and hydrogen and the alpha carbon of an amino acid first, the sidechain atoms in the middle, and the carbonyl carbon and oxygen at the end of each residue.

The `ORDER` command takes effect starting with the next residue completed. Thus, at the beginning of a file, it affects the entire file. If specified after a `RESIDUE` command, it will affect that residue.

3.1.5.7 RTF Residue Command

Syntax

```
RESIDUE word [real] [ADJUST]
```

Function

The `RESIDUE` is used to start a new residue. If another residue has already been specified, then it is completed. The `word` in the command specifies the residue name, and the optional real number specifies the total electrostatic charge of the residue. If no total charge is specified, then 0.0 is assumed. If `ADJUST` is specified, then the charges of all the atoms will be adjusted by the following formula:

$$\frac{\text{total} - \sum_{\text{atoms}} q_i}{N}$$

where N is number of atoms in the residue.

3.1.5.8 RTF Copy Command

Syntax

```
COPY word [INVERT]
```

Function

The `COPY` command is used to copy the information stored for a previous residue into the current residue. The name of the previous residue is given by the `word` following the `COPY` verb. It can only be used at the beginning of a residue specification. The `INVERT` option cause the program to invert all the torsion angles specified in the BILD commands, see Section 3.1.5.17 [RTF Build Command], page 34.

The `COPY` command is most useful when two residues are nearly identical, and you do not wish to keep multiple copies of identical information. For example, it is used in the specification of D amino acids.

3.1.5.9 RTF Attribute Command

Syntax

```
ATTRIBUTE [DELETE] repeat(word)
```

Function

The `ATTRIBUTE` command is used to specify residue attributes. Each `word` following the `ATTRIBUTE` command is added to the residue attribute list, unless the `DELETE` option is specified. In that case, the words are deleted from the residue attribute list.

At present, the only residue attribute that has any significance is the `D` attribute, which informs the conformational search code, see Chapter 12 [Conformational Search], page 111, that the residue is a D amino acid, and should be processed accordingly.

3.1.5.10 RTF Atom Command

Syntax

```
ATOM iupac word real repeat(iupac)
```

Function

The `ATOM` command specifies the atoms in a residue. The first word in the `ATOM` specifies the IUPAC name of the atom. The second word in the atom specification gives the chemical type code as specified in the second operand of the `MASS` command, see Section 3.1.5.3 [RTF Mass Command], page 29. The third operand specifies the partial charge of the atom as a real number.

Finally, the remaining words specify the names of atoms which are to be excluded from non-bonded interactions. Such exclusions are made because they are directly bonded or separated by only two covalent bonds. Note that 1-2 and 1-3 non-bonded exclusions can be constructed automatically at segment generation time, see Section 4.2 [Nbxmod], page 45.

3.1.5.11 RTF Bond Command

Syntax

```
BOND [DELETE] repeat(iupac iupac)
```

Function

The `BOND` command is used for specifying the bonds in a residue. Each pair of `iupac` atom names specifies a bond. Atoms outside the current residue can be specified using the scheme described in Section 3.1.5.1 [Linkage Atom Naming], page 28.

The `DELETE` option causes CONGEN to delete the named bonds from the current residue. This option is useful when the `COPY` command is used.

3.1.5.12 RTF Angle Command

Syntax

```
{ ANGLE } [DELETE] repeat(iupac iupac iupac)
{ THETA }
```

Function

The `ANGLE` command, and its synonym, `THETA`, are used to specify bond angles in a residue. Each triple of `iupac` names specifies a bond angle. The RTF Generate Command, see Section 3.1.5.19 [RTF Generate Command], page 35, can be used to generate the bond angles within a residue automatically, but bond angles involving atoms outside the current residue must always be specified "by hand".

The `DELETE` option causes CONGEN to delete the named angles from the current residue. This option is useful when the `COPY` command or the automatic generation of angles is used.

3.1.5.13 RTF Torsion Command

Syntax

```
{ TORSION  } [DELETE] repeat(iupac iupac iupac iupac)
{ DIHEDRAL }
```

Function

The `TORSION` command, and its synonym, `DIHEDRAL`, are used to specify torsion angles in a residue. Each quadruple of `iupac` names specifies a torsion angle with the middle pair of atoms defining the bond being rotated (and used to chose parameters). When the parameter file contains dihedral angles specified by all four atoms, every dihedral angle is first checked to see if it matches any of this type. If so, then the four atom parameters values are used. If a particular four atom dihedral is specified twice in adjacent positions, then it is assumed that the corresponding parameter file specifies two separate parameter values for this four atom dihedral, and both will be used.

The RTF Generate Command, see Section 3.1.5.19 [RTF Generate Command], page 35, can be used to generate the torsion angles within a residue automatically, but torsion angles involving atoms outside the current residue must always be specified "by hand".

The `DELETE` option causes CONGEN to delete the named torsion angles from the current residue. This option is useful when the `COPY` command or the automatic generation of torsion angles is used.

3.1.5.14 RTF Improper Command

Syntax

```
{ IMPROPER } [DELETE] repeat(iupac iupac iupac iupac)
{  IMPHI   }
```

Function

The `IMPROPER` command, and its synonym, `IMPMI`, are used to specify improper dihedrals in a residue. Each quadruple of `iupac` names specifies an improper dihedral with the first atom is the atom being kept planar or chiral. As with the proper dihedral, four atom specifications may be used and when an improper dihedral is repeated, multiple parameter values will be sought.

There does not exist any automatic mechanism for constructing improper dihedrals. It is entirely a function of the parameters used to build the residues.

The `DELETE` option causes CONGEN to delete the named improper dihedral angles from the current residue. This option is useful when the `COPY` command is used.

3.1.5.15 RTF Donor Command

Syntax

```
DONOR [DELETE] [ iupac ] iupac [ iupac iupac ]
```

Function

The `DONOR` command specifies hydrogen bond donors. The only required operand is the heavy atom donor (such as an amide nitrogen). If a hydrogen is present, then it must be specified before the heavy atom donor. In the case of proteins, two antecedent atoms must also be specified. This antecedents are used by the hydrogen construction routine, `HBUILD`, to construct the hydrogens automatically.[1]

Note that if the first atom is outside of the current residue, CONGEN will assume it is a hydrogen. Also note that only one hydrogen donor can be specified at a time. This is different than the previous commands.

Hydrogen bond donors can be automatically constructed, but only within the current residue, and without the construction of antecedents. The `DELETE` option may be used to remove hydrogen bond donors introduced by either the automatic generation scheme, or by the `COPY` command.

3.1.5.16 RTF Acceptor Command

Syntax

```
ACCEPTOR [DELETE] iupac [iupac [iupac] ]
```

Function

The `ACCEPTOR` command specifies the acceptors of hydrogen bonds. The first IUPAC name specifies the acceptor. Antecedents to this acceptor are stored, but they are not used anywhere.

The `DELETE` option may be used to remove an automatically generated acceptor or one copied using the `COPY` command.

3.1.5.17 RTF Build Command

Syntax

```
{ BILD  }
{ BUILD } [DELETE] iupac iupac iupac iupac real real real real real
```

Function

The `BILD` command is used to specify the rules for constructing atoms. The first four operands are atom names, and they specify a set of four atoms which are linked together, which are referred to as I, J, K, and L. If the third atom is prefixed by an asterisk, then the rule is an improper torsion, where atom K is in the middle, and atoms I, J, and L are bound to it. Otherwise, the rule is a proper torsion, and the four atoms are bound together in a chain.

In the case of a proper torsion, the five real numbers specify the bond length between I and J, the bond angle between I, J and K, the torsion angle between I, J, K, and L, the bond angle between J, K, and L, and the bond length between K and L. In the case of an improper torsion, the five real numbers specify the bond length between I and K, the bond angle between I, K, and J, the improper torsion between I, J, K, and L, and bond angle between J, K and L, and the bond length between K and L. Units for bonds are in Angstroms; units for angles are in degrees.

[1] They can also be constructed by the internal coordinate commands, see Chapter 14 [Internal Coordinates], page 147, but `HBUILD` is more intelligent than the internal coordinate commands.

In either case, the rules can be used to construct the positions of atoms I or L depending the positions of the remaining three atoms. If either bond length or either bond angle is specified as 0.0, then the program will search the parameter files for the values to use. The torsion angles are used as specified, unless they are edited by an internal coordinate editing command, see Chapter 14 [Internal Coordinates], page 147.

The `DELETE` option may be used to remove a `BILD` rule either automatically generated, or copied using a `COPY` command.

3.1.5.18 RTF Group Command

Syntax

```
GROUP name first-atom-iupac last-atom-iupac
```

Function

The `GROUP` command was incorporated to provide electrostatic groups. It is not used.

3.1.5.19 RTF Generate Command

Syntax

```
                  { ANGLES               }
                  { THETAS               }
                  { { TORSIONS } [ ALL ] }
GENERATE repeat(  { { PHIS     } [ ONE ] }
                  { DONORS               }
                  { ACCEPTORS            }
                  { BILDS                }
                  { BUILDS               }
                  { ALL                  }
```

Function

The `GENERATE` command may be used to automatically generate parts of the topology file. Angles, torsions, donors, acceptors, and IC constructors may be generated either singly or in any combination. `ALL` specifies that all of these entities be constructed. Automatic generation is only performed with the atoms defined within a residue. Atoms involved with linkages to other atoms are not used in the automatic generation process, and must be generated "by hand".

The bond angle construction works by examining the current list of bonds for the current residue, and generating angles for all pairs of bonds. The torsion angle construction can work in either of two ways. If the `ALL` suboption is specified, then for each bond, a torsion angle is constructed for all combinations of atoms attached to this torsion. If the `ONE` suboption is specified, the program will first look for a pair of heavy atoms attached to the central bond, and failing that, it will repeat the search looking at all atoms including hydrogens. The donors and acceptors generation depend on the use specifying which atom types are donors or acceptors in the MASS command above.

The generation of IC constructors is the most difficult generation task, and it is done primarily to provide a set of constructors which can be edited for better results. The program will first find three atoms connected together. Next, it will look at the central atom and see if an adjacent atom needs to be constructed. If so, it will generate a improper torsion constructor. It will then recursively try the new atom in conjunction with the previous two. Once these constructions complete, the program will attempt to add atoms on each end of the original three atoms, and will recurse on any successful constructions.

Because not all of these automatic generation commands work perfectly, there exists two mechanisms to edit the results. First, the RTF may be written out in free field format using a `WRITE RTF CARD` command and the result can be edited. Second, the `DELETE` options in other commands may be used to delete particular entries after they are generated. Also, additional entries for a residue may be made after the `GENERATE` command is given.

Note that non-bonded exclusions are generated automatically by default. See Section 4.2 [Nbxmod], page 45, for more information.

3.1.5.20 RTF Print Command

Syntax

```
PRINT { ON  }
      { OFF }
```

Function

The `PRINT` command may be used to control the display of lines as they are read by the RTF reader. The initial setting for printing is controlled by the `READ` command itself. If `PRINT` is specified, then printing will initially be enabled; otherwise, the commands will not be echoed. `PRINT ON` turns on echoing of RTF specifications; `PRINT OFF` turns them off. This command is useful for debugging an addition to a previously tested topology file.

3.1.5.21 An Example RTF File

This is a small RTF example.

```
*  title for documentation example
*
  200    1
TYPE HPRO
MASS     1 H      1.00800
MASS    11 C     12.01100
MASS    12 CH1E  13.01900
MASS    13 CH2E  14.02700
MASS    14 CH3E  15.03500
MASS    31 N     14.00670
MASS    38 NH1   14.00670
MASS    51 O     15.99940
MASS    56 OH2   15.99940

DECL -C
DECL -O
DECL +N
DECL +H
DECL +CA

RESI ALA     0.00000
ATOM N    NH1    -0.20000     H    CA   CB   C
ATOM H    H       0.12000     CA
ATOM CA   CH1E    0.07500     CB   C    O    +N
ATOM CB   CH3E    0.02000     C
ATOM C    C       0.35000     O    +N   +H   +CA
ATOM O    O      -0.36500     +N
BOND N    CA         CA   C          C    +N         C    O          N    H
BOND CA   CB
THET -C   N    CA             N    CA   C              CA   C    +N
THET CA   C    O              O    C    +N             -C   N    H
THET H    N    CA             N    CA   CB             C    CA   CB
DIHE -C   N    CA   C         N    CA   C    +N        CA   C    +N   +CA
IMPH N    -C   CA   H         C    CA   +N   O         CA   N    C    CB
DONO H    N    -C   -O
ACCE O
BILD -C   CA   *N   H      0.0000    0.00  180.00    0.00   0.0000
BILD -C   N    CA   C      0.0000    0.00  180.00    0.00   0.0000
BILD N    CA   C    +N     0.0000    0.00  180.00    0.00   0.0000
BILD +N   CA   *C   O      0.0000    0.00  180.00    0.00   0.0000
BILD CA   C    +N   +CA    0.0000    0.00  180.00    0.00   0.0000
BILD N    C    *CA  CB     0.0000    0.00  120.00    0.00   0.0000

RESI OH2     0.00000
ATOM OH2  OH2    -0.40000     H1   H2
ATOM H1   H        0.20000     H2
ATOM H2   H        0.20000
BOND OH2  H1         OH2  H2
THET H1   OH2  H2
DONO H1   OH2
DONO H2   OH2
ACCE OH2

END
```

3.1.6 Reading Other Files

The parameter files (PARAMETER) and internal coordinate files (IC) can be read as card images or binary files. Specifying CARD signifies card image input; specifying FILE signifies binary file input. Please note that topology file must be read in before the parameters can be read. More information about the Internal Coordinate files can be found in their I/O routines, READIC and WRITIC, in the file 'intcor.flx'.

Hydrogen bond (HBOND), protein structure files (PSF) files, harmonic constraints (CONSTRAINT), and non bonded lists (NBOND) can only be read as binary files.

The Image file (IMAGES) containing transformation information can only be read in card image format. This is not to be confused with the Images data structure (see Chapter 9 [Images], page 87).

3.2 WRITE — Writes Data Structures to External Files

3.2.1 Syntax

```
WRITe { { PSF        } [FILE]                 } UNIT unit-number
      { { HBONd      }                        }
      { { PARAmeter  }                        }
      { { NBONd      }                        }
      { { CONStraint }                        }
      {                                       }
      { { RTF        } [FILE]                 }
      {                [CARD]                 }
      {                                       }
      { { IC         } [CARD]                 }
      {                [FILE]                 }
      {                                       }
      { { COORdinate } [CARD      ] coor-spec }
      {                [FILE      ]           }
      {                [KONNert   ]           }
      {                [BROOkhaven]           }
      {                [BRKHvn    ]           }
      {                                       }
      { IMAGes         [CARD]                 }

title

coor-spec:== { [MAIN] }  [ OFFS int ] [ HNUSe ] [ WRAP ] atom-selection
             {  COMP  }               [NONHUSe] [NOWRAP]
             {  DIFF  }
```

3.2.2 Function

The primary purpose of this command to save some of CONGEN's data structures on file in unformatted form. In addition, the coordinate and internal coordinate data structures can be written in formatted form so that they be edited independent of CONGEN using GNU Emacs or a similar text editor. The option, `FILE`, specifies that a file is to be written in unformatted form (binary). The option, `CARD`, specifies that a file is to written in formatted form. For the coordinate and internal coordinate file, `CARD` is the default.

A set of title lines must follow the `WRITE` command. This title will be written at the start of the file and serves to document the file. For your protection, one should always make good use of this title, as it may be the only documentation for the file.

The `UNIT` keyword specifies what Fortran unit the output should be written to. It cannot be omitted.

Additional options are available for writing coordinates in text format. The option, `KONN`, will write the coordinates in Konnert-Hendrickson format. The synonymous options, `BROOKHAVEN` and `BRKHVN`, will write the coordinates in Brookhaven Protein Data Bank format. The option pair, `HNUSE` and `NOHNUSE`, control whether the hydrogen on the peptide nitrogen is written with a name of '`HN`' or '`H`. The default is `NOHNUSE` which uses '`H`'. The option pair, `WRAP` and `NOWRAP`, controls whether hydrogens which have a terminating digit are written with the terminating digit first. For example, the arginine atom, `HH12`, is written as '`2HH1`' if `WRAP` is enabled, and written as '`HH12`' if `NOWRAP` is enabled. The default is `NOWRAP`.

3.3 `PRINT` — Writes Information to Output File (Unit 6)

3.3.1 Syntax

```
PRINt { PSF                   }
      { RTF                   }
      { CONStraint            }
      { PARAmeter             }
      { RESIdue               }
      { COORdinate coor-spec }
      { IC                    }
      { HBONd       [ ANAL ] }
      { IMAGes                }
      { NMR nmr-options       }
      { FROM unit-number      }

coor-spec::= { [MAIN] } [ OFFS int ] atom-selection
             {  COMP  }
             {  DIFF  }

nmr-options ::= [ALL ]
                [NONE]

                [[NO]NOE] [[NO]JCOUPLING] [[NO]TABLE] [[NO]SORT] [TOP int]

                [INDIVIDUAL int] [[NO]ENERGY] [[NO]FORCE] [[NO]VIOLATION]

                [MIN real] [MAX real] [ROWS int] [COLUMNS int]
```

Syntactic ordering: All commands must be typed in the order shown except for the `nmr-options` which can be in any order after the `NMR` option.

3.3.2 Function

This command is used to list information contained in data structures used by the program or to list a formatted file. The information must already have been created through use of a `READ`, `GENERATE`, `HBONDS`, etc., command. The printable output is sent to unit 6.

If the `FROM` option is used, the `PRINT` command will print a formatted file onto unit 6. The file will be rewound after printing so it may be used again.

For hydrogen bonds, `ANAL` gives a geometrical and energy analysis of the hydrogen bonds. Representing the hydrogen bond as A2-A1-X-H....Y-, the distances X-Y, H-Y, the angle (180 - <(X-H-Y)), the dihedral angle A2-A1-X-H and the hydrogen bond energy contribution are listed.

The `PRINT NMR` command invokes an analysis of the NMR constraints. There a number of components in the analysis, and The various `nmr-options` control which components appear. If no options are specified then all components will be displayed. However, if there are any options, then only those specified by the user will be displayed. The keyword, `ALL`, may be used to turn on the display of all components, and then the user may modify the display with additional operands.

The keywords are interpreted as follows:

`ALL`	This enables the display of all analysis components.
`NONE`	This disables the display of all analysis components.

`NOE`
`NONOE` The `NOE` option enables the display of all components which pertain to NOE's. `NONOE` disables this display.

`JCOUPLING`
`NOJCOUPLING`
The `JCOUPLING` option enables the display of all components which pertain to J coupling constraints. `NOJCOUPLING` disables this display.

`TABLE`
`NOTABLE` This enables the display of a table of either NOE's or J coupling constraints possibly sorted by decreasing energy. `NOTABLE` disables this display.

`SORT`
`NOSORT` The `SORT` keyword controls whether the above tables are sorted.

`TOP` The `TOP` option specifies how many entries are to be printed in the tables of constraints. The default is 20.

`INDIVIDUAL`
In the analysis of NOE constraints, CONGEN can print many details about the NOE's. The `INDIVIDUAL` option controls approximately how many are output. The program converts the value of this option into a frequency of output starting with the first NOE. In addition, a brief summary of NOE satisfaction is listed. If this option is specified as zero or negative, then no individual NOE output or summary is made.

`ENERGY`
`NOENERGY` The `ENERGY` and `NOENERGY` options controls the display of a histogram of non-zero energies for the NOE or J coupling constraints. The form of the histogram can be controlled by other options below.

`FORCE`
`NOFORCE` The `FORCE` and `NOFORCE` options controls the display of a histogram of non-zero forces for the NOE constraints. No histogram for J coupling forces is currently available. The form of the histogram can be controlled by other options below.

`VIOLATION`
`NOVIOLATION`
The `ENERGY` and `NOENERGY` options controls the display of a histogram of violations for the NOE or J coupling constraints. The form of the histogram can be controlled by other options below.

`MIN` The `MIN` option specifies the minimum in the data used by any histogram. If omitted, then the histograms will use the minimum of whatever data is being plotted.

`MAX` The `MAX` option specifies the maximum in the data used by any histogram. If omitted, then the histograms will use the maximum of whatever data is being plotted.

`ROWS` The `ROWS` option specifies the number of rows (buckets) in the histogram. The default is 18.

`COLUMNS` The `COLUMNS` option specifies the number of columns in the histogram. The default is 80 characters.

3.4 Open File Command — `OPEN`

3.4.1 Syntax

```
OPEN UNIT integer NAME filename [WRITE] [UNFORMatted]
                                [READ]  [FILE]
                                        [FORMatted]
                                        [CARD]
```

3.4.2 Function

The `OPEN` command is used to open logical units to specific files specified from the input file rather than logical name assignments made prior to the run. This is useful in setting up test cases and interactive use of the program. `OPEN` can be used to redirect the output that appears on unit 6 to different files by opening unit 6 in the middle of a run. However, it may not be possible to restore unit 6 back on some machines, so be careful with this.

3.5 Close File Command — `CLOSE`

3.5.1 Syntax

```
CLOSe  UNIT integer  [SAVE  ]
                     [KEEP  ]
                     [DELETE]
                     [PRINT ]
```

3.5.2 Function

The `CLOSE` command closes a logical unit. This frees the associated file and logical unit so that they can be used for other purposes. The default disposition of the file is `SAVE` or `KEEP`.

3.6 `REWIND` Command

3.6.1 Syntax

```
REWInd  UNIT integer
```

3.6.2 Function

The `REWIND` command causes the requested logical unit to be rewound. When used with the `STREAM` command, a particular sequence can be used more than once.

3.7 `STREAM` Command

3.7.1 Syntax

```
STREam  UNIT integer
```

3.7.2 Function

The `STREAM` command allows the input of command sequence to be shifted to another file. This is useful when parts of an input file are to be used many times or used by many different calculations. The only input value is the unit number to transfer to.

3.8 `RETURN` (Restore Previous Command Stream) Command

3.8.1 Syntax

```
RETUrn
```

3.8.2 Function

The `RETURN` command causes the input of command sequence to return to the stream that called the current stream. Streams may be nested to up to 20 calls. There are no parameters for this command.

4 Structure Generation and Manipulation (The PSF)

The commands described in this node are used to construct and manipulate the PSF, the central data structure in CONGEN. The PSF is comprised of lists giving every bond, bond angle, torsion angle, and improper torsion angle as well as information needed to generate the hydrogen bonds and the non-bonded list. It is essential for the calculation of the energy of the system. A separate data structure deals with symmetric images of the atoms (see Chapter 9 [Images], page 87).

There exists one other command for manipulating the PSF, the `SPLICE` command, see Section 13.1 [Splice Command], page 143. The `SPLICE` command can change the sequence of PSF and shuffle the coordinates so that the `CONGEN` command, see Chapter 12 [Conformational Search], page 111, which can be used to find conformations for the new amino acids.

There is an order with which commands to generate and manipulate the PSF must be given. First, segments in the PSF must be generated one at a time. Prior to generating any segments, one must first have read a residue topology file, see Appendix A [Residue Topology Files], page 289. To generate one segment, one must first read in a sequence using the `READ` command, see Section 3.1.2 [Read Sequence], page 16. Then, the `GENERATE` command must be given. RTF's may be changed as needed; however, the `SPLICE` command will only work correctly when only one RTF is used to build the entire structure.

Once a segment is generated, it may be manipulated. It may be edited using the `EDIT` command. Cystine bridges may be added using the `DISULFIDE` command. A histidine heme crosslink may be added using the `PATCH HEME` command.

4.1 The Generate Command - Construct a Segment of the PSF

4.1.1 Syntax

```
GENErate [segid] [NBXMod int] [rtf-type] [CYCLic]

         [[NO]ANGLes] [NOTORSions      ] [[NO]DONOrs] [[NO]ACCEptors]
                      [TORSions {ALL}]
                      [         {ONE}]

              { PROT }
              { HPRO }
              { ALLH }
rtf-type ::=  { DNA  }
              { A94N }
              { A94P }
              { AM94 }
```

4.1.2 Function

Using the sequence of residues specified in the last `READ SEQUENCE` command and the information stored in the residue topology file, this command generates the next segment in the PSF. Each segment contains a list of all the bonds, angles, dihedral angles, and improper torsions needed to calculate the energy. It also assigns charges to all the atoms, sets up the nonbonded exclusions list, and specifies hydrogen bond donors and acceptors. If a special type of segment has been specified in the `READ SEQUENCE` command or by the `rtf-type` option, modifications for structural features not contained in the residue topology file, for example terminal group modifications and proline modifications, are made automatically. The `CYCLIC` option controls whether a cyclic structure is built. Cyclic structures are made by omitting any terminal residues, and wrapping references to atoms beyond each end of the segment back to the other end.

The processing of terminal groups varies depending on the `rtf-type` setting. If a CONGEN topology file is read; `rtf-type` equals `PROT`, `HPRO`, `ALLH`, or `DNA`; then extra residues, like `NTER` and `CTER`, are added to the sequence. This has the undesirable side-effect of adding extra residues into your sequence, and confusing the residue numbering.

If the AMBER 94 potential is used, a different scheme is used. Here, the topology file, Section A.6.5 [AMBER94RTF], page 297, contains special terminal residues, which have different atoms and charges. The `GENERATE` command will translate the terminal residues to the names in the topology file, generate the segment, and then translate the names back. The following example table illustrates the naming conventions used for alanine.

ALA	Regular alanine.
NALA	N terminal alanine.
CALA	C terminal alanine.
DALA	D alanine.
MALA	N terminal D alanine.
BALA	C terminal D alanine.

The `A94P` keyword specifies that an AMBER 94 protein sequence is to be generated, and the `A94N` keyword specifies a nucleic acid sequence is to be generated. As a special case, leucinol (LEOH) is handled correctly.

The `GENERATE` command is capable of automatically generating some of the information needed to compute the energies from other sources within the PSF. In the case of the AMBER potential, CONGEN sets these options on by default. In addition, the value of these switches is saved in the PSF (in the `NICTOT` array), so that they can be used by the `SPLICE` command, see Section 13.1 [Splice Command], page 143. The automatic generation options have the following interpretations:

`ANGLES`
`NOANGLES` Specifies that bond angles are to be generated from the bond list. All bonds joined by a common atom will be added to the angle list, and all duplicates removed. The keyword, `NOANGLES`, turns this option off.

`TORSIONS`
`NOTORSIONS`
Specifies that torsion angles are to be generated from the bond list. This process begins by CONGEN looping over all bonds in the system. It will then examine all atoms bound to the two atoms in the selected bond, and If the keyword `ONE` is specified, the program will add a torsion for the first pair of bound atoms which are both non-hydrogen. If no pair of heavy atoms can be found, then the program will select the first pair of bound atoms. If the keyword `ALL` is specified, then all sets of four atoms connected by three bonds in tandem will be added as torsions. Note that this code detects three membered rings and will ignore them. In addition, you *must* specify either `ONE` or `ALL` if you specify `TORSIONS`. The keyword, `NOTORSIONS`, turns off this option.

`DONORS`

`ACCEPTORS`
Specifies that hydrogen bond donors and acceptors be added automatically. This depends critically on the atom type specifications in the residue topology file, see

Section 3.1.5.3 [RTF Mass Command], page 29. The keywords, `NODONORS` and `NOACCEPTORS`, turn off these options, respectively.

`NBXMOD` The `NBXMOD` option controls the automatic generation of non-bonded exclusions, see Section 4.2 [Nbxmod], page 45, for more details.

The actual generation process proceeds in four phases. First, all the atoms specified by the residues are added to the PSF. Next, all the terms are added, and all linkage references can be correctly handled, see Section 3.1.5.1 [Linkage Atom Naming], page 28. Next, the automatic generation operations are performed. Finally, the patches are performed.

4.2 `NBXMOD` — Automatic Generation of Non-bonded Exclusions

Some pairs of atoms are excluded from the nbond exclusion lists because their interactions are described by other terms in the hamiltonian. By default directly bonded atoms and the 1-3 atoms of an angle are excluded from the nonbond calculation. In addition the diagonal interactions of the six membered rings in tyrosine and phenylalanine and ring atom interactions in tryptophan are excluded in the current topology files. Hydrogen bonds, and dihedral 1-4 interactions are not excluded (note that other workers may differ from us on one or both of these points).

The list of nonbonded exclusions is generated in two steps. First a preliminary list is made at generation by `GENIC` using any information that may be present in the topology file (as for example might be diagonal interactions in rings). The second step is an automatic compilation of all the bond and angle interactions, followed by a sorting of the list, performed in `MAKINB`. The list is stored in the linked list pair `IBLO/INB`, where `IBLO(i)` points to the last exclusion in `INB` to atom `i`. If the list is modified after `MAKINB`, then either `MAKINB` should be called again to resort the list, or care must be taken to see that the `INB` list is ascending with all `INB` entries having higher atom numbers than `i` and that all atoms have at least one `INB` entry.

`MAKINB` is called by default after any operation which changes internal coordinates such as generate, patch, edit, or splice.

The default list can be modified in three ways. First, interactions that are to be excluded can be placed in the topology file. Second, the `NBXMOD` option can be specified as a qualifier to any of the commands which change internal coordinates. Its values and actions are:

0	use only the exclusions in the topology file. *This option is not recommended, because there are no check to ensure that the non-bonded exclusions for an atom are defined only above each atom.* `MAKINB` *will correct this automatically.*
1 or -1	include bond interactions automatically.
2 or -2	also include 1-3 interactions automatically.
3 or -3	also include 1-4 interactions automatically.

Note that the 1-3 and 1-4 interactions are determined from examination of the bond list regardless of any torsions or improper torsions which are defined.

Negative values suppress the use of the information present in the topology file. Positive values add to the information that was in the topology file. If `NBXMOD` is not specified for a command, it defaults to 2.

The third way to change exclusions is the use of the EDIT command, see Section 4.5 [Edit Command], page 47.

4.3 DISULFIDE — Creates Internal Coordinates for Disulfide Bridges

4.3.1 Syntax

```
DISUlfide [NBXMod int]

NCYST      (I5)

IRES,JRES  (2I5)  repeated NCYST times
```

This command requires formatted (not free field) input following the DISULFIDE line.

4.3.2 Function

The first line following the command gives the number of cystine cross bridges to be made; The lines following give the residue numbers (not residue identifiers) of the cysteines to be linked. The residue number of a residue is its position in the list of all residues in the structure including any special termini which have been added. The linkage process involves adding bonds, bond angles, torsion angles, and non-bonded exclusions for the additional bond. The NBXMOD option controls the automatic generation of non-bonded exclusions, see Section 4.2 [Nbxmod], page 45, for more details.

An attempt has been made to ease the burden of going from residue identifiers to residue numbers which will be different for segments which have N-terminal residues added. Whenever the type of segment as specified in the READ SEQUENCE command is CHARMM explicit hydrogen or all hydrogen, the residue numbers will first be increased by one. If the two residues given are not cysteines, the residue numbers will be decremented by one, and the attempt repeated. If this fails, the command will die. Note that this does not help you if you have more than one segment in your structure.

When using disulphides, it is important that the sequences reference a cysteine residue which is intended to be joined. In an all atom topology file, see Section A.6.5 [AMBER94RTF], page 297, there are two cysteine residues, one of which has a thiol and one of which has a sulfur.

It is not possible to bridge an atom in the primary space with that of a symmetric image using this command (see Chapter 9 [Images], page 87).

4.4 PATCH — Patches Special Structures

4.4.1 Syntax

```
PATCh HEME [NBXMod int]
histidine-heme-spec

          or

PATCh LIGA [NBXMod] int]
carbonmonoxide-heme-spec
```

4.4.2 Function

PATCH HEME is used to patch the ligation of a histidine to a heme residue. The histidine-heme-spec is a pair of integers read by a format of 2I5, IRES and JRES, are the residue numbers of the histidine and heme, respectively. The bond is formed between the NE2 of the histidine and FE of the HEME. For each bond formed, additional bond angles, torsion angles, and non-bonded exclusions are added.

`PATCH LIGAND` is used to patch the ligation of a carbon monoxide to a heme. The carbon monoxide-heme-histidine-spec consists of three integers, read by a format of `3I5`, and again are not free format. The three numbers refer to the histidine, heme, and CO residue numbers respectively. This patch takes care of the carbon monoxide heme bond. It should be called after `PATCH HEME` is called.

The `NBXMOD` option controls the automatic generation of non-bonded exclusions, see Section 4.2 [Nbxmod], page 45, for more details.

WARNING: This code has been tested only with the extended atom topology file. It may not work for current editions of the other amino acid topology files, and it will not work if there is a proton placed on the NE2 of the histidine.

It is not possible to patch any interaction involving atoms of a symmetric image using these command (see Chapter 9 [Images], page 87).

4.5 `EDIT` — Edits the PSF and Hydrogen Bonds)

4.5.1 Syntax

```
EDIT [NBXMod int] [[NO]ANGLes] [NOTORSions      ] [[NO]DONOrs] [[NO]ACCEptors]
                               [TORSions {ALL}]
                               [         {ONE}]

edit-commands
```

`edit-commands` are described below. They must be terminated with an `END` command. The `edit-commands` are not free field.

The automatic generation options which apply for the `GENERATE` command also apply here. See Section 4.1 [Generate Command], page 43, for more information.

4.5.2 Function

`EDIT` is used to edit the PSF and also the a hydrogen bond list without explicit hydrogens. The following operations are possible: Any bond, bond angle, torsion angle, improper torsion angle, hydrogen bond donor, hydrogen bond acceptor, non-bonded exclusion, or hydrogen bond may be deleted or added. In addition, the parameter type code, charge, and IUPAC name of any atom may be changed.

The operations with hydrogen bonds and with hydrogen bond donors and acceptors are obsolescent as one cannot add the proton to any new hydrogen bonds, nor can one add the various antecedents to the hydrogen bond donor. At some point, this will be fixed.

The edit-commands are all fixed format commands. Each command except the `END` command consist of three parts. First, one specifies an alphabetic command using words read with a `2(A4,6X)` format. The following line consists of single integer using `(I5)` format giving the number of changes. Finally, that number of lines follows, where each line specifies one change.

The `END` command consists of the word `END` in the first of column of a line with nothing else on the line. This terminates the `EDIT` command.

The `NBXMOD` option controls the automatic generation of non-bonded exclusions, see Section 4.2 [Nbxmod], page 45, for more details.

4.5.3 `ADD` and `DELETE` — Modify the Structure Arrays

4.5.3.1 Syntax

```
{ ADD    } { BOND     }     (2A10) Fixed format
{ DELEte } { THETa    }
           { PHI      }
           { IMPHi    }
           { HBONd    }
           { DONOr    }
           { ACCEptor }
           { NONBond  }

 NCHANG   (I5)

 NRESI,ATOMI,NRESJ,ATOMJ,NRESK,ATOMK,NRESL,ATOML  (4(I5,1X,A4))
    repeated NCHANG times
```

4.5.3.2 Function

Most of the keywords are self explanatory. In the case on `NONBOND`, the nonbonded exclusions list is changed, not the actual list of nonbonded interactions. `NCHANG` is the number of elements to be added or deleted. `NRESI` specifies the residue number of the first atom, and `ATOMI` specifies its name. For example, deleting a peptide bond between the fourth and fifth residue would be specified as

```
    4 C       5 N
```

If a number of different internal coordinates are to be changed, separate `ADD` or `DELE` commands must be used. They can appear in any order. One must specify only as many atoms as there are in the interaction. For hydrogen bonds, only two atoms may be specified even though the proton may be explicit represented.

WARNING: This routine does not make the correct checks if you make a mistake. If you specify an atom incorrectly, or if the interaction you wish to delete is not there, the results are unpredictable. At some point, this may be fixed.

4.5.4 MODIFY — Changes the Attributes of an Atom

4.5.4.1 Syntax

```
MODIfy         (A4)    Fixed format

NCHANG  (I5)

NREST,ATOMT,ICODE,ANAME,CHARG  (I5,1X,A4,I5,1X,A4,F10.3)
   repeated NCHANG times
```

4.5.5 Function

`NCHANG` is the number of atoms to be modified. A card with the changed values must be read in for each atom to be changed (in any order). `NREST` and `ATOMT` are the residue number and atom type of the atom to be changed. `ICODE`, `ANAME` and `CHARG` are the new chemical type code (`IAC` array entry), new atom type and new atomic charge given to the atom. If a field is left blank, the old value is retained. Editing of this type could be used, for example, to change the carbonyl oxygen of a terminal residue to an atom with attributes corresponding to a carboxyl oxygen (in the standard execution of the program, modifications like this are done automatically if the proper type is specified with the sequence).

5 Generation of Hydrogen Bonds

The generation of hydrogen bonds is one of the major steps in evaluating the energy of a system. The process of hydrogen bond generation involves looking at all possible pairs of hydrogen bond donors and acceptors and selecting those which are "good". The meaning of "good" is determined by parameters to be described below. Depending on the setting of the `H-A` or `D-A` flags in the parameter file, see Section 3.1.4.1 [Parameter File Format], page 21, hydrogen bonds can involve either two or three atoms, respectively. Two atom hydrogen bonds will not have any angular term calculated.

The selection of hydrogen bonds involves several checks. First, any good hydrogen bond has be shorter than the distance cutoff, `CUTHB`. If the `H-A` parameter flag is set, hydrogen bond distances are measured from the hydrogen to the acceptor. If the `D-A` flag is set, hydrogen bond distances are measured from the heavy atom donor to the heavy atom acceptor. A second test involving the heavy atom donor, hydrogen, and acceptor applies for the three atom case. The angle off linearity has to be less than the angular cutoff, `CUTHBA`. This angle is measured as the complement of donor - hydrogen - acceptor angle. Thus, a linear hydrogen bond would be measured as a zero angle. In all cases, hydrogen bonds are not calculated for any set of atoms where any pair of atoms are excluded from non-bonded interactions. see Chapter 6 [Non-bonded Interactions], page 51.

Because there are cutoffs involved with the selection of hydrogen bonds, and because the hydrogen bond list must be updated during dynamics, and because energy must be conserved, switching functions are needed to smooth the transition over a cutoff. Therefore, the specification of hydrogen bond generation also allows the specification of switching function parameters.

The generation is performed by CONGEN at several different points. One can request the hydrogen bonds be generated explicitly using a hydrogen bond command. This is useful prior to analyzing the system. The hydrogen bonds can be generated during any energy manipulation, see Chapter 7 [Energy Manipulations], page 55. When two calculations are being compared in the analysis facility, hydrogen bonds may be generated, see Chapter 18 [Comparisons], page 173. When any such generation is requested except in the analysis facility, the same specifications are made for the hydrogen bond generation and switching function parameters.

If images are present (see Chapter 9 [Images], page 87) and the selection of image atoms has occurred, then all hydrogen bonds between primary atoms and image atom will also be generated. During dynamics or minimization, the image hydrogen bonds must be updated each time new image atoms are selected (in nonbonded update), thus there is a forced generation of image hydrogen bonds (unless the update frequencies match).

5.1 Syntax of the Hydrogen Bond Command

```
HBONd   hbond-spec

hbond-spec ::= [DUMMy] [CUTHB real] [CUTHBA real]

               [CTONHB real] [CTOFHB real] [CTONHA real] [CTOFHA real]
```

Note: this syntax is also used in the `HBUILD` command, see Section 15.2 [Function of Coordinate Manipulations], page 151, for more information.

5.2 Purpose of the Various Hydrogen Bond Variables.

```
Variable  Default      Function
```

DUMMy		Sets CUTHB and CUTHBA to zero. This will result in no hydrogen bonds which is desirable when one is not interested in the hydrogen bond energy. The selection will be done very quickly in this case.
CUTHB	4.5	Maximum distance allowed for a hydrogen bond. This distance is measured between the heavy atoms
CTOFHB	CUTHB-0.5	Distance where distance switching function is off Once specified, it will only change if respecified.
CTONHB	CTOFHB-0.5	Distance where distance switching function is on. Once specified, it will only change if respecified.
CUTHBA	90.0	Maximum out of line angle allowed for a three atom hydrogen bond. The angle is 180 - D--H...A angle. If this value is less than or equal to 0, then all possible angles will be allowed.
CTOFHA	CUTHBA-20.0	Angle where angle switching function is off Once specified, it will only change if respecified.
CTONHA	CTOFHA-20.0	Angle where angle switching function is on. Once specified, it will only change if respecified.

6 Generation of Non-bonded Interactions

Nonbonded interactions (frequently abbreviated NBONDS) refer to all those terms in the hamiltonian which do not refer specifically to a bond, bond angle, dihedral, or hydrogen bond. They include the attractive and repulsive van der Waals terms and the electrostatic terms between all atom pairs that are not specifically excluded from nonbond calculations. Directly bound atoms (1-2 interactions) and atoms bound to a common atom (1-3 interactions) are usually not counted in non-bonded interactions. See Section 4.2 [Nbxmod], page 45, for an automated command to handle non-bonded exclusions. These terms are defined on atom pairs and to a first approximation would require the number of atoms squared calculations. To avoid this burden, various truncation and approximation schemes can be employed in the program, breaking the nonbonded calculation into two parts, initialization and actual energy calculation.

The method of approximation, cutoffs, and other relevant parameters can be entered either at the time of initialization (by calling NBOND from the main program or in the COMPARE command in analysis), or in the MINI and DYNA commands.

6.1 Syntax for Generation of Non-bonded Interactions

```
NBOND   [keyword value]

COMPARE  ... NBOND [keyword value] <del>  ...

MINI    [keyword value]

DYNA    [keyword value]
```

In all cases as many keywords and values as desired may be specified.

6.2 Keywords for Generating Non-Bonded Interactions

The key words, their functions, and defaults are:

ATOM	(default) Non-bonded interactions are calculated atom by atom with switching functions being used.
RESI	Non-bonded interactions are determined residue by residue using a rectangular box search. Switching functions are calculated for every atom-atom pair.
SHFT	Non-bonded interactions are calculated atom by atom and the potential is shifted to smooth it at the cutoff.
EXEL	Extended electrostatics using dipoles for all residues outside the CUTNB cutoff.
EXFL	Extended electrostatics using dipoles and quadrupoles for all residues outside the CUTNB limits.
EXNQ	Extended electrostatics to first order without residue quadrupoles.
CONS	Similar to ATOM except a constant dielectric is used.
DUMMy	Generates an empty non-bonded interaction list.
CUTNB	Distance cutoff in generating the list of pairs. The default is 8.0 Angstroms for ATOM, RESI, and SHFT; the default is 5.0 Angstroms for EXFL, EXEL, CONS.
CTOFNB	Distance cut at which the switching function eliminates all contributions from a pair in calculating energies. Once specified, This value is not reset unless respecified. The default is CUTNB-0.5, or CUTNB-0.2 for EXFL and EXEL.

`CTONNB`	Distance cut at which the smoothing function begins to reduce a pair's contribution. This value is not used with `SHFT`. Once specified, This value is not reset unless respecified. The defaults is `CTOFNB`-1.0, or `CTOFNB`-0.3 for `EXFL` and `EXEL`.
`CUTMP`	Distance cut at which the extended electrostatics routines change from atom by atom evaluation of distant residues to a multipole and polynomial expansion. The default is 0.0.
`EPS`	Dielectric constant for the extended electrostatics routines (`ATOM` and `RESI` options set the dielectric equal to r times the `EPS` value). The default is 2.5 for constant dielectric, and 1.0 for r dependent dielectric.
`VDWO`	If `VDWO` is specified then `EPS` is set to 0.
`WMIN`	Warning cutoff for minimum atom to atom distance. Pairs are checked during close contact list compilation using the `ATOM`, `EXEL`, `EXFL` and `CONS` options. The default is 1.5 Angstroms.
`WRNMXD`	Warning cutoff for maximum atom displacement from the last close contact list update (used only in `EXEL` and `EXFL`). The default is 0.5 Angstroms.

6.3 Algorithms for Generating Non-bonded Interactions

There are four algorithms used in calculating the nonbonded energies, each making different approximations in an attempt to speed the calculation. Electrostatic interactions are the most difficult to deal with for two reasons. They do not fall off quickly with distance (so it is inappropriate to simply ignore all interactions beyond some cutoff), and they depend on odd powers of r necessitating expensive square root calculations for each pair evaluated. The approximations used to make the electrostatics calculation more tractable are setting the dielectric constant equal to r or using a constant dielectric but only calculating distant interactions periodically (and storing the value in between).

Setting the dielectric constant equal to the atom atom distance times a constant factor (determined by the `EPS` keyword value) makes the computation easier by eliminating the need to calculate square roots and by making the calculated contribution fall off more quickly. It also introduces problems. The force calculated using an *r* dependent dielectric will be larger than the force from a constant dielectric at short distances (5.0 angstroms or less by comparsion to a constant dielectric of 2.5). In addition, the electrostatic contribution still falls off relatively slowly and large distance cutoffs are needed. As the number of atom pairs included will be proportional to the cutoff cubed, this is a significant disadvantage.

The `RESI` options use an *r* dependent dielectric (as do all of Bruce Gelin's protein calculations). In the `RESI` searches, the program constructs a rectangular box around every residue, and examines every residue–residue pair that can yield atom atom pairs within the cutoff. Every atom–atom pair within those residue–residue pairs is added to the non-bonded list. When switching functions are used, `RESI` and `ATOM` will yield the same energy, but `RESI` will generate a longer list.

The `SHFT` option is similar to `ATOM` except, the potential:

$$E = \frac{q_i q_j}{\mathrm{EPS}}\left(\frac{1}{R^2} + \frac{R^4}{2.0 \times \mathrm{CTOFNB}^6} - \frac{1.5}{\mathrm{CTOFNB}^2}\right)$$

is used when (R < `CTOFNB`) and zero otherwise. This potential and it first derivative approach zero as R becomes `CTOFNB`, without the messy computation of switching functions and steep forces at large R.

The EXEL and EXFL routines use a constant dielectric at large distances joined to the short distance function. The long distance terms are approximated using interpolation and periodic updating. At CUTNB the constant dielectric potential is joined to a dielectric equal to r potential with a shift and scale that preserve the continuity of the energy and forces. CONS uses a constant dielectric everywhere. This requires a square root to be calculated in the inner loop of ENBOND, slowing things down a bit, but it is physically more reasonable and widely employed by other groups doing empirical energy modelling (ex. ST2 water).

This form allows a small CUTNB (5.0 angstroms with EPS=2.5) even though the electrostatic terms are still varying rapidly at that distance. The short range forces are identical to those calculated with the other options, reflecting the decrease in dielectric shielding at short ranges. For the long range forces there is effectively no cutoff in the electrostatic energy. Interactions between atoms beyond the CUTNB are calculated when the list is updated and stored together with their first (EXFL,CONS) or first and second (EXEL) derivatives. The energy is calculated by explicitly evaluating pairs in the list and using the stored potentials, fields, and gradients to approximate the distant pairs. In essence the routines assume that for distant pairs the atom movements will be small enough that the changes in their electrostatic interactions can be accurately calculated using local expansions.

The constant dielectric routines compile the close contact list using the same two stage minimum rectangle box search that is described above. In this way the efficiency of a residue by residue search is exploited while being certain that all necessary pairs are included. For close residue pairs an atom by atom search is then performed. Atom pairs are either included in the list of close contacts or their electrostatic interactions are calculated and stored.

An option is offered to increase the accuracy of residue residue interactions by using a multipole expansion of one residue evaluated for each atom of the other. This cutoff for this treatment is CUTMP. For residue pairs outside of CUTMP only a single multipole evaluation is made and second order polynomial expansion is used to extrapolate to each atom. Ordinarily this is sufficient and CUTMP is set to 0.0.

The DUMMY option results in no non-bonded interactions whatsoever. The creation of a dummy list can be done very quickly. This is useful when one is analyzing only the other terms in the energy.

6.4 Implementation of Non-bonded Interactions

The initialization and list compilation is performed by the subroutine NBONDS. It functions by guessing how much space will be needed to store the close contact list, allocating that space (and space for electrostatic potentials and gradients if necessary) on the heap, and calling the appropriate subroutine to actually compile the nonbonded list. NBONDA, NBONDR, NBONDE, and NBONDF handle ATOM, RESI, EXEL, and EXFL respectively. If sufficient space was not available, 1.5 times as much is allocated and another attempt is made.

ENBOND evaluates the nonbonded energy, calling EEXEL to evaluate the stored electric potentials and fields. Single precision is used for most arithmetic, but the energy and derivative sums are accumulated in double precision.

All of the nonbonded cutoffs and lists are stored on the heap. BNBND is the descriptor array passed through most of the program (in some of the analysis routines an additional array BNBNDC is used for the comparison data structure). BNBND holds heap addresses and LNBND holds the lengths of the elements in the data structure. To actually access the data it is necessary to include '`inbnd.fcm`' (an index common block) and specify `HEAP(BNBND(xxx))` where `xxx` is the desired element name in '`inbnd.fcm`'. This is arrangement has the advantage of allowing dynamic storage allocation and easy modification of the types of information passed from routine to routine.

6.5 Data Structures for Non-bonded Interactions

The contents of the nonbonded data structure are described in the file '`inbnd.fcm`' in the CONGEN source directory. The following gives some of the details.

```
NBOPT - the nonbonded option number
        0 = RESI
        1 = ATOM
        2 = EXEL
        3 = EXFL
        4 = SHFT
        5 = EXNQ
        6 =
        7 = CONS
        8 = DUMMY

CUTNB

CTOFNB

CTONNB

CUTMP

EPS

WMIN

WRNMXD

JNB - the list of the second atom in close contact list

INBLO - a list of pointers into JNB organized on the
        first atoms of each pair INBLO(i) is the location of
        the last entry in JNB that pairs with atom i
```

The following elements are only used by `CONS` `EXEL` and `EXFL`:

```
ATSX, ATSY, ATSZ - the atom locations at the last update

ATPOT - the atom potentials

ATFX, ATFY, ATFZ - the first derivatives
```

The following elements are used only by `EXEL`

```
ATGXX, ATGYY, ATGZZ, ATGXY, ATGYZ, ATGZX - the second derivatives
                                           in the potential
```

7 Minimization and Dynamics

An important function of CONGEN is the evaluation and manipulation of the potential energy of a macromolecular system. Two types of manipulation are supported. First, one can minimize the energy by adjusting the coordinates of all the atoms in order to reduce its value. Several minimization algorithms are provided. Second, one can solve Newton's equations of motion using the potential energy in order to obtain a dynamics trajectory.

The commands which invoke these three types of manipulation are all parsed by the same routine, and any common actions are done therein.

7.1 Syntax for Energy Manipulation Commands

```
{ MINImize }
{ DYNAmics } repeat(keyword) repeat(keyword value)
{ ENERgy   }

value ::= {  real   }
          { integer }
```

Syntactic ordering: All keywords and keyword value pairs can appear in any order.

Table of Keywords with references

Keyword	Reference (where to read about it)
ABNR	See Section 7.5 [Minimization], page 59.
AKMASTP	See Section 7.6 [Dynamics], page 61.
CONJ	See Section 7.5 [Minimization], page 59.
CRASHU	See Section 7.6 [Dynamics], page 61.
DYNAmics	See Section 7.6 [Dynamics], page 61.
EIGRNG	See Section 7.5 [Minimization], page 59.
FINALT	See Section 7.6 [Dynamics], page 61.
EKETEST	See Section 7.6 [Dynamics], page 61.
ETETEST	See Section 7.6 [Dynamics], page 61.
FIRSTT	See Section 7.6 [Dynamics], page 61.
hbond-spec	See Chapter 5 [Hydrogen Bonds], page 49.
IASORS	See Section 7.6 [Dynamics], page 61.
IASVEL	See Section 7.6 [Dynamics], page 61.
ICHECW	See Section 7.6 [Dynamics], page 61.
IEQFRQ	See Section 7.6 [Dynamics], page 61.
IHBFRQ	See Section 7.4 [General Energy Operands], page 58, and see Chapter 9 [Images], page 87
IHTFRQ	See Section 7.6 [Dynamics], page 61.
IMAXP	See Section 7.4 [General Energy Operands], page 58.

`INBFRQ`	See Section 7.4 [General Energy Operands], page 58.
`IPRFRQ`	See Section 7.6 [Dynamics], page 61.
`IPRINT`	See Section 7.4 [General Energy Operands], page 58.
`ISCALE`	See Section 7.6 [Dynamics], page 61.
`ISCVEL`	See Section 7.6 [Dynamics], page 61.
`ISEED`	See Section 7.6 [Dynamics], page 61.
`IUNCRD`	See Section 7.6 [Dynamics], page 61.
`IUNREA`	See Section 7.6 [Dynamics], page 61.
`IUNVEL`	See Section 7.6 [Dynamics], page 61.
`IUNWRI`	See Section 7.6 [Dynamics], page 61.
`KUNIT`	See Section 7.6 [Dynamics], page 61.
`MASS`	See Section 7.5 [Minimization], page 59.
`MINDIM`	See Section 7.5 [Minimization], page 59.
`MINImize`	See Section 7.5 [Minimization], page 59.
`NCGCYC`	See Section 7.5 [Minimization], page 59.
`non-bond-spec`	See Chapter 6 [Non-bonded Interactions], page 51.
`NPRINT`	See Section 7.6 [Dynamics], page 61.
`NRAP`	See Section 7.5 [Minimization], page 59.
`NSAVC`	See Section 7.6 [Dynamics], page 61.
`NSAVV`	See Section 7.6 [Dynamics], page 61.
`NSTEP`	See Section 7.4 [General Energy Operands], page 58.
`NTRFRQ`	See Section 7.6 [Dynamics], page 61.
`PCUT`	See Section 7.5 [Minimization], page 59.
`PRTMIN`	See Section 7.5 [Minimization], page 59, and Chapter 10 [Vibrational Analysis], page 89.
`SCALE`	See Section 7.6 [Dynamics], page 61.
`SD`	See Section 7.5 [Minimization], page 59.
`STEP`	See Section 7.5 [Minimization], page 59.
`STPLIM`	See Section 7.5 [Minimization], page 59.
`STRICT`	See Section 7.5 [Minimization], page 59.
`TEMINC`	See Section 7.6 [Dynamics], page 61.
`TFREQ`	See Chapter 10 [Vibrational Analysis], page 89.
`TIMESTP`	See Section 7.6 [Dynamics], page 61.
`TLIMIT`	See Section 7.6 [Dynamics], page 61.
`TOL`	See Section 7.6 [Dynamics], page 61.
`TOLENR`	See Section 7.5 [Minimization], page 59.

`TOLGRD` See Section 7.5 [Minimization], page 59.

`TOLITR` See Section 7.5 [Minimization], page 59.

`TOLSTP` See Section 7.5 [Minimization], page 59.

`TSTRUC` See Section 7.6 [Dynamics], page 61.

`TWINDH` See Section 7.6 [Dynamics], page 61.

`TWINDL` See Section 7.6 [Dynamics], page 61.

`VIBRation`
See Chapter 10 [Vibrational Analysis], page 89.

7.2 Requirements for Energy Manipulations

Before the energy of a system can be evaluated and manipulated, a number of data structures must be present.

First, a PSF must be present.

Second, a parameter set must be present. It must contain all parameters which are required by the PSF being used.

Third, coordinates must be defined for every atom in the system. An undefined coordinate has a particular value, and if two coordinates have the same value, division by zero will occur in the evaluation of the energy. If the positions of hydrogens are required, the hydrogen bond generation routine, see Section 15.2.17 [HBUILD Command], page 155), must be called before the energy is evaluated. In the case of energy evaluations, CONGEN will ignore interactions involving undefined atom positions. However, minimization and dynamics will fail if any positions are unknown.

Fourth, provisions must be made for having a hydrogen bond list and a non-bonded interaction list. Having non-zero frequencies for updating this lists is one way, one can also read these lists in, see Section 3.1 [Read Command], page 15, or generate them with separate commands, see Chapter 5 [Hydrogen Bonds], page 49, or Chapter 6 [Non-bonded Interactions], page 51.

All of these requirements can be met through the use of a restart file, see Section 7.6 [Dynamics], page 61.

7.3 Energy Manipulation Options

There exist several commands which can modify the way the potential energy is calculated or can affect the way energy manipulations are performed.

The `CONSTRAINT` command, see Chapter 11 [Constraints], page 95, can be used to set constraints of various kinds. First, it can be used to set flags for particular atoms which will prevent them from being moved during minimization or dynamics. Second, it can be used to add positional constraint term to the potential energy. This term will be harmonic about some reference position. The user is free to set the force constant. Third, the user can place a harmonic constraint on the value of particular torsion angles in an attempt to force the geometry of a molecule. Fourth, atoms can be fixed in place. When atoms are fixed, they are not allowed to move at all. Any energy terms involving nothing but fixed atom are ignored, because they will have no effect. In addition the trajectories written by the dynamics command only have one copy of the fixed atom coordinates written. Thus, such trajectories take less disk space.

The `SHAKE` command, see Section 11.6 [Shake Command], page 107, is used to set constraints on bond lengths and also bond angles during dynamics. It is very valuable in that it permits a larger step size to be used during dynamics. This is vital for dynamics where hydrogens are explicitly represented as the low mass and high force constant of bonds involving hydrogen require a ridiculously small step size.

The `WEIGHT` command, see Section 27.12 [Weight Command], page 233, is used to change the weights on each of the terms in the potential energy function. This is useful during dynamics with NMR constraints when one is interested in getting to the final solution as quick as possible.

The user interface commands, see Section 2.11 [CONGEN Modifications], page 11, can be used to modify the calculation of the potential and to add another term to the potential energy.

7.4 Options Common to Minimization and Dynamics

The following table describes the keywords which apply to both minimization and dynamics. See Chapter 6 [Non-bonded Interactions], page 51, for a more detailed description of the non-bonded options. See Chapter 5 [Hydrogen Bonds], page 49, for more information on hydrogen bonds.

```
Keyword  Default  Purpose

NSTEP    100      The number of steps to be taken. For minimization, this
                  is the number of cycles of minimization, not the
                  number of energy evaluations. For dynamics, this is
                  the number of dynamics steps which is also equal to
                  the number of energy evaluations.

INBFRQ   50       The frequency of regenerating the non-bonded list.
                  The list is regenerated if the current step number
                  modulo INBFRQ is zero and if INBFRQ is non-zero.
                  Specifying zero prevents the non-bonded list from being
                  regenerated at all.

IHBFRQ   50       The frequency of regenerating the hydrogen bond list.
                  Analogous to INBFRQ.

non-bond-         The specifications for generating the non-bonded list.
-spec

hbond-            The specifications for generating the hydrogen bond list.
-spec

IPRINT   0        If given a non-zero value, causes the printing of
                  all contributions to the energy whose atoms are less
                  than IMAXP, more or less, every IPRINT cycles. Useful
                  in debugging although somewhat obsolete since the
                  analysis facility can provide the same information in
                  a more readable form.  There may be bugs in the
                  frequency of when the printing occurs.

IMAXP    0        In conjunction with IPRINT, causes the print out of
                  contributions to the energy. This keyword determines
                  the largest atom number which has its contributions
                  printed. In some cases the check is made on all atoms
                  in the interaction; in others, the check is made on
                  just the first.
```

```
NPRINT   1        The frequency with which energies are printed during
                  course of dynamics or ABNR minimization.
```

7.5 Minimization Options

There are several different algorithms for minimizing the energy of the system. They all involve calculating the derivative of the potential energy, and possibly the second derivative, and using that information to adjust the coordinates in order to find a lower energy. In the descriptions of the algorithms below, a capitalized keyword at the beginning of each paragraph is the key word used to invoke that method. After the descriptions is a listing of all keywords associated with minimization.

The simplest minimization algorithm is steepest descents (`SD`). In each step of this iterative procedure, we adjust the coordinates in the negative direction of the gradient. It has one adjustable parameter, the step size, which determines how far to shift the coordinates at each step. The step size is adjusted depending on whether a step results in a lower energy. I.e., if the energy drops, we increase the step size by 20% to accelerate the convergence. If the energy rises, we overshot a minimum, so the step size is halved. Steepest descents does not converge in general, but it will rapidly improve a very poor conformation.

A second method is the conjugate gradient technique (`CONJ`) which has better convergence characteristics (R. Flectcher & C.M. Reeves, *The Computer Journal* **7**:149 (1964)). The method is iterative and makes use of the previous history of minimization steps as well as the current gradient to determine the next step. It can be shown that the method converges to the minimum energy in `N` steps for a quadratic energy surface where `N` is the number of degrees of freedom in the energy. Since several terms in the potential are quadratic, it requires less evaluations of the energy and gradient to achieve the same reduction in energy in comparison to steepest descents. Its main drawback is that with very poor conformations, it is more likely to generate numerical overflows than steepest descents. The algorithm used in CONGEN has a slightly better interpolation scheme and automatic step size selection.

A third method involves solving the Newton-Raphson minimization equations iteratively (`NRAP`). This procedure requires the computation of the derivative of the gradient which is a matrix of size O(n**2). The procedure here is to find a point where the gradient will be zero (hopefully a minimum in energy) assuming that the potential is quadratic. The Newton-Raphson equations can be solved by a number of means, but the method adopted for CONGEN involves diagonalizing the second derivative matrix and then finding the optimum step size along each eigenvector. When there are one or more negative eigenvalues, a blind application of the equations will find a saddle point in the potential. To overcome this problem, a single additional energy and gradient determination is performed along the eigenvector displacement for each small or negative eigenvalue. From this additional data, the energy function is approximated by a cubic potential and the step size that minimizes this function is adopted. The advantages of this method are that the geometry cannot remain at a saddle point, as sometimes occurs with the previous procedures, and that the procedure converges rapidly when the potential is nearly quadratic (or cubic). The major disadvantage is that this procedure can require excessive storage requirements, o(n**2), and computation time, o(n**3), for large molecules. Thus we are currently restricted to systems with about 200 atoms or less for this method. This method may not be used when Images are used (see Chapter 9 [Images], page 87).

The fourth method available is an adopted basis Newton-Raphson method (`ABNR`) (D. J. States). This routine performs energy minimization using a Newton-Raphson algorithm applied to a subspace of the coordinate vector spanned by the displacement coordinates of the last (`MINDIM`) positions. The second derivative matrix is constructed numerically from the change in the gradient vectors, and is inverted by an eigen vector analysis allowing the routine to recognize and avoid saddle points in the energy surface. At each step the residual gradient vector is calculated and used to add a steepest descent step onto the Newton-Raphson step, incorporating the new direction into the basis set.

A fifth method involving an amalgam of conjugate gradient and steepest descents (`GCSD`) is currently unavailable due to a bug in the command parser.

In the table which follows, keywords enclosed in square brackets means that one can choose one in the set. Such enclosed keywords do not expect a value after them. All other keywords are used for specifying values, See Section 7.1 [Energy Manipulation Syntax], page 55. The method column shows which method the keyword affects. See Section 7.4 [General Energy Operands], page 58, for common variables.

Keyword	Default	Method	Purpose
[CONJ]	CONJ		Do conjugate gradient minimization.
[SD]			Do steepest descent minimization.
[NRAP]			Do Newton-Raphson minimization.
[ABNR]	[MASS]		Do Adopted Basis Newton-Raphson minimization, with mass weighted forces if specified.
[CGSD]			This is to combine CONJ and SD (Not available)
STEP	.02	ALL	Initial step size for the minimization algorithms. Reasonable values for the various methods are best determined by trial and error.
PRTMIN	1	ALL	A flag indicating how much to print during minimization.
		SD	No effect
		CONJ	If less than 2, the energy is printed only once each cycle. A setting of 2 shows the energy for each evaluation plus variables used in the method.
		NRAP	if greater than 1, a brief overview of the least squares cubic fitting procedure given for eigenvalues less than TFREQ.
		ABNR	If less than 2, the energy is printed out only for successful steps (improvements in total energy). Some description of how the step was chosen is printed if it is set to 2, and a very verbose description is given for values of 3 or higher.
NCGCYC	100	CONJ	The number of conjugate gradient cycles executed before the algorithm restarts. This number will be automatically lowered to the shortest hydrogen bond or non-bonded list update frequency. The algorithm will fail if the potential is changed will it is running.
PCUT	.9999	CONJ	If the cosine of the angle between the old and new P vector is greater than PCUT, the algorithm will be restarted. This prevents the algorithm from plodding down the same path repeatedly. If PRTMIN is less than 2, one effect of the restart is that the step

			size will go its initial value. If this happens many times, you've converged.
EIGRNG	.0005	ABNR	The smallest eigenvalue (relative to the largest) that will be considered nonsingular.
MINDIM	5	ABNR	The dimension of the basis set stored.
STPLIM	1.0	ABNR	The maximum Newton Raphson step that will be allowed.
STRICT	0.1	ABNR	The strictness of descent. The energy of a new step must be less than the previous best energy + STRICT for the new step to be accepted.
MASS	---	ABNR	Use unweighted forces by default or mass weighted if specified. Mass weights converge more slowly but allow association with normal mode frequencies.
TFREQ	1.0	NRAP	The smallest eigenvalue that is considered to be non-negative (i.e. do cubic fitting on all eigenvalues smaller than this).
NFREQ	NATOM*3	NRAP	The number of degrees of freedom to be optimized (the number of lowest eigenvalues). Use the default whenever practical.
TOLENR	0.0	ABNR	A tolerance applied to the change in total energy change during a cycle of minimization (NCYCLE steps) If the energy change is less than or equal to TOLENR, the minimization routine will exit.
TOLGRD	0.0	ABNR	A tolerance applied to the average gradient during a cycle of minimization. If the average gradient is less than or equal to TOLGRD, the routine will exit.
TOLITR	100	ABNR CONJ	The maximum number of energy evaluations allowed for a single step of minimization.
TOLSTP	0.0	ABNR	A tolerance applied to the average step size during a cycle of minimization. If the average step size is less than or equal to TOLSTP, the routine will exit.

7.6 Running Molecular Dynamics

The theoretical basis for dynamical simulations is elementary physics. The force on a particle is equal to the negative gradient of the potential energy of the particle. CONGEN can solve this equation numerically for all atoms in the molecule. It has two different methods available, a simple second order predictor two step method due to Verlet and a fifth order predictor - corrector multivalue method described by Gear.

In either algorithm, the choice of step size is very important. One must weigh the increased accuracy of using a small step size against the longer real time that can be simulated with a given amount of execution time when a larger step size is used. The time step may be entered in either picoseconds (using the `TIMESTP` keyword) or the internal AKMA units (using the `AKMASTP` keyword).

CONGEN provides information on the accuracy of the numerical solution. Since the system has no external forces, the total energy should be conserved. Numerical errors will result in some fluctuations in the total energy so a good test is to compare the fluctuations in total energy to the fluctuations in kinetic energy as these fluctuations are proportional to the heat capacity of the system. See the next node for a description of dynamics output.

Because the force constants for the bonds and bond angles are fairly large, it is reasonable under certain circumstances to constrain their values during dynamics. Such constraints are applicable if the harmonic motions are weakly coupled to other motions. The advantage of such constraints is that the step size of the numerical integration may be increased without sacrificing accuracy as these terms have the largest gradients in macromolecules simulated at physiological temperatures. We use the SHAKE algorithm for applying the constraints, see Section 11.6 [Shake Command], page 107. SHAKE can be applied to just the bonds involved with hydrogens, all bonds, all bonds and the angles involving hydrogens, or all bonds and angles.

A dynamics run has basically four parts; initialization, heating, equilibration, and the simulation itself. Initialization means providing an initial position and velocity for all the atoms. Heating is the process of increasing the kinetic energy of the system up to a final temperature at which the simulation will be conducted. Equilibration is the process where the kinetic energy and the potential energy of the system evenly distribute themselves throughout the system. Only when the average temperature of the system stabilizes can one collect the trajectory information for analysis.

The initial coordinates of a simulation are obtained after applying the minimization algorithm to a complete coordinate set. One cannot start with a system with a large potential energy as it will quickly heat up to unreasonable temperatures. For initializing the velocities, the user can specify zero velocity, a uniform distribution of kinetic energy along each coordinate with random sign of the motion along each axis (`IASORS` 0) or a Gaussian distribution of velocities (`IASORS` 1 the default). The temperature at which velocities are assigned is determined by `FIRSTT` and `TSTRUC` by the algorithm:

Tassign = 2*(FIRSTT-TSTRUC) + TSTRUC.

For a harmonic system equilibrated to `TSTRUC` equal partition of the energy will result in an equilibrated temperature of roughly `FIRSTT`. If `TSTRUC` is not specified 1.25*`FIRSTT` will be used for assignment.

The heating of a system is performed gently by increasing the kinetic energy by a small amount periodically. The number of integration steps between heating applications, the final temperature, and the kinetic energy increment are all user specified. In addition, there is a choice in the method of increasing the kinetic energy of the system. One may scale existing velocities or reassign them. The velocities can be scaled by either one scale factor calculated for the kinetic energy of the system averaged over many time steps or by scale factors established for each atom based on the ratio of its time averaged kinetic energy with that of the system. If reassignment is chosen, the velocities can have either a uniform or Gaussian distribution.

To equilibrate the structure, one can specify a window around the final temperature where velocity adjustments will be made. The choice of velocity adjustments is the same as described above for heating.

For the actual run, CONGEN will output the position and velocities of all atoms at intervals specified by the user. The temperature window can be set larger so that any gross conformational changes which result in a different potential energy will cause the temperature to be maintained.

At any time energy is added to the system, the angular momentum of the system will be reduced to zero and translational motion will be stopped. One can also request that these operations be performed at any time during the dynamics run.

When dynamics is used for simulated annealing, it is useful to use the `TLIMIT` option. This option applies a velocity damping to any atom whose temperature exceeds the limit. It is especially useful for solving structures using NMR constraints, see Section 11.5 [NMR Constraints], page 97. Consider the case of two atoms which are supposed to be close in space, but are not at the beginning of the simulation. When they approach, they will both acquire substantial kinetic energy as they travel down the potential well. With the `TLIMIT` option, their velocity will be reduced so that they will not accelerate to such high speeds that the numerical integration of their motion will fail because the step size is too large. In addition, the velocity damping will help to hold them in position. You can see details of the damping process by turning on the `TLIMIT` debugging variable, see Section 27.11 [Debug Command], page 230.

The use of a restart file is essential for running dynamics. Since running dynamics requires storing various derivatives of the position with respect to time, this information must be stored for the life of the dynamics run. This capability is provided by the restart file. When the run is initiated, a restart file must be written using the `IUNWRI` keyword. As the dynamics routine complete `NCYCLE` steps of dynamics, the Fortran unit specified by `IUNWRI` will be rewound and a restart file will be written. In case of crashes, one has restart files corresponding to various points in the run. The CRASHU variable may prove valuable. Successive runs of CONGEN to continue the dynamics run must read the previous restart file using the `IUNREA` keyword and write it out for the next part of the run. See Section 7.7 [Dynamics Outputs], page 67, for a description of these variables.

There are many numbers giving the frequency of actions to be taken during dynamics such as updating the non-bonded list, heating the molecule etc. Some of these numbers are adjusted along with the number of steps to run so that numbers all have a common divisor. At the present time, there are combinations which result in errors. At some point an attempt may be made to catalog all the actions, and check for erroneous processing.

If one is interested in simulating the motion of part of the system with the rest of the system remaining fixed, it is possible to fix atoms in place. See Section 11.4 [Fixed Atoms], page 97, for more information. If this is done, there are several effect on the dynamics. First, since the system is now anchored in space, the center of mass motion and total angular velocity is never stopped. Second, the number of degrees of freedom used for calculating the temperature is set to the number of free atoms times 3 minus 6. Third, the coordinate and velocity trajectory files will contain the position of the fixed atoms only once, and all other records will hold just the moving atoms. This saves a great deal of disk space.

The trajectory produced by the dynamics procedure can be analyzed in great detail using the analysis facility (see Chapter 16 [Analysis], page 157). In addition, trajectory files can be merged, broken in smaller pieces, and sampled at different intervals. See Section 7.8 [Manipulating Trajectories], page 67, for details. Likewise, said operations can be performed on coordinate trajectories while rotating the coordinates to match a given coordinate set. See Section 18.4 [Reorienting a Coordinate Trajectory], page 178, for details.

In the table of keywords which follow, one can select on keyword from a set of keywords enclosed in square brackets and such keywords take no values after them. All other keywords must be specified with a value (see Section 7.1 [Energy Manipulation Syntax], page 55). See Section 7.4

[General Energy Operands], page 58, for other variables which apply. See Section 2.4 [AKMA], page 7, for a description of the AKMA units system.

```
Keyword  Default  Purpose

[ VERL ] VERL     Verlet algorithm is used for integration in dynamics.
[ GEAR ]          Gear ( 6 vector ) algorithm is used for integration.
                  If SHAKE is used, GEAR option is overridden and Verlet
                  algorithm is used.

[ STRT ] STRT     The dynamics is assumed to start from the input
[      ]          coordinates using an assignment of velocities given by
[      ]          IASVEL. No restart file is read.
[ REST ]          The dynamics is restarted by reading the restart file
                  from unit IUNREA.

AKMASTP  .02      Time step for Dynamics in AKMA units. The AKMASTP
TIMESTP           keyword is used to enter a step size in AKMA units.
                  TIMESTP is used for picoseconds. The default value is
                  0.02 AKMA units (0.000977 picoseconds).

TOL      1.0E-6   Shake tolerance, i.e. the maximum relative error allowed
                  in the constraining of a SHAKEn bond length or bond angle.

IUNREA   -1       Fortran unit from which the dynamics restart file should
                  be read. A value of -1 means don't read any file

IUNWRI   -1       Fortran unit on which the dynamics restart file for
                  the present run is to be written. A value of -1 means
                  don't read any file.

IUNCRD   -1       Fortran unit on which the coordinates of the dynamics run
                  are to be saved. A value of -1 means no coordinates should
                  be written. Unformatted output.

IUNVEL   -1       Fortran unit on which the velocities of the dynamics run
                  are to be saved. -1 means don't write. Unformatted output.

KUNIT    -1       Fortran unit on which the total energy and some of its
                  components along with the temperature during the run are
                  written using formatted output.

CRASHU   -1       Fortran unit where a single DCL command file will be
                  written. If the machine crashes before a restart file
                  is written, this file won't be touched. If the crash
                  occurs after a restart is written but before the run
                  completes, this file will contain the line, "$ @CRASH".
                  If the run completes, the file will contain
                  the line, "$ @COMPLET". This allows for an automatic
                  recovery system after crashes.

NSAVC    5        The step frequency for writing coordinates.
```

NSAVV	5	The step frequency for writing velocities.
NPRINT	1	The step frequency for storing on KUNIT as well as printing on unit 6, the total energy data of the dynamics run.
IPRFRQ	50	The step frequency for calculating averages and rms fluctuations of the major energy values. If this number is less than NTRFRQ and NTRFRQ is not equal to 0, square root of negative number errors will occur.
IHTFRQ	0	The step frequency for heating the molecule in increments of TEMINC degrees in the heating portion of a dynamics run. Zero means do no heating.
IEQFRQ	0	The step frequency for assigning or scaling velocities to FINALT temperature during the equilibration stage of the dynamics run.
NTRFRQ	0	The step frequency for stopping the rotation and translation of the molecule during dynamics. This operation is done automatically after any heating.
FIRSTT	0.0	The initial temperature at which the velocities have to be assigned at to begin the dynamics run. Important only for the initial stage of a dynamics run.
FINALT	298.0	The desired final (equilibrium) temperature for the system. Important for all stages except initiation.
TEMINC	5.0	The temperature increment to be given to the system every IHTFRQ steps. Important in the heating stage.
TSTRUC	-999.	The temperature at which the starting structure has been equilibrated. Used to assign velocities so that equal partition of energy will yield the correct equilibrated temperature. -999. is a default which causes the program to assign velocities at T=1.25*FIRSTT.
TWINDH	5.0	The temperature deviation from FINALT to be allowed on the high temperature side.(+ve). i.e. high side of the temperature window. Useful during equilibration.
TWINDL	-5.0	The temperature deviation from FINALT to be allowed on the low temperature side.(-ve). i.e. low side of the temperature window. Useful during equilibration. *This number must specified as a negative number to be meaningful.*
TLIMIT	0.0	The temperature limit for atoms. When this option is positive, CONGEN will compute the velocity that corresponds to this temperature for all atoms. After

		every dynamics time step, the new velocity for each atom will be checked against this limit. If it is exceeded, the atom's velocity will be lowered to the limit, and a new corresponding position will be calculated. The use of this option can result in a loss of energy in the system. The limit should be set above the desired temperature of the system, but CONGEN makes no check to see that the limit is reasonable. Also, this option only works with Verlet integration. It will also work with the SHAKE algorithm.
IASORS	0	The option for scaling or assigning of velocities during heating (every IHTFRQ steps) or equilibration (every IEQFRQ steps). .eq. 0 - scale velocities. .ne. 0 - assign velocities.
IASVEL	1	The option for different assignments of velocities. .eq. 0 - zero velocity assignment .gt. 0 - gaussian distribution of velocity. (+ve) .lt. 0 - uniform distribution of velocity. (-ve) kinetic energy of 3N velocity components are same.
ISEED	314159	The seed for the random number generator used for assigning velocities.
ISCVEL	0	The option for two ways of scaling velocities. .eq. 0 - single scale factor for all atoms .ne. 0 - a scale factor for each atom proportional to the kinetic energy average ratio between the system and along every degree of freedom for that atom.
ICHECW	1	The option for checking to see if the average temperature of the system lies within the allotted temperature window (between FINALT+TWINDH and FINALT+TWINDL) every IEQFRQ steps. .eq. 0 - do not check i.e. assign or scale velocities. .ne. 0 - check window i.e. assign or scale velocities only if average temperature lies outside the window.
ISCALE	0	This option is to allow the user to scale the velocities by a factor SCALE at the beginning of a restart run. This may be useful in changing the desired temperature. .eq. 0 no scaling done (usual input value) .ne. 0 scale velocities by SCALE. WARNING: Please use this option only when you are changing the temperature of the run.
SCALE	1.0	Scale factor for the previous option.

```
ETETEST  20.0     This variable is used for the total energy conservation
                  test in dynamics. If the total energy varies by more
                  than this amount and EKETEST multiplied by the kinetic
                  energy, the run will be terminated. This check is turned
                  off if TLIMIT is set.

EKETEST   0.1     See ETETEST above.
```

7.7 Dynamics Output

The first part of CONGEN's output after a dynamics command lists all of the options that apply to that part of the run. Then, any information about velocity assignments (temperature changes) follows. Any time the velocities are changed in an anisotropic way, the motion of and about the center of mass will be stopped. This results in a printout both before and after this operation of the '`DETAILS ABOUT CENTRE OF MASS`'. Its position and velocity are output followed by the components of the angular momentum. The last line gives the translation kinetic energy of the system, and thus one should expect a drop in the total energy and temperature of the system afterwards.

Non-bonded interaction and hydrogen bond updates will appear intermittently and are cleared labeled.

Every `NPRINT` steps, the total energy and various contributions will be printed. This output is preceded by a title which gives the correspondence of numbers to energy names. After `IPRFRQ` steps will appear the averages and RMS fluctuations. After the second such printout of averages and RMS fluctuations, the averages and RMS fluctuations for the run up to the last turning of the molecule will be given. This gives you longer range statistics. Such a calculation will not be done if `IPRFRQ` equals `NTRFRQ`. The ratio of total energy to kinetic energy fluctuations is an excellent measure of the accuracy of the run.

After the averages are printed, a least squares fit of the total energy against the step number will be made to look for drift in the energy. Two such values are printed, one for the last `IPRFRQ` steps, and one to the previous turn. Next, the initial energy for the statistics, both short range and long, are printed. Finally, the correlation coefficient of the energy versus step is given for both ranges. A value close to zero indicates no systematic drift; a magnitude near 1 means you have a real problem with the dynamics.

This process of printout continues until the end of the run is reached. Just before the last energy is printed will appear a message about the writing of coordinates and velocities to their respective files.

7.8 Manipulating Trajectories

The trajectories produced from a dynamics calculation generally require some sort of additional processing to make them easier to analyze. Currently, two such commands are provided, `MERGE` and `ROTATE`. `MERGE` is described below, and `ROTATE` is described under the `COMPARE` command, see Section 18.4 [Reorienting a Coordinate Trajectory], page 178.

Frequently, one generates a trajectory into small files to minimize the CPU time of one job. However, so many files are usually hard to manage so it is desirable to merge said files into larger units. This command provides that capacity. In addition, it is possible to break up the trajectory into smaller pieces and to sample the trajectory less frequently than originally generated.

7.8.1 Merge Trajectory Syntax

```
MERGE DYN [ COOR ] [FIRSTU unit-number] [NUNIT integer] [SKIP integer]
```

```
                    [ VEL  ]       [OUTPUTU unit-number] [NFILE integer]
```

7.8.2 Merge Trajectory Options

Option	Default	Purpose
[COOR] [VEL]	COOR	Specification of the type of trajectory file. COOR is coordinates; VEL is velocities.
FIRSTU	51	The first unit of the trajectory to be read.
NUNIT	1	The number of units to be read starting with FIRSTU
SKIP	1	Only those coordinate whose dynamics step number modulo SKIP will be reoriented and written out.
OUTPUTU	61	The first unit number of the output trajectory
NFILE		The number of coordinates written to each output file. If left out, this will be set to the number of coordinates in the first input file times the number of input files. WARNING: This default will generate a bad trajectory file if SKIP is not set to the interval actually present in the trajectories. Further, if you set its value to be larger than the number of coordinates that are actually written in any output file, you will have problems. The error that is generated results from the control array in the beginning specifying that there are more coordinates than actually exist in the file. EOF errors will result when the trajectory is read.

The title of the output trajectory will be copied from the input trajectory.

8 Poisson-Boltzmann Electrostatics

The Poisson-Boltzmann equation (PBE) provides a “fast” and approximate method for calculating the effects of solvent on electrostatic interactions in the modeled system. The current solution of the PBE in CONGEN has one unique feature, it can spread the charge of each atom over the van der Waals volume which results in superior convergence properties and makes the calculation less dependent on the position of the system with respect to the grid[1]. In addition, CONGEN can display the partition of the Poisson-Boltzmann energy across all the atoms in the system, see the `EPBE` property described in Section 17.1.3.3 [Atom Properties], page 160.

The implementation provides two capabilities. First, one can perform individual electrostatic calculations on any set of atoms within the system. Second, one can replace the Coulomb electrostatic energy with the Poisson-Boltzmann energy in the context of a conformational search, see Section 12.4.5 [Poisson-Boltzmann Options], page 128. We hope to increase the applicability of this methodology in the future.

8.1 Introduction to Poisson-Boltzmann Electrostatics

One of the most successful implicit models for the treatment of electrostatic effects is the Poisson-Boltzmann equation[2] which is given below

$$\nabla \cdot (\epsilon(\mathbf{x})\nabla\phi(\mathbf{x})) - \bar{\kappa}^2(\mathbf{x})\mathrm{sinh}[\phi(\mathbf{x})e/kT] = -4\pi k_e\rho(\mathbf{x})$$

where ϕ is the electrostatic potential, ρ is the charge density, $\epsilon(\mathbf{x})$ is the dielectric constant in units where the vacuum dielectric is exactly 1, k_e is the electrostatic force constant, e is the charge on the electron, k is the Boltzmann constant, T is the temperature, and $\bar{\kappa}$ is a dielectric independent Debye-Huckel parameter,

$$\bar{\kappa} = \sqrt{\epsilon}\kappa$$

The equation can be solved over a volume enclosing a molecule of interest using finite difference methods[3][4][5] appropriate for the solution of boundary value problems. In these methods, a grid is laid down over the volume, and each grid point is assigned a value for the ionic strength and charge density. The lines between each grid point are assigned values for dielectric constant. Boundary values for the potential are set according to a variety of models and then, interior values for the electrostatic potential are calculated. Typically, the equation is linearized by substituting x for sinh(x) in the above equation.

The implementation of the finite difference solution of the Poisson-Boltzmann equation in CONGEN is written in C and uses dynamic storage allocation throughout so any size grid can be accommodated. The process of a potential calculation begins with the determination of the grid position. This can be centered on the spatial origin, the center of a rectangular box enclosing the molecule, or

[1] R. E. Bruccoleri, “Grid Positioning Independence and the Reduction of Self-Energy in the Solution of the Poisson-Boltzmann Equation”, *J. Comput. Chem.* **14**, 1417-1422 (1993).

[2] Stephen C. Harvey, “Treatment of Electrostatic Effects in Macromolecular Modeling”, *Proteins: Struct. Funct. Gen.* **5**, 78-92 (1989)

[3] J. Warwicker and H. C. Watson, “Calculation of the Electric Potential in the Active Site Cleft Due to alpha-Helix Dipoles”, *J. Mol. Biol.* **157**, 671-679, (1982).

[4] Malcolm E. Davis and J. Andrew McCammon, “Solving the Finite Difference Linearized Poisson-Boltzmann Equation: A Comparison of Relaxation and Conjugate Gradient Methods”, *J. Comput. Chem*, **10**, 386-391, (1989).

[5] Anthony Nicholls and Barry Honig, “A Rapid Finite Difference Algorithm, Utilizing Successive Over-Relaxation to Solve the Poisson-Boltzmann Equation”, *J. Comput. Chem* **12**, 435-445, (1991).

the geometric center of the molecule. Next, the dielectric constant stored on all of the grid edges is set to the solvent value, and the Debye-Huckel parameters are likewise set. Next, the program loops over all atoms. Grid points of the protein excluded space are assigned grid charges as described above, their dielectric constants are set to interior values, and the Debye-Huckel parameters are set to zero. The van der Waals, accessible, or molecular surface can be used to delimit the interior. Upon the direction of the user, the program can set the boundary using one of three possible rules. The first possibility is to set it to zero. This is very fast, but is only appropriate when the boundary is very far from the atoms in the system. The second possibility, which is denoted as the system Debye rule, is to set it to the potential caused by a single charge equal to the total charge of the system and whose radius is equal to that of a sphere whose volume equals the solvent excluded volume of the system. This method is also efficient, but it can capture some of the electrostatic screening due to the solvent if there is a thick solvent boundary around the system. The final boundary setting method, which is denoted as the atomic Debye rule, sets the boundary points to the sum of the potentials due to all of the atoms scaled by the Debye-Huckel rule,[6] The atomic Debye rule is the most accurate.

$$\phi = \frac{k_e q_i e^{-\kappa r_s}}{\epsilon_s r(1+\kappa R)}$$

where r is the distance from boundary point to the atom, R is the radius of the atom plus the water radius plus the length of the ion exclusion (Stern) layer and r_s is $r - R$.

This last option does a more detailed calculation of the boundary, but is more expensive computationally.

The potential is calculated using Gauss-Seidel iteration with odd-even ordering of processing grid points.[7] When the linearized form of the equation is solved, overrelaxation is used. The default spectral radius used for the overrelaxation parameter is given by

$$\rho = 1/3\left(\cos\frac{\pi}{n_x} + \cos\frac{\pi}{n_y} + \cos\frac{\pi}{n_z}\right)$$

where n_x, n_y, and n_z are the grid dimensions in the x, y, and z directions.

8.2 PBE Data Structures

In order to understand the function of the commands which use the Poisson-Boltzmann equation, it is important to know the underlying data structures.

The primary data structure is the `pbe_info` data structure, defined in the file, '`$CGS/pbe.h`'. This data structures contains the grids for the potential, charge density, dielectric constant, and Debye-Huckel parameter, $\bar{\kappa}$. There are three grids stored for the dielectric constant because values for the dielectric constant are stored for the lines connecting grid points. Since there are three perpendicular lines connecting grid points to their neighbors, there are three dielectric constant grids, for the x, y, z directions. The `epsx` grid stores the dielectric constant for the midpoint of the line pointing in the x direction between grid points. The value of `epsx[i,j,k]` corresponds to the dielectric constants between grid points `[i,j,k]` and `[i+1,j,k]`. The value of `epsy[i,j,k]` corresponds to the dielectric constants between grid points `[i,j,k]` and `[i,j+1,k]`, and the value

[6] Malcolm E. Davis and J. Andrew McCammon, "Electrostatics in Biomolecular Structure and Dynamics", *Chem. Rev.* **90** 509-521 (1990).

[7] W. H. Press and B. P. Flannery and S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1986, pp. 652-659.

of `epsz` is analagous for the the third indices, respectively. One two-dimensional cross section of each grid is unused.

The `pbe_info` data structure also contains the spatial origin of the grid, dimensions, and grid size so that the grid can be laid out over the space of the molecule of interest. It also contains a list of atoms included in the grid as well as their positions, radii, and charges. Finally, it contains a number of parameters which control the calculation.

The `PBE SETUP` command is used to setup this data structure. After the first `PBE SETUP`, the `REUSE` option controls whether the data structure is initialized or updated, see Section 8.3.1 [PBE SETUP Command], page 71.

It is important to remember that this data structure is not automatically updated when the coordinates change (except in the context of a conformational search).

It is possible to generate multiple `pbe_info` data structures, and to do operations on them. This is useful for comparing results generated by different methods or parameters.

8.3 PBE Commands

All the Poisson-Boltzmann commands begin with the keyword, `PBE`. The second word on each command line specifies one of the command below. In brief, the commands perform the following operations.

`SETUP`	Setup all the grids and other parameters needed for the calculation of the potential.
`POTENTIAL`	Calculate the electrostatic potential.
`ENERGY`	Calculate the electrostatic energy of the system.
`READ`	Test `WRITE` command (Not very useful).
`WRITE`	Write one of the grids to a file.
`TEST`	Setup test potentials.
`SAVE`	Save a copy of the `pbe_info` data structure under a variable name.
`RECALL`	Recall a `pbe_info` data structure into the main data structure.
`DIFF`	Compute a difference in potentials.
`DESTROY`	Delete a saved `pbe_info` data structure.
`STATISTICS`	Compute simple statistics on the grids.
`MASK`	Change values in one of the grids based on a masking operation.

8.3.1 PBE SETUP Command

Syntax

```
PBE SETUP repeat(pbe-setup-options) atom-selection
```

```
                      [SOLVent real]
                      [interior-options]
                      [charge-options]
                      [NWARN int]
                      [boundary-options]
                      [IONStr real]
                      [GRID real]
                      [SUBDivisions int]
                      [SPILl]
                      [TEMPerature real]
pbe-setup-options ::= [WATEr real]
                      [STERn real]
                      [XDIM int]
                      [YDIM int]
                      [ZDIM int]
                      [MARGin int]
                      [REUSE]
                      [OLDGRID]
                      [CENTER]
                      [MOLSURF]
                      [MINRadius real]
                      [charge-edit-options]
                      [EPSAve averaging-option]
                      [SMOOTH repeat(smooth-option) END]

interior-options ::= repeat([INTErior real exposure-opts atom-selection END])

exposure-opts ::= [ABSQ real] [EXPOsure real] [RESAve]

charge-options ::= [UNIForm  ]
                   [TRILinear]

                              [ZERO    ]
boundary-options ::= BOUNdary [ADEBye  ]
                              [SDEBye  ]
                              [PREVious]

charge-edit-options ::= repeat([CHARge edit-type real atom-selection END])

              [SET]
edit-type ::= [SCALe]
              [SHIFt]

averaging-option ::= [HARMonic  ]
                     [ARIThmetic]

                   [TYPE type-option]
                   [WEIGht weight-option]
smooth-options ::= [OFFGrid offgrid-option]
                   [RADIus real]
                   [POINts int]
                   [ALPHa real]
```

```
                [NONE  ]
type-option ::= [VOLUME]
                [GRID  ]

weight-option ::= [CONStant]
                  [GAUSsian]

offgrid-option ::= [SOLVent]
                   [EDGE   ]
```

Function

The PBE SETUP command creates the pbe_info data structure, see Section 8.2 [PBE Data Structures], page 70, which stores the grids upon which the electrostatic potential is computed. The command has a large number of options and an atom-selection, see Section 11.7 [Atom Selection], page 108. The atom selection allows the user to select any portion of the system for the calculation. For example, if one is interested in the calculation of the binding energy of a complex, the atom selection can be used to select each component and then all the atoms for separate electrostatic calculations. The options are described in the following table.

Option	Purpose
SOLVENT	Specifies the dielectric constant of the solvent in units of the vacuum dielectric. Thus, the vacuum dielectric would be 1, and the dielectric constant of water is around 78. The default is 78.
interior-options	The INTERIOR option is used to specify the dielectric constant of the interior of the atom selection in the INTERIOR option. Any number of these options can be specified, and thus, it is possible to specify different dielectric constants for different parts of the system. It is possible to adjust the dielectric constant of exposed atoms to that of the solvent[8]. If the either option, ABSQ or EXPOSURE, is specified, then the exposure adjustment calculation will be done. Any atom whose exposure and charge meet the criteria will have the dielectric constant set to the solvent value as specified by the SOLVENT keyword. N.B. This option is highly experimentally and has not been proven useful in any system. The meaning of the keywords is given as follows:

ABSQ	The magnitude of the charge of the atom must be greater than or equal to this option in order for surface charge adjustment to be performed.
EXPOSURE	The relative molecular surface exposure of the atom for the adjustment is specified by this parameter. If the relative molecular surface exposure is less than EXPOSURE - 0.1, then the dielectric constant will be left alone. If the relative molecular surface exposure is greater than EXPOSURE + 0.1, then the dielectric constant will be set to the solvent value. Otherwise, it will be scaled harmonically between the interior value and solvent value.
RESAVE	The average relative molecular surface is calculated for all the atoms within a residue, and the exposure test above and the scaling is applied for the average.

[8] This is suggested by a paper by T. Simonson and D. Perahia, "Internal and interfacial dielectric properties of cytochrome *c* from molecular dynamics in aqueous solution. *Proc. Natl. Acad. Sci. USA* **92**, 1082-1086 (1995).

`charge-options`
: The charging options, `UNIFORM` and `TRILINEAR`, along with the options `SUBDIVISIONS` and `SPILL`, control how the charge grid is set. With the `TRILINEAR` option, the charge of each atom is divided among the eight nearest points to the atom center. With the `UNIFORM` option, the charge of each atom is divided evenly among all the points within the van der Waals radii of the atom, except in the case where the number of such points is less than eight. In that case, trilinear interpolation is used. The option, `NWARN`, controls how many warning of this conversion are displayed.

  The uniform charging option can be further improved using the `SUBDIVISIONS` and `SPILL` options. One problem with the use of grid based methods to represent molecular structure is the quantization of space. The same problem has arisen in the development of computer graphics using raster devices, and using the techniques of anti-aliasing, CONGEN can smooth the charge distribution of atoms. When the `SUBDIVISIONS` option is used, the space around each atom is subdivided into $\texttt{SUBDIVISIONS}^3$ virtual cubes, and the charge distribution is calculated over these virtual cubes, and then added onto the real cubes in the grid. If the `SPILL` option is turned on (recommended!), then charge can spill onto cubes outside the van der Waals radius of each atom. If `SPILL` is not specified, then the charge will not extend beyond the van der Waals radius.

`NWARN`
: The `NWARN` option specifies the maximum number of warning messages to be issued if an atom is too small for the uniform charging scheme to charge eight points. The default is 5.

`boundary-options`
: The `boundary-options` are used to specify how the potential is set for the boundary of the grid. The `ZERO` option specifies that the boundary potential should be zero. The `ADEBYE` option specifies that the atomic Debye method should be used, see Section 8.1 [Introduction to Poisson-Boltzmann Electrostatics], page 69. This option is very slow. The `SDEBYE` options specifies that the system Debye method should be used. The keyword, `PREVIOUS`, is not implemented.

`IONSTR`
: The `IONSTR` parameter is used to set the ionic strength. The units are in molarity. The default value is 0.0.

`GRID`
: The `GRID` option sets the physical spacing between grid points. This is a critical parameter in the calculation of electrostatics using the Poisson-Boltzmann equation. The smaller the grid, the better the accuracy, but computation time scales as the grid size to the 4th power. The default value is 1.25 Angstroms.

`TEMPerature`
: The `TEMPERATURE` option is used to set the temperature for calculating the Debye-Huckel parameter, κ. The default value is 300 K.

`WATER`
: The radius of water in Angstroms is set using the `WATER` option. The radius of water is added to the van der Waals radii of all atoms and the dielectric constant of the midpoints of grid lines within this combined radius is set to the interior dielectric value for the atoms. Use of the `WATER` option causes the program to use the solvent accessible surface to define the interior. The default value is 0.0. Our current view is that this parameter should be left at 0.0 because ion solvation energies are calculated more accurately that way.

`STERN`
: The thickness of the ion exclusion layer (Stern layer) is set to the `STERN` option. Within this distance of any atom, it is assumed that ion screening does not take place, and the Debye-Huckel parameter is set to zero. The default value is 2.0 Angstroms.

`CENTER` If the `MARGIN` option is non-negative, this option has no effect. Otherwise, when the `CENTER` option is specified, the center of grid is placed at the geometric center of the system. Otherwise, the center of grid is placed at the origin.

`XDIM` The `XDIM` option sets the number of grid points in the X direction.

`YDIM` The `YDIM` option sets the number of grid points in the Y direction.

`ZDIM` The `ZDIM` option sets the number of grid points in the Z direction.

`MARGIN` The `MARGIN` option sets the number of empty grid points surrounding the molecule. If set to a non-negative number, then CONGEN determines a rectangular box that will surround the molecule including the van der Waals radius, plus two grid spacings if the `SPILL` option is on. Then, the dimensions of the box will be increased in order to center this box within the grid with `MARGIN` grid points around all sides. When the `MARGIN` option is non-negative, the `CENTER` option has no effect.

`MINRADIUS`

The `MINRADIUS` keyword allows the user to specify the minimum radius for any atom placed on the grid. This keyword is important for potentials, such as AMBER94, see Section B.1.13 [AMBER94PARM], page 306, which have several atoms with zero or small radius. Because the electrostatic self energy of a sphere is inversely proportional to the radius, such small atoms cause convergence problems. Therefore, one can set the minimum radius using the above option.

`REUSE`
`OLDGRID` The `REUSE` option specifies that the current `PBE SETUP` command should update rather than initialize the `pbe_info` data structure. For the keyword options described here, the default values are taken from the current values rather than the values described in this documentation. The `OLDGRID` option is similar, except it maintains only the grid, but not the values within

These options are important for doing binding energy calculations. In order to cancel errors due to grid positioning, it is essential to first calculate the energy of the complex, and then, using the `OLDGRID` option, calculate the energies of the components by selecting the atoms of the components, but leaving the atoms and grids in the same spatial positions.

When this option is used, you should not specify any grid dimensions or the `MARGIN` option. If you do, and they are different than the actual values, then the `REUSE` or `OLDGRID` option will be turned off.

`MOLSURF` This option specifies that the interior of the molecule is defined by the molecular surface as computed by the GEPOL algorithm, see Section 17.1.3.3 [Atom Properties], page 160, for more information about GEPOL. The dielectric constant of points covered by the molecular surface but not atomic spheres is determined by the following scheme: GEPOL generates additional spheres which define the molecular surface. Each of these spheres derives from one or two previous spheres or atoms. The dielectric constant of each sphere is set to the average of its “parents”, where the average is determined by the `EPSAVE` keyword. Then, points covered by each of these spheres is averaged into the dielectric grid.

`EPSAVE` Whenever dielectric constants are combined, the `EPSAVE` controls how the averaging is performed. The averaging can either be `ARITHMETIC` (the "standard" mean) or `HARMONIC` (the reciprocal of the mean of the reciprocals). This option applies to averaging dielectric constants when atoms overlap in the grid, for dielectric smoothing, and for generating the dielectric constant for spheres generated by the molecular surface algorithm.

`charge-edit-options`
: The `charge-edit-options` allows the user to modify the charges used for a calculation. Charges for the atoms in the `atom-selection` can be changed in one of three ways. The `SET` option specifies the charge explicitly. The `SCALE` option specifies a multiplicative scale factor to be applied to the current charges of the selected atoms. The `SHIFT` option specifies an additive term to be added to the current charges of the selected atoms.

`SMOOTH`
: The `SMOOTH` command "smooths" the dielectric grids. After the dielectric grids have been assigned internal and external values "smoothing" averages each point with the other points in its neighborhood to produce a dielectric grid with less abrupt changes in value. One hope of the process is to reduce the position dependence of the PBE electrostatic calculations. Its success in this endeavor is still a topic of research.

Several types of smoothing are supported. The smoothing `TYPE` can either be `NONE`, `VOLUME`, or `GRID`. Of course, no smoothing is done when `NONE` (the default) is specified. For `VOLUME` based smoothing the dielectric is averaged over a fixed volume in real space defined by the `RADIUS` keyword. For `GRID` based smoothing the dielectric is averaged over a set number of points specified by the `POINTS` keyword (9 and 15 point averaging are currently supported). In both case all three dielectric grids are averaged togeher. In 9-point averaging each point is averaged with the eight grid points from the other two grids which surround this point in space. In 15-point averaging the six nearest neighbors from the same grid are also included. (Note that volume smoothing with a radius just smaller (larger) than the grid spacing is equivalent to 9-point (15-point) averaging.)

Which ever `TYPE` is selected, The type of averaging is determined by the `EPSAVE` keyword above. The `WEIGHTING` can either be `CONSTANT` (all points in average counted equally), or `GAUSSIAN` weighted (points are weighted by $\exp(-\alpha^2 r^2)$ where r is the distance from the central point in units of the grid dimension and α is specified by the `ALPHA` keyword).

For points at the edge of the grid the averaging extends to points beyond the grid. Using the `OFFGRID` keyword, these offgrid points can either be assumed to be `SOLVENT` or equal to the value at the nearest `EDGE` of the grid. (Note that the `SOLVENT` assumption will run faster.)

8.3.2 PBE POTENTIAL Command

Syntax

```
PBE POTENTIAL repeat(pbe-potential-options)

                           [NODIelectric]
                           [TOLerance real]
                           [RTOLerance real]
pbe-potential-options ::= [MAXITS int]
                           [HISTsize int]
                           [RADIUS real]
                           [problem-option]

problem-option ::= {LINEar   }
                   {NONLinear}
```

Function

The `PBE POTENTIAL` command invokes the Gauss-Siedel PBE solver, and the above options affects how the solution is performed. If the `PARALLEL LOOPS` command is used, see Section 27.1 [Parallel Command], page 227, the potential will be calculated in parallel across available CPU's on a Silicon Graphic multiprocessor workstation. Parallel efficiency is high. N.B., for large memory jobs running on SGI workstations, it is critically important to set the `STACK` resource limit down from the default value. See Section 27.16 [Rlimit Command], page 241, for more information.

After the potential is calculated, the PBE energy per atom is calculated and stored in an array for this purpose.

The options have the following meaning:

Option Purpose

`NODIELECTRIC`
: If this option is specified, the potential is calculated assuming the dielectric is equal to the solvent value, keyword `SOLVENT` in Section 8.3.1 [PBE SETUP Command], page 71. The Poisson-Boltzmann equation is not used, but rather the potential contribution from each atom is summed onto each grid point. The boundary is also modified.

`TOLERANCE`
: If the maximum change in any potential value during a cycle over all the grid points is less than the `TOLERANCE` option, then the solver terminates. Values smaller than 10^-6 are generally too small to be achieved. The default setting is 0.001.

`RTOLERANCE`
: If the RMS of the changes in potential value during a cycle over all the grid points (maximum residuals) is less than this option, the solver terminates. This option is less restrictive than the `TOLERANCE` option above, and therefore, terminates sooner. The maximum change is typically an order of magnitude bigger than the RMS change. The default setting is 0.0, in which case, the test is not used.

`MAXITS`
: The specifies the maximum number of iterations allowed for the solver. Because of the odd-even checkboard cycling, two iterations are necessary to cover all the grid points. The default is 100.

`HISTSIZE`
: When the PBE solver is working toward a small tolerance, it is possible for a potentially infinite loop to be entered. In order to detect this, CONGEN maintains an array of maximum residuals for twice the number of cycles given by `HISTSIZE`. At the end of each iteration over all the atoms, the minimum value for the maximum residual for the last `HISTSIZE` cycles is compared against the minimum for the previous `HISTSIZE` cycles. If it has not improved, then the PBE solver terminates. Effectively, this mechanism checks for any improvement in the solution of `HISTSIZE` cycles. The default value is 30.

`RADIUS`
: Overrides the default setting, see Section 8.1 [Introduction to Poisson-Boltzmann Electrostatics], page 69, for the spectral radius. If this value is set to zero, then overrelaxation is not used. If the value is negative, then the absolute value is used as the overrelaxation parameter.

`LINEAR`
`NONLINEAR`

These two options control whether the linearized form of the Poisson-Boltzmann equation is solved. Generally, only the linearized form should be used. The non-linear form has a modest radius of convergence, and large charges will result in divergence of the solver. When the non-linear form is used, execution time is much slower, and overrelaxation is turned off. The default is `LINEAR`.

8.3.3 PBE ENERGY Command

Syntax

```
PBE ENERGY [ATOM]
```

Function

This command calculates the electrostatic energy of the system using the formula,

$$E = 1/2 \sum_{ijk} \rho_{ijk} \phi_{ijk}$$

If the option, `ATOM`, is specified, the PBE energy per atom will be printed. It is usually more convenient to get these energies using the `ANALYSIS` commands, see Section 17.1.3.3 [Atom Properties], page 160.

8.3.4 PBE READ Command

Syntax

```
PBE READ UNIT unit
```

Function

This command reads a Poisson-Boltzmann data structure which has been written to a disk by the `PBE WRITE` command, see Section 8.3.5 [PBE WRITE Command], page 78. The only option, which is required, is the unit number of the file containing the data structure.

8.3.5 PBE WRITE Command

Syntax

```
PBE WRITE [UNIT unit] [TITLE string del] { ALL   }
                                         { PHI   }
                                         { RHO   }
                                         { EPSX  }
                                         { EPSY  }
                                         { EPSZ  }
                                         { KAPPA }

          [X range] [Y range] [Z range] [LINEsz int] [CARD]
                                                     [FILE]

range ::= [[int]:[int]]
```

Function

This command is used to write the entire Poisson-Boltzmann data structure to a binary file, or it can be used to write one of the grids to an output file in either binary or text format. The binary format for the grids is compatible with the PLT2 program, so that contour maps of sections of potentials or other grids can be made.

N.B. the behavior of this command changes depending on whether `ALL` is specified, so please take note of these confusing changes as described below.

The options have the following meaning:

Option	Purpose
`UNIT`	Unit number to which the file will be written.
`CARD` `FILE`	These two options control whether the output is in binary or text form. `CARD` specifies text form, and `FILE` specified binary. The default is text format, except for the `ALL` option.
`TITLE`	Specifies an output file title. It is used for the text format or writing the entire Poisson-Boltzmann data structure. You may specify three slashes, "`///`", as a line delimiter, so multi-line titles can be specified.
`ALL`	When `ALL` is specified, this command is used to write the entire Poisson-Boltzmann data structure to a file in binary format. Use of this option make specification of the `UNIT` and `TITLE` options mandatory, and you cannot specify the `X`, `Y`, `Z`, `LINESZ`, `CARD`, or other data array name.
`PHI`	Specifies the use of the potential grid.
`RHO`	Specifies the use of charge density grid.
`EPSX`	Specifies the use of the X axis dielectric grid.
`EPSY`	Specifies the use of the Y axis dielectric grid.
`EPSZ`	Specifies the use of the Z axis dielectric grid.
`KAPPA`	Specifies the use of dielectric independent Debye-Huckel constant array.
`X range` `Y range` `Z range`	Specifies the range of indices to be output for X, Y, Z. Counting starts from zero and ends with the number of grid points minus one.
`LINESZ`	Specifies the maximum number of characters per line for the text output. The default is 130.

At least one of `RHO, PHI, EPSX, EPSY, EPSZ, KAPPA` must be specified.

8.3.6 PBE TEST Command

Syntax

```
PBE TEST { CV cv-options }  repeat(pbe-test-options)
         { DH dh-options }

cv-options ::= [CHARge real    ]
               [RADIus real    ]
```

```
                    [CAVIty-eps real]
                    [QPOS real      ]

                    [CHARge real      ]
     dh-options ::= [RADIus real      ]
                    [CAVIty-eps real  ]
                    [IONStr real      ]
                    [TEMPERATURE real ]

                          [XDIM int    ]
                          [YDIM int    ]
     pbe-test-options ::= [ZDIM int    ]
                          [GRID real   ]
                          [SOLVent real]
```

Function

The `PBE TEST` command is used to generate test potentials for comparing the finite difference solution against two particular problems for which analytic solutions are available. The two problems are a charge in a spherical cavity of one dielectric with a different surrounding dielectric (the `CV` option), and a charged centered in a sphere surrounded by a Debye-Huckel fluid (the `DH` option). These calculations are done using analytic series representations, but they are not optimized for numerical accuracy, so beware of artefactual peaks. CONGEN will report the number of failures of the series convergence. Also, a setting of 3 for the `PBE DEBUG` variable will display the terms in each series calculation, see Section 27.11 [Debug Command], page 230. A debug setting greater than 10 will cause CONGEN to substitute the number of iterations in place of the potential.

The analytic potentials used in this code were derived by Malcolm Davis.

The command options have following meaning:

Option	Purpose
`CV`	Specifies a potential for a spherical cavity with an internal charge with an external dielectric be computed.
`DH`	Specifies a potential for a spherical cavity with a charge at the center surrounded by a fluid with Debye-Huckel charge screening.
`CHARGE`	Specifies the interior charge value. The default is 1.0.
`RADIUS`	Specifies the cavity radius. The default is 5.0 Angstroms.
`CAVITY-EPS`	Specifies the dielectric of the cavity. The default is 1.0.
`QPOS`	Specifies the position of the charge in the cavity test case along the X axis. The default is 0.0 (the origin).
`IONSTR`	Specifies the ionic strength in molarity units for the Debye-Huckel test case. The default is the current value in the `pbe_info` data structure.
`TEMPERATURE`	Specifies the temperature for the Debye-Huckel test case. The default is the current value in the `pbe_info` data structure.

`XDIM`
`YDIM`
`ZDIM` Specifies the X, Y, and Z dimensions of the test grid, respectively. If there is no grid defined in the `pbe_info` data structure, then the default for all three dimensions is 20. Otherwise, the current grid dimensions provide defaults.

`GRID` Specifies the grid dimension. The current `pbe_info` data structure provides the default.

`SOLVENT` Specifies the solvent (external) dielectric constant. The current `pbe_info` data structure provides the default.

8.3.7 PBE SAVE Command

Syntax

```
PBE SAVE name
```

Function

The entire `pbe_info` data structure is saved under the given `name`. Any other `pbe_info` data structure saved under the given `name` will be destroyed. The `PBE RECALL` can be used to recover it.

8.3.8 PBE RECALL Command

Syntax

```
PBE RECALL name
```

Function

The entire `pbe_info` data structure is replaced with the one stored under `name`. Memory associated with the previous data structure is freed.

8.3.9 PBE DIFF Command

Syntax

```
PBE DIFF name
```

Function

The `PBE DIFF` command calculates the difference in the potential grid and in the PBE energies for each atom between the current `pbe_info` data structure, and the one stored under `name`. The results are left in the current data structure.

8.3.10 PBE DESTROY Command

Syntax

```
PBE DESTROY
```

Function

The `PBE DESTROY` command destroys the current `pbe_info` data structure. The command has no options.

8.3.11 PBE STATISTICS Command

Syntax

```
PBE STATISTICS data-name [MIN real] [MAX real]
               [LINES int] [COLUMNS int] [IGNORE real]

              { PHI   }
              { RHO   }
data-name ::= { KAPPA }
              { EPSX  }
              { EPSY  }
              { EPSZ  }
```

Function

The `PBE STATISTICS` command prints a histogram of the values in any of the six arrays used in the `pbe_info` data structure. The options are as follows:

Option	Purpose
`data-name`	Specifies which grid is used. See `PBE WRITE Command`, for the precise table.
`MIN`	Specifies the minimum value of histogram buckets to be used. The default is the minimum in the data.
`MAX`	Specifies the maximum value of histogram buckets to be used. The default is the maximum in the data.
`LINES`	Specifies the number of lines to use in the histogram. The default is 60.
`COLUMNS`	Specifies the number of columns to use in the histogram. The default is 80.
`IGNORE`	Specifies specific data points to ignore. For example, the `PBE POTENTIAL NODIELECTRIC` command can generate grid points with “infinite” potentials whose value is set to 1.0E9. Such points can be ignored in the construction of the histogram.

8.3.12 PBE DUMP Command

Syntax

```
PBE DUMP
```

Function

The `PBE DUMP` command displays detailed information about the Poisson-Boltzmann data structures. It is useful primarily for debugging and regression testing.

8.3.13 PBE MASK Command

Syntax

```
PBE MASK data-name relop real [ABS] [USING data-name] [SETTO real]
         [RECALL name]

              { PHI   }
              { RHO   }
data-name ::= { KAPPA }
              { EPSX  }
              { EPSY  }
              { EPSZ  }
```

```
                { GT }
                { GE }
     relop ::=  { LT }
                { LE }
                { EQ }
                { NE }
```

Function

The PBE MASK command allows the user to identify or mask values in any of the grids. This is useful for identifying interesting regions prior to plotting, or for gathering statistics about regions of the system. For example, it is possible to mask all regions of the potential that lie within the van der Waals surface, and then calculate statistics on the outer regions.

The options have the following meanings:

Option	Purpose
data-name	The data structure being changed.
relop real	The relop specifies a relational operator for which each array element is compared against the reference number, given by the real, and if the relationship is true, the corresponding array element is changed.
ABS	Absolute values of the reference number and the array elements are used for comparison.
USING data-name	By default, the comparison done by this command are done on the same array as the one being modified. When the USING option is invoked, it specifies the array to be used for comparisons.
SETTO real	Specifies the new value for data elements which satisfy the masking condition. The default is zero.
RECALL name	Normally, the reference data structure is the same as the one being masked. The RECALL option specifies the name of the another pbe_info data structure to be used for the reference array.

The following commands illustrates setting the potential to zero for all points within and including the Stern layer. Note that this command depends on the ionic strength being positive.

```
PBE MASK PHI EQ 0.0 USING KAPPA
```

8.4 PBE Examples

Two examples of using the Poisson-Boltzmann equation are presented. One example shows the calculation of electrostatic binding energy, and the other shows the basics of using the PBE in a conformational search.

8.4.1 Calculation of the Electrostatic Binding Energy

In this example, we illustrate how to calculate the electrostatic binding energy of a complex between two molecules. Assume that the molecules have segment identifiers of A and B.

```
! Calculate energy of the complex.
pbe setup solvent 78.0 interior 2 all end -
    uniform boundary zero ionstr 0.15 -
    grid 0.4 temp 300.0 water 0.0 stern 2.0 -
    xdim 190 ydim 170 zdim 135 -
    all
pbe potential maxiter 2000 rtol 1.0e-4
pbe energy

! Calculate energy of segment A
pbe setup solvent 78.0 interior 2 all end reuse -
    uniform boundary zero ionstr 0.15 -
    grid 0.4 temp 300.0 water 0.0 stern 2.0 -
    xdim 190 ydim 170 zdim 135 -
    clear atom A * *
pbe potential maxiter 2000 rtol 1.0e-4
pbe energy

!Calculate energy of segment B
pbe setup solvent 78.0 interior 2 all end reuse -
    uniform boundary zero ionstr 0.15 -
    grid 0.4 temp 300.0 water 0.0 stern 2.0 -
    xdim 190 ydim 170 zdim 135 -
    clear atom B * *
pbe potential maxiter 2000 rtol 1.0e-4
pbe energy
```

The electrostatic binding energy would be the difference between the first energy and the sum of the last two.

In this example, it should be noted that the calculations of each component are done using the same grid as the complex. This results in the clean subtraction of the self-energies.

8.4.2 Poisson-Boltzmann Equation Usage in Conformational Search

This example is taken from '`$CGT/cgpbe2.inp`'. It illustrates using the substitution of the Poisson-Boltzmann electrostatic energy for the Coulomb energy for evaluating all the conformers generated by search, where the sidechains are optimized using the CHARMM potential. This is necessary to keep the execution time reasonable, but this is not a good example for a real application.

```
Conformational search over NEW
Test of PBE evaluation in parallel
*
OPEN NAME CGDATA:RTOPH8.MOD UNIT 01 READ UNFORM
OPEN NAME CGDATA:PARAM5.MOD UNIT 03 READ UNFORM
OPEN NAME CGTD:newpsf.mod UNIT 12 READ UNFORM
OPEN NAME CGTD:NEWV.MOD UNIT 14 READ UNFORM
READ      RTF UNIT 1
READ      PARAMETER UNIT    3
READ PSF FILE UNIT 12
READ COOR FILE UNIT 14
parallel loops ncpu 4
!
! Setup the conditions for using Poisson-Boltzmann equation.
! The grid size is too large for a real calculation, though.
!
PBE SETUP IONSTR 0.15 BOUNDARY SDEBYE CENTER SOLVENT 78 GRID 1.5 -
          WATER 0.0 STERN 2.0 XDIM 50 YDIM 50 ZDIM 50 -
          TRILINEAR INTERIOR 2.0 ALL END
PBE POTENTIAL RTOLER 1.0E-3 MAXITER 1000
PBE ENERGY
!
! Generate conformations for the loop H 31-35 using the CHARMM
! potential energy.
!
OPEN UNIT 60 NAME CGPBE2.CG1 UNFORM WRITE
OPEN UNIT 70 NAME CGDATA:TOPCGEN2.INP FORM READ
OPEN UNIT 51 NAME CGDATA:EMAPGLY30.OMP FORM READ
OPEN UNIT 52 NAME CGDATA:EMAPALA30.OMP FORM READ
OPEN UNIT 53 NAME CGDATA:EMAPPRO30.OMP FORM READ
OPEN UNIT 55 NAME CGDATA:PRO.CNS FORM READ
CGEN -
SEARCH DEPTH END -
NBCG CUTNB 5.0 CTONNB 98.0 CTOFNB 99.0 END -
HBCG CUTHB 4.5 CTONHB 98.0 CTOFHB 99.0 -
     CUTHBA 90.0 CTONHA 90.0 CTOFHA 90.0 END -
BACK CISTRANS STARTRES H 31 LASTRES H 32 MAXEVDW 100 CLSA H 35 CA $ -
CHAIN CISTRANS STARTRES H 33 MAXEVDW 100 $ -
SIDE STARTRES H 31 LASTRES H 35 MAXEVDW 100 SIDEOPT ITER SGRID MIN EVAL E $ -
WRITE CUNIT 60 $ -
ERINGPRO 50 GLYMAP 51 ALAMAP 52 PROMAP 53 PROCONS 55 -
GLYEMAX 2 ALAEMAX 2 PROEMAX 2 STUNIT 70
CGPBE2.CG1
CGPBE2 test case
Conformations of H1 loop in NEW.
*
!
! Now evaluate the conformations using the Poisson-Boltzmann energy
! substituting for the Coulomb energy.
!
CLOSE UNIT 60
OPEN UNIT 59 NAME CGPBE2.CG1 UNFORM READ
OPEN UNIT 60 NAME CGPBE2.CG UNFORM WRITE
OPEN UNIT 70 NAME CGDATA:TOPCGEN2.INP FORM READ
OPEN UNIT 51 NAME CGDATA:EMAPGLY30.OMP FORM READ
OPEN UNIT 52 NAME CGDATA:EMAPALA30.OMP FORM READ
OPEN UNIT 53 NAME CGDATA:EMAPPRO30.OMP FORM READ
OPEN UNIT 55 NAME CGDATA:PRO.CNS FORM READ
CGEN -
PBE NEWB END -
SEARCH DEPTH END -
NBCG CUTNB 5.0 CTONNB 98.0 CTOFNB 99.0 END -
HBCG CUTHB 4.5 CTONHB 98.0 CTOFHB 99.0 -
     CUTHBA 90.0 CTONHA 90.0 CTOFHA 90.0 END -
RBEST UNIT 59 NBEST 9999 $ -
EVL MINI ENERGY END $ -
```

```
WRITE CUNIT 60 $ -
ERINGPRO 50 GLYMAP 51 ALAMAP 52 PROMAP 53 PROCONS 55 -
GLYEMAX 2 ALAEMAX 2 PROEMAX 2 STUNIT 70
CGPBE2.CG
CGPBE2 test case
Conformations of H1 loop in NEW.
*
!
! Extract the best conformation
!
OPEN UNIT 60 NAME CGPBE2.CG READ UNFORM
XCONF 60 BEST 1
```

9 Symmetry and Molecular Images

The Images commands allows symmetric images of the primary molecule(s) to participate in the energy and structure of a system.

The syntax of image input and output commands is given in Chapter 3 [I/O], page 15.

By reading an image file (‘.IMG’), the images of the primary atoms are included in any energy and force determinations for the remainder of the calculation (unless another image file is read). The main use for this facility is to simulate crystal environments or to obtain periodic boundary conditions. It is also acceptable to use this facility in finite systems such as dimers and tetramers.

9.1 Image Files

An Image file contains all of the information needed to define the position and orientation of all symmetric images of the primary atoms. The file is a card image file and a sample file is given:

```
*  IMAGE FILE FOR BETA SHEET TEST CASE
*  ONLY ONE TRANSFORMATION (Z AXIS ROTATION)
*
    1
  1.0       1.0       1.0
ZROT
 -1.0       0.0       0.0
  0.0      -1.0       0.0
  0.0       0.0       1.0
  0.0       0.0       0.0
```

The title is in standard CONGEN format, see Section 2.12 [Syntactic Glossary], page 12. The next contains the number of transformations (`I5`). The following line contains the unit cell dimensions (multiplying factors) (`3F10.5`).

The remaining lines are in groups of five, one group for each transformation . For each group, the first line contains the name of the transformation (`A8`). The next three lines contains the rotation matrix, and the last line contains the translation vector (all `3F10.5`).

The rotation matrix must be unitary, however, a non-unitary rotation matrix will only result in a warning. The use of an anti-unitary matrix can be used to include the mirror image of any system into the calculation. The values of the translation vector are multiplied by the cell dimensions listed at the beginning of the file to generate the actual translation steps. For the use in this program the translation is done *after* the rotation has been made.

One other restriction on the transformations is that every transformation *must* have an inverse. Again, there is only a warning if this restriction is violated as there may be examples where this is desired. In the example above, this transformation is its own inverse. Non-physical results will be obtained if this restriction is violated.

The maximum number of allowed transformations is 26 (the number needed for simple 3-dimensional periodic boundaries). This limit can easily be increased by modifying `INIMAG`.

For other examples of image files see the test cases.

9.2 Image Operation

The routines in IMAGES.FLX can be classified into three sections. These categories are:

1. Set up images — `IMIGIO`, `IMREAD`, `IMWRIT`, `REIMAG`, `INIMAG`.

2. Set up energy terms — `IMHBON`, `NEWHBL`, `IMHBFX`, `NBONDM`.
3. Compute energy — `EIMAGE`, `TRANSO`, `TRANSI`, `SFLSET`.

The first category involves reading the image file (`IMREAD`) and setting up the data structure (`REIMAG`, `INIMAG`) see the CONGEN source file '`IMAG.FCM`'.

The second category in addition to finding the energy terms, also selects which image residues are to be kept. This selection process is repeated each time the nonbonded list is updated. Since the list of atoms can change its not possible to maintain a fixed list of hydrogen bonds explicitly for image interactions. It is also the case that the image Hbond list must be recomputed each time a new nonbond list is generated. This is done automatically, but it is highly recommended to keep INBFRQ and IBHFRQ (see Section 7.4 [General Energy Operands], page 58) the same to avoid confusion. The ST2 interactions are also computed when the nonbonded list is updated (`NBONDM`).

The third category is concerned with the computation of energy terms. For the actual computation of energy, standard routines are used (`ENBOND`, `EHBOND`, `ENST2`) with a modified calling sequence. The procedure used is:

1. Compute coordinates for all image atoms.
2. Set up arrays for self energy terms (atom with its own image).
3. Compute self terms, divide energy by 2, zero out image forces.
4. Compute remaining terms including forces on image atoms.
5. Transform forces on image atoms back into the primary space.

Using a procedure where the forces on image atoms is kept, allows for a substantial reduction in the number of necessary image atoms. This results in the necessity that all transformations have an inverse. This procedure has the drawback that the self energy terms must be treated specially and that all hydrogen bonds between image and primary atoms must be computed and then trimmed of any repeats.

Since there is no treatment of the second derivative of the energy for image atoms, the procedures involving Newton-Raphson minimizations (see Section 7.5 [Minimization], page 59) and vibrational analysis (see Chapter 10 [Vibrational Analysis], page 89) will not function properly.

10 Vibrational Analysis

The vibrational analysis routines are designed to provide the user a variety of analysis options as well as features designed to facilitate the interface to the analysis (see Chapter 16 [Analysis], page 157). The vibrational analysis and this documentation was written by Bernard R. Brooks who is presently (1990) at NIH.

To use the vibrational analysis, one must first execute then `VIBRAN` command. This enters a separate command parser which has its own set of commands. Control is returned to the top level of CONGEN with an `END` command.

In order to process commands with the vibrational analysis routines, The energy terms must all be defined, and the structure must be determined (see Section 7.2 [Energy Needs], page 57). At present, fixing constraints (see Section 11.4 [Fixed Atoms], page 97) SHAKE (see Section 11.6 [Shake Command], page 107), and images (see Chapter 9 [Images], page 87) are not supported.

Keywords used to define hydrogen bonds and nonbonded interactions may be included in the command that invokes `VIBRAN`.

10.1 Syntax for Vibrational Analysis Commands

10.1.1 Main Command Syntax

```
VIBRan  [hbond-spec] [nbond-spec] [nmode-spec]

nmode-spec ::= NMODe integer
```

The syntax for `hbond-spec` is given in Chapter 5 [Hydrogen Bonds], page 49. The syntax of `nbond-spec` is given in Chapter 6 [Non-bonded Interactions], page 51. The `nmode-spec` allocates space on the heap. For large systems, if normal modes are to be used, It should be set to the largest number needed. Its default value is set to 10 if the number of atoms (`NATOM`) is greater than 50, and 3*`NATOM` otherwise.

10.1.2 Subcommand Syntax

```
READ    { { NORMal-modes[APPEnd]              } [FILE] unit-spec  }
        { { COORdinate  [COMParison] coor-spec }                  }

        { { COORdinate [COMParison] coor-spec } unit-spec  }
WRITe   { { NORMal-modes  [mode-spec]        } [FILE]     }
        { { SECOnd-derivatives               }            }
        { { TRAJectory trajectory-spec       }            }
        { { IC                               }            }

trajectory-spec::= mode-spec magnitude-spec [PHAS real] [SEQUential-files]

        { { COORdinate       } [COMParison]                   }
PRINt   { { IC               }                                }
        {                                                     }
        { NORMal-modes  [mode-spec] [magnitude-spec] print-spec }

print-spec ::= [INTDerivatives] [VECTors] [DOTProducts] [DIPOle]

PROJect { { FORC   }                              }
        { { USER   } [mode-spec] [magnitude-spec] }
        { { COMP   }                              }

DIAGonalize [NOMAss]  [NFREquencies integer]
```

```
EDIT    { INCL { TRAN } [NOMAss] [NONOrmalize] [ORTHogonalize] }
        {      { ROTA }                                        }
        {      { COMP }                                        }
        {      { FORC }                                        }
        {      { USER }                                        }
        {                                                      }
        {  { DELEte} [mode-spec]                               }
        {  { ORTHogonalize } [NONOrmalize]                     }

FILL    {  NORM } [mode-spec] [magnitude-spec] [APPE]
        {  COMP }

EXPLORE [mode-spec] [magnitude-spec] [COMP] [GRID integer]

END

unit-spec ::= UNIT unit-number

mode-spec ::= MODE integer [ THRU integer ]

magnitude-spec ::= { TEMP real TFRE real } [NOMAss] [NONOrm]
                   { KCAL real TFRE real }
                   { RMS  real           }
                   { FACT real           }
```

10.2 I/O For Normal modes

The `VIBRAN` section supports it own I/O commands. Commands to read, write and print coordinates and internal coordinates are identical with those in the main program. This section can read and write the Normal Mode data structure and write out the second derivative matrix (for external use) and normal mode trajectories (for the movie programs). Also, useful information about normal modes may be printed using the `PRINT NORM` command.

Normal Mode vectors may only be read and written in binary (`FILE`) format. When writing, a unit must be specified, and a contiguous subset of modes may be specified using the `mode-spec`. When reading modes, all of the modes in the normal mode file will be read (assuming there is enough space). If the available space is exhausted, a warning is issued, and further reading stops. Existing modes will be deleted when the `READ NORM` command is executed unless the append option is used, in which case, the new modes are added sequentially at the end. No modification of modes is done upon reading (i.e. normalization, or orthogonalization).

When printing Normal Modes, a variety of options may be specified. A contiguous subset of modes may be specified (the default is all modes), and an appropriate magnitude may be specified (see Section 10.4 [Normal Modes], page 91). For each specified mode, the frequency, eigenvalue, force projection, percentage of translation-rotation, and magnitude information will be printed. In addition, internal derivatives (by finite differences of the internal coordinate data structure), displacements in coordinate space, and dotproducts with other modes can be printed.

Trajectory files may be written out for a set of modes with a given magnitude factor. The modes for all specified modes may be written out in one file, or in separate files for different modes where sequential unit numbers are used starting with the specified unit number. The `SEQUENTIAL` keyword will cause sequential files to be written.

10.3 Second Derivatives

The second derivatives are computed during energy determination and they are stored in temporary array that are allocated dynamically. Once obtained, they can be written out, or diagonalized internally for small systems (up to 200 atoms).

The following is from comments in ENERGY;

```
IF 'NSECD=.FALSE.' THEN THE SECOND DERIVATIVES OF THE ENERGY
ARE RETURNED IN THE ARRAYS DD1,DD2,DD3,DD4,IDD4,JDD4,AND NDD4.
TREIR USAGE CAN BE SUMMARIZED BY;
    ARRAY        SPACE      TYPE         DESCRIPTION
     DD1         6*NATOM    REAL*8       DIAGONAL ELEMENTS
     DD2         6*NNB      REAL*4       NONBONDED INTERACTIONS
     DD3         9*IBLO(NATOM) R*4       EXCLUDED INTERACTIONS
     DD4         9*NND4     REAL*4       EXTRA INTERACTIONS
     IDD4        NDD4       INTEGER*2    FIRST ATOM OF EXTRAS
     JDD4        NDD4       INTEGER*2    SECOND ATOM OF EXTRAS
 IF THE LOGICAL VARIABLE 'NSECD' IS SET TO .FALSE. SECOND
DERIVATIVES WILL BE GENERATED. 'DIAGSD' IS SET TO .TRUE. IF ONLY
THE DIAGONAL (DD1) SECOND DERIVATIVES ARE DESIRED.
FOR DD1 EACH ATOM REQUIRES 6 SEQUENTIAL R*8 LOCATIONS. THE
SEQUENCE IS XX,XY,YY,XZ,YZ,ZZ  WHERE EACH IS THE PARTIAL SECOND
DERIVATIVE OF THE ENERGY WRT THE CORRESPONDING MOTIONS OF THE
PARTICULAR ATOM. DD2 IS STORED IN THE SAME MANNER EXCEPT THAT
EACH 6 R*4 LOCATIONS CORRESPOND TO A PARTICULAR NONBONDED
INTERACTION. THE SEQUENCE OF NONBONDED INTERACTIONS IS DEFINED
BY THE INTEGER ARRAYS 'INBLO' AND 'JNB'. SINCE THE NONBONDED
INTERACTIONS ONLY INVOLVE RADIAL FORCES, ONLY 6 LOCATIONS ARE
NEEDED.(IE  XY=YX). DD3 CONTAINS THE SECOND DERIVATIVES FOR ATOMS
THAT INTERACT THROUGH THE EXCLUDED LIST. SINCE SYMMETRY IS NOT
PRESENT FOR ANGLES,ETC... 9 LOCATIONS ARE NEEDED GIVEN BY THE
SEQUENCE  XIXJ,YIXJ,ZIXJ,XIYJ,YIYJ,ZIYJ,XIZJ,YIZJ,ZIZJ. ATOM
I IS ALWAYS LESS THAN ATOM J. THE SEQUENCE OF EXCLUDED
INTERACTIONS IS DEFINED BY 'IBLO' AND 'INB'.
DD4 CONTAINS ALL OTHER INTERACTIONS NOT CONTAINED ABOVE.
THESE ARE USUALLY 1-4 DIHEDRAL INTERACTIONS AND HYDROGEN BONDS.
INFORMATION IS STORED AS IT IS IN DD3. SINCE SOME CODE REFERENCES
BOTH DD3 AND DD4, THE DIFFERENCE OF THEIR BASES IS NEEDED. THIS
IS STORED IN IDD3AD (IE  IDD3AD=BASE(DD3)-BASE(DD4) ).
CODE TO CONVERT THIS STORAGE SCHEME TO SIMPLE UPPER TRIANGULAR
FORM CAN BE FOUND IN 'MINMIZ' UNDER THE NEWTON RAPHSON SECTION.
```

10.4 Normal Modes

There are two ways that normal modes are stored internally in CONGEN. The most common usage is as one double precision mass weighted array. A series of such arrays usually span an orthonormal basis (as would be the case upon diagonalization). The second method is to represent a normal mode as three non mass weighted coordinate displacement arrays, stored in single precision. The program automatically converts between them as necessary. Whenever interconversion is to be done, a "magnitude specification" may be given. This specification requires a step type and step length. The valid step types are; `FACT` (simple factor), `TEMP` (put mode at this temperature), `KCAL` (put in the specified Kcals), and `RMS` (step along until this RMS is reached). When specifying

`TEMP` or `KCAL`, a terminal frequency may be specified (default `TFREQ` is 5.0) which prevents excessive stepping along very low, or translation-rotation modes.

The procedure used in going from double precision to coordinate displacement arrays is:

1. Normalize double precision vector (unless NONOrm keyword is used).
2. Mass weight by dividing by root(mass) (unless NOMA keyword is used).
3. Multiply by appropriate scale factor from step type an length.

To convert from single precision into double precision, the procedure is:

1. Save inner product (as initial step length).
2. Mass weight by multiplying by root(mass) (unless NOMA keyword is used).
3. Normalize vector (unless NONO keyword is used).
4. Compute scale factor using initial step length and step type.

Whenever a magnitude-specification is called for, some interconversion will take place. The conversion from double precision to coordinate displacements takes place in the subcommands:

```
PRINt NORMal-modes
WRITe TRAJectory
PROJect
FILL
EXPLore
```

The interconversion from coordinated displacements to double precision takes place in the subcommands:

```
PROJect
EDIT NORMal-modes
```

Normal modes are always stored in double-precision. Only one mode is represented as coordinate displacements (at any one time), and its use is usually temporary. The arrays for this one coordinate displacement mode are `XNORM`, `YNORM`, and `ZNORM`. These arrays can be filled using the `FILL` command and then used in other pars of CONGEN.

The Normal Mode data structure of double precision arrays is local to the Vibrational analysis section of CONGEN, and the storage space for these arrays is released when exiting to CONGEN via the `END` command.

10.5 Manipulation of Normal Modes

There are several commands that con modify the normal mode vector space. In addition to the obvious ones such as `READ NORMal-modes` and `DIAGonalize`, there is also an `EDIT` command which can be used to modify the normal mode data structure. The `EDIT DELETE` command will delete specified modes from the date structure. The `EDIT ORTHogonalize` command is used to orthogonalize and optionally normalize a particular subset of normal modes. Additional modes can be added three at a time with the `EDIT INCLude TRANslation` and `EDIT INCLude ROTAtion` commands. These commands will add on the translation and rotation eigenvectors respectively. This option is important when individual Coreolis coupling terms are needed. Single additional modes may be added with the `EDIT INCLude COMP` or the `EDIT INCLude FORCe` or the `EDIT INCLude USER` commands. For the `COMP` option, the difference between the comparison and main coordinate sets will be appended, for the `FORCe` option, the current values in the force arrays (from the last energy evaluation) will be appended. There is also the ability to specify a user vector. this is done by the inclusion of the subroutine `USERNM` in your `USERSB` and using the `USERLINK` facility, see Section 2.11

[CONGEN Modifications], page 11. For all of the `EDIT INCLude` options, the defaults are to mass weight, normalize, and orthogonalize to the rest of the vector space. To skip any of these steps, the `NOMA`, `NONO`, and `NOOR` keywords must be specified.

There are two commands which are used to obtain information about particular normal modes without modifications. These are the `PRINT NORMal-modes` and `PROJect` subcommands. The `PROJect` command will project one of; `FORCe` (the current force), `COMParison` (the difference between the main coordinates and the comparison set), and `USER` (a user supplied mode or coordinate displacement). The information displayed is:

```
MODE integer    - mode number
FREQUENCY       - frequency of this mode
NORMAL DOTPR    - actual dotproduct of normalized vectors
PROJECTION      - ratio of guess vector projection to step length
APPROX DEL E    - estimate of energy increase along this mode
TYPE            - type of step (FACT, RMS, KCAL, TEMP)
STEP            - step length for this step type
```

The `NOMAss` and `NONOrm` keywords may be used to prevent mass weighting or normalization of modes.

The comparison coordinates can be modified with the `FILL COMP` command. This command will copy the main coordinate set to the comparison set, and then step along the specified mode by the specified magnitude. When the append (`APPE` keyword) is used, the main coordinates are not first copied. The `FILL NORM` command will fill the normal mode coordinate displacement arrays with the specified vector. Again, the append option will prevent the zeroing of these arrays before stepping along the mode. Several other commands use and modify the normal coordinate displacement vectors, so the `FILL NORM` command should be executed just before exiting to CONGEN with the `END` command.

10.6 Exploring Energy Surfaces

The `EXPLORE` command does a linear search along a given normal mode. The `GRID` option gives the number of points evaluated.

This command is presently incomplete in CONGEN.

11 Constraints

Six forms of constraints are available in CONGEN: harmonic atom constraints, harmonic dihedral constraints, fixed atom constraints, Nuclear Overhauser Enhancement (NOE) distance constraints from NMR spectroscopy, dihedral angle constraints based on J coupling constants determined by NMR, and fixed bond and angle constraints (`SHAKE`).

11.1 Restraining Atomic Movements

11.1.1 Syntax of Harmonic Atom Constraints

```
CONStraint HARMonic FORCE real [PRINT] [REF coor-spec del]
           [MASS] atom-selection
```

`coor-spec` is a specification for the coordinates. See Section 3.1.1 [Read Syntax], page 15, for the syntax.

Syntactic ordering: `HARMonic` must follow `CONStraint`, and `FORCE` must follow `HARMonic`.

11.1.2 Function of Harmonic Atom Constraints

The potential energy has a harmonic constraint term which allows one to prevent large motions of individual atoms. The form for this potential is as follows for coordinates:

$$EC = \sum_{\text{atoms}} k_i (x_i - refx_i)^2$$

where *refx* is a reference set of coordinates. If `MASS` is specified in the command line, then k is multiplied by the mass of the atom resulting in a natural frequency of oscillation for the constraint of sqrt(k) in AKMA units. An atom constrained with `MASS FORCE 1.0` will oscillate at 7.6 cycles/picosecond if free of other interactions.

CONGEN supports a number of operations on the coordinate constraints. The constraint for any atom can be set to any positive value (specified by the `FORCE` keyword followed by the desired value). The reference coordinates can be the current set at the point when constraints are specified (the default) or a set can be read from a coordinate file (specified by `REF` and a `coor-spec`). The force constants and reference coordinates can read or written as a unit. The `PRINT` option prints a list of all the current harmonic constraints that are applied to the system after this command has been executed.

It is important to understand some aspects of how the constraints are set in order to get the most flexibility out of this command. When CONGEN is loaded, each atom has associated with it a harmonic force constant initially set to zero. Each call to the `CONSTRAINT HARMONIC` command changes the value of this constant for only those atoms specified.

11.1.3 Other Commands for Harmonic Atom Constraints

The harmonic constraints may be read and written to files. The file name to be specified in the `READ` and `WRITE` command is CONS. The files may be read or written only in binary. The `PRINT` command will also work for constraints. See Chapter 3 [I/O], page 15, for more details. In addition, one may look at the contributions to the energy in detail using the analysis facility, see Chapter 16 [Analysis], page 157.

`PRINT` specifies that a listing of of all the atoms currently constrained should be printed out. This is done by segments of constrained atoms, which is concise in most cases. Unfortunately in the case of IUPAC specified constraints it is quite verbose.

11.2 Holding Dihedrals Near Selected Values

Using this form of the `CONS` command, one may put constraints on the dihedral angles formed by sets of any four atoms. The constraints may be set to either a single angle, or to be bounded by two angles (a flat bottom well). The improper torsion potential is used to maintain said angles.

The command for setting the dihedral constraints is as follows:

11.3 Syntax of Dihedral Constraints

```
CONStraint { DIHEdral [BYNUM] dihe-spec FORCE real MINimum real MAXimum real }
           { CLDH                                                             }
```

Syntactic ordering: `DIHEDRAL` or `CLDH` must follow `CONSTRAINT`, and `FORCE` and `MIN` must follow `DIHEDRAL`.

If `BYNUM` is specified, then

```
dihe-spec ::= integer integer integer integer
```

If `BYNUM` is not specified, then

```
dihe-spec ::= atom-spec : atom-spec : atom-spec : atom-spec :
```

where:

```
atom-spec ::= segid resid iupac
```

Note that colons must be used as delimiters following each `atom-spec`.

11.3.1 Function of Dihedral Angle Constraints

`DIHEDRAL` adds a torsion angle to the list of constrained angles using the specified atoms, force constant, and minimum and maximum dihedral angles. `CLDH` clears the list of constrained dihedrals so that different angles or new constraint parameters can be specified.

When a range of angles is specified, it is important to keep in mind that torsion angles are periodic. Therefore, the `MINIMUM` and `MAXIMUM` bounds are taken literally. Reversing the order of values specified for these variables has the effect of complementing the range of the angular constraints. For example, a specification of `MINIMUM 175 MAXIMUM -175` is just 10 degrees, whereas the specification of `MINIMUM -175 MAXIMUM 175` is 350 degrees.

In order to simplify the specification of constraints, there are a number of defaulting rules used by this command. First, for each atom which follows the first, you can omit `segid`'s, `resid`'s, and `iupac` names if they match values from the previous atom. Be careful here if any of the identifiers are also used for higher order structures. For example, if a protein has a `segid` of 1 and a `resid` of 1, then specifying a single 1 will be interpreted as a segment identifier. In all cases, use the `PRINT CONS` command to check that you got what you expected.

Next, omitted `FORCE` values will be taken from the previous constraint if it exists. `MINIMUM` and `MAXIMUM` values will also default to previous values if neither option is specified. However, if either one is specified, but the other is missing, then they default to each other, so that the effect is to use a single point harmonic well.

Do not use a value of -9999 as the minimum or maximum dihedral angle, since the program uses this value to indicate that no angle was found on the command line.

These constraints can be coupled to a conformational search. See the option, `EPCONS`, in Section 12.4.9 [Miscellaneous Global Options], page 131.

11.3.2 Other Commands

The PRINT CONS command, see Section 3.3 [Print Command], page 38, will work for constraints. As of now (October 13, 1992), one cannot analyze the contributions of this term using the analysis facility nor can one read or write the description of this term out. Someday . . .

11.4 Fixing Atoms in Place

11.4.1 Syntax for Fixing Atoms

```
CONS FIX atom-selection-spec [PURG] [BOND] [THET] [PHI] [IMPH]
```

11.4.2 Function of Fixing Atoms

This command fixes atoms in place by setting flags in an array (IMOVE) which tells the minimization and dynamics algorithms which atoms are free to move. If atoms are fixed, it is possible to save computer time by not calculating energy terms which involve only fixed atoms. The nonbond and hydrogen bond algorithms in CONGEN check IMOVE and delete pairs of atoms that are fixed in place from the nbond and hbond lists respectively. In addition the PURG or individual energy term options specified with the CONS FIX command allow all or some of the internal coordinate energies associated with fixed atoms to be deleted. Interactions between fixed and moving atoms are maintained.

N.B. Because some energy terms are deleted from fixed systems, the total energy calculated with fixed atoms will be different from the total energy of the same system with all atoms free. The forces on the moveable atoms will however be identical.

The way CONGEN keeps track of fixed atoms is by the IMOVE array in the PSF. The IMOVE array is 0 if the atom is free to move, and has some other value if the atom is fixed. WARNING: the use of IMOVE is not yet universal in CONGEN. At present (November 15, 1990), it is supported for dynamics, all forms of minimization except Newton-Raphson. The vibrational analysis does not yet support it.

NOTE: If you use SHAKE in conjunction with fixed atoms, you should specify the SHAKE command after you have specified the fixed atoms. This will prevent SHAKE from compiling a list of bonds from which the CONS command will delete after it has a list of fixed atoms.

If PURGE is specified, every bond, bond angle, torsion angle, or improper torsion involving only fixed atoms is deleted. One can limit this elimination process to one type of interaction by specifying BOND, ANGL, PHI, or IMPH. By playing games with combinations of these commands, one can eliminate whole terms from the energy expression.

11.5 NMR Constraints

Constraints derived from NMR spectroscopy can be incorporated into the CONGEN energy function and used for minimization, simulated annealing using dynamics, or conformational search. The constraints from J-coupling constants can also be used to restrict atom construction in conformational search, see the option, EJCONS, in Section 12.4.9 [Miscellaneous Global Options], page 131, for more information.

The user may add either a NOE-derived distance constraint term or J-coupling constraint term to the usual potential function when calculating the energy and forces of a macromolecular system. There are two parts to specifying NMR constraints; the NMRC command and the constraints file. The NMRC command signals the program that the NMR constraints are to to be read in and the parameters associated with the function. The specification of which type of constraint function to add is specified in the constraint file. The constraint file is discussed in the following sections.

It also possible to account for conformational flexibility by using ensemble averaging. This facility is still experimental, and you should talk to Bob Bruccoleri or Keith Constantine for more information before using it.

A detailed listing of the NMR constraints can be obtained using the `PRINT NMR` command, see Section 3.3 [Print Command], page 38.

11.5.1 Theory for NOE Constraints

The NOE constraint function is designed improve upon a flat bottomed harmonic constraint function. The problem with the harmonic constraints is the high forces placed on atom pairs that are very far from their constraint distance. The potential in CONGEN uses a harmonic potential if the distance is close to correct, or if the distance is less than the lower bound of the constraint. However, at large distance, the constraint force is constant rather than harmonic. The connection between the harmonic section of the potential and the constant force section is done by an inverted harmonic piece. The main advantage of this functional form is that a user can input all constraints at once in the beginning of the run.

The functional form is as follows:

$$E_{noe} = \sum_i \begin{cases} 0.0 & \text{if } r_l \le r_{eff} \le r_u \\ K_i(r_{eff} - r_l)^2 & \text{if } r_{eff} < r_l \\ K_i(r_u - r_{eff})^2 & \text{if } r_u < r_{eff} < r_{sw} \\ -K_i(r_{plat} - r_{eff})^2 + 2\text{Emax} & \text{if } r_{sw} < r_{eff} < r_{out} \\ K_i\,\texttt{SLOPE}(r_{out} - r_{eff}) + \text{Emax2} & \text{if } r \ge r_{out} \end{cases}$$

where

i is taken over all NOE's,

K_i is the weight associated with the constraint,

r_{eff} is the effective interproton distance calculated from a single structure or over an ensemble of structures (see below),

r_l is the lower bound of the constraint,

r_u is the upper bound of the constraint,

r_{sw} is the point where the function switches to an inverted harmonic, and has the form, $r_{sw} = \text{FMAX}/2K + r_u$,

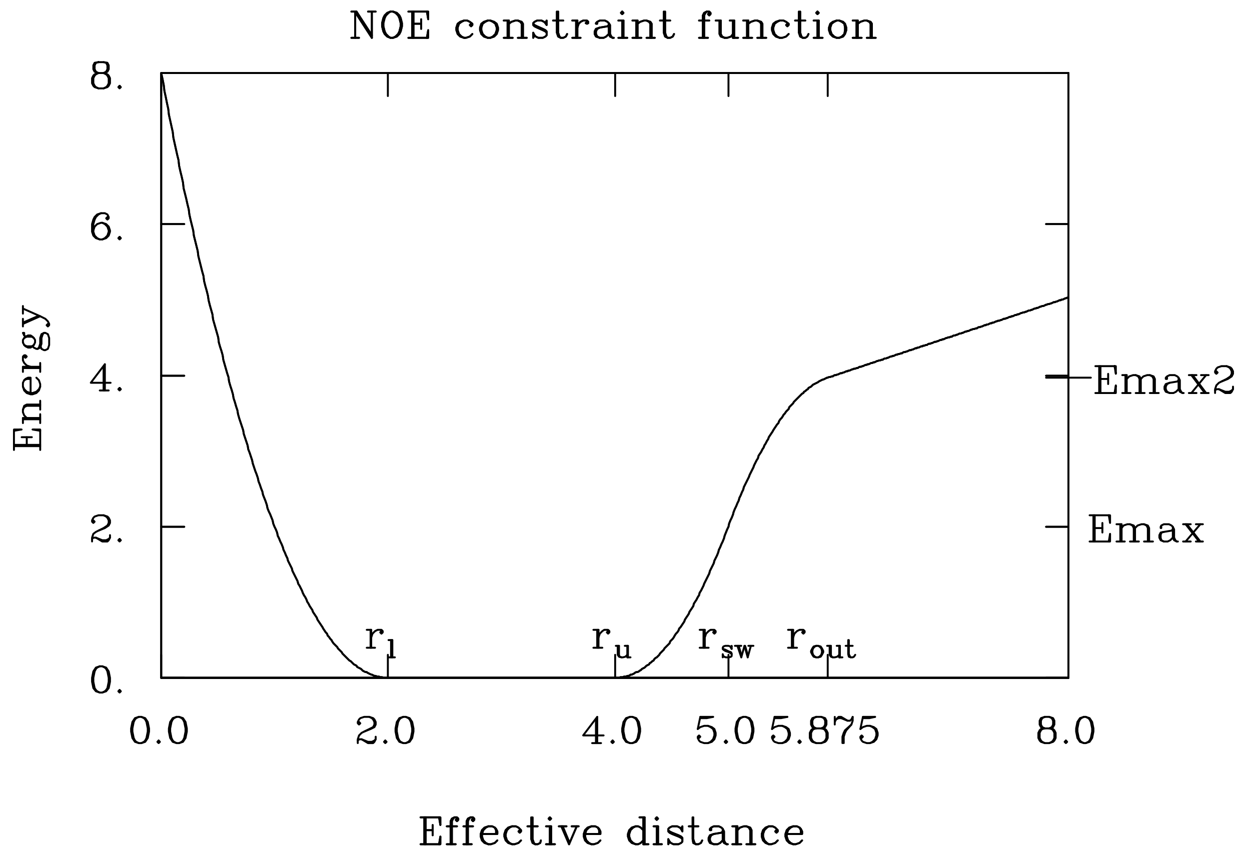
r_{plat} is the point where the derivative of the inverted harmonic would go to zero and has the form, $r_{plat} = 2r_{sw} - r_u$,

r_{out} is the point where the function goes linear and has the form, $r_{out} = r_{plat} - \texttt{SLOPE}/2K_i$,

Emax is the energy where the inverted harmonic switches in and is equal to $K(r_{sw} - r_u)^2$, and

Emax2 is the energy where the function goes linear and has the form, $\text{Emax2} = 2 * Emax - K(r_{out} - r_{plat})^2$.

The form of this function is given in the following figure where the lower bound is 2 Angstroms, the upper bound is 4 Angstroms, `K` = 2, `SLOPE` = 0.5, and `FMAX` = 4.0. Values for the various distances were computed from these parameters.



NOE constraints can be used in both standard and ensemble-averaged calculations. In a standard calculation, r_{eff} is just the distance derived from the individual structure being refined. In the ensemble average approach, multiple different structures, currently stored as different segments separated in space, are refined simultaneously, with distances for NOE constraints being computed using the following formula:

$$r_{eff} = \left(\frac{1}{n} \sum_{k=1}^{n} {r_{p_k}}^{-x} \right)^{-1/x}$$

where

r_{eff} is the effective interproton distance used in E_{noe} below,

k ranges over the conformations in the ensemble,

n is the number of conformations in the ensemble for which r_{p_k} can be calculated,

r_{p_k} is the interproton distance as calculated below, and

x is 6 or 3, depending on the averaging conditions.

Each member of the ensemble must have identical atoms and residues as determined by their IUPAC names and residue identifiers.

In cases where the NOE is due to the interaction of motionally averaged or prochiral protons, the interproton distance, r_p, is calculated over all possible pairing of two sets of protons. By default, the following expression is used:

$$r_p = \left(\sum \left(\frac{1}{r_i} \right)^6 \right)^{-1/6}$$

where the sum is taken over all possible pairs of atoms involved in the constraint. The constraints are specified using two sets of atom selections, see Section 11.7 [Atom Selection], page 108, so that any combination of atoms may be specified.

If NOE intensity scaling is done such that the calculated distances should reflect averages instead of sums, the `AVERAGE` option can be used when specifying the constraint. In this case, the following expression is used:

$$r_p = \left(\frac{\sum \left(\frac{1}{r_i} \right)^6}{N} \right)^{-1/6}$$

where N is the number of pairs used in the average. If any atom in the two sets of atoms in the constraint has undefined coordinates, then the interproton distance is omitted from other calculations. Likewise, if all the atoms in the two sets are fixed, see Section 11.4 [Fixed Atoms], page 97, the distance is ignored.

11.5.2 Theory for J Coupling Constraints

The J coupling constraint term allows one to incorporate constraints based on scalar J coupling measurements in two different ways, one where all atoms in the J coupling are known, and the other where two measurements are made each involving one proton in a pro-chiral pair, but for which the prochiral assignment is unknown. In this section, the form of the scalar J coupling equation will be discussed first followed by the two forms of the constraint equations.

The J-coupling constraint is based on the Karplus equation.[1][2]

$$J = c_1 \cos^2 \phi + c_2 \cos \phi + c_3$$

where J is the calculated value for the coupling constant given the current coordinates of the system.

ϕ is a torsion angle defined over four bonded atoms.

c_1, c_2, c_3 are coefficients for the coupling constant. The default setting for these coefficients is 6.4, -1.4, and 1.9 (the value for the J_{HNH_α}), but they can be changed for any constraint.

Ensemble averaging can be performed over a set of constraints as long as each constraint belongs to a different segment with the same structure, as described for the NOE ensemble averaging, see Section 11.5.1 [Theory for NOE Constraints], page 98. In the case of averaging, J is computed as follows:

$$\langle J \rangle = c_1 \langle \cos^2 \phi \rangle - c_2 \langle \cos \phi \rangle + c_3$$

[1] M. Karplus, *J. Chem. Phys.* **30**, 11-15 (1959).

[2] M. Karplus, *J. Am. Chem. Soc.* **85**, 2870-2871 (1963).

where the averages are computed over the members of the ensemble.

There are two ways to incorporate constraints from these scalar coupling constants into the energy function. When all the atoms involved in a scalar coupling constant are known, the following equation can be used:

$$E_j = \sum_i \begin{cases} 0.0 & \text{if } J_l \le J \le J_u \\ K_j K_n (J - J_l)^2 & \text{if } J < J_l \\ K_j K_n (J - J_u)^2 & \text{if } J_u < J \end{cases}$$

where i ranges over all J coupling constraints. Every parameter in the equation for E_j can be varied for each constraint.

J_l and J_u are the lower and upper bounds for a measured J coupling constant.

J is the calculated value for the coupling constant given the current coordinates of the system.

When one of the atoms involved in the J coupling constant measurement is a prochiral atom and if coupling constants for both prochiral atoms are known but unassigned, then the following form of the equation can be used. In this equation the two J coupling constants are "joined" together, and the constraint function is computed based on relationships involving the sums and magnitudes of the differences of the two J couplings, which obviates the need for stereospecific assignments. In the constraint file, the `JOIN` command is used to link to measured J constraints together.

In this functional form, the sum term is just a harmonic restraint based on the sum. The difference term is more complex because the sign of the difference depends on the arbitrary choice chirality on the prochiral center. The difference function V_d is harmonic where the magnitude of the calculated difference is bigger than the experimental difference. If the magnitude of the calculated difference is less, then a piecewise harmonic function with a maximum at $|J_1 - J_2| = 0$ is used. A additional factor is used for this term to smooth the overall function.

$$E_j = K_j K_n (V_s + V_d)$$

$$V_s = \begin{cases} \left((J_1 + J_2) - J^s_{max}\right)^2 & \text{if } J_1 + J_2 > J^s_{max} \\ 0 & \text{if } J^s_{max} \ge |J_1 + J_2| \ge J^s_{min} \\ \left(J^s_{min} - (J_1 + J_2)\right)^2 & \text{if } J_1 + J_2 < J^s_{min} \end{cases}$$

$$V_d = \begin{cases} \left(|J_1 - J_2| - J^d_{max}\right)^2 & \text{if } |J_1 - J_2| > J^d_{max} \\ 0 & \text{if } J^d_{max} \ge |J_1 - J_2| \ge J^d_{min} \\ k'\left(J^d_{min} - |J_1 - J_2|\right)^2 & \text{if } J^d_{min}/2 \le |J_1 - J_2| < J^d_{min} \\ k'\left((J^d_{min})^2/2 - |J_1 - J_2|^2\right) & \text{if } 0 \le |J_1 - J_2| < J^d_{min}/2 \end{cases}$$

$$J^s_{max} = J^s + \epsilon$$

$$J^s_{min} = J^s - \epsilon$$

$$J^d_{max} = Max(|J^d + \epsilon|, |J^d - \epsilon|)$$

$$J^d_{min} = Min(|J^d + \epsilon|, |J^d - \epsilon|)$$

$$\epsilon = \sqrt{\epsilon_1^2 + \epsilon_2^2}$$

$$J^s = J_1^{exp} + J_2^{exp}$$

$$J^d = J_1^{exp} - J_2^{exp}$$

where

K_n is a optional normalization factor. If it is used, it is set to $1/Max(E_j)$ for E_j calculated as described for the `JNORM` option below.

J_1 and J_2 are the calculated values for the two scalar coupling constants involved the two prochiral atoms.

J_1^{exp} and J_2^{exp} are the experimental values for the scalar coupling constants. These are calculated from the average of the lower and upper bounds specified in the constraint file for the J coupling.

ϵ_1 and ϵ_2 are calculated from the experimental scalar coupling as half the difference of the upper and lower bounds on the coupling constants.

k' is a scale factor to adjust the difference term down to eliminate local minima in E_j.

11.5.3 Syntax for NMR Constraint Command

The `NMRC` command is used to specify the NMR Constraints file along with some of the parameters of the NMR constraint functions.

```
NMRC [UNIT unit]        default: 5
     [FMAX real]        default: 1
     [SLOPe real]       default: 1
     [NOEWeight real]   default: 1
     [JWEIght real]     default: 1
     [KJDIff real]      default: 0.2
     [ECHO]
     [j-normalization-options]
     [noe-proton-averaging-options]
     [clear-options]
     [title-options]

j-normalization-options ::= [JNORmalize  ]
                            [NOJNormalize]

                                 [SUM      ]
noe-proton-averaging-options ::= [AVErage  ]
                                 [NOAVerage]

title-options ::= [TITLe  ]
                  [NOTItle]

clear-options ::= repeat( CLEAr {Jcoupling} )
                        (       {Noe      } )
```

11.5.4 Keywords for NMR Constraint Command

The purpose of the keywords is given in the following table:

`UNIT`
: Unit number associated with the constraint file name (opened previously) If a `UNIT` number is not specified it is assumed that the constraints are to be read from the main input file.

`FMAX`
: Maximum force for NOE constraints. This specifies the point in the constraint function when the function turns flatter, and the force begins to be reduced.

`SLOPE`
: The constant force which applies to the outer region of the constraint function. This value should not be larger than `FMAX`.

`NOEWEIGHT`
: The energy weight assigned to all NOE constraints read in after this point. This weight can also be set in the NOE constraint file.

`JWEIGHT`
: The energy weight assigned to all J coupling constraints read in after this point. This weight can also be set in the NOE constraint file.

`KJDIFF`
: The `KJDIFF` option sets the value of k' in the equation for V_d. The default value gives only two minima for the χ_1 torsion angle in the test case in the Constantine et al paper.

`ECHO`
: If present, all lines read from the constraint file are echoed as they are read, and errors from searching atoms in the PSF will be printed.

`JNORMALIZE`
`NOJNORMALIZE`
: This option controls the normalization of J constraint energy. The default state is normalization off. If it is turned on, then for single J's, CONGEN will calculate the value of E_j over all values of the torsion angle ϕ for each J separately, and it will normalize the energy calculations for each of the J's. In the case of joined J's, there are two possibilities. If the first three atoms or the last three atoms of joined J's are the same and the remaining atom is different, the program will assume that the J's represent the calculation of a prochiral group around an sp^3 center. CONGEN will calculate the maximum of E_j over the range of the first torsion angle, with the second angle being set to 120 degrees plus the first angle. If the three atoms do not match, then the program will calculate the maximum of E_j over all possible values of the torsion angles for both J's. The grid size used for all these calculation is 2 degrees.

`SUM`
`AVERAGE`
`NOAVERAGE`
: When the NOE constraints are specified, you can control whether constraints involving equivalent, degenerate, or non-stereospecifically assigned groups of protons are summed or averaged. The `SUM` and `NOAVERAGE` keyword specifies summing, the `AVERAGE` keyword specifies averaging. This specification will affect all succeeding constraints read in by the program.

`CLEAR`
: Normally, the `NMRC` commands add constraints to the current lists. The `CLEAR` option will clear the named list before new ones are read in. If you specify `NOE`, then the NOE constraints are cleared, and if you specify `JCOUPLING`, then the J coupling constraints are cleared. Both of these keywords may be abbreviated to one letter, and you may specify multiple `CLEAR` options.

`TITLE`
`NOTITLE`
: By default, a title, see See Section 2.12 [Syntactic Glossary], page 12, is read from any constraint file read from any unit other than the default command input. From unit 5,

no title is required. These options can override that default behavior. `TITLE` specifies that a title will be read from the constraint file, and `NOTITLE` specifies that no title will be read.

11.5.5 NMR Constraint Files

The NMR Constraint file actually contains the constraints, both NOE and J coupling. Besides the specification of the constraints, the file also contains specifications for the NOE and J coupling weights as well as the segments involved with any ensemble averaging.

The constraint file is a free format file containing a series of commands. Depending on the setting of the `TITLE` option on the `NMRC` command, the first lines of the constraint file should contain title lines which end with a * on an otherwise blank line. The file is terminated by either an `END` command or the physical end of file. You can put the constraint file in line in the CONGEN input file by specifying `UNIT 5` on the `NMRC` command.

The commands are as follows :

11.5.5.1 TYPE command

The `TYPE` command specifies the type of constraint which follows in the file. For NOE constraints, it is also used to specify whether individual distances should be averaged or summed, see Section 11.5.1 [Theory for NOE Constraints], page 98, for more information. For J coupling constraints, the coefficients of the Karplus equation can be specified.

Syntax

```
TYPE { noe-spec }
     { J-spec   }

                    [ AVErage   ]
noe-spec ::= NOE [ NOAVerage ]
                    [ SUM       ]


               {PHI}
J-spec ::=  {J  } [COEF1 real] [COEF2 real] [COEF3 real]
```

Function

The `TYPE` command specifies the type of constraint file being read, either `NOE` or one of `PHI`, `PSI`, `X1S`, or `X1R`, for NOE distance constraints of J-coupling data respectively. You *must* specify a constraint `TYPE` or the program will not continue execution. You can freely change the `TYPE` specification within a file.

When the `NOE` type is specified, you can control whether multiple proton pairs in a constraint are summed or averaged. The `SUM` and `NOAVERAGE` keyword specifies summing, the `AVERAGE` keyword specifies averaging. This specification takes affect with all succeeding NOE constraint specifications.

The coefficients for the Karplus equation can be changed using the `COEFn` keyword value pairs. `COEF1` is used to change the coefficient on the `cos^2` term; `COEF2` is used for the `cos` term, and `COEF3` changes the final term. Any changes made to these coefficients will apply to all succeeding J coupling constraints within one invocation of the `NMRC` command. They will be reset to (6.4, -1.4, and 1.9) at the start of the `NMRC` command.

11.5.5.2 WEIGHT command

Syntax

```
WEIGHT [J] [Noe] [real]
```

Function

The `WEIGHT` command will define the weights used on succeeding constraints. If the type keywords are specified, then the weight of those constraints will be changed only. If no type keyword is specified, then the current constraint type will be changed.

A convenient way to change weights during the course of simulated annealing protocol is to omit weights from the constraint file, and set them prior to reading the constraint file in.

11.5.5.3 ENSEMBLE Command

Syntax

```
ENSEmble [EXP real] repeat(segid)
```

Function

The `ENSEMBLE` command specifies the use of ensemble averaging. Each segment specified by a segment identifier (`segid`) will be treated as a conformer in the ensemble. Currently, CONGEN does not automatically ignore interactions between conformers in the ensemble, and therefore, you must take steps to separate the conformers and set the non-bonded cutoffs small enough to avoid interactions, see Chapter 6 [Non-bonded Interactions], page 51. Every segment in the ensemble must have identical residue and atom identifiers in the same order for this command to work.

Once the `ENSEMBLE` command is specified, any NOE constraints involving the first segment specified in the `ENSEMBLE` command are automatically replicated for all the other segments in the ensemble. Further, you may not specify constraints involving the remaining segments.

The `EXP` keyword is used to the exponent in the ensemble averaging equation, see Section 11.5 [NMR Constraints], page 97. The default value is 6.

11.5.5.4 CONS Command

Syntax

For NOE constraints:

```
CONS {LOWEr real} {UPPEr real} atom-selection : atom-selection
```

For J-coupling constraints:

```
CONS atom-spec atom-spec atom-spec atom-spec {LOWER real} {UPPER real}

     [COEF1 real] [COEF2 real] [COEF3 real]

atom-spec::= segid resid iupac
```

Function of NOE Constraints

The `CONS` command signals that this line will be a constraint. The syntax of each constraint varies depending on the constraint type as set by the `TYPE` option, see Section 11.5.1 [Theory for NOE Constraints], page 98. In the case of the NOE constraints, you must specify the upper and lower bounds for the distance using the `UPPER` and `LOWER` keywords, respectively, and two `atom-selections` (see Section 11.7 [Atom Selection], page 108) to specify each half of the constraint pair. Each `atom-selection` defaults to not selecting any atoms. If each side of the constraint specifies just one atom, then just plain distances are computed. However, if multiple atoms are specified on either side, CONGEN will use $(1/r)^6$ summing or averaging to compute the distance. The average will be computed over all possible pairs of atoms between the first and second `atom-selections` given for each constraint.

The choice of atoms to include for each constraint will depend on your NOE data. If you can assign each proton or atom explicitly, then only single atoms should be used. If there is signal averaging occurring, then use the multiple atom specification to get a more realistic constraint. This choice is independent of whether or not ensemble averaging is subsequently used. Note that ensemble averaging affects how the constraints are interpreted. See the description of ensemble averaging above for more information.

Here are some examples: Suppose that one has an NOE between two protons which have been uniquely assigned. Then, the following command would specify a constraint between two atoms:

```
CONS  ATOM MOL1 2 HA : ATOM MOL1 4 HB1  LOWER 2.0  UPPER 4.0
```

In this example, there is a distance constraint between the HA of residue 2 in segment MOL1 and the HB1 of residue 4 in segment MOL1 which has a lower bound of 2.0 and an upper bound of 4.0. However, if one wanted to include the $(1/r)^6$ sum of all the equivalent beta hydrogens in residue 4 the command line would look like as follows:

```
CONS  ATOM MOL1 2 HA : ATOM MOL1 4 HB*  LOWER 2.0  UPPER 4.0
```

Function of J Coupling Constraints

For a J-coupling constraint, the syntax is partially order dependent. The keyword argument pairs can be specified anywhere on the command line, but the four `atom-spec`'s must be specified in the correct sequence. The four atoms must be the four atoms involved in the measured coupling constant. The `LOWER` option specifies the lower bound of the measured J coupling, and the `UPPER` option specifies the upper bound. Coefficients for Karplus equation can be changed for this constraint, and any changes will be permanent for all succeeding J's.

11.5.5.5 JOIN command

Syntax

```
JOIN
```

Function

The `JOIN` command signals that the next two J coupling constraints are to be "joined" together, and to use the second form of the J constraint as described in the section, Section 11.5.2 [Theory for J Coupling Constraints], page 100. It is an error for there to be less than two J constraints in the NMR constraint file after the `JOIN` command is specified.

11.5.5.6 END command

Syntax

```
END
```

Function

The `END` command signals the end of the constraint file to be read in.

11.5.5.7 Examples

```
Example of NOE Constraints
*
WEIGHT 1.0
TYPE NOE
CONS  ATOM MOL1 2 HA : ATOM MOL1  4 HB* LOWER 2.0  UPPER 4.0
CONS  ATOM MOL1 3 HN : ATOM MOL1  5 HN  LOWER 3.0  UPPER 4.4
WEIGHT 0.5
CONS  ATOM MOL1 2 HN : ATOM MOL1 10 HN  LOWER 3.0  UPPER 4.0
END
```

```
Example of J-coupling Constraints around the chi 1 angle of leucine.
*
WEIGHT J 1.0
TYPE PHI COEF1 9.5 COEF2 -1.6 COEF3 1.8
JOIN
CONS 1 1 HA  1 1 CA 1 1 CB 1 1 HB2 LOWER 12.9 UPPER 12.9
CONS 1 1 HA  1 1 CA 1 1 CB 1 1 HB1 LOWER 3.375 UPPER 3.375
TYPE PHI COEF1 7.2 COEF2 -2.0 COEF3 0.6
JOIN
CONS 1 1 C   1 1 CA 1 1 CB 1 1 HB2 LOWER 1.4 UPPER 1.4
CONS 1 1 C   1 1 CA 1 1 CB 1 1 HB1 LOWER 9.8 UPPER 9.8
TYPE PHI COEF1 -3.75 COEF2 0.26 COEF3 -0.54
JOIN
CONS 1 1 N  1 1 CA 1 1 CB 1 1 HB2 LOWER -1.3475 UPPER -1.3475
CONS 1 1 N  1 1 CA 1 1 CB 1 1 HB1 LOWER -1.3475 UPPER -1.3475
END
```

Further examples may be found in the '`JTEST*`' and '`NOETEST*`' test cases, see Section 30.10 [Tests], page 266.

11.6 `SHAKE` — Fixing Bond Lengths Or Angles in Dynamics

SHAKE is a method of fixing bond lengths and, optionally, bond angles during dynamics. The method was brought to CHARMM by Wilfred Van Gunsteren, and is referenced in *J. Comp. Phys.* **23**, 327 (1977). When hydrogens are present in a structure, it will allow a five fold increase in the step size if `SHAKE` is used on the bonds.

To use `SHAKE`, one specifies the `SHAKE` command before any dynamics are run. The `SHAKE` command has the following syntax:

```
SHAKE [ BONH [ BOND [ ANGH [ ANGL ]]]]
```

`BONH` specifies that all bonds involving hydrogens are to be fixed. `BOND` specifies all bonds. `ANGH` specifies that all angles involving hydrogen must be fixed. `ANGL` specifies that all angles must be shaken. `BOND` must be specified if angles are fixed.

When the `SHAKE` command is used, it will check that there are degrees of freedom available for all atoms to satisfy all their constraints. Angles cannot be fixed with `SHAKE` if one has explicit hydrogen arginines in the structure as the CZ carbon has too many constraints. This is a general problem for any structure which has too many branches close together.

SHAKE is not recommended for fixing angles. The algorithm converges very slowly in the case where one has three angles centered on a tetravalent atom and the constraints are satisfiable only using out of plane motions.

The use of SHAKE modifies the output of the dynamics command. The number appearing to the right of the step number is the number of iterations SHAKE required to satisfy all the constraints. This number should generally be small.

If atoms are fixed rigidly in place, see Section 11.4 [Fixed Atoms], page 97, then the SHAKE command should follow the CONS command for to prevent SHAKE from shaking deleted bonds. This will make it run more efficiently.

Each time the SHAKE command is executed, the list of constraints is initialized. Thus, if you wish to eliminate the use of SHAKE, specify a SHAKE command with no arguments.

11.7 Atom Selection

Many commands in CONGEN operate on a subset of the atoms in the system. The *Atom Selection* syntax described in this section is generally used to identify the subset.

11.7.1 Syntax of an Atom-Selection

```
atom-selection ::= repeat(token)
```

where token can be:

```
SHOW
ALL
*

[ ATOM ] segid* resid* atomname*
[ CELL ]

RESName resname*
RAMA    resname* atomname*
RANGe   segid1 resid1 atom1     segid2 resid2 atom2
RANGe BYNUm integer integer
BYNUm repeat(integer)
AROUnd  real

                  {GE}
              {X} {GT}
COOR [ABS] {Y} {LE} real
              {Z} {LT}
                  {EQ}
                  {NE}

CONT probe cutoff
SURF probe cutoff
CLEAR
BYREs
ENTEr
OR
AND
NOT
EXCL
EXCH int
```

11.7.2 Interpretation of Atom Selection Tokens

The selection parser operates like a Hewlett-Packard calculator, performing its operations using Reverse Polish notation using an internal stack. Tokens are parsed and performed from left to right. Operations that select atoms based on the structure simply turn on the flags for the atoms selected; the stack manipulation operators can combine this flags in arbitrary ways. When the selection string is completed, the top of the stack is returned as the selection.

All of the tokens followed by a `*` are interpreted using wildcard characters as follows:

`*`	matches any string of characters (including none)
`%`	matches any single character
`#`	matches any string of digits (including none)
`+`	matches any single digit

11.7.3 Selection Tokens

`ALL` and `*` include all the atoms.

`ATOM` selects atoms by name. Likewise, `CELL` selects tags by name where the tag substitutes for the `atomname` in the syntax.

`RESNAME` selects by residue names (eg, GLY, TRP, etc.)

`RAMA` selects atoms by IUPAC name and residue name.

`RANGE` allows for a selection over a range of atoms.

`BYNUM` allows for a selection by number.

`AROUND` will add to the selection all the atoms within the distance specified of the atoms currently selected.

`COOR` will select those atoms whose coordinate component satisfies the given relationship to the number specified.

`CONTACT` and `SURFACE` will compute the accessible contact area or surface, respectively, for each atom and will mark all atoms whose value exceeds or equals the cutoff. The probe gives the size of the probe. If zero is used for the probe, then the current default probe size will be used (typically 1.4 Angstrom).

`BYRES` will include all the atoms in every residue for which at least atom has already been selected.

11.7.4 Operators on Selections

`CLEAR` removes (clears) all the atoms from the current selection.

`ENTER` pushed the current selection into the stack, and initializes the top of the stack to the default value.

`OR` performs a logical OR operation between the top of the stack and next deepest selection, pops the stack twice, and pushes the result onto the stack.

`AND` does the same thing as `OR` except for performing and AND operation.

`NOT` takes the inverse of the top of the stack.

`EXCL` is equivalent to `NOT AND`. In other words, it deletes all the selected atoms from the previous selection on the stack. This operation is provided for simplifying conversions from the previous form of the atom selection syntax.

`EXCH` *i* exchanges the top of the stack with the *i*th deepest element. The top of the stack is numbered 0.

`SHOW` will print the atoms that are currently included at the point in string parsing where `SHOW` is encountered. If you want the list that will be returned to the calling routine, make sure `SHOW` is the last entry in the string.

11.7.5 Examples of Atom Selections

```
CLEAR ATOM * * CA                will include all C alphas in the list.
ALL ENTER CLEAR ATOM * * H* EXCL will include all non-hydrogen atoms.
CLEAR RANGE BYNU 1 100           will include atoms number 1 to 100.

CLEAR RANGE MAIN 1 CA MAIN 10 CA ENTER -
CLEAR ATOM * * H ATOM * * N ATOM * * O EXCL
                        will include all the atoms from CA of reside
                        1 to CA of residue 10 in the segment MAIN
                        except atoms H, N, and O.

CLEAR BYNU 1 3 5 7 9 11 13 15 ATOM SOLV * *
                        will include atoms number 1, 3, 5, 7, 8, 11, 13,
                        and 15, and the SOLV segment.

CLEAR ATOM S1 10 * AROUND 8.0 ENTER -
CLEAR ATOM S2 * * ATOM S3 * * AND BYRES
                        selects all atoms which are in residues which
                        have atoms in segments S2 and S3 that are within
                        8.0 A of residue 10 in segment S1.

CLEAR COOR X GE 0 ENTER -
CLEAR COOR Y GE 0 AND ENTER -
CLEAR COOR Z GE 0 AND ENTER -
CLEAR COOR X LE 5 AND ENTER -
CLEAR COOR Y LE 6 AND ENTER -
CLEAR COOR Z LE 7 AND
                        selects those atoms within a rectangular box
                        between the origin and (5,6,7)
```

All of the atom selections are interpreted using the `SELCTA` routine in ‘`SELCTA.FLX`’. Wildcard interpretation is handled by the `EQSTWC` routine in ‘`STRING.FLX`’.

12 Conformational Search

The `CONGEN` command performs a conformational search over a set of degrees of freedom that you specify. The first section in this chapter provides an introduction to this command. The remaining sections describe how to use the command.

Conformational search can take advantage of parallel processing in a highly efficient manner. See Section 27.1 [Parallel Command], page 227, for information on how to enable parallel processing. In addition, there are some options which affect the efficiency of the processing as described below.

There are a number of commands to assist you in searching conformational space. See Section 12.7 [Support Programs for Conformational Search], page 141, for more details.

N.B. The conformational search is designed to work only with explicit hydrogen or all hydrogen topology files.

12.1 Overview of Conformational Search

The conformational search process is a sampling over degrees of freedom within a macromolecule. With the `CONGEN` command, the term, "degree of freedom" is used somewhat more freely than in the statistical mechanical sense. It means any operation that determines any number of atomic positions (including zero) and which can be iterated at least once over some variable. The reason for this generalization is to allow input, output, and energy evaluation operations into the course of the search in an simple and powerful way.

The sampling process is a series of nested iterations applied over all the degrees of freedom in the order specified by the user. All of the variables are sampled discretely, although there are provisions to solve for certain variables over an continuous range where constraints may be applied.[1] Thus, the computer time required for a search grows exponentially with the number of degrees of freedom. It is easy to set up a run that could run for the age of universe.

There are several different methods for directing the search process. The simplest method is a depth-first search where the program tries every sample in turn using an algorithm that requires a minimum of temporary storage to keep track of its progress. There are also methods for sampling based on the quality of the partial conformations, and these techniques can result in better quality conformations being generated early in the search process. It is also possible to generate random structures. Section 12.1.3 [Overview of Directed Searching], page 116, for more information.

The program as described in the *Biopolymers* 1987 paper was originally designed to search the conformational space of a single polypeptide segment within a protein. The version described here provides that capability in a more general way, so that multiple segments can be searched, the local environment around a segment can be considered, or terminal segments can be sampled.

The degrees of freedom presently implemented can be divided into three categories, those which construct atoms within the system, those which do I/O, and finally, one for the evaluation of the conformations.

There are three degrees of freedom involved with construction; Backbone, Chain Closure, and Sidechain; and together, they can search over a polypeptide segment. In addition, by creating new sidechain topology files, the Sidechain degree of freedom can be adapted for any molecule. See Section 12.3.3 [Sidechain Topology], page 123, for more information. The backbone and chain closure degrees of freedom work together to construct the backbone for an internal polypeptide

[1] For example, the van der Waals avoidance in the sidechain construction will adjust a chi angle selection until a close contact is avoided. Also, the chain closure procedure generates torsion angles over the complete domain of angles. In addtion, backbone and sidechain degrees of freedom can construct atoms using fixed torsion angles.

segment. The sidechain degree of freedom is used for making the sidechains. See the menu below for more description of these degrees of freedom.

There are two degrees of freedom for I/O, `WRITE` and `RBEST`. The `WRITE` degree of freedom writes to a CONGEN conformation file the position of all atoms constructed up to that point in the search along with the latest evaluation of the conformation, see Section 12.3.4 [Conformation File], page 126. This file can be read back by the `RBEST` degree of freedom, it can be scanned for a particular conformation using the `XCONF` command, merged with other conformation files using the `MERGE CG` command (see Chapter 13 [CONGEN Related], page 143), and scanned with the `CMPLOOP` command (see Section 12.7 [Support Programs for Conformational Search], page 141). The `RBEST` degree of freedom is used to read the best conformations from a CONGEN conformation file. By using this degree of freedom, real space renormalization (see H. Scheraga, *Biopolymers* (1983) **22**, 1-14) can be implemented.

Finally, there is the `EVL` degree of freedom. `EVL` is used to evaluate the conformation currently being constructed. Any type of energy manipulation is possible, see Chapter 7 [Energy Manipulations], page 55, but typically, only energy evaluation is done. The `EVL` degree of freedom can also be used for comparing generated conformations against a known structure, so that the theoretical limits of the sampling can be assessed. Finally, the `EVL` option can invoke a user written evaluation function or it can assign a random number to the evaluation of the conformation.

Although CONGEN was written for searching protein segments, it can be applied to arbitrary molecules. The sidechain degree of freedom reads a topology file, see Section 12.3.3 [Sidechain Topology], page 123, which can be used to describe the conformational degrees of freedom in any molecules. The sidechain degree of freedom is capable to searching any subset of the degrees of freedom, and therefore, search protocols like those used for proteins can be executed where the central part of a small molecule can be done exhaustively, and the peripheral moieties can be iteratively searched.

Because long searches are common, CONGEN can periodically save the state of a search in a checkpoint file and restart the run from such a point. In addition, the status of the run can be periodically written to a file which can be typed by the user as the program is executing.

When CONGEN performs a search, it initializes the positions of all atoms involved in the degrees of freedom. This prevents collisions between newly constructed atoms and their prior positions if any. If you are planning several sequential searches, then you should initialize the position of all the atoms involved (using a `COOR INIT` command, see Section 15.2 [Function of Coordinate Manipulations], page 151).

In some cases, it is desirable to repeat a search over a particular degree of freedom. For example, consider the problem of finding structures which satisfy a set of NMR constraints. Because many of the constraints involve sidechain atoms, it is desirable to search sidechains after each backbone. However, since constraints bridge across multiple sidechains, it is desirable to rebuild all the sidechains after each new one is added. CONGEN will support this type of operation in general by examining if atoms in one degree of freedom are reconstructed by a later degree of freedom. If so, then these “overlapping” atoms will be removed just prior to the sampling of any degree of freedom which generates new atomic positions. Note that this approach can be quite inefficienct, but it might be improved if this capability proves to be useful.

There is a limited capability to treat part of the molecule as a rigid body while other parts are being searched. The backbone and sidechain degrees of freedom both have `FIX` options, see Section 12.5.1 [Backbone Degree of Freedom], page 131, and Section 12.5.3 [Sidechain Degree of Freedom], page 133, which specify that atoms be constructed with the same bond lengths, bond angles, and torsion angles that they had when the `CONGEN` command was invoked. This can be used to explore how two domains interact with one another when a linker joining them is flexible.

It is possible to include a cavity formation term in the energy function used by the conformational search. See Section 27.15 [Gepol Command], page 239, for more information.

12.1.1 Overview of the Backbone and Chain Closure

To generate the positions of the backbone atoms, an extension (*Macromolecules* (1985) **18**, 2767-2773) of the local chain deformation and chain closure procedure of Go and Scheraga (*Macromolecules* (1970) **3**, 178-187) is used. Given fixed, oriented endpoints and a chain of bonded atoms containing six freely rotatable torsions, their procedure determines a set of values for the torsion angles that permit the chain to bridge the endpoints. The free torsions are the phi and psi angles so three is the minimum number of residues for which a search over an internal polypeptide segment can be performed (it is presumed that the omega peptide torsion angle is planar and normally trans, although cis peptides are considered as described below).

It turned out that the original Go & Scheraga procedure was overly restrictive, particularly for bridging regular structures like alpha-helicies. To avoid this problem, we have modified the method to permit limited alterations in the bond angles and the procedure is named `CLSCHN`. The main option controlling bond angle variations is `MAXDT` which gives the maximum variation from standard bond angles. Its default value is 5 degrees which improves its performance significantly while incurring a bond angle energy penalty of at most 1 kT per angle. The number of solutions obtained from the chain closure method is always even and has not exceeded eight in our experience with peptides.

The ring in proline creates special problems. The proline ring constrains the phi torsion to be close to -65 degrees; any deviation from -65 degrees distorts the ring. Prior to running `CONGEN`, we determine the minimum energy configuration of the proline ring (specifically, 1,2 dimethyl pyrrolidine) for a range of phi angles (+/- 90 degrees) about -65 degrees using energy minimization with a constraint on phi, and we construct a file (‘`PRO.CNS`’) which contains these energies and the construction parameters necessary to calculate the position of CB, CG and CD of the proline. All of these energies are adjusted relative to a minimum ring energy equal to zero. After a chain closure is performed, we discard any conformations which have a proline phi angle whose energy exceeds the minimum energy by more than the parameter, `ERINGPRO`. Generally, we use a large value for `ERINGPRO`, 50 kcal/mole, so `CLSCHN` does not overly restrict proline closures. We handle cis-trans peptide isomerization by trying all possible combinations of cis and trans configurations. The user has complete control over which residues can be built in the cis isomer. Since there are only three residues involved in the chain closure, this results in no more than eight (2^3) attempts at chain closure.

The backbone search of an N residue segment begins by using backbone degrees of freedom to sample the free torsions of N-3 residues and then using the chain closure degree of freedom to close the chain. As the free torsions are sampled, we can discard any segment if the end of the constructed chain is too far from the other framework end for closure to be possible. See Section 12.5.1 [Backbone Degree of Freedom], page 131, for a description of the `CLSA` and `CLSD` options which control this process. The determination is made by calculating the distance between the last atom constructed and the other fixed endpoint and comparing that to the distance spanned by m peptides with all torsions being trans and all bond angles increased by `MAXDT`, where m is the number of peptides still to be constructed.

The direction of backbone construction is arbitrary, although the endpoints of the search are conserved regardless of the order. The N-terminus of the internal segment is anchored on the peptide nitrogen; the C-terminus is anchored on the alpha carbon. When the construction direction is from the N terminus to the C terminus, the first torsion to be sampled in a residue is the omega angle (which normally is sampled just at 180 degrees, and sampled at 0 degrees and 180 degrees for prolines). It determines the alpha-carbon and the peptide hydrogen positions. The phi angle determines the position of the carbonyl carbon and the beta carbon of the sidechain; and finally, the psi angle determines the carbonyl oxygen and peptide nitrogen of the next residue. When the construction is in the reverse direction; the psi angle determines the peptide nitrogen; the phi angle determines the carbonyl carbon of the preceding residue, the peptide hydrogen, and the beta carbon; and the omega angle determines the position of the preceding residue's alpha carbon and carbonyl oxygen.

Rather than treating each of the three torsion angles in a amino acid residue as three separate degrees of freedom, we combine them into a single degree of freedom. This permits the use of Ramachandran type plots to limit the range of phi, psi values to those that are energetically acceptable and found in known structures.

To determine the allowed phi,psi angles, the `CONGEN` command uses energy maps. These energy maps are stored as files, see Section 12.3.1 [Backbone Maps], page 122, and they are expressed in tabular form with entries composed of omega, phi, and psi angle values along with the energy for that angle combination. Thus, any arbitrary criterion may be used in place of the energy in these maps. There are three different types of maps, one for glycine, one for proline, and one for the other amino acids which are modeled by alanine. Typically, the glycine and alanine maps are computed by modeling a dipeptide and using the van der Waals energy, whereas the proline maps is computed using a dipeptide, but the energy in the map is the sum of the van der Waals energy plus the ring energy. The actual values for phi and psi in these tabulations are usually multiples of 15 degrees or 30 degrees with all possible combinations of angles present. Thus, these maps determine the sampling grid. For the alanine and glycine map, phi and psi both range over -180 degrees to 180 degrees, and the omega angle can be either 0 or 180 degrees. Normally, the omega angle of 0 degrees is not used. In the proline map, the range of phi angles is -150 degrees to 30 degrees; psi goes from 0 degrees to 360 degrees; and omega has values of 0 degrees and 180 degrees.

Each backbone degree of freedom can specify its own map. However, in most applications, each backbone residue will use the correct map for its type (proline, glycine, or alanine), and the grid spacing in all the maps will be the same. Therefore, default maps are typically specified for each type of amino acid, and the user can override these maps for individual residues. Fortran unit numbers for default maps for glycine, proline, and other amino acids are specified with the global variables; `GLYMAP`, `PROMAP`, and `ALAMAP`; respectively.

The particular values of torsion angles used for generating conformations is determined by these maps and the so-called `EMAX` options. The maps specify all the possible angles. The `EMAX` options restrict these sets as they specify the maximum allowed energy relative to the minimum energy value found in each map. The global options; `GLYEMAX`, `ALAEMAX`, and `PROEMAX`; specify the selection of the default backbone maps, see Section 12.4 [Global Options for Conformational Search], page 126. `GLYEMAX` specifies that for the glycine map; `PROEMAX` for the proline map; and `ALAEMAX` for the alanine map. The backbone degree of freedom option, `EMAX`, specifies the allowed energy for an individual backbone degree of freedom. For example, when using the alanine map with a sequence of alanines and a value of `ALAEMAX` of 2.0 kcal/mole, all the conformations generated will have phi, psi angles corresponding to only right-handed alpha-helices or beta-sheets. For a value around 5 kcal/mole, phi, psi angles for left-handed alpha-helices will also be selected. If values for the `EMAX` options are set to very large values then the entire phi, psi space will be sampled.

D amino acids are indicated by the presence of the word, "`D`", in the residue attributes. These residues are handled by inverting all torsion angles for the backbone maps and for the proline constructor files.

For more details on the commands which implement these degrees of freedom, see Section 12.5.1 [Backbone Degree of Freedom], page 131, and Section 12.5.2 [Chain Closure], page 133.

12.1.2 Overview of Sidechain Degree of Freedom

Given a set of backbone conformations, it remains to generate a set of side chain atom positions for each of the backbone conformations. Before we explore the problems inherent in side chain generation, we describe the side chain atom placement.

As with the backbone atom placement, the side chain atoms are positioned based on free torsion angles. The side chain torsions are processed from the backbone out as each succeeding atom requires the position of the previous atom for its placement. The sampling interval of each torsion (the option `SGRID`) can be either some fixed number of degrees or the period of the torsion energy. When the latter is used, the sidechain torsions will be at minima in the torsion angle potential involving the free atom and its antecedents. It is also possible to modify the sampling to avoid van der Waals contacts (`VAVOID` option). It is common for one free torsion to generate the position of more than one atom because of side chain branching, non-rotatable bonds, and rings. For example, although tryptophan has 11 side chain atoms to be placed, it has only two free torsion angles. Also, certain torsions have symmetry so we can reduce the sampling necessary. Finally, a search of the surrounding space is made for any constructed atom to see if there are any close contacts with a repulsive energy greater than `MAXEVDW`, and if so, that structure is eliminated.

Although this data structure was designed for amino acids, it can be applied to an arbitrary molecule. The only prerequisite is the presence of a few known atomic position upon which the remaining atoms can be constructed.

The information needed for side chain construction is stored in a side chain topology file, see Section 12.3.3 [Sidechain Topology], page 123.

Given these specifications for generating side chain atomic positions, we need to introduce a protocol that generates only a limited number of conformers. The procedure analogous to the backbone generation procedure would result in a series of nested iterations over each chi torsion angle with the number of levels being equal to the sum of the free torsions in all the side chains of the peptide segment. The large number of free torsions in the side chains and the absence of a connectivity constraint, such as exists for the backbone, result in an enormous number of possible sidechain conformations. Consequently, such a direct approach is not feasible except in limited cases.

However, the situation is not that bleak. First, the backbone construction process provides the position of CB which gives a strong bias to the side chain orientation. Thus, an acceptable course of action is the generation of only one sidechain conformation for each backbone conformation. We must strive to make this one conformation the lowest energy possible for the given backbone. Second, because the side chains close together in sequence frequently are not close together in space, and therefore, do not interact strongly, it is a reasonable approximation to treat the side chains quasi-independently. Instead of finding all combinations of side chain atomic positions, we can handle the side chains sequentially so the time required for side chain placement increases linearly, rather than exponentially, with the number of residues.

In order not to limit the options for using the program, six possible methods for generating side chain positions have been implemented. These are specified using the `SIDEOPT` option in the sidechain degree of freedom. All of these methods discard conformations which have any repulsive contacts exceeding `MAXEVDW` in van der Waals energy. The first two methods described, `ALL` and `FIRST`, assume no quasi-independence of the sidechains whereas the others do.

The first method, `ALL`, generates all possible conformations by a series of nested iterations over every sidechain as described above. The second method, `FIRST`, uses the same algorithm as `ALL` except that all the iterations terminate when the first conformation for all the sidechains has been found. This method is useful for determining if a backbone conformation will accommodate the sidechains when details about the sidechain energetics are not required.

The next three methods all depend on a function which evaluates the side chain positions as they are generated so that the best ones can be selected. "Best" is defined as the conformation whose evaluation function is numerically smallest. Two evaluation functions are currently provided, one based on positional deviations, and one based on the CHARMM energy function. The evaluation function based on positional deviations is present for testing CONGEN as it provides a means for determining the limit of CONGEN's ability to generate a known structure. If coordinates are present for the peptide gap, this evaluation function will determine the RMS shift between a generated side chain conformation and the initial coordinates. The second evaluation function computes the CHARMM energy of the sidechain atoms omitting the bond and bond angle energies because the generation procedure does not vary either of these two terms. At present, either the r dependent dielectric or the constant dielectric for the electrostatic energy is used. The other electrostatic calculations, see Chapter 6 [Non-bonded Interactions], page 51, are not available.

The `INDEPENDENT` method assumes that the side chains in the peptide chain being generated do not interact with one another. The atoms of each side chain are placed independently, with those of the other side chains in the peptide being ignored; interactions with all other atoms in the system are included. The conformation which has the lowest value for the evaluation function is selected for each side chain. When the RMS evaluation function is used, this method gives the optimum conformation, though it may be sterically inappropriate. Thus, it cannot be used when the energy is the evaluation function unless the possibility of large repulsive van der Waals is not important.

The method, `COMBINATION`, begins by generating a small number of the best side chain conformations for each side chain independently, as above. Then, these side chain conformations are assembled in all possible combinations, and those combinations which do not have bad van der Waals contacts are accepted. The number of conformations saved for each side chain must be small to avoid a combinatorial explosion.

The `ITERATIVE` method starts with an energetically acceptable side chain conformation for all the side chains. This conformation is generated, if possible, using the `FIRST` method (see above). Starting with this conformation, we regenerate all the possible positions for the side chain atoms of the first residue, and select the conformation with the lowest energy. We also save the value of the evaluation function. This regeneration is done with all the other side chain atoms present so we can account for their effect. The process is repeated sequentially for the rest of sidechains in the gap. We then return to the first residue and go through the process again until the energies of the side chain atoms do not change or until the number of passes reaches an iteration limit. This method has the virtue that only one conformation is generated per backbone conformation, and it is an energetically reasonable one. However, if there are significant interactions between the sidechain atoms, the first part of the process will bias the iterative process toward the initial side chain arrangement selected, and we may miss the lowest energy side chain conformation.

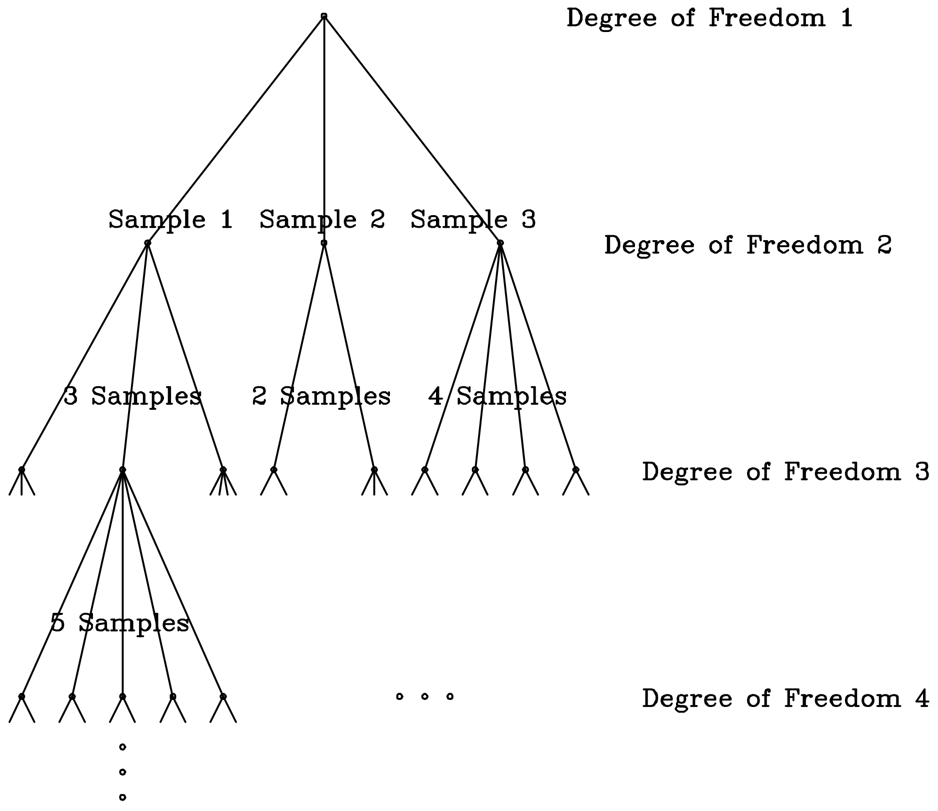
The `FIXED` method is used to construct sidechains in a fixed conformation. When this method is specified, the program calculates the construction bond lengths, angles, and torsions for all atoms in the degree of freedom from the starting coordinates, and will generate just one sample using those values. If there are van der Waals overlaps, then no conformation will be generated.

With any of the methods described above, the `CONGEN` command can apply any of the minimization algorithms to the generated conformations before they are written out for further analysis. Minimization provides an ability to reduce the small van der Waals repulsions that are inevitable with coarse torsion grids used.

See Section 12.5.3 [Sidechain Degree of Freedom], page 133, for more information on this degree of freedom.

12.1.3 Overview of Directed Searching

The process of conformational search can be viewed as a search of tree, where a tree means a graph with no cycles. See the following figure:



In this analysis, the root of the tree represents the system at the beginning of the search process, where no degrees of freedom have been sampled. The next level represents the search process after the first degree of freedom has been sampled. In the above figure, there are three nodes at this level which signifies that the good conformations were found. The third level in the tree represents the effects of sampling the second degree of freedom, and so on.

Each node in the tree represents the result of a particular set of samplings for the degrees of freedom. Leaf nodes are those nodes which exist at the very end of the tree, and represent complete conformations, where every degree of freedom has been completely sampled. Nodes in between the root and leaves are partial conformations. Any node which has been sampled is called an expanded node, and any node which has not been sampled is called an open node.

The goal of a conformational search is to find the lowest energy leaf node. An exhaustive search of the tree can be programmed easily by simply following down the leftmost branch of the tree until either a leaf is hit, or a conformation which is blocked because of bad contacts or other problems. The program then backs up to the previous level, and examines the next node over, and again searches down towards the leaves. Effectively, the search proceeds from left to right across the entire tree, and every leaf is eventually generated. This is a depth-first search. Because each degree of freedom increases the number the nodes by an approximately constant factor, the time for an exhaustive search is exponential in the number of degrees of freedom.

In principle, knowledge of the path through the search tree to the best leaf would make it theoretically possible to find the best leaf in time linear with the number of degrees of freedom.

Such knowledge is generally not available in advance. However, in some situations, information about the partial conformations can provide a guide towards finding the best leaf. For example, if CONGEN is used to reconstruct the backbone of a protein from the alpha carbon coordinates, then the RMS deviation of the partial conformations from the known alpha carbon coordinates is an excellent guide to a good quality fit.

The use of information about partial conformations is the basis for the directed search options. When these options are used, energies or RMS deviations for conformations at each node in the tree are calculated. The program maintains a sorted index of the open nodes in the tree, and it samples degrees of freedom in an order that depends on the order of the energies or RMS deviations.

The evaluations of the open nodes can be affected by the use of the `EINHERIT` and `EIMMEDIATE` keywords for particular degrees of freedom. `EINHERIT` causes all nodes expanded for a particular degree of freedom to inherit the evaluation of their father. `EIMMEDIATE` sets the evaluation of nodes at this level to the smallest possible value which forces the program to expand nodes at this level ahead of all others. This is useful when you do not want the energies or RMS's of this degree of freedom to influence the directed search.

Three selection strategies exist at present. The first strategy, the evaluation strategy which is selected by the `EVAL` keyword, always selects the lowest energy or RMS deviation node. It has a serious drawback when a degree of freedom results in conformers whose energy or RMS deviation is raised. In that case, the program will first expand all nodes before that degree of freedom before moving on towards the leaves.

The second strategy, the deepening evaluation strategy which is selected by the `DEVAL` keyword, is intended to circumvent the drawback of the `EVAL` strategy. Here, CONGEN maintains a sorted index of open nodes for each degree of freedom. When the search begins, all the indices are empty except for the first degree of freedom which contains just the root node. The program then cycles through each degree of freedom, and selects the lowest energy or RMS deviation node at that level for expansion. If there are no open nodes available, or if the search reaches the leaves, then it cycles back to the first degree of freedom. This cycling forces the program to make progress toward the leaves.

The third strategy, the mixed method selected by the `MIX` keyword, is a combination of `EVAL` and `DEVAL`. The `MIX` strategy maintains both types of indices, and simply alternates between each rule for the selection of nodes to expand. So far, it is the best directed search strategy.

It is possible to use the directed search methods to generate random structures. The trick here is to use random numbers for the evaluation of nodes rather than energy values. There are two possible approaches to the generation of random structures, generating multiple conformations in a single CONGEN run or making multiple runs of CONGEN with each generating one conformation. In the first case, the `EVAL` search option generally leads to more variation in the structures, although there is substantial overlap of structures. In the second case, the `DEVAL` option should be used because it is faster for each run.

The directed search capabilities are still under active development, and suggestions for improvement are always welcome.

When the directed search methods are used, the program maintains a search tree in memory. Since each node requires several hundred bytes of memory, it is impractical to keep the entire search tree. Thus, there is a tree pruning mechanism which can be tailored to periodically eliminate nodes that are either unnecessary or unlikely to be sampled in a reasonable amount of time given the search strategy in use. See Section 12.4 [Global Options for Conformational Search], page 126, for a description of the `TREE` options.

12.1.4 Parallel Processing

Because individual conformations in the search tree can be processed independently of one another, CONGEN can operate on different nodes simultaneously. This allows the program to use multiple CPU's to speed the progress of a search.

In order to minimize the interference between processors, CONGEN partitions its activities into three nearly disjoint operations. The most time consuming operation is the sampling of degrees of freedom. This operation takes one partial conformation, samples it, and generates new partial conformations. The second operation is the decision making one, deciding what conformations should be sampled next, and in addition, taking care of the search tree. The third operation is taking new conformations from the sampling operation and putting them into the search tree.

CONGEN maintains two queues in order to minimize the contention between these operations. The first is the work queue (`cg_work_q`) which stores conformations to be expanded. The second is the new node queue (`cg_new_node_q`) which holds new conformations for insertion into the search tree.

In order to obtain maximum parallelism, the work queue should be a large as possible. However, for directed searches, it is best to keep the tree up to date so that decisions about what to explore next can be made with the most information. Therefore, directed searches have a requirement to keep the work queue small. There are two variables which control the size of the queue, `QMIN` and `QMAX`. `QMIN` specifies target value for the minimum size of the work queue, and `QMAX` specifies the target value for the maximum size of the work queue. CONGEN automatically sets these variables depending on the type of search being performed, but they can be changed if you wish.

There is one significant pitfall with CONGEN running in parallel, non-determinism. Since one cannot predict when a processor will finish sampling a conformation, the order in which the search tree is traversed will vary from run to run. In the case of exhaustive searches, this doesn't matter because all runs will produce the same set of conformations, albeit in different orders.

However, in the case of a directed search, the decisions on which node to expand next is made based on the contents of the search tree at the time. Since the contents will vary from run to run, the searches will all be different. It is not known as yet if all the results will be consistent. In simple trials where the peptide backbone is reconstructed from the alpha carbons, consistent results were observed.

12.2 `CONGEN` Command Syntax

```
{ CONGEN } repeat(global-options) repeat(dof-commands)
{ CGEN   }

global-options ::=

    GLYMap unit-number
    ALAMap unit-number
    PROMap unit-number
    PROCons unit-number
    STUNIT unit-number
    [GLYEmax energy] default: 100
    [ALAEmax energy] default: 100
    [PROEmax energy] default: 100
    [ERINgpro energy] default: infinity
    [HBCG hbond-spec END]
    [NBCG nbond-spec END]
    [EIGNore EVDW]
    [COOR SAVE END]
    [MAXLeaf int]
    [MAXNode int]
    [DEBUg int]
    [NOSOPT]
    [QMIN int]
    [QMAX int]
    [SEED int]
    [EPCOns real] default: infinite
    [EJCOns real] default: infinite

            { DEPTH                             }
            { BREADTH                           }
            { RDEPTH                            }
    [SEARCH { { EVAL  }                         } END]
            { { DEVAl } [evaluation-option-word] }
            { { MIX   }                         }

    [TREE [PRTFRQ int] [LIMIT int] [REDUction real]
          [TOPSAVE int] [SDSAVE real] END]

    [STATUS [UNIT unit-number] [SETPRN] END]

    [CHECKPOINT [UNIT unit-number] [NODEfreq int] [TIMEfreq int]
                [[NO]FLUSH]
                END]

    [RESTart UNIT unit-number END]

    [PBE [NEWB] [SOLV [VACUum real]] [ONLY] END]
         [FIXB]


                { BACK backbone-options          }
                { CHAIn chain-closure-options    }
                { SIDE sidechain-options         }
dof-command ::= { RBESt rbest-options            } [dof_options] del
                { WRITe write-coordinate-options }
                { STATus status-options          }
                { EVLuate evaluate-options       }

dof_options ::= [MAXNode int] [[NO]EINHERIT] [[NO]EIMMEDIATE]

backbone-options ::=
    STARtres segid resid
    [LASTres segid resid]
    [MAXEvdw real]  default:100
    [CISTrans]
    [ALLCistrans]
    [FORWard]
    [REVErse]
```

```
         [GRID real]     default:30
         [NOTERSymmetry]
         [CLSA segid resid iupac]
         [CLSD [DELTa] real]
         [MAXDt real]
         [MAP unit]
         [EMAX real]
         [FIX]

     chain-closure-options ::=
         STARtres segid resid
         [MAXDt real]  default:5
         [MAXG real]  default:100
         [MAXEvdw]    default:100
         [CISTrans]
         [ALLCistrans]

     sidechain-options ::=
         repeat( STARtres segid resid [LASTres segid resid])
         repeat( CLUMP clump-spec )
         repeat( [MAXEvdw real] )  default:100
         repeat( [SGRId { real                      } ] )
                 [      { MIN                       } ]
                 [      { AUTO                      } ]
                 [      { SELECT repeat({real}) END } ]
                 [      {               {MIN }      } ]
         repeat( { VAVOID   } )
                 { NOVAVOID }
         repeat( { SYMMetry   } )
                 { NOSYmmetry }
         [SIDEOPT sidechain-option-word]
         [NCOMb number-of-combinations]
         [EVAL evaluation-option-word]
         [MAXSideiter integer]

                       { ALL           }
         clump-spec ::= { word          }
                       { [word]:[word] }

                                   { FIRst       }
                                   { INdependent }
         sidechain-option-word ::= { All         }
                                   { Combination }
                                   { ITerative   }
                                   { FIXed       }

                                    { Energy }
         evaluation-option-word ::= { RMs    }
                                    { Wrms   }
                                    { RAndom }

     rbest-options ::=
        UNIT unit
        [NBESt int    ]  Default: 100
        [MAXEvdw real ]

    write-coordinates-options ::=
        CUNIt unit
        [MAXCOnf integer]  default: 2**30
        [REF COMP]
        [CUT real]
        [MINCUt real]
        [MAXCut real]
        [FILTER]

    evaluate-options ::=
        [MINI minimization-commands END]
        [RMS                           ]
        [USER                          ]
        [RANDOM                        ]

        [NPRInt integer]  default 0
        [CUT real]
        [MINCut real]
        [MAXCut real]
        [FILTER]
```

Notes:

1. Global options may be placed anywhere outside of degrees of freedom.
2. The order for degrees of freedom is the order of the search.
3. A *title* MUST follow the command.
4. `CGEN` is a synonym for the command, `CONGEN`.

12.3 `CONGEN` Input and Output Files

CONGEN has a large requirement for I/O. All of the files are specified via unit numbers in the `CONGEN` command, so `OPEN` statements are generally needed to set these up. See Section 3.4 [Open Command], page 40, for more information.

12.3.1 Torsion Angle Maps

The torsion angle maps are simple formatted files containing a set of record giving the energy for each possible value of `omega`, `phi`, and `psi`. The format of the file is as follows:

```
    title
    count                             (I5)
    omega,phi,psi,etrans,ecis         (3F10.0,2F15.0)
```

where the `count` is the number of entries in the file. The `omega`, `phi`, and `psi` torsion angles are in degrees, and the energies, `etrans` and `ecis`, are in kcal/mol. Torsion angles are selected using the minimum of `etrans` and `ecis`.

A number of standard maps are available in '`CGDATA:`'. The files are named as '`EMAPresnn.OMP`' where '`res`' is one of '`ALA`', '`GLY`', or '`PRO`'; and '`nn`' is one of '`30`', '`15`', '`10`', '`5`', or '`MN`'. The numeric values are used for files which contain uniform grids over the torsion angles. MN stands for a minimal map which has met with only limited utility in our experience (see the Biopolymers paper for details). When using the 'MN' files, be sure to specify the EMAX variables (see Section 12.4 [Global Options for Conformational Search], page 126) all to zero. The energy values in the ALA and GLY maps are computed for alanine and glycine dipeptides, and consist of just the van der Waals energy for those dipeptides. The trans value (etrans) is computed for the second peptide being trans; the cis value (ecis) is computer for the second peptide being cis. In the case of the proline maps, the energy in the file is the sum of the van der Waals energy and the internal energy of the ring as taken from the proline constructor file, see Section 12.3.2 [Proline Constructors], page 123.

The 'makefile' in '$CGP/emap' can be used to generate these files. See Section 29.1.8 [Emap], page 255, for more information. The files are read by the subroutine, RDEMAP.

12.3.2 Proline Constructor File

The construction of prolines is more involved than the other amino residues because of the ring. The approach used in CONGEN is to precompute a table of proline geometries based on the particular value of the phi torsion angle. When a ring with a particular phi angle is required, the table is linearly interpolated to find the geometry to use.

The geometries are determined using a model of just the ring with two terminating methyl groups, specifically, 1,2-dimethyl pyrrolidine. The geometry and energies for particular phi angles are computed by constructing the ring with the given phi angle, setting a torsion angle constraint (see Section 11.2 [Harmonic Dihedrals], page 96), with a high force constant. Then, the ring is minimized, and the actual value of phi, and the bond lengths, angles, and torsions needed to construct CB, CG, and CD are used. In addition, the energy of system e, the van der Waals energy evdw, and constraint energy ec are stored in the table.

This table is stored in the proline constructor file and has the following format:

```
title
count                                           (I5)
phi,e,evdw,ec,(bond(i),theta(i),phi(i),i=1,3)  (13F10.5)
```

where count is the number of table entries. phi is the phi backbone torsion angle after minimization. e, evdw, and ec are given above. The triples of bonds, angles, and torsions given the construction data needed to construct CB, CG, and CD, using internal coordinates, C-N-CA-CB, N-CA-CB-CG, CA-CB-CG-CD, respectively.

A proline constructor file good for general use is stored in 'CGDATA:PRO.CNS'. See Section 29.1.8 [Emap], page 255, for a description of its construction. The subroutine, RDPROCONS, is used this file.

12.3.3 Sidechain Topology File

The sidechain topology file describes how sidechain atoms are to be constructed. It is assumed that backbone atoms N, CA, and CB have been constructed prior to sidechain construction. The construction of sidechain is broken into clumps of atoms, where a single free torsion angle determines all the atoms in the clump.

The sidechain topology file may be used for limited conformational searches of any molecule. As long as one has a core of atoms whose positions are either known or easily constructable, one can define a description of how to construct the remaining atoms while sampling over all the rotatable torsion angles.

The sidechain topology file is a free format file containing a series of commands which fill the sidechain topology data structure used by CONGEN command. This data structure consists of a set of residues where each residue is made up of a set of free torsions (called clumps). Each clump has a rotational symmetry number associated with it as well as an option identifier for the clump. The identifier is useful when one is searching over a subset of clumps.

The clumps consist of a set of atom specifications. Each atom specification has the IUPAC names of four atoms arranged to form a constructor as used in the internal coordinates (see Chapter 14 [Internal Coordinates], page 147), and the bond length, bond angle, and torsion angle used in that constructor.

There are two sidechain topology files available, '`CGDATA:TOPCGEN3.INP`', for explicit hydrogen topology files, and '`CGDATA:TOPALLHCG.INP`', for all hydrogen topology files. The subroutine, `STREAD`, is used to read this file.

The commands read in the sidechain topology file follow closely to the data structure. The commands are as follows:

12.3.3.1 `RESIDUE` command

Syntax

```
RESIdue res [SPECIAL]
```

Function

The residue command specifies the start of a new residue specification. The residue name given is matched against the residue names in the PSF (see Section 2.5 [Data Structures], page 7), when a sidechain degree of freedom is processed. The `SPECIAL` option is used to signify a sidechain that requires special processing, but presently, it is not used by the program. A `RESIDUE` command must be specified before any clumps.

12.3.3.2 `CLUMP` command

Syntax

```
CLUMP int [NAME word]
```

Function

The `CLUMP` command specifies the start of a new clump within a residue. A clump command must precede any atom specifications.

The integer operand specifies the rotational symmetry of the clump. E.g., a symmetry of 1 means the full 360 degree range of the torsion must be sampled; a symmetry of 3 means only 120 degrees must be sampled. The optional name is used to identify the clump. In the absence of a specification, the name is taken to be the number of the clump within the residue.

12.3.3.3 `ATOM` command

Syntax

```
                                         { FREE     }
ATOM iupac iupac iupac iupac real real { real     }
                                         { ADD real }
```

Function

The `ATOM` command specifies how a particular atom is constructed. The first four operands give the IUPAC name for the atom. Prefixes may be used with this IUPAC names, and they are interpreted in the same way as residue topology file names are, see Section 3.1.5.1 [Linkage Atom Naming], page 28. An IUPAC name of `N` is processed specially, to wit, if it is not found within the residue, then `NT` is searched for.

The next two operands specify the bond length in Angstroms and the bond angle in degrees. If specified as 0, then CONGEN will search the parameter file (see Section 2.5 [Data Structures], page 7), for the equilibrium values. In the files provided with CONGEN, the rings (i.e. His, Phe, Trp and Tyr) have the bond lengths and angles explicitly specified; their values are determined using energy minimization with constraints to ensure planarity and symmetry. (The CONGEN parameters for bond lengths and angles are not perfectly consistent for these rings, which result in small deviations from symmetry and higher energies.)

The torsion angle is specified in the final parameter as either `FREE`, meaning that it is the degree of freedom for the clump; `ADD` *real*, when the real number is added to the value of the degree of freedom (the `FREE` value); or just a number which is used directly as the torsion angle. All angles are in degrees.

The order of atom constructors is important in that the first three atoms in any constructor must be present when the fourth atom is built. Thus, the order must reflect construction from the backbone out.

12.3.3.4 `PRINT` command

Syntax

```
PRINT { ON  }
      { OFF }
```

Function

This command controls whether the topology file commands are printed as they are read. The default is no print.

12.3.3.5 `COPY` command

Syntax

```
COPY { res } [INVERT]
```

Function

This command will copy the information about one residue into the description of a second residue. The `INVERT` option will cause all the torsion angles specified in the non-free atom constructor to be negated. This results in an inversion of chirality for the residue, and is currently used for constructing D amino acids. The `COPY` command is useful when two residues have either similar or identical sidechains, and one wishes to avoid the problems with duplicating data that should be the same. The effect of the `COPY` command is the same as if the commands for the copied residue were inserted when the `COPY` command is specified.

12.3.3.6 `END` command

Syntax

```
END
```

Function

This command terminates processing of sidechain topology commands.

12.3.4 Conformations File

The conformations file, which is written by the Write Degree of Freedom (see Section 12.5.5 [Write Degree of Freedom], page 135), stores the list of conformations generated by the `CONGEN` command. This file is read and written by subroutines in the file, '`CGS:CGIO.FLX`'. The file is unformatted, has a typical extension of '`.CG`', and has the following format:

```
HDR,ICNTRL(20)      ! ICNTRL(1)=NRES, ICNTRL(2)=NATOM, ICNTRL(3)=NCONS
NTITLE,TITLE
IBASE               ! Taken from PSF
RES                 ! Taken from PSF
TYPE                ! Taken from PSF
CONSP               ! List of atoms constructed.
! The following record is repeated for each conformation plus one more
! at the beginning of the file for the initial coordinates.
NTITL,((TITLE(I,J),I=1,10),J=1,MIN(10,NTITL)),
      (X(CONSP(I)),Y(CONSP(I)),Z(CONSP(I)),I=1,NCONS),TOTE
```

The first six records in the file provide enough information to restore coordinates from a conformation in a structure, and they also allow the program, `CMPLOOP`, see Section 12.7 [Support Programs for Conformational Search], page 141, to differentiate between backbone and sidechains. The title record is taken from the *title* specified after the CONGEN command.

The remaining records hold conformations. The first conformation stored is not a real conformation, rather it is the coordinates of the constructed atoms prior to the beginning of the search, the so-called reference set. The variable, `TOTE`, stores the value returned by the last `EVL` degree of freedom (see Section 12.5.6 [Evl Degree of Freedom], page 136), or if no `EVL` was executed, then the total energy for the sidechains as determined in the Sidechain Degree of Freedom, (see Section 12.5.3 [Sidechain Degree of Freedom], page 133), or 0.0 otherwise.

12.3.5 Status File

The status file is a formatted file that reports the time of day when the file was written, the number of times the end of the search was reached, and statistics for open nodes at each level of the search.

12.3.6 Checkpoint File

The checkpoint file is an unformatted file which stores the complete state of a search. All search tree nodes are stored as well as some global information about the search, and various local variables used to control the process. The checkpoint file is necessary for restarting a conformational search which has terminated for either planned or unplanned reasons.

12.4 Global Options for the CONGEN Command.

The global options control aspects of the entire searching process. See Section 12.2 [Syntax of Conformational Search], page 119, for the syntax of the global options.

12.4.1 Search Method Options

The conformational search method specifies how CONGEN explores the conformational space dictated by the degrees of freedom. The `SEARCH` global option is used to specify which method is used. If no `SEARCH` option is specified, then the program uses the depth-first search method. The directed search methods are still under development, and are subject to more caveats than the rest of the program. See Section 12.1.3 [Overview of Directed Searching], page 116, for more information.

If the `DEPTH` option is specified, the program does a depth first search of the conformational space. The program tries to go down each branch of the search tree until it is either blocked or reaches the leaves. It then backs up to the closest open node to continue its search. This method is the most space efficient, and should be used for shorter loops where a complete search is feasible.

The `RDEPTH` option is similar to the `DEPTH` option, except that the children of a node expansion are randomly permuted before being put into the search tree. Effectively, it randomizes the order of search, but is still exhaustive and space efficient.

The `BREADTH` option specifies the use of a breadth-first search. Here all open nodes for one degree of freedom are expanded before the next degree of freedom is processed. This method is the worst case search method from the space utilization aspect, but was included for completeness.

The `EVAL`, `DEVAL`, and `MIX` options all specify different types of directed searches. These searches can be directed by four different evaluation criteria; `ENERGY`, `RMS`, `WRMS`, and `RANDOM`. The `ENERGY` criteria is the calculation of the energy, and it is applied to all of the atoms constructed at the level of the node. The `RMS` criteria is the Root Mean Square deviation of the constructed atom coordinates with respect to the coordinates of those atoms at the start of the CONGEN conformational search command. The `WRMS` keyword is mnemonic for *W*orst *RMS*, and directs by the worst match rather than the best. The `RANDOM` criteria is just a random number between 0 and 1. It is most useful when random structures are to be generated using the directed search. See Section 12.4.9 [Miscellaneous Global Options], page 131, for a description of the `SEED` keyword for setting the seed for the random number generator.

The `EVAL` option specifies the evaluation directed search. Here, the program always expands the node which has the lowest evaluation criteria.

The `DEVAL` option specifies the deepening evaluation strategy. Each degree of freedom is selected in turn, and the open node with the lowest evaluation at the level is selected for expansion. This method drives the search toward the leaves, but generally does not yield high quality results.

The `MIX` option specifies the mixed strategy, which alternates between the `EVAL` and `DEVAL` methods. Currently, it is the best directed search strategy.

The options, `EINHERIT` and `EIMMEDIATE`, control evaluation and expansion for specific degrees of freedom during a directed search. These options can be specified for any degree of freedom. The option, `EINHERIT`, specifies that the value of nodes generated for a degree of freedom inherit their values from their parents.

The option, `EIMMEDIATE`, specifies that nodes generated from a degree of freedom will be expanded immediately. This is done by setting the value of the node to the largest negative number. The effect is to bypass the degree of freedom from the selection process. This option is very useful in energy directed searches where one wants to direct the search based on backbone and sidechain energies together. In such cases, the `EIMMEDIATE` option is specified for all backbone and chain closure degrees of freedom. See Section 12.6.3 [Energy Directed Search Example], page 140, for a nice example.

With the directed search strategies, tree pruning is essential. See Section 12.4.6 [Tree Pruning Options], page 129, for a description of this process and how it can be controlled.

The `QMIN` and `QMAX` options control the size of the work queue when parallel processing is used. The default values for this variables depends upon the type of search. For an exhaustive search, they are set to 10 and 40 times the number of CPU's in use, respectively. For a directed search, they are both set to the number of CPU's. See Section 27.1 [Parallel Command], page 227, for a description of how the number of CPU's can be set. `QMIN` must always be less than or equal to `QMAX`.

12.4.2 Conformational Search Limits

The `MAXLEAF` option controls how much of the search is executed. Each time `CONGEN` samples the last of the degrees of freedom, i.e., reaches the bottom of the search tree, the leaf count (variable `LEAFNUM`) is incremented. If `MAXLEAF` is specified, then the search is terminated when `LEAFNUM` equals or exceeds `MAXLEAF`.

The `MAXNODE` options controls how many search tree nodes are generated during a search. Once the number is exceeded, the search is stopped safely. `MAXNODE` can also be specified for each degree of freedom to limit the number of samples taken for any one partial conformation. This is useful when experimenting with the `ALL` sidechain option, see Section 12.5.3 [Sidechain Degree of Freedom], page 133.

12.4.3 Maps and Construction Tables

The `GLYMAP`, `PROMAP`, and `ALAMAP` options specify the unit numbers for the default torsion angle maps (see Section 12.3.1 [Backbone Maps], page 122), for glycine, proline, and all other amino acids, respectively. Maps for individual amino acids can be selected specifically using the `MAP` and `EMAX` keywords for the backbone degree of freedom, see Section 12.5.1 [Backbone Degree of Freedom], page 131, for more information.

The `GLYEMAX`, `PROEMAX`, and `ALAEMAX` options specify the how much of the torsion angle maps are used for glycine, proline, and all other amino acids, respectively. When each map is read in, the minimum energy is determined. Then all entries which have energy within `EMAX` of this minimum are marked.

The `PROCONS` option specifies the unit number for the proline constructor file, see Section 12.3.2 [Proline Constructors], page 123. The `ERINGPRO` option specifies which constructors are used by energy. When the procons file is read, the minimum of the energy of ring conformations minus the constraint energy used to the hold the ring is calculated. All constructors whose energy difference is within `ERINGPRO` of the minimum difference are used for proline construction. Any constructors outside this range are ignored. If this option is omitted, then all constructors are used.

The `STUNIT` option specifies the unit number for the sidechain topology file. See Section 12.3.3 [Sidechain Topology], page 123, for more information.

12.4.4 Energy Calculation Options

The `HBCG` option specifies the hydrogen bond energy parameters to be used for this search. See Section 5.1 [HBOND Syntax], page 49, for more details. The `NBCG` option specified the non-bonded energy parameters to be used for this search. See Chapter 6 [Non-bonded Interactions], page 51, for more details. Please note that only the `ATOM` and `CONS` electrostatic options are supported. In addition, the `CONS` option used for evaluating sidechain conformations is slightly different than used for other energy evaluations, in that the surrounding field is ignored.

The `EIGNORE EVDW` option directs the program to ignore the van der Waals term in the calculation of all energies.

12.4.5 Poisson-Boltzmann Options

By using the `PBE` option, it is possible to use electrostatic energies calculated using the Poisson-Boltzmann equation (PBE) in evaluating conformations. This code is still experimental, and is far from robust. In order to use it, you must first issue a `PBE SETUP` command, see Section 8.3.1 [PBE SETUP Command], page 71 which constructs the grids using all the atoms that will be used in the conformational search. Also, using this code is very slow because solving the Poisson-Boltzmann equation is expensive. If you can use parallel processing to speed execution, you should use the

`LOOPS` option in the `PARALLEL` command, see Section 27.1 [Parallel Command], page 227, in order to conserve memory.

Several possibilities for using the Poisson-Boltzmann equation are provided. By default, the Poisson-Boltzmann electrostatic energy substitutes for the Coulomb energy. If you specify the `ONLY` keyword, then the PBE energies substitute for the total energy. If you specify the `SOLVATION` keyword, then the electrostatic solvation energy is used. The electrostatic solvation energy is the difference between the Poisson-Boltzmann energy of the system with a solvent dielectric as specified in the `PBE SETUP` command and the Poisson-Boltzmann energy of the system with a solvent dielectric as specified in the `VACUUM` option. By default, the `VACUUM` dielectric is 1.0. The `ONLY` option can be specified with the `SOLVATION` option, and the result is that the electrostatic solvation energy substitutes for the total energy. The `NEWB` option specifies that the boundary be recalculated for each new conformation, whereas the `FIXB` option specifies the boundary be left untouched. `NEWB` is the default. An example of these options in use is given in Poisson-Boltzmann section, see Section 8.4 [PBE Examples], page 83.

12.4.6 Search Tree Pruning Options

The `TREE` global option is used to control the pruning and display of the search tree. When directed searches are performed one large problems, see Section 12.1.3 [Overview of Directed Searching], page 116, the search tree can grow very rapidly, and consume all available virtual memory. Generally, one is not interested in running these searches to completion, so removing open nodes that are not likely to be expanded is an appropriate step to conserve memory. Depending on the search strategy, the method used for pruning the tree will vary.

In the case of the evaluation directed search strategy, the tree pruning strategy is obvious, the nodes with the highest evaluations are deleted. In the deepening evaluation strategy, the tree pruning strategy is much less clear. Depending on the nature of the degree of freedom, the number of open nodes at each level may vary substantially. Only those levels with many nodes should be pruned. In addition, since the top level nodes are visited most often through the search, these levels should not be pruned. The specific strategy for the deepening evaluation strategy is as follows:

1. The average and standard deviation of the number of open nodes per degree of freedom is calculated. Only those degrees of freedom that have more than one open node are used in the calculation.
2. No pruning is done within `TOPSAVE` levels of the root.
3. No pruning is done where the number of nodes on a level is less than or equal to the average number minus the standard deviation times the value of the `SDSAVE` option.
4. All other levels are reduced in number by a factor of the `REDUCTION` option.

In the case of the mixed strategy, the program applies both pruning strategies and deletes only those nodes that satisfy both rules.

The meaning of each option is given below:

`PRTFRQ`	The printing frequency for printing a display of search tree in units of generated nodes. If the frequency is set to 0, then no printing will be done. The default is 0.
`LIMIT`	When the number of nodes in the search tree exceeds `LIMIT`, pruning is done. The default is 10000 for directed searches and 1000 for depth first and breadth first searches.
`REDUCTION`	The reduction factor for pruning. When evaluation directed searching is used, the number of nodes will be reduced by this factor. When deepening evaluation directed searching is used, those levels in the search tree that pass their tests will be reduced by

this factor. When the mixed strategy is used, then only those nodes that would have been pruning by both strategies will be pruned. The default is 3.0.

`TOPSAVE` The number of levels closest to the root node that are not pruned when the deepening evaluation directed search or the mixed strategy are used. The defaults is 1.

`SDSAVE` When the deepening evaluation directed strategy is used, only those levels whose number exceeds the average minus `SDSAVE` times the standard deviation of the number distribution will be considered for pruning. The default value of `SDSAVE` option is 0.5.

12.4.7 Checkpoint and Restart Options

Since conformational searches can execute for long periods of time, it is necessary to be able to save the state of search and restart it at a later point in time. The `CHECKPOINT` and `RESTART` options provide this capability. The frequency of checkpoints are also used to specify the frequency of status writing operations. Finally, a checkpoint is written at the end of the run, so that a long run that is terminated by your options can be continued at greater length later.

The `CHECKPOINT` options are as follows:

`UNIT` The `UNIT` options specifies what file unit the checkpoints should be written to. If this is omitted or given a value of -1, then no checkpoints are written. CONGEN handles the file attached to this unit in a special way. When a checkpoint is about to be written, CONGEN attempts to determine the name of the file associated with the unit. It then issues an operating system command to rename or move the file to a different name made up of the original name concatenated with '`_bak`'. On VMS systems, any previous '`_bak`' file is deleted first. Then, a new file with the original name is opened, and the checkpoint is written. Thus, one good checkpoint file is always maintained, and usually there are two.

`NODEFREQ` The `NODEFREQ` option specifies how many nodes must be allocated between writing a checkpoint. The default is 100000.

`TIMEFREQ` The `TIMEFREQ` option specifies the time in minutes between writing checkpoint files. Checkpoints are written when either `NODEFREQ` or `TIMEFREQ` tests are passed, and both the counters and timers are reset when the checkpoint is written.

`[NO]FLUSH`
This options controls when CONGEN periodically flushes the output buffers for the log file (Fortran unit 6) and the output files specified in the write coordinate degrees of freedom. This option is normally on, but it can be turned off using `NOFLUSH`.

To restart a CONGEN run, one should edit the CONGEN input file that used for the checkpoint file, and a `RESTART` option should be used. The unit number specified in the `RESTART` option should be a valid checkpoint from a previous run. It is permitted to change the options which control the operation of search such as `MAXNODE` and `MAXLEAF`, but the degrees of freedom must not be changed. Only limited testing of option changing has been performed after restarting, so be careful.

12.4.8 Status Display Options

The `STATUS` option specifies whether a status file is written periodically through the conformational search. The status file is written using the same file naming scheme as the checkpoint file described in Section 12.4.7 [Checkpoint and Restart Options], page 130. The frequency of writing is also dictated by the `CHECKPOINT` option. The status file is written to the unit specified by `UNIT` option.

The `SETPRN` option directs the program to set the process name to indicate the progress of the search. This option works only for batch jobs on VMS systems. The process name is set to the number of leaves found thus far followed by the minimum evaluation encountered thus far during the search.

12.4.9 Miscellaneous Global Options

After a conformational search is complete, the coordinates are left in the same state as they were before the search began. The `COOR SAVE END` option is ignored.

The `DEBUG` option controls the setting of the `CGEN` debug variable. The default value of 0 prints no debugging information. Larger values print more information. A debug value of 2 is good for getting an idea of why a search failed without generating too much output. The maximum value of 5 will fill a disk as fast as the computer can write to it. See Section 27.11 [Debug Command], page 230, for more details.

The `NOSOPT` option will turn off an optimization in the sidechain `FIRST` placement method. See Section 12.5.3 [Sidechain Degree of Freedom], page 133, for more details. This option is used only for debugging.

The `SEED` option sets the seed for the random number generator used by `RANDOM` evaluation option, see Section 12.4.1 [Search Method Options], page 126. When CONGEN runs in parallel, each process gets a copy of this seed. The seeds are not saved in the restart file, so a run interrupted and restarted will not generate the same random structures.

The options, `EPCONS` and `EJCONS`, are used to apply dihedral angle constraints, see Section 11.2 [Harmonic Dihedrals], page 96, or NMR J coupling constraints, see Section 11.5 [NMR Constraints], page 97, in the generation of atomic positions. This functionality is experimental, and requires some care in its application. Both options specify the maximum allowed energy for constructing an atom which is either the first or last atom in the specification of a dihedral angle or J coupling constraint, and whose antecedent atoms are also in the specification of the constraint. No dihedral angle constraints are applied if the J coupling constraint derives from an average over multiple conformations or if the J's are joined. For example, if you have a J coupling constraint for a backbone phi torsion angle, then the construction of the backbone carbonyl carbon at the end of that constraint may be affected. The option, `EPCONS`, is used to select the dihedral angle constraints. The option, `EJCONS`, is used for the J coupling constraints. Both of these options specify energies in kcal/mole. If these values are very large, then they will not have any affect on the conformational search. At the present time, these constraints are not coupled with van der Waals avoidance when sidechains are constructed, see Section 12.1.2 [Sidechain Overview], page 114. If there is sufficient interest, they can be.

12.5 Degrees of Freedom

There are six "degrees of freedom" currently available in CONGEN. Three; backbone, chain closure, and side chain; are involved with atom construction; two; `RBEST` and `WRITE`; are involved with I/O; and the last, `EVL`, is involved with evaluation of conformations.

12.5.1 Backbone Degree of Freedom

The backbone degree of freedom constructs the backbone atoms of a polypeptide. It can handle all the natural amino acids as well amino-isobutyric acid (AIB). For our purposes, the backbone atoms are N, H, CA, CB, C, and O and the ring atoms in proline.

This degree of freedom is implemented like a macro in that the program treats the backbone of each residue as a separate degree of freedom. Thus, when this command is used to generate conformations for n backbones, n backbone degrees of freedom are generated internally.

The `STARTRES` option specifies the starting residue for which a backbone is to be constructed. The `LASTRES` option specifies the final residue for construction. Together with the `FORWARD` and

REVERSE options, these options also specify which way the construction should be done, i.e. constructing from N to C or C to N. The following decision table gives the order used for linear structures:

	FORWARD	REVERSE	NONE
STARTRES<LASTRES	N->C	C->N	N->C
STARTRES=LASTRES	N->C	C->N	N->C
STARTRES>LASTRES	N->C	C->N	C->N

The following decision table gives the order used for cyclic structures:

	FORWARD	REVERSE	NONE
STARTRES<LASTRES	N->C	C->N	Use shortest distance.
STARTRES=LASTRES	N->C	C->N	Forward = true
STARTRES>LASTRES	N->C	C->N	Use shortest distance.

The range of residues cannot span between two segments of the PSF.

The MAP option specifies the Fortran unit number for the backbone energy map, see Section 12.3.1 [Backbone Maps], page 122, to be used. If no map is specified, then the default map unit is used. The defaults are specified by the PROMAP, GLYMAP, and ALAMAP global options, see Section 12.1.1 [Backbone and Chain Closure Overview], page 113. The EMAX option specifies the maximum energy of entries in the map for the range of residues. If no EMAX option, then the global EMAX option appropriate for the given residue applies. You can use this option to override the default EMAX option for a particular backbone specification.

The MAXEVDW option specifies the maximum energy for any repulsive contact between any generated atom and atoms in the surroundings.

The CISTRANS and ALLCISTRANS options control whether cis peptides are included in the search. By default only trans peptides are used. CISTRANS specifies that only prolines may have cis peptides. ALLCISTRANS specifies that any amino acid may have cis peptides. In this context, CISTRANS and ALLCISTRANS applied to residue *n* refer to the peptide bond between residue *n-1* and *n*. N.B. If no cis peptides are included in the torsion angle map, then this option has no effect. Generally, all the peptide maps defined over a grid have both cis and trans peptides, but you should check if this option is important to your problem.

The CLSA, CLSD, and MAXDT options control an important optimization in the search. Since this degree of freedom is generally used in conjunction with a chain closure, there is no point constructing backbones that stray too far for closure to take place. The CLSA option specifies the atom which will terminate the chain closure. For a construction in the N->C direction, this atom should be a CA; for the opposite direction, this atom should be an N. The program will construct a model backbone, all torsion angles being trans, and all bond angles involved in chain closures being stretched by MAXDT degrees (see below for more details) which would connect from the residue whose backbone is being searched to that CLSA atom, and it will measure this distance. If a backbone conformation is constructed with a separation distance greater than the CLSA distance, that search path will be abandoned. CLSD permits the closing distance to be modified. If just a number is specified with CLSD, then the program uses this value in place of the model backbone calculation. If the keyword, DELTA, is also used, then the real number is added to the model distance.

The construction of the model peptide attempts to correctly account for the flexibility permitted in peptide bond angles. If MAXDT is specified, then all bond angles in the model calculation are stretched by this amount. If MAXDT is not specified in the backbone command, then CONGEN searches through all the degrees of freedom to see which involve chain closure. If chain closures are found, then the associated bond angles are stretched accordingly. If the search involves no chain closures, then the default value for MAXDT will be used to stretch all bond angles.

The backbone degree of freedom also handles backbones at the N and C termini. The N terminus and C terminus cannot use the standard torsion angle maps because there are fewer atoms than assumed in the construction of the maps and because of the rotational symmetry present at each end. Thus, the `CONGEN` command will perform a complete sampling for each end using only the `MAXEVDW` test. The grid for this sampling will be set to value of the `GRID` option. Also, the rotational symmetry of the terminus will be used to reduce the search unless `NOTERSYMMETRY` is specified.

N.B., there is currently an error in the design of the code with regard to the construction of prolines at the amino terminus when using the AMBER94 potential. The problem manifests itself in geometric errors in the amino terminal nitrogen. Until this problem is fixed, you should avoid such constructions.

The `FIX` option is used to specify that the backbone should be constructed in a single fixed conformation without searching. The bond lengths, angles, and torsions for all the atoms in the degree of freedom are calculated from the current coordinates. The normal test of van der Waals overlap applies.

12.5.2 Chain Closure Degree of Freedom

The chain closure degree of freedom calls the modified Go and Scheraga chain closure procedure. This procedure find atomic positions for the backbone atoms starting with the residue given by the `STARTRES` option and continuing to residue `STARTRES+2`. In residue `STARTRES`, the position of the N must be known; in residue `STARTRES+2`, the position of CA, C, and O must be known. In addition, `STARTRES` cannot be the first residue in a protein chain, although `STARTRES+2` can be the C terminal residue. This degree of freedom can handle all the natural amino acids as well as amino-isobutyric acid (AIB).

The `MAXDT` option specifies (in degrees) how much variation in peptide bond angles is permitted for a chain closure. The default value of 5 degrees should be adequate for most applications.

The `MAXG` option controls how bond angle adjustments are made. The default value gives good results. See the source code file, ‘`clschn.flx`’, for more details.

`MAXEVDW` specifies the largest energy permitted for a repulsive contact for any atom generated in the chain closure. Its value is specified in degrees.

`CISTRANS` controls whether the chain closure routine will generate cis prolines for any position within the three residues being constructed. `ALLCISTRANS` controls whether cis prolines can be generated for any residue, proline or otherwise.

12.5.3 Sidechain Degree of Freedom

The sidechain degree of freedom is the most complex. There are presently six different ways of constructing sidechains. See Section 12.1.2 [Sidechain Overview], page 114, for more information about the sidechain degree of freedom.

Unlike the other degrees of freedom, many of the options are associated with particular residues. I.e., they are position dependent. These position dependent options are `MAXEVDW`, `SGRID`, `VAVOID`, `CLUMP`, and `NOSYMMETRY`. The program will first scan the command string for the first of these options, and it will use that value as the default for all the residues. Then, starting with this default, it will scan for options and residues (`STARTRES` and `LASTRES` options below). Each time it finds an option specification, the current value is changed. Each time it finds a residue, the option settings for that residue will be set to the current value.

The residues whose sidechains are to be constructed are specified with an arbitrary number of `STARTRES` and `LASTRES` commands. Each `LASTRES` command is paired with the previous `STARTRES` command, and all residues within the pair are constructed. However, a `STARTRES` command need not have a LASTRES pair, and in that case, only the residue specified in the STARTRES command will be processed. E.g., STARTRES H 30 STARTRES H 10 LASTRES H 12 STARTRES H 25 will search the sidechain conformational space for residues; H 10, H 11, H 12, H 25, and H 30. If not STARTRES command is given, then the residue specified in the last BACKBONE degree of freedom will be used.

The MAXEVDW option specifies the maximum repulsive contact for any atom within the sidechain. This option is position dependent as described above.

VAVOID and NOVAVOID control whether van der Waals repulsion avoidance is used. VAVOID signifies that avoidance is to be done. Van der Waals repulsion avoidance results in the program using only those torsion angles which avoid any van der Waals contact greater than MAXEVDW. Every atom that can make contact with the atoms in a clump are checked. Any sampling of the sidechain grid that falls into a repulsive range is moved to the closest torsion angle within an acceptable range.

SGRID controls the sampling grid. If a numeric value is specified, the value is taken for the grid in degrees. The first value for the torsion angle is taken from the global minimum of the torsion angle potential for the free atom and its antecedents. If there are multiple minima, then the smallest value in the range $[-\pi, \pi)$ is taken. If MIN is specified, then the grid is set to the local minima in the energy of the free torsion angle of the clump. If SELECT is specified, then you can specify the torsion angle grid as a function of the number of free torsions in the sidechain. Each number or MIN keyword specifies the sidechain grid to use starting with sidechains with one clump on up. The last value applies to all the larger sidechains. The keyword, AUTO, is equivalent to “SELECT 10 30 30 60 60 MIN END”.

The CLUMP option allows the user to specify a subset of clumps for succeeding residues. This is useful when one wishes to search over only a portion of a sidechain or a portion of an arbitrary molecule. The syntax of this option is intended to provide a simple means for specifying a range of clumps. If the option, ALL, is used, then succeeding residues will include all clumps. If a single word is given, then only clumps named by that word will be included. If the colon form of the option is used, then only clumps which are sequentially between the two names in the sidechain topology file entry for the residues will be used. If a word is omitted on either side of the colon, then either the first or last clump of each residue will be used as the default, respectively. A solitary colon is thus the same as the ALL option. If you want to use disjoint subsets of clumps within one residue, simply specify two sets of CLUMP and STARTRES commands.

For example, suppose one wished to search over the gamma carbons and gamma 2 hydrogens of a valine alone. These are currently listed as clumps 1 and 3 in the valine residue in the sidechain topology file. Assume that the valine is in segment MAIN and has residue identifier 46. Then the following command segment would be appropriate:

```
SIDE CLUMP 1 STARTRES MAIN 46 -
     CLUMP 3 STARTRES MAIN 46 ...
```

SYMMETRY and NOSYMMETRY controls whether clump symmetry is used. Normally, the rotational symmetry associated with a clump (see Section 12.3.3 [Sidechain Topology], page 123), reduces the search space. However, the elimination of this symmetry is occasionally desirable because the comparison command of the analysis facility (see Chapter 18 [Comparisons], page 173) does not recognize symmetric elements in residues and can generate artifactually large differences. The effect of NOSYMMETRY is make all clumps have symmetry of 1.

SIDEOPT specifies the sidechain construction method. See Section 12.5.3 [Sidechain Degree of Freedom], page 133, for more details. The default is FIRST.

There is an important optimization which is performed when the FIRST sidechain construction method is used. (It is also important for the ITERATIVE method too, because this method uses a conformation generated by FIRST to get started.) If the FIRST method fails to find a conformation for a particular residue, it will backtrack in the set of residues being constructed to the first residue which made contact with the failed residue. If no such residue exists, then the search fails. In order to make this work in the optimal way, you should group residues whose sidechain interact together when specifying their order. In addition, the global option, NOSOPT, will turn off this optimization.

NCOMB specifies the number of combinations to be used for the COMBINATION method. The default is 2.

EVAL specifies how the sidechain conformations are to be evaluated when the ITERATIVE, INDEPENDENT, and COMBINATION options are used. For the RMS and WRMS options to work, the coordinates stored in the program prior to the execution of the CONGEN command must be the reference set. The WRMS option means *W*orst *RMS*.

MAXSIDEITER controls how many iterations over all the residue are performed during an iterative sidechain construction. The default is 10.

12.5.4 Read Best Conformations Degree of Freedom

The Read Best Conformations degree of freedom reads a conformation file ('CG' file) produced by another run of the CONGEN command, and selects the best conformations from it. Each conformation selected is considered one sampling for this degree of freedom. It is potentially useful when a complete search would take too long to complete. In such a case, a search over part of the atoms would be written out, and then a low energy subset would be read back for use.

The UNIT option specifies the Fortran file unit where the input conformation file is read. NBEST specifies how many conformations are read back in the following way: The file is first scanned and the energy of each conformation is recorded in an array. The array is sorted and then, the NBESTth lowest energy is selected. During the actual sampling process, the conformation file is scanned again, and any conformation having an energy less than or equal to the selected energy is used. Thus, if several conformations have the same energy, it is possible for more than NBEST conformations to be read in.

MAXEVDW specifies the maximum van der Waals repulsion for any atom read by this degree of freedom. If the parameter file specifies that hydrogen bonds should exclude the non-bonded energy, see Section 3.1.4.1 [Parameter File Format], page 21, then the hydrogen bond potential will be used to calculate the distance corresponding to MAXEVDW for interactions involving hydrogen bondable atoms.

12.5.5 Write Conformations Degree of Freedom

The Write Conformations Degree of Freedom writes the coordinates of all atoms constructed in the degrees of freedom which precede this command. See Section 12.3.4 [Conformation File], page 126, for a description of the output file format. Note that the last energy evaluation done by either the EVL degree of freedom (see Section 12.5.6 [Evl Degree of Freedom], page 136), or the read best degree of freedom (see Section 12.5.4 [Read Best Degree of Freedom], page 135), is written with the conformation.

The CUNIT operand specifies the Fortran unit number where the file is to be written. Note that this file must be opened UNFORMATTED.

The MAXCONF option can be used to truncate the search. Once MAXCONF conformations have been written to the file, the search is stopped.

The REF COMP option directs the program to write the coordinates in the comparison coordinate set. See Chapter 15 [Coordinate Manipulations], page 151, for more information.

The three cutoff options; `CUT`, `MINCUT` `MAXCUT`; are used to limit the number of conformations written to the file. If neither option is specified, then all conformations are written to the output file. If `CUT` is set, then only those conformations whose last evaluation was less than or equal to `CUT` will be written. If `MINCUT` is set, then only those conformations whose last evaluation is within `MINCUT` of the lowest evaluation seen so far will be written. If `MAXCUT` is set, then only those conformations whose last evaluation is within `MAXCUT` of the highest evaluation seen so far will be written. These last two tests are an incomplete, but hopefully useful way of limiting the output of CONGEN to conformations within the cutoffs of the extreme energy conformation. If multiple options are specified, then the conformation is written if any test is satisfied.

The `FILTER` option is used in conjunction with cutoff options to stop conformers for being passed down to succeeding degrees of freedom. When the `FILTER` option is absent, then all conformers pass to succeeding levels. When the `FILTER` option is present, then only those conformers which are printed are passed to the next degree of freedom.

12.5.6 Evaluate Degree of Freedom

The evaluate degree of freedom evaluates the conformation generated up to this point in the search. **N.B.**, the evaluate degree of freedom is specified using the `EVL` keyword. The letter, `A` is intentionally omitted. In addition, it has the capability of running any energy commands from the main program. These degrees of freedom keep running totals of the minimum and maximum values found, and they are printed at the end of normal execution. In addition, the evaluations are saved for writing to the conformation file if a `WRITE` degree of freedom follows the evaluation. If no options for this degree of freedom are specified, then the energy is used.

If the `CAVITY` option for the GEPOL molecular surface code is set, see Section 27.15 [Gepol Command], page 239, and if energies are being evaluated, then a cavity formation energy will be added to the energy evaluation. This cavity formation energy calculation is currently quite expensive.

The `MINI` evaluation option call the energy minimization and dynamics command (see Chapter 7 [Energy Manipulations], page 55). Prior to call this option, CONGEN saves the coordinates of the generated coordinates and will reset them when the degree of freedom passes control to the previous degree of freedom. Any modifications to the coordinates are passed to the next degree of freedom. In addition, the coordinates of the surrounding atoms are fixed, (see Section 11.4 [Fixed Atoms], page 97).

The `RMS` evaluation option computes the RMS difference between the current coordinates and the coordinates when the CONGEN command was invoked.

The `RANDOM` evaluation option assigns a random number as the evaluation of this conformer. In conjunction with the `CUT`, `MINCUT`, `MAXCUT`, and `FILTER` options below, it can be used to select a number of conformers at random for succeeding constructions.

The `USER` evaluation option calls the user defined evaluation. This option is not well developed, see the source code files cgen.c and usersb.flx for more information.

The `NPRINT` option is used to limit evaluation listings to every `NPRINTth` evaluation. If `NPRINT` is set to zero, then all evaluations are listed. At present, `NPRINT` does not turn off the output of `ECNTRL` when a minimization is done on the conformations.

The three cutoff options; `CUT`, `MINCUT` `MAXCUT`; are used to limit the number of conformations written to the file. See Section 12.5.5 [Write Degree of Freedom], page 135 for a description of how these options work to limit energy outputs into the log file.

The `FILTER` option is used in conjunction with cutoff options to stop conformers for being passed down to succeeding degrees of freedom. When the `FILTER` option is absent, then all conformers pass to succeeding levels. When the FILTER option is present, then only those conformers which are printed are passed to the next degree of freedom.

NPRINT and the cutoff interact as follows: The NPRINT check is made first. If the conformation's energy is to be printed, the cutoff check is then made.

12.6 Examples of CONGEN Commands

There are many examples of CONGEN commands in the directory, 'CGT:'. Most of the test cases begin with the letters, CG. Here, we give two examples of using CONGEN; a search over five residues, and a reconstruction of all sidechains in a protein.

12.6.1 Five Residue Search

In this example, we search over the conformational space of residues 127 to 131 in flavodoxin. The search is done using backbone degrees of freedom over residue 127 and 128, with a chain closure over residues 129 to 131. We do searches to find both the minimum energy conformation and minimum RMS deviation to the X-ray crystal structure.

```
Construction of helix segment 127-131 in flavodoxin.
In this run, we find both the theoretical lower limit and the
lowest energy conformations.
*
OPEN NAME CGDATA:RTOPH8.MOD UNIT 01 READ UNFORM
READ      RTF UNIT 1
OPEN NAME CGDATA:PARAM5.MOD UNIT 03 READ UNFORM
READ      PARAMETER UNIT    3
OPEN UNIT 10 NAME CGTD:FLVDOXPSF.MOD UNFORM READ
READ PSF FILE UNIT 10
OPEN UNIT 11 NAME CGTD:FLVDOX.MOD UNFORM READ
READ COOR FILE UNIT 11
COOR COPY COMP
OPEN UNIT 60 NAME 127FLVRMS.CG UNFORM WRITE            ! Conformation file
OPEN UNIT 70 NAME CGDATA:TOPCGEN2.INP FORM READ        ! Sidechain topology
OPEN UNIT 51 NAME CGDATA:EMAPGLY30.OMP FORM READ       ! Glycine torsion map
OPEN UNIT 52 NAME CGDATA:EMAPALA30.OMP FORM READ       ! Alanine torsion map
OPEN UNIT 53 NAME CGDATA:EMAPPRO30.OMP FORM READ       ! Proline torsion map
OPEN UNIT 55 NAME CGDATA:PRO.CNS FORM READ             ! Proline constructor
OPEN UNIT 40 NAME 127FLV.STS FORM WRITE                ! Status file
!
!   Run the search now.
!
CGEN -
STATUS SETPRN UNIT 40 END -
HBCG CUTHB 4.5 CUTHBA 90.0 -
     CTONHA 98.0 CTOFHA 99.0 CTONHB 98.0 CTOFHB 99.0 END -
- !   Evaluate energy using constant dielectric of 50, distance cutoff = 5 A
NBCG CUTNB 5.0 ATOM CTONNB 98.0 CTOFNB 99.0 END -
- !
- !   The following Backbone degree of freedom contains CLSA optimizations to
- !   the correct terminators for the search.
- !
BACK MAXEVDW 20 STARTRES 1 127 LASTRES 1 128 CISTRANS CLSA 1 131 CA $ -
CHAIN STARTRES 1 129 CISTRANS MAXEVDW 20 $ -
```

```
SIDE SGRID MIN STARTRES 1 127 LASTRES 1 131 -
     SIDEOPT INDE MAXEVDW 20 EVAL RMS $ -
- !
- !   We compare RMS's to the starting coordinates
- !
EVL RMS $ -
WRITE CUNIT 60 $ -
GLYMAP 51 ALAMAP 52 PROMAP 53 PROCONS 55 -
GLYEMAX 2 ALAEMAX 2 PROEMAX 2 STUNIT 70 -
ERINGPRO 50 -
  ! The following contains the title used in the conformation file
127FLVRMS.CG
Conformations of helix segment 127-131 in flavodoxin.
RMS driven.
*
COOR COPY              ! Restore original coordinates.
OPEN UNIT 60 NAME 127FLVE.CG UNFORM WRITE              ! Conformation file
OPEN UNIT 70 NAME CGDATA:TOPCGEN2.INP FORM READ        ! Sidechain topology
OPEN UNIT 51 NAME CGDATA:EMAPGLY30.OMP FORM READ       ! Glycine torsion map
OPEN UNIT 52 NAME CGDATA:EMAPALA30.OMP FORM READ       ! Alanine torsion map
OPEN UNIT 53 NAME CGDATA:EMAPPRO30.OMP FORM READ       ! Proline torsion map
OPEN UNIT 55 NAME CGDATA:PRO.CNS FORM READ             ! Proline constructor
OPEN UNIT 40 NAME 127FLV.STS FORM WRITE                ! Status file
!
!   Run the search now.
!
CGEN -
STATUS SETPRN UNIT 40 END -
HBCG CUTHB 4.5 CUTHBA 90.0 -
     CTONHA 98.0 CTOFHA 99.0 CTONHB 98.0 CTOFHB 99.0 END -
- !   Evaluate energy using constant dielectric of 50, distance cutoff = 5 A
NBCG CUTNB 5.0 ATOM CTONNB 98.0 CTOFNB 99.0 END -
- !
- !   The following Backbone degree of freedom contains CLSA optimizations to
- !   the correct terminators for the search.
- !
BACK MAXEVDW 20 STARTRES 1 127 LASTRES 1 128 CISTRANS CLSA 1 131 CA $ -
CHAIN STARTRES 1 129 CISTRANS MAXEVDW 20 $ -
SIDE SGRID MIN STARTRES 1 127 LASTRES 1 131 -
     SIDEOPT ITER MAXEVDW 20 EVAL E $ -
- !
- !   We do a simple energy evaluation for the five residues. The
- !   energies will be written to the conformation file.
- !
EVL MINI ENERGY END $ -
WRITE CUNIT 60 $ -
GLYMAP 51 ALAMAP 52 PROMAP 53 PROCONS 55 -
GLYEMAX 2 ALAEMAX 2 PROEMAX 2 STUNIT 70 -
ERINGPRO 50 -
  ! The following contains the title used in the conformation file
127FLVE.CG
Conformations of helix segment 127-131 in flavodoxin.
```

```
Energy calculations.
*
```

12.6.2 Complete Sidechain Reconstruction Example

This example illustrates how every sidechain in a structure can be reconstructed. This run serves as an important test of CONGEN. Note that cysteines would not rebridged correctly because the `CONGEN` command doesn't handle the disulphides at present.

```
Reconstructing side chains on flavodoxin backbones
*
OPEN NAME CGDATA:RTOPH8.MOD UNIT 01 READ UNFORM
READ RTF UNIT 1
OPEN NAME CGDATA:PARAM5.MOD UNIT 03 READ UNFORM
READ PARAMETER UNIT 3
OPEN UNIT 10 NAME CGTD:FLVDOXPSF.MOD UNFORM READ
OPEN UNIT 11 NAME CGTD:FLVDOX.MOD UNFORM READ
READ PSF FILE UNIT 10
READ COOR FILE UNIT 11
COOR COPY COMP
ENERGY
OPEN UNIT 70 NAME CGDATA:TOPCGEN3.INP FORM READ
!
!   The map and proline constructor files are required for all CONGEN
!   runs even if they are not used.
!
OPEN UNIT 51 NAME CGDATA:EMAPGLY30.OMP FORM READ
OPEN UNIT 52 NAME CGDATA:EMAPALA30.OMP FORM READ
OPEN UNIT 53 NAME CGDATA:EMAPPRO30.OMP FORM READ
OPEN UNIT 55 NAME CGDATA:PRO.CNS FORM READ
CONGEN -
SIDE VAVOID MAXEVDW 20 -
     SIDEOPT ITER EVAL E MAXSIDE 30 -
     SGRID MIN -
     STARTRES 1 1 LASTRES 1 138 $ -
EVL MINI ENERGY END $ -
GLYMAP 51 ALAMAP 52 PROMAP 53 PROCONS 55 -
GLYEMAX 2 ALAEMAX 2 PROEMAX 2 STUNIT 70 -
ERINGPRO 50 -
HBCG CUTHB 4.5 CUTHBA 90 CTONHB 98 CTOFHB 99 CTONHA 98 CTOFHA 99 END -
NBCG CUTNB 8.0 ATOM CTONNB 98.0 CTOFNB 99.0 END
Conformations of sidechains in flavodoxin model
*
!
!    Compare sidechain coordinates
!
COOR RMS CLEAR ATOM * * NT ATOM * * N ATOM * * CA ATOM * * C ATOM * * CB -
               ATOM * * O ATOM * * OT1 ATOM * * OT2 ATOM * * H -
               ATOM * NTER * ATOM * 1 HT3 NOT
OPEN UNIT 21 NAME FLVDOXS.MOD WRITE UNFORM
WRITE COOR FILE UNIT 21
FLVDOXS.MOD
Sequentially constructed flavodoxin. Side chains generated by CONGEN.
```

```
SGRID 30 SGRID 60 for ARG and LYS.
*
ANAL
SET LINESZ 80
COMPARE COOR COMP $
BUILD DIFF ATOM R
DELETE VALUE ABS LT 0.0005 $             ! Delete backbone comparisons
ADD STATS RMS $ PLACE RESIDUE SEGMENT $    ! Add residue statistics
PRINT TABLE PRETTY
END
```

12.6.3 Energy Directed Search Example

This example illustrates how a energy directed search can be executed where the search is driven by the energies of complete residues along the loop. Note that we reconstruct sidechains repeatedly in order to get an accurate energy.

```
Energy directed conformational search
*
!
! Open and read RTF, parameters, PSF, and coordinates.
!
OPEN NAME CGDATA:RTOPH8.MOD UNIT 01 READ UNFORM
READ RTF UNIT 1
OPEN NAME CGDATA:PARAM5.MOD UNIT 03 READ UNFORM
READ PARAMETER UNIT 3
OPEN UNIT 10 NAME CGTD:FLVDOXPSF.MOD UNFORM READ
OPEN UNIT 11 NAME CGTD:FLVDOX.MOD UNFORM READ
READ PSF FILE UNIT 10
READ COOR FILE UNIT 11
ENERGY
!
! Open files needed for the search
!
OPEN UNIT 60 NAME EXAMPLE.CG UNFORM WRITE
OPEN UNIT 70 NAME CGDATA:TOPCGEN3.INP FORM READ
OPEN UNIT 51 NAME CGDATA:EMAPGLY30.OMP FORM READ
OPEN UNIT 52 NAME CGDATA:EMAPALA30.OMP FORM READ
OPEN UNIT 53 NAME CGDATA:EMAPPRO30.OMP FORM READ
OPEN UNIT 55 NAME CGDATA:PRO.CNS FORM READ
OPEN UNIT 40 NAME EXAMPLE.STS FORM WRITE
PARALLEL                          ! Setup a parallel search
CONGEN -
SEARCH EVAL ENERGY END -
- ! Allocate a big tree so many branches can be explored.
TREE LIMIT 1000000 TOPSAVE 4 REDUCTION 2.0 SDSAVE 0.0 PRTFRQ 1000000 END -
- ! Write a status file so we can get see the progress.
STATUS UNIT 40 END -
- ! The CHECKPOINT option is used to set the frequency of status file updates.
CHECKPOINT NODE 1000000 TIME 120 UNIT -1 END -
- !
- ! Build the loop from each end toward the middle
- ! Force the program to expand all backbone and chain closure open nodes
- ! so that only complete residues will be used to direct the search process.
-
BACK EIMMED STARTRES 1 38 MAXEVDW 20.0 CISTRANS CLSA 1 48 CA $ -
SIDE VAVOID MAXEVDW 5.0 SIDEOPT ITER EVAL E MAXSIDE 30 SGRID MIN -
     STARTRES 1 38 $ -
BACK EIMMED STARTRES 1 48 MAXEVDW 20.0 CISTRANS CLSA 1 39 N REVERSE $ -
SIDE VAVOID MAXEVDW 5.0 SIDEOPT ITER EVAL E MAXSIDE 30 SGRID MIN -
     STARTRES 1 38 -
     STARTRES 1 48 $ -
BACK EIMMED STARTRES 1 39 MAXEVDW 20.0 CISTRANS CLSA 1 47 CA $ -
SIDE VAVOID MAXEVDW 5.0 SIDEOPT ITER EVAL E MAXSIDE 30 SGRID MIN -
     STARTRES 1 38 LASTRES 1 39 -
     STARTRES 1 48 $ -
BACK EIMMED STARTRES 1 47 MAXEVDW 20.0 CISTRANS CLSA 1 40 N REVERSE $ -
SIDE VAVOID MAXEVDW 5.0 SIDEOPT ITER EVAL E MAXSIDE 30 SGRID MIN -
     STARTRES 1 38 LASTRES 1 39 -
     STARTRES 1 47 LASTRES 1 48 $ -
BACK EIMMED STARTRES 1 40 MAXEVDW 20.0 CISTRANS CLSA 1 46 CA $ -
SIDE VAVOID MAXEVDW 5.0 SIDEOPT ITER EVAL E MAXSIDE 30 SGRID MIN -
     STARTRES 1 38 LASTRES 1 40 -
     STARTRES 1 47 LASTRES 1 48 $ -
BACK EIMMED STARTRES 1 46 MAXEVDW 20.0 CISTRANS CLSA 1 41 N REVERSE $ -
SIDE VAVOID MAXEVDW 5.0 SIDEOPT ITER EVAL E MAXSIDE 30 SGRID MIN -
     STARTRES 1 38 LASTRES 1 40 -
     STARTRES 1 46 LASTRES 1 48 $ -
BACK EIMMED STARTRES 1 41 MAXEVDW 20.0 CISTRANS CLSA 1 45 CA $ -
SIDE VAVOID MAXEVDW 5.0 SIDEOPT ITER EVAL E MAXSIDE 30 SGRID MIN -
     STARTRES 1 38 LASTRES 1 41 -
     STARTRES 1 46 LASTRES 1 48 $ -
BACK EIMMED STARTRES 1 45 MAXEVDW 20.0 CISTRANS CLSA 1 42 N REVERSE $ -
SIDE VAVOID MAXEVDW 5.0 SIDEOPT ITER EVAL E MAXSIDE 30 SGRID MIN -
     STARTRES 1 38 LASTRES 1 41 -
     STARTRES 1 45 LASTRES 1 48 $ -
CHAIN EIMMED STARTRES 1 42 CISTRANS MAXEVDW 20.0 $ -
SIDE VAVOID MAXEVDW 5.0 SIDEOPT ITER EVAL E MAXSIDE 30 SGRID MIN -
     STARTRES 1 38 LASTRES 1 48 $ -
EVL MINI ENERGY END $ -
WRITE CUNIT 60 MINCUT 3 $ -
GLYMAP 51 ALAMAP 52 PROMAP 53 PROCONS 55 -
GLYEMAX 2 ALAEMAX 2 PROEMAX 2 STUNIT 70 -
ERINGPRO 50 -
HBCG CUTHB 4.5 CUTHBA 90 CTONHB 98 CTOFHB 99 CTONHA 98 CTOFHA 99 END -
NBCG CUTNB 8.0 ATOM CTONNB 98.0 CTOFNB 99.0 END
Directed search example.
*
!
! Now extract the best conformation
!
CLOSE UNIT 60
OPEN UNIT 60 NAME EXAMPLE.CG UNFORM READ
XCONF 60 BEST 1
```

12.7 Pointers to Relevant Programs

There are a number of programs available for assisting in the effort to analyze a conformational search. See Section 29.1.1 [Cmploop], page 249, for a description of a program which determines the RMS deviations for all conformations in a conformation file from the reference coordinates stored in that file. In addition the energies are printed. The program `SORTN` can be used to sort the output of `CMPLOOP`. See Section 29.2.1 [Sortn], page 256, for more information.

13 Commands to Assist in Conformational Search

There are several commands available in the CONGEN program to assist in a conformational search. The `SPLICE` command can be used to change the sequence of a structure to greatly simplify the modeling of mutations. The `XCONF` command is used to read a single conformation from a conformation file. The `MERGE CG` command is used to manipulate conformation files; either to merge them, and also to divide and recombine them.

13.1 `SPLICE` — Change the Sequence of the PSF.

13.1.1 Syntax

```
SPLIce [NBXMod int] { segid resid resid repeat(res resid)      }
                    { RESEquence segid resid resid repeat(res) }
```

13.1.2 Function

The `SPLICE` command replaces one set of residues with another set of residues. The coordinate set is shuffled to preserve backbone positions to the extent possible. Insertions of residues can be performed by splicing multiple residues in place of one residue. The `COMPARE` command, see Chapter 18 [Comparisons], page 173, in the analysis facility, see Chapter 16 [Analysis], page 157, can be used to generate the necessary `SPLICE` commands for a homologous transformation of one sequence into another.

When the `SPLICE` command is used, the residue topology file used to construct the PSF must be present when this command is used because this command regenerates the PSF. A corollary of this is that the `SPLICE` command can only be used when the entire PSF was generated from a single RTF. Also, any editing on the PSF, (see Section 4.5 [Edit Command], page 47), will be lost. More details on the implementation are given below.

The residues to be replaced are specified by a segment identifier followed by two residue identifiers which start and end the changed segment. If only one residue is to be changed, then both residue identifiers will be the same.

The specification of replacement residues depends on the presence of the `RESEQUENCE` option. If `RESEQUENCE` is not specified, then you must specify both the residue names and the residue identifiers for all replaced residues. The advantage of this approach is that residue identifiers outside of the changes are consistent. If `RESEQUENCE` is specified, then you must specify only the new residue names. All residues within the segment will be renumbered from the first residue.

The shuffling of coordinates is an important feature of this command. During the resequencing operation, all backbone positions (atoms not constructed by the sidechain degree of freedom operator) are conserved for residues that are in the same position relative to the N-terminal side of the splice. Thus, for an equal length change, the backbone will be preserved. For insertions or deletions, then the C-terminal side of the splice will be wrong; although the anchor for a CONGEN search (atoms CA, C, and O) will be preserved. If a splice results in no change in the residue name, then all the sidechain atoms are copied. In addition, when a glycine is changed to any other residue, the position of CB is constructed assuming perfect geometry for a amino-acid as specified in the parameter set.

The actual implementation of the command is to use the equivalent of the `GENERATE` command (see Section 4.1 [Generate Command], page 43) after modification of the sequence. The `NBXMOD` option controls the automatic generation of non-bonded exclusions when the `GENERATE` command is simulated. See Section 4.2 [Nbxmod], page 45, for more details. The first operation performed after the `SPLICE` command has been successfully parsed is to shuffle coordinates, and to record the position of all disulphides. Then, the PSF is erased and completely regenerated using the new sequence. Disulphides are now added back, and the coordinates for the backbone and endpoints are corrected. The internal coordinates are then cleared, so that a new set of construction rules can be generated, see Chapter 14 [Internal Coordinates], page 147. Finally, the hydrogen bond list, non-bonded list, fixed constraints, harmonic atom constraints, and dihedral angle constraints are all cleared. Any automatic generation options, see Section 4.1 [Generate Command], page 43, are also applied.

13.2 `XCONF` — Extract Conformation

13.2.1 Syntax

```
XCONf {[UNIT] unit} { [ BEST ]     } [ESURF real]
                    { [ WORSt] int }
                    { [NUMBer]     }
```

13.2.2 Function

The `XCONF` command reads a conformation from a conformation file, see Section 12.3.4 [Conformation File], page 126. The conformation can be specified by number, by evaluation ranking, or by accessible surface ranking. Accessible surface ranking was the method used in the paper, R. E. Bruccoleri, E. Haber, J. Novotny, “Structure of Antibody Hypervariable Loops Reproduced by a Conformational Search Algorithm”, *Nature* **335**, 564-568 (1988); *Nature* **336**, 1266 (1988).

The three possible ways of selecting conformations from the file are controlled by the `BEST`, `WORST`, `NUMBER`, `ESURF` options. If `NUMBER` is specified, or if neither `BEST`. `WORST`, nor `NUMBER` is specified, then the integer is used as the sequential conformation number in the file. This number may be specified as 0, which means that the reference coordinate set in the conformation file should be used (see Section 12.3.4 [Conformation File], page 126, for a description). If `BEST` or `WORST` is specified and `ESURF` is not specified, then the integer which follows `BEST` or `WORST` gives the rank ordering in evaluation terms. `BEST` directs the program to use the ordering with the lowest energy first, and `WORST` directs the program to use the highest energy first. If `BEST` or `WORST` is specified and `ESURF` is specified, then the accessible surfaces of the conformations within `ESURF` of the minimum will be calculated (regardless of the settings of `BEST` or `WORST`, and the selection will be based on the rank order of the accessible surfaces of this subset. This latter selection *is* controlled by the `BEST` or `WORST` keywords. The surfaces are calculated in the context of the entire system.

The conformation file is specified by UNIT operand. unit number. Only those atoms which were constructed in the search are read.

N.B. Make sure that the coordinates of the atoms outside of the constructed atoms are exactly the same as they were when CONGEN was run. Otherwise, the connections between the constructed atoms and their surroundings will be distorted. Also, the program will die if the conformation number is too large.

Examples

In these examples, let’s assume that the evaluation used in the conformational search was `ENERGY`.

To select the ninth conformation from a conformation file on unit 60:

```
XCONF UNIT 60 NUMBER 9
```

To select the second lowest energy conformation from the conformation file on unit 50:

```
XCONF UNIT 50 BEST 2
```

To select the conformation which has the lowest surface area among the conformations within 3 kcal/mole of the lowest energy conformations in the file on unit 43:

```
XCONF UNIT 43 BEST 1 ESURF 3
```

13.3 `MERGE CG` — Conformation File Manipulations

13.3.1 Syntax

```
MERGE CG repeat(IN unit [[int]:[int]]) OUT unit

                                        [       {MAIN} ]
      [SELEct atom-selection END] [TITLE] [ FIRST {COMP} ]
                                        [       {unit} ]
```

If the `TITLE` option is used, then a *title* must follow this command.

13.3.2 Function

The `MERGE CG` command reads a set of conformation files and writes a selected subset of the conformations to a single file. It is possible to change the initial title of the file, the reference coordinate set, and the atoms stored. See Section 12.3.4 [Conformation File], page 126, for details on the contents of the file.

The conformation file to be read are specified using the `IN` operands. Each `IN` operand consists of the Fortran unit where the file is to be read followed by an optional specification of the range of conformations to be read. The specification gives the starting and final conformation number. If either number is omitted, the program assumes either the beginning or the end of conformation file, as appropriate.

The conformation file to be written is specified by its Fortran unit number using the keyword `OUT`. You can change the opening title in the file by specifying the `TITLE` keyword, and following the command by a *title*.

The reference coordinate set for the output conformation file can be changed by using the `FIRST` option. You can change it to the current coordinate set (use `MAIN` keyword), the comparison coordinate set (use `COMP` keyword), or the reference coordinate set from one of the input files (use its unit number after `FIRST`).

By using the atom selection, one can eliminate atoms from the conformations. Only the selected atoms are written to the merged file. By default, all atoms are selected.

14 The Internal Coordinate Commands

The commands in this section can be used to construct cartesian coordinates from internal coordinate values. The internal coordinate data structure can also be used for analysis purposes. There are flexible editing commands for manipulating the data structure.

14.1 Syntax of Internal Coordinates Commands

```
{IC   } { SETUp                         }
{BUILd} { FILL    [COMP]                }
        { EDIT                          }
        { BILD    [COMP]                }
        { SEED atom atom atom  [COMP]   }
        {                               }
        { { DELete }  { BYNUM int [int]  }  }
        { { KEEP   }  { atom-list-spec   }  }
        {                               }
        { DIFF    [COMP]                }
        { PURGe                         }
```

```
atom ::= residue-number atom-name
```

See Section 11.7 [Atom Selection], page 108, for a description of the '`atom-list-spec`'.

The syntax for the EDIT subcommands are:

```
{ [DISTance]   atom  atom  real                        }
{ [BOND    ]                                           }
{ ANGLe        atom  atom  atom  real                  }
{ DIHEdral     atom  atom  [*]atom  atom  [DELTA] real }
{ END                                                  }
```

14.2 Purpose of the Various Internal Coordinate Commands

These commands are used to setup, modify and process the internal coordinates of the molecule. This operation is very useful in setting up atom coordinates whenever they are not known. This occurs when a protein structure is built from scratch or when an existing structure is modified. The modification can simply be a conformational change, or a change in the residue sequence through replacement,insertion, or deletion. Many of these modifications can be processed within the program as it currently stands. Other more difficult modifications can be facilitated by editing the internal coordinate file by using external programs.

The Internal Coordinate commands (except `SETUP` and `EDIT`) can only be used if internal coordinates exist (i.e. if the `IC` common is filled). This can only be filled by reading an `IC` file from disk, or by using the `SETUP` subcommand. The use of `SETUP` assumes that the residue sequence has already been generated. The information used to setup is obtained from the current residue topology file. If you change to a different residue topology file, you must do a `IC SETUP` before reading in the new topology file. Otherwise, you'll be reading internal coordinate information from the wrong place.

The subcommands are interpreted as follows:

SETUP Setup the internal coordinates using standard values from the parameter file, unless otherwise specified in the residue topology file (see Section 3.1.5 [RTF File Formats], page 27). The internal coordinates are built from the current RTF.

FILL Fill the internal coordinate values wherever possible from the known atomic coordinates. No changes are made to internal coordinates which have atoms that have unknown positions. If the COMP keyword is used, then the alternate coordinate set will be used to fill the IC data structure.

EDIT Edit the internal coordinate file. This command causes the input stream to transfer to the IC edit mode. The edit mode commands are:

```
DISTance atom atom real
ANGLE atom atom atom real
DIHEdral atom atom [*]atom atom real
END

atom::=residue_number atom_type
```

The first three commands will specify a particular internal coordinate value. The DIST and ANGLE commands require that such an IC already exists and will be modified only. The DIHE command will search for desired IC, and if it is not found a new set will be added (including the associated distances and angles). In either case the torsion angle value in the internal coordinate will be set to the value specified. The DELTA option specifies that the current torsion angle in the internal coordinates be added to the specified real number to make the new torsion angle. If the dihedral angle specified in the DIHEDRAL command is not found in the internal coordinates, then DELTA option has no effect. The optional * on the third atom denotes that this is the central atom of an improper dihedral angle (i.e. the angle is determined by planes 1-3-2 and 4-3-2, also the associated angles use the same pattern for this type). The END command is used to exit from the edit IC mode.

BILD This command determines the cartesian coordinates for all unspecified atoms from the data in the IC file (wherever possible). The user is responsible to make sure that the designation for all atoms is unique. In the case that the system is over specified, An atom is placed on the first opportunity. If it is desired to modify the position of atoms with known coordinates, the coordinates for those atoms must be reinitialized using the COOR INIT command. Again, if the COMP keyword is used, then the alternate coordinate set will be used and modified.

SEED When the cartesian coordinates are not specified for any atoms, the BILD command cannot be used to generate positions since all positions are determined relative to known positions. The SEED command specifies the positions of the three atoms . It puts the first at the origin, the second on the X-axis, and the third in the XY-plane. The three atoms must have entries in the IC file of (dist 1-2, angle 1-2-3, dist 2-3). The COMP keyword causes the alternate coordinate set to be used.

DELETE This commands deletes a specified set of IC's from the data file. The delete can be by number (using the BYNUM keyword and a range), or by atoms selection. Any IC that contains a selected atom will be removed.

KEEP The keep command is the logical opposite of the DELETE command. Its options are identical, except that the selected set of IC's is kept, and the remaining ones are deleted.

PURGE This command causes all partially specified IC's to be deleted. Only the IC's with all four atoms defined will be kept. The IC's that remain will be renumbered.

14.3 Internal Coordinate Concepts

Given the positions of any three atoms, the position of a fourth atom can be defined in relative terms (internal coordinates) with three values: a distance, an angle, and a dihedral specification. Where many atoms are connected in a long sequence (as in proteins) it is easiest to consider four atoms in a chain. If the positions of one end of the chain is known, it is possible to find the positions of all of the remaining atoms with a series of internal coordinate values. But in the more general case, where some central portion of a molecule is known it is necessary to be able work in both directions. This lead to the present form of the internal coordinate data structure (five values for four atoms) where if either endpoint is unknown and the other three atoms are determined, the position of the end atom can be found. The improper type of internal coordinate data structure was created for branching structures (as opposed to simple chains). Since there are roughly five values in the data structure for every atom it is clear that the positions are overspecified Keep this in mind when externally editing IC files. The program will use the first acceptable value when building a structure and ignore any redundancies. The `EDIT` commands will always modify all occurrences of each edited parameter.

14.4 Internal Coordinate File Structure

The internal coordinate file can be stored in either card or binary form. for most purposes the card form will be used (since it can be edited). There are two types of elements in the internal coordinate file, those that correspond to normal dihedral angles and those that correspond to improper dihedrals. They can be distinguished by the presence of a '*' just before the iupac name of the third (k) atom (its presence denotes an improper dihedral type). For each element there are four atoms (referred to as `I`,J,`K`,`L`) and five values. The values for ordinary elements correspond to: `R(IJ)`, `THETA(IJK)`, `PHI(IJKL)`, `THETA(JKL)`, and `R(KL)` respectively. For improper dihedral types, the values are: `R(IK)`, `THETA(IKJ)`, `PHI(IJKL)`, `THETA(JKL)`, and `R(KL)`. Notice that for the second type that the K atom is at the center of both angles. Elements of the IC file are symmetric with respect to inverting the order of the atoms except that for improper types only atoms I and L can be interchanged (also the sign of `PHI` must be changed since `PHI(IJKL) = -PHI(LJKI)`.)

15 The Coordinate Manipulation Commands

The commands in this section are primarily used for manipulating atoms. There is a wide range of commands and options. All of the commands except for the `HBUILD` command and the unit cell commands may be used on either the main coordinate set, or the comparison set. Some commands require both sets of coordinates.

15.1 Syntax of Coordinate Manipulation Commands

```
COORdinates { INITialize                       } [COMP] [atom-selection]
            { COPY                             }
            { SWAP                             }
            { AVERage  [ FACT real ]           }
            { TRANslate vector-spec [DIST real] }
            { ROTAte { vector-spec } PHI real   }
            {        { byatom-spec }            }
            { ORIEnt [MASS] [RMS] [SAVE name]   }
            { RMS    [MASS]                     }
            { RANDOM random-options             }
            { DIFF [ FACT real ]                }
            { XFORm name                        }
            { XINVert name                      }
            { SCALE real                        }
            { CONV real real real real real real}
            { EQUI                              }

random-options ::= [UNIForm] [SEED int] [SD real]
                   [NORMal ]

name ::= word

COORdinate  { TRAF-translate-unit-cell }   you-better-look-at-the-source

HBUILD hbond-spec

vector-spec ::= [XDIR real] [YDIR real] [ZDIR real]

byatom-spec ::= BYATom segid resid iupac segid resid iupac END
```

The syntax of the `atom-selection` is given in Section 11.7 [Atom Selection], page 108. The syntax of the `hbond-spec` is given in Section 5.1 [HBOND Syntax], page 49.

15.2 Purpose of the Coordinate Manipulation Commands

All of these commands (except the unit cell and `HBUILD` commands) allow either the main coordinate set (default), or the comparison set (`COMP` keyword) to be modified. The other coordinate set is only changed by the `SWAP` command and the `ORIEnt RMS` command when the specified atoms are not centered about the origin.

Each of these commands may also operate on a subset of the full atom space (except the `ORIEnt` and unit cell commands). The selection specification should be at the end of the command. The default atom selection includes all atoms.

15.2.1 The `INITIALIZE` command

The `INITIALIZE` command returns the coordinate values of the specified atoms to their start up values (the parameter, `ANUM` which is currently 9999.0). The main use of this command is in connection with the `BUILD BILD` command, which may only find coordinates for atoms with the initial value.

15.2.2 The `COPY` command

The `COPY` command will copy the coordinate values into the specified set FROM the other coordinate set.

15.2.3 The `SWAP` command

The `SWAP` command will cause the coordinate values of the specified atoms to be swapped with the comparison set.

15.2.4 The `AVERAGE` command

The `AVERAGE` command will generate a new coordinate set at a point along the displacement vector between the present coordinate set and the other set. The `FACTOR` value determines the relative step along this vecor. Its default value is 0.5 (a true average). A `FACTOR` value of 1.0 is equivalent to the `COPY` command. Negative or greater than unit positive values are also allowed.

15.2.5 The `TRANSLATE` command

The `TRANSLATE` command will cause the coordinate values of the specified atoms to be translated. The translation step may be specified by either X, Y, and Z displacements, or by a distance along the specified vector. When no distance is specified, The `XDIR`, `YDIR`, and `ZDIR` values will be the step vector.

15.2.6 The `ROTATE` Command

The `ROTATE` command will cause the specified atoms to be rotated about the specified axis vector through the origin. The vector can specified either in terms of its Cartesian components, using the `XDIR`, `YDIR`, and `ZDIR` options, or by the vector between two atoms using the `BYATOM` option. The vector need not be normalized, but it must have a non zero length. The `PHI` value gives the amount of rotation about this axis in degrees (in the right handed sense). Only the atoms specified will be rotated.

15.2.7 The ORIENT Command

The `ORIENT` command will modify the coordinate values of *all* of the atoms. The select set of atoms is first centered about the origin, and then rotated to either align with the axis, or the other coordinate set. The `RMS` keyword will use the other coordinate set as a rotation reference. The `MASS` keyword cause a mass weighting to be done. This will align the specified atoms along their moments of inertia. When the `RMS` keyword is not used, then the structure is rotated so that its principle geometric axis coincides with the X-axis and the next largest coincides with the Y-axis. The `SAVE` option will result in the coordinate transformation being saved under the given `name`, and it can be used by `XFORM` option described below. This command is used for preparing a structure for graphics and viewing, for finding RMS differences, for model building, and in conjunction with the vibrational analysis.

15.2.8 The `RMS` Command

The `RMS` command will compute the Root Mean Square (RMS) or mass weighted RMS coordinate differences between the selected set of atoms just as they lie. This differences from the `COOR ORIENT RMS` command in that no coordinate modifications are made and no translation is done.

15.2.9 The RANDOM Command

The RANDOM command will randomize the selected set of atoms around their current coordinates. Two probability distributions are provided, normal (Gaussian) and uniform. For the normal distribution, the SD option specified the standard deviation of the distribution. For the uniform distribution, the SD option specifies the range of possible variation in both directions, i.e., new coordinates will have values anywhere between −SD and +SD. The default distribution is normal. The SEED option specifies the random number seed used to initialize the random number generator. It defaults to 1234.

15.2.10 The DIFF Command

The DIFF command will compute the differences between the main and comparison set (or the reverse) and store this difference in the coordinate displacement arrays, also referred to as the normal mode arrays (see Section 10.4 [Normal Modes], page 91). Each difference is scaled by the FACT keyword value. The differences are not appended onto current values. The differences may then be printed, etc...

15.2.11 The XFORM Command

The XFORM command will apply a stored coordinate transformation to a set of atoms. The coordinate transformation is specifed by name, and can be created by the ORIENT option above or by a analysis comparison command, see Chapter 18 [Comparisons], page 173. This command is useful when you want to move a set of atoms onto a position that is determined by another set of atoms.

15.2.12 The XINVERT Command

The XINVERT command will invert a stored coordinate transformation. The transformation is specified by the given name. No coordinates are affected by this command. For example, this command may be used to invert the transformation to orient a set of atoms, so that one set of atoms could be placed in the space occupied by another set. E.g.

```
COOR ORIE SAVE A CLEAR ATOM 1 * *      ! Get transform for moving
                                       ! first segment to x axis.
COOR XINV A                            ! Invert the transform
COOR ORIE CLEAR ATOM 2 * *             ! Put second set on x axis
COOR XFORM A CLEAR ATOM 2 * *          ! Move second set onto position
                                       ! of first segment.
```

15.2.13 The SCALE Command

The SCALE command will multiply the selected coordinates in the system by the given scale factor.

15.2.14 The CONVERT Command

The CONVERT command will convert from fractional coordinates to Cartesian coordinates. This command requires six real numbers; A, B, C, ALPHA, BETA, GAMMA; which are the unit cell parameters for the crystal. It assumes the current coordinates are in the fractional cell system, and it converts the selected atoms to Cartesian coordinates.

15.2.15 The EQUIVALENCE Command

The EQUIVALENCE command will generate symmetry related coordinates of the molecule. To use this command, you must first create a PSF which has multiple copies of the structure you wish to replicate by symmetry, and you must have read in the coordinates for the first assymetric unit. The EQUIVALENCE will first read a line containing just the number of transformations. Then it will read that number of transformations in the following format:

```
ROTxx ROTxy ROTxz TRANSx
ROTyx ROTyy ROTyz TRANSy
ROTzx ROTzy ROTzz TRANSz
```

Each of the symbols above is parsed as a real number. The first transformation is expected to be the identity transformation and is not applied.

The coordinates are divided into blocks whose number equals the number of transformations. New coordinates for every block after the first are generated by multiplying the rotation matrix (`ROTxx`) by the coordinates (as a column matrix) of the first block and then adding the translation vector (`TRANSx`). Blank lines in this input are ignored.

The following input to CONGEN illustrates an example of generating coordinates for the alanines in a crystal of $P4_1$ symmetry.

```
READ SEQUENCE CARD A94P UNIT 5
Alanine
1
ALA
GENERATE A
READ SEQUENCE CARD A94P UNIT 5
Alanine
1
ALA
open unit 11 name ala.crd form read
read coor card unit 11
GENERATE B
READ SEQUENCE CARD A94P UNIT 5
Alanine
1
ALA
GENERATE C
READ SEQUENCE CARD A94P UNIT 5
Alanine
1
ALA
GENERATE D
```

```
COOR EQUI
4
 1  0  0   0
 0  1  0   0
 0  0  1   0

-1  0  0   0
 0 -1  0   0
 0  0  1   0.5

 0 -1  0   0
 1  0  0   0
 0  0  1   0.25

 0  1  0   0
-1  0  0   0
 0  0  1   0.75
```

15.2.16 The Unit Cell Coordinate Manipulation Commands

The command `COOR TRAN` is a relic of the old code in CHARMM. There are no other keywords, but all of these commands require further input in formatted, fixed field form.

`TRAN` generates translated coordinates for molecules. See the routine `TRAFAC` in the file '`CORMAN.FLX`' for details.

15.2.17 `HBUILD` Command

The `HBUILD` command is used to construct positions for hydrogens. Generally, x-ray crystallography cannot resolve the position of hydrogens. However, the position of many of the hydrogens can be determined from the local geometry, e.g. the peptide hydrogen is easy to construct. In other cases, one must examine the local environment to find the best position for a hydrogen, e.g. the position of the gamma hydrogen in serine.

The `HBUILD` command is a variant of the `HBOND` command in that it performs a complete determination of all the possible hydrogen bonds with a given set of options, and it uses this set to determine how best to place ambiguous hydrogen positions. This code depends on the current protein topology and parameter files, and will break if the parameter type codes are changed. It will not work with DNA or other molecules.

After using the `HBUILD` command, it is a good idea to minimize the hydrogen positions only. A sample input follows:

```
CONS FIX CLEAR ATOM * * H* NOT
MINI ABNR NSTEP 250 INBFRQ 25 IHBFRQ 25 CUTNB 8.0 CTONNB 98.0 CTOFNB 99.0
```

16 The Analysis Facility of CONGEN

This facility provides most of CONGEN's capability for analyzing structures and calculations done on them. If you've never used the analysis facility, see the introduction below. If there are terms you do not understand, consult the glossaries, Section 2.12 [Syntactic Glossary], page 12, and Section 2.13 [Glossary], page 13. The command notation uses the same meta syntax as described for all other free field CONGEN commands, see Section 2.2 [Meta-Syntax], page 6.

16.1 An Overview of the Analysis Facility

CONGEN provides a facility for analyzing the results of any calculation made as well as comparing one's results to any other calculation. This facility is general in the sense that it will work with arbitrary residues, any set of parameters, and will permit a broad range of comparisons. It provides all the features of Bruce Gelin's ANC program plus many others.

There are several important aspects of the design of this facility which are important to its users. First, the facility provides a small number of simple commands which can be combined to do a variety of tasks. Secondly, the program is well adapted is the job of comparing results, regardless of whether the results were obtained on the same system or on a homologous one. This will permit previously impossible comparative studies to be performed – such as comparing the dynamics of hemoglobin and myoglobin in homologous regions or comparing the results obtained from the explicit hydrogen or extended atom models.

These two design considerations dictate a great deal about the analysis facility's operation. The first consideration, being able to combine commands, require that the facility store the results of one operation so that it can be used in another. There are two data structures (see Section 2.13 [Glossary], page 13) that the facility uses to store such results, and they are important to understand.

The major data structure is the table. In analyzing the large amounts of data inherent in a macromolecule, we need a method for organizing it. Consider, for example, the 631 bond angles in bovine pancreatic trypsin inhibitor. Without a good ordering of these angles, it would difficult for a person to see any relationships in these angles. However, since a structure in CONGEN consists of a number of segments which, in turn, consist of a number of residues which a number of atoms or internal coordinates, we can organize the data along these lines.

Therefore, a table contains a list of segments which are identified by their segment identifiers as specified in the `GENERATE` command. Each segment contains a list of residues. The residues are named (GLY, ALA, etc.) and have identifiers as well. The identifiers are the character form of the sequence number of residue. Each residue in the table contains a list of data arrays where every array is "tagged". "Tagged" means that each array point has associated with it a character string which serves to identify it. The tags are easily constructed. For example, the tag for a bond is the IUPAC name for each atom in the bond separated by a dash. Each element of the array contains a property of the atom or of the internal coordinates. For example, the minimum energy and average length of bonds during a dynamics run are properties of bonds. The table also contains a title which identifies the entire table and is printed along with it.

Many operations can be performed on these tables. First, the `BUILD` command will generate a table. Currently, there are dozens of different tables which can be generated. Tables can be printed in several different ways using the `PRINT` command. Simple statistical information can be added to them using the `ADD` command. The `DELETE` command may be used to delete data from them so that one more easily study a subset of the data. Finally, the `SELECT` command may be used to select data from a table and record the results in the second major data structure, the selection.

The selection is another data structure which is a collection of data which is less organized than the table. The selection consists an array of numbers where every number has associated with it its position in the table as well as the residue to which it belonged. Two selections are provided in the analysis facility, and the following operations are supported: First, data may be selected from the table using `SELECT` command. Second, a histogram of the selected data may be made using the `HISTO` command. Third, using the `PLOT` command, the data in the selection may be plotted against its position on the table or against the residue number of the residue to which each data point belongs. Finally, the two selections may be plotted together using the `2DPLOT` command to yield a scatter plot. We can therefore make a scatter plot of any two sets of numbers; we are not limited to phi-psi plots.

In addition to analyzing the static properties of the structure which are maintained in CONGEN, the analysis facility can analyze the results from a dynamics calculation. Properties of the internal coordinates and atoms which are fairly easy to calculate can be built into a table. The `ACCUM` and `COMBINE` commands are used for preparing the data for inclusion into a table. Correlation functions may also be calculated using the `CORREL` command.

Complementing the above commands, there are commands which perform more isolated functions. The analysis facility has `READ` and `WRITE` commands for reading and writing data structures which are peculiar to it. There is a close contact search command, `SEARCH`, which searches for close contacts of atoms to other atoms or to spatial positions. There is a `DRAW` command which prepares input to the PLT2 plotting program, and MOLD, a molecule drawing program. The `SET` command may be used to change I/O units and the size of the page. Finally, there is a command, `DELIM`, which changes the command delimiter.

16.2 Using the Analysis Facility

To call the analysis facility of CONGEN, one places `ANALYSIS` as the first and only word on a command line (it may be abbreviated to four characters). Prior to calling the analysis facility, the user must have taken the following steps: A PSF must have been read or constructed. A parameter set and complete coordinate set are required also. If hydrogen coordinates are missing from the coordinate set or if hydrogen bonds are to be analyzed, a `HBONDS` command must have specified before the `ANALYSIS` command (see Chapter 5 [Hydrogen Bonds], page 49). Likewise, if non-bonded interactions are to be analyzed, an `NBONDS` command must be invoked prior to analysis (see Chapter 6 [Non-bonded Interactions], page 51). If the total energy per atom is to be analyzed, then both `NBONDS` and `HBONDS` must be specified.

Although the analysis facility has a command interpreter which is different from that in the main program, commands are specified in free field format, as in the main part of CONGEN, see Section 2.1 [Run Control], page 5. To exit the analysis facility, use the `END` command, which is just the word, `END`, on a command line by itself.

16.3 Messages and Errors

The analysis facility can generate many messages, warnings, and errors. The messages are self-explanatory. The warnings generally tend to be self-explanatory, but there are some warnings that have to do with the program logic which are obscure. Likewise, errors that relate to bad sets of user commands will give understandable messages; internal logic errors give incomprehensible messages. If you get an error message you do not understand, please see Bob Bruccoleri or mail to '`bruc@acm.org`'. Chances are that you have found a bug, and I should always be told about that.

17 Table Manipulation Commands

The table is one of the central data structures in the analysis facility. It is described in the introduction to the analysis section, see Section 16.1 [Introduction to Analysis], page 157. Here we describe the various commands available for construction and manipulation of tables.

17.1 Building Tables

The `BUILD` command generates a table. Each term in the energy function related to bonded interaction defines a class of tables where the data in the table are properties of the internal coordinate. Likewise, atoms define another class of tables, which contain data that are properties of atoms. Each class has its own rule for the selection of tags as well as criteria for deciding what residue a data point belongs to.

17.1.1 Syntax of the `BUILD` Command

```
BUILD [COMPARE] [PARM ] class repeat([WRITE unit-number] property del)
      [ DIFF  ] [IUPAC]

      [TITLE string del]

          {  BOND  }
          {  ANGLE }
class ::= {TORSION }
          {IMPROPER}
          {  HBOND }
          {  ATOM  }

property ::= string
```

Syntactic ordering: The operands must be specified in the above order, except for `TITLE` which can be specified anywhere after the class.

17.1.2 Table Classes and the Selection of Tags

The tags and placement of data for each of the classes is given below:

Class	Property of	Placement	Tags
BOND	Bond	Use first atom	Tags for atoms separated by dashes.
ANGLE	Angles	Use middle atom	As for bonds.
TORSION	Torsion angles	Use second atom	As for bonds.
IMPROPER	Improper torsion angles	Use first atom	As for bonds.
HBOND	Hydrogen bonds	Use donor atom	The tag for the donor atom is concatenated with a string giving its segment id, residue name, and residue id.
ATOM	Atom	Itself	See below

The placement gives the atom which is used to determine which residue the data belongs to. The tag for an atom is specified by the third word of the command. If nothing is specified or if

IUPAC is specified, the tag for an atom is its IUPAC name as given in the residue topology file. PARM specifies that the atom type name as given in the parameter file should be used.

17.1.3 Table Properties

Each class of table allows different sets of properties to be specified. Any number of properties may be specified in a table subject to the limits of what can be contained in memory and what can be easily read when printed.

There are two groups of properties, static properties and dynamic properties. The static properties are those properties which are calculated just from the structure, a parameter set, and one set of coordinates. The dynamic properties are those which are calculated using a dynamics trajectory. In this section, we will discuss only the static properties; the dynamic properties are discussed in the section on dynamics analysis, see Section 19.4 [Dynamical Table Properties], page 184. Also, a property may be read from a file, see Section 17.1.6 [Table I/O], page 163. Note that dynamic and static properties as well as properties read from a file may be freely mixed in the table.

17.1.3.1 Static Properties of Internal Coordinates

All the internal coordinate tables (BOND, ANGLE, TORSION, and IMPROPER) have the same set of properties. The syntax for these properties is as follows:

```
              {GEOMETRY}
property ::= { ENERGY }
              { NUMBER }
```

The GEOMETRY property returns the geometry of the internal coordinate. For example the BOND GEOMETRY is the bond length; the TORSION GEOMETRY is the torsion angle. The ENERGY property returns the energy of the the internal coordinate. For example, IMPROPER ENERGY returns the energy of the improper torsion angles. The NUMBER property returns the sequential position of the internal coordinate in the list contained in the PSF. For example, the numbers of the sulfur-sulfur bonds in a cysteine bridge will be very large, because they are added last. The NUMBER property is useful in figuring out error messages from the energy routines as the numbers the energy routines output correspond to the numbers returned by this property.

17.1.3.2 Static Properties of Hydrogen Bonds

The syntax for hydrogen bond properties is as follows:

```
              {DISTANCE}
property ::= { ANGLE  }
              { ENERGY }
              { NUMBER }
```

The DISTANCE property returns the heavy atom donor – acceptor distance. The ANGLE property returns the complement of heavy atom donor – hydrogen – acceptor angle. The ENERGY property is the hydrogen bond energy. The NUMBER property gives the sequential position of the hydrogen bond.

17.1.3.3 Static Properties of Atoms

The following table describes the syntax and function of the various properties available for atoms.

Syntax	Function
‘X’	X coordinate of atom.
‘Y’	Y coordinate of atom.

'Z' Z coordinate of atom.

'R' Distance of atom from geometric center. When this property is computed for a difference table, it returns the length of the vector between the two atoms being compared. If either atom in the pair has an undefined position, then the distance will be calculated as zero.

'NUMBER' Sequence number of atom.

'ENERGY, FORCE'
The total potential energy (or force) per atom. This is calculated as follows: we start with the non-bonded energy per atom. The energy of each bond is split evenly among the two atoms. The bond angle energy is summed onto the central atom. The torsion angle energy is split between the center two atoms. If the harmonic form of the improper torsion term is used, see Section 3.1.4.1 [Parameter File Format], page 21, then the improper torsion energy is summed to the first (center) atom. If the cosine form is used, then the improper torsion energy is summed to the third (center) atom. The harmonic constraint energy is is added to every atom. Finally, the hydrogen bond energy is split between the donor and acceptor atom. `FORCE` is magnitude of the force vector at each atom ($\sqrt{(\partial E/\partial x)^2 + (\partial E/\partial y)^2 + (\partial E/\partial z)^2}$). The individual force terms are summed prior to calculating the magnitude.

'EB, FB' Bond energy (or force) per atom - the energy of each bond split evenly among two atoms is summed for each each atom.

'ET, FT' Angle energy (or force) per atom - The sum of the energies of all angles whose central atom is the atom.

'EP, FP' Torsion angle energy (or force) per atom - The energy of each torsion angle is split among the two central atoms and summed over all the atoms.

'EI, FI' Improper torsion angle energy (or force) per atom. The energy of all improper torsions is summed onto the center atom.

'EHB, FHB' Hydrogen bond energy (or force) per atom - The energy of each hydrogen bond is split across each atom in the pair.

'EVDW, FVDW'
Van der Waals energy (or force) per atom - calculated the same way as `ELEC`.

'ELEC, FELE'
Electrostatic energy (or force) per atom - this is calculated by taking the pairwise electrostatic energy for each non-bonded pair and summing half of this energy on each atom in the pair.

'ELED, FELD'
Direct part of the electrostatic interaction. This is the same as `ELEC` if reaction fields are not being calculated.

'ERXN, FRXN'
The reaction field energy (or force). *WARNING:* the reaction field calculation was under development in CHARMM v.16. Please review the code before attempting any calculations with this option.

'ENB, FNB' Non-bonded energy (or force) per atom - sum of `EVDW` and `ELEC`.

'EC, FC' Harmonic constraint energy, or force.

'EUSER, FUSER'
The energy (or force) returned by the `USERE` subroutine. See Section 2.11 [CONGEN Modifications], page 11, for a description of `USERE` which provides more information on

writing user energy routines. Note that the `USERE` routine must be written to return the appropriate data for analysis of energies (versions of `USERE` routines which do not support the analysis of energies will run in the main part of the program even though they will bomb out if called from analysis).

'`EPBE`' The electrostatic energy of each atom as calculated by the Poisson-Boltzmann equation. This calculation is done by creating an empty charge density array, looping through each atom in the system, charging the array specifically for each atom, and calculating the electrostatic energy, see Section 8.3.3 [PBE ENERGY Command], page 78.

'`EPSPBE`' The actual dielectric constant for each atom as used by the Poisson-Boltzmann equation. This property is useful for checking the effects of adjusting dielectric constants based on exposed surfaces, see Section 8.3.1 [PBE SETUP Command], page 71.

'`SURFACE`' The accessible surface of the atom in square Angstroms as calculated by Lee and Richard's accessible surface program. A probe diameter of 1.4 A is used with a fractional error parameter of 0.05. These two parameters can be changed using the analysis set command, see Section 25.1 [Analysis Set Command], page 209. This surface includes the radius of the probe.

'`CONTACT`' The contact area as calculated by Lee and Richard's accessible surface program[1]. The contact area does not include the radius of the probe. The parameters are the same as in the SURFACE property.

'`RADIUS`' The Van Der Waals radius of the atom as used in the non-bonded energy calculation.

'`CHARGE`' The charge of the atom.

'`POLAR`' The polarizability of the atom as used in the non-bonded energy calculation.

'`NEFF`' The effective number of electrons as used in the non-bonded energy calculation.

'`VSGEPOL, ASGEPOL, MSGEPOL`'

These three table properties use the GEPOL algorithm[2345] to calculate three different surface properties of the atoms; van der Waals, accessible, and molecular surfaces. The van der Waals surface (keyword `VSGEPOL`) is the surface of the atoms calculated using the van der Waals radii and accessibility is ignored. The accessible surface (keyword `ASGEPOL`) is the area of the locus of the center of water probe. It is the same surface as calculated by the Lee and Richards algorithm referenced above. The molecular surface (keyword `MSGEPOL`) is the locus of points on the surface of the probe sphere when the probe is in contact with at least one atom in the molecule.

When calculating van der Waals and accessible surfaces, the GEPOL algorithm uses a points on tesselated sphere to calculate what parts of the sphere are exposed, and it adds all the contributions of the tesserae to determine the surface. The calculation of the molecular surface uses this accessible surface algorithm, but additional spheres are added to the calculation of the accessible surface, and these additional spheres closely define the molecular surface.

---

[1] B. Lee and F. M. Richards, *J. Mol. Biol.,* **55**, 379 (1971)

[2] J.L. Pascual-Ahuir, E. Silla and I. Tunon, *QCPE* 554, 1993

[3] J. L. Pascual-Ahuir and E. Silla GEPOL: An improved description of molecular surfaces. I. Building the spherical surface set. *J. Comput. Chem.*, **11** (1990) 1047-1060.

[4] E. Silla, I. Tunon and J. L. Pascual-Ahuir GEPOL: An improved description of molecular surfaces. II. Computing the molecular area and volume. *J. Comput. Chem.*, **12** (1991) 1077-1088.

[5] J. L. Pascual-Ahuir, I Tunon and E. Silla GEPOL: An improved description of molecular surfaces. III. A New algorithm for the computation of the Solvent-Excluding Surface. To be submitted to *J. Comput. Chem.* during 1993

The GEPOL command, see Section 27.15 [Gepol Command], page 239, may be used to set operating parameters for the GEPOL algorithm.

'RVSGEPOL, RASGEPOL, RMSGEPOL'
These three table properties are the "relative" equivalents of VSGEPOL, ASGEPOL, and MSGEPOL. In this context, "relative" means relative to the surface area of a sphere of radius equal to the atomic radius in the case of the van der Waals surface and molecular surface, and relative to a sphere of radius equal to the atomic radius plus the solvent probe radius in the case of the accessible surface.

17.1.4 Execution Time of the BUILD Command

The BUILD command executes very quickly taking on the order of seconds for a table with several properties. The only exception to this is the SURFACE, CONTACT and GEPOL properties. For PTI, the calculation of Richard's surface takes about 15 seconds for the explicit hydrogen model (580 atoms) on a Iris 200 series workstation. The GEPOL molecular surface takes minutes on a R4000 based Silicon Graphics workstation.

17.1.5 DIFF and COMPARE Build Table Options

Normally, the BUILD command will build a table using the main calculation. However, if one specifies COMPARE, the comparison data structures will be used for the table generation. The specification of DIFF directs that a difference table be built. Difference tables require different handling than standard tables. The data in the table results from subtracting the data generated using the comparison calculation from the data generated from the main calculation. I.e. TABLE = MAIN - COMPARISON. Further, atoms must be paired properly in order to ensure that the correct quantities are subtracted. Let us examine the use of the atom pairs more closely.

If a table is constructed from atom properties, we can use the atom pair list directly to define what data points are equivalent, and therefore, should be subtracted form one another. For difference tables of internal coordinate properties (we consider hydrogen bonds to be internal coordinates in this discussion), the process of finding equivalent internal coordinates is more complex. It can be done as follows: for every internal coordinate in the main calculation, we look up each of the atoms in the atom pair list. If any atom is not found, we ignore that internal coordinate. From the equivalent atoms, we construct the equivalent internal coordinate for the comparison structure. The presence of the equivalent internal coordinate in the comparison structure indicates that we have found an equivalent pair. The pairing that is generated after examining all internal coordinates is used to relate the numbers generated for non-atom properties.

The other accommodations that must be made for the difference tables results from the various identifiers used in the table. Segments in the table correspond to a segment pair, so two segment identifiers must be kept for every table segment. Each residue in the table requires to two residue names and two residue identifiers. Likewise, the tags for atoms now refer to an atom pair. If the tags for both atoms in a pair are the same, then the tag for the pair is the same as the tag for one atom. If the atoms are different, the tag is formed by concatenating the tags for both atoms and separating them by a colon. Finally, the table is marked as a difference table so that the PRINT command can handle the double segment and residue identifiers correctly.

17.1.6 Table Input and Output

In order to allow users to input their own data into a table and to save the output from a particularly long property calculation (such as an accessible surface), the BUILD command allows properties to be read and written from files. The organization for such files is as follows:

```
title              (up to 10 lines)
*                  (line must be blank after asterisk)
```

```
short header      (80A1)
long header       (80A1)
number of items   (I5)
format            (80A1)
data              (using format given above. One per line)
```

The title is short description of the file. The text is printed when the file is read. The short header is the header used over columns in a column printout (see Section 17.2 [Printing Tables], page 164). The long header is the title for printout when the table is pretty printed (see below). The number of items must be the same as the number of entities in the structure for the particular class. For example, for a table of class, `ANGLE`, the number of items must be equal to `NTHETA`, the number of angles in the PSF. The data uses an `F15.0` format to allow 8 digits of accuracy plus exponents.

The order of the data is determined by the order in which CONGEN keeps its list of internal coordinates, hydrogen bonds, or atoms. For example, the bond list will contain the list of bonds for the chain of a protein followed by the bonds for any disulphide bridges which may been patched in later. If one is preparing a property input, it must follow this ordering exactly. If a bond is deleted, as is the case for the first bond in an extended atom protein, one must have an entry for it anyway in the property input. The `NUMBER` static property, which is defined for all classes, gives the correct position for each item in the table. In addition, when a property is written out, identifying information is written as well. Therefore, a property file may also serve as a guide for the order of the data.

The syntax for a property to be read from a file is as follows:

```
property ::= READ unit-number
```

The `WRITE` option, which may be specified with any property, causes that property to be written to the given unit. The title used in writing that property is the one given by the `TITLE` option in the `BUILD` command. Only one title may be specified per `BUILD` command, so two different properties written out will have the same title. As a result, it is advantageous to write only one property at a time with the `BUILD` command. It is permissible to specify the `WRITE` option for a property which is being read in. The title is processed the same way title in the analysis command, `WRITE`, are processed.

The use of difference tables necessitates some different processing for the `READ` properties and the `WRITE` option. Since a difference table requires the calculation of two sets of properties, the `READ` property must use two units. Therefore, the property for the main calculation is read from the given unit, and property for the comparison calculation is read from unit plus 1. For example, if we are comparing hemoglobin to myoglobin, and hemoglobin is the main structure, then READ 24 will read the hemoglobin property from unit 24 and the property data for myoglobin will come from unit 25.

The `WRITE` option is not permitted with a difference table.

17.2 Printing Tables

The `PRINT` command prints a table. There are two different methods available for printing a table. The first method is called pretty printing the table which can only be used for one property at a time. The table is printed in accordance with its hierarchical organization. The second method is the multiple column format. This allows one to print all the data in the table, but in a more amorphous form.

17.2.1 Print Command Syntax

```
PRINT TABLE

    {PRETTY [PROPERTY property del] [BYRES] [TAGVAL] [CUTOFF real] }
    {                                                               }
    {COLUMN [SORT repeat([DESCEND] property del) deldel]            }
    {       [IGNORE repeat(property del) deldel]                    }

    [SYSTEM string del] [FRAC integer]
```

17.2.2 Pretty Table Printing

In order to pretty print the table, one uses the options associated with the `PRETTY` as specified in the syntax above. Since pretty printing can only be done with one property, the `PROPERTY` option must be specified if the table has more than one property. The table printing can be done in four different ways depending on the appearance of `BYRES` and `TAGVAL` in the command line. Let us consider the case when neither option is specified.

If we had an arbitrarily wide piece of paper, a readable way of outputting all the data in a table would be to have one column for every tag in the table, one row for every residues' data, and a page for every segment. Each data point would be output under its tag, and it would be easy to see how a particular type of data varied from one residue to the next.

However, we do not have infinitely wide paper. We can accommodate this shortcoming by taking advantage of the fact that some tags appear very frequently. The program selects those tags which appear most frequently and uses these to be the headers of the columns. As each residue's data is printed, the program checks to see if any of the data's tags are the same as any of the headers. If they are, the data is printed in that column. Any data which is not printed in these columns is printed at the right of the page in tag-value format. Tag-value format means that for each data point, the tag is printed alongside the data so that one can see what the numbers stand for.

The `TAGVAL` option specifies that the entire table is to be printed in tag value format. No selection of column headers is made. The `BYRES` option causes the residues to be collected into groups of the same name and printed in alphabetical order of the residues. The selection of headers is done on the frequency of tags within each groups rather than with the entire segment. As a result, using the `BYRES` option will increase the amount of data printed in columns. `BYRES` and `TAGVAL` may both be specified which means that the collection of data by residue still occurs, but the data is output using tag value format.

The `CUTOFF` option is used to mask out numbers whose magnitude is small. When a cutoff is specified and the magnitude of a data point in a column is smaller than the magnitude of the cutoff, an `M` is printed in place of the number. If a data point is to be printed in tag value format and its magnitude is smaller than the cutoff's magnitude, the tag value pair is not printed at all.

17.2.3 Multiple Column Printing

Using multiple column output, the `PRINT` command can print all of the properties in a table at once. This output format is specified using the `COLUMN` option and all options associated with it as indicated in the syntax of the command, see Section 17.2.1 [Print Command Syntax], page 164. The multiple column output is as follows: Each property in the table is assigned a column. In addition, the segment identifier, the concatenation of the residue name and residue identifier, and the tag are also assigned columns. On output the data in each array is printed along with segment, residue, and tag all in a line. If more than one line of data will fit on the page, then they will packed across the page so they make use of the available space. If the line is too long for the line size as specified for the page, the line will be broken across multiple printer lines and each line of data will be followed by a blank line.

The table is identified with a title which specifies which class table is being printed, and every column of data is titled individually. If the lines of data are too long to fit in the line size provided, the titles over the columns will be broken over several lines as well.

Using this method of outputing the table, it is possible to sort the data. One may use the `SORT` option specify that the table is to be sorted on particular properties. In addition, one may specify sorting on the segment identifier, residue name, residue identifier, and tag. The default order for sorting is ascending. However, one may specify that a particular property should be sorted in descending order. The `IGNORE` option may be used to have certain properties not be included in the printout. This option also allows the segment, residue, and tag to be specified. The specifications for these additional properties is as follows:

Property	Function
`SEGID`	Segment Identifier
`RES`	Residue Name
`RESID`	Residue Identifier
`TAG`	Tag

Note that for the `IGNORE` option, to eliminate the printing of residues, one must specify both `RES` and `RESID`.

17.2.4 Print Table Options

This section describes options that pertain to both forms of table printing.

The `SYSTEM` option helps to further denote what is being printed out. The string specified by this option is printed along with the table's title at the top of each page.

The `FRAC` option gives the number of fractional digits in the format used to output data points in the table. The selection of formats used by the printing routine is automatic in that the width of the format is determined by the magnitudes of largest positive and negative numbers in the table. The default setting for `FRAC` depends on the table. If the table has just one property which is the `NUMBER` property or if pretty printing is done with a `NUMBER` property, the default value for FRAC is zero; otherwise, it is three.

The Fortran I/O unit on which the table is printed, the line size of ultimate print out device, and number of lines per page are all adjustable. Section 25.1 [Analysis Set Command], page 209, for more information.

17.3 Adding Statistical Information to the Table

This command will add statistical information to the table. Each clause in the command specifies what information is to be gathered, how it is gathered, and where in the table it is placed. Each clause is evaluated in parallel, i.e. any information added by one add clause is not seen as the other add clauses are evaluated.

17.3.1 Syntax of the `ADD` command

```
ADD repeat(add-clause deldel)

add-clause ::= STATS repeat(stat-option) del
```

```
                  {          ALLTAG              }
         [COLLECT {          EACHTAG             } del ]
                  {name [EXCEPT] repeat(tag-pat) }

         [PLACE [RESIDUE] [SEGMENT] [STRUCTURE] del]

                { AVE }
                { RMS }
                { SUM }
stat-option ::= { SD  }
                { MIN }
                { MAX }
                { M3  }
                { M4  }
                { ALL }

name ::= string with no blanks

tag-pat ::= string with no blanks
```

17.3.2 Selection of Table Statistics

The `STATS` part of add clause determines what statistics are collected. The following table gives the meaning of each of the '`stat-options`':

Option	Statistic
`AVE`	Average
`RMS`	Root mean square
`SUM`	Sum
`SD`	Standard deviation
`MIN`	Minimum
`MAX`	Maximum
`M3`	Third Moment (Skewness)
`M4`	Fourth Moment (Excess)
`ALL`	All of the above

17.3.3 Collection of Table Data

The `COLLECT` clause specifies what the data the statistics are collected over and, indirectly, how the statistics are tagged. There are three options permitted.

The `ALLTAG` option mean that statistics are to be collected for every data point irrespective of tag. The tag assigned to a particular statistic is the same as the '`stat-option`' (except of course for `ALL`).

The `EACHTAG` option specifies that statistics are to be collected individually for each tag that appears. The tag for these statistics is the tag of the data point followed by a colon followed by the '`stat-option`'.

The final option allows one to select a set of tags for which data is collected and do statistics on that set. The tags are specified using tag patterns which allow the same wild cards as atom or cell selections, see Section 11.7 [Atom Selection], page 108. The tag used for this option is the same as the `name` specified in the command, and therefore, specifying more than one '`stat-option`' will result in an ambiguous table. The use of `EXCEPT` with this option allows one to collect data over all tags except those specified. This option may be used for collecting statistics over the backbone or sidechain of a protein.

The range of data over which the statistics are collected is determined by the `PLACE` option described in the next node.

17.3.4 Placement of Table Statistics

The `PLACE` option determines two things - the range of the table over which the collection of data occurs and where the statistics get added. When `RESIDUE` is specified, the statistics are collected over every residue and the new data is placed in every residue. Therefore, with `ALLTAG`, the statistics are collected for every data point in each residue. With `EACHTAG`, data is collected for each tag separately, and statistics for each tag are added to every residue (this particular combination of options is not very desirable.). With named collection, the data is collected for all matched tags in each residue and then added to each residue.

When `SEGMENT` is specified, the same rules apply as for `PLACE`'ing in residues except that the collection is over segments and statistical information is placed differently. A new residue is created in every segment. The residue has the name `STAT` and identifier `SEG`. All the statistical information for the segment is placed in this residue.

When `STRUCTURE` is specified, the collection operations occur over the whole table. A new residue is in the last segment of the table whose name is `STAT` and whose identifier is `ALL`. The statistics are placed in this residue.

Each of these options may be used in any combination in an add clause, and there is only one minor limitation in combining add clauses in an `ADD` command. That limitation is that `EACHTAG` may be specified only once for each possible placement.

17.4 Deleting Data from Tables

The `DELETE` command is used to delete information from a table. This command is useful for omitting data from further analysis because one is not interested in it at the time. For example, if one is interested only in the results of a statistical calculation made by the `ADD` command, one can delete everything except the statistical information using the `DELETE` command.

17.4.1 `DELETE` Command Syntax

```
DELETE [COMPare]

        [SEGID [EXCEPT] repeat(segid) del]

        [RESID [EXCEPT] [ALLSEG] repeat(resid) del ]
                        [segid ]

        [RESNAME [EXCEPT] [ALLSEG] repeat(resname) del ]
                          [segid ]

        [TAGS [EXCEPT] [ALLSEG] {ALLRES} repeat(tag) del ]
                       [segid ] {resid }
```

```
                                             { { LT }      }
                                             { { GT }      }
          [VALUE [ABS] [PROPERTY propst del] { { LE } real } deldel ]
                                             { { GE }      }
                                             { { EQ }      }
                                             { { NE }      }

          [IDENT cell-selection del]

     cell-selection ::= atom-selection
```

Syntactic ordering: The ordering of options in the sub commands cannot be changed except for the `VALUE` sub-command; there, the ordering is immaterial.

For the syntax of the '`atom-selection`', please see Section 11.7 [Atom Selection], page 108.

17.4.2 Deletion from Standard Tables

Each of the six options specifies deletion by different rules. However, in many of the options, the word, `EXCEPT`, may be specified. Its usage dictates that the complement of whatever has been specified will be deleted. This allows one to specify what parts of the table one wishes to keep in a concise way.

17.4.2.1 Deletion of Segments

Deletion by the `SEGID` option specifies that entire segments in the table are to be deleted. One specifies what segments to delete by the segment identifier. If `EXCEPT` is specified, then only those segments which are named are kept; all the others are deleted.

17.4.2.2 Deletion of Residues

Deletion by `RESID` or `RESNAME` option specifies that residues are to be deleted. Each option may specify either one segment from which residues are deleted or all segments using the `ALLSEG` word. The difference between the two options is that `RESID` specifies deletion by residue identifier; `RESNAME` specifies deletion by the residue name.

The `EXCEPT` sub-option works in a more complicated way for these two commands. In order to understand what is done, we must examine the algorithm more closely. Before these two commands are interpreted, two marking arrays are allocated. All residues which are specified by the `RESID` sub-command are marked in the first mark array, and all residues specified in the `RESNAME` command are marked in the second mark array. If `EXCEPT` is specified in the `RESID` option, then its mark array is complemented. Likewise, if `EXCEPT` is specified in the `RESNAME` option, then its array is complemented. Then, any residue which is marked in either array is deleted.

For example, if we build a table on atom properties of pancreatic trypsin inhibitor (PTI), the command,

```
DELETE RESID EXCEPT 1 2 3 4 5 6 7 8 9 10 -
       11 12 13 14 15 16 17 18 19 20 $ -
       RESNAME EXCEPT GLY ALA
```

will delete every residue after the twentieth and every residue which is not a glycine or an alanine. In other words, the table will consist of the glycine and alanine residues in the first 20 residues of PTI.

17.4.2.3 Deletion of Data

There are three methods provided for deleting data in a table, by tag, by value, or by identifiers. By tag allows you select data for deletion based on its tag in the table; by value allows you to delete data based on a relational test on values of properties; by identifier allows you to delete by all the identifiers in the table using the atom selection syntax described in Section 11.7 [Atom Selection], page 108.

The `TAGS` option specifies the deletion of data by tag. After the optional `EXCEPT`, one may use `ALLSEG` or a segment identifier. `ALLSEG` means that the search for the tag will go over all segments; a segment identifier means that the search will take place over only the specified segment. Next, one must specify either `ALLRES` or a residue identifier. If we specify `ALLRES`, then the search is done over all residues within the searchable segments. If a residue identifier is specified, then only residues with that identifier will be searched. Finally, all data points within the searched residues which have tags as specified in the command will be deleted. If `EXCEPT` is given, then all data points which are found during the search are kept, and all others are deleted.

The `VALUE` option specifies deletion by property values. The marking of cells to be deleted is done by comparing values for properties against a real number you specify using a relational test you specify. All six Fortran relational tests are specified with the data in the table going on the left side of the comparison and the number you specify on the right. For example, `LT 10` will delete all cells in which the property value is less than 10. The relational test is obligatory for this command.

The specification of a property to be checked is optional. If a property is specified, only that property's values will be checked; otherwise, all properties in the cells will be checked. The `ABS` option specifies that the absolute values are to be taken of numbers before they are compared.

For example, the following two commands will produce a difference torsion angle table where all torsion angle changes whose magnitude is less than 10 degrees will be eliminated. This would be useful for looking for conformational changes between coordinates.

```
BUILD DIFF TORSION GEOMETRY $ ENERGY $
DELETE VALUE PROPERTY GEOMETRY $ ABS LT 10 $$
```

The `IDENT` option allows deletion based on all the identifiers at once. A variant of the atom selection syntax is used, see Section 11.7 [Atom Selection], page 108. Recall that the structure of a table mirrors that of the PSF – segments composed of residues composed of cells (atoms in the PSF). For this option, the table is treated as if it were a PSF where the tags replace the atom names. The same atom selection syntax can then be applied. Any tags included in the selection are deleted from the table. The initial state of the selection is empty, ie. `DELETE IDENT` ***`del`*** does nothing.

For example, to set up a table which has data only from residues whose resid is divisible by 5, one would build the table and then use the following delete command:

```
DELETE IDENT CLEAR CELL * #5 * CELL * #0 * $
```

17.4.2.4 Cleaning of the Table

Once the deletion process is complete, all empty entries in the table are deleted. I.e., any residues which have no cells are deleted, and any segments which have no residues are deleted. This simplifies the use of the `SELECT` command, see Section 22.1 [Select Command], page 199, when it is used for making two dimensional scatter plots.

17.4.3 Accommodations for Difference Tables

The form of the `DELETE` command for difference tables is similar to that of standard tables except that one must decide where segment and residue identifiers come from. In a difference table, there are two sets of these identifiers along with two sets of residue names. If you specify nothing, the command will use the identifiers from the main set of data structure. On the other hand, if you specify `COMPARE` in the `DELETE` command, the identifiers will be taken from the comparison identifiers.

18 Comparisons

The COMPARE command is used to specify that a comparison is to be performed. The comparison may be done within the current calculation or may be done with a different PSF, a different coordinate set, a different parameter set, a different hydrogen bond list, a different non-bonded list, a different set of constraints, a different set of averaging data structures, a different Poisson-Boltzmann data structure or any combination thereof. In addition, the COMPARE command can be used to reorient an entire coordinate trajectory with respect to a reference coordinate set.

Before one uses the COMPARE command, it is important to understand what the the program needs to do a comparison. When the analysis facility is called, it needs at least seven data structures; a PSF, a coordinate set, a set of harmonic constraints, a parameter set, a parameter coding set, hydrogen bond list, and a non-bonded list. It may also require the Poisson-Boltzmann data structure or one of dynamical average data structures. The function of the COMPARE command is to create another set of these seven data structures to be used for comparison calculations, and to establish a relationship between the two sets of data structures. Let us examine each of these tasks in turn.

The creation of the comparison data structures is a very straightforward operation. The COMPARE command allows one to specify Fortran unit numbers which refer to binary files for the comparison PSF, coordinate set, harmonic constraints, parameter set, hydrogen bond list, and non-bonded list. The parameter coding list is generated without user intervention. In addition, the user may generate a new hydrogen bond list or non-bonded list with different selectional rules than were used in the main calculation.

If a user does not specify how to create a particular data structure, the data structures are taken from the main calculation (in the case of the PSF, constraints, coordinates, and parameters) or generate them using parameters as specified in the main calculation (the case for the hydrogen bond and non-bonded list). In addition, if any dynamical averaging data structures or Poisson-Boltzmann data structures have been created and the comparison PSF is the same as the main, then the comparison dynamical averaging data structures and Poisson-Boltzmann data structures will be assigned to those of the main.

The relating of the two calculations is more involved. Basically, if one has a pairing between the atoms in the main PSF and the atoms in the comparison PSF, it is possible to relate any internal coordinate or hydrogen bond. This pairing is generated in a three step process where we first match segments, then residues, and finally atoms — mirroring the hierarchy of organization used in the PSF.

18.1 COMPARE Command Syntax

```
COMPARE

    [PSF unit-number] [PARM unit-number] [CONS unit-number]

          {   MAIN    }
    [COOR { coor-spec } [STORe] del]
          {  COMPare  }

    [SEGMATCH repeat(segid segid) del ]

    [repeat(resmatch-command)]

    [repeat(atomatch-command)]
```

```
                                                [FIRSTU  unit-number]
            [SIDE ]                             [NUNIT integer      ]
   [COORMATCH [BACK ] [SAVE name] [ DYNAMICS  [OUTPUTU unit-number] [MASS] ]
            [NOROT]                             [NFILE integer      ]
                                                [SKIP integer       ]

   [HBOND [UNIT unit-number] hbond-specs del]
          [      COPY      ]

   [NBOND [UNIT unit-number] non-bond-specs del]
          [      COPY      ]

   [PBE [UNIT unit-number] del]
        [NAME word       ]

   [VERBOSE]

resmatch-command ::=

    RESMATCH {  ALLSEG   } [PRINT] [       NOHOMOLOGY        ]
             {segid segid}         [HOMOLOGY homology-options]

             repeat(resid resid) deldel

homology-options ::= [range-spec] [cons-spec] [prot-spec]

range-spec ::= RANGES repeat(range-set) del

range-set ::= resid resid  resid resid

cons-spec ::= repeat(CONSERVE repeat(resname) del)

prot-spec ::= PROTECT repeat(resid resid) del

atomatch-command ::=

             [  ALLSEG   ] [        ALLRES         ]
    ATOMATCH [segid segid] [   RESID resid resid   ]
                           [RESNAME resname resname]

             [ONLY] repeat(iupac iupac) del

name ::= word
```

Syntactic ordering: The operands of the 'resmatch-command' and 'atomatch-command' must be specified in the given order, except for the PRINT option in the resmatch-command.

The 'coor-spec' is a file specification for coordinates. See Section 3.1.1 [Read Syntax], page 15, for the detailed syntax.

The 'non-bond-specs' are described in Section 6.1 [Non-bonded Syntax], page 51.

The 'hbond-specs' are described in Section 5.1 [HBOND Syntax], page 49.

NOTE: The syntax above requires that single delimiters sometimes be placed next to double delimiters. In this case, be sure to leave a space between the single delimiter and the succeeding double delimiter. `$ $$` is OK; `$$$` or `$$ $` is not.

18.2 The Creation of the Comparison Data Structures.

The `PSF`, `CONS`, and `PARM` keywords give the unit numbers of the comparison protein structure file, harmonic constraints, and parameter set; respectively. They must all be binary files created by a `WRITE` command to the main part of CONGEN, see Section 3.2 [Write Command], page 37. If any are not specified, the corresponding comparison data structure is presumed to be the same as the main data structure.

The `COOR` keyword specifies where the comparison coordinates come from and, optionally, where they are to be placed when the command finishes execution. The comparison coordinates can come from three sources; the main set, the comparison set in the main part of CONGEN, or from an external file. If the `COOR` option is left out or if you use the option `MAIN`, the comparison coordinates come from the main set. If you specify `COMPARE` in the `COOR` option, the coordinates come from the comparison coordinates in the main part of CONGEN, see Section 2.5 [Data Structures], page 7, and Chapter 15 [Coordinate Manipulations], page 151, for more details about this set. Otherwise, the coordinates will be read from a file. You are free to use either card image or binary files for these coordinates; but be sure that all coordinates you need are specified. See Section 3.1.3 [Read Coordinates], page 18, on the format of the coordinate files.

The `NBOND` and `HBOND` sub-commands are used to specify the comparison non-bonded and hydrogen bond list, respectively. The `UNIT` option is each of commands allows the specification of a Fortran unit number which refers to the binary file containing the lists. The `COPY` option requests that the main data structure be assigned to the comparison data structure. If neither option is specified, the program will generate a new list. The '`nbond-specs`', see Section 6.1 [Non-bonded Syntax], page 51, and '`hbond-specs`', see Section 5.1 [HBOND Syntax], page 49, specify generation and sigmoid switching function parameters.

When either of these data structures are read in, it is possible to modify the settings for the various switching function and energy evaluation parameters. Simply specify the values you want changed. You cannot change the selection method when something is read in (that doesn't make sense anyway). In the case of the hydrogen bonds, the file does not contain any of the selection and switching function parameters. If you specify a selection option, it will ignored even though you will get a message telling you what the values the program read are. Please ignore such information.

If any of the options or cutoffs for the `HBOND` or `NBOND` commands are not specified, the reference structure parameters will be used. If no `HBOND` or `NBOND` command is given, the list will be generated for a comparison PSF or coordinate set using the method employed for the reference structure.

The `PBE` keyword is used to specify the Poisson-Boltzmann data structure for the comparison structure. The data structure may be read from a disk file or may be recalled from an internal symbol. The `UNIT` keyword specifies that the Poisson-Boltzmann data structure be read from a binary file created using the `PBE WRITE` command, see Section 8.3.5 [PBE WRITE Command], page 78. The `NAME` keyword specifies that the data structure be taken from a symbol as specified by the `PBE SAVE` command, see Section 8.3.7 [PBE SAVE Command], page 81. If no option is specified and if the comparison PSF is the same as the main PSF, then the comparison PBE data structure will be assigned to the main PBE data structure

18.3 Matching of the Comparison Data Structures

The matching process begins after the `PSF`, `COOR`, and `PARM` options are handled. Segments are matched first.

The `SEGMATCH` command specifies how segments are to be matched. In the absence of a `SEGMATCH` command, the segment are paired by matching the segments which have the same identifier. If the command is given, the pairs of segment identifiers dictate the matching of segments.

Once the segments are matched, we must match the residues in every matched segment. Central to this problem is the finding of homologies between two sequences.[1]

The problem of finding homologies has been attacked by many others. The method used in the analysis facility is a modification of the string to string correction problem,[2] a problem well known in the computer science literature. The string to string correction problem can be stated as follows: Given a cost function for deleting any character from a string, a cost function for inserting a character into a string, and cost function for changing some character in the string into another character; find the lowest cost sequence of insertions, deletions, and changing operations that transform one string into the other. The cost functions in this algorithm are general in that they specify a different value for every possible character or pairs of characters. This problem is solvable in time and computer storage proportional to the product of the lengths of the strings.

We can see that this problem is applicable to finding a homology. As a protein genetic code evolves, insertions, deletions, and changes occur in its sequence. The best homology is one which minimizes these mutations. The algorithm calculates what is needed, a matching of residues that gives the lowest cost transformation of one sequence into another. Further, by establishing an appropriate change function, we can find homologies which permit conservative changes.

With the homology finding process understood, we can turn to the `RESMATCH` command. Any number of `RESMATCH` commands may be specified up to the number of segment pairs. Each command can specify the matching of residues in one segment pair or all. The first part of a `RESMATCH` command specifies what segment pair the command applies to. A segment identifier pair refers to that pair only; `ALLSEG` refers to all segments pairs which have not yet been processed. The `RESMATCH` keeps a record of which segment pairs have been specified in a command; any which are not specified are matched by finding the homology between their sequences. Further, each segment can be matched only once.

The next part of the `RESMATCH` command specifies how the homology should be found. If `NOHOMOLOGY` is specified, then the homology finding step is bypassed. If `HOMOLOGY` is not specified either, then the homology is taken over the complete sequences in the segment pairs. When `HOMOLOGY` is specified, one may specify the range of residues over which the homology should be taken. The ranges are specified as pairs of pairs of residue identifiers. The first pair gives the starting and ending residues in the first segment; the second pair gives the starting and ending residues for the second segment. More than one set of ranges may be specified.

Any number of conserved sets of residues may be specified using the `CONSERVE` command. Each use of `CONSERVE` specifies that all residues whose names are given may be considered equivalent when the homology is found.

It is possible to raise the penalty by a factor of 10 for making insertions and deletions by using the `PROTECT` option. The `PROTECT` option specifies pairs of residues in the main sequence which can only be deleted with a high penalty or which can have insertions within the specified range with a high penalty. Multiple ranges can be specified. This option is most useful when one wishes to conserve regions of secondary structure.

[1] You can use the `homology` program, see Section 29.1.5 [Homology], page 250, to find the homology between two sequences.

[2] R. A. Wagner and M. J. Fischer, "The String To String Correction Problem", *J. Association of Computing Machinery* **21**, 168-173 (1974).

The `PRINT` option, if specified, causes all calls to the homology finding subroutine in a `RESMATCH` command to print the matching of the sequences, an alignment showing the relationship of the sequences, and a set of `SPLICE` commands, see Section 13.1 [Splice Command], page 143, which will effect the transformation from the main sequence to the comparison sequence.

Finally, in addition to the homology operations, one may specify that certain residues are to be matched. The residue identifier pairs at the end of a `RESMATCH` command match those residues.

Once the matching of residues with segment pairs has taken place, we can tackle the final step of matching the two PSF's — matching their atoms. If no `ATOMATCH` commands are given, the default action is to use the IUPAC nomenclature for each atom in the residues and match atoms which have the same IUPAC name. This is usually acceptable as the IUPAC nomenclature results in many equivalent atoms being properly matched.

With `ATOMATCH` commands, one can tailor the atom matching process as well. Any number of `ATOMATCH` commands, up to the number of residue pairs, may be specified. As with `RESMATCH`, the `ATOMATCH` command processor keeps a record of what residue pairs have had their atoms matched. Any residue pairs which have not been matched at the end of `ATOMATCH` processing have their atoms matched by the default algorithm.

The first part of an `ATOMATCH` command specifies what segment pair the command applies to. If `ALLSEG` is specified or if no segment identifier pair is given, the command will apply to the residue pairs in all segment pairs. If a segment identifier pair is given, the command will be applied to that pair only.

The next part of the command specifies what residue pairs the command applies to. `ALLRES` or no specification of residue results in the command applying to all residues within the selected segment pairs. `RESID` means that the next two words are the residue identifiers of the residues whose atoms are to be matched. `RESNAME` means that the following two words are residue names and that the atom matching specified in the command applies to all such pairs found in the given segment pairs. For this option, the program will check for the residue name pair in either order, e.g. if we say `RESMATCH RESNAME ALA GLY ...`, it does not matter which sequence has GLY and which has ALA. On the other hand, the order in the pair does matter with the `RESID` option.

The option, `ONLY`, specifies that the default operation of matching atoms by IUPAC names should not take place. Normally, the atoms in a residu pair are matched by IUPAC names before the user matching of atoms is executed. Specifying `ONLY` will override the preliminary matching.

Finally, the pairs of words at the end of the command give the matching of atoms specified through their IUPAC names.

Once the atom pairing is complete, it is checked for duplicate references to any atom as the pairing must be an equivalence relation. Any pair which refers to an atom which is referred to by another pair is deleted. Should this occur, it is important to see what command is causing a duplication and to fix the problem. The most common source for this error is to forget to specify `ONLY` in the `ATOMATCH` command. Usually, the pair deleted from a duplicate is the pair one is interested in.

It should be noted that the matching process is performed even if just coordinates are being compared. The default action results in an atom pairing that matches every atom to itself.

Once the atom pairing is established, one can specify that the coordinates of paired atoms be matched. There are two operations involved in fitting coordinates — translation and rotation. The translation operation is as follows: The geometric center or center of gravity of the main atoms in the atom pair list and the comparison atoms in the atom pair list are calculated. The difference between the centers is added to the comparison coordinates to minimize the least squares translational differences between the coordinates. The rotation operation is as follows: The comparison coordinate set can be rotated about its center to minimize the least square differences between the two sets. The rotation can be done with respect with several different sets of atoms.

The `COORMATCH` option specifies how the coordinate matching take place. If `COORMATCH` is not specified, no attempt is made to match the coordinates. If `COORMATCH` is specified without any modifier, it results in a translation followed by a least square rotational fit of all paired atoms. If `MASS` is specified, the center of gravities are used for the translational fit; otherwise, the geometric centers are used. If `SIDE` is specified, the least square rotational fit takes place with all atoms pairs where at least one of the atoms in the pair does not have an IUPAC name that corresponds to a protein backbone. If `BACK` is specified, the least square rotational fit takes place with the complement of atoms that would be matched by `SIDE`. I.e. `BACK` will do the matching using only backbone atoms, and `SIDE` will do the match using side chain atoms. If `NOROT` is specified, no rotation takes place. Finally, the `SAVE` option will save the coordinate transformation necessary to fit the comparison coordinates onto the reference coordinates. This can be used later by the coordinate transformation commands to move atoms around according to this transformation, see Chapter 15 [Coordinate Manipulations], page 151.

18.4 Reorienting a Coordinate Trajectory

If one is interested in reorienting every set of coordinates found in a dynamics trajectory with respect to some reference structure, one can use the `DYNAMICS` option in conjunction with the `COORMATCH` options in the `COMPARE` command. Whatever options specified in the `COORMATCH` command are used to translate and/or rotate all the coordinates in the trajectory. The matching of the two PSF's determines which atom pairs are used in the fit.

The process of reorienting a coordinate trajectory works as follows: A series of files containing the trajectory are assigned to successive units prior to a CONGEN run. The coordinates stored therein are presumed to have been written every `NSAVC` steps. CONGEN will read each coordinate, select some periodically, reorient them, and write them to successive units where each output file will have a user specified number of coordinates. The following table lists the options involved:

```
Option  Default  Purpose

FIRSTU     51    The first unit of the trajectory to be read.

NUNIT      1     The number of units to be read starting with FIRSTU.

SKIP       1     Only those coordinate whose dynamics step number
                 modulo SKIP will be reoriented and written out.

OUTPUTU    61    The first unit number of the output trajectory.

NFILE            The number of coordinates written to each output file.
                 If left out, this will be set to the number of
                 coordinates in the first input file.
```

The title of the output trajectory will be copied from the input trajectory.

If atoms were fixed during the dynamics, see Section 11.4 [Fixed Atoms], page 97, the new trajectory produced will not have fixed atoms because the rotations applied to each coordinate set will be different thereby yielding different coordinates for the fixed atoms. Fixing the coordinates leads to a large space reductions, so the reorientation process will therefore result in potentially much larger trajectory files.

When the process is complete, the comparison coordinates will be the last set of coordinates written to the output trajectory files. This is true even if '`COOR` *`unit`*' is specified. Also, if the `SAVE` option is used, then the saved transformation will be the one used for the last set of coordinates.

18.5 Comparison Messages

As the `COMPARE` command processor proceeds, it will output messages on its progress. If you make a mistake, the messages may give you some idea of what you did wrong. In interpreting any error messages you receive, keep in mind the fact that as each part of the command is interpreted, it is removed from the command line. If you make a typo so something doesn't get deleted, it may be interpreted as something else. For example, if one were to specify,

```
COMPARE SEGMATCH A B $ -
        RESMATCH A B RANGES 1 100 10 110 $$
```

the error messages would say that `RANGES` is too long to be a residue identifier and that the residue id's are not all paired. The error in the command is that the `RANGES` command is not terminated by a single delimiter. This results in `RANGES` being left in, and it and the rest of the command being interpreted as resid pairs. If this seems hopelessly obscure, it unfortunately is.

If one desires a detailed print out of the coordinate matching and atom matching, specify the option, `VERBOSE`. Your printout will weigh pound instead of ounces.

19 Dynamics Analysis

The analysis facility provides the means for analyzing the trajectory of a dynamics calculation. Since there is a wealth of information in these runs, one must be able to look at the information at various levels of detail. Two methods are provided for such analysis: one can look at numerical measures such as the average or various moments of the time distribution of all internal coordinates or atoms at once or one can calculate correlation functions on a few atoms or internal coordinates. In this chapter we shall discuss the methods used to look at all the properties simultaneously, and in the next chapter, we shall look at correlation functions, see Chapter 20 [Correlation Functions], page 189.

To analyze the dynamical properties of the system en masse, the table, see Chapter 17 [Tables], page 159, is used. This allows one to use the table manipulation commands to massage the information in the most appropriate way.

In order to perform the analysis successfully, one must understand the flow of data. The coordinates and/or velocities output during the dynamics runs are summed in various ways to provide the basis for calculating various moments of the time distribution. This information is put into one of several averaging data structures which are described below. One can read or write these averages using the analysis facility's I/O commands, see Chapter 21 [I/O in Analysis], page 197. One can combine two averaging data structures to get long time averages, and one can add more points from other trajectories to an existing average. The flow of data is diagrammed below:

```
  Dynamics Trajectory
           |
           v
   Accumulation Command
           |
           v
Averaging Data Structures<----+
           |                  |
           |         Combination Command
           |                  |
           +------------------+
           |
           v
     Build Command
           |
           v
         Table
```

19.1 Averaging Data Structures

There are three types of averaging data structures – one for all the internal coordinates, one for the positional time distributions for all the atoms, and one for velocity time distributions for all the atoms.

For internal coordinates, there is one averaging data structure called `ICAV` (Internal Coordinate AVerages). It contains the minimum and maximum values as well as measures of the first four moments of the time distribution for the geometries and energies for the bonds, bond angles, torsion angles, and improper torsion angles in the structure.

The measures of the first four moments is as follows: For the first moment, the average is used.

The second moment is $\sqrt{<(X-<X>)^2>}$.

The measure of the third moment is $\sqrt[3]{<(X-<X>)^3>}$.

The measure of the fourth moment is $\sqrt[4]{<(X-<X>)^4>}$.

For averaging atomic positions and velocities, a complication arises because each position or velocity is specified by three coordinates which do not change independently of each other. In computing second and higher order moments, we must maintain all the cross correlations. These cross correlations form a tensor whose order is the same as the moment. Fortunately, the second moment tensor can be diagonalized so that the cross correlations are zero. In other words, the fluctuations can be resolved into independent components. The transformed second moment defines a set of principal axes. The principal axis can then be used to define a rotation which can be applied to the third and fourth moments as well.

The most important aspect of this transformation from the data management point of view is that it is not easily inverted; one cannot conveniently go from the principal axis set of moments to the cartesian axis set. Therefore, the averaging data structures for positions and velocities of atoms consist of two data structure each. The ones for the coordinates are `AVX` (AVerage X) and `PAX` (Principal Axis X). The ones for the velocities are called `AVV` (AVerage Velocity) and `PAV` (Principal Axis Velocity).

The AV data structures are required only for accumulating averages for an dynamics runs. The `BUILD` command only requires the `PA` data structures. The analysis facility automatically constructs a new `PA` data structure from an `AV` data structure whenever necessary.

If it is necessary to rotate all the coordinates in a trajectory with respect to a particular coordinate set, the `COMPARE` command may be used. See Section 18.4 [Reorienting a Coordinate Trajectory], page 178, for details. If it is necessary to merge several trajectories together, the `MERGE DYN` command in the main part of CONGEN can be used, see Section 7.8 [Manipulating Trajectories], page 67.

19.2 Accumulation Command — `ACCUM`

19.2.1 Syntax

```
                        { IC }
    ACCUM [COMPARE] [ADD] {COOR} FIRSTU unit-number [NUNIT integer]
                        {VEL }

            [BEGIN integer] [STOP integer] [SKIP integer]
```

19.2.2 Function

The function of the `ACCUM` command is to accumulate averaging information for the internal coordinates, positions, or velocities. The `ACCUM` command can add to existing accumulations or initiate a new set of accumulations. It will also perform the principal axis transformation on positional or velocity moments automatically once the accumulation is complete.

19.2.3 Files Required

The data used by `ACCUM` is produced during the dynamics calculations. Two sets of data are produced, the coordinates and velocities of all the atoms at periodic intervals during the runs. One specifies a set of these files by specifying the first unit number of the first file and specifying how many unit numbers after that should also be read. These files should be arranged in time order.

19.2.4 Required Operands in the Command

To use the ACCUM command, two operands must always be specified — the first unit from which the coordinate or velocity sets will be read and which data structures the accumulations should go into. FIRSTU specifies the unit number of the first unit. IC specifies that the internal coordinate averages are to be accumulated; the files which are read must be dynamical coordinates. COOR specifies that the coordinates averages are to be accumulated; the files must dynamical coordinates. VEL specifies that velocity averages are to be accumulated; the files must be dynamical velocities. If the files do not have what they are expected to have, the analysis facility will stop execution of CONGEN. The files themselves may be read repeatedly.

19.2.5 Options

The NUNIT option specifies how many units are to be read for the accumulation. If not specified, it defaults to 1.

The COMPARE option specifies that the accumulation should be done using the comparison structure's averages. This allows comparisons between dynamical calculations. If not specified, the accumulation is done with respect to the main structure.

The ADD option allows one to add to a preexisting accumulation. If not specified, the accumulation is started from scratch. When it is specified, the data in the files is added to what is already present in the accumulation. Some important conditions must be satisfied for this to be done. First, the sampling interval for the pre-existing accumulation and the added data must be the same. The sampling interval is determined by the SKIP option (see below). Next, the range of the dynamics steps covered by the pre-existing accumulation and the files to be read must be adjacent in time. In other words, the stopping step of one set plus the sampling interval must equal the beginning step of the other set. These conditions ensure that the averages will be meaningful.

The BEGIN option specifies the first step in the files to be accumulated. All records which have a step number greater than or equal to BEGIN will be accumulated. If not specified, it defaults to 1 so that the range of steps to be accepted is not bounded from below.

The STOP option specifies the last step in the files to be accumulated. All records which have a step number less than or equal to STOP will be accumulated. If not specified, it defaults to infinity so that the range of steps to be accepted is not bounded from above.

The SKIP option indirectly specifies the sampling interval. As the coordinates or velocities are read in, the step number modulo SKIP is computed. If this number is zero, the step is used in the accumulation. If the step number modulo SKIP is not zero, the step is ignored. If not specified, SKIP defaults to 1. One must careful when using this option. If coordinates were written every 5 steps of dynamics and one specifies SKIP to be 7, the sampling interval will be 35 steps.

Reorienting a coordinate trajectory is possible using the COMPARE command. For details see Section 18.4 [Reorienting a Coordinate Trajectory], page 178.

19.3 COMBINE Command — Combination of Averages

19.3.1 Syntax

```
                    { IC }
COMBINE [COMPARE] {COOR} UNIT unit-number
                    {VEL }
```

19.3.2 Function

The COMBINE command is used to combine two averages together. One average is within the analysis facility, the other is read from a file. The averages on file are produced by the analysis facility and written there by the WRITE command. The result of the combination replaces the average kept within the facility.

The COMPARE option specifies that the comparison data structure should be used. Otherwise, the main data structures are used.

IC, COOR, or VEL is used to specify which of the three averaging data structures should be used. IC refers to the internal coordinate averages; COOR refers to the average coordinates; and VEL refers to the average velocities.

The UNIT operand specifies what unit the second average is coming from. The average stored in this file must be the same type as the one referred to be the command.

19.4 Dynamical Table Properties

The data contained in the averages data structures is placed into a table using the BUILD command, see Section 17.1 [Building Tables], page 159. There are a large number of dynamic properties available for table construction.

Prior to using any of the dynamic properties, one must be ensure the correct data structures have been created within the analysis facility. In order to get dynamical information about internal coordinates, an ICAV must have been built. If such information is to be compared, the ICAV's for both main and comparison data structures must have been created. For positional dynamical properties of atoms, one requires a PAX data structure. Likewise, for velocity dynamical properties, one requires a PAV data structure. Note that both positional and velocity properties of atoms may be mixed in the same table. If one fails to ensure that the correct data structures have been created, the BUILD command will print an error message and not build the table.

19.4.1 Dynamical Properties of Internal Coordinates

For the internal coordinates (bonds, angles, torsion angles, and improper torsion angles), the syntax is as follows:

```
property ::= {GEOMETRY} [DYN moment-spec]
             { ENERGY }
```

A 'moment-spec' has the following syntax:

```
                { M1  }
                { M2  }
                { M3  }
moment-spec ::= { M4  }
                { MIN }
                { MAX }
                { AVE }
                { SD  }
```

GEOMETRY and ENERGY specify whether the geometric or energetic values should be used. DYN specifies that we are dealing with dynamical properties. M1 through M4 stand for the measures of the first through fourth moments as explained earlier. AVE is synonymous with M1, and SD is synonymous with M2. MIN requests the minimum value, and MAX requests the maximum value.

19.4.2 Dynamical Properties of Atoms

A variety of dynamical properties of atoms exist. One can look at various moments of the position and velocity distribution over time. One can look at the anisotropy of the motion. One can look at temperature related properties, such as the kinetic energy and heat capacity. Finally, one can look at the orientation of the principal axes of the motion with respect to the bond vectors.

Reorienting a coordinate trajectory is possible using the `COMPARE` command. For details see Section 18.4 [Reorienting a Coordinate Trajectory], page 178.

19.4.2.1 Dynamic Properties of Position and Velocity

Syntax

```
                   { X  }
                   { Y  }
                   { Z  }
property ::= DYN { R  } moment-spec [PAX]
                   { VX }
                   { VY }
                   { VZ }
                   { VR }

property ::= DYN TEMPFACTOR
```

Function

The function of this class of properties is to extract the first four moments of either the positional or velocity time distribution for all atoms. The moment-spec specifies which moment or minimum or maximum of the distribution for each atom to extract. `X`, `Y`, and `Z` specify the x, y, and z components of the positions. `R` specifies different things for different moments. For the first moment, it is the distance between the origin and average coordinates. For the second through fourth moments, it is the second through fourth power of the moments summed together and then raised to the reciprocal of the moment order. For example, `R M4` is

$$\sqrt[4]{\mathrm{M4X}^4 + \mathrm{M4Y}^4 + \mathrm{M4Z}^4}$$

`PAX`, which may be specified only for the second through fourth moments, causes the principal axis values to be used. Note that when `PAX` is not specified, the cross moment terms are not accessible. The same rules apply for `VX`, `VY`, `VZ`, and `VR` except they apply to velocities.

The `TEMPFACTOR` property converts the `DYN R M2` property into the units used by crystallographers for isotropic temperature factors.

19.4.2.2 Properties of the Anisotropy of the Motion

Syntax

```
property ::= DYN { ANISOTROPY } { X }
                 { ANISTROPY  } { V }
```

Note: the program will accept the above incorrect speling as well as the normal spelling.

Function

This property specification extracts the anisotropy of the fluctuations. Since the principal axis transformation orders the second moment fluctuations by their size, one can measure the anisotropy of the motion using the following measure:

$$\frac{\mathrm{M2X}}{\sqrt{\frac{\mathrm{M2Y}^2+\mathrm{M2Z}^2}{2}} + 10^{-12}} - 1$$

where the second moments are in the principal axis system. For this measure, isotropic motion will yield an anisotropy of 0; motion along a single line will yield an anisotropy of 1,000,000,000,000. For large anisotropies, the measure will roughly yield the ratio of the dominant motion to the other motions. The anisotropy of `X` specifies the coordinate anisotropy; the anisotropy of `V` specifies the velocity anisotropy.

19.4.2.3 Atomic "Temperature" Properties

Syntax

```
                              { M1 }
property ::= DYN { KE } { M2 }
                  {TEMP} {AVE }
                              { SD }
```

Function

This property specifies the calculation of the average or standard deviation of the kinetic energy or temperature. This information is calculated from the velocity principal axis averages. The units for the average kinetic energy is kcal/mole. The unit for the standard deviation of the kinetic energy is cal/(deg*mole) – the normal units for heat capacity. The units of temperature moments is degrees Kelvin.

The temperature factors of the atoms are available, see Section 19.4.2.1 [Dynamical Properties of Position], page 185.

19.4.2.4 Properties of the Principal Axis of Motion

Syntax

```
                                  [ 1 ]
                            { X } [ 2 ]
property ::= DYN PAANG { Y } [ 3 ]
                            { Z } [ 4 ]
                                  [ 5 ]
                                  [ 6 ]

                        [ 1 ]
                        [ 2 ]
property ::= DYN PABOND [ 3 ]
                        [ 4 ]
                        [ 5 ]
                        [ 6 ]
```

Function

These property specifications are used for determining the angles between the principal axes of positional time distribution and the bond joined to each atom. Each atom has associated with it space for 6 times 3 sets of angles. Six is required because each atom may have up to six bonds to neighboring atoms. Three is required for the angle between each principal axis and a bond vector. The six sets of three angles are sorted in ascending order on the magnitude of the angle between the x principal axis and the bond vector. Therefore, to obtain the principal axis bond vector angles, one uses the `PAANG` property specification. In this specification, `1` specifies the angle between the bond whose direction is closest to the first principal axis. `2` specifies the next closest angle, etc. If the number is omitted, `1` is assumed. `X`, `Y`, or `Z` specify which principal axis angle should be included.

To allow one to see how the bond vectors have been sorted, the `PABOND` property specification may be used. `PABOND 1` gives the atom number of the first bond in `PAANG` collection, `PABOND 2` gives the next etc. If the number is omitted, 1 is used. The static atom property `NUMBER` may be used to assist in the identification of the bonded atoms.

19.5 Comparing Dynamics Results

To compare the results from two dynamical calculations, one must perform the following steps: First, one must invoke a `COMPARE` command, see Chapter 18 [Comparisons], page 173, to establish the comparison PSF and other comparison data structures and to construct the matching of atoms between the two PSF's. If the PSF is not changed, then any averaging data structures will be assigned from the main calculation to the comparison calculation. Then, any additional comparison averaging data structures must either read in or constructed for both main and comparison. The `COMPARE` option in the `READ`, `ACCUM`, and `COMBINE` command can be used to direct the averaging data structures to the comparison data structure collection. Finally, the `BUILD` command using the `DIFF` option, see Section 17.1 [Building Tables], page 159, can be used to build tables which are the differences between the main and comparison dynamical averages.

20 Correlation Functions

The `CORREL` command is used to obtain a time series for a given property (‘`correlation-spec`’) and then manipulate it as required (‘`manip-spec`’) to plot the time series , correlation function, spectral density, etc. and determine the correlation times.

Reorienting a coordinate trajectory is possible using the `COMPARE` command. For details, see Section 18.4 [Reorienting a Coordinate Trajectory], page 178.

N.B.: The documentation for correlation functions is not in the best of shape. Until there is an opportunity to go through all of the code and associated documentation in order to verify its accuracy, please use this command with especial care. *Don't* put all your faith in what you are about to read.

20.1 Overall Syntax of the `CORREL` Command

```
CORREL [COMPARE] FIRSTU unit [NUNIT integer] [BEGIN integer ]

       [STOP integer] [SKIP integer] [ ENERGY ]  [VELOCITY]
                                     [GEOMETRY]

       [loop-spec] [FRUNIT unit]

       correlation-spec deldel

       repeat(manip-spec del)
```

20.2 Specification of the Trajectory Files

The `CORREL` command reads a number of trajectory files whose Fortran unit numbers are specified sequentially. The first unit is given by the `FIRSTU` keyword and must be specified. `NUNIT` gives the number of units to be scanned, and defaults to 1.

`BEGIN`, `STOP`, and `SKIP` are used to specify which steps in the trajectory are actually used. `BEGIN` specifies the first step number to be used. `STOP` specifies the last. `SKIP` is used to select steps periodically as follows: only those steps whose step number is evenly divisible by `STEP` are selected. The default value for `BEGIN` is the first step in the trajectory; for `STOP`, it is the last step in the trajectory; and for `SKIP`, the default is 1.

Reorienting a coordinate trajectory is possible using the `COMPARE` command. For details, Section 18.4 [Reorienting a Coordinate Trajectory], page 178.

20.3 Options for the `CORREL` command.

The `VELOCITY` option is specified if the correlation function is required for velocities. The program then expects the files in `NUNIT`'s from `FIRSTU` to be velocity files from a dynamics run.

The `ENERGY` or `GEOMETRY` option applies only when the `correlation-spec` refers to internal coordinates. `ENERGY` causes the energies to be used; `GEOMETRY` refers to the geometries. The default is `GEOMETRY`.

20.4 Correlations on Normal Mode

The `loop-spec` is used in the calculation of correlation functions of atomic positions projected onto the normal modes. Unfortunately, documentation on this is very skimpy, and anyone who is interested will have to dive into the code to understand what is going on.

The `loop-spec` syntax is as follows:

```
LOOP-SPEC ::= LOOP [DIAG] [BRIEF]

              repeat([integer [integer [integer [integer]]]] del) deldel
```

`BRIEF` reduces the amount of output showing what calculations will be done.

20.5 Time Series Specifications

In this section, we describe what time series can be calculated for use by the correlation functions.

20.5.1 Syntax

```
                     { ATOM     atom-correl-spec              }
                     { BOND     bond-correl-spec              }
                     { ANGLE    angle-correl-spec             }
                     { TORSION  torsion-correl-spec           }
                     { IMPROPER improper-torsion-correl-spec  }
                     { VECTOR   vector-correl-spec            }
                     { SCP/VEC          vector-correl-spec    }
correlation-spec ::= { CRS/SCP/VEC      vector-correl-spec    }
                     { SCP/ATM          atom-correl-spec      }
                     { CRS/SCP/ATM      vector-correl-spec    }
                     { FLCT/ATM         atom-correl-spec      }
                     { SCP/FLCT/ATM     atom-correl-spec      }
                     { CRS/SCP/FLCT/ATM vector-correl-spec    }
                     { RSQ/ATM          atom-correl-spec      }
                     { GYRATION         real                  }
                     { DENSITY          real                  }
                     { MODE                                   }
                     { TEMP                                   }
```

20.5.2 Function

The `correlation-spec` option is required. This option specifies the property for which the time dependence is sought. The program determines whether the correlation function requested is auto or cross correlation. When a colon appears in the middle of `correlation-spec`, the program assumes that a cross correlation has been requested between the quantities on either side of the colon. (In special cases like `SCP/ATM`, `SCP/VEC`, `SCP/FLCT/ATM`, which are auto correlation quantities, a colon may be needed .)

The options in `correlation-spec` are the same for both coordinates and velocities . Those options that are exclusive to coordinates (c) or velocities (v) are thus noted. The Time series is determined for one (auto) or two (cross) quantities desired. If more than one quantity is specified, then the average of those quantities are determined as the time series. For example,

```
ATOM A 1 CA                 ! auto correlation for first alpha carbon
ATOM A 1 CA : A 2 CA        ! cross correlation between first and second
                            ! alpha carbons.
ATOM A 1 CA : A 1 CB $ -
     A 2 CA : A 2 CB $      ! cross correlation between the average of
                            ! alpha carbons in residues 1 and 2 and the
                            ! average of beta carbons in residues 1 and 2.
```

The options are listed below with the property they represent.

`ATOM atom-correl-spec`
: Positions of atoms or velocities.

`BOND bond-correl-spec`
: Bond lengths (`GEOMETRY`) or energies (`ENERGY`).

`ANGLE angle-correl-spec`
: Bond angles or bond angle energies.

`TORSION torsion-correl-spec`
: Torsion angles or energies.

`IMPROPER torsion-correl-spec`
: Improper torsion angles or energies.

`VECTOR vector-correl-spec`
: Vectors joining pairs of atoms. (c)

`SCP/VEC vector-correl-spec`
: Auto correlation for the scalar product of pairs of vectors. If the two vectors in each pair or not identical, the colon is used to distinguish the two vectors.

`CRS/SCP/VEC vector-correl-spec`
: Cross correlation for the scalar product of two pairs of vectors. Note that even number of vectors are required on either side of the colon.

`SCP/ATM atom-correl-spec`
: Auto correlation for the scalar product of the positions of atoms. If the atom pairs are not the same then a colon is used.

`CRS/SCP/ATM vector-correl-spec`
: Cross correlation for the scalar product of the positions of atoms. Note that the atom pairs are described as vectors.

`FLCT/ATM atom-correl-spec`
: Fluctuations of atom positions.

`SCP/FLCT/ATM atom-correl-spec`
: Auto correlation for the scalar product of fluctuations of atoms. If the atom pairs are not the same then a colon is used.

`CRS/SCP/FLCT/ATM vector-correl-spec`
: Cross correlation for the scalar product of the fluctuations of atoms. Note that the atom pairs are described as vectors.

`RSQ/ATM atom-correl-spec`
: Square of the radial distance of the atom from the center. (c) Kinetic energy of the atom. (v)

`GYRATION real`
: The radius of gyration of the molecule calculated using a cut-off given by the `real` number.

`DENSITY real`
: The particle density of the molecule calculated using a cut-off given by the `real` number.

`MODE` Projects the mass-weighted coordinates into the specified normal mode.

`TEMP` The temperature of the molecule. (v)

The following are the syntax of the specifications used above.

```
atom-correl-spec ::= repeat(atom-spec [: atom-spec] del)

bond-correl-spec ::= repeat(atom-spec - atom-spec
                            [: atom-spec - atom-spec] del)

angle-correl-spec ::= repeat(atom-spec - atom-spec - atom-spec
                             [: atom-spec - atom-spec - atom-spec ] del)

torsion-correl-spec ::=
     repeat(atom-spec - atom-spec - atom-spec - atom-spec
            [: atom-spec - atom-spec - atom-spec - atom-spec ] del)

vector-correl-spec ::= repeat(atom-spec - atom-spec
                             [: atom-spec - atom-spec] del)

atom-spec ::= [segid] [resid] [iupac]
```

NOTE: The unnamed parts of an atom-spec default to those used previously in the individual correlation specification.

20.6 Manipulations of the Time Series and Correlation Functions

The `manip-spec` option is used to manipulate the time series without having to use the files again and to plot the desired data.

20.6.1 Manipulation Syntax

```
manip-spec ::= {PLOT [LP]  [PAGE [SAMPLE] ] [A]      }
               {     [TEX] [     [FIRST ] ] [B]      }
               {           [FULL        ]            }
               {                                     }
               {        [ DELTA          ]           }
               {        [ NORMAL         ]           }
               {MANTIME [ SQUARE         ]           }
               {        [ COS            ]           }
               {        [ COS2           ]           }
               {        [ AVERAGE integer ]          }
               {        [ SQRT           ]           }
               {        [ FLUCT          ]           }
               {        [ DELTAI         ]           }
               {        [ DELTAN integer  ]          }
               {        [ OSC            ]           }
               {        [ SAVE integer   ]           }
               {        [ ADD integer    ]           }
               {        [ PROB integer   ]           }
               {        [ LOG            ]           }
               {        [ MULT real      ]           }
               {        [ DELTAMIN       ]           }
               {        [ ABS            ]           }
               {        [ NORM           ]           }
               {                                     }
               {                        [P0]         }
               {CORFUN [ FFT  ] [LTC ] [P1] [NONORM] }
               {       [DIRECT] [NLTC] [P2]          }
               {                                     }
               {       [    INTEGRATE     ]          }
               {MANCOR [ INTEGRATE/SQUARE ]          }
               {       [       LOG        ]          }
               {       [ SPECTRAL/DENSITY ] [NRESET] }
               {       [ SQUARE           ]          }
               {       [ SAVE integer     ]          }
               {       [ ADD integer      ]          }
               {       [ DIV integer      ]          }
               {       [ NORM             ]          }
               {                                     }
               {RESET                                }
```

20.6.2 Making Plots of the Time Series or Correlation Functions

This `PLOT` option plots the time-series, correlation function or spectral density. There is no specific option to request any of these. But the program prompted by this command always plots the result of the last manipulation operation other than `PLOT` itself. As the first manipulation command it will plot the time series.

`LP`	`LP` is used to get the lineprinter plot.
`TEX`	`TEX` was once used to get the plot on a Tektronix 4662 printer. This code no longer works.
`PAGE`	`PAGE` is used for lineprinter plots to stay within a page.

`FULL` `FULL` is used for lineprinter plots with a line per time interval.

`SAMPLE` `SAMPLE` allows the whole plot to be plotted in a page by sampling.

`FIRST` `FIRST` is to plot all points in the initial section of the plot in a page.

The defaults are `LP`, `PAGE`, `SAMPLE`.

20.6.3 Manipulating the Time Series

The `MANTIME` option takes the time series that is active and performs the operation requested by the option and leaves the resultant time series as the active time series. This helps in performing various permutations of manipulations to increase the options without increasing the number of option commands.

`DELTA` Subtracts the average of the time series from all elements.

`NORMAL` Normalizes the vectorial time series. (i.e. creates the unit vector).

`SQUARE` Squares all the elements.

`COS` Obtains the cosine of all elements.

`COS2` Calculates 3*cos**2 - 1 for all elements.

`AVERAGE integer`
: Calculates the average for every n consecutive points and increases the time interval by a factor of n where n is the specified `integer`.

`SQRT` Obtains square root for all elements.

`DELTAI` Subtracts the value of the first element from all elements.

`DELTAN integer`
: Subtracts the value of the nth element from all where n is the specified `integer`. Q(I) = Q(I) - <Q(I)>, I from 1 to N, from N+1 to N+N etc. (Check code on this)

`OSC` Counts the number of oscillations in Q(T).

`SAVE unit` Save the time series into the given Fortran I/O unit.

`ADD unit` Add the time series read in the file specified by '`unit`' to the existing time series.

`PROB integer`
: Give the probability to find a specific value of the time series. The `integer` specifies the number of subdivisions of the time series are considered.

`LOG` Q(T) = LOG(Q(T))

`MULT` Q(T) = `real` * Q(T)

`DELTAMIN` Q(T) = Q(T) - QMIN. QMIN being the minimum of the time series.

`ABS` Q(T) = ABS(Q(T))

`NORM` Computes the norms of the vectors in the time series. The new time series contains these norms (`VECCOD` is set to 1).

20.6.4 Calculating a Correlation Function

The `CORFUN` option takes the active time series and calculates the desired correlation function from it.

`FFT` This option is to calculate the correlation function using the Fast Fourier Transform method.

`DIRECT` This option is to calculate the correlation function using the direct multiplication method.

`P0` This option gives the direct correlation function.

`P1` This is to obtain the correlation function of first order Legendre Polynomial.

`P2` This is to obtain the correlation function of second order Legendre Polynomial.

`LTC` Long tail correction (subtracting the value of last element of the correlation function.)

`NLTC` No long tail correction.

`NONORM` Correlations are not normalized. This is useful for adding correlations computed in different trajectories.

The defaults are `FFT`, `P0`, `LTC`.

20.6.5 Manipulating the Correlation Functions

The `MANCOR` option takes the correlation function output from the `CORFUN` operation and performs the following operations.

`INTEGRATE`
Integrates the correlation function along time.

`INTEGRATE/SQUARE`
Integrates the square of the correlation function.

`LOG` Takes the logarithm of the correlation function to see any apparent exponential decay.

`SPECTRAL/DENSITY`
Fourier transforms the correlation function to obtain the spectral density.

`NRESET` When specified the latest series determined in `MANCOR` is considered, there is no-resetting to the original time series determined by `CORFUN`. This allows to do multiple treatments with `MANCOR`.

`SAVE unit` The correlation time series is saved.

`ADD unit` The correlation time series read in file '`unit`'. is added to the actual correlation time series.

`DIV integer`
The correlation time series are divided by the specified `integer`.

`NORM` Correlations are normalized. Note that `CORFUN` is equivalent to `CORFUN NONORM` followed by `MANCOR NORM`.

20.6.6 Resetting Back to the Original Time Series

This `RESET` option resets the active time series to be the original time series.

21 Reading and Writing Analysis Data

The analysis facility provides the ability to read and write data structures which are generated within it. This allows one to save the data structures which take a long time to calculate (such as the averaging data structures) or to get coordinates which are produced in the facility. In addition, the properties of tables may be read and written, see Section 17.1.6 [Table I/O], page 163.

21.1 The Analysis READ command

The analysis read command is used to read dynamical average data structures into CONGEN.

21.1.1 Syntax

```
                  {ICAV}
                  {AVX }
READ [COMPARE] {AVV } UNIT unit-number
                  {PAX }
                  {PAV }
```

21.1.2 Function

The `READ` command is used for reading data structures from a file into the analysis facility. The only data structures which can be read are the averaging data structures which are produced using the `ACCUM` and `COMBINE` commands, see Chapter 19 [Dynamics Analysis], page 181. `ICAV` is the Internal Coordinate AVerages. `AVX` is the Average X (coordinates). `AVV` is the AV Velocities. `PAX` is the Principal Axis X, and `PAV` is the Principal Axis Velocities. The `COMPARE` option specifies that the data structures are to be read in the comparison structures; if not specified, the main structures are used. The `UNIT` operand must be specified and gives the unit number where the data structure should be read.

The format of the files which are written is very simple. All data is in binary. The first few records are consistent with the format used in the rest of CONGEN – a header, control array, and title array. What follows is the data structure with its own control information. For specific details, see the subroutines, `READDT` and `WRITDT`.

21.2 The Analysis WRITE Command

The analysis `WRITE` command is used to write averaging data structures as well as coordinates from the analysis facility. Coordinate writing is especially useful after a COMPARE command has been performed, see Chapter 18 [Comparisons], page 173.

21.2.1 Syntax

```
                 {  ICAV }
                 {  AVX  }
                 {  AVV  }
WRITE [COMPARE] {  PAX  } TITLE string del UNIT unit-number
                 {  PAV  }
                 {AVECOOR}
                 {  COOR }
```

21.2.2 Function

The `WRITE` command is used for writing data structures to files. The dynamics averages data structures may be specified as with the `READ` command. Also, one specify either the dynamical average coordinates which are produced from manipulating the average X (`AVX`) averaging data structures or one may specify the coordinates. For the main structure, the coordinates will be identical to those read in the main program, but with comparison coordinates, one gets coordinates which may have been translated and rotated to fit the main coordinates in the last `COMPARE` command.

The `COMPARE` option specifies that the comparison data structures should be used. If left out, the main structure is used. In reference to the data structure being written, the first five operands (`ICAV`, `AVX`, `AVV`, `PAX`, and `PAV`) are the same as in the `READ` command. `AVECOOR` specifies the average coordinates as collected in the average X data structures, specifically `PAX`. `COOR` is the coordinates. The `TITLE` operand specifies a title to be written on the file. The `UNIT` operand specifies what Fortran unit the file should be written to. Both `TITLE` and `UNIT` are obligatory; if not specified, no file will be written.

The title specified in this command undergoes some processing before it is put into the file. Since the CONGEN format for files requires a title of 80 character records, the title string is broken at spaces (if possible) to keep each line within 80 characters. Since the command processor in the analysis facility ignores the existence of records, one needs a special way to break a line at a point other than at 80 characters. One may specify three slashes, `///`, to indicate a break. No spaces are inserted on either side of the three slashes when the title is constructed.

22 Selection and Plotting Commands

The generation of plots in the analysis facility is done using a data structure known as a selection. A selection consists of a list of data points associated with a position number, residue number, and a residue name (more detail is available in the description of the `SELECT` command below and the introduction, see Section 16.1 [Introduction to Analysis], page 157). Data for a selection comes from the table, see Chapter 17 [Tables], page 159. All plots are generated from the selection. Two such data structures are provided allowing scatter plots to be produced as well.

22.1 `SELECT` Command — Select Data from the Table

22.1.1 Syntax

```
SELECT [ADD] [TWO] {      ALLTAG            }
                   {TAGS repeat(tag-pat) del}

       [PROPERTY property del]

tag-pat ::= string of non-blank characters
```

22.1.2 Function

The `SELECT` command takes data from the table and places it in a selection. Data is specified by its tag and its property. Only one property may be specified, and one must specify a property if the table has more than one. Either all the entries in the can be selected by specifying `ALLTAG` or only particular data using the `TAGS` option to give the tags. The tags are specified using string patterns with wildcards permitted as in the atom or cell selection syntax, see Section 11.7 [Atom Selection], page 108. If `TWO` is placed in the command, then the data goes into the second selection; otherwise, it goes into the first selection. If `ADD` is specified, the data is added to the data already present in the selection. Otherwise, the selection is cleared first before data is added to it.

In each selection, every data point has the following associated information: a position number, a residue number, and a residue name. The position number is obtained by counting data points as they are scanned in the table and setting the position number to the current count. By printing a table in tag value format, one can see the order in which the data is scanned. The residue number is generated in the same way except the count is kept on residues. The residue name is the name of the residue where the data point was taken. Any comparison residue name is ignored. For standard tables to which no deletions have been applied, the residue numbers will be reasonable in that they will be close to the residues position in the sequence. However, in a difference table where not all the atoms matched, the number will not be meaningful by themselves. Likewise, in a table where deletion of residues has taken place, the residue numbers will bear no resemblance to the residue identifiers.

22.2 `HISTO` Command — Print a Histogram

22.2.1 Syntax

```
HISTO [TWO] [RANGE range] [TITLE string del]
```

Syntactic-ordering: `TWO`, if specified, must come first.

22.2.2 Function

The `HISTO` command will make a printable histogram from the numbers in a selection. `TWO` specifies that the second selection should be used; it absence indicates that the first selection should be used.

If RANGE is not specified, the program will use the minimum and maximum values in the selection as the extents of the histogram, and will use as many lines as will fit on one printed page. If RANGE is specified, the first two numbers give the minimum and maximum extent of the histogram, and the third number gives the number of lines in the histogram. If the third number is less than 2, then the number of lines will be set to fill the page completely. Finally, the TITLE option allows one to specify a title which will appear at the top of the histogram.

22.3 PLOT Command — Plot Data Against Position or Residue

22.3.1 Syntax

```
             [RESIDUE]
PLOT [TWO]   [ CODE  ]  [UNIT unit-number]  [TITLE string del]
             [NUMBER ]

        [VERTICAL range]  [HORIZ range]
```

Syntactic ordering: The first two options must come first.

22.3.2 Function

The PLOT command allows one to plot data in a selection against residue number or position number. The plot can go to the printer or an output file may be created which is suitable for the plotting program, PLT2, invoke `man plt2` for more information. The residue number is plotted on the y axis (vertically) and data point in plotted on the x axis (horizontally).

The specification of TWO in the command directs that the second selection's data be used; otherwise, the first selection is used. Omitting the next option or specifying RESIDUE causes a plot by residue number to be used. Specifying NUMBER directs that a plot by position number be set up. Specifying CODE says that a plot by residue number plot should be output and that the plot symbols be the one letter amino acid codes for the residues. D amino acids are plotted using lower case letters. The specification of the UNIT option causes the plot data to be output on the specified unit; otherwise, the output goes on the printer. Note that CODE is meaningless when UNIT is specified since the code are not output with the numbers.

The VERTICAL option gives the minimum and maximum labels and number of lines for the vertical axis. If not specified, the range is set up so that each data point gets a line and the labels are set up in reverse order (smaller number on top). If the number of lines is specified as zero, then CONGEN will set it to the maximum number that will fit the page, see Section 25.1 [Analysis Set Command], page 209, for a description of the PAGESZ variable.The HORIZ option gives the minimum and maximum values for the horizontal label as well as the number of columns to be used. The number of columns may be adjusted down to make the plot fit in the available line size (see Section 25.1 [Analysis Set Command], page 209, to change this number.) If the number of columns is specified as a zero, then the number of columns will be set to maximally fill the page. The TITLE option allows one to specify a title to appear at the top of the plot.

22.4 2DPLOT Command — Make a Scatter Plot

22.4.1 Syntax

```
       [RESIDUE]
2DPLOT [ CODE  ]  [UNIT unit-number]  [TITLE string del]
       [NUMBER ]

       [VERTICAL range]  [HORIZ range]
```

Syntactic ordering: The first option must appear first in the command.

22.4.2 Function

This command is used for plotting the second selection as a function of the first selection. Each selection has residue numbers and position numbers associated with each point. These numbers can serve as identifiers for the data points making it possible to relate the data points in two selections and produce matched pairs of data points. Plots of these pairs can be used to demonstrate correlations or patterns between the two selections (for example phi-psi or Ramachandran plots).

The first option in the command determines whether matching will be done by residue number or sequence number. If all of the data in both selections is uniquely identified by the specified identifier then a rapid algorithm is used to match the data. If the data is not uniquely identified then the match will use the order of occurrence in selections to resolve ambiguities. For example if the first selection was generated by selecting all dihedral angles, then the selection will be unique by sequence number but not unique by residue. If the second selection also contained all of the dihedrals from a second structure, it would not be unique by residue either. If `NUMBER` is specified then the two lists, then a unique mapping of one selection onto the other exists (although if the sequences of the two structures were different it may not be the desired mapping). Conversely, `RESIDUE` will result in multiple possible matchings of which one must be chosen to produce a plot.

The routine resolves this ambiguous mapping situation by matching the first available data for a residue (or number) in selection one with and first available data for the same residue (or number) in the second selection. In the example above `RESIDUE` mapping will align the first dihedrals of each residue etc. If a sequence mismatch exists between the structures and the mismatched residues contain different numbers of dihedrals, then for that residue pair the first dihedral will be matched with the first and so forth until all of the dihedrals in one residue are matched. Note that the extra data will now be ignored rather than offsetting all subsequent matches. `NUMBER` mapping would align the first dihedral in each structure etc., but would misalign all of the data past the mismatch.

`CODE` results in residue matching and the use of one letter amino acid codes for plot symbols. The default for the first option is `RESIDUE`.

The x (horizontal) axis is used for numbers from the first selection; the y (vertical) axis is used for number from the second selection.

If `UNIT` is specified, the pairs are output to the unit and no printer plot is made. Otherwise, the plot goes on the printer.

`TITLE` gives a title that appears at the top of plot.

`VERTICAL` and `HORIZ` can be used to specify the vertical and horizontal ranges and sizes of the plot. If unspecified, the limits will be set to the minimum and maximum in the data and the number of columns will set as large as will fit on a line. If specified, the first two numbers give the limits, and the third number gives the number of rows or columns to be used in the plot. The number of rows or columns may be reduced to get the plot to fit on the page. If either the number of rows or columns is specified as zero, then they will set to fill the page maximally. See Section 25.1 [Analysis Set Command], page 209, for a description of the `PAGESZ` and `LINESZ` variables which control this.

A rudimentary statistical analysis of the data in the selections is included with the plot.

23 Non-interactive Molecular Line Drawings

The DRAW command allows a user to draw a molecule under the control of either one of two plotting programs. The command will produce files which can then be read by the MOLD or PLT2 plotting programs, which actually prepare the drawing. CONGEN also has a command for making space filling drawings of a molecule, see Section 26.1 [Sphere Drawing], page 211.

The program, PLT2, is provided as a support program, see Chapter 29 [Support], page 249. MOLD is an obsolete program, and is only available on backup tapes from Harvard.

When using PLT2, in addition to plotting the molecule, one can also plot atomic properties in a table on the molecule as well. One can either represent scalar quantities by varying the size of the circle drawn for each atom, or one can represent vector quantities by drawing vectors attached to each atom.

When using either plotting program, it is also possible to draw a portion of the molecule or only selected atoms. The commands for deleting data from the table are used to delete atoms from an atom table. This table is then used to determine which atoms should be plotted. When atoms are deleted, one can specify an option to this command causing it to relink all atoms which previous had a chain of bonds connecting them.

23.1 Syntax of the DRAW Command

```
DRAW [COMPARE] [CONNECT] [TABLE]

         [MOLD ATOMU unit BONDU unit                              ]
         [                                                        ]
         [PLT2 [SCALAR prop del                    ]              ]
         [     [VECTOR [CENTER] prop del prop del prop del del]   ]
         [                                                        ]
         [     [RADII [IUPAC] repeat({atom-type} real) del]       ]
         [                         {   ALL   }                    ]
         [                                                        ]
         [     [LABEL [IUPAC word] [FREQ int] [HEIGHT real] NAME del] ]
         [     [GLASS]                                            ]
         [     [NOFR]                                             ]
         [     [SCALE real]                                       ]
         [     [HBOND [DASH int]                                  ]
         [     [SIMDASH int]                                      ]
         [     UNIT unit]                                         ]
```

23.2 Options Independent of Plotting Program

The COMPARE option specifies that the comparison data structures should be used for the drawing. Leaving this option causes the main structure to be used.

The TABLE option specifies that the table should be consulted to see which atoms should be plotted. In the absence of references to the table via the SCALAR or VECTOR options, the entire molecule will be plotted. However, when this option is present or when the table is referenced, the table must be an atom table, and only those atoms which are in the table will be plotted.

The CONNECT option causes the command to reconnect bonds which link atoms that are not be drawn, i.e. deleted atoms. Normally, if an atom is not drawn, any bonds to that atom are not drawn either. However, this is not always satisfactory. For example, if one wants an alpha carbon plot, one wants all the alpha carbons drawn with bonds provided they were in the same chain.

The algorithm used by the connect algorithm is simple, but it can yield strange results in certain circumstances. If a deleted atom has only one bond going to it, the bond is deleted. If it has two bonds going to it, the two atoms are bonded to each other except if they are already bonded to each other. If there are multiple bonds, then all the atoms linked to the deleted atom are rebound into a cycle. One example where this algorithm doesn't work best is if only the gamma carbon of phenylalanine is deleted. In this case, a triangle will be drawn connecting C-beta, C-delta-1, and C-delta-2.

23.3 Operands for MOLD Drawings

To generate MOLD files, one must specify MOLD and the ATOMU and BONDU operands. These give the unit numbers where the atomic coordinates and bonds are written, respectively.

23.4 Operands for PLT2 output.

To generate a plotting command file for PLT2, one specifies the keyword, PLT2. One must also specify the unit where the graphics command file should go using the UNIT operand.

The file which is produced for PLT2 gives commands necessary to draw the molecule. They are ordered into portions so that editing the various parts of the plot is simplified. A CM command with a comment described the portion precedes each portion. At the end of the file is a FR 0 command which empties any plotter buffers and pauses. This FR command can be omitted by the presence of the NOFR command. Each of the commands is written with a format of (1X,A2,2X,*n*F10.3). With this scheme it is easy to write to postprocessors that will further manipulate the drawing.

To specify that the size of the atoms be proportional to a single property in the table, the SCALAR option is used to specify that property. The size of the circle in Angstroms will be equal to the scale factor specified by the SCALE option times the value of property.

To specify that a vector corresponding to three property values be drawn attached to each atom, the VECTOR option is used. The three properties specified become the X, Y, and Z components of the vector. The units are presumed to be Angstroms. The scale factor specified by the SCALE option is multiplied by the components to get the final length.

In order to control the size of atoms when the SCALAR option is not used, the RADII option is available. With the RADII option, one specifies the radius to be used for an atom with a particular IUPAC name or a particular atom type name. One can also specify the default size to be used. The default size of an atom is normally zero. The IUPAC suboption specifies that IUPAC names are being given, otherwise, parameter type names are being given.

Two labeling possibilities are available. Both of these options are invoked using the LABEL option. The first possibility, which is the default, labels residues by residue number within the segment. The IUPAC suboption specifies the IUPAC name of the atom near where the label will be drawn. If such an atom is not present in the residue or if this option is omitted, the first atom will be used. If this selected atom has been deleted, then the next present atom will be labelled. If no such atom can be found, no label will be drawn. The second possibility is specified using the operand, NAME. Here, all atoms which match the IUPAC suboption are labeled by their IUPAC name. The IUPAC option may have wildcards (see Section 11.7 [Atom Selection], page 108) so that multiple atoms within a residue can be labeled. The frequency of labels is given by the FREQ option. When the residue number in the segment can be evenly divided by FREQ, an attempt will be made to place a label. The default value for FREQ is 10. The height of the label in Angstroms is specified using the HEIGHT option. Its default value is 0.5 Angstroms.

The `GLASS` option specifies that any vector, either bond vector or property vector, start at a distance of the drawn radius of the atom, i.e. as if the atom were a glass sphere. This generates a slightly more realistic drawing. If the vector is totally contained within atoms, it is not drawn.

The `SIMDASH` option simulates drawing of dashed lines. Dashed lines are drawn by breaking up the three dimensional bond vectors into smaller pieces. In so doing, stereo plots of the dashed lines look OK. The argument specifies the number of segments used for the lines.

The `HBOND` option causes hydrogen bonds to be drawn as dashed lines. The bond reconnection is NOT done with these bonds, so if an atom in a hydrogen bond is deleted, it isn't drawn. The number of segments in the dashed lines is given by the `DASH` suboption. This currently defaults to 3.

24 Close Contact Searches

The analysis facility in CONGEN can search the system for close contacts. One can search for contacts to particular atoms, residues, or segments, as well to points in space. In addition, the energy of individual atom-atom interactions can be displayed.

24.1 Close Contact Command Syntax

```
SEARCH [COMPARE] [ATOM segid resid iupac del]
                 [POINT real real real del  ]
                 [INSIDRES segid resid del  ]
                 [INSIDSEG segid del        ]
                 [OUTOFRES segid resid del  ]
                 [OUTOFSEG segid del        ]

         [     {DISTANCE}]
         [SORT {  ELEC  }]  [INCLUDE [ELEC] [EVDW] [ENB] del]
         [     {  EVDW  }]
         [     {  ENB   }]

         [CUT real] [NSEP integer] [LIMIT integer] [FRAC integer]
```

Syntactic ordering: COMPARE must appear first.

24.2 Search Command Function

This command can be performed three different close contact searches — search over all pairwise close contacts, search of close contacts to a particular atom, and searches close to a spatial point. Searches can be limited to pairs in which both atoms are in a segment or residue using INSIDSEG and INSIDRES, or to pairs in which one atom is in the segment or residue and the other is outside using OUTOFSEG and OUTOFRES. The output of a search gives distance information, and can include energies. The output can be sorted on distance or on one of the energy contributions.

The first word allows one to search the comparison structure - if not specified, we default to the main structure. If the ATOM option is specified, the search takes place with respect to the given atom. The atom is specified by segment identifier, residue identifier, and iupac name. If the POINT option is specified, the search takes place with respect to that spatial point. INSIDRES and INSIDSEG limit the search to pairs in which both atoms are in the residue or segment specified. OUTOFRES and OUTOFSEG limit the search to pairs in which one atom is in the specified residue or segment and the other is outside it. Only one type option can be specified for a search. If no option is specified, the program assumes a search over all atom pairs.

The CUT option specifies what distance is considered “close”. It defaults to 3.5 Angstroms if not specified. NSEP specifies that the residue number of any two atoms being checked must differ by at least NSEP for the pair to be considered. This is useful for omitting contacts for adjacent residues. NSEP defaults to 0. It is not used for searches within a spatial point. LIMIT gives the number of close contacts to be saved. If a search results in more that this list, the most distant contact is removed to make room for the new pair. The default for this is 1000.

The INCLUDE option allows one to add extra information to the print out. ELEC specifies the addition of the electrostatic energy; EVDW specifies the addition of Van der Waals energy; and ENB specifies the addition of the non-bonded energy. Electrostatic contributions are calculated using the method specified in the NBOND command, see Chapter 6 [Non-bonded Interactions], page 51, (an r dependent dielectric for ATOM and RESI options and a constant specified dielectric for the

EXEL option). The INCLUDE option is not applicable to searches around a spatial point. The SORT option specifies that the output should be sorted by the given quantity.

The FRAC option controls the number of fractional digits used for printing any of the floating point quantities. It defaults to 3.

The output of this program consist of a title describing the search and columns of close contacts. The columns are packed onto a page automatically, so that the wider the page, the more columns that can be output.

25 Miscellaneous Analysis Commands

This node describes commands used for controlling the analysis facility. See Section 3.4 [Open Command], page 40, for a description of the `OPEN` command which is also accepted by the analysis facility, and see Section 27.3 [Delim Command], page 228, for a description of how to change the command delimiter. The `GEPOL` command, see Section 27.15 [Gepol Command], page 239, may also be specified in the analysis facility.

25.1 `SET` — Modify Analysis Facility Variables

25.1.1 Syntax

```
SET [INUNIT unit-number] [PRUNIT unit-number]

    [LINESZ integer] [PAGESZ integer]

    [ASURf real] [RH2O real]
```

25.1.2 Function

`INUNIT` sets the unit number of the unit where analysis commands are read from. This allows other command files to be read. This command uses the stream switching arrays available in the main part of CONGEN. The file read in must start with a title. To return back to the previous stream, the `END` command, see Section 25.3 [Analysis End Command], page 210, will suffice or hitting the end of file on the stream. Note that if you return from the stream which was current when the analysis facility starts, you will return back to the main program.

`PRUNIT` sets the unit number of the outputs of any commands which produce voluminously. `PRINT`, `HISTO`, `PLOT`, `2DPLOT`, `CORREL`, and `SEARCH` are commands in that class. Initially, `INUNIT` is 5 and `PRUNIT` is 6.

`LINESZ` gives the number of characters on a line for `PRUNIT`. Various commands have certain minimum limits on the value for `LINESZ`, and will adjust the value they use accordingly. The upper limit is what the printers can handle. `PAGESZ` gives the number of lines on a page.

`ASURF` controls the accuracy of the Lee and Richard's accessible surface algorithm. The default value is 0.05. Smaller values will improve the accuracy at the expense of speed. `RH2O` specifies the radius of the probe sphere. The default value is 1.4 A. See Section 17.1.3.3 [Atom Properties], page 160, for more information.

25.2 `TWIST` — Preparing to Compute Peptide Twist

25.2.1 Syntax

```
TWIST [COMPare]
```

25.2.2 Function

The `TWIST` command rewrites the PSF used in the analysis facility in order to create virtual dihedrals spanning the alpha and beta carbons in the backbone of a protein. For residue *i*, the dihedrals formed are CB(*i*) — CA(*i*) — CA(*i-1*) — CB(*i-1*). If a residue is a glycine, then no entry is made for it. The old list of dihedrals is destroyed (so be forewarned!), but only in the analysis facility. The PSF used by the top level of CONGEN is still intact so exiting the analysis facility and reentering it will restore the dihedrals list.

The `COMPARE` option controls which analysis PSF is used. When this keyword is omitted, then the main PSF is used; otherwise, the comparison PSF is used.

Once this command has been executed, then the twists can be obtained by a `BUILD TORSION GEOMETRY` command, and subsequent processing of the table by the analysis facility.

25.3 `END` command

25.3.1 Syntax

```
END
```

25.3.2 Function

This terminates a set of analysis commands. If the current stream is not unit 5, the next stream to read from is popped from the stream stack, see Section 3.7 [Stream Command], page 41. Otherwise the analysis facility terminates, and control reverts to the main program.
Listing of All Analysis Commands

26 Molecular Graphics

CONGEN provides three mechanisms for generating molecular graphics. The first is a simple CPK drawing facility accessed via the `SPHERE` command. This facility will generate unformatted binary files which can be displayed on an RGB raster device. The second is an auxiliary program called `peer` which provides an interactive graphics capability; see Section 29.1.6 [Peer], page 250, for a complete description. The third mechanism is the `DRAW` command in the analysis facility, see Chapter 23 [Drawing], page 203, for a description.

26.1 Sphere Drawing — CPK Style Molecular Graphics

Using a color raster display, it is possible to make solid sphere pictures of a macromolecule as stored in CONGEN. The implementation described here generates an image as an arbitrary size pixel array using up to 256 intensity values for three colors; red, green, and blue. This image is written to a file, and there are several programs for displaying these images. `diris` may be used for a Silicon Graphics Iris 4D series machine (see Section 26.1.11.1 [Diris], page 224), `DI340` may be used for an Evans and Sutherland PS340 (see Section 26.1.11.3 [DI340], page 224), and `DISPIMG` and `DISPMANY` may be used on the Deanza image processor (see Section 26.1.11.4 [DISPIMG], page 225 and Section 26.1.11.5 [DISPMANY], page 225). `DI340`, `DISPIMG`, and `DISPMANY` are written only for VAX/VMS and are specific for the computers at Massachusetts General Hospital.

To use this command, specify `SPHERE` as a single command to CONGEN. This will enter the `SPHERE` command parser which will accept the commands described in this chapter.

26.1.1 Overview of Sphere Drawing

The basic algorithm for generating the images is known the Z-buffer algorithm. The image is composed of an array of pixels, and each pixel has a height (a Z value) associated with it. Before the picture is constructed, the Z value is set to minus infinity. As pixels are set during the image generation, the height of each newly generated pixel is compared with that of the image already constructed, and if it will obscure the previous pixel (new Z is greater than that of the pixel), it is overwritten. Otherwise, it is left alone. This takes care of hidden surface removal trivially.

To generate a picture of spheres, we orthographically project each sphere in turn onto the pixel array, and determine the intensity for each such pixel. No anti-aliasing is done to reduce boundary effects.

The intensity of the pixels for the sphere is determined using Phong shading (*Communications of the Association for Computing Machinery* **18**, 311 (1975)) which is a simple shading model involving two types of reflectance, diffuse and specular. Diffuse reflectance means uniform scattering, so that the surface will scatter light the same regardless of its orientation. The only determinant is what the projected area of the surface the observer can see. Specular reflectance takes into account the reflection of light off the surface, with the intensity the viewer sees being proportional to some power of the cosine of the reflected light vector and the observer. The higher the power, the more plastic the surface appears. The observer is presumed to be at z=infinity; the light source can be anywhere; however, shadows are not computed, so some of the lighting effects can be weird especially for side lighting. All the shading parameters can be varied, see Section 26.1.6 [Sphere Lighting], page 219.

The color of the atoms is determined by what combination of primary colors; red, green, or blue; are used. The program predefines these three primary colors, three secondary colors (yellow, magenta, cyan), and orange, black, gray and white. Any number of additional colors may be defined by the user.

For efficiency, the spheres are sorted with the spheres closest to the observer being processed first. Additional sorting on X and Y is done to minimize paging into the pixel arrays.

There are several transformations that must take place in going from the atomic representation stored in CONGEN to a picture on the display device. The first transformation is from CONGEN coordinates to display device coordinates. The second transformation is from these atomic coordinates into pixels which can be stored in a image file.

The first transformation depends on the coordinates of the display device. The process of generating the image operates in a space where two of the dimensions are those of the display device. The third dimension must be scaled to match that of the screen, and so must the radii of the atoms. Therefore, the first transformation prepares a listing of the atoms along with their radii in coordinates appropriate for the display device. The color of each atom is also included. The user is free to select any atom in the PSF to be displayed. Finally, there is an 80 character title associated with the picture to identify it.

The atomic transformation in this first step is performed on the atoms selected by the user and consists of the following steps:

1. The rotation origin (`OD`) is subtracted from each component.
2. A rotation matrix is applied to each atomic position and the rotation origin is added back. The user can freely specify the rotation.
3. The components and radii are divided by the angstroms/cm scale factor.
4. The origin shift (`OC`) is added to the coordinates.
5. The components and radii are multiplied by the pixels/cm scale factor. The Y coordinate is inverted to make the origin into the bottom left of the screen.

The resulting list of atomic coordinates, radii, and colors is called the spheres list. It is expected that these lists can be concatenated to produce complex images. The spheres list can be written to a file in formatted form so that it can edited if desired. The first line of the file contains the title; the rest are atoms stored as X, Y, Z, radii, color codes, IUPAC name, residue name, residue identifier, and segment identifier with a format of `(4F10.5,I5,4(1X,A4))`.

The next transformation into an image is done by the Z-buffer algorithm described earlier. This image can be written to a file in unformatted, run-length encoded form.

One special purpose option provided is the ability to change the coloring of pixels based on their accessibility. When this option is used, the program checks to see if the surface point corresponding to a pixel is accessible to a probe sphere of given radius as in the Lee and Richards accessible surface algorithm.

It is possible to go backwards in a crude sense from a spheres list back into some PSF arrays so that new transformations are possible. The spheres lists can be read in with the coordinates therein overwriting the coordinates in CONGEN and also resetting the number of atoms and all the arrays which partition atoms in residues and residues into segments as well redefining the residue names, residue identifiers, and segment identifiers. See Section 26.1.5 [Data Flows for Drawing Spheres], page 218, and the `RS` command for details.

26.1.2 Using the Sphere Drawing Commands

The space filling image drawing facility of CONGEN runs its own command interpreter. It is entered by specifying `SPHERE` to CONGEN. The commands to this facility consist of 2 or 4 letter keywords which may be followed by various arguments. Except for the title command, (see Section 26.1.5 [Data Flows for Drawing Spheres], page 218), the commands expect their arguments free field. If any arguments are not specified, they will default to 0 unless otherwise noted.

To exit from `SPHERE`, specify `EX` or `END` as a command and you will exit.

When `SPHERE` is first called, all of its data structures are initialized. Thereafter, leaving and reentering it will not affect anything internal to it. This permits you to construct images of multiple molecules very easily. To initialize everything back to its starting state, use the `HO` command, see Section 26.1.5 [Data Flows for Drawing Spheres], page 218.

Since the data structures sphere drawing are used for the `PEER` command, see Section 26.2 [Peer Command], page 225, the initial display device is a small dummy device that does not require much pixel storage. Therefore, before making a picture, you must also specify an output device size using the `DV` command or use the various screen size and scale factor commands described in Section 26.1.4 [Transformations for Sphere Drawing], page 215.

Before invoking `SPHERE`, you must have read in a PSF, coordinate set, and parameter set; unless you read in a spheres list in which case nothing is needed. Note that if the PSF is changed by a `SPLICE` command or by an editing command, the `HOME` command should be executed, see Section 26.1.4 [Transformations for Sphere Drawing], page 215.

Pointers to all the commands in `SPHERE`.

Most commands in `SPHERE` are two characters long, the rest are four. Each menu item gives the name of the command with capital letters showing what CONGEN is looking for. The pointer to a node gives the node where information about the command can be found.

26.1.3 Atom Manipulations

Prior to the transformation of atoms into a spheres list, the user may select the atoms he wishes to display, what color they will be displayed with, and the radii to be used. By default, all atoms in the PSF are displayed. The radii are initialized to the radii given by the van der Waals parameters. The default colors for the atoms are set using the following table:

Nitrogens	Cyan
Carbons	Gray
Hydrogens	White
Oxygens	Red
Sulfur	Yellow
Others	Magenta

The commands for this section are as follows:

26.1.3.1 `ADD` — Mark Atoms for the Spheres List

Syntax: `ADd` *atom-selection*

Function: The atoms in the atom selection (see Section 11.7 [Atom Selection], page 108) are marked for inclusion into the spheres list.

26.1.3.2 `COLOR` — Color Atoms

Syntax: `COlor` *color atom-selection*

Function: The color for atoms in the atom-selection (see Section 11.7 [Atom Selection], page 108) is set to the color. Colors may be specified as either a word or an integer. The word gives the color name, and the integer gives the position of the color in the color table. Ten colors have guaranteed numbers, namely,

1	Red
2	Green
3	Yellow
4	Orange
5	Blue
6	Magenta
7	Cyan
8	Gray
9	White
10	Black

You may defined additional colors using the DC command below.

26.1.3.3 `DC` — Define Colors

Syntax: `DC` *color real real real real real real*

Function: Defines a color. The color keyword is the name of the color and must not be an integer. The six real numbers following give the range of intensities using the RGB (Red, Green, and Blue) color scheme. The first two reals give the low and high limits for the red channel, the next two give the limits on green, and the last two specify blue. All limits should be between 0 and 1.

26.1.3.4 `SFR` — Set Radius for Surface Coloring

Syntax: `SFR` *real*

Function: Specifies the radius for identifying pixels that are accessible. If this variable is positive, then all pixels which correspond to accessible points will be colored according to the `SFCO` (surface color) commands. NB: This option is very expensive to compute.

26.1.3.5 `SFCO` — Set Color for Accessible Surface

Syntax: `SFCO` *color atom-selection*

Function: Same as the `COLOR` command above except the surface colors are set.

26.1.3.6 `RA` — Set Atom Radii

Syntax: `RA` *real atom-selection*

Function: The radii of the atoms in the atom selection, see Section 11.7 [Atom Selection], page 108, are set to the real number specified.

26.1.3.7 `RC` — Reset Colors

Syntax: `RC`

Function: Reset Colors. The colors of the atoms are reset to the default scheme given above.

26.1.3.8 `RM` — Unmark (Remove) Atoms

Syntax: `RM` *atom-selection*

Function: The atoms in the atom selection (see Section 11.7 [Atom Selection], page 108) are removed from the list of atoms to be included into the spheres list. This is the inverse of the `AD` operation.

26.1.3.9 `RR` — Reset Atomic Radii

Syntax: `RR`

Function: Reset Radii. The radii of the atoms are reset to the values given in the parameter set. Be careful using this after reading in a spheres list, see Section 26.1.5 [Data Flows for Drawing Spheres], page 218, because the PSF entries pointing to the parameter list (if it exists at all) are likely to be inconsistent.

26.1.3.10 `ZMARK` — Mark Atoms by the Z Coordinate

Syntax: `ZMARk` *real*

Function: Mark atoms by their Z coordinate. Any atom whose Z coordinate plus its radius is greater than the parameter specified by this command will be removed from the list of atoms to be transformed into the spheres list. See the command, `ZCUT`, Section 26.1.4 [Transformations for Sphere Drawing], page 215, a variant of this command.

26.1.4 Transformations from the Coordinates to the Image

The commands for controlling the transformation from atomic coordinates to the spheres list are described below. They take effect for any `TR` commands executed thereafter.

26.1.4.1 `AS` — Automatic Scaling

Syntax: `AS` *real*

Function: Automatic scaling. The purpose of automatic scaling is to set transformation parameters so that the image will be centered in the Deanza screen and make full use of the screen. Both the scale factor and the origin shifts (`OC`) are set. In addition, the rotation origin (the point about which the rotation matrix is applied) is set to midpoint of the molecule. If a real number is specified, it is used to scale down the size of the image so that it will less of the screen. E.g., a scale factor of 2 will result in a image that whose maximum vertical or horizontal dimension will be less than half the screen.

26.1.4.2 `DV` — Set Device Drawing Parameters

Syntax: `DV` *string*

Function: Device setting. This command will set the number of pixels in the vertical and horizontal dimensions as well as the screen size according to the device you specify by the string. The following devices are recognized:

```
Device name   Xpixels   Ypixels  Xscreen  Yscreen

IRIS or 4D      1250       994     33.75    26.84
PS390           1024       864     32.5     27.42
DEANZA           512       512     20.0     20.00
DUMMY             25        25      1.0      1.0
```

If the device name is not recognized, then the DEANZA dimensions are used.

26.1.4.3 `HOME` — Reset Sphere Drawing Parameters

Syntax: `HOme`

Function: Initialize everything to starting state. I.e. origin shifts are set to zero, the rotation matrix is set to the identity matrix, scale factor is set to 1 Angstrom for the screen size, colors back to their default, etc.

26.1.4.4 `IR` — Input Rotation Matrix

Syntax: `IR` *real real real real real real real real real*

Function: Input rotation matrix. The elements are read in the following subscript order, 11 12 13 21 22 23 31 32 33.

26.1.4.5 `OC` — Shift Screen Origin in Centimeters

Syntax: `OC` *real real real*

Function: Origin shift in centimeters. The image is moved relative to the screen in units of centimeters by the amounts given in the command. The first number gives the horizontal shift; the second number gives the vertical shift; the third the depth shift, which will be significant if two separate images are overlaid upon each other.

26.1.4.6 `OD` — Set Origin of Data

Syntax: `OD` *real real real*

Function: Set Origin of Data, ie. origin of rotation. The rotation matrix will be applied about this origin. The origin is specified by X, Y, and Z in the space of atomic coordinates.

26.1.4.7 `OR` — Set Origin of Data in Screen Size Units

Syntax: `OR` *real real real*

Function: Set the picture origin in units of the screen size. The three arguments give the X, Y, and Z; horizontal, vertical, and depth shifts; respectively. Upon initialization, the origin is set at the bottom left of the screen. This command moves the origin so that image is shifted by the amounts specified in the command. For example, if the atomic coordinates are centered about the origin, an `OR 0.5 0.5` command will center the image about the center of the screen. An `OR 1.0` command will center the image about the lower right corner of the screen. The Z shift can be useful to control how two different molecules will overlay each other. This command works by changing the `OC` parameters described above.

26.1.4.8 `SA` — Scale in Angstrom Units

Syntax: `SA` *real*

Function: Sets the scale factor which is in units of Angstroms / cm. For example, `SA 2` will set a scale factor of 2 Angstroms/cm. The actual size of a centimeter on the screen depends on the device parameters which specify the number of pixels and screen size in each dimension. If the argument is omitted, the scale factor will be set to 1.

26.1.4.9 `SCALE` — Scale by a Multiplicative Factor

Syntax: `SCale` *real*

Function: Multiplies the current scale factor by the argument to this command. If the argument is not specified, the command has no effect.

26.1.4.10 `SHOW`, `SR` — Show Drawing Parameters

Syntax: `SHow`

Function: Show Rotation and other transformation parameters. Prints the rotation matrix, data origin (`OD`), origin shift (`OC`), scale factor (`SA`), and a checksum for the image.

Syntax: `SR`

Function: Same as `SH` above.

26.1.4.11 `SO` — Set Stereo Offset

Syntax: `SO` *real real*

Function: Stereo Offset command. The first argument specified the rotation in degrees about the Y axis to be applied. The second argument moves the image that many centimeters to the right. A typical command here would be `SO 6 6`.

26.1.4.12 `XP` — Set Horizontal (X) Pixel Count

Syntax: `XP` *int*

Function: Sets the number of pixels in the horizontal (X) dimension.

26.1.4.13 `XR` — X Rotation

Syntax: `XR` *real*

Function: Rotation about X axis. The argument specifies how degrees about the X axis the image should be rotated. The command results in the modification of the rotation matrix.

26.1.4.14 `XS` — Set Horizontal (X) Screen Size

Syntax: `XS` *real*

Function: Sets the screen size in centimeters for the horizontal (X) dimension. N.B. Be careful to avoid a difference between the number of pixels per centimeter in the horizontal and vertical dimensions unless you are trying to generate some unusual visual effects.

26.1.4.15 `YP` — Set Vertical (Y) Pixel Count

Syntax: `YP` *int*

Function: Sets the number of pixels in the vertical (Y) dimension.

26.1.4.16 `YR` — Y Rotation

Syntax: `YR` *real*

Function: Rotation about Y axis. The argument specifies how degrees about the Y axis the image should be rotated. The command results in the modification of the rotation matrix.

26.1.4.17 `YS` — Set Vertical (Y) Screen Size

Syntax: `YS` *real*

Function: Sets the screen size in centimeters for the vertical (Y) dimension. N.B. Be careful to avoid a difference between the number of pixels per centimeter in the horizontal and vertical dimensions unless you are trying to generate some unusual visual effects.

26.1.4.18 `ZCUT` — Mark Atoms by Transformed Z's

Syntax: `ZCUT` *real*

Function: `ZCUT` defines a visibility plane normal to the Z axis. Only those atoms whose upper surface is less than the parameter specified to this command will be displayed. The determination of Z is made after the molecule has been rotated. See the `ZMARK` command, Section 26.1.3 [Atom Manipulations in Sphere Drawing], page 213, for a variant.

26.1.4.19 `ZR` — Z Rotation

Syntax: `ZR` *real*

Function: Rotation about Z axis. The argument specifies how degrees about the Z axis the image should be rotated. The command results in the modification of the rotation matrix.

26.1.5 Controlling Data Flow

The commands in this section control the flow of data from the CONGEN coordinates into images.

26.1.5.1 `CS` — Clear Spheres List

Syntax: `CS`

Function: Clear Spheres list. Initializes the spheres list so a new image can be created.

26.1.5.2 `IMAGE` — Make the Sphere Drawing Image

Syntax: `IMage`

Function: Construct an image from the currently stored spheres list. This command generally takes a minute or two of CPU time.

26.1.5.3 `RS` — Read Sphere List from a File

Syntax: `RS` *integer*

Function: Read Spheres list from unit given as integer. This command reads a spheres list back in and uses the nomenclature information in the file to build a new set of nomenclature, residues, and segments in the PSF (thereby destroying the old info.) If you create a spheres list by another program containing only coordinates, radii, and colors; you'll be able to perform coordinate manipulations and image generation just fine; but you will not be able to reference atoms by name since all the identifying information will be blank (and therefore equal and indistinguishable).

26.1.5.4 `TL` — Set Title

Syntax: `TL` *specially-delimited-string*

Function: Sets the title of image. This line will be appear at the bottom of the screen. Spaces will not obscure the image so you can use multiple spaces to shift the title around your picture. The *specially-delimited-string* begins with a delimiter character which can be any character followed by a string which does not contain the delimiter followed by the delimiter character. This construction allows you to use any character in your title.

An example title command would be

```
TL ;          KOL;
```

26.1.5.5 `TRANSFORM` — Transform Atoms to Spheres

Syntax: `TRansform`

Function: Transform the currently selected list of atoms and add them to the spheres list. This command actually performs the transformation specified by the transformation commands in Section 26.1.4 [Transformations for Sphere Drawing], page 215.

26.1.5.6 `WI` — Write Image to a File

Syntax: `WI` *integer*

Function: Writes the current Image to the unit specified as an integer. The file format consists of unformatted records containing just the non-zero pixels.

26.1.5.7 `WS` — Write Spheres to a File

Syntax: `WS` *integer*

Function: Write the Spheres list to the unit specified as an integer. The file format is a simple formatted list of coordinates, radii, and colors with the title being the first record, and the number of spheres being the second.

26.1.6 Adjustment of the Lighting and Shading

The equation for determining the intensity of a particular pixel is as follows:

```
                            _   _       _   _ G
I = I    + (I   - I   ) (K  N . L + K  (R . O)
     LOW     HI    LOW    D          S

where
  I      minimum intensity for a pixel. This provides for
   LOW   an ambient light level. Defaults to 10.

  I      maximum intensity for a pixel. Useful if the
   HI    display saturates below 255. Defaults to 255.

  K      coefficient of diffusive reflection, ie. uniform
   D     scattering of light. Defaults to 0.5.

  K      coefficient of specular reflection, ie. where
   S     light reflects in a particular direction. Defaults to
         0.5.

  _      Unit normal vector of the sphere's surface at the pixel
  N      coordinate.

  _      Unit vector pointing to the light source. Defaults to
  L      (0,0,1). A vector of (0,0,-1) is perfect backlighting
         which sets all pixels to I   . (0,1,0) gives right lighting.
                                   LOW

  _      Unit vector giving the reflected light vector off the
  R      surface. Computed as follows:

                _     _     _ _   _
                R = 2 N . L N - L

  _      Vector to observer. This is preset to (0,0,1)
  O      so that the observer is at z = infinity

  G      Glossiness exponent. This controls how accurately
```

```
                  the reflected light vector must point toward the
                  observer. Low values (3-6) give a metallic appearance.
                  Larger values (30-100) give a more plastic appearance.
                  Defaults to 6.
```

All negative dot products are converted to zero which models the opacity of the surface.

The commands for manipulating the shading parameters are as follows:

26.1.6.1 `BC` — Set Background Color

Syntax: `BC` *color real*

Function: Sets the background color. The color is specified as either a string or an integer, see the color commands in Section 26.1.3 [Atom Manipulations in Sphere Drawing], page 213. The second parameter is an intensity which may range between 0 and 1. It selects what intensity the background color will be.

26.1.6.2 `GLOS` — Set Glossiness Drawing Parameter

Syntax: `GLOS` *integer*

Function: Sets G to whatever value you specify. Negative arguments are converted to 0.

26.1.6.3 `LRAN` — Set Lighting Range

Syntax: `LRAN` *integer integer*

Function: Sets the lighting range. I_{LOW} is set to the first argument. I_{HI} is set to the second argument. The range of permissible values are 0 to 255.

26.1.6.4 `LV` — Set Lighting Vector

Syntax: `LV` *real real real*

Function: Set the X, Y, and Z components of the lighting vector to the three arguments, respectively. The vector is normalized automatically.

26.1.6.5 `SDCO` — Set Lighting Coefficients

Syntax: `SDCO` *real real*

Function: Specular Diffuse COefficient setting. The specular and diffuse scattering coefficients are set to the first and second arguments, respectively. The coefficient are normalized so that their sum is one.

26.1.6.6 `SL` — Show Lighting Parameters

Syntax: `SL`

Function: Show Lighting parameters. Prints the various shading parameters.

26.1.7 Miscellaneous Sphere Drawing Commands

26.1.7.1 `END`, `EXIT` — Exit Sphere Drawing

Syntax: `END`

Function: Return to CONGEN. All data structures are maintained until `SPHERE` is reinvoked.

Syntax: `EXIT`

Function: Same as END.

26.1.8 Example 1 — Full Screen Image

In this example, we produce a full screen image of the Fv of KOL.

```
Example of drawing KOL
*
OPEN NAME CGDATA:RTOPH8.MOD UNIT 01 READ UNFORM
OPEN NAME CGDATA:PARAM5.MOD UNIT 03 READ UNFORM
OPEN NAME CGTD:KOLPSF.MOD UNIT 12 READ UNFORM
OPEN NAME CGTD:KOLV.MOD UNIT 14 READ UNFORM
READ      RTF UNIT 1
READ      PARAMETER UNIT    3
READ PSF FILE UNIT 12
READ COOR FILE UNIT 14

! At this point, all requisite data structures have been read.

OPEN UNIT 20 NAME KOL.IMG WRITE UNFORM          ! Get a file for the new image
SPHERE                                          ! Enter SPHERE.
TL ;       KOL;                                 ! Simple title
AS                                              ! Full screen scaling
TR                                              ! Make spheres list
IM                                              ! Make the image
WI 20                                           ! Write the image out.
EX                                              ! All done.
```

26.1.9 Example 2 — Making a Stereo Image

In this example, we produce a stereo image of the Fv of KOL.

```
Drawing a stereo image of KOL.
*
OPEN NAME CGDATA:RTOPH8.MOD UNIT 01 READ UNFORM
OPEN NAME CGDATA:PARAM5.MOD UNIT 03 READ UNFORM
OPEN NAME CGTD:KOLPSF.MOD UNIT 12 READ UNFORM
OPEN NAME CGTD:KOLV.MOD UNIT 14 READ UNFORM
READ      RTF UNIT 1
READ      PARAMETER UNIT    3
READ PSF FILE UNIT 12
READ COOR FILE UNIT 14

! At this point, all requisite data structure have been read.

OPEN UNIT 20 NAME KOLSTEREO.IMG WRITE UNFORM    ! Get a file for the new image
SPHERE                                  ! Enter SPHERE.
TL ;       KOL;                         ! Simple title
AS 2.0                                  ! Half size scaling
OC -3.25                                ! Move the image to the left side
TR                                      ! Make spheres list for left side
SO 6 6.5                                ! Make right side transformation
TR                                      ! Add spheres for the right side
IM                                      ! Make the image
WI 20                                   ! Write the image out.
EX                                      ! All done.
```

26.1.10 Example 3 — Nine Drawings in One Image

In this example, we prepare an image which has 9 pictures generated from time points in a molecular dynamics calculation of McPC 603. Only the hypervariable loops are displayed and the loop are color coded so that they can be distinguished.

```
Display of 9 images of hypervariable loops of McPC 603
*
OPEN NAME CGDATA:RTOPH8.MOD UNIT 01 READ UNFORM
OPEN NAME CGDATA:PARAM5.MOD UNIT 03 READ UNFORM
OPEN NAME [BRUC.SEARCH]M603VPSF.MOD UNIT 12 READ UNFORM
OPEN NAME [BRUC.M603.A.NEW]M6VAMN18.MOD UNIT 14 READ UNFORM
READ      RTF UNIT 1
READ      PARAMETER UNIT    3
READ PSF FILE UNIT 12
READ COOR FILE UNIT 14
!
! At this point, everything we need except the trajectory has been read.
! We now orient the coordinates to a standard orientation.
!
COOR ORIENT CLEAR ATOM H 22 CA ATOM H 98 CA ATOM L 23 CA ATOM L 94 CA
OPEN UNIT 20 NAME MCPDYN.IMG WRITE UNFORM       ! File where image will go
SPHERE
TL ;         McPC 603 DYNAMICS, 500 K. 0,6,12,20,34,54,74,94,114 PS;
!
! Display only the hypervariable loops.
!
RM ALL
AD CLEAR RANGE H 31 N H 35 O -
         RANGE H 51 N H 65 O -
         RANGE H 101 N H 107 O -
         RANGE L 24 N L 40 O -
         RANGE L 56 N L 62 O -
         RANGE L 95 N L 103 O
AS 3.2                                          ! Need three fold reduction
YR 180                                          ! Reorient for better view.
CO 1 CLEAR RANGE H 31 N H 35 O                  ! Color the loops
CO 2 CLEAR RANGE H 51 N H 65 O
CO 3 CLEAR RANGE H 101 N H 107 O
CO 2 CLEAR RANGE L 24 N L 40 O
CO 1 CLEAR RANGE L 56 N L 62 O
CO 4 CLEAR RANGE L 95 N L 103 O
OC -4.3 3.8                             ! First picture in top left
TR                                      ! Make spheres for first picture
EX                                      ! Back to CONGEN for more work
OPEN UNIT 14 NAME D18.CRD UNFORM READ   ! Trajectory
READ COOR FILE UNIT 14 IFILE 30         ! Get fresh coordinates
!
! Reorient like the first set
!
COOR ORIENT CLEAR ATOM H 22 CA ATOM H 98 CA ATOM L 23 CA ATOM L 94 CA
SPHERE
OC 4 0                                  ! Second picture to right of first.
```

```
TR                                        ! Add spheres for next picture. The
                                          ! previous transformation applies.
EX
!
! Spheres for the next seven pictures are generated the same way.
!
OPEN UNIT 14 NAME D18.CRD UNFORM READ
READ COOR FILE UNIT 14 IFILE 60
COOR ORIENT CLEAR ATOM H 22 CA ATOM H 98 CA ATOM L 23 CA ATOM L 94 CA
SPHERE
OC 4 0                                    ! Third picture to right of second.
TR
EX
OPEN UNIT 14 NAME D18.CRD UNFORM READ
READ COOR FILE UNIT 14 IFILE 100
COOR ORIENT CLEAR ATOM H 22 CA ATOM H 98 CA ATOM L 23 CA ATOM L 94 CA
SPHERE
OC -8 -3.7                                ! Fourth picture at middle left
TR
EX
OPEN UNIT 14 NAME D18.CRD UNFORM READ
READ COOR FILE UNIT 14 IFILE 170
COOR ORIENT CLEAR ATOM H 22 CA ATOM H 98 CA ATOM L 23 CA ATOM L 94 CA
SPHERE
OC 4 0                                    ! Fifth picture at center.
TR
EX
OPEN UNIT 14 NAME D18.CRD UNFORM READ
READ COOR FILE UNIT 14 IFILE 270
COOR ORIENT CLEAR ATOM H 22 CA ATOM H 98 CA ATOM L 23 CA ATOM L 94 CA
SPHERE
OC 4 0                                    ! Sixth picture at center right
TR
EX
OPEN UNIT 14 NAME D18.CRD UNFORM READ
READ COOR FILE UNIT 14 IFILE 370
COOR ORIENT CLEAR ATOM H 22 CA ATOM H 98 CA ATOM L 23 CA ATOM L 94 CA
SPHERE
OC -8 -3.7                                ! Seventh picture at bottom left
TR
EX
OPEN UNIT 14 NAME D18.CRD UNFORM READ
READ COOR FILE UNIT 14 IFILE 470
COOR ORIENT CLEAR ATOM H 22 CA ATOM H 98 CA ATOM L 23 CA ATOM L 94 CA
SPHERE
OC 4 0                                    ! Eighth picture at bottom center
TR
EX
OPEN UNIT 14 NAME D18.CRD UNFORM READ
READ COOR FILE UNIT 14 IFILE 570
COOR ORIENT CLEAR ATOM H 22 CA ATOM H 98 CA ATOM L 23 CA ATOM L 94 CA
SPHERE
```

```
OC 4 0                                  ! Ninth picture at bottom right
TR
EX
SPHERE
IM                                      ! Finally, construct the image from
                                        ! all the spheres we've collected.
WI 20                                   ! Write the image out
EX                                      ! All done.
```

26.1.11 Displaying Pictures

There are five programs available for displaying images, `diris` for the Silicon Graphics Iris 4D workstations, `sphrgb` for converting images to Silicon Graphics Iris 4D RGB image format, `DISPIMG` and `DISPMANY` for the Deanza belonging to the Cardiac Unit, and `DI340` for an Evans and Sutherland Picture System 340. `DISPIMG`, `DISPMANY`, and `DI340` will currently work only on a VAX/VMS system, and are configured for the computer environment at Massachusetts General Hospital. All programs take files generated by the `WI` command, see Section 26.1.5 [Data Flows for Drawing Spheres], page 218, and send them to the appropriate device. They are defined as shell commands when correctly installed, see Section 30.13 [VMS Installation], page 272, and Section 30.14 [UNIX Installation], page 273, and are part of the support programs, see Chapter 29 [Support], page 249. `DISPIMG` and `DI340` take a single file as an operand, and display the image stored therein. `DISPMANY` accepts a list of file names from `SYS$INPUT:` and displays them in sequence on the Deanza by overwriting the pixels without erasing the image first. This has the advantage of letting the eye see differences between two images.

In addition to `DI340`, there is a program, `SLU`, for setting the color lookup tables in the PS340 raster display.

26.1.11.1 `diris` – Display on Iris 4D workstations

`diris` takes an image file as its only argument, and displays on the Iris 4D screen. You have the option of setting the window size, and `diris` will center the image within that space. Use the right mouse button to bring up the normal pop-up menu manipulate the window. Typing ⟨ESC⟩ will kill the window.

26.1.11.2 `sphrgb` — Convert to SGI RGB Format

`sphrgb` converts an image file to the Silicon Graphics RGB format. Silicon Graphics provides a large number of unsupported[1] tools for manipulating these images. For example, you can create a series of snapshots of your system, and then use the `movie` program to display them sequentially at high speed. The tools are part of the '`4Dgifts`' subsystem.

`sphrgb` expects two arguments as follows:

```
sphrgb congen-image-file rgb-file
```

The size of image will be set to the size specified as the pixel dimensions in your `XP` and `YP` commands to CONGEN, see Section 26.1.4 [Transformations for Sphere Drawing], page 215, for more information. There is currently a limit of 4096 in the horizontal dimension for these images.

26.1.11.3 `DI340` — Display images on PS340

`DI340` takes an image file as its only argument and displays it on the E&S PS340 in Jackson 1306 using the Ethernet. (Note: this display location is determined by a `PATTCH` subroutine call in the

[1] as of Irix 3.3

program). Because the display is slow, the program displays every third scan line so that the overall appearance of the image can be seen quickly.

N.B. This program only works on a VAX/VMS system.

26.1.11.4 `DISPIMG` — Display Images on the Deanza

`DISPIMG` takes an image file as its only parameter and displays it on the Cardiac Unit Deanza image processor. This Deanza has only red and green display channels, so two qualifiers for the command are provided. The syntax is

```
                       { RED   }        { RED   }
DISPIMG file-spec /RED={ GREEN } /GREEN={ GREEN }
                       { BLUE  }        { BLUE  }
```

The value of each qualifier determines what color plane is displayed on the Deanza's color planes.

N.B. This program only works at Massachusetts General Hospital on the VAX connected to the Deanza Image Processor.

26.1.11.5 `DISPMANY` — Display Many Images on Deanza

`DISPMANY` will display a series of images on the Cardiac Unit Deanza image processor. This Deanza has only red and green display channels, so two qualifiers for the command are provided. The syntax is

```
                        { RED   }        { RED   }
DISPMANY file-spec /RED={ GREEN } /GREEN={ GREEN }
                        { BLUE  }        { BLUE  }
```

The value of each qualifier determines what color plane is displayed on the Deanza's color planes.

The files containing the images to be displayed are read one per line from `SYS$INPUT`:

N.B. This program only works at Massachusetts General Hospital on the VAX connected to the Deanza Image Processor.

26.1.11.6 `SLU` — Gamma Correction for E&S PS 340

`SLU` (*S*et *L*ook*U*p) will set the color lookup tables for the Evans & Sutherland Picture 340. The color lookup tables are used to adjust for the scaling of intensities for each of three primary colors. In our eyes, the lower end of the scales are too dark, so `SLU` provides a smooth mechanism for shifting the values. The function used is

$$f(x) = kx^{1/b}$$

where x is the position in the color lookup table and *f(x)* gives the adjusted value. k, which depends on b, is adjusted to correctly normalize *f(x)*. The program asks for b. `DI340` sets b to 1.

N.B. This program works only a VAX/VMS computer.

26.2 PEER Command

The `PEER` command is used for generating input files to the `peer` program, see Section 29.1.6 [Peer], page 250, for a complete description.

26.2.1 Syntax

```
PEER UNIT unit [RADIus real] [SEQLink] atom-selection
```

The `atom-selection` is described in Section 11.7 [Atom Selection], page 108.

26.2.2 Function

Each invocation of the `PEER` command generates a single object for use by `peer`. The object consists of three parts; a list of colors, a list of atoms, and a list of bonds. The colors come from the `SPHERE` command. The atoms are selected from `atom-selection` you specify in the command, and only atoms which have defined coordinates are eligible for inclusion into the object. The bonds can be specified in two ways. If you omit the `SEQLINK` option, then only bonds between selected atoms are included. If you include the `SEQLINK` option, then a bond is drawn between each selected atom. This latter option is useful for making alpha-carbon displays. The object is written to the file on the unit you specify with the `UNIT` operand. It is a formatted file, and if multiple `PEER` commands are issued to the same unit in a CONGEN run, each object will be appended to the file. Note that only the first color table is used by `peer`.

Before using the `PEER` command, you must first invoke the `SPHERE` command, see Section 26.1.2 [Usage of Sphere Drawing], page 212. It suffices just to enter the command, and then leave it. For example,

```
SPHERE
EX
```

The `SPHERE` command defines the color table, as well as specifying a default color for all the atoms in the system.

Normally, atomic radii are specified by the parameter set as the van der Waals radii. However, if the `RADIUS` option is used, then you can specify the radii of all the atoms explicitly. Do not specify a negative or zero radii, as CONGEN will interpret this to mean that the van der Waals radii should be used.

27 Miscellaneous Commands

The commands described in this section are generally more simple in nature than those of previous sections. Some are perhaps obsolete, but included for the sake of completeness.

27.1 PARALLEL Command

27.1.1 Syntax

```
PARALLEL  [CG     ] [NCPU integer] [SCHED {GANG} ] [OVERride]
          [CONGen ]                [      {FREE} ]
          [LOOPs  ]
          [OFF    ]
```

27.1.2 Function

The `PARALLEL` command is used to control the usr of parallel processing. Presently, parallel processing can be used for either the conformational search part of CONGEN, see Chapter 12 [Conformational Search], page 111, or compute intensive loops in the Poisson-Boltzmann electrostatics calculations, see Chapter 8 [Poisson-Boltzmann Electrostatics], page 69. It cannot be used for both. Parallel processing is only supported on multiprocessor Silicon Graphics workstations or servers. Parallel processing is not supported on any other machine as yet.

The options, `CG` or `CONGEN`, are used to turn parallel processing on for the conformational search. The option, `LOOPS`, is used to turn on parallel processing for compute intensive loops. The option, `OFF`, is used to turn parallel processing off. If none of these options is specified, then `CONGEN` searching is run in parallel.

The `NCPU` option is used to specify the number of CPU's to use for the calculation. If this is not specified, then the program will check the following in order:

1. The value of the environment variable, `MP_SET_NUMTHREADS`.
2. The value of the enrironment variable, `NUM_THREADS`.
3. The number of CPU's on the machine where CONGEN is running.

The maximum permitted value for `NCPU` is the minimum of 32 and the number of CPU's on the system where CONGEN is running, except if `OVERRIDE` is specified. In that case, the setting for `NCPU` is the minimum of 32 and any environmental setting or value of the `NCPU` keyword.

The `SCHED` option controls how process execution is scheduled. The option, `GANG`, specifies that all processes working together be scheduled as a unit. `GANG` is the default for compute intensive loops. The option, `FREE`, specifies that all processes are scheduled independently. This option is the default for conformational search.

See Chapter 12 [Conformational Search], page 111, for more information about parallel processing in the conformational search.

If you are running a conformational search which uses Poisson-Boltzmann elecstrostatics, it is more economical to parallelizes the PBE calculation, and leave the search running serially because this optimizes memory usage. Parallel searching requires one copy of the PBE data structure per process, and this takes up a great deal of space.

For large memory jobs running in parallel on SGI workstations, it is critically important to set the `STACK` resource limit down from the default value. See Section 27.16 [Rlimit Command], page 241, for more information.

27.2 CODES Command

27.2.1 Syntax

```
CODEs
```

27.2.2 Function

The CODES command invokes the subroutine, CODES, which determines all the parameter type code indices for the internal coordinates and hydrogen bonds. It is useful to call this function before invoking the CONGEN command if energy values must be computed in a conformational search, see Chapter 12 [Conformational Search], page 111.

27.3 Set Delimiter Command — DELIM

27.3.1 Syntax

```
DELIm char
```

27.3.2 Function

The DELIM command sets the default command delimiter for command options which are specified as strings. The delimiter may be only one character. The default value for the delimiter is a dollar-sign, $.

27.4 DRAW Structure Command (Obsolete)

27.4.1 Syntax

```
DRAW derivative-factor-spec frame-spec

derivative-factor-spec ::= [DFACt real] [NOMO]

frame-spec ::= UNIT integer [DASH real] [FRAMe integer] [RETUrn integer]
```

27.4.2 Function

The DRAW command (called directly from CONGEN, not to be confused with the DRAW command found under the ANALYSIS command) is useful for displaying small molecules. The output is a command file that can be read by various displaying and plotting programs such as PLT2. This command file can be edited for different types of displaying. In addition to atom positions and bonds, velocity and forces may also be displayed. The current keywords are:

NOMO	No molecule option (only velocities or derivatives).
DFACT	Derivative factor. The default is 0.0.
DASH	Spacing of dashed line used for hydrogen bonds. The default is .01.
FRAME	Specifies that a frame tag will be written first. The default is not to specify a frame.
RETURN	Specifies which stream the plotting program will return to after plotting this section. The default is no change.

27.5 STOP or EXIT Command

27.5.1 Syntax

```
{ STOP }
{ EXIT }
```

27.5.2 Function

The `STOP` or `EXIT` commands cause the program to terminate and to ignore all command that follow this command. This is useful for making temporary modifications to input files.

27.6 `DISTANCE` Command (Obsolete)

27.6.1 Syntax

```
DISTance
```

27.6.2 Function

The `DISTANCE` command will cause the distance for every pair of atoms to be printed on a separate line. This command is not recommended for systems with more than 50 atoms. It is a great way to waste paper and there are other way to obtain this information such as with the builder commands or the analysis commands.

27.7 Call User Subroutine Command — `USER`

27.7.1 Syntax

```
USER
```

27.7.2 Function

The `USER` command is described in greater detail in Section 2.11 [CONGEN Modifications], page 11.

27.8 Set Timer Variable — `TIMER`

27.8.1 Syntax

```
TIME integer
```

27.8.2 Function

The `TIME` command sets the value of `TIMER` in `COMMON /TIMER/` to the specified value. This variable is used to time different functions in the program.

1	will print out the time to evaluate ENERGY.
2	will print out individual component times in `ENERGY`, and the times for various components of the `EXEL` nbonds update.

27.9 Set Warning Level Command — `WRNLEV`

27.9.1 Syntax

```
WRNLEV integer
```

27.9.2 Function

The `WRNLEV` command sets the value of the `WRNLEV` variable in `COMMON /TIMER/` to the specified value. At the present (17-Nov-1990) this variable is not widely used. Suggested values for future use:

0	(default) should print brief warning and error messages for conditions that will affect outcome.

1	more extensive information on errors and some information on normal partial results and conditions.
2	verbose error messages and more normal processing information for debugging.
3	all information that might be relevant to an error condition plus checking results.
4, 5	debugging levels for anything you might conceivably want.
10 or higher	for term by term outputs from energy routines, or other tasks where huge amounts of data useful only in debugging might be generated.

27.10 `NOBOMB` Command

27.10.1 Syntax

```
NOBOmb
```

27.10.2 Function

The `NOBOMB` command prevents the program from bombing out if it hits an unrecognized command in the main program. This is useful for attempting interactive work.

27.11 Set Debugging Variables — `DEBUG`

27.11.1 Syntax

```
     DEBUg repeat(name int)

                     { ALLHP       }
                     { ALLOC       }
                     { ALLSTK      }
                     { CGCONS      }
                     { CGEN        }
                     { CLSCHN      }
                     { CORE        }
                     { DIVZERO     }
                     { ESOLVE      }
                     { FPE         }
                     { GENIC       }
                     { GEPOL       }
                     { GRID        }
                     { GRIDSIG     }
           name ::=  { INEXACT     }
                     { INVALID     }
                     { JCOUP       }
                     { MALLOC      }
                     { NOE         }
                     { OVERFLOW    }
                     { PARA        }
                     { PBE         }
                     { SEARCH_NEAR }
                     { TLIMIT      }
                     { TREE        }
                     { UNDERFLOW   }
                     { XCONF       }
```

27.11.2 Function

The `DEBUG` command set the value for various debugging variables in the system. Although you should check the source code for the exact details, the following table gives an approximate idea of what the variables do.

`ALLHP` Displays details about heap management (see Section 31.2 [Heaps], page 280). The settings are as follows:

1	Heaps are initialized when allocated.
2	Messages are printed for all heap allocation and freeing requests.
3	The state of the heap is printed with every allocation and freeing request.

`ALLOC` Attempts to debug allocation errors. The value for this variable is interpreted in two parts, the units digit and the tens digit. A "1" in the tens digit signifies messages about allocation and freeing should be printed; a "0" means no messages. The units digit is interpreted in a cumulative way (e.g., a setting of 2 implies the actions of a setting of 1). *Warning:*, there is bug in use of this debugging variable. Settings of 2 or higher are incompatible with a setting of 1 or 0. Once you raise this variable up over 1, do not lower it, otherwise, your storage will be mutilated. Also, when the program exits, there is a chance is will crash because some memory allocated before the first command will not be freed correctly.

0	No checking.

1	All memory allocated through `cgalloc` is set to -1 before being returned.
2	The heap allocation routines are used to allocate memory for `cgalloc`, and some checks are done when memory is freed using `cgfree`.
3	The memory allocated for `cgalloc` usage is checked everytime `cgalloc` and `cgfree` are called.

`ALLSTK` Display details about stack management (see Section 31.1 [Stacks], page 275).

`CGCONS` Controls display of information about dihedral angle and J coupling constraints in conformational search. When set to 1, a list of atoms affected by these constraints will be output. When set to 2, any sampling operation which is restricted by a constraint will be displayed as well. This second setting can be voluminous.

`CGEN` Displays the course of a conformational search.

`CLSCHN` Displays details about chain closure.

`CORE` Controls whether core dumps are made on Unix systems when an error occurs. By default, this variable is set to 1 which means core dumps are taken. If set to 0, then core dumps are not taken.

`DIVZERO` Controls handling of division by zero errors. See `FPE` option for meaning of the *value*. Normally set to abort, but only on the Iris.

`ESOLVE` Controls debugging display of simple equation solving. Default is 0, which means no output.

`FPE` Controls handling of all floating point exceptions. Only the Iris currently provides control of floating point exceptions. This option effectively substitutes for `DIVZERO`, `INEXACT`, `INVALID`, `OVERFLOW`, and `UNDERFLOW`. The value is interpreted as follows:

`0`	Continue execution, but substitute an appropriate operand when the error occurs.
`1`	Print a count of errors on `stderr` at the end of execution.
`2`	Provide a traceback each time the error occurs.
`3`	Abort execution with an attempted core dump when the error occurs.

Note: on SGI Irix 5.3, there is a bug which prevents this code from functioning. As a result, no exceptions are trapped at all. On SGI Irix 6.0, libfpe is unavailable, so no exceptions are trapped here either. Hopefully, future releases will take care of the problem.

`GENIC` Controls whether displays are made of atoms which are not found within generated segments. Normally, topology files are set up so that some atoms at the end of segments stick out the sides, and these generate errors.

`GEPOL` Turns on the print flag in the GEPOL code, see Section 27.15 [Gepol Command], page 239, and Section 17.1.3.3 [Atom Properties], page 160.

`GRID` Displays details about the space grid.

`GRIDSIG` Displays a signature of the space grid before every node expansion. This may be useful in diagnosing malloc errors or problems with serial vs parallel execution.

`INEXACT` No effect on any machine. Although the IEEE floating standard provides this exception for inexact arithmetic, it is not controllable on any machine. See `FPE` option for meaning of the *value*.

`INVALID` Controls handling of invalid floating point errors. See `FPE` option for meaning of the *value*. Normally set to abort, but only on the Iris.

`JCOUP` When non-zero, display details about the J coupling NMR constraints.

`MALLOC` When non-zero, turns on debugging of `malloc` calls on some machines (definitely, the Iris).

`NOE` Displays details about NOE constraint calculations. A setting of 1 shows the energy calculations in detail. A setting of 2 also display the NOE constraints prior to an energy calculation.

`OVERFLOW` Controls handling of overflow errors. See `FPE` option for meaning of the *value*. Normally set to abort, but only on the Iris.

`PARA` Controls output of debugging messages from parallel processing routines.

`PBE` Controls debugging information from the Poisson-Boltzmann electrostatics code. When set to 1, additional useful information about the calculation is output. At 4, statistics about the potential and charge densities are displayed, as well as other information. In the `PBE TEST` command, a setting of 3 will display the calculation of the terms in the spherical cavity test potential. At 5, individual calculations of the potential are output (don't do this unless you want to fill your disk!). At 11, the number of iterations for the test cavity calculations will be put into the potential grid.

`SEARCH_NEAR`
Displays details about searches near atoms.

`TLIMIT` Display information about atoms which have been retarded during molecular dynamics using the `TLIMIT` option, see Section 7.6 [Dynamics], page 61. The options are interpreted as follows:

`0` All debugging output is turned off.

`1` The number of atoms which have been retarded on each dynamics cycle is displayed.

`2` Each atom whose motion is slowed is displayed. This can be voluminous.

`TREE` Displays details about the conformational search tree.

`UNDERFLOW`
Controls handling of underflow errors. See `FPE` option for meaning of the *value*. Normally ignored, but only on the Iris.

`XCONF` Controls whether accessible surfaces are printed when the `ESURF` option is used with the `XCONF` command.

27.12 Set Weight Command Variables — `WEIGHT`

27.12.1 Syntax

```
WEIGHT repeat(name real) [END]

               { EB       }
               { ET       }
               { EP       }
               { EI       }
      name ::= { ENB      }
               { EEL      }
               { EHB      }
               { EC       }
               { ENOE     }
               { EJCP     }
```

27.12.2 Function

The `WEIGHT` command set the weight for each term in the potential energy function. The default value is 1.0 if no weights are specified. Since the code for the nonbonded and electrostatic terms are interdependent, these weights nust be the same. If not, the weight for the electostatic term will default to the weight for the nonbonded term.

`EB`
: Changes the weight of the bond energy term.

`ET`
: Changes the weight of the angle energy term.

`EP`
: Changes the weight of the torsion angle energy term.

`EI`
: Changes the weight of the improper torsion angle energy term.

`ENB`
: Changes the weight of the van der Waals energy term.

`EEL`
: Changes the weight of the electrostatic energy term.

`EHB`
: Changes the weight of the hydrogen bond energy term.

`EC`
: Changes the weight of the harmonic atom and dihedral angle constraints, see Chapter 11 [Constraints], page 95.

`ENOE`
: Changes the weight of the Nuclear Overhauser Enhancement constraints, see Section 11.5 [NMR Constraints], page 97.

`EJCP`
: Changes the weight of the NMR J coupling constraints, see Section 11.5 [NMR Constraints], page 97.

27.13 `GAUSSIAN` Command — Invoke Gaussian Program

The `GAUSSIAN` command is used to invoke the Gaussian program[1] Currently, this interface may only be used to calculate partial charges for fragments of the system. It uses the Gaussian 94 program to calculate the wavefunction and electrostatic field for the fragment. Four different methods are provided for calculating partial charges from the wavefunction. All of these methods determine partial charges by performing a least squares fit of the potential generated by the partial charges to the potential calculated using the wavefunction. The fundamental difference between the methods is the layout of points where the electrostatic potential is determined by the wavefunction, and subsequently used for the least squares fit of atomic charges.

The first method, `PDM`, uses two programs written by Don Williams, `PDM88` and `PDGRID`.[2][3] The `PDGRID` program lays out a grid of points around the fragment where the potential will be calculated, and the `PDM88` program does the least squares fit to determine the best values for the partial charges.[4] The other methods have been incorporated directly into Gaussian 94, and use different grid layouts. There is the scheme due to Merz, Singh, and Kollman,[5][6] identified by the keyword, `MK`; the scheme due to Chirlian and Francl,[7] identified by the keyword, `CHELP`; and the scheme to Breneman and Wiberg,[8] identified by the keyword, `CHELPG`.

All of these schemes have their own values for van der Waals radii encoded within them. However, the default in this interface is to use the radii from the parameters in CONGEN. If you want to use the radii in the external programs, use the `EXTRADII` keyword.

This command simplifies the use of these three programs. You specify the atoms you want charges for, and the programs are invoked in turn to calculate the charges. Remember that the time for the calculation increases approximately with the fourth power of the number of electrons. A number of files are generated when this command is executed. Normally, these files are deleted after the command is complete, but you can request that they be saved.

Note that the collection of atoms that you specify should be a complete molecule including hydrogens. It does not make physical sense to do anything different, although the program does not check for completeness.

If you use this command to calculate results that are eventually published, please ensure that both Gaussian 94 and the fitting scheme you use is properly referenced.

---

[1] Gaussian 94, Revision E.2, M. J. Frisch, G. W. Trucks, H. B. Schlegel, P. M. W. Gill, B. G. Johnson, M. A. Robb, J. R. Cheeseman, T. Keith, G. A. Petersson, J. A. Montgomery, K. Raghavachari, M. A. Al-Laham, V. G. Zakrzewski, J. V. Ortiz, J. B. Foresman, J. Cioslowski, B. B. Stefanov, A. Nanayakkara, M. Challacombe, C. Y. Peng, P. Y. Ayala, W. Chen, M. W. Wong, J. L. Andres, E. S. Replogle, R. Gomperts, R. L. Martin, D. J. Fox, J. S. Binkley, D. J. Defrees, J. Baker, J. P. Stewart, M. Head-Gordon, C. Gonzalez, and J. A. Pople, Gaussian, Inc., Pittsburgh PA, 1995.

[2] D.E. Williams, Quantum Chemistry Program Exchange, Program 568; PDM88 (which includes PDGRID)

[3] D. E. Williams, "Representation of the Molecular Electrostatic Potential by Atomic Multipole and Bond Dipole Models", *J. Comput. Chem.* **9**, 745-763 (1988).

[4] The program, PDM88, is obsolete. Although it serves the need for a charge calculation in CONGEN, the newer version has more features for those users interested in exploring charge calculations. For further information please contact Dr. Donald E. Williams, Department of Chemistry, University of Louisville, Louisville, Kentucky 40292, USA, Tel: (502)588-5975, Fax: (502)588-8149, E-mail: `dewill01@ulkyvx.bitnet`.

[5] U. C. Singh and P. A. Kollman, *J. Comput. Chem.*, **5**, 129 (1984).

[6] B. H. Besler, K. M. Merz, and P. A Kollman, *J. Comput. Chem.*, **11**, 431 (1990).

[7] L. E. Chirlian and M. M. Francl, *J. Comput. Chem.*, **8**, 894 (1987).

[8] C. M. Breneman and K. B. Wiberg, *J. Comput. Chem.*, **11**, 361 (1990).

27.13.1 Syntax

```
GAUSsian CHARges {SELECT atom-selection END}

         [BASIs word] [SCF word] [TOTAl real] [MEMOry int]

         [UNIT unit] [PREFix word] [EXTRadii] [DIPOle]

         [PDM [UNDEr real] [SHELl real] [SPACing real] ]
         [MK                                           ]
         [CHELP                                        ]
         [CHELPG                                       ]

         [STEPs repeat(step-options) END]

         [SAVE] [NORUn]

         [MERGe atom-selection END]

         repeat( AVERage atom-selection END )

                [ ALL           ]
                [ NONE          ]
                [ [NO]CREAte    ]
                [ [NO]HF        ]
step-option ::= [ [NO]GRID      ]
                [ [NO]POTEntial ]
                [ [NO]FIT       ]
                [ [NO]SCAN      ]
                [ [NO]DELEte    ]
```

See Section 11.7 [Atom Selection], page 108, for the syntax of an `atom-selection`.

27.13.2 Function

The `GAUSSIAN` command functions by writing a set of input files for Gaussian, `PDGRID`, and `PDM88`; preparing a Bourne shell script to execute each program in turn; executing the script; and reading the results. The options are interpreted as follows:

`CHARGES`	This keyword must be specified. It is anticipated that other functions of GAUSSIAN will be invoked in the future.
`SELECT`	The `SELECT` keyword is used to demark an `atom-selection` that identifies the atoms to be selected. By default, no atoms are selected, so you must specify something.
`BASIS`	The word which follows `BASIS` gives the basis set to be used by Gaussian. The default is `6-31G`.
`SCF`	The `SCF` option is used to replace the `DIRECT` and `QC` keywords to the `SCF` command in Gaussian.
`TOTAL`	The `TOTAL` option specifies the total charge of the fragment. The default is zero.
`MEMORY`	This specifies the number of words of memory to be allocated to Gaussian. This option is used for the %mem keyword. The default is 5000000.
`UNIT`	This option is used to specify the Fortran unit for all I/O done by this command. The default is 99.

`PREFIX` This option specifies the file name prefix to use for all the intermediate files generated by this command. The default is '`cgq_<pid>`' where '`<pid>`' is the current process id.

`EXTRADII` This option specifies that the atomic radii in the external programs be used. Normally, CONGEN supplies the programs with the van der Waals radii from the parameter file.

`PDM` This keyword indicates that the `PDM` programs of Don Williams should be used.

`UNDER` This keyword specifies the distance under the van der Waals radii for grid points to be placed. The default is 0.0, which means that no points are placed under the van der Waals radius. This option is used only if the `PDM` option is selected.

`SPACING` This keyword specifies the spacing between grid points. The default is 0.8 Angstroms. This option is used only if the `PDM` option is selected.

`SHELL` This keyword specifies the maximum distance of any grid point to the van der Waals surface of a molecule. This option is used only if the `PDM` option is selected.

`MK` This keyword specifies that the Singh, Besler, Merz and Kollman gridding scheme should be used.

`CHELP` This keyword specifies that the CHELP gridding scheme should be used. Note that this keyword may not be abbreviated.

`CHELPG` This keyword specifies that the CHELPG gridding scheme should be used. Note that this keyword may not be abbreviated.

`DIPOLE` For the `MK`, `CHELP`, and `CHELPG` schemes, this option specifies that the fit of charges shall also consider the quantum mechanical calculation of the dipole moment.

`MERGE` This option requests that charges on the selected atoms be merged with the atoms that they are bound to. Only atoms which have exactly one bond can be treated this way, and this option was created primarily for use with hydrogens.

`AVERAGE` The `AVERAGE` options are used to average charges. They are most appropriate when certain atoms are symmetric, but are exposed to different electrostatic environments. You can specify as many `AVERAGE` options as needed, but the sets of atoms may not overlap within the system being analyzed. After the averaging and merging steps are performed, the program adjusts all the charges to bring the total to the value you have specified.

`STEPS` The `STEPS` option allows detailed control over the execution of the programs. CONGEN maintains a list of boolean variables which specify whether certain steps will be executed, and the `STEPS` option controls the setting of these variables. Each option is interpreted sequentially to affect the variables. The presence of the string, `NO`, preceding some of the keywords means that the variable should be turned off. The options are interpreted as follows:

- `ALL` Turn on all steps.
- `NONE` Turn off all steps.
- `CREATE` Create all the input files and shell scripts necessary to run the charge calculation. If a particular step is omitted from the operation, the affected command in the shell script is commented out, so you can manually enable it by editing the script. In order to use this command effectively, you must use the same file prefix every time.
- `HF` Perform the initial Hartree-Fock single point calculation. In all methods except `PDM`, this option also controls the steps up to the `FIT` step.

`GRID` Execute `PDGRID` which lays out the grid where the electrostatic potential will be evaluated. This applies only to the `PDM` option.

`POTENTIAL`
Calculate the electrostatic potential on the grid points from the wave function. This applies only to the `PDM` option.

`FIT` Invoke `PDM88` which fits the charges to the calculated potential. This applies only to the `PDM` option.

`SCAN` Scan the results from the calculations back into CONGEN.

`DELETE` Delete intermediate files.

`SAVE` This option requests that all intermediate files be saved after the command runs. It has the same effect as `STEPS NODELETE END` option. Normally, all intermediate files are deleted.

`NORUN` This option requests that all initial input files be saved and not executed. This is useful when you want to modify the Gaussian and PD input files created by the program. Note that this option complete overrides the settings in the `STEPS` option.

The following table gives the file types for all the intermediate files used:

'`.chk`'	Gaussian checkpoint file
'`.hfi`'	Hartree-Fock input
'`.hfo`'	Hartree-Fock output
'`.gri`'	PDGRID input file
'`.gro`'	PDGRID output file
'`.grd`'	PDGRID grid specification
'`.pti`'	Electrostatic potential calculation input
'`.pto`'	Electrostatic potential calculation output
'`.pdi`'	PDM88 input.
'`.pdo`'	PDM88 output.
'`.cho`'	Filtered charge calculation output.
'`.sh`'	Shell script to run everything.
'`.log`'	Output from shell.

27.14 `TEST` Command — Test Internal Functions

The `TEST` command is used to test internal operations within CONGEN. Currently, three test operations are provided; energy derivatives, calculation of maximum contact distances for the VAVOID sidechain option when the hydrogen bond energy replaces the van der Waals energy, and calculation of torsion angle minima.

27.14.1 Test Command Syntax

```
TEST [DERIV  [DELTA real] [CUT real]]
     [VAHB   [MAXEVDW real]]
     [PHIMIN [SGRID real] [SYMMETRY int]]
```

27.14.2 Energy Derivative Test

The energy derivative test is invoked by the `DERIV` keyword. The code works by computing numerical derivatives of energy as shown:

$$\frac{\partial U}{\partial x_i}(x_i) = \frac{U(x_i + h) - U(x_i - h)}{2h}$$

and comparing them against the analytic derivatives computed by the code. Statistics of the differences are calculated and a histogram of the differences is displayed.

The keyword, `DELTA`, specifies the value of h in the above expression. The default value is 0.001. For single precision arithmetic, this value is close to optimum.[9] The keyword, `CUT`, specifies a printing cutoff. Any difference in derivative whose magnitude exceeds this value will be individually displayed.

27.14.3 Vavoid Hydrogen Bond Test

The VAVOID hydrogen bond test checks the calculation of maximum distance for a given value of `MAXEVDW` for the hydrogen bond potential. It is invoked using the `VAHB` keyword. It performs the calculation for all hydrogen bond parameters and shows the distance along with the calculated energy at that distance. The keyword, `MAXEVDW`, is used to set a value for the cutoff energy. The default value is 20 kcal/mole.

27.14.4 Torsion Energy Minimum Test

The torsion energy minimum test is designed to check the code which finds torsion angle values to use when performing a sidechain degree of freedom in a conformational search, see Section 12.5.3 [Sidechain Degree of Freedom], page 133. In order to see the effect of this test, the `CGEN` debug variable, see Section 27.11 [Debug Command], page 230, must be set to 2 or larger. The `SYMMETRY` keyword specifies the rotational symmetry for the clump and defaults to 1. The `SGRID` keyword specifies the sidechain grid to use. A value of -1 means use the minimum energy periodicity. Positive values are interpreted as the grid in degrees. The default is -1.

27.15 `GEPOL` Command — Set GEPOL Defaults

The `GEPOL` command is used to set defaults for GEPOL surface calculations[10] in the analysis facility. See Section 17.1.3.3 [Atom Properties], page 160, for more information about the GEPOL surfaces.

There is also an experimental capability to invoke the `GEPOL_INCR` subroutine using the `RUN` option.

27.15.1 Syntax

```
GEPOL [NDIV int] [OFAC real] [RMIN real]

      [RSOL real] [RGRI real] [CAVIty real] [[NO]BULK]

      [RUN run-options]

run-options ::= [WSURF] {INIT                } [PEER <UNIT>] [PROP <UNIT>]
                [ASURF] {PUSH atom-selection}
                [ESURF] {POP                 }
```

[9] See S.D. Conte and C. de Boor, *Elementary Numerical Analysis, an Algorithmic Approach*, McGraw-Hill Book Company, New York 1972, p. 282

[10] J.L. Pascual-Ahuir, E. Silla and I. Tunon, *QCPE* 554, 1993

27.15.2 Function

The options in the `GEPOL` command control parameters used by the GEPOL algorithm.[11][12][13] They have the following interpretation:

`NDIV`
: `NDIV` specifies the division level for the triangles on the surface. It may be a number between 1 and 5. The accuracy of the calculation improves as NDIV rises, but the CPU time rises a lot faster. The default for this command is 3.

`OFAC`
: This parameter will be used only if the molecular surface is computed. In the second field goes a real number that can take values between 0.0 and 1.0. This parameter is the Overlapping FACtor. The accuracy improves as the OFAC value increases. The default value is 0.8.

`RMIN`
: This parameter will be used only if molecular surface is computed. In the second field goes a real number that can take values larger than 0.0. This parameter is the radius of the smallest sphere that can be created. The accuracy improves as the `RMIN` value decreases. The default value is 0.50.

  `OFAC` and `RMIN` are the parameters that control the creation of new spheres.

`RSOL`
: This parameter will be used only if accessible or molecular surface is computed. In the second field goes a real number. It is the probe or solvent radius. The default value is 1.4 Angstroms.

`RGRID`
: This parameter sets the dimension of the space grid used to find neighbors when GEPOL runs. The default setting of 2.5 Angstroms was optimal for running GEPOL on a large protein. A value of 0.0 will result in the dimension being set by code within '`$CGS/gridc.c`'.

`CAVITY`
: This parameters sets the cavity energy term. If this value is non-zero, then the conformational search command, see Chapter 12 [Conformational Search], page 111, will include a energy term equal to the molecular surface area calculated by GEPOL multiplied by this cavity energy factor. The units for the parameter are kcal/mole/$\mathring{A}^2$X. A good value for this parameter is 0.072.[14]

`BULK`
: The `BULK` flag controls whether the `BULK` subroutine is invoked as part of GEPOL. `BULK` adds spheres in the interior of large molecules to speed up the molecular surface generation.

The `RUN` keyword specifies that the `GEPOL_INCR` subroutine is to be run. The keywords; `WSURF`, `ASURF`, and `ESURF`; specify van der Waals, accessible, and molecular surfaces, respectively. The keywords; `INIT`, `PUSH`, and `POP`; specify initialization, pushing the selected set of atoms, and popping the last set of atoms, respectively. The `PEER` keyword specifies a unit where all the spheres will be written as input to the `peer` program, see Section 29.1.6 [Peer], page 250. The `PROP` keyword specifies a unit where the atomic surfaces will be written as a property table suitable for use in the Analysis Facility, see Section 17.1.6 [Table I/O], page 163.

[11] J. L. Pascual-Ahuir and E. Silla GEPOL: An improved description of molecular surfaces. I. Building the spherical surface set. *J. Comput. Chem.*, **11** (1990) 1047-1060.

[12] E. Silla, I. Tunon, and J. L. Pascual-Ahuir, GEPOL: An improved description of molecular surfaces. II. Computing the molecular area and volume. *J. Comput. Chem.*, **12** (1991) 1077-1088.

[13] J. L. Pascual-Ahuir, I. Tunon and E. Silla GEPOL: An improved description of molecular surfaces. III. A New algorithm for the computation of the Solvent-Excluding Surface. *J. Comput. Chem.* **15** (1994) 1127-1138.

[14] I. Tunon, E. Silla, and J. L. Pascual-Ahuir, Molecular surface area and hydrophobic effect. *Prot. Eng.* **5** (1992) 715-716.

27.16 RLIMIT Command — Set Resource Limits

The RLIMIT command is used to set and display computer resource limits. It is critically important to set the stacksize limit (STACK) when running very large calculations in parallel on an SGI workstation. The command is only implemented on SGI and Linux systems. On Linux systems, the VMEM option is not accepted.

27.16.1 Syntax

```
RLIMIT  repeat(limit [int     ])
                     [INFinity]

          [CORE  ]
          [CPU   ]
          [DATA  ]
limit ::= [FSIZE ]
          [NOFILE]
          [STACK ]
          [VMEM  ]
          [RSS   ]
```

27.16.2 Function

The options in the RLIMIT command set computer resource limits. After the command finishes execution, the current limits are printed. The keywords have the following meaning:

CORE	Sets the maximum core file size in bytes. A zero limit means no core files.
CPU	Sets the maximum CPU utilization per process in seconds.
DATA	Specifies the maximum extent of the data segment in a process in units of bytes.
FSIZE	The maximum size of a file in bytes. N.B. In Irix 5.3 (and probably 6.1), this limit can only be changed downward, because of a bug in setting this limit, where the system call requires both soft and hard limits to be changed simultaneously.
NOFILE	Maximum number of open files.
STACK	Maximum stack size. For parallel execution of CONGEN, this parameter is very important. The default value of the stack size, 64 MB, results in memory consumption of 1.28GB for a 20 processor run. CONGEN can run very well using a stack size of 1MB, which is recommended for these large jobs.
VMEM	Maximum amount of virtual memory allowed to a process. This is effectively the sum of the data segment and stack memory.
RSS	Declared amount of physical memory required by the process. When total system memory is limited, any process using more than this amount of memory will have its pages written to the swap file first.

27.17 SHELL Command — Execute Shell Command

The SHELL executes a command under the Bourne shell for Unix systems, or under DCL for VMS systems. The syntax is

```
SHELL string
```

where string is the command to be executed.

28 Experimental Commands

In this chapter, we describe commands that are more experimental in nature that the rest of CONGEN (although CONGEN is an experimental system throughout!). These commands are more likely to change and/or less trustworthy than other commands in the program.

28.1 `IMP` Command

The `IMP` command provides access to Inducible Multipole Solvation model calculations, which model solvation effects through inducible multipole moments at the atom centers of the solute.

N.B. This code fails one of the test cases (‘`imptest0`’) on the R8000 platform running Irix 6.1 and the 6.1 compilers.

28.1.1 Inducible Multipole Solvation Theory

28.1.2 Syntax and Usage of the `IMP` command.

28.1.2.1 Initializing an IMPS calculation

Syntax

```
IMP SETUP repeat(imp-setup-options) atom-selection

imp-setup-options ::= [SOLVent real]
                      [INTErior real]
                      [MAXIter integer]
                      [OUTIter integer]
                      [TOLErance real]
                      [radius-edit-options]
                      [charge-edit-options]
                      [KEEP]
                      [very-bogus-experimental-options]

radius-edit-options ::= repeat([RADIus edit-type real atom-selection END])

charge-edit-options ::= repeat([CHARge edit-type real atom-selection END])

edit-type ::= [SET]
              [SCALe]
              [SHIFt]

very-bogus-experimental-options ::= [VOLUme volume-option SCALe real]

volume-options ::= [LINEar]
                   [QUADratic]
                   [CUBIc]
                   [POWEr]
```

See Section 11.7 [Atom Selection], page 108, for the syntax of `atom-selection`.

Function

The `IMP SETUP` command initializes the data structure in preparation for solving for the induced multipole moments.

The environment can be defined. `SOLVent real` sets the solvent dielectric constant (default 78). `INTErior real` sets the solute interior dielectric constant (default 2).

The iteration control can be set (although this can be postponed or reset on the Section 28.1.2.2 [IMP SOLVE], page 244 command line). `MAXIter integer` sets the maximum number of iterations (default 20). `OUTIter integer` prints the convergence status every `integer` iterations. Zero, the default value, indicates that the convergence status should only be printed at termination. `TOLErance real` sets the fractional change in the total energy below which iteration will be terminated to `real` (default `1.E-06`).

Atomic parameters can be modified. Multiple `RADIus` and `CHARge` options may be used to modify the atomic radii and charges of selected groups of atoms. The modifications include setting them to a value with `SET`, multiplying them by a some value with `SCALe`, and adding a particular value to them with `SHIFT`. These effects of these options are cumulative and applied in the order they appear on the command line. So, for example, `RADIUS SCALE 2 END RADIUS SHIFT -1 END` would change all radii to twice their original values minus one Angstrom.

The `KEEP` option requests that the Coulombic tensors be saved between iterations. This saves considerable calculation time, but requires storage of `7*NATOM*(NATOM-1)` additional reals.

The "very-bogus-experimental-options" are all part of ongoing efforts to improve the method. These will disappear even before IMPS rises from being listed in the experimental section.

28.1.2.2 Solving for the induced multipole moments

Syntax

```
IMP SOLVE repeat(imp-solve-options)

imp-solve-options ::= [MAXIter integer]
                      [OUTIter integer]
                      [TOLErance real]
```

Function

Performs the iteration to self-consistency of the induced multipoles.

Iteration parameters can be set on the command line. `MAXIter integer` sets the maximum number of iterations (default 20). `OUTIter integer` prints the convergence status every `integer` iterations. Zero, the default value, indicates that the convergence status should only be printed at termination. `TOLErance real` sets the fractional change in the total energy below which iteration will be terminated to `real` (default `1.E-06`). Values specified on the `IMP SOLVE` line override those specified with the Section 28.1.2.1 [IMP SETUP], page 243 command, or any previous `IMP SOLVE` commands.

28.1.2.3 Print IMP parameters and results

Syntax

```
IMP PRINT imp-print-options atom-selection

imp-print-options ::= [repeat(imp-print-energy-options)]
                      [imp-print-non-energy-options atom-selection]


imp-print-non-energy-options ::= [PARAmeters]
                                 [CONVergence]
                                 [ATOMlist]
                                 [EFFEctive]
                                 [COULombic]
                                 [POTEntial]
                                 [DIPOle]

imp-print-energy-options ::= ENERgy [imp-print-energy-calc]

imp-print-energy-calc ::= symbol-name repeat( energy-operator symbol-name )

energy-operator ::= PLUS
                    MINUS
```

See Section 11.7 [Atom Selection], page 108, for the syntax of `atom-selection`.

Function

The `IMP PRINT` command permits the printing of values in the IMP structure. These include input parameters, as well as calculated effective parameters, and results. In addition, simple sums and differences between total energies from saved IMP results can be printed.

The `PARAmeters` keyword prints the setup parameters, including requested iteration limits. `CONVergence` prints the iteration results. `ATOMlist` prints the input atom parameters (coordinates, charge, and radius). `EFFEctive` prints the effective atom parameters (effective Born radius, Charge-charge correction, and polarizability). `COULombic` prints the Coulombic tensors, if they have been kept with the `KEEP` keyword on the Section 28.1.2.1 [IMP SETUP], page 243 command line. `POTEntial` prints the resultant potential and field at each of the atoms. `DIPOle` prints the resultant induced multipoles at each of the atoms.

The `ENERgy` keyword with no other specification prints the total energy of the current IMP structure. The total energy of an IMP structure saved with the Section 28.1.2.6 [IMP SAVE], page 245 command can be printed by specifying the `symbol-name` of the saved structure. Sums and differences in total energies can also be produced by including additional `PLUS symbol-name` and `MINUS symbol-name` specifications on the command line.

28.1.2.4 Read an IMP structure from a file

Not currently implemented.

28.1.2.5 Write an IMP structure to a file

Not currently implemented.

28.1.2.6 Save an IMP structure by name

Syntax

```
IMP SAVE symbol-name
```

Function

The `IMP SAVE` command takes the current IMP structure and saves it in a symbol table under the specified symbolic name. This structure can be recalled using the Section 28.1.2.7 [IMP RECALL], page 246 command, or be used directly in energy calculations using the Section 28.1.2.3 [IMP PRINT], page 244 command. To delete saved structures and free their associated memory use the Section 28.1.2.8 [IMP REMOVE], page 246 command.

28.1.2.7 Recall a saved IMP structure

Syntax

```
IMP RECALL symbol-name
```

Function

The `IMP RECALL` command copies a saved IMP structure (see Section 28.1.2.6 [IMP SAVE], page 245) into the current IMP structure.

28.1.2.8 Remove a saved IMP structure

Syntax

```
IMP REMOVE symbol-name
```

Function

This permanently removes an IMP structure saved with the Section 28.1.2.6 [IMP SAVE], page 245 command and frees the associated memory.

28.1.2.9 Clear the current IMP structure

Not currently implemented.

28.1.2.10 Generate a PBE-type potential grid

Syntax

```
IMP TOGRID repeat(imp-togrid-options)

imp-togrid-options ::= [GRID real]
                       [XDIM int]
                       [YDIM int]
                       [ZDIM int]
                       [XORIG real]
                       [YORIG real]
                       [ZORIG real]
                       [CENTER]
                       [REUSE]
```

Function

The `IMP TOGRID` command generates a PBE grid containing the electrostatic potential at each point. This is useful for comparison between the methods and for plotting electrostatic contour plots.

It must be noted that the PBE method is superior when it comes to generating a potential grid as it is a natural product of the method. The IMPS method, however, is geared toward obtaining the potential at the atoms, and requires that the potential grid be calculated as an afterthought. The current implementation, also does not approximate the transformation in any way. The potential at each point is take as the sum of the potentials from the final multipoles. As a result the conversion to the grid can be quite time consuming, requiring `XDIM*YDIM*ZDIM*NATOM` electrostatic potential calculations.

28.1.2.11 Print atomic volumes

Syntax

```
IMP VOLUME [PRECision real] atom-selection
```

Calculate the volume and surface area of the van der Waals surface by a number of methods: Dodd & Theodorou's exact method, a simple method ignoring overlaps of overlaps, a simple cubic gridding method.

This will most certainly be modified to improve the interface in the future.

28.1.3 Usage Examples

28.1.3.1 IMPS solvation energy calculation

```
IMP SETUP SOLVENT 78.
IMP SOLVE
IMP SAVE SOLVENT
IMP SETUP SOLVENT 1.
IMP SOLVE
IMP SAVE VACUUM
IMP PRINT ENERGY SOLVENT MINUS VACUUM
```

The first line sets up the IMPS data structure for the calculation of the induced moments when the external solvent has dielectric constant 78. The second line solves for the self consistent induced moments, and the third line saves the results under the name "SOLVENT". The fourth line sets up a new data structure with the external "solvent" dielectric set to 1 for vacuum. The fifth line solves for the new set of induced moments, and the sixth line saves the results under the name "VACUUM". The seventh line calculates and prints the difference between the total energy for the two saved results.

28.1.4 Deciphering error messages

The error messages are self explanatory. There are obtuse messages such as:

```
WARNING cosxx1=1+0.0004 reset to 1.
```

These indicate that the roundoff error has accumulated to the point where a calculated cosine is greater than 1 (or less than -1), which, of course, will cause the ensuing arccosine function call to choke. As long as the value following the plus sign (or minus sign) is small w.r.t. 1 you're okay.

If this number is large, it indicates a serious problem in the code, which should be brought to the developers' attention.

29 Support Programs

There are a number of programs and procedures available which assist in the use of CONGEN and which aid its development. We can divide these programs into three categories: programs which aid in the use of CONGEN, tools used in developing the program, and general programs for the manipulation of data. The tools used for program development are described in Chapter 30 [Implementation], page 259, and the other programs are described below. All of these programs are stored in the support directory. '`CGP`'. In the normal CONGEN setup, any user may execute these commands by simply typing their names. See Section 30.14 [UNIX Installation], page 273, and Section 30.13 [VMS Installation], page 272, for more details about the setup.

29.1 CONGEN Support Programs

A number of utilities exist to support the use of CONGEN as well as to create data files used by the program. There are also several programs for the display of space filling pictures of molecules. The interactive display program, `peer`, is described below, and the static display programs are described in Section 26.1.11 [Displaying Sphere Drawings], page 224.

29.1.1 `cmploop` – Preliminary Analysis of CGEN Files

The `cmploop` command provides a simple method to analyze CONGEN conformation file. The syntax is

```
cmploop [/b] [input-file [output-file]]
```

If no `input-file` is specified, the program defaults to '`congen.crd`', and if no `output-file` is specified, then standard output on Unix machines or '`SYS$OUTPUT`' on VMS machines is used.

The program will read the CONGEN conformation file specified as the `input-file` and it will compute the RMS difference between each conformation in the file and reference coordinate set stored in the beginning, see Section 12.3.4 [Conformation File], page 126, for more information. The RMS difference is computed only over those atoms which are defined in both coordinates and which are different. If there are no such atoms, as might be the case when the reference coordinate set is empty, then the RMS is reported at 9.999. The `/b` switch directs the program to use just backbone atoms; N, C, and CA.

`cmploop` also reports the energy of each conformation as evaluated by the last `EVL` degree of freedom used in the search, see Section 12.5.6 [Evl Degree of Freedom], page 136. It will also display the torsion angles for all backbone degrees of freedom used.

Frequently, one is interested in the conformations with the lowest energy or RMS deviations. See Section 29.2.1 [Sortn], page 256, for a command that will sort the output of `cmploop` by these values.

Since `cmploop` computes RMS deviations over only those atoms which have different coordinates, it can report different RMS deviations than a `COOR RMS` command using the same atoms, see Chapter 15 [Coordinate Manipulations], page 151.

29.1.2 `comparecg` — Compare Two Conformation Files

The `comparecg` command provides a mechanism for comparing two CONGEN conformation files which may have different orders of conformations. It is very useful when comparing results from two CONGEN runs when parallel processing has been enabled. The command takes the two conformation files, generates output from `cmploop`, see Section 29.1.1 [Cmploop], page 249, removes the ordinal numbers and summaries from the listing, sorts the files by the energy values, and runs a difference program on the results. This program is available only on Unix machines.

The syntax is

```
comparecg [-g] [-f font] [-b|-i|-w|-W|-D] [-N name] cg-file1 cg-file2
```

The `-g` option is used to specify which difference program to use. When missing, `diff` is used. Otherwise, `gdiff` is used. All the other options are passed to the difference program you select.

29.1.3 `comparecmp` — Compare Two `cmploop` Files

The `comparecmp` is identical to the `comparecg` command, see Section 29.1.2 [Comparecg], page 249, except that the input files are the outputs of the `cmploop` program.

29.1.4 `brkchm` — Converting Brookhaven to Congen Format

The program, `brkchm`, can be used to convert a Brookhaven data bank file to Congen format. The program will also generate a Congen input file to construct the PSF including disulphides. The program is interactive and reasonably self-explanatory. However, the program is not very robust, and it will often be necessary to edit the files it produces in order to get the structure built.

29.1.5 `homology` — Simple Homology Program

WARNING: This program is out of date with respect to current options in CONGEN.

The program, `homology`, is a simple sequence homology program which uses the homology finding code in CONGEN, see Section 18.3 [Matching of Comparison Data], page 175. In order to use the program, you must prepare two sequence files as described in Section 3.1.2 [Read Sequence], page 16, which will be the first requests of the program. Then, the program will ask for conserved residue sets, and you must respond with a set of lines giving residues which are to be considered equivalent. A blank line terminates this section. The program will then compute and report the homology and repeat the questions. You can terminate the program by specifying a blank file name for the first sequence file.

29.1.6 `peer` — Interactive Molecular Display Program

`peer` is a simple graphics program for "peering" into a set of molecule. It can also be used to manipulate the molecules orientation with respect to one another in a very crude way.

The fundamental operation of the program is to display the molecule using either vector or space filling representations. Transparent spheres are also supported. You can move or rotate the view at will using the dials or the keyboard. The program's input file specifies the molecules by their atoms and bonding connections. Up to 32 molecules can be read in. These molecules can be selected using the buttons, and when they are selected, they can be moved relative to the non-selected molecules.

Not all of these features of this program can be used on Personal Irises. In particular, anti-aliased vectors, depth-cued vectors, and transparent spheres are not supported on PI's.

The peer file is normally generated by the `peer` command in CONGEN, see Section 26.2 [Peer Command], page 225. It is a text file and describes of a set of molecules, each of which is given in three parts. Each of the parts consists an initial line count which is on a line by itself followed that many lines. The first part of molecule is the color table. Only the color table of the first molecule is used. Next comes a list of atoms specifying the segment id, residue name, residue id, IUPAC atomic name, XYZ coordinates, radius, and color. Finally, a list of bond connections is given, one per line where each bond connection is specified by atom numbers. In principal, such a specification could be made by hand.

There are two major modes of operation for peer, one where no molecules are selected, the other when some are. The difference between these two modes is the intepretation of the dials as we shall describe below.

29.1.6.1 Dial Usage in Peer

The dials are used to control either the entire view or to move individual molecules. When no molecules are selected, the dials have the following interpretation:

```
X rotation  o o  X translation
Y rotation  o o  Y translation
Z rotation  o o  Z translation
      Slab  o o  Scale
```

The rotations and translations apply to the entire view.

`Slab` and `Z translation` apply to clipping and display with regard to the front and rear of the viewing box. `Slab` controls the size of the viewing box using units of the vertical dimension of window. When this value is changed by the dials, it is displayed in the window title. The `z translation` specifies the position of the molecule within this viewing box. Its value is also displayed with this dial is changed.

You can make the molecule completely disappear by either shrinking the viewing box down too small, or by z translating the molecules out of the box.

When some molecules are selected, then the interpretation is similar except that `Slab` and `Scale` have no effect, the translations and rotations apply to the selected molecules only. The origin of rotation can be set with the `set origin` menu item.

29.1.6.2 Switches for Peer

The switch box is used to select molecules. When a molecule is selected, its color changes to the last color specified in the color table. Also, the light on the switch turns on. The switches act as toggles, so that successive operations of a switch select and then deselect the molecule.

29.1.6.3 Mouse Buttons for Peer

The left mouse button is used for atom identification. Move the cursor to the center of any atom, and touch the left mouse button. All atoms fairly close to the cursor will be displayed in the text window as well as the title bar. *Note:* it is difficult to pick an atom when drawn as a sphere because the pick area is very small compared to the usual sphere size. It is usually helpful to switch to the vector drawing before picking. The center mouse button has its default action of window moving. The right mouse button brings up the pop-up menu.

29.1.6.4 Keyboard Usage in Peer

In the event that dials and buttons are not available, keys on the keyboard may be used for those functions. The following set of keys are supported:

⟨F1-F12⟩	Toggle objects 1 through 12, respectively.
⟨Keypad 0⟩	Scale down in size if no objects selected.
⟨Keypad .⟩	Scale up in size if no objects selected.
⟨Keypad 2⟩	Z rotation, 2.9 degrees.
⟨Keypad 3⟩	Z rotation, -2.9 degrees.
⟨Keypad 5⟩	Y rotation, 2.9 degrees.

⟨Keypad 6⟩ Y rotation, -2.9 degrees.

⟨Keypad 8⟩ X rotation, 2.9 degrees.

⟨Keypad 9⟩ X rotation, -2.9 degrees.

⟨Left Arrow⟩ Translate Left.

⟨Right Arrow⟩
Translate Right.

⟨Up Arrow⟩ Translate Up.

⟨Down Arrow⟩
Translate Down.

⟨Esc⟩ Exit `peer`.

29.1.6.5 Pop-up Menu Items for Peer

The right mouse button brings up a large menu of options for controlling `peer`. In the commands which specify how a molecule is drawn, the behavior of `peer` depends on whether any objects are selected. If no objects are selected, then all molecules are drawn in the specified way. However, if any objects are selected, then only the selected objects are drawn according to the command. For example, if you wanted to draw a transparent space filling view of a molecule on top of a vector drawing of the same molecule, you would do the following:

1. `cat` the peer input file twice and pipe it into `peer`.
2. Hit button 1, and select the `Vector` menu item.
3. Hit button 1 again to deselect the first object, and hit button 2. Select the space filling drawing type of your choice.
4. Select the `Toggle object transparency` menu item, and you are done.

The pop-up menu options are as follows:

`Vector` Draw using vectors. If no objects are selected, then all molecules are drawn using vectors. If objects are selected, then only the selected objects are drawn using vectors.

`Anti-aliasing`
Toggles anti-aliasing for all vector drawn objects. Anti-aliasing greatly improves the appearance of vectors at a 50% performance penalty. This option does not work on Personal Irises.

`Toggle depth cueing`
Toggles depth cueing for all vector drawn objects. Depth cueing results in distant vectors being dimmer than close ones. This effect is not very pronounced for Irises in bright environments. This option does not work on Personal Irises.

`12 sided polyhedron`
Displays atoms as a regular dodecahedron. Object selection rules are the same as for `Vector`.

`60 sided polyhedron`
Displays each atom as 60 sided polyhedral sphere approximation. Object selection rules are the same as for `Vector`.

`240 sided polyhedron`
Displays each atom as a 240 sided polyhedral sphere approximation. Object select rules are the same as for `Vector`.

`Sphere library sphere`
: Uses the Silicon Graphics sphere library for drawing spheres. The lowest resolution sphere is an icosahedron, and the higher resolution spheres are generated by dividing the triangular faces. The next menu item sets the resolution. This option can give the best quality spheres, but at a great penalty in performance.

`Set sphere divisions`
: Sets the number of sphere divisions used for sphere library spheres. The text window where `peer` was invoked is popped to the foreground, and you are queries for your selection. If your response cannot be parsed as a number, then no change will be made.

`Stereo`
: Toggles stereo display. In a stereo display, the screen is split in half, and left and right eye images are displayed. The stereo is "wall eye", so the left eye image is on the left.

`Y rot +6`
: Rotates the view by six degrees. This option is essential for making stereo slides from the screen. To use it, simply take one picture, touch this item to rotate six degrees, and take the second member of the stereo pair.

`Y rot -6`
: Rotates the view by six degrees in the direction opposite to the previous item. This is used to restore the view after a stereo shot is made.

`Set Origin`
: Set the origin of rotation. The origin is set to the last atom picked using the left mouse button. When no molecules are selected, then the origin is for the entire picture. When molecules are selected, the origin for the rotation of the selected atoms. In this case, the picked atom must be in a selected molecule.

`Reset object position`
: Reset positions. When no molecules are selected, the view is restored to its initial state. When molecules are selected, then each of the molecules is returned to its initial position. In order to do a complete reset of the picture to its starting point, start with no molecules selected. Then, touch this menu item. Then, select all molecules, and hit this menu item again.

`Highlight objects`
: Normally, when objects are selected, they are displayed with a different color than normal. Alternatively, they can be displayed with their normal colors. This option toggles between those two possibilities.

`Toggle object display`
: Toggles whether selected objects are displayed.

`Color objects`
: This menu item will change the color used for selected objects. When touched, it will display another menu giving the choice of colors to use. Touch the color you desire, and all selected molecules will now have this color.

`Color touched residue`
: This menu item will change the color of all the atoms in the residue of the last atom touched. When touched, it will display another menu giving the choice of colors to use. Touch the color you desire, and the residue will now have this color. *Note:* This color will be seen only when the molecule is *not* selected.

`Reset residue color`
: This menu item will change the color of all the atoms in the residue of the last atom touched back to its initial value.

`Color touch segment`
: This menu item will change the color of all the atoms in the segment of the last atom touched. It works analogously to `Color touched residue`.

`Reset segment color`
: Reset the color of all the atoms in the segment of the last atom touched.

`Define color`
: Allows you to define new colors. When this item is selected, the window where peer was invoked will be popped to the foreground, and the current color table will be displayed. You can then specify a new color as a name and three sets of low, high pairs of color intensities. The intensities must not exceed the range of 0.0 to 1.0, and the pairs are given in red, green, and blue order. Once you specify a new color, any commands which request color changes will show your new color as an option. If you make a mistake in the specification, then nothing will happen.

`Toggle touched atom transparency`
: Toggles whether the last atom touched is displayed transparently.

`Toggle touched residue transparency`
: Toggles whether the last residue touched in displayed transparently.

`Toggle touched segment transparency`
: Toggles whether the last segment touched in displayed transparently.

`Toggle object transparency`
: Toggles whether the currently selected objects are made transparent

`Set transparency factor`
: Sets the transparency factor (alpha) for all transparent atoms. The window where `peer` was invoked is used.

`Set background`
: Sets the background color. Currently, there is a problem with the interaction of transparent atoms with a non-black background, whose solution remains a mystery.

`Change light direction`
: Change the direction of the light source. The coordinate system used for lighting has the x axis in the horizontal direction, the y axis in the vertical direction, and the z axis toward the viewer. The default direction is (0,0,1). The window where `peer` was invoked is used to type in this information.

`Set shininess`
: Change the shininess of the spheres. This parameter is the same as the glossiness parameter (`G`) in the sphere drawing commands, see Section 26.1.6 [Sphere Lighting], page 219. The window where `peer` was invoked is used for this.

`Write peer file`
: Ask for a file name, and write the current view to a new peer file. The new file presents the atoms in their current position, so molecule motions and views can be stored for later use, or for conversion to CONGEN format by `peercg`, see Section 29.1.7 [Peercg], page 255 for more information.

`Write CONGEN coordinates`
: Ask for a file name and a one line title, and write the current view to a new CONGEN file. The new file presents the atoms in their current position.

`Remove Border`
: Turns off the border display. *Note:* This does not take affect until the window is completely redrawn, eg by a resize or an iconification. This option is useful for a full screen display.

`Set draw frequency`
: This option is used to reduce the complexity of a drawing so that large objects can be moved more quickly. It specifies a reduction factor in the number of either atoms or bonds drawn. A value of 1 specifies everything should be drawn, and a value of *n* specifies that every *nth* atom or bond should be drawn. The value is taken from the window where `peer` was invoked.

`exit`
: Exits the program. Nothing is saved by default.

29.1.7 `peercg` — Convert `peer` Files to CONGEN Coordinates

The command, `peercg`, runs a simple `awk` script to convert a `peer` file into a CONGEN formatted coordinate file. The usage is as follows:

```
peercg peer-file congen-coordinate-file
```

29.1.8 `emap` — Backbone Maps and Proline Constructors

The construction of the backbone energy maps and the proline constructors is performed by a number of programs and user subroutines added to CONGEN, and is orchestrated by a makefile. The directory controlling the process is the '`emap`' directory under the '`CGP`' directory. To rebuild all these files within this '`emap`' directory, change your default directory to that one, and issue either a `make all` command on UNIX or an `mms all` command on VMS. To install new copies of these files in '`CGDATA`', use a target of `install` for the `make` or `mms`.

The pieces of the construction process are described briefly below:

'`proring.inp`'
: A CONGEN run which minimizes the energy of a proline ring constrained to a wide range of phi angles along a one degree grid.

'`mkprocns`'
: A simple program which reads '`proring.out`' and generates the proline constructor file, '`pro.cns`'.

'`emap`'
: A user subroutine for CONGEN which read in a specification of a map grid and calculates Van der Waals energies for all the points. These are written into the map files. This program is used for the glycine and alanine maps

'`emappro`'
: An analogue to '`emap`' that is used for the proline maps. The construction of the proline map requires an additional step to balance the energies of the cis and trans peptides. Only the five degree map is constructed using the input file, '`emappro.inp`'. Additional processing by '`remappro`' is needed to make the larger grid maps.

'`remappro`'
: A simple program to divide the 5 degree map produced by '`emappro`'.

29.1.9 `PDM88` — Potential Derived Multipole Program

`PDM88` is a program written by Donald E. Williams which is used in the calculation of charges. See Section 27.13 [Gaussian Command], page 235, for bibliographic references and for complete information on the interface from CONGEN to this program.

This version of the program is very similar to the program published in QCPE. It has been modified to detect I/O errors so that it will fail cleanly if the Gaussian job fails.

29.1.10 PDGRID — Potential Derived Grid Program

`PDGRID` is a program written by Donald E. Williams which is used in the calculation of charges. See Section 27.13 [Gaussian Command], page 235, for bibliographic references and for complete information on the interface to this program.

This version of the program is very similar to the program published in QCPE. It has been modified to accept an external specification of radii, and to detect I/O errors.

29.2 Data Manipulation Commands

There are a number of commands which manipulate data in a general way. Similar tools are available on most Unix systems, but most VMS systems lack them, and there are subtleties that make some of these programs more useful than similar commands.

29.2.1 sortn – Sort a Text File by Numbers (Real or Integer)

`sortn` and `bigsortn` sort files based on numbers found in each line in the file. The operation is specified through the command line, or if blank, the operations are specified interactively. `sortn` will sort a file containing no more than 25000 lines each containing no more than 133 characters. For `bigsortn`, the limits are 100000 lines.

The command line syntax is

```
sortn [/r] input-file output-file col1 size1 [col2 size2 ...]
```

`col`*n* are the positions of data, `size`*n* is the number of characters for each data entry. The output file is sorted based on these keys after they have been converted to numbers. If the fields cannot be converted, then zero is used instead. If `input-file` is specified as a ‘`-`’, then standard input is used. If `output-file` is specified as a ‘`-`’, then standard output is used. The flag, `/r`, specifies that the sorting should be reversed, i.e. biggest first.

29.2.2 extract — Extract Columns of Data from a File

`extract` extracts columns of data from a file and places them in another file. This program is similar to the Unix program, `cut`, and was primarily written to provide that functionality for VMS. The operation is specified through the command line, or if blank, the operations are specified interactively.

The command line syntax is

```
extract input-file output-file col1 size1 [col2 size2 ...]
```

`col`*n* are the positions of data, `size`*n* is the number of characters for each data entry. The output file consists of the columns concatenated together in the order specified with no blanks in between. If `input-file` is specified as ‘`-`’, then standard input is used. If `output-file` is specified as ‘`-`’, then standard output is used.

29.2.3 histo — Compute a Histogram from Data in a File

`histo` computes a histogram from the data in a file. The command line syntax is

```
histo input-file
```

If `input-file` is specified as a ‘`-`’, then standard-input will be used. Every word in the file which can be interpreted as a number is used.

As the program executes, you will be asked for the dimensions of the graph. The histogram is sent to standard output, and is designed to printed or displayed on a character device. `histo` uses the same code as the `HISTO` command in the Analysis facility, see Section 22.2 [Histo Command], page 199.

29.2.4 `scat` — Compute a Scatter Plot

`scat` computes a scatter plot from the data in a file. The command line syntax is

```
scat input-file
```

If `input-file` is specified as '`-`', then standard input will be used. The data is free form, mixed with non-numeric input. Each line containing two numbers is used with the first two numbers going into the plot. `scat` uses the same code as the `2DPLOT` command in the Analysis Facility, see Section 22.4 [2DPlot Command], page 200.

29.2.5 `numdiff` — Difference Based on Numbers

`numdiff` is used to compare two text files which contain numbers. The files must have the same number of lines, and must have the same structure for this command to work. The command line syntax is

```
numdiff file1 file2
```

Neither file may be specified as a hyphen ('`-`') — explicit names must be used. `numdiff` reads a line from each file, translates all punctuation to spaces, and then read each word on the line. An attempt is made to convert each word to a floating point number, and the minimum of the absolute and relative difference in magnitude is computed. If this number is larger than 0.01, then a message is printed and the two offending lines are displayed, otherwise, the difference is used to calculate the maximum difference over the entire pair of files. Large differences are not used in the calculation of the maximum, so differences in dates and CPU times will usually not affect the results. Messages are printed if numbers are not found in equivalent positions or if the number of words in a line do not match.

29.2.6 `ndiffpost` — Numerical Difference Postprocessor

`ndiffpost` is an alternate program for comparing two text files which contain numbers. To use `ndiffpost`, you first use the `diff` program to compare the files and then use `ndiffpost` to calculate numerical differences. `ndiffpost` is documented on a man page of its own.

30 The Implementation of CONGEN

The implementation of CONGEN is complex, because of the age of many parts of the code, the desire to preserve useful functions from CHARMM, the requirements of portability, and the need to provide as much functionality as possible. This chapter of the manual and the following one on storage management are essential to anyone interested in modifying the program.

CONGEN is implemented as single program. As a result, it is big. However, because of the use of dynamic storage allocation, it requires less initial storage than many contemporary modeling programs. By placing everything together, the task of modifying the program is made more reliable because errors in modifying the program are more likely to be noticed. This philosophy requires that testing be an integral part of the implementation of CONGEN. Ideally, there should be tests that exercise every line of code in CONGEN. As changes are made, the pre-existing tests are run to verify that changes were made correctly. As new features are added, new tests are constructed to ensure their correct operation.

CONGEN is implemented using FORTRAN, the FLECS preprocessor, and C. The reason for this mix is largely inertia. Early work on CHARMM was in FORTRAN because of its "easy" portability and convenience for numerical processing. Later, FLECS was used to make the program easier to read and modify. When the conformational search code was added, recursion became essential which lead to the use of C. There are no plans to rewrite CONGEN into one language because it is anticipated that others may wish to add Fortran code into the program.

The use of two languages in CONGEN requires an interlanguage interface. Most modern computers provide a mechanism for communication between C and Fortran. However, these interfaces are invariably machine dependent. In order to reduce the portability problems this entails, the program, `wrapgen`, has been written to localize the machine dependencies into one place, and free the CONGEN programmer from this problem while working on the regular code.

Besides the problem of the interlanguage interface, portability is a very important aspect in the implementation of CONGEN. In general, most operations in CONGEN are implemented using standard language features, but there are instances where machine dependent features are essential. In order to accommodate these machine dependencies, all the source code in CONGEN is run through a preprocessor. For the C code, the normal C preprocessor is used. For the FLECS/Fortran code, a modified version of the GNU Emacs C preprocessor is used, namely `fcpp`, or *F*ortran *C* *P*re*P*rocessor. Certain C preprocessor variables are defined which indicate compilation on a particular machine, and these variables are used to determine which code to compile. A configuration file, '`config.h`' is used to determine the settings of generic preprocessor variables.

Binaries for different platforms can be kept under one directory tree and mounted using NFS. See Section 30.14 [UNIX Installation], page 273, for more information.

Please note that this section of the manual is intended to be an overview. There is no substitute for studying the source code carefully. Although it is an important goal to keep this documentation up to date, one should not trust this documentation too much since it is effectively a copy of information which is always being changed. If there are any doubts about accuracy, the source code is the final arbiter.

30.1 The Structure of CONGEN

CONGEN has a very simple organization. The main program is primarily a command dispatcher. It reads each command from the input file, parses the first word which is a command verb, and executes code to parse the remainder of the command and then perform its function. Data storage is simple in principle — there are several dozen COMMON blocks which store information about the system being modeled, and the commands normally act to use or modify these COMMON blocks. Thus, many of the subroutines are independent of each other since they only depend on the data in the COMMON blocks.

Within CONGEN, there are also a large number of subroutines and functions that provide a convenient programming environment. For example, there are string manipulation routines, array manipulation routines, storage management capabilities, and operating system interfaces. Most of these routines are found in the source files; '`string.flx`', '`array.flx`', '`util.flx`', '`cutil.flx`', and '`cgutil.flx`'; and others can be found scattered throughout the program. The programming tool, `autodoc`, see Section 30.9 [Tools], page 266, can be used to list the comments from the subprograms within FLECS files.

30.2 Programming Environment

CONGEN is implemented using Fortran 77, the FLECS Fortran preprocessor, and C. All code is passed through a C preprocessor in order to handle conditional compilation. In general only standard features of the languages are used, but this rule is violated when there are strong reasons for doing so. (For example, the six letter maximum length of identifiers in Standard Fortran is far too onerous to bear, so the C limit of 31 characters is used, and there is a tool, `makeshort`, which can generate C preprocessor definitions to reduce the size of the variables to whatever machine dependent limits are necessary).

The management of revisions is done using RCS, the Revision Control System. We use version 5.5 obtained from the Free Software Foundation. Every source file should be kept under RCS control.

30.2.1 Use of Fortran and FLECS

The main reasons for using Fortran as a base language is its widespread usage among the molecular modeling community and its wide availability. Most hardware manufacturers concentrate their compiler optimization capabilities into Fortran. However, Fortran lacks good control constructs, data structures, and operating system interfaces, so steps have been taken to circumvent these limitations. The control constructs are provided by using FLECS, data structures are provided through either an elaborate storage management scheme (see Chapter 31 [Storage Management], page 275) or by using C, and the operating system interfaces are generally provided using C.

FLECS is a Fortran preprocessor that allows us to use a much more robust set of control constructs than normally provided in Fortran. This allows for much more readable and understandable code than could be obtained via Fortran alone. In addition, the use of FLECS allows the development of new programs much more quickly because much less time must be spent working out the flow of control. It is described in detail in section "Introduction" in *FLECS Manual*.

Specifically, FLECS provides a block structured `IF` — `THEN` — `ELSE` clause, a structured iteration clause, `WHILE` and `REPEAT WHILE` clause as well as their inverses, `CONDITIONAL` and `SELECT` clauses, and internal procedures. The internal procedures are especially valuable because they allow long subroutines to be broken into small pieces while leaving all variables accessible. In addition, one can give very long names to these procedures which makes their purpose far more clear. For example, `INTERPRET-COMMAND-AND-BUILD-CLAUSES` tells you a lot more than `CALL INTCBC` or `GOTO 285` does.

FLECS operates by translating any FLECS constructs into Fortran. Any non-FLECS constructs are merely copied. The Fortran compiler must be then be invoked to compile the translated code into machine language.

To invoke FLECS, type `FLECS` *file1 file2* .... Any number of files can be translated. The default file extension for FLECS programs is '`.flx`'. The Fortran translations will be produced in files whose extension is '`.f`' or '`.for`' depending on the machine FLECS is executed on. The FLECS listing files appear with extension, '`.fli`'.

Concerning the portability of FLECS, FLECS was written in itself. It was designed to be transported and has been modified to use the C preprocessor to encode machine dependencies.

Not all of CONGEN has been written using FLECS, as the preprocessor was not adopted until August 1979. All new code should be written using it, and old code in needed of major modification should use FLECS as well.

Two non-standard feature of most Fortran compilers is routinely used, variable name lengths and memory overlays. Variable names can be any length. There is a programming tool, `makeshort` which can be found in '`$CGP`', which can be used to translate all long names into short names. (In this day, six character variable names is really quite an anachronism!)

The dynamic storage allocation schemes, see Chapter 31 [Storage Management], page 275, depend on mapping arrays of one type onto arrays of another type. It is necessary that `REAL` variables occupy the same amount of space as `INTEGER` variables. Numeric arrays are not mixed with character arrays, which avoids many alignment problems.

30.2.2 Use of C

C was first used in CONGEN for implementing the conformational search algorithm. This algorithm required complicated data structure processing and recursion, for which the language is eminently suitable. Later on, C was used for operating system interfaces because many of the operating system calls in C are portable. For example, the storage management Fortran library uses `malloc` in the C library to obtain additional storage from the operating system without a requirement for machine dependent code.

One limitation in the use of C is I/O. The Fortran and C I/O libraries are not compatible. Therefore, all I/O from C must be done using Fortran routines. See the source code for `for_printf`, `for_scanf`, etc. in '`cutil.c`' for routines which provide this functionality.

30.3 FCPP — The Fortran C Preprocessor

FCPP is a modified version of the GNU Emacs C Preprocessor. This preprocessor provides all the functionality of the old style C preprocessor as well as `#elif` and the macros; `__LINE__`, `__DATE__`, `__FILE__`, and `__TIME__`. In addition, it contains additional features for Fortran code, and it has been ported to the VAX/VMS operating system in addition to other Unix platforms.

It is used on all Flecs code in CONGEN, and is described in greater detail in the `fcpp` manual page.

Note:, since `fcpp` is derived from the GNU Emacs C Preprocessor, it is licensed under the GNU Emacs License, which provides for more freedom to redistribute it than is available with CONGEN itself. Please see the license text in the source file, '`cccp.c`', or in your CONGEN license agreement, which incorporates the GNU Emacs License.

30.4 WRAPGEN — The Wrapper Generator

The program, `wrapgen`, is used to generate the interfaces between Fortran and C. It is described in greater detail in the `wrapgen` manual page. Briefly, `wrapgen` takes a file of prototypes which describe functions written in either C or Fortran, and it generates procedures which can be called from the Fortran or C, respectively. The wrappers takes care of the machine dependencies inherent in character string representations, character string lengths, call by value or reference, and naming rules, so that the programmer need not be concerned about these details. All source code must include a wrapper header file generated by wrapgen in order to correctly handle the substitution of wrapper function names.

30.5 MKPROTO — Make C Function Prototypes

The program, `mkproto`, will generate ANSI C function prototypes for all the procedures in a C source file. It is described in greater detail in the `mkproto` manual page.

In CONGEN, `mkproto` is used to avoid duplicating all the function definitions when preparing prototypes. Currently, it is used for all C files, except for '`tree23.c`' and '`noproto.c`'. The file, '`tree23.c`', is not run through `mkproto` because it is anticipated that it will be released as a separate procedure. The file, '`noproto.c`', is used for procedures for which `mkproto` doesn't work correctly. Presently, this occurs only with files containing machine dependencies where function definitions require the declaration of structures that are specific to a machine. Any necessary prototypes from this file can be put directly into '`funct.h`'.

30.6 The Configuration File, '`config.h`'

In order to simplify the use of machine dependent features, preprocessor variables have been defined which describe particular features. For example, the declarations of double precision variables in Fortran is given by the symbol, `DOUBLE_TYPE`. On most machines, this translates into `DOUBLE PRECISION`, but on the Cray, it translates to `REAL`. The source code file, '`config.h`', contains all these configuration variables. The values are set using a small number of machine identifiers which are set by the makefiles used to build CONGEN.

30.7 Making CONGEN

CONGEN is constructed using either the `make` command on Unix machines or the `MMS` utility on VMS. There is a master '`makefile`' or '`descrip.mms`' file in '`CG`', which will invoke all the necessary '`makefile`'s needed to build the entire system. These '`makefile`'s work fine on most machines, but have trouble on the Convex because their `make` program is older.

In order to handle the different compiler names and flags necessary for each different machine, many of the CONGEN directories have a file named '`makefile.gen`' which is missing definitions for these macros. In the '`CG`' directory, there are a set of '`*.make`' files which contain the macros needed for each type of machine. The names of the '`*.make`' files is as follows:

'`sgi...`' There are several files for Silicon Graphics workstations which are described in Section 30.14 [UNIX Installation], page 273. Currently, there are four versions of these files, `sgi_r10k_i6.4_c7.1_m4_a64.make`, `sgi_r3k_i5.3_c5.3_m1_a32.make`, `sgi_r4k_i5.3_c5.3_m2_a32.make`, and `sgi_r8k_i6.1_c6.1_m4_a64.make`

'`unicos`' Cray YMP running Unicos.

'`rs6000`' IBM RS/6000 workstation running AIX.

'`convex`' Convex computers.

'`hpux`' Hewlett Packard 700 series workstations running HPUX.

'`sparc`' Sun Sparcstations using `gcc` as the C compiler.

'`alphaosf`'
DEC Alpha machines running OSF.

'`fujitsu`' Fujitsu VP240.

There is a program, `fixmake`, in the '`CG`' directory which will take one of these files, and incorporate into a '`makefile`'.

The selection of machine is done via the `CGPLATFORM` environment variable which is set in '`$CG/cgdefs`' or '`$CG/cgprofile`'.

The master '`makefile`' has several different targets that can be specified to build parts of the system. Use them with care. They are listed as follows:

`prepare`	Prepare for rebuilding the entire system. This will set up to rebuild everything including binary data files. It should be used only when porting CONGEN to a new machine, or when file structures are changed.
`clean`	Clean up unnecessary copies of executables, intermediate data files, objects, etc.
`all`	Make everything in place. Executables made by this target are not copied to the directories which are included into user `PATH` environment variables.
`install`	Make everything and put them where users will access them. Note that the install script from the X window system is used.[1]
`tovax`	Copy files to `dino`. This should be used only at Bristol-Myers Squibb, but will not have any effect on other sites except if the file name '`/u/dino/fc/congen/...`' leads someplace real.
`setup`	This will construct all the '`makefile`'s from the '`makefile.gen`'s.

In order to construct CONGEN from scratch on a brand new machine, the following steps must be followed:

1. Modify '`$CG/cgdefs`' and '`$CG/cgprofile`' to reflect your directory structure and the hardware platform.
2. Create a platform file in '`$CG`' which specifies the necessary switches for compilation.
3. Set your default directory to '`$CG`'.
4. Do a `make prepare`.
5. Do a `make install`. Some of the utility programs like `peer` may not build correctly on all SGI platforms. This should not interfere with the installation of CONGEN.
6. Compare the test cases found in '`$CGT`'.

To make a new directory tree for a different SGI platform, perform the following steps:

1. Verify that the master tree name in `$CGROOT/master_machine` is correct.
2. Add a new platform file in '`CG`' which specifies the necessary switches for compilation.
3. Execute `cd $CGROOT`
4. Execute `./update_tree` *new_platform_name*
   The script, `./update_tree`, copies all the files from the master directory stored in the file `$CGROOT/master_machine` into the directory specified above as *new_platform_name*. It sets

[1] The installation script is Copyright © 1984, 1985,1986, 1987, 1988, 1989, 1990, 1991 by the Massachusetts Institute of Technology.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of MIT not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. MIT makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

up links for all RCS directories except the test case archive, which is duplicated for the new platform, because it is likely that the test values will change slightly.

5. Execute `source cgundef`
6. Follow the procedure for building CONGEN above starting at the `cd $CG` step.

30.8 Standards (Rules) for Writing CONGEN Code

The following set of rules is designed to help keep CONGEN readable and modifiable.

1. All routines should have similar organization.

   Each Fortran subroutine should have the following structure:

```
      SUBROUTINE DOTHIS(ARG1,ARG2,....
C
C     A comment which describes the purpose of this subroutine.
C     This comment is essential because it provides the only
C     documentation for nearly all subroutines in CONGEN. The
C     program, AUTODOC, can be used to get this comment from all
C     subroutines.
C
      <Declarations>
C
      <Code>
```

   The separation of the code from the declarations by a blank comment aids in reading the code. It becomes obvious where executed code begins.

   Each C procedure should be written like this:

```
dothis(type par1, type par2,...)
/*
*   A comment which describes the purpose of the routine. This
*   comment must come here so that automatic documentation can
*   be implemented (a program similar to AUTODOC is planned.)
*/

{
    local declarations

    code
}
```

2. Prototypes for all the C functions should be provided through the use of `mkproto`, see Section 30.5 [Mkproto], page 262. See the `makefile` for CONGEN for details of the mechanism. Note that not all files can be processed correctly, such as functions which are declared with structures or types that are machine dependent. All such functions should be placed in the file, '`noproto.c`'.
3. All source files must have a copyright notice at the top. See any source file for the appropriate text.
4. All C source files must include '`config.h`' and '`wrappers_c.h`' in that order. It is a good idea to use an existing source file to provide the copyright and includes to get started.
5. All FLECS source files must include '`config.h`' and '`wrappers_f.h`'.
6. All code should be written clearly. Since the code must be largely self-documenting, clarity should not be sacrificed for insignificant gains in efficiency. The use of C and the FLECS preprocessor is encouraged as it graphically illustrates the flow of control and allows for internal

procedure calls. Variable names should be chosen with care so as to illustrate their purpose. Avoid using one or two letter variable names in any COMMON blocks. Comments should be used where the function of code is not obvious.

7. All usages of integers, floats, doubles in C code must use the `F77_INTEGER`, `F77_REAL`, and `F77_DOUBLE` macros defined in '`config.h`'. Boolean variables should use the `BOOLEAN` macro, and Fortran logical variables defined in Fortran should use the `F77_LOGICAL` macro. The `F77_INTEGER` macro should declare to a `long int`. If not, you must review calls to the different `scanf` and `printf` functions to ensure correct typing.
8. Be careful to distinguish between Fortran 77 logical variables and C integers being used to hold Boolean variables. The testing conditions are *machine dependent.* Use the macros provided in '`macros.h`' for Fortran logicals.
9. Be careful that the type of any numeric constant match correctly with its usage across the various platforms that CONGEN is implemented on. For example, there may be problems with using a real constant in an intrinsic function call with multiple variables in Fortran, eg. `SIGN(1.0,P)`. If `P` is `DOUBLE_DECL`, you will have a problem because `DOUBLE_DECL` can map to either `REAL` or `DOUBLE PRECISION` depending on the machine. In such cases, it is better to use a variable or parameter to store the constant.
10. All usages of `DOUBLE PRECISION` variables in Fortran must be declared using the `DOUBLE_DECL` macro. This allows CONGEN to switch double precision variables to single precision on 64-bit computers.
11. Any variable in Fortran code that holds a pointer to be used by the C code must be declared using the `POINTER_DECL` macro. On 64 bit architectures, this macro will expand to 64 bit integers. The equivalent type in C is given by the `F77_POINTER` macro.
12. Any subroutine defined in C which can be called from FLECS code must have its prototype entered into the source file '`wrap_cdef.proto`'. Likewise, any subroutine defined in FLECS which can be called from C must have its prototype entered into the source file '`wrap_fdef.proto`'.
13. Whenever Fortran common blocks are accessed within C, you must use the predefined macro for the common block name. The macro is the upper case name for the common block. Header files (suffix '`.h`') are defined for all common blocks used in C code. For example, if you want to refer to the X coordinate array in C, use `COORD.x`.
14. There are number of rules associated with input and output:
    1. All input commands should be free field. The command processor should check that the entire command is consumed.
    2. Short outputs, messages, warnings, and error should be sent to unit 6 for output.
    3. All inputs should be echoed to unit 6. All values read by the command should also be output to unit 6.
    4. All warning and fatal messages should state what subroutine generated it, so that one find the location in the source code where the problem arose.
    5. All data structures output with unformatted I/O statements must have a `HDR`, `ICNTRL`, and `TITLE` in the first two records. See any existing binary output subroutines for the exact format.
    6. Unformatted I/O file formats should remain upward compatible. Use an `ICNTRL` array element to indicate which version of CONGEN wrote the file. Such upward compatibility must be maintained only across production versions of CONGEN. In other words, a file format for the developmental version may be freely changed until a new version is generated, at which point all future versions must be able to read it.
    7. All I/O must be done through Fortran I/O. C I/O is not to be used. See the procedures in the source code file, '`CUTIL.C`', for useful analogs of C I/O functions to make this rule easy to follow.

15. All error conditions must terminate with a `CALL DIE`. The subroutine, `DIE`, provides a traceback or core dump so the program statements causing the error can be seen.
16. Large or variable storage requirements for Fortran code must be met on the stack or heap. In C, `cgalloc` and `cgfree` should be used for all variable storage needs.
17. Array overflows must always be checked for when arrays are being written. This is especially important when the array being constructed might be dynamically allocated. Error checking in general should be as complete as feasible.
18. The code should use a minimum of non-standard Fortran or C features. All non-standard features must be conditionally compiled so that any CONGEN programmer is informed that the code is special.
19. In order to make subroutines callable from different contexts, parameter passing should be done through the subroutine call rather than through COMMON blocks.
20. All common blocks which are shared between multiple subprograms are to be placed in files and `#include`'d into the program. The common blocks should have comments describing each variable in the common block so that new users will know what's there. No directory should be specified for the `#include`'d files, so that the `-I` option to the C preprocessor can be used to select the directory at will. If a common block is to be shared between C and Fortran code, use the existing code in the `*.h` files to implement the needed name equivalence.
21. Avoid the use of static memory for initialization purposes. As more sections of CONGEN are implemented on parallel computers, making the subroutines reentrant is essential. Also, avoid the use of `EQUIVALENCE` and `DATA` statements in Fortran, since all storage referenced by these statements is allocated statically on the Iris.
22. When using `scanf` functions in C, use only long int's or doubles for your I/O, and then convert to your type. This avoids the need to control for machine dependent variations in data lengths.

30.9 Programming Tools

Presently, there is one tool for assisting in the development of CONGEN besides the language tools described within this section.

The program, `autodoc`, will collect information on all the entry points in a large Fortran or Flecs program and write them out using several different methods. For each entry point, the program collects the module name if different than the entry name, the file that entry point is in, the definition line of entry point, and the first block of comments which hopefully document the function of the routine. The files to be scanned are specified on the command line, and are written to file whose name is requested from the user when the program executes.

When `autodoc` is run, it will read the command line for files, and if none are found, it will ask you for files. Then, it will ask for an output file. It will then scan the files, and subsequently, it will ask you if you wish to sort the entry points by name. If not, the output will be in the order the files were read. Then it will ask if you want the short form of the listing. The short form is all the information on each entry except the comments. You will then be presented a list of subroutines which have no comments.

30.10 CONGEN Test Cases

The test cases may be found in '`CGT`' (as well as their developmental counterparts). All of these file generate output files which are to be compared with previous runs. In addition, some of the tests will generate other files which have the same file name, and these should be compared too. Scratch files have file names of '`FOO`' and file types which begin with the file name. For example, '`DYNTEST1`' generates a number of scratch files named, '`FOO.DYNTEST1_`*`nn`*', where *nn* is the unit number. These files should be deleted when the runs complete. The CPU time listed below is given in minutes for version 2 of CONGEN running on a single CPU of a Silicon Graphics 4D/200 series workstation.

Test cases run on platforms other than the Iris can be found in subdirectories under '`$CGT`' whose names match the platforms. For example, the Cray test case outputs are found in '`$CGT/unicos`'.

All the tests are run using the equivalent of the `RUNCG` command. On Unix machines, there are makefiles in both directories for running the test cases, and on VMS machines, there is a '`descrip.mms`' file. A target of `diffs` will make difference files for all the test cases.

The differences are always run through the program, '`ndiffpost`', see Section 29.2.6 [Ndiffpost], page 257, and the output are written to files which have a suffix of '`.dif`'. The raw difference files are output to files which have a suffix of '`.dif.raw`'.

File Name	CPU Time* (min)	Purpose
AM94CYCLE	1.9	Test of AMBER94 using a cyclic peptide. Modified from CGCYCLE.
AM94GENER	0.1	Simple generation test for AMBER94.
AM94SPL	2.6	Test of splicing using AMBER94.
AM94TEST1	4.4	Construction of all major residues in the AMBER94 topology file.
AM94TEST2	3.1	Repeat of first AMBER 3 demonstration run, energy calculation of alpha-lytic protease.
AM94TEST3	0.7	Simple conformational search testing AMBER94 energy calculation.
AM94TEST4	0.4	Test minimized ring constructions for AMBER94.
AM94TEST5	0.1	Checks the backbone and sidechain degrees of freedom work correctly at the endpoints of chains and prolines. Modified from CGTEST9.
AM94TEST6	0.6	Test parser errors when AMBER94 is used. Modified from CGTEST1.
AM94TEST7	2.3	Test D amino acid construction. Modified from CGTEST14.
AM94TEST8	2.2	Test AMBER 94 amino acid constructions. Modified from CGTEST15.
AMTEST1	0.4	Amber test 1, check terminal charges, part 1
AMTEST2	0.3	Check terminal charges, part 2
AMTEST3	0.6	Check conformational search with DNA protein complex.
AMTEST4	0.1	Test multi-term torsion term and conformational search.
AMTEST5	0.1	Test hydrogen bond term.
AMTEST6	0.3	Test amino acid construction in conformational search.
AMTEST7	0.2	Test antibody loop construction using AMBER potential.
BRBTEST	0.1	Tests Builder, Newton-Raphson minimization, and vibrational analysis.
CGCYCLE	6.4	Tests construction of cyclic peptides.
CGFIX	0.2	Test fixed atom construction in CONGEN.
CGFIX2	1.2	Test mixture of fixed atom and regular construction in conformational search.

CGHBUILD	0.2	Tests partial sidechain construction in the context of rebuilding hydrogen bonds.
CGMERGE	0.2	Tests merging of conformation files.
CGPARA1	1.8	Tests parallel processing in searching. The time given is the elapsed time.
CGPBE	17.1	Tests use of Poisson-Boltzmann equation with conformational search.
CGPBE2	21.4	Tests parallel implementation of PBE with conformational search.
CGPBE3	3.7	Test parallel conformational search using serial PBE evaluation.
CGRAND	0.3	Tests random node evaluation.
CGRESTART	2.1	Tests restarting when directed searching is done and MIX strategy used. (Currently fails on the CONVEX in `malloc`. No real idea why).
CGRESTART2	1.6	Repeat of CGRESTART, but without restart step. Output should match CGRESTART except for command processing.
CGRESTART3	6.9	Tests restarting when depth first search is used.
CGRESTART4	6.8	Repeats CGRESTART3 without restarting. Output should match CGRESTART3 except for command processing.
CGTEST1	0.1	Checks the CGEN parser. Many error messages are tested and no conformation file is written.
CGTEST2	0.2	Check ALL and FIRST sidechain construction options
CGTEST3	0.2	} Together, CGTEST3 and CGTEST4 check that the optimization
CGTEST4	0.7	} of the sidechain search for FIRST and ALL in the case where sidechains interact. CGTEST3 has the optimization, whereas CGTEST4 omits it. The CG files generated by both tests should match each other except for the first record, but CGTEST4 should take more CPU time.
CGTEST5	0.3	} CGTEST5 and CGTEST6 verify that the CLSA optimization
CGTEST6	0.5	} used with backbone degrees of freedom works correctly. The CG files should be the same, but CGTEST6 should take longer to get the results.
CGTEST7	0.6	Checks the energy calculations in the sidechain degree of freedom.
CGTEST8	0.5	Checks esoterica of CLSA and CLSD options
CGTEST9	1.9	Checks backbone termini processing and handling of prolines in both backbone and chain closure.
CGTEST10	0.9	Checks all sidechain construction options
CGTEST11	0.2	Tests van der Waals avoidance and Nosymmetry options in a single sidechain construction.
CGTEST12	2.6	Test of van der Waals avoidance in context of full search. Iterative option.
CGTEST13	0.9	Similar to CGTEST12 except Independent option used.

CGTEST14	6.4	Test of D amino acid construction and all amino acid sidechains
CGTEST15	6.4	Similar to CGTEST14, except we test the all hydrogen topology file.
CGTEST16	1.1	Simple test of overlapping degrees of freedom.
CGTEST17	0.5	Second test of overlapping degrees of freedom (sidechains).
CGTEST18	0.7	Test of coordinate writing and energy display filters.
CGTEST19	1.9	Test of ALLCISTRANS options.
CGTEST20	0.1	Test of other non-bonded energy calculations.
CGTEST21	0.1	Test RDEPTH search option.
CGTEST22	1.5	Test cavity energy calculation.
CGTEST23	1.7	Test combination of cavity and PBE energies.
CGTEST24	0.5	Test Worst RMS evaluation option.
CGTEST25	0.5	Test SGRID SELECT and AUTO options.
CGTEST26	4.2	Test cavity energy in evaluate degree of freedom.
CONGEN	0.3	A simple conformational search over five residues
CONGEN2	0.4	A two part conformational search over five residues
CORMANTST	0.1	Tests some coordinate manipulations.
CORTST1	0.1	A virtually worthless test of the correlation functions
DELTEST	0.1	Tests deletion by value in the analysis section
DJSTEST	0.1	Tests ABNER
DRAWTEST	0.1	Tests drawing capability of the program.
DYNTEST1	0.2	A series of tests on the dynamics algorithms. Not a complete test. Checks Gear and Verlet algorithms, SHAKE, ability to fix atoms in place. Also checks that the analysis facility can rotate a trajectory with respect to a fixed coordinate set. Some simple checks of dynamics analysis are also present.
GAUSSIAN	6.0	Test of interface to Gaussian 92.
GENERTEST	0.1	Tests some of the generation and patching routines.
GEPOL	0.1	Test GEPOL surface calculation.
GEPOL2	4.9	Test incremental GEPOL options.
GEPOL3	4.8	Another incremental GEPOL test.
H2OTST	0.1	Runs a water dimer to convergence and a true minimum. Also tests TLIMIT option.
HBCOMP	0.5	A self comparison of hemoglobin. Tests the comparison command in the analysis facility
HBMBCOMP	1.4	A comparison of hemoglobin to myoglobin. Tests comparison command and construction of difference tables.
ICTEST	0.1	Tests the routine that deal with internal coordinates.
IMH2OTEST	0.4	Water with periodic boundaries
IMPTEST0	0.1	Inducible multipole code, volume test.
IMPTEST1	0.1	Inducible multipole code, dimer test.
IMPTEST2	0.1	Small molecule test, analysis interface.
IMST2TEST	0.5	ST2 water with periodic boundaries.

IMTEST	0.1	Checks Images for a small system with C2 symmetry.
JTEST1	0.2	J coupling calculations on one leucine.
JTEST2	0.2	Ensemble averaging of J coupling calculations on two leucines.
JTEST3	0.2	J coupling calculations on one leucine with J errors.
JTEST4	0.3	Ensemble averaging of J coupling calculations on two leucines with joining with convergence tests.
JTEST5	0.1	Four leucine J coupling, ensemble averaging test with real data.
NANATST1	0.8	Tests most of the features of the analysis facility
NANATST2	0.7	Tests more features of the analysis facility
NANATST3	1.5	Tests the dynamic properties in the analysis facility
NOETEST	0.2	Tests NOE constraint calculations and calculation of energy derivatives.
NOETEST2	0.1	Test NOE ensemble averaging on a three atom system
NOETEST3	0.1	Test NOE ensemble averaging on a four atom system
NOETEST4	0.5	Test NOE code on a larger system.
NOETEST5	0.1	Test NOE code on beta hairpin using real data.
PBETEST	9.1	Test Poisson-Boltzmann electrostatics.
PBETEST2	1.5	More PBE testing. Thorough testing of options.
PBETEST3	0.1	Test of dielectric smoothing.
PBETEST4	0.1	Test of dielectric cavity
PBETEST5	0.3	Test of cavity in a Debye-Huckel fluid.
PBETEST6	4.2	Test of molecular surface usage in PBE code.
PBETEST7	5.5	Test of charge anti-aliasing
PBETEST8	1.1	Test of dielectric smoothing in a protein.
PBETEST9	0.1	Test of dielectric combination rules
PBETEST10	1.6	Test of dielectric constant modification based on accessible surface.
PBETEST11	0.2	Test of margin option.
PBETEST12	3.9	Margin and origin test using BPTI
PDBTEST1	0.4	Test #1 of Brookhaven Data Bank reading. Read tendamistat.
PDBTEST2	0.7	Test #2 of Brookhaven Data Bank reading. Read Fab KOL.
READTEST	0.1	Incomplete test of coordinate reading.
READTEST2	0.1	Test of sequence reading by atom.
READTEST3	0.1	IDREAD reading in PDB.
READTEST4	0.1	Test of alternate coordinate reading in PDB I/O.
READTEST5	0.1	Test of alternate model reading in PDB I/O.
SEARCHNOE	0.3	Tests conformational search with NOE's and also runs some simple tests of All Hydrogen construction.
SPHERE	2.7	Rudimentary test of sphere drawing.
ST2TEST	0.2	ST2 water without boundary conditions.
SURFTST	0.2	Checks the accessible surface calculation
TEST	0.1	Short test that hits a lot of stuff. Must always be run.
TESTCONS	0.7	Tests the harmonic constraints.

TESTCONS2	1.9	Tests the interaction of dihedral and J coupling constraints with the conformational search.
TESTHB	0.1	Test hydrogen bond calculations.
TESTHOM	1.7	Test homology finding code.
TESTPARM	0.1	Test AMBER parameter reading code.
TESTRTF	0.4	Tests the RTF I/O commands, and a simple test of PEER output
TESTRTF2	0.1	Test of charge generation in the RTF code.
TESTRTF3	0.1	Test automatic generation code on three and four membered rings.
TESTSEL	0.4	Tests the atom selection routines and use of wildcards in commands in the analysis section
TESTSPL	0.2	Tests SPLICE command
TRANSFORM	1.4	Tests coordinate transformation commands.
TWIST	0.1	Tests TWIST command in the analysis facility
VIBRTST	0.1	Tests vibrational analysis

* The CPU time is for code not optimized by the compiler.

30.11 Modifications to CONGEN

The following steps should be taken when making a change to CONGEN. They are intended to ensure that the change will be maintained in the future and does not unwittingly affect other program functions.

1. If you have not already done so, establish a directory of your own for working on the source, and set up a symbolic link to the source RCS directory, '`$CGS/RCS`'.
2. Make your modifications and debug them. Please follow the guidelines in Section 30.8 [Coding Standards], page 264, so that the code will be consistent. Use either `make` (on Unix systems) or `MMS` (on VMS systems) to rebuild the program.
3. Run the standard test case and conformational search test case and compare them. On a Unix machine using the C-shell, do the following:

   ```
   cd $CGT
   make test.dif congen.dif
   more test.dif congen.dif
   ```

   On VMS, do the following:

   ```
   $ SET DEFAULT CGT:
   $ MMS TEST.DIF,CONGEN.DIF
   $ TYPE TEST.DIF
   $ TYPE CONGEN.DIF
   ```

   The files should be identical except for the first four lines, version numbers or locations of files, and the last few lines giving the free list on the heap. If they are different in any other way, you must be able to prove that the results are correct. If you change any commands, the test case must be modified so that it will give the same results as before if possible. If you cannot duplicate the test case, you must eliminate your changes.
4. Run all the test cases in '`CGT`'. Use either `make diffs` on Unix machines, or `MMS DIFFS` on VMS machines. Any signigicant changes must be accounted for.
5. If your modification involves a new feature, you must either modify an existing test or make a new test to demonstrate and check its operation. See Section 30.10 [Tests], page 266, for a

description of the tests currently available. If you add a new test, please update that node. *WARNING*: Any additions made without this will stop working as the entropy of programming randomizes your code without detection.

6. Checkin your change (using the `co` command), and enter a good descriptive log entry for what you have done.
7. If your change involves adding or modifying a command or adding or modifying a feature, modify existing documentation or if none is available, make new documentation. Recreate the INFO file and the manual using the makefile in '`CGD`'.
8. If you modify or add new energy functions, use the `TEST` command, see Section 27.14 [Test Command], page 238, to verify that the derivatives of your energy calculations are correct.

30.12 Making New Versions of CONGEN

This section of the manual is not complete, but is left as a guide for future work on the process of generating new versions.

This section describes the steps in generating a new version of the protein system. It is constantly in flux and should be viewed as a guide.

1. Make sure the version number and date in opening output of CONGEN.FLX is correct for this new version.
2. Relink CONGEN if necessary.
3. Redo a `make depend` in those directories where it is supported.
4. Run all the test cases and compare against previous versions.
5. Recompile the program with optimization, and compare results again.
6. Rebuild the documentation.
7. Clean up all directories of garbage.
8. Make the tar files for distribution.
9. Do a global setting of symbolic version number for this release.
10. Backup the directory tree for posterity.

30.13 Installation of CONGEN on VMS

The installation of CONGEN on VAX/VMS is a very straightforward process. The files are organized so that CONGEN can be installed by either a system manager or an individual user without privilege. There are only a few steps to be taken:

1. The tape on which CONGEN is shipped contains a single saveset, CONGEN.BAC. Restore the tape while preserving the directory structure into a directory of your own choosing, e.g.

   ```
   $ BACKUP MUA0:CONGEN.BAC [CONGEN...]
   ```

   You will need about 100000 blocks to restore the saveset.
2. Modify the file, '`[CONGEN.V2]CGDEFS.COM`', to reflect your own directory structure.
3. Change either the system site specific startup file, '`SYS$MANAGER:SYSTARTUP.COM`', or your '`LOGIN.COM`' file to include a call to '`CGDEFS`' to set up logical names. Use an argument of `SYSNAM` or `JOBNAM` as appropriate.
4. Change either the system '`LOGIN.COM`' file or your '`LOGIN.COM`' file to include a call to `CGDEFS` to define commands, thusly, '`@CG:CGDEFS COMMANDS`'
5. You may wish to delete rarely used files such as the bulk of the test cases or source code object files.

6. Copy the INFO files (‘`congen`’, ‘`congen-*`’, ‘`flecsdoc`’, and ‘`flecsdoc-*`’) in ‘`CGD:`’ into the GNU Emacs manual directory and modify the INFO directory file so GNU Emacs can access the CONGEN documentation.
7. It is helpful if GNU Emacs is installed so you can read the documentation online.

30.14 Installation of CONGEN on UNIX

The installation of CONGEN on UNIX is a very straightforward process. The files are organized so that CONGEN can be installed by either a system manager or an individual user without privilege. There are only a few steps to be taken:

1. CONGEN is provided as a single gzip'ped tar file (you can get gzip from the Free Software Foundation (ftp to ‘`prep.ai.mit.edu`’ or other mirrors). Restore the tapes while preserving the directory structure into a directory of your own choosing, e.g.

   ```
   zcat congen.tar.gz | tar xvfo -
   ```

   You will need about 600000 blocks to restore the tar file.
2. Modify the files, ‘`./congen/cgdefs`’ and ‘`./congen/cgprofile`’, to reflect your own directory structure and machine usage. The files, ‘`cgdefs`’ and ‘`cgprofile`’, contain some special code for the definition of `CGROOT` which is used at our site to allow us to have two copies of CONGEN on different machines, and to switch based on the machines' availability. You can remove all the conditionals, and simply give it a definition.

   The machine designation is more complicated. The program, ‘`./congen/setcgplatform`’, will determine what machine you are running on. For most machines, there is only one possible machine, but on the Silicon Graphics computers, there are many possibilities based on the type of processor, operating system, compiler, instruction set architecture, and application binary interface. The SGI machine string is encoded as `sgi_r`*`chip`*`_i`*`O.S`*`_c`*`compiler`*`_m`*`architecture`*`_a`*`abi`*. The choices for each part are as follows:

chip	3k, 4k, 5k, 8k, 10k
irix	5.3, 6.1, 6.2, 6.3, 6.4
compiler	5.3, 6.1, 6.2, 7.0, 7.1, 7.2
MIPS	1, 2, 3, 4, 5
abi	32, n32, 64

   Most distributions of CONGEN contain only the most compatible version which is found under the machine directory, `sgi_r3k_i5.3_c5.3_m1_a32`. See the section on rebuilding CONGEN, Section 30.7 [Making CONGEN], page 262, for a description on how to build other versions of CONGEN which will give better performance on the newer architectures.

   The environment variable, `CGPLATFORM`, specifies which machine directory to use. The Perl program, `./congen/matchdir`, finds the closest compatible machine directory for the environment you are running on. You can override the automatic selections by setting `CGPLATFORM` prior to the execution of `cgdefs` or at the beginning of `cgdefs`.
3. Change either the system wide profile or ‘`.cshrc`’ file or your own profile or ‘`.cshrc`’ file to source ‘`cgdefs`’ or ‘`cgprofile`’.

   The file, ‘`cgdefs`’, is for the C shell, and the file, ‘`cgprofile`’, is for the Bourne or Korn shells. Once these files are executed, all of the commands will work.
4. Change your default directory to ‘`$CGT`’, and run two tests as follows:

```
make test.dif congen.dif
```

Examine those two files. The only differences you should see are file names, version numbers, dates, allocations in the heap, and execution times.

5. You may wish to delete rarely used files such as the bulk of the test cases or source code object files.
6. Copy the INFO files (‘`congen`’, ‘`congen-*`’, ‘`flecsdoc`’, and ‘`flecsdoc-*`’) in ‘`$CGD`’ into the GNU Emacs manual directory and modify the INFO directory file so GNU Emacs can access the CONGEN documentation.
7. It is helpful if TeX is installed. This will allow you to modify the documentation.

31 Storage Management in Fortran

One of the great limitations in Fortran is its inability to dynamically allocate storage. In a large program which must deal with problems are highly variable sizes, it is extremely valuable to allocate storage based on the size of the problem. Thus, a given computer can process a mix of problems as long as the aggregate size is not too large.

Even though Fortran does not allow dynamic storage allocation, we can conveniently simulate it so that arrays can be established at run-time with whatever size is necessary. These storage allocation schemes are written entirely in Fortran and do not require any machine dependent features.

The storage schemes described here are ubiquitous, and it is very difficult to follow the code without a good feel for these storage schemes. However, once they are comprehended, modifications are straightforward.

There are numerous sources in the computer science literature on stacks, heaps, and data structures. An excellent starting point is Jeffrey D. Ullman, Fundamental Concepts of Programming Systems, Addison Wesley (1976). Chapters 3, 6, and 7 are most useful.

31.1 Stacks

In all large programs, the need for “local” storage arises very frequently. “Local” storage or temporary storage is defined as storage needed by a subroutine for itself or any subroutine it calls. Invariably, local storage is required for work arrays whose sizes can be determined prior to their use. Local storage is not meant to be accessible when the subroutine is not executing.

An example requirement for local storage is a routine which squares a matrix. Because array multiplication cannot be done in place, one would require local storage to store a temporary array which would hold the squared matrix.

The data structure which is ideally suited for this task is a stack. At minimum, a stack consists of an array and an integer. Storage is allocated and freed from the top end of the array only. The integer is the stack pointer and keeps track of what part of the stack is in use. When a subroutine requires local storage, it gets it from the top of the stack. If it calls another subroutine which requires more local storage, the local storage currently being used is, in effect, buried, and the additional storage is allocated on the top of the stack. “Burying” such storage is reasonable because the first subroutine’s execution is suspended when the next one is executing. When it finishes, the storage is freed, and the top of the stack moves back to where it was. The first subroutine then resumes execution with the same stack environment before the other subroutine was called.

As seen above, the stacks allow nested subroutines to each have their own local storage. This is very important because CONGEN uses many subroutines, and they can nest many levels. Using blank common for local storage fails because a nested subroutine would overwrite storage used by its caller.

31.1.1 Implementation of Stacks in CONGEN

CONGEN maintains two stacks, one for numeric data and one for character string data. (The Fortran standard forbids the mixing of numeric and character data in a common block, so for portability, we keep them separate even though one could get away with mixing them on most machines.)

The two stacks are stored in two common blocks named `STACK` and `STACK`_ST. Both common blocks are defined in ‘`stack.fcm`’ or ‘`stack.h`’. The current declaration is as follows:

```
C
C     Variables:
C
C     LSTUSD      Stack pointer which refers to last used element in
C                 stack array.
C     MAXUSD      Maximum value of LSTUSD seen during execution.
C     STACK       Numeric variable stack.
C     LSTCUSD     Character string version of LSTUSD
C     MAXCUSD     Character string version of LSTUSD
C     CSTACK      Character string stack.
C
C     Parameters:
C
C     STKSIZ      Maximum size of stack.
C     CSTKSIZ     Character string version of STKSIZ
C
      INTEGER LSTUSD,STKSIZ,MAXUSD,STACK,ALLSTK,LSTCUSD,CSTKSIZ,
     2        MAXCUSD,ALLCSTK
      PARAMETER (STKSIZ = 100000, CSTKSIZ = 100000)
      COMMON /STACK/ LSTUSD,MAXUSD,LSTCUSD,MAXCUSD,
     2              STACK(STKSIZ)
      REAL RSTACK(STKSIZ)
      EQUIVALENCE (RSTACK(1),STACK(1))
      CHARACTER*1 CSTACK
      COMMON /STACK_ST/ CSTACK(CSTKSIZ)
```

LSTUSD and LSTCUSD hold the index of the last element used in each stack; they are the stack pointers. STKSIZ and CSTKSIZ hold the size of the stacks which is used for overflow checking. MAXUSD and MAXCUSD accumulate the maximum value of the stack pointers so that one determine how much storage must be dimensioned for the stack without wasting any space. As one can see, the stack must be dimensioned to some appropriately large size, and all local storage requirements will be taken from it. If the stack is not big enough for the needs of the program, the stack must be redimensioned. Changing the size of the stack is far easier than redimensioning every array that would have declared locally.

There are some valuable techniques for using the character string stack. Note that the stack is declared as an array of CHARACTER*1. This was done to allow any size character string because on some machines (e.g. VAX/VMS), strings are limited to 32767 characters. When an allocated character string is passed to a subroutine, the space should be referenced as a character substring of the allocated length. Then, the subroutine will see the length of character string as the allocated length, and subscript range checking is simplified. If a character string array is allocated, the passed length should still be the length of the string element. Please note that if the passed length derives from a constant, then most compilers will detect the apparent substring range violation. Therefore, it will be necessary to allocate an extra integer variable, assign the constant to it, and use the integer variable in the subroutine call.

To refer to the stack in a subroutine, use a #include statement and refer to the file 'STACK.FCM'. For convenience in debugging, the stack is also declared as a REAL array using an equivalence statement.

The stacks are limited in size, and it is not memory effective to increase their size greatly. Therefore, if you have large local storage needs, use heap storage, see Section 31.2 [Heaps], page 280, for the really big arrays.

31.1.2 Useful Subprograms

A number of subroutines and functions are provided to perform all the necessary manipulations of each stack. They are found in the source code file, '`util.flx`'.

31.1.2.1 `INISTK`

To initialize the stacks, one must call INISTK with no arguments. INISTK sets all the stack variables appropriately, and if `DBG_ALLSTK` is greater than zero (it normally is), the stacks are filled with non-zero elements. The initial filling of the stacks helps to catch error which occur because local storage is not properly initialized.

31.1.2.2 `ALLSTK` and `ALLCSTK`

To obtain storage on the stacks, one uses the integer function `ALLSTK` or `ALLCSTK`. Both functions take one argument, the number of elements required, and returns the index into the array, `STACK` or `CSTACK`, of the first element allocated. In the case of `ALLSTK`, the elements are integers. The function, `FSIZEOF`, should always be used to calculate the number of integers required for a given data type because the number of integers is machine dependent. **Never** do this calculation in your own code! Also, `ALLSTK` will round your request up to the nearest multiple of 2 so that double precision alignment will be maintained. In the case of `ALLCSTK`, the elements are characters. If you are allocating space for a character string array, pass `INICSTK` the product of the array length times the string lengths. The functions revise the stack pointers, `LSTUSD` or `LSTCUSD`, to indicate the allocation, and they check for stack overflow. Stack overflow results in the termination of the program. Negative arguments are not permitted to these functions. Note that common block predeclares this function for you.

31.1.2.3 `FRESTK` and `FRECSTK`

To free storage on the stack, the subroutines `FRESTK` or `FRECSTK` are used. `FRESTK` is used for the numeric data stack, and `FRECSTK` is used for the character string stack. They each take one argument, the number of elements to be freed. They decrement the stack pointers by their arguments to indicate the release of storage, and they check to see that the stack pointers have not become negative. If they have, an error message is output, and program execution stops. Both routines check that their arguments are non-negative. `FRESTK` will round its argument to the next multiple of 2.

31.1.2.4 `FSIZEOF`

The integer function, `FSIZEOF`, accepts two arguments, `TYPE` and `SIZE`, and returns the number of integers needed to store `SIZE` variables of data type, `TYPE`. The `TYPE` argument is a character string and may have one of the following values: '`REAL`', '`REAL*8`', '`DOUBLE`', '`INTEGER`', '`INTEGER*1`', '`INTEGER*2`', '`LOGICAL`', '`LOGICAL*1`', '`SHORT`', '`BYTE`', '`CHARACTER`', or '`COMPLEX`'. The lengths provided by `FSIZEOF` vary from machine to machine.

31.1.2.5 Other Stack Functions and Variables

The function, `PRINSTK`, may be used to print the current state of the stack. The debugging variable, `DBG_ALLSTK`, may be used to debug stack usage. The default setting of 1 results in the initialization of the stack to non-zero values. If `DBG_ALLSTK` is set to 2 or greater, all calls to the allocation and freeing routines will generate messages displaying the argument to each call.

31.1.2.6 Using Storage on the Stack

The main method for using the space allocated on the stack is as follows: Suppose we have a subroutine, `FOO`, which requires local storage to work. Suppose further that `FOO` takes two arguments, `A` and `B`, and needs two working arrays, `C` and `D`. The code in subroutine `FOO` will call `ALLSTK` twice;

once for each work array needed. Since C and D are names of the work arrays, the indices into the stack returned by ALLSTK are assigned to the variables C and D. A second subroutine, FOO1, is defined with four arguments, the original two and two arguments which are work arrays. In FOO1, C and D are defined as arrays. Once FOO has finished allocating the work storage, it calls FOO1 as follows:

```
          CALL FOO1(A,B,STACK(C),STACK(D))
```

FOO1 actually does the computational work for FOO. Once FOO1 completes, FOO then calls FRESTK to free the work storage used, and it returns having completed its task. In summary, referencing the work arrays is accomplished by using a subroutine call to map an array onto the stack.

A second method exists for accessing local storage on the stack. This involves using subroutines which access one element of an array at a time. They are not very useful for the stack, but they are discussed further in the section on data structures in the heap, see Section 31.3 [Dynamic Data Structures], page 283.

31.1.3 Stack Example 1 — Matrix Squaring

To illustrate the use of the stack, we show how the matrix squaring subroutine is programmed. The main program sets up the stack and calls the routine which is named MSQUAR.

```
      PROGRAM TSTMSQ
C
C     TESTS MSQUAR, A MATRIX SQUARING ROUTINE. WE READ THE MATRIX IN
C     FROM UNIT 5 AND OUTPUT ITS SQUARE ON UNIT 6. THE SIZE OF THE TEST
C     MATRIX, A, IS 10 BY 10.
C
#     include "stack.fcm"
C
      REAL A(10,10)
C
      CALL INISTK
      READ (5,100) ((A(I,J),J=1,10),I=1,10)
  100 FORMAT(10F8.0,9(/10F8.0))
      CALL MSQUAR(A,10)
      WRITE (6,200) ((A(I,J),J=1,10),I=1,10)
  200 FORMAT(' RESULTS OF CALLING MSQUAR -- THE MATRIX A:'/
     2       10(/1X,1P10G11.4))
      CALL PRINSTK(6)
      STOP
      END
      SUBROUTINE MSQUAR(A,N)
C
C     SQUARES A WHICH IS A REAL ARRAY ASSUMED TO BE N BY N. WE USE THE
C     STACK TO STORE THE SQUARED ARRAY.
C
      REAL A(N,N)
#     include "stack.fcm"
      INTEGER SQUARE
      INTEGER OLDLST
C
C     WE ALLOCATE SPACE FOR THE SQUARED ARRAY. NOTE THAT WE SAVE THE
C     VALUE OF LSTUSD UPON ENTERING THIS SUBROUTINE. THIS MAKES IT
```

```
C     EASIER TO FREE THE AMOUNT OF STORAGE ALLOCATED BY EVERY ALLSTK
C     CALL WHEN THE SUBROUTINE QUITS.
C
      OLDLST=LSTUSD
      SQUARE=ALLSTK(N*N)
      CALL MSQUA1(A,N,STACK(SQUARE))
      CALL FRESTK(LSTUSD-OLDLST)
      RETURN
      END
      SUBROUTINE MSQUA1(A,N,SQUARE)
C
C     MSQUA1 DOES THE JOB OF SQUARING THE MATRIX A. WE CAN NOW REFER TO
C     THE CONTENTS OF THE TEMPORARY ARRAY, SQUARE, DIRECTLY.
C
      REAL A(N,N),SQUARE(N,N)
C
      DO 1 I=1,N
        DO 1 J=1,N
          S=0.0
          DO 2 K=1,N
    2       S=S+A(I,K)*A(K,J)
    1     SQUARE(I,J)=S
C
C     NOW COPY SQUARE BACK INTO A
C
      DO 3 I=1,N
        DO 3 J=1,N
    3     A(I,J)=SQUARE(I,J)
      RETURN
      END
```

31.1.4 Stack Example 2 — JOINWD

In this example, we show the use of the character string stack. JOINWD is used to insert one character string, WD, onto the beginning of a second character string, ST. Stack space is required because we wish to avoid an overlapped copy. JOINWD first copies ST to local storage, copies WD to ST followed by space, and then copies the old value of ST back into ST. Note the use of a substring reference into CSTACK in order to provide COPYST the correct size of the string it is copying into.

```
      SUBROUTINE JOINWD(ST,STLEN,WD)
C
C     DOES NEARLY THE INVERSE OF NEXTWD, I.E. JOINS WD ONTO
C     THE BEGINNING OF ST.
C
      IMPLICIT INTEGER(A-Z)
      CHARACTER*(*) ST,WD
#     include "stack.fcm"
C
      STL = STLEN
      WDLEN = LEN(WD)
      COPY = ALLCSTK(STL)
      CALL COPYST(CSTACK(COPY)(1:STL),COPYLN,ST(1:STLEN))
      CALL COPYST(ST,STLEN,WD(1:WDLEN))
      IF (COPYLN .GT. 0) CALL ADDST(ST,STLEN,' ')
      CALL ADDST(ST,STLEN,CSTACK(COPY)(1:COPYLN))
      CALL FRECSTK(STL)
      RETURN
      END
```

31.2 Heaps

The stack is a very efficient means of providing dynamic storage for a great variety of uses. However, its limitation to local storage presents a problem when a subroutine must allocate storage for use by the routine which calls it. An example where the stack cannot be used is as follows: Suppose we wanted to write a routine, `NBO`, which finds all close non-bonded interactions, and suppose further that we wanted `NBO` to allocate the storage itself without any preset limitations. The stack cannot be used for this because the storage being allocated for the non-bonded list is not local; it must be used by the subroutine which calls `NBO`.

To circumvent this problem, we need a storage mechanism which allows completely arbitrary allocation of storage. Such flexibility is obtained with a heap.

A heap consists of an array and a list. The array is used for storing data. The list, referred to as the free list, keeps track of what areas in the heap are free. Before the heap is used, the free list is initialized to indicate that the entire heap is free. When space is required, the free list is searched for a space large enough to accommodate the request. The free list is then modified to indicate that less storage is available. To free storage, the freed area of storage must be put back on the list and recombined with adjacent free areas if possible.

As with the stacks, CONGEN provides two heaps, one for numeric data and one for character string data. The numeric data heap is managed entirely within Fortran, but the character string heap is really nothing more than the heap maintained by the C run time library routines `malloc` and `free`. There is a character string heap array, and the pointers returned by `malloc` are mapped into the address for the character string heap.

31.2.1 Implementation of the Heap for CONGEN

In CONGEN, the heaps are implemented using three common blocks. `HEAPCM` stores numerical information and the numeric heap, `SVHEAP` stores a copy of critical heap pointer for error detection, and `HEAP_ST` stores the initial address for the character string heap. All of the declarations are in the files, '`heap.fcm`' and '`heap.h`', and the appropriate file should be `#include`'d wherever the heaps are used. The real number array, `RHEAP` and the logical array, `QHEAP`, are equivalenced to the `HEAP` so references to other data types are simplified. (It should be noted that the use of these equivalences presumes that these data types have the same sizes as integers. This presumption may not always be true.)

31.2.2 Useful Subprograms

There are a number of subprograms which simplify the use of the heaps.

31.2.2.1 `INITHP`

To initialize the heap, `INITHP` is used. First, one sets HEAPSZ to the dimension of the heap, and then, INITHP is called with no arguments.

31.2.2.2 `ALLHP` and `ALLCHP`

To obtain storage on the heap, we use the integer function `ALLHP`. `ALLHP` is called with one argument, the amount of storage needed on the `HEAP` array, in units of `INTEGER`s. You should always use the

INTEGER function FSIZEOF, see Section 31.1 [Stacks], page 275, to determine the number of integers required for particular data type and number of elements. ALLHP then returns the index of the first element of the array allocated in the heap. This index shall be referred to as the base of the allocated storage.

ALLCHP works in a similar way to ALLHP except that it operates on the character string heap. The argument to ALLCHP is the number of characters to be allocated, and the return value is the index into CHEAP for the storage. Internally, ALLCHP is implemented using malloc.

31.2.2.3 FREHP and FRECHP

To return storage to the heap, one calls FREHP or FRECHP. Unlike FRESTK, FREHP and FRECHP require two arguments, a base and a length. FREHP and FRECHP return the storage demarked by the base and length to their respective free lists. If this storage can be recombined with adjacent free areas of storage, it is done.

31.2.2.4 Error Detection

ALLHP, ALLCHP, FREHP, and FRECHP make extensive checks on their arguments and on themselves. Since these two routines are vital to the operation of CONGEN, they are designed to catch many programming errors that can result from using the heap. In addition, there are debugging variables that can be used to improve the error detecting capabilities of the heap subprograms. See the source code for the usage of the debugging variables alloc and allhp.

31.2.2.5 PRINHP

To obtain a listing of the free storage areas available on the heap, PRINHP is available. It is called with one argument, a unit number to which the output is sent.

31.2.3 Using Storage on the Heap

Referencing the data in the heap can be done using the same techniques as described for the stack. However, a more sophisticated method of referencing data on the heap is the subject of the next section.

31.2.4 Heap Example — Unlimited Read of a File

To illustrate the use of a heap, consider the following example: Suppose we want a subroutine which reads in a list of card images from a file and stores them into an array on the heap. Assume that the number of cards which can be read is to be limited only by the space available on the heap.

To implement this, we assume that we can store 4 characters to a word and that our cards have 80 columns. The card array is then dimensioned (20,N). The subroutine which is called to get the card images is called RDCARD.

```
      PROGRAM TSTRDC
C
C     This program tests RDCARD. It initializes the HEAP, calls RDCARD
C     on unit 5, prints the card images, and then prints the HEAP.
C
#     include "heap.fcm"
      INTEGER CRDBAS,CARDSZ
C
      CALL INITHP
      CARDSZ = 80
      CALL RDCARD(CRDBAS,5,NCARD,CARDSZ)
      CALL PRTCRD(CHEAP(CRDBAS)(1:CARDSZ),NCARD)
      CALL PRINHP(6)
      STOP
      END
      SUBROUTINE RDCARD(BASE,UNIT,NCARD,CARDSZ)
C
C     Reads all the card images from UNIT into the HEAP. BASE is
C     returned giving the first word in the HEAP where the card images
C     are stored. NCARD returns the number of cards found.
C
      INTEGER BASE,UNIT,NCARD,START,END,CARDSZ
#     include "heap.fcm"
      LOGICAL EOF
      INTEGER INC
      PARAMETER (INC = 100)
C
C     Initially, we allocate space for INC cards. If this is exceeded
C     during the reading of the file, space for an additional INC cards
C     will be allocated, and the reading will continue.
C
      LIMIT=INC
      BASE = ALLCHP(LIMIT*CARDSZ)
      NCARD=0
C
C     RDCAR1 actually reads the file, putting card images into the HEAP.
C
      REPEAT UNTIL (EOF)
      CALL RDCAR1(CHEAP(BASE)(1:CARDSZ),NCARD,LIMIT,UNIT,EOF)
      UNLESS (EOF)
C
C     At this point, we know that RDCAR1 ran out of space. We allocate a
C     array larger in size and copy the cards already obtained into the
C     new array, and then free the old array.
C
      NEWBAS = ALLCHP(CARDSZ*(LIMIT+INC))
      CALL COPYCA(CHEAP(NEWBAS)(1:CARDSZ),LIMIT+INC,
     2            CHEAP(BASE)(1:CARDSZ),LIMIT)
      CALL FRECHP(BASE,CARDSZ*LIMIT)
      LIMIT = LIMIT+INC
      BASE = NEWBAS
      FIN
      FIN
      END
      SUBROUTINE RDCAR1(CARDS,NCARD,LIMIT,UNIT,EOF)
C
C     Reads cards from UNIT into CARDS, setting EOF if end of file is
C     found. If the routine returns with EOF not set, it means that
C     space was exhausted in the card array.
C
      CHARACTER*(*) CARDS(LIMIT)
      INTEGER UNIT
      LOGICAL EOF
```

```
C
      EOF = .FALSE.
      REPEAT WHILE (NCARD .LT. LIMIT)
      NCARD=NCARD+1
      READ (UNIT,100,END=99) CARDS(NCARD)
  100 FORMAT(A)
      FIN
      RETURN
C
C     Come here on end of file. We must decrement NCARD by one because
C     the READ statement did not add another card image to the array.
C
   99 NCARD = NCARD-1
      EOF = .TRUE.
      RETURN
      END
      SUBROUTINE PRTCRD(CARDS,NCARD)
C
C     Prints the card images in CARDS onto unit 6, the printer.
C
      CHARACTER*(*) CARDS(*)
C
      WRITE (6,200) NCARD
  200 FORMAT('0OUTPUT OF ',I5,' CARDS:')
      DO (I=1,NCARD)
      WRITE (6,201) CARDS(I)
  201 FORMAT(1X,A)
      FIN
      RETURN
      END
      SUBROUTINE COPYCA(ST1A,NST1A,ST2A,NST2A)
C
C     Copies strings from ST2A into ST1A. The lengths of the arrays are
C     given by NST2A and NST1A, respectively.
C
      IMPLICIT INTEGER(A-Z)
      CHARACTER*(*) ST1A(*),ST2A(*)
      INTEGER NST1A,NST2A
C
      DO (I = 1,MIN(NST1A,NST2A))
      ST1A(I) = ST2A(I)
      FIN
      RETURN
      END
```

31.3 Data Structures on the Heap in CONGEN

Fortran, the language that CONGEN is implemented in, suffers from a lack of flexibility with respect to the storage of data. The structuring of data is limited to scalars, complex numbers, and arrays. For many programming tasks, there is a need for more complicated structures for holding data. For example, in CONGEN, the protein structure file, the PSF, consists of a number of arrays and scalars that collectively describe the structure of a macromolecule. It would very useful to be able to refer to the PSF as a whole as opposed to referring to all of its constituents.

This inability to refer to a number of related scalars and arrays by one name results in a number of problems. Passing the information to a subroutine requires either using a large number of parameters or common blocks.

Using large numbers of parameters makes subroutine calls very bulky and increases the chances of errors resulting from the mistyping of a parameter. On the order of 100 parameters are required to call the analysis subroutine.

Common blocks could be used to pass information to a subroutine. However, these require allocating all needed space at compile time. Also, having more than one instance of a data structure is difficult. For example, the analysis facility requires a second PSF, coordinate set, and other data structures. Without solving this data structure problem, we need to establish another huge set of scalars and arrays. Further, as the program is changed, this second set would have been modified as well which would have a great potential for programming errors. These problems required a better method be introduced to handle the use of data structures in Fortran.

Having decided to implement a more succinct method for using data structures, we must consider other design criteria which would be useful to meet. First of all, the elements of each data structure must convenient to access, and the clarity of the program must not be unduly restricted. Next, since we have a mechanism for dynamic storage allocation, data structures should use only as much space as they need and should be able to grow. Third, the sharing of readonly data is desirable.

Given these design goals, how are they implemented for use in CONGEN? First, the heaps are ideally suited for this task as it allows completely arbitrary use of storage. Scalars are assumed to require one word of storage on the heap. Arrays will take as many words as their dimension and data type requires. Character strings are stored in the character string heap.

To keep track of the location and data types of elements in a data structure, three arrays; the base, length, and type arrays; are used. The base array stores the index of the first word or character of each element in the data structure. The length array gives either the total number of integers required for the element or the total number of characters, depending on the data type. The type array specifies the data type of the element. If the type array contains a positive number, this indicates that the element is a character string, and the number in the type array specifies the length of the character strings. Otherwise, if the number is equal to the parameter, `DT_NUMERIC`, then the element is numeric. Any other values are illegal. Multiple instances of a data structure require multiple sets of arrays. The naming scheme for the base, length, and type arrays is to prefix a ‘`B`’, ‘`L`’, or an ‘`T`’, respectively, in front of the data structure name.

To identify the word in the base array which corresponds to an element in the data structure, we use a common block which is named for the data structure. If the data structure contains n elements, the common block of indices contains $n+1$ variables. The first variable gives the size of the base array needed to hold one instance of the data structure. The remaining variables all have the same name as the element in the data structure. Before proceeding any further, an example is required.

31.3.1 Example – The Non-bonded List Data Structure

Consider a simplified and slightly contrived version of the data structure for storing the non-bonded list. Two arrays; `JNB` and `INBLO`; six scalars; `NNNB`, `CTONNB`, `CTOFNB`, `CUTNB`, `WMIN`, and `NBOPT`; and one character string, `NAME` are needed. We will use the name, `NBND`, for the data structure. The base, length, and type arrays for this data structure are called `BNBND`, `LNBND`, and `TNBND`, respectively, and the following declarations in a subroutine would suffice to use parts of the non-bonded list for a calculation:

```
       IMPLICIT INTEGER(A-Z)
#      include "heap.fcm"
       COMMON /INBND/ SNBND,JNB,INBLO,NNNB,CTONNB,CTOFNB,CUTNB,
      2               WMIN,NBOPT,NAME
       INTEGER BNBND(SNBND)
```

To make reference to the scalar `NBOPT`, one would write `HEAP(BNBND(NBOPT))`. To make reference to the character string, `NAME`, one would write `CHEAP(BNBND(NAME))(1:TNBND(NAME))`. The substring reference serves to specify a legitimate length for the name, and would be accessible in a called subroutine as the length of the string.

Three points must be noted before we continue. First, we can store any type of data in these data structures, provided that the numeric and character data are identified and separated. All we must know is how much space they require, which is provided by the `FSIZEOF` function, see Section 31.1 [Stacks], page 275. For example, `DOUBLE PRECISION` variables require two integers per variable on most machines, and one integer on the Cray.

Second, we ensure that the index common blocks are the same from one subroutine to the next by using an `#include` statement which is processed by the Fortran C preprocessor, see Section 30.3 [Fcpp], page 261. An `#include` statement takes a file name and inserts the contents of the file in the place of the `#include` statement. By setting up files which contain the common block definitions and using `#include` statements exclusively for getting the common block, we can ensure that the common blocks are the same.

Third, in general, one cannot access the data directly. The cases where direct access is possible is for numeric data whose size is the same as an `INTEGER`, such as `REAL`s and `LOGICAL`s. Since the heap is declared as `INTEGER`, we make a direct reference. Single precision real scalars can be accessed directly using `RHEAP`, and logicals can be reference using `QHEAP`. References to any other parts of the data structure can be done using two general techniques.

The first of these reference techniques is to call a subroutine with the desired element of the data structure as a parameter. In the subroutine, one declares the parameter for what it is, and one can make references as desired. For example, if we wish to access the `JNB` array in the non-bonded data structure, we can make a subroutine call as follows:

```
       CALL SUB(HEAP(BNBND(JNB)))
```

and declare in the subroutine

```
       SUBROUTINE SUB(JNB)
       INTEGER JNB(1)
```

and make references to any element of the `JNB` array. This is the same method as described for stacks. The overhead involved with this scheme is that one must create a subroutine for the references.

The second technique which has a much higher overhead, but doesn't require a new subroutine is to use referencing and storage functions which have been written for CONGEN and which are found in the source file '`array.flx`'. The referencing functions take an array and indices into it and return the value of the specified element. For example, the function `REFI` accepts two arguments, an integer array, `A`, and an index, `I`. `REFI` returns `A(I)`. The storage functions take an array, indices, and a value and store the value into specified element. For example, the subroutine STORI takes an integer array, `A`; its length, `N`; and index, `I`; and a value, `VALUE`, and executes the following statement: `A(I)=VALUE`. The higher overhead of this method results from the fact that a subroutine call is required for each array reference.

31.3.2 Useful Subprograms

Next, what subroutines and functions are provided to assist in the use of these data structures? There are many which cover most of the needs of CONGEN.

31.3.2.1 `STRTDT` and `SETUPI`

The subroutine `STRTDT` is used to initialize the common blocks. Whenever a new data structure is defined or an old one is changed in size, `STRTDT` must be changed to accommodate this. `STRTDT` calls `SETUPI`.

The subroutine, `SETUPI`, is used to initialize the index common blocks. It depends on being able to equivalence the integer variables in the index common block to an integer array by using a subroutine call.

31.3.2.2 `INITDT`

The subroutine `INITDT` is used to initialize a base array. Before a base array can be used, it must be initialized to the free state. The free state means that all bases are set a value which cannot point into the heap. This value, known as the free value, is a sufficient large negative number such that any references using it will cause an access violation which is useful in catching programming errors before they do subtle damage.

31.3.2.3 `ALLDT`

The subroutine `ALLDT` is used to allocate space for a data structure on the heaps. It is called with filled length and type arrays and a free base array. The contents of the length array is summed to get the space requirements for the data structure. `ALLHP` and `ALLCHP` are then called to get space on the heaps for the data structure. Finally, the bases are all set to point to their part of data structure. Lengths of zero are permitted. In these cases the bases for the element will be left as the free value. This will catch any illegal references.

31.3.2.4 `FREEDT`

The subroutine `FREEDT` is used to free a data structure. The data structure must be allocated before it can be freed. The base array is set to the free value.

31.3.2.5 `USEDDT`

The logical function `USEDDT` is used to determine if a base array is free or is pointing to a data structure. It returns `.TRUE.` if the base array passed to it is in use.

31.3.2.6 `RESIZE`

`RESIZE` is used to change the size of any of the elements in a data structure. It is called with a base, length, and type array and another trio of arrays which give elements to be changed and the new sizes. Note that only the lengths of character string elements can be changed, data types of elements may not be. `RESIZE` allocates space for a copy of the data structure with new sizes; copies the old data into the new structure; and then frees the old data structure.

31.3.2.7 `READDT` and `WRITDT`

`READDT` and `WRITDT` may be used to write a data structure in binary form to a file. The trio of arrays must be specified along with a Fortran unit number, `HDR` and `ICNTRL` information, and a title. `READDT` will read what `WRITDT` creates.

31.3.2.8 `DUPLDT`

`DUPLDT` is used to duplicate a data structure. It simply makes a complete copy of the arrays and the data stored in the heaps.

31.3.3 Sharing Data and `ASGNDT`

With the subroutines used to implement these data structures explained, we can return to a discussion of other implementation details. The next issue is the question of sharing data. Because the base arrays are pointers, there is nothing to prevent two base arrays from pointing to the same data. Obviously, this can lead to great problems if this occurs unintentionally, but one can save space and underscore the identity of data if we allow sharing of read-only data.

Implementing sharing is very easy with the above scheme. We reserve the first element of the base array to be a pointer to the reference count for a particular instance of a data structure. When the data structure is allocated, the reference count is set to 1. If we assign another data structure to this instance by copying its bases, the reference count is incremented by 1. If we free the data structure, the reference count is decremented by one. When the reference count drops to zero, the space on the heap is freed. The use of the first element of the base arrays has an additional benefit because the length of the reference count is 1 word. This fact ensures that the first word in the base array cannot have a free value when the base array is in use.

The subroutine, `ASNGDT`, implements the assignment of data structures which are to shared. It will take a base and length array and copies them to another base and length array. The reference count for the data structure will be incremented by one.

31.3.4 Switching to Data Structures on the Heap

The final detail to be discussed is the interface between this scheme for manipulating data structures and the usual scheme. Currently, the analysis subroutine and non-bonded list uses this more concise means for handling data structures; the rest of CONGEN works with the separate arrays and scalars. When the analysis subroutine is invoked, the data structures must be built. Copying subroutines exist for the five data structures passed to the analysis subroutine. These copying subroutines will take the arrays and scalars which comprise a data structure and build a data structure on the heap for it. Once the copying operation is complete, the actual analysis subroutine can be called and it can freely use the data structures existing on the heap.

The copying operation presents a naming problem in that the indices kept in the index common blocks have the same names as the arrays and scalars. To circumvent this problem, a second set of index common blocks is kept for all data structures which are not isolated in the analysis subroutine. This second set of index common blocks have the same common block names as the first, but the names of the indices are prefixed with ‘`I`’.

A good example of the use of these data structures is the subroutine `NBONDS` and some of its subsidiary routines in the file, ‘`nbonds.flx`’.

31.4 General Applicability of These Schemes

One final aspect of these storage schemes must be considered — when should they be used? It is clear that for new work in CONGEN, C should be used for programming because it provides all these facilities in a straightforward manner. (CHARMM and CONGEN would have been more advanced had C be used from the beginning.) However, if new Fortran code must be integrated or maintained, then these schemes are applicable. The stack and the data structure storage on the heap were developed primarily to deal with an enormous number of arrays whose sizes must be able to change. The other design criteria came later. It should be obvious that one pays a price for these schemes, and one must determine whether the benefits are worth the cost.

31.4.1 Disadvantages

One disadvantage of these methods is that they are unfamiliar, and therefore, they take time to learn. Unfortunately, a subjective judgment is required to estimate how large a disadvantage this is. In my opinion, this chapter will enable an interested person to learn this material in a day or two. The subroutines to use these schemes are sufficiently comprehensive to permit effective use of these schemes without additional programming. Further, the ideas presented in this document are worth learning as they are valuable outside the context of CONGEN. Besides its application for local storage, the stack is useful for problems which require backtracking or which can be solved recursively. The heap is useful for problems which involve management of complex lists.

A second disadvantage of these schemes is that they are more cumbersome than the techniques they replace. There is no arguing this point. However, when used with the conventions described here, clarity is not sacrificed.

31.4.2 Advantages

Let us now consider the advantages. We shall confine ourselves to the analysis facility where they are used extensively.

First, as the size of a system being analyzed grows to the limits of what is supported, one need only change the size of the stack and the heap to accommodate a bigger system. When a subroutine has as many arrays as the analysis facility has, this is vital.

Next, although references to parts of a data structure are more cumbersome, referencing a whole structure is much more concise. The subroutines in the analysis facility illustrates the compactness of the subroutine call despite the large amount of information that is being passed. Further, the inclusion of the comparison data structures is trivial; we merely have to create another set of base and length arrays.

Thirdly, the data structures can be referred to as a unit. This aids the job of programming because now we can deal with bigger units of information without having to worry about the details. These data structures on the heap allow a programmer to ignore their contents until they are ready to be used. This can simplify the task of modifying a program the size of CONGEN.

Appendix A Residue Topology Files

This appendix contains a description of the various topology files available for CONGEN users. For CONGEN version 2 and beyond, we anticipate providing only one topology file in binary form, but all older and obsolete files will be provided in source form, although they may not be readable without some editing. All topology files are stored in '`CGDATA`'. The file names used consists of an alphabetic part followed by a number, e.g. `RTOPH`. There are two copies of each file; one with extension, '`.INP`', which is a character files used as an command file to generate the binary file, with extension, '`.MOD`'. The '`.INP`' is meant for human eyes; the '`.MOD`' files are meant for CONGEN to read. The numeric part of each name is its version number. In general, one should use the highest version number of a file.

Five types of residue topology files are currently available. Three contain the twenty amino acids, but with different representations. '`RTOP`*n*' holds extended atom topologies; '`RTOPH`*n*' holds explicit hydrogen residue topologies; and '`RTOPALLH`*n*' holds all hydrogen residue topologies. The fourth topology file type, '`RTOPDNA`*n*', is for DNA residues and the phosphates connecting them. These DNA files have not been used very much, so results obtained using them must be treated with especial care. Also, the DNA and protein files cannot be used together because of differences in the selection of atom type codes.

The fifth type of topology file is the AMBER topology file based on the AMBER potential[1]. At the present time, only the united atom topology file is available.

The remainder of this section lists the comments in the topology file themselves, along with some further explanations of the rationale behind particular versions.

All the topology file except for the AMBER files were either in CHARMM version 16 or were derived from them.

A.1 Extended Atom Topology Files

A.1.1 RTOP8

'`RTOP8`' is a modification of '`RTOP7`' that has all the atoms specified in PARAM5. In addition, all atoms are typed as hydrogen bond donors and acceptor for use by the `GENERATE` command in the RTF reader, see Section 3.1.5 [RTF File Formats], page 27. The linkages between residues are corrected as required in Congen v.2, and D amino acids are added.

Title in the topology file:

```
*   RESIDUE TOPOLOGY FILE FOR EXTENDED ATOMS -- VERSION 8
*   CONVERTED TO NEW FORMAT -- 10/29/81
*   HBOND 2nd donor antecdants CA for peptides CD for arg he 30-Dec-1981
*   PCA and PC added by Robert Bruccoleri, 2/20/91
*   Adapted for symbolic generation, Congen v2, REB 30-Jul-1991.
*   D amino acids added, REB 9-Sep-1991.
```

A.1.2 RTOP7

'`RTOP7`' is '`RTOP6`' converted to free field input format.

Title in the run:

[1] Scott J. Weiner, Peter A. Kollman, Dzung T. Nguyen and David A. Case, "An All Atom Force Field for Simulations of Proteins and Nucleic Acids", *J. Comput. Chem* **7**, 230-252 (1986)

```
*   RESIDUE TOPOLOGY FILE FOR EXTENDED ATOMS -- VERSION 7
*   CONVERTED TO NEW FORMAT 10/29/81 -- BRB
*   HBOND 2nd antecedants converted to CA for peptides and CD arg he
```

Title in the topology file:

```
*   RESIDUE TOPOLOGY FILE FOR EXTENDED ATOMS -- VERSION 7
*   CONVERTED TO NEW FORMAT -- 10/29/81
*   HBOND 2nd donor antecdants CA for peptides CD for arg he 30-Dec-1981
```

A.1.3 RTOP6

New set of atom names for the bilder information was added by Bernie. Values for setting the position of the CB carbons added.

Comments taken from the run itself:

```
*   RESIDUE TOPOLOGY FILE FOR EXTENDED ATOMS -- VERSION 5
*   DEFINED WITH 19 ATOM TYPES -- RESIDUES FOR AMINO ACIDS, HEME AND
*   WATER -- INCLUDES IMPROPER TORSIONS -- COMPILED BY WVG, BDO AND
*   SNS -- 3/15/79
*   PROTEIN BUILDER INFORMATION ADDED 11/24/79 -- BDO
```

Title actually in the file:

```
*   RESIDUE TOPOLOGY FILE FOR EXTENDED ATOMS -- VERSION 3
*   DEFINED WITH 19 ATOM TYPES -- RESIDUES FOR AMINO ACIDS, HEME AND
*   WATER -- INCLUDES IMPROPER TORSIONS AND HYDROGEN DONOR ANTECEDENT ARRAYS
*   COMPILED BY WVG, BDO AND SNS -- 3/15/79
*   LAST UPDATE -- 9/3/79 -- BRB
```

A.1.4 RTOP5

Comments taken from the file itself:

```
*   RESIDUE TOPOLOGY FILE FOR EXTENDED ATOMS -- VERSION 5
*   DEFINED WITH 19 ATOM TYPES -- RESIDUES FOR AMINO ACIDS, HEME AND
*   WATER -- INCLUDES IMPROPER TORSIONS -- COMPILED BY WVG, BDO AND
*   SNS -- 3/15/79
*   PROTEIN BUILDER INFORMATION ADDED 11/24/79 -- BDO
```

The title in the binary file:

```
*   RESIDUE TOPOLOGY FILE FOR EXTENDED ATOMS -- VERSION 3
*   DEFINED WITH 19 ATOM TYPES -- RESIDUES FOR AMINO ACIDS, HEME AND
*   WATER -- INCLUDES IMPROPER TORSIONS AND HYDROGEN DONOR ANTECEDENT ARRAYS
*   COMPILED BY WVG, BDO AND SNS -- 3/15/79
*   LAST UPDATE -- 4/3/79 -- BDO AND SNS
```

Note: the bilder information referred to above is the atoms involved for each internal coordinate. The five real numbers required are blank.

A.1.5 RTOP3 (Unreadable)

Comments taken from file itself:

```
*   RESIDUE TOPOLOGY FILE FOR EXTENDED ATOMS -- VERSION 3
*   DEFINED WITH 19 ATOM TYPES -- RESIDUES FOR AMINO ACIDS, HEME AND
*   WATER -- INCLUDES IMPROPER TORSIONS AND HYDROGEN DONOR ANTECEDENT ARRAYS
*   COMPILED BY WVG, BDO AND SNS -- 3/15/79
*   LAST UPDATE -- 4/3/79 -- BDO AND SNS
```

A.1.6 RTOP1 (Unreadable)

Comments taken from the file itself:

```
*   RESIDUE TOPOLOGY FILE USING EXTENDED ATOMS -- VERSION 1
*   BRG TOPOLOGY FILE FOR PTI IS A SUBSET OF THIS FILE -- NO IMPROPER
*   TORSIONS ARE INCLUDED -- LAST UPDATE 4/10/79 -- BDO AND SNS
```

A.2 Explicit Hydrogen Topology Files

A.2.1 RTOPH8

'`RTOPH8`' is basically '`RTOPH7`' converted for symbolic generation. D amino acids are also included.

Title in run:

```
*   RESIDUE TOPOLOGY FILE FOR PROTEINS USING EXPLICIT HYDROGEN
*   ATOMS -- VERSION 7 -- BRB -- 10/24/81
*   DEVELOPMENTAL VERSION FOR CHARMM_16
*   RESIDUE 'FRM' AND 'ACE' DELETED  - BRB 10/24/81
*   BUILDER INFORMATION GIVEN FOR ALL EXPLICIT HYDROGENS - BRB 10/24/81
*   CONVERTED TO NEW FORMAT - BRB 10/26/81
*   H bond second antecedants changed to fix ARG and cis peptides DJS 12/26/81
```

Title in file:

```
*   RESIDUE TOPOLOGY FILE FOR PROTEINS USING EXPLICIT HYDROGEN
*   ATOMS -- VERSION 7 -- BRB -- 10/20/81
*   BUILDER INFORMATION CORRECT TO DETERMINE ALL EXPLICIT HYDROGENS
*   H bond second antecdants set to CA for peptides and CD for ARG NE
```

A.3 Explicit Hydrogen Topology Files

A.3.1 RTOPH7

'`RTOPH7`' is basically '`RTOPH6`' converted to free field input.

Title in run:

```
*   RESIDUE TOPOLOGY FILE FOR PROTEINS USING EXPLICIT HYDROGEN
*   ATOMS -- VERSION 7 -- BRB -- 10/24/81
*   DEVELOPMENTAL VERSION FOR CHARMM_16
*   RESIDUE 'FRM' AND 'ACE' DELETED  - BRB 10/24/81
*   BUILDER INFORMATION GIVEN FOR ALL EXPLICIT HYDROGENS - BRB 10/24/81
*   CONVERTED TO NEW FORMAT - BRB 10/26/81
*   H bond second antecedants changed to fix ARG and cis peptides DJS 12/26/81
```

Title in file:

```
*   RESIDUE TOPOLOGY FILE FOR PROTEINS USING EXPLICIT HYDROGEN
*   ATOMS -- VERSION 7 -- BRB -- 10/20/81
*   BUILDER INFORMATION CORRECT TO DETERMINE ALL EXPLICIT HYDROGENS
*   H bond second antecdants set to CA for peptides and CD for ARG NE
```

A.3.2 RTOPH6

The changes that have been made to date are the inclusion of two additional improper dihedrals spanning the rings of tryptophan to distribute the out of plane bending forces more evenly. When used with the '`PARAM1`' parameters, see Section B.1 [PARAM1], page 299, this gives a correct out of plane skeletal vibration for tryptophan at between 400 and 500 cm-1 with relatively few high frequency modes induced by coupling of improper dihedral terms.

A second change is the repair of an error in the angles of the heme residue. Two angles were transposed in the second and third atoms resulting in incorrect specification of the desired angle. The old angles were 6-9-10 and 3-9-10. The correct angles are 6-10-9 and 3-10-9. This change results in a lower angle energy for the heme, but little change in geometry.

A third set of changes have been made in the builder information sections of the aromatic rings. The dihedral angle values have been set so that more or less correct rings will be built (the old values would result in figure eights and other such oddities).

A.3.3 RTOPH5A

Title in the topology file itself:

```
*   RESIDUE TOPOLOGY FILE FOR PROTEINS USING EXPLICIT HYDROGEN
*   ATOMS -- VERSION 5 -- BRB -- 9/3/80
*   BILDER INFORMATION INCLUDED FOR ALL ATOMS
```

Numbers for all the bilder values are present. No difference in other information with '`RTOPH5`' or '`RTOPH4`'.

A.3.4 RTOPH5

Title in the topology file itself:

```
*   RESIDUE TOPOLOGY FILE FOR PROTEINS USING EXPLICIT HYDROGEN
*   ATOMS -- VERSION 4 -- SNS AND BDO -- 1/24/80
*   BILDER INFORMATION ADDED - WATER CHARGES CHANGED
*   IMPROPER TORSIONS AND HYDROGEN DONOR ANTECEDENTS ADDED
```

This file differs from '`RTOPH4`' in that backbone atoms needed for building have been added. No numbers are specified.

A.3.5 RTOPH4

Title in the topology file itself:

```
*   RESIDUE TOPOLOGY FILE FOR PROTEINS USING EXPLICIT HYDROGEN
*   ATOMS -- VERSION 4 -- SNS AND BDO -- 1/24/80
*   BILDER INFORMATION ADDED - WATER CHARGES CHANGED
*   IMPROPER TORSIONS AND HYDROGEN DONOR ANTECEDENTS ADDED
```

The `BILDER` information added is just blank lines so that the RTF reading routine reads the right number of records.

A.3.6 RTOPH2 (Unreadable)

Title in the topology file itself:

```
*   RESIDUE TOPOLOGY FILE FOR PROTEINS USING EXPLICIT HYDROGEN
*   ATOMS -- VERSION 2 -- SNS AND BDO -- 4/4/79
*   IMPROPER TORSIONS AND HYDROGEN DONOR ANTECEDENTS ADDED
```

A.4 All Hydrogen Topology Files

A.4.1 RTOPALLH7

This file is a modification of '`RTOPALLH6`' where all of the bilder records have been inserted and checked. This file was modified for symbolic generation and to include D-amino acids. Earlier Bugs were corrected. These modifications were made by Donna Bassolino and Bob Bruccoleri.

Run title:

```
* rtf creation run for allh
*
!
! The charges in this file were set up using the experimental formamide
! and N-methyl acetamide dipole moments of 3.7 D through the following
! procedure.
!       1 - the C=O and N-H2 of formamide were required to be neutral
!       2 - the HA charge was arbitrarily set to zero (various Mulliken
!               population analyses give it small magnitude and variable
!               sign).
!       3 - The above conditions and the direction of the formamide dipole
!               competely determine the formamide charges
!       4 - The CA peptide charge was arbitrarily set to 0.1 (various
!               Mulliken population analysis give it 0.0 to 0.12 charges).
!       5 - The peptide charges were obtained by keeping the same C=O
!               and HN charges.  This requires a readjustment of the
!               N charge to maintain neutrality.
!
! The resulting set of charges gives close to the same dipole moment
! when applied to N-methyl acetamide, but a somewhat different
! direction.
!
read rtf card
* Residue Topology File for Proteins Using All Atom Hydrogens (ALLH)
* Cambridge notation for atom names
* atom type 3    ha  added for aliphatic hydrogens
* atom type 16   ct  added for tetrahedral carbons
* Impropers rearranged to match rtoph7 for gaunido & sidechain amide n
* 2nd donor antecedants converted to CA for peptides CD for arg HE
* Amide charges from Hayes and Kollman JACS 98:3335,7811 (1796)
* HA charges from Wiberg and Wendoloski J. Comp. Chem. 2:53 (1981)
* IC's added, and file checked, Donna Bassolino, March 1990
* Adapted for symbolic generation, Robert Bruccoleri, July 1991
```

Title in the file:

```
* Residue Topology File for Proteins Using All Atom Hydrogens (ALLH)
* Cambridge notation for atom names
* 2nd donor antecedants converted to CA for peptides CD for arg HE
* Amide charges from Hayes and Kollman JACS 98:7811 (1796)
* HA charges from Wiberg and Wendoloski J. Comp. Chem. 2:53 (1981)
* IC's added, and file checked, Donna Bassolino, March 1990
* Adapted for symbolic generation, Robert Bruccoleri, July 1991
* Redundent IC's removed for methyl groups Donna Bassolino Aug 1991
```

A.5 All Hydrogen Topology Files

A.5.1 RTOPALLH6

This file is a modification of 'RTOPALLH5' where all of the bilder records have been inserted and checked. These modifications were made by Donna Bassolino.

Run title:

```
* rtf creation run for allh
*
!
! The charges in this file were set up using the experimental formamide
! and N-methyl acetamide dipole moments of 3.7 D through the following
! procedure.
!       1 - the C=O and N-H2 of formamide were required to be neutral
!       2 - the HA charge was arbitrarily set to zero (various Mulliken
!               population analyses give it small magnitude and variable
!               sign).
!       3 - The above conditions and the direction of the formamide dipole
!               competely determine the formamide charges
!       4 - The CA peptide charge was arbitrarily set to 0.1 (various
!               Mulliken population analysis give it 0.0 to 0.12 charges).
!       5 - The peptide charges were obtained by keeping the same C=O
!               and HN charges.  This requires a readjustment of the
!               N charge to maintain neutrality.
!
! The resulting set of charges gives close to the same dipole moment
! when applied to N-methyl acetamide, but a somewhat different
! direction.
!
read rtf card
* Residue Topology File for Proteins Using All Atom Hydrogens (ALLH)
* Cambridge notation for atom names
* atom type 3    ha  added for aliphatic hydrogens
* atom type 16   ct  added for tetrahedral carbons
* Impropers rearranged to match rtoph7 for gaunido & sidechain amide n
* 2nd donor antecedants converted to CA for peptides CD for arg HE
* Amide charges from Hayes and Kollman JACS 98:3335,7811 (1796)
* HA charges from Wiberg and Wendoloski J. Comp. Chem. 2:53 (1981)
* IC's added, and file checked, Donna Bassolino, March 1990
*
```

Title in the file:

```
* Residue Topology File for Proteins Using All Atom Hydrogens (ALLH)
* Cambridge notation for atom names
* atom type 3    ha  added for aliphatic hydrogens
* atom type 16   ct  added for tetrahedral carbons
* Impropers rearranged to match rtoph7 for gaunido & sidechain amide n
* 2nd donor antecedants converted to CA for peptides CD for arg HE
* Amide charges from Hayes and Kollman JACS 98:7811 (1796)
* HA charges from Wiberg and Wendoloski J. Comp. Chem. 2:53 (1981)
* IC's added, and file checked, Donna Bassolino, March 1990
```

A.5.2 RTOPALLH5

This file is 'RTOPALLH4' converted to the new format.

Title and comments from the run:

```
* Residue Topology File for Proteins Using All Atom Hydrogens (ALLH)
* Cambridge notation for atom names
* atom type 3    ha  added for aliphatic hydrogens
* atom type 16   ct  added for tetrahedral carbons
* Impropers rearranged to match rtoph7 for gaunido & sidechain amide n
* 2nd donor antecedants converted to CA for peptides CD for arg HE
* Amide charges from Hayes and Kollman JACS 98:3335,7811 (1796)
* HA charges from Wiberg and Wendoloski J. Comp. Chem. 2:53 (1981)
*
!
! The charges in this file were set up using the experimental formamide
! and N-methyl acetamide dipole moments of 3.7 D through the following
! proceedure.
!       1 - the C=O and N-H2 of formamide were required to be neutral
!       2 - the HA charge was arbitrarily set to zero (various Mulliken
!               population analyses give it small magnitude and variable
!               sign).
!       3 - The above conditions and the direction of the formamide dipole
!               competely determine the formamide charges
!       4 - The CA peptide charge was arbitrarily set to 0.1 (various
!               Mulliken population analysis give it 0.0 to 0.12 charges).
!       5 - The peptide charges were obtained by keeping the same C=O
!               and HN charges.  This requires a readjustment of the
!               N charge to maintain neutrality.
!
! The resulting set of charges gives close to the same dipole moment
! when applied to N-methyl acetamide, but a somewhat different
! direction.
```

Title in the file:

```
* Residue Topology File for Proteins Using All Atom Hydrogens (ALLH)
* Cambridge notation for atom names
* atom type 3    ha  added for aliphatic hydrogens
* atom type 16   ct  added for tetrahedral carbons
* Impropers rearranged to match rtoph7 for gaunido & sidechain amide n
* 2nd donor antecedants converted to CA for peptides CD for arg HE
* Amide charges from Hayes and Kollman JACS 98:7811 (1796)
* HA charges from Wiberg and Wendoloski J. Comp. Chem. 2:53 (1981)
```

A.5.3 RTOPALLH4

This file is the same as 'RTOPALLH3' except it has builder information for alanine added.

Title from the run:

```
*   RESIDUE TOPOLOGY FILE FOR PROTEINS USING ALL ATOM HYDROGENS
*   CAMBRIDGE NOTATION FOR ATOM NAMES
*   ATOM TYPE 3    HA  ADDED FOR ALIPHATIC HYDROGENS
*   ATOM TYPE 16   CT  ADDED FOR TETRAHEDRAL CARBONS
*   BACKBONE DIHEDRALS AT 1,2,3, N IMPROPER AT 1
*   DIHEDRALS REPLACED BY IMPROPERS FOR GAUNIDO & SIDECHAIN AMIDE N
*   IMPROPER TORSIONS AND HYDROGEN DONOR ANTECEDENTS ADDED
*   IMPROPER DIHEDRALS INCLUDED FOR SP2 PLANAR ATOMS BUT NOT CA
*   DIHEDRAL AND NONBONDED EXCLUSION LISTS SORTED FOR ALL AA
```

Title in the file:

```
*   RESIDUE TOPOLOGY FILE FOR PROTEINS USING ALL ATOM HYDROGENS
*   CAMBRIDGE NOTATION FOR ATOM NAMES
*   ATOM TYPE 3    HA  ADDED FOR ALIPHATIC HYDROGENS
*   ATOM TYPE 16   CT  ADDED FOR TETRAHEDRAL CARBONS
*   DIHEDRALS REPLACED BY IMPROPERS FOR GAUNIDO & SIDECHAIN AMIDE N
*   IMPROPER TORSIONS AND HYDROGEN DONOR ANTECEDENTS ADDED
*   IMPROPER DIHEDRALS INCLUDED FOR SP2 PLANAR ATOMS BUT NOT CA
*   PHI,PSI,OMEGA DIHEDRALS AND NONBONDED EXCLUSION LISTS SORTED
*   3/26/80 -- DAVID J. STATES
```

A.5.4 RTOPALLH3

Title taken from the run which makes the file.

```
*   RESIDUE TOPOLOGY FILE FOR PROTEINS USING ALL ATOM HYDROGENS
*   CAMBRIDGE NOTATION FOR ATOM NAMES
*   ATOM TYPE 3    HA  ADDED FOR ALIPHATIC HYDROGENS
*   ATOM TYPE 16   CT  ADDED FOR TETRAHEDRAL CARBONS
*   BACKBONE DIHEDRALS AT 1,2,3, N IMPROPER AT 1
*   DIHEDRALS REPLACED BY IMPROPERS FOR GAUNIDO & SIDECHAIN AMIDE N
*   IMPROPER TORSIONS AND HYDROGEN DONOR ANTECEDENTS ADDED
*   IMPROPER DIHEDRALS INCLUDED FOR SP2 PLANAR ATOMS BUT NOT CA
*   DIHEDRAL AND NONBONDED EXCLUSION LISTS SORTED FOR ALL AA
```

Title in the binary file itself:

```
*   RESIDUE TOPOLOGY FILE FOR PROTEINS USING ALL ATOM HYDROGENS
*   CAMBRIDGE NOTATION FOR ATOM NAMES
*   ATOM TYPE 3    HA  ADDED FOR ALIPHATIC HYDROGENS
*   ATOM TYPE 16   CT  ADDED FOR TETRAHEDRAL CARBONS
*   DIHEDRALS REPLACED BY IMPROPERS FOR GAUNIDO & SIDECHAIN AMIDE N
*   IMPROPER TORSIONS AND HYDROGEN DONOR ANTECEDENTS ADDED
*   IMPROPER DIHEDRALS INCLUDED FOR SP2 PLANAR ATOMS BUT NOT CA
*   PHI,PSI,OMEGA DIHEDRALS AND NONBONDED EXCLUSION LISTS SORTED
*   3/26/80 -- DAVID J. STATES
```

A.6 DNA Topology Files

The DNA topology files are largely untested. Also, they are not designed to work together with the protein topology files. These deficiencies will hopefully be corrected for version 3 of CONGEN.

A.6.1 RTOPDNA4

Modified version of '`RTOPDNA3`' with charges changed and other changes.

Title in the run:

```
*   RESIDUE TOPOLOGY FILE FOR DNA USING EXPLICIT HYDROGEN
*   ATOMS -- VERSION 4 -- BRB -- 2/13/82
```

Title in the file:

```
*  BROMINATED CYT AND WATER ADDED 10/20/81
*  RESIDUE TOPOLOGY FILE FOR DNA WITH EXPLICIT HYDROGENS
*  --  VERSION 4 --  BRB  -- 2/13/82
```

A.6.2 RTOPDNA3

Title in the run:

```
*   RESIDUE TOPOLOGY FILE FOR DNA USING EXPLICIT HYDROGEN
*   ATOMS -- VERSION 2 -- BRB -- 10/20/81
*   BROMINATED CYT AND WATER ADDED 10/20/81
*   CONVERTED TO NEW FORMAT 10/26/81
```

Title in the file:

```
*  BROMINATED CYT AND WATER ADDED 10/20/81
*  RESIDUE TOPOLOGY FILE FOR DNA WITH EXPLICIT HYDROGENS
*  --  VERSION 2 --  BRB  -- 10/20/81
```

A.6.3 RTOPDNA1

Title in the topology file itself:

```
*   DNA WITH EXPLICIT HYDROGEN TOPOLOGY FILE CREATION RUN
*   1/20/81 -- BRB
*
READ      RTF       CARDS
*   RESIDUE TOPOLOGY FILE FOR DNA USING EXPLICIT HYDROGEN
*   ATOMS -- VERSION 1 -- BRB -- 1/24/81
```

This topology file contains all the residues needed for a DNA structure. The phosphate linkage is stored as a separate residue (PHL) and must be included in the read sequence command (where needed).

A.6.4 AMBERUNIRTF

This file contains the united atom AMBER topology file. Whenever this topology file is used, you must use the automatic generation options in the GENERATE command, see Section 4.1 [Generate Command], page 43, as follows:

```
GENERATE [segid] ANGLE TORSIONS ALL DONOR ACCEPTOR
```

This file is now obsolete, and is replaced by the AMBER94 topology file, which follows.

A.6.5 AMBER94RTF

This file contains the AMBER 94 all atom topology file. There is no united atom topology file available for this parameter set. Whenever this topology file is used, the automatic generation options in the GENERATE command, see Section 4.1 [Generate Command], page 43, will be turned on by default. This file must be used with the AMBER 94 parameter file, see Section B.1.13 [AMBER94PARM], page 306.

This file contains the standard amino acids, several charge variants for polar amino acids, amino and carboxy terminal amino acids, and DNA and RNA nucleotides with no terminations, terminations on each end, and single complete nucleotides. In addition, D amino acids are also present. These are generated by inverting all the construction rules for the L amino acids, so that all the chiralities are changed, including those involving just nomenclature.

Atom names in this topology file are somewhat different from those used in the previous CHARMM and CONGEN topology files. It is intended to provide closer adherence to the IUPAC naming, see Section 2.12 [Syntactic Glossary], page 12, so that it will be easier to read Brookhaven Protein Data Bank files.

A new mechanism for patching the ends of a protein or nucleic acid is used with this topology file. The topology file stores different versions of each residue for each end as well as for interior positions. When a segment is generated, see Section 4.1 [Generate Command], page 43, the appropriate end residue will be used, but the final name of the residue will be reset to the original name. Thus, the terminal residues will have the usual name, and no terminal residues, like NTER or CTER, will be used.

The URL, `http://www.amber.ucsf.edu/amber/amber.html`, was the source for data in this file.

Appendix B Parameter Files and Their Origins

There are number of parameter files available. Immutable versions are stored in the directory, '`CGDATA`'. The file names used for these files consists of an alphabetic part followed by a number, e.g. '`PARAM5`'. There are two copies of each file; one with extension, '`.INP`', which is a character files used as an command file to generate the binary file, with extension, '`.MOD`'. The '`.INP`' is meant for human eyes; the '`.MOD`' files is meant for CONGEN to read. The numeric part of each name is its version number. In general, one should use the highest version number of a file.

There are four types of parameter files available; protein, DNA, AMBER, and obsolete. The protein parameter files are called '`PARAM`*n*'. The DNA parameter files are called '`PARMDNA`*n*'. The AMBER parameters files begin with '`AMBER`', and the obsolete files are the other files which begin with '`PARM`'.

This appendix begins with an essay by David J. States describing the development of the '`PARAM`*n*'. The remainder of the appendix lists the titles from all the parameter files.

All the parameter files except for the AMBER files were either in CHARMM version 16 or were derived from them.

B.1 Development of the PARAM1 Parameter Set

by David J. States

This essay describes the PARAM1 parameterization for the CHARMM molecular modelling program, compiled in May of 1981 by David States. My goal was to obtain a documented set of parameters which described the experimentally observable properties of small molecules as well as possible in the context of the CHARMM potential energy form. This was achieved by a combination of explicit least squares fitting where possible, and trial and error model building where it was not. Extended atom representations were used for aliphatic and aromatic hydrogens with the exception of benzene, where it was necessary to model the system using both extended and all hydrogen potentials. Explicit hydrogens were used for all protonated heteroatoms. Parameters have been included in the parameter file for atom types used in the `PROT` and `ALLH` topology files by generalizing the most appropriate HPRO parameter (ex. using the C-NH1 bond constants for C-NH1E).

B.1.1 Geometric Constants

The first step in the process of obtaining bond, angle, and dihedral parameters was to establish the mean bond and angle geometries. Bond lengths were taken from Bruce Gelin's Ph. D. thesis (the dipeptide modelling section), and agree quite well with other tabulations of standard bond lengths such as the CRC Handbook and Karplus and Porter. Angles were chosen as the average value observed in protein crystal structures available at better than 2.0 angstroms resolution on the Brookhaven Protein Data Bank tape. This includes the following structures:

beta trypsin (benzamidine inhibited)
    Fehlhammer and Bode (1975) *J. Mol. Biol.* **98**: 683.

beta trypsin complexed with pancreatic trypsin inhibitor
    Ruehlman et al (1973) *J. Mol. Biol.* **77**: 417

pancreatic trypsin inhibitor
    Deisenhofer and Steigemann (1975) *Acta Cryst. B* **31**: 238

trypsin inhibited with diisopropyl phosphate
    Chambers and Stroud (1979) *Acta Cryst. B* **35**: 1861

actinidin	Baker and Dodson (1977) *J. Mol. Biol.* **115**: 263
parvalbumin	Moews and Ketsinger (1975) *J. Mol. Biol.* **91**: 201
erythrocrurin - deoxy, carbon-monoxy, aquomet, and cyanomet	Steigemann and Weber (1979) *J. Mol. Biol.* **127**: 309
flavodoxin - oxidized and semiquinone	Smith et al (1977) *J. Mol. Biol.* **117**: 195

These coordinates and their sequences were read off the tape and tables of the angle geometries built using the CHARMM program. The average values for each angle type were collected over all of these structures weighted equally. This compilation allowed the averaging of twenty to fifteen hundred occurrences for each angle type, with variances ranging from less than one degree to as much as fifteen degrees, and ranges of two to sixty five degrees. In general these average values are within two degrees of the '`PARMFIX10`' values. It should be noted that all of the above structures are crystallographically refined. A certain amount of bias is therefore inevitable, but I feel justified by the higher resolutions that these methods achieve.

B.1.2 Force Constants

Having established these geometric parameters, the next step was the definition of force constants. Force constants describing simple hydrocarbons and amides were obtained by least squares fitting to the vibrational spectra of propane, butane, pentane, and N-methyl acetamide (Snyder et al and Warshel, Levitt, and Lifson). This fitting was carried out in stages, first choosing parameters and evaluating the resulting normal modes until a set was obtained with correctly ordered vibration frequencies. These parameters were then refined by least squares fitting to vibrational spectra of all the isotopic variants available. This fitting was performed with a user subroutine linked to CHARMM that called the CHARMM energy and vibrational analysis routines for normal mode analysis, and used an IMSL minimization routine (`ZXSSQ`). This fitting algorithm uses finite differences to construct an approximate first derivative vector and second derivative matrix, requiring a rapidly increasing number of function evaluations as the number of free variables rises. To reduce the complexity of this fitting, parameters were grouped (bonds vs. angles vs. dihedrals and impropers) and minimized separately until good agreement was obtained over the entire range of observed frequencies. The experimental and calculated normal modes for N-methyl acetamide are shown in the first table. The most prominent differences between the current and the previous parameterization are seen in the low frequency and out of plane bending modes.

The dihedral angle potential for alkanes was found to be 1.6 kcal/mole by this procedure in good agreement with the value reported in the dipeptide and acetylcholine work of Gelin (1.8 kcal/mole). It should be noted that the previous parameter sets used a much lower value (0.2 kcal/mole). The old value resulted in essentially unhindered rotation between trans and gauche isomers and a very small gauche well depth (0.03 kcal/mole). The new parameters are in much better agreement with experimental observations as can be seen in the second table.

For the aromatic systems it was not possible to perform this least squares fitting because the simple potential energy form that we are using is not able to correctly order the various normal modes (Varsanyi and Dollish et al). This is apparently a result of the strong valence interactions between internal coordinates, and presumably could be overcome using a potential form that includes valence cross terms. The large number of modes (30) in benzene and the strong interaction of carbon and hydrogen motions also complicated the analysis of this system. For these reasons, benzene was analyzed by manually adjusting parameters for a model including all hydrogens. The heavy atom parameters obtained from this system agree with those used previously, and were kept fixed in modelling more elaborate aromatic systems.

Using the CHARMM potential energy function, aromatic out of plane bending behavior is determined by both the arrangement and parameterization of the improper dihedrals. For histidine, phenylalanine, and tyrosine the current parameterization and arrangement gave good agreement for the intrinsic ring modes (based on the benzene modelling), and they were retained. The ring substituent out of plane modes were not well described, and it was necessary to parameterize the system using the lowest out of plane bending modes for toluene (217 cm-1) and phenol (241 cm-1) to fix new constants for the ring CB and ring hydroxyl bends respectively. The present improper dihedral was retained, but in both cases the force constants required large increases (from 25 kcal/mol/rad2 to 120 and 150 respectively).

Tryptophan required both rearrangement and reparameterization of the improper dihedrals to obtain reasonable agreement with the skeletal out of plane bend (400 to 500 cm-1 in purines and napthalene) without creating very high frequency modes involving the ring intersection. The new arrangement replaces the improper dihedrals centered on the atoms at the intersection of the rings (CD2 and CE2) with terms that are trans two fold dihedrals beginning on one ring, running across the bridging atoms and ending on the opposite ring as shown below:

Tryptophan

```
      CZ2           NE1
    *     *       *     *
   *       *     *       *
CH2         CE2           *
 |           |             *
 |           |              CD1
 |           |             *
 |           |            *
CZ3         CD2          *
   *        *    *      *
    *      *      *    *
      CE3           CG
                     |
                     |
                    CB
   PARAM1                    PARMFIXn

CZ2                        CZ2----------NE1
   *                          *        *
    *                          *      *
     CE2                        *  CE2
      |                          *
      |                           *
     CD2                           CD2
        *
         *
          CG

          NE1
         *
        *
     CE2                           CE2
      |                           *
      |                          *
     CD2                        *  CD2
    *                          *      *
   *                          *        *
CE3                        CE3-----------CG
```

In addition, two improper dihedrals were added as shown below.

```
CH2--------CE2                               CE2 *
            |                                 |    *
            |        * CD1                    |       * CD1
            |     *                           |
           CD2 *                   CZ3-------CD2
```

This new arrangement distributes the out of plane bending forces evenly across the ring in a manner analogous to the delocalized electronic structure of the system, and does so without excessively elevating the frequency of CD2/CE2 out of plane modes. With the 'PARMFIX10' parameters these modes had frequencies of about 1000 cm-1, but the skeletal out of plane bend occurred at 120 cm-1. Simply raising the force constants without rearranging the dihedrals resulted in a bridge atoms out of plane mode at 3000 cm-1! Aside from being unrealistic, this high frequency would force the use of small step sizes to retain accuracy in the dynamics integration.

B.1.3 Hydrogen Bonds and Non-bonded Interactions

My effort concentrated on the bond, angle, and dihedral force constants. The non-bond interaction parameters from Bruce Gelin's thesis were used without modification or cutoffs for the model systems. These nonbond parameters are the same as those found in 'PARMFIX10'.

The hydrogen bond geometries and well depths were obtained from a table of typical values found in Vinogradov and Linell. Because the van der Waals and electrostatic terms contribute significantly to the total hydrogen bond energy, the hydrogen bond parameters were adjusted to give an energy minimized water dimer with the desired geometry and well depth. The other terms were shifted similarly, although by smaller amounts because the van der Waals interaction is not as large in the longer nitrogen-oxygen and nitrogen-nitrogen pairs. No hydrogen bonds were present in the model systems used above.

B.1.4 Summary and Discussion

'`PARAM1`' is a refined set of parameters for the CHARMM potential energy form that describes the molecular mechanics of systems containing the covalent structures found in biological molecules. While an effort has been made to include different model compounds and conformations to generalize these parameters, they are primarily based on simple amides and hydrocarbons and intended for use in protein simulations. For non-protein systems it would be advisable to verify the properties of the constituents using model compounds (such as sugars and nucleotides) before macromolecular simulations are attempted. The accuracy of the geometric and force constants presented in this set is difficult to estimate. For some parameters such as bond stretching constants isolated modes could easily be identified and the parameter set to an accuracy of five percent or better. Conversely, angle bending and dihedral constants could not always be associated with particular modes, and significant correlations between the various parameters were observed during fitting. Even where least squares fitting was possible, none of these force constants should be regarded as accurate to better than twenty per cent. One corollary of this interdependence is the danger in naively readjusting parameters without repeating the analysis of all of the various model systems affected.

Given the number and extent of the changes presented in this work, the question of how they will affect results naturally arises. It should be noted that almost all of the changes presented here will tend to make molecules more rigid than they were previously. This will have implications in both the accuracy of the dynamics algorithm at a given step size, and the magnitude of the expected fluctuations. The behavior of the peptide bonds should not be markedly changed since the net effect of the parameter change on their vibrational modes is small. The aliphatic sidechains and coupling of motions through angle and dihedral terms is expected to show more prominent effects.

References:

Dollish, Fately, and Bentley (1975) *Characteristic Raman Frequencies of Organic Compounds* Wiley and Sons, New York.

Gelin Ph. D. Thesis (Chapters 2 and 4, Tables 12 to 15)

Gelin and Karplus (1975) *JACS* **97**: 6996

Snyder and Schachtscheider (1965) *Spectrochimica Acta* **21**: 169-195

Varsanyi (1974) *Assignments to Vibrational Spectra of 700 Benzene Derivatives* Halsted Press, Budapest.

Vinogradov and Linell (1971) *Hydrogen Bonding* van Nostrand.

Warshel and Lifson (1970) *J. Chem. Phys.* **53**: 582

Warshel, Levitt, and Lifson (1971) *J. Mol. Spect.* **33**: 84

B.1.5 Table I — N-Methyl Acetamide Vibrational Analysis

mode	Experimental	New	Old
peptide torsion	192	189	162
N bend	289	281	291
C in plane	437	444	375
C out of plane	600	551	312
Amide IV	628	621	544
NH out of plane	725	730	390
C C stretch	883	866	864
C N stretch	1120	1115	1112
Amide III	1323	1314	1262
Amide II	1570	1597	1580
Amide I	1660	1661	1675
N H stretch	3200	3193	2969

B.1.6 Table II — n-Butane Dihedral

Adiabatic Potential Energy Surface for the n-Butane Dihedral

Angle	Experimental	New	Old
180.0	0.0	-0.23	-0.23
165.0		0.21	0.00
150.0		1.32	0.27
135.0		2.49	0.65
120.0	3.4	3.04	0.86
105.0		2.69	0.87
90.0		1.82	0.84
75.0	0.8	1.32	1.05
60.0		1.67	1.65
45.0		2.97	2.61
30.0		4.92	3.68
15.0		6.73	4.54
0.0	6.1	7.43	4.87

B.1.7 Table III — Comparison of Force Constants

Term	New Set	Old Set
C O bond	580	600
C CH bond	405	400
C NH bond	471	500
NH CH bond	422	450
NH H bond	405	350
CH C O angle	85	40
CH C NH angle	20	25
NH C O angle	65	50
C NH CH angle	77.5	60
C NH H angle	35	30
CH NH H angle	30	30
CH CH CH angle	45	30
C NH torsion	8.0	7.0
CH CH torsion	1.6	0.2
C improper	105	25
N improper	50	25
toluene improper	120	25
phenol improper	150	25

B.1.8 PARAM2 – Modification of PARAM1 for CO and Heme

'`PARAM2`' has new parameters added pertaining to carbon monoxide binding to heme proteins.

B.1.8.1 Bonds

CM-OM
: carbon monoxide stretch, see Caughey et al (*Biochemical and Clinical Aspects of Hemoglobin Abnormalities*).

FE-CM
: Iron-carbon monoxide stretch, see Kroeker et al (*J. Chem. Phys.* **72**: 4846, 1980) Values for the chemisorption of CO onto Fe matrix, data based on tunnelling spectroscopy; no data for proteins

FE-NR
: Fe-proximal histidine N epsilon stretch, see K. Nagai et al (*J. Mol. Biol.* **136**: 271, 1980); Kincaid et al (*Proc. Nat. Acad. Sci. USA* **76**: 549, 1979) No change in equilibrium bond distance from that in Gelin's thesis.

B.1.8.2 Angles

FE-CM-OM
: Iron carbon monoxide bend, see Kroeker et al (vide supra)

NP-FE-CM
: Porphyrin nitrogen Fe carbon monoxide bend. No data available, values selected to be consistent with that for proximal histidine (5th ligand) nitrogen Fe porphyrin nitrogen bend.

B.1.8.3 Dihedrals

X-FE-CM-X
: no data available, values selected to be consistent with that for X-FE-NR-X (i.e. just as the heme can rotate without great hindrance about the Iron-proximal histidine bond so can the CO rotate freely about the Iron-carbon bond)

B.1.8.4 Polarizabilities Etc.

The values for CM same as that for carbonyl carbon, and the values for OM same as that for carbonyl oxygen.

B.1.9 PARAM3

'`PARAM3`' is a modification of '`PARAM2`' which has peptide geometries from Ramachandran et al *BBA* **359**: 298 (1974), and torsions from Hagler et al *J. Amer. Chem. Soc.* **98**: 4600 (1976).

B.1.10 PARAM4

'`PARAM4`' is a modification of '`PARAM3`' which has a few additional parameters for carbon monoxide and molecular oxygen as well as revised torsions for better agreement to infrared spectra for phenol, toluene, and napthalene.

B.1.11 PARAM5

'`PARAM5`' is a modification of '`PARAM4`' which has a few additional parameters for ring carbons, and those needed for the all hydrogen topology ('`ALLH`') files.

B.1.12 AMBERPARM

The AMBER parameter set[1] is available as '`AMBERPARM`'.

This parameter file is now obsolete, and is replaced by the AMBER 94 parameter file, which is described in the next section.

B.1.13 AMBER94PARM

The AMBER 94 parameter set[2] is available as '`AMBER94PARM`'. This parameter file must be used with the AMBER 94 topology file, see Section A.6.5 [AMBER94RTF], page 297.

Several atoms in the AMBER94 potential have zero van der Waals radii. This causes problems for CONGEN because of the possibility that such atoms may get close enough to another atom for electrostatic fusion to occur. Therefore, such atoms have been given a small van der Waals radius and a small van der Waals repulsive interaction. Please see the file for the exact value.

This potential file has some features which you should be aware of. First, the amide torsion terms (as found in the aspargine and glutamine sidechains) are designed so that the minimum energy is non-zero, and the shape is correct only when both hydrogens are considered. Second, there is no explicit hydrogen bond energy. Such interactions are accounted for by the electrostatic energy. Third, zero energy improper torsion terms have been added for chiral centers so that the analysis tables, see Section 17.1.3.1 [Internal Coordinate Properties], page 160, will show the chirality and allow the user to check it.

[1] Scott J. Weiner, Peter A. Kollman, Dzung T. Nguyen and David A. Case, "An All Atom Force Field for Simulations of Proteins and Nucleic Acids", *J. Comput. Chem* **7**, 230-252 (1986)

[2] Wendy D. Cornell, Piotr Cieplak, Christopher I. Bayly, Ian R. Gould, Kenneth M. Merz, Jr., David M. Furguson, David C. Spellmeyer, Thomas Fox, James W. Caldwell and Peter A. Kollman, "A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules", *J. Am. Chem. Soc.* **117**, 5179-5197 (1995)

B.2 Obsolete Parameters for Proteins

B.2.1 PARMEXEL

Title found in the run creating the parameters:

```
*   PARAMETER FILE FOR EXEL VERSION 11 CREATION RUN
*   HYDROGEN BOND STRENGTHS INCREASED BY 1.0 KCAL/MOLE FROM PARMFIX11
*   LENGTHS DECREASED BY 0.1-0.2 ANGSTROM WALLWORK ACTA. CRYST. (66) 15:758
*       7-Apr-1981  21:33:52 - DJS
```

Title written to the binary file:

```
*   PARAMETER  VERSION 10 -- EXTENDED AND EXPLICIT SET -- 27 ATOM TYPES
*   COMPILED BY WFVG, SNS AND BDO -- MARCH 1979
*   HYDROGEN BOND PARAMETERS CHANGED FROM VERSION 5 -- ONLY O-O,O-N,N-N
*   H BOND STRENGTHS INCREASED 1.0 KCAL/MOLE FOR EXEL LENGTHS SHORTENED
*   NONBONDED PARAMETERS FOR HYDROGEN (A-0.044,N-1,R-0.8 == EMIN-0.1)
*   PHIS INVOLVING C(CARBONYL) AND NITROGENS MADE CONSISTENT IN VER. 8
*   O-H BOND FORCE CONSTANTS CHANGED TO 400.0 IN VERSION 9
*   ADDED ANGLE FOR N-TERMINAL PROLINE -- BRUC
*   LAST UPDATE 7-Apr-1981  21:20:20
```

B.2.2 PARMFIX10

Title written to the binary file:

```
*   PARAMETER  VERSION 10 -- EXTENDED AND EXPLICIT SET -- 27 ATOM TYPES
*   COMPILED BY WFVG, SNS AND BDO -- MARCH 1979
*   HYDROGEN BOND PARAMETERS CHANGED FROM VERSION 5 -- ONLY O-O,O-N,N-N
*   NONBONDED PARAMETERS FOR HYDROGEN (A-0.044,N-1,R-0.8 == EMIN-0.1)
*   PHIS INVOLVING C(CARBONYL) AND NITROGENS MADE CONSISTENT IN VER. 8
*   O-H BOND FORCE CONSTANTS CHANGED TO 400.0 IN VERSION 9
*   ADDED ANGLE FOR N-TERMINAL PROLINE -- BRUC
*   LAST UPDATE 16-Mar-1981  21:54:36
```

Additional angle for proline cropped up when a bug was found in `PATEXH` and explicit hydrogen topology file. The angle added was CH1E-NH3-CH2E which was given the parameters for HC-NH3-CH1E which appeared to be the closest angle to it. –Bruc

B.2.3 PARMFIX9

Title written to the binary file:

```
*   PARAMETER  VERSION 9
*   EXTENDED AND EXPLICIT SET -- 27 ATOM TYPES
*   COMPILED BY WFVG, SNS AND BDO -- MARCH 1979
*   HYDROGEN BOND PARAMETERS CHANGED FROM VERSION 5 -- ONLY O-O,O-N,N-N
*   NONBONDED PARAMETERS FOR HYDROGEN (A-0.044,N-1,R-0.8 == EMIN-0.1)
*   PHIS INVOLVING C(CARBONYL) AND NITROGENS MADE CONSISTENT IN VER. 8
*   O-H BOND FORCE CONSTANTS CHANGED TO 400.0 IN VERSION 9
*   LAST UPDATE 5/4/80
```

B.2.4 PARMFIX8

Title written to the binary file:

```
*   PARAMETER  VERSION 8
*   EXTENDED AND EXPLICIT SET -- 27 ATOM TYPES
*   COMPILED BY WFVG, SNS AND BDO -- MARCH 1979
*   HYDROGEN BOND PARAMETERS CHANGED FROM VERSION 5 -- ONLY O-O,O-N,N-N
*   NONBONDED PARAMETERS FOR HYDROGEN (A-0.044,N-1,R-0.8 == EMIN-0.1)
*   PHIS INVOLVING C(CARBONYL) AND NITROGENS MADE CONSISTENT IN VER. 8
*   LAST UPDATE 3/23/80
```

B.2.5 PARMFIX7

Title written to the current binary file:

```
*   PARAMETER  VERSION 7
*   EXTENDED AND EXPLICIT SET -- 27 ATOM TYPES
*   COMPILED BY WFVG, SNS AND BDO -- MARCH 1979
*   HYDROGEN BOND PARAMETERS CHANGED FROM VERSION 5 -- ONLY O-O,O-N,N-N
*   NONBONDED PARAMETERS FOR HYDROGEN (A-0.044,N-1,R-0.8 == EMIN-0.1)
*   LAST UPDATE /1/24/80
```

B.2.6 PARMFIX5 (Unreadable)

Title written to the binary file:

```
*   PARAMETER  LIST  -  VERSION 5
*   EXTENDED AND EXPLICIT SET -- 27 ATOM TYPES
*   COMPILED BY WFVG, SNS AND BDO -- MARCH 1979
*   LAST UPDATE 4/10/79
```

B.2.7 PARMFIX2 (Unreadable)

Title written to the binary file:

```
*   EXTENDED ATOM PARAMETERS -- VERSION 2
*   15 ATOM TYPES -- THIS PARAMETER FILE WAS THE MOST LIKELY
*   FILE WITH WHICH THE LONG ( 100 PICOSECS ) PTI DYNAMICS RUN
*   WAS MADE -- IT HAS SOME LARGE ANGLE FORCE CONSTANTS AND NO
*   PARAMETERS FOR IMPROPER TORSIONS -- PARAMETERS NOT NEEDED IN
*   PTI HAVE ALSO BEEN ADDED -- 3/15/79 -- BDO AND SNS
```

B.2.8 PARMALLH

This file is obsolete. The latest version of '`PARAM`*n*' should be used for all hydrogen topology files.

Title in the input file generating the binary file.:

```
*   PARAMETER FILE FOR ALL EXPLICIT HYDROGENS BASED ON CFF PARAMETERS
*   IN BRUCE GELIN'S THESIS (PARMFIX7 VALUES HAVE TRAILING 0'S)
*   BOND ANGLE PARAMETERS INCLUDE THE HARMONIC CONTRIBUTION FROM UREY-BRADLEY
*   F TERM ( (R13-R0)**2 )
*   DIHEDRALS INCLUDE MULTIPLICITY FROM CFF
*   MODIFIED TO INCLUDE HA AND CT
*   HIS AND TRYP INCLUDED 4/13/80 DJS
```

Title written to the parameter file:

```
*   ALLH PARAMETER SET MODIFIED FROM GELIN THESIS 3/4/80
*   MODIFIED TO INCLUDE HA AND CT   -- 2/24/80 -- DJS
```

B.3 DNA Parameter Files

The DNA parameter files have had very limited usage. Do not trust the results using them unless you satisfy yourself that the parameters are reasonable.

The DNA parameters are inconsistent with the protein parameters. Do not mix proteins and DNA together. You will run into problems with chemical type codes being reused and general confusion and errors.

B.3.1 PARMDNA4

'`PARMDNA4`' is the same as '`PARMDNA3`' except a sodium ion has been added to the nonbonded parameters.

Creation title:

```
*   PARAMETER FILE -- VERSION 4 -- CREATION RUN -- 10/20/81 -- BRB
```

Title in the file:

```
*   PARAMETER  VERSION 4
*   DNA
```

B.3.2 PARMDNA3

Titles from the creation run:

```
*   PARAMETER FILE -- VERSION 2 -- CREATION RUN -- 10/20/81 -- BRB
*   BROMINATED CYT AND WATER ADDED 10/20/81
```

Title in the file:

```
*   PARAMETER  VERSION 2
*   PARAMETERS FROM UCSF EXCEPT FOR HBONDS AND IMPHIS
*   DNA
```

B.3.3 PARMDNA1

Title from the input file:

```
*   PARAMETER FILE -- VERSION 1 -- CREATION RUN -- 1/29/81 -- BRB
```

Title written to the binary file:

```
*   PARAMETER  VERSION 1
*   PARAMETERS FROM UCSF EXCEPT FOR HBONDS AND IMPHIS
*   DNA
```

Command and Option Index

This index contains entries for all CONGEN commands and all CONGEN keywords. If you're trying to comprehend an example input, and you don't know what an option means, this index is the place to look.

Concept Index