DYANA

Version 1.5

User's Manual

Peter Güntert, Christian Mumenthaler and Torsten Herrmann

Institut für Molekularbiologie und Biophysik
Eidgenössische Technische Hochschule
CH-8093 Zürich
Switzerland

June 1998

Contents

Introduction

The main purpose of the program package DYANA ("Dynamics algorithm for NMR applications") is the calculation of three-dimensional protein and nucleic acid structures on the basis of conformational restraints derived from the NMR experiments. DYANA is the successor of DIANA (Güntert *et al.*, 1991a; Güntert & Wüthrich, 1991). It provides all functionality of DIANA and many extensions, in particular a new, efficient structure calculation method: torsion angle dynamics.

During the last years, the steady increase in size of macromolecular structures that can now be solved by NMR has called for improved and more efficient structure calculation methods. The program DIANA relied on conjugate gradient minimization of a variable target function in torsion angle space (Braun & Gō, 1985) to find three-dimensional structures that fulfil the conformational restraints. The success rate (percentage of structures reaching low target function values) of this strategy was, in particular for larger β-sheet proteins, often hampered by the fact that the target function has many local minima into which the conjugate gradient minimizer may become trapped because it takes exclusively downhill steps. A decisive improvement of this situation could be expected from other structure calculation algorithms that have the possibility to "escape" from unfavorable local minima. Therefore, we created a new NMR structure calculation program, DYANA (Güntert *et al.*, 1997), that uses simulated annealing combined with molecular dynamics in torsion angle space (torsion angle dynamics), i. e., the numerical solution of the classical mechanical equations of motion (Lagrange equations) with torsion angles as generalized coordinates. The target function takes the role of the potential energy, and the system is coupled to a temperature bath which is cooled down slowly from its initial high temperature, thereby allowing the system to cross barriers between local minima of the target function.

When compared with other structure calculation algorithms that are based on simulated annealing (Brünger, 1992), the principal difference of torsion angle dynamics is that it works with internal rather than with Cartesian coordinates. The covalent structure parameters (bond lengths, bond angles, chiralities and planarities) are always kept fixed at their optimal values. The strong potentials required in conventional Cartesian space molecular dynamics to retain the covalent structure and, concomitantly, the high frequency motions caused by them are absent in torsion angle dynamics. This results in a simpler potential energy function and in longer permissible time-steps for the numerical integration of the equations of motion, and thus in a much higher efficiency of the algorithm.

To fully exploit the great potential advantages of torsion angle dynamics, a careful consideration of its implementation was required because the Lagrange equations of motion with torsion angles as degrees of freedom are much more complex than Newton's equations in Cartesian coordinates. A "naive" implementation of torsion angle dynamics (Mazur *et al.*, 1991) would entail in every time-step the solution of a system of N linear equations (N being the number of degrees of freedom), and thus require a prohibitive computational effort proportional to N^3. In contrast, DYANA uses a fast recursive implementation of the equations of motion—originally developed for spacecraft dynamics and robotics (Jain *et al.*, 1993)—with a computational effort proportional to N.

In addition, the new structure calculation program DYANA incorporates a method for the automatic assignment of NOESY spectra on the basis of known sequential resonance assignments, peak positions, and peak intensities (Mumenthaler *et al.*, 1997). Initial NOESY cross peaks assignments derived from matching chemical shifts are subsequently refined in several cycles consisting of structure calculations using an error-tolerant target function followed by an assessment of the possible peak assignments in the light of the (preliminary) three-dimensional structures obtained. This method has the potential to largely replace the manual method for NOESY assignment. Given that currently the NOESY assignment and the collection of conformational constraints for a protein of ~150 amino acid residues may require several months of manual work, the far-reaching consequences of the availability of a reliable automatic NOESY assignment method become evident.

Since the calculation of macromolecular three-dimensional structures is a computationally intensive procedure, it is important to make best possible use of the available computing resources. With the advent of a variety of parallel computers and distributed computing networks, the optimization of our structure calculation software, which was up to now geared to the requirements of vector supercomputers, has turned towards optimal parallelization. A structure calculation that consists of the independent generation of many conformers has a high degree of inherent

parallelism. It can be exploited almost ideally for parallel computing and renders feasible applications that have so far been unpractical because of their high computational demands.

DYANA is written in standard FORTRAN-77 and was implemented on a variety of computers. The program is optimized for shared-memory multiprocessor and vector computers.

Any reports or publications of results obtained with the program DYANA must acknowledge its use by an appropriate citation:

> P. Güntert, C. Mumenthaler and K. Wüthrich:
> Torsion angle dynamics for NMR structure calculation with the new program DYANA. *J. Mol. Biol.* **273**, 283–298 (1997).

The structure of this manual is as follows. The *Tutorial* gives an introduction to the program for first users and also explains how the most common tasks are performed with DYANA. The interactive command language, INCLAN, is described in the next chapter. *Commands* gives a complete list and description of all DYANA commands. *Variables and Functions* gives a complete list of all DYANA system variables and functions. *Selections* describes the syntax used to select atoms, angles, peaks, distance constraints, and structures. *File formats* describes the formats of various data files used by DYANA. The supporting program COFIMA is described in separate chapters. *Installation* describes the installation of DYANA on Unix systems.

In this manual names of commands, variables etc. and literal input is printed in **bold Helvetica**, other input is printed in *italics*. Optional parameters are given in square brackets [. . .], and optional parameters that may be repeated zero or more times are given in curly braces {. . .}. In examples, input to the program is printed in **`bold Courier`**, and output from the program DYANA is printed in regular `Courier` font.

Comments, suggestions, and bug reports are welcome. Please send them by electronic mail to peter@guentert.com.

Tutorial

This chapter explains how to do some of the tasks that are commonly performed with the program DYANA and its interactive command language, INCLAN, in the course of an NMR structure determination. The example input files used for this tutorial are distributed together with the program and can be found in the "example" subdirectory of the library directory.

Running DYANA

This is a simple example to introduce a new user to the program DYANA. It is not CPU intensive and can be executed interactively on any workstation. Note, however, that this example does not provide the recommended strategy for a protein structure calculation. See the next section for a complete, prototypical example of a realistic NMR structure calculation.

The program DYANA is started by ("%" is the UNIX prompt; user input is printed in bold):

```
% dyana

DYANA, version 1.5 (sgi, double precision)

Copyright (c) 1996-98 ETH Zurich

dyana>
```

The title line shows the version number of the program, the computer architecture for which it has been compiled, and an indication of whether single (32 bit) or double (64 bit) precision arithmetics will be used. "dyana>" is the prompt of DYANA which is shown when the program is ready to accept commands from the user. If the prompt does not appear after starting the program, the program is not installed correctly.

The command **readdata** reads the input files for a structure calculation (input files are in the "example/helix" directory):

```
dyana> readdata helix
    Library file "/soft/lib/dyana-1.1/lib/dyana.lib"
       read, 42 residue types.
    Sequence file "helix.seq" read, 11 residues.
    Distance constraint file "helix.upl" read,
       145 upper limits.
    Distance constraint file "helix.lol" read,
       14 lower limits.
    Angle constraint file "helix.aco" read,
       26 constraints.
```

A command may have parameters that are separated from the command name and from each other by blanks. In the above example, the **readdata** command has one parameter, the name of the input data files.

The **readdata** command reads the standard residue library (dyana.lib), the sequence of the molecule under study (helix.seq), and files with conformational constraints: upper distance limits (helix.upl), lower distance limits (helix.lol), and dihedral angle constraints (helix.aco) for a 11 residue polypeptide—the second helix of a mutant of the *Antennapedia* homeodomain from *Drosophila melanogaster* (Güntert *et al.*, 1991b).

Output from the program is indented, for example the confirmation that the sequence was successfully read: "`Sequence file "helix.seq" read, 11 residues.`"

The following command calculates a group of 5 conformers:

```
dyana> calc_all 5
    5 random structures created (seed 3771).
    Structure annealed in 2 s, f = 0.630422.
    Structure annealed in 2 s, f = 0.112963.
    Structure annealed in 2 s, f = 0.197074.
    Structure annealed in 2 s, f = 0.314798.
    Structure annealed in 2 s, f = 0.384202.
    5 structures finished in 11 s (2 s/structure).
```

First, 5 structures with random values for the dihedral angles are created. Then, for each of these random structures violations of the conformational constraints are minimized by simulated annealing using torsion angle dynamics (Güntert *et al.*, 1997).

Cartesian coordinates of all resulting structures can be saved with the **write cor** command:

```
dyana> write cor helix all
    DG coordinate file "helix.cor" written,
       5 conformers.
```

To get an overview of the quality of the minimized structures, the **overview** command can be used:

```
dyana> overview helix
```

A complete protein structure calculation

This example provides a complete, prototypical protein structure calculation from experimental NMR data for the pheromone Er-2 from *Euplotes raikovi* (Ottiger *et al.*, 1994; note, however, that the data set used for this example is *not* the one described in the publication). Data are in the directory "example/er2". The basic input data files are:

er2.seq	amino acid sequence
er2.prot	chemical shift list
er2_h2o.peaks	H_2O NOESY peak list
er2_d2o.peaks	D_2O NOESY peak list
er2.cco	vicinal scalar coupling constants

In addition, there are four DYANA macro files, "init.dya", "CALIBA.dya", "GRIDSEARCH.dya" and "ANNEAL.dya", to perform the various steps of the structure calculation.

The initialization macro, **init**, is executed each time the program DYANA is started from this directory:

```
name:=er2                          protein name (used as file name)
rmsdrange:=3..37                   default residue range for RMSD
dyanalib                           read library
read seq $name.seq                 read sequence
```

This macro defines two variables for later use, and reads the library and sequence file of the protein.

Calibration, i.e, conversion from NOESY peak volumes to upper distance bounds is performed by the macro **CALIBA**:

```
read seq $name.seq                 read sequence and initialize
read prot $name.prot               read proton list
read peaks {$name}_h2o.peaks assigned integrated
caliba                             read and calibrate first peak list
read peaks {$name}_d2o.peaks assigned integrated
caliba                             read and calibrate second peak list
distance unique                    keep strongest constraint for each distance
write upl caliba.upl               save upper limits before modifications
```

Calibration is performed separately for each peak list using the default method implemented in the macro **caliba**. See the section "Calibrating NOEs" later in this tutorial for more details on calibration.

The next step is a systematic analysis of the local conformation around the C^{α} atom of each residue using grid searches. On the basis of local distance constraints from the file “caliba.upl” and scalar coupling constants from the file “er2.cco” allowed conformations for the dihedral angles ϕ, ψ, χ^1 and χ^2 of each residue are determined using the macro **GRIDSEARCH**:

```
read seq $name.seq                    read sequence and initialize
read upl caliba.upl                   read NOE upper distance limits
read cco $name.cco                    read scalar coupling constants

atoms stereo HB2 2 5
atoms stereo QD1 39
                      Define stereospecific assignments that are already known.
habas angles="CHI1 CHI2*" tfcut=0.05
                      Perform grid searches for all amino acid residues including the dihedral angles φ, ψ, χ1 and χ2. Allow conformations with local target function values up to 0.05 Å2.
gridplot habas.ps              create plot(s) with allowed angle ranges
atom stereo list               List stereospecific assignments

distance modify                modify distance constraints
write upl $name.upl            save upper limits
write aco $name.aco            save angle restraints
```

The command **distance modify** removes irrelevant constraints (constraints that involve fixed distances and constraints that cannot be violated), retains maximally one distance limit for each atom pair and introduces corrections for constraints with diastereotopic substituents for which stereospecific assignments are not available.

The result of this step are the modified upper distance limits in the file “er2.upl”, the dihedral angle constraints in the file “er2.aco” and stereospecific assignments.

The actual structure calculation is performed with the macro **ANNEAL** using torsion angle dynamics:

```
read seq $name.seq             read sequence and initialize
read upl $name.upl             read upper distance limits
read aco $name.aco             read angle constraints
ssbond 5-20 12-37 17-28        generate constraints for S-S bonds

seed=35621                     random number generator seed
#nproc=4                       number of processors
calc_all 30                    simulated annealing
```

```
overview $name structures=20 ang cor
```
produce overview

After reading the input data file, constraints for the three disulfide bonds 5–20, 12–37 and 17–28 are generated with the macro **ssbond**. The random number generator seed and, if applicable, the number of processors that can be used in parallel is set, and 30 conformers are calculated with simulated annealing in torsion angle space, using the standard annealing protocol, i.e. the macro **anneal**. For larger proteins it could be necessary to increase the number of time-steps and/or the number of conformers. Finally, the 20 best conformers are analyzed, an overview file "er2.ovw", an angle file "er2.ang", and a coordinate file "er2.cor" are written. The angle file "er2.ang" contains all 20 conformers, sorted by increasing target function value. For a later analysis, all 20 conformers can be loaded into the program with the command

```
read ang er2.ang
```

Building an INCLAN macro

The possibility to create new commands from existing ones by combining them in *macros* is a powerful feature of INCLAN. A macro is created by saving a sequence of commands into a file with the extension ".dya". It can be invoked in the same way as existing commands simply by typing its name.

Suppose that we want to build a macro to execute the example structure calculation in the first section of this tutorial, "Running DYANA". The macro shall be called **calculate** (i. e. it is stored in a file called "calculate.dya") and have two parameters, the file name of the input and output files and the number of structures to calculate. A first implementation is:

```
readdata $p1
calc_all $p2
write cor $p1 all
overview $p1
```

p1 and **p2** denote the two command line parameters. The corresponding call of this macro in order to execute the above example is

```
calculate helix 5
```

A second implementation of the **calculate** command uses the INCLAN command **syntax** to declare an interface with names and, possibly, default values of the command line parameters:

```
## calculate - calculate a group of structures
##
```

```
## Usage: calculate file=<file> [struct=<n>]

syntax file=* struct=@i=5

readdata $file
calc_all $struct
write cor $file all
overview $file
```

Now, the two parameters are available inside the macro under the names **file** and **struct**, respectively. The asterisk "*****" indicates that the value of the **file** parameter can be any character string, whereas "**@i**" restricts the **struct** parameter to have only integer values. The **struct** parameter has a default value of 5, and there is no default value for the **file** parameter. All the following calls of the **calculate** command are equivalent:

```
calculate helix 5                    Positional parameters
calculate helix                      Default value for parameter struct
calculate file=helix struct=5        Named parameters
calculate struct=5 file=helix        Any order of parameters
calculate helix str=5                Abbreviated parameter name
```

Lines at the beginning of the **calculate** macro that start with "**##**" are comment lines used by the on-line help system: These are displayed when on-line help is requested about the macro:

```
dyana> help calculate

    calculate - calculate a group of structures

    Usage: calculate file=<file> [struct=<n>]
```

Calculating structures using torsion angle dynamics

The macro for the calculation of a structure by simulated annealing with molecular dynamics in torsion angle space (torsion angle dynamics,; TAD) is **anneal**. The standard simulated annealing protocol that is used if the **anneal** macro is called without parameters, consists of 4000 TAD steps. One fifth of these are performed at an initial high temperature, followed by slow cooling during the rest of the schedule. Various parameters of the standard annealing protocol can be changed by the user. For instance, 6000 TAD steps will be performed with the command

```
anneal steps=6000
```

An ensemble of structures can be calculated using TAD with the macro **calc_all**. With

```
calc_all 30
```

30 structures are calculated by applying the standard protocol, anneal, to 30 start cconformers with random torsion angles. The resulting conformers are stored in structure memories 1–30. To use instead of **anneal** another, modified annealing schedule that is, say, stored in a macro **myanneal**, the command is:

```
calc_all 30 myanneal steps=5000
```

steps=5000 is an example of a parameter that will be passed to the **myanneal** macro.

An overview file ("helix.ovw"), an angle file ("helix.ang") and a coordinate files ("helix.cor") of the 20 conformers with lowest target function value can be generated after the structure calculation with the command

```
overview helix structures=20 ang cor
```

Molecular dynamics in torsion angle space is the preferred structure calculation method for all proteins except, maybe, small helical peptides.

Calculating structures using the REDAC strategy

In the *REDAC strategy* (Güntert & Wüthrich, 1991), an ensemble of *n* structures is first calculated with the variable target function method and then analyzed with regard to the distribution of the values of the dihedral angles. Redundant dihedral angle constraints are generated for all residues with a local target function value below *ang_cut* in at least *nallow* structures. These constraints are used to re-calculate an ensemble of *n* structures. The procedure can be repeated several times. At the end the structures are minimized on the highest minimization level against the original angle constraints. The different *ang_cut* values for every REDAC cycle and the number of structures, *n*, are given as parameters to the macro **redac**, and *nallow* is a DYANA variable.

To calculate 50 structures of the protein E*r*-2 using one REDAC cycle with an *ang_cut* value of 0.3 you can write:

```
redac er2 schedule=0.3,0.0,0.0 structures=50
```

The two zeros in the *ang_cut* list stand for the cycle which uses the redundant angle constraints calculated previously and the cycle where the structures are minimized at the top level. In these two cycles no new redundant angle constraints are generated. Several files will be created:

- The angle files "er2a.ang", "er2b.ang" and "er2c.ang" containing all

50 conformers calculated in the cycles a, b and c. These files can be reloaded into the structure memory at any time with "**read ang** er2a".

- The overview files "er2a.ovw", "er2b.ovw" and "er2c.ovw" containing the target functions of the calculated structures as well as the constraints violations.
- The redundant angle constraint file "er2a.aco" used for the REDAC cycle.

For a calculation with three REDAC cycles and more minimization steps one could write:

```
redac kt 0.8,0.6,0.4,0.0,0.0 50 iter=300,800,1200
```

For more details on the REDAC strategy please refer to Güntert & Wüthrich (1991).

Running a parallel calculation

INCLAN is able to distribute a calculation over different processors of a shared memory parallel computer by virtue of parallel **do** loops. These differ from ordinary loops only by the presence of the keyword **parallel**:

```
nproc=8
do i 1 20 parallel
  ...
end do
```

Parallel execution of a loop is accomplished by creating (through the **Unix** system call "fork()") several copies of the program in its current state. These copies are identical except for the value of the loop counter (the variable **i** in the above example) and run in parallel. Except for one instance of the program (the "main process") all copies terminate after the execution of the last statement in the loop body.

The special variable **nproc** defines the maximal number of processes running in parallel. By default, the value of **nproc** is 1, i. e. it is necessary to explicitly set this variable to a value larger than one in order to execute a loop in parallel.

There is no mechanism within the program to return data from inside a parallel loop to the program, i. e. in general each iteration of a parallel loop must create an output file.

As an example, a parallel version of the macro **calculate** from a previous section without using the **calc_all** command is:

```
## calc_para - calculate a group of structures
##
```

```
## Usage: calc_para file=<file> [struct=<n>]

syntax file=* struct=@i=5

readdata $file
random_all $struct              Create struct random structures

do i 1 struct parallel                    Parallel loop
  structure copy i 0       Copy structure i to active structure
  anneal                     Minimize with standard protocol
  write cor {$file}$i(I3.3).cor             Save structure
end do

do i 1 struct                 Serial loop to collect structures
  read cor {$file}$i(I3.3).cor              Read structure
  structure copy 0 i                 Copy to structure i
end do
overview $file
```

The call

```
nproc=3
calc_para helix
```

will then perform the structure calculation in parallel on up to three processors.

Some DYANA commands, for example **calc_all**, are executed implicitly in parallel if the **nproc** variable is set to a value larger than one. Therefore, without any change already the simple **calculate** macro of the previous section can perform the same parallel computation as the **calc_para** macro.

Handling groups of structures

Besides the *current structure* (structure 0), the program DYANA can store a number of other structures (structures 1, …, *N*). The maximal number of structures, *N*, that can be stored depends on the size of the structure and is given by the function **maxang**. In general, a structure is stored in the form of all dihedral angle values. However, some commands (e.g. **rmsd**) require direct access to the Cartesian coordinates of the structures. For this reason, for a limited set of structures both, the dihedral angle values and the Cartesian coordinates are stored. The maximal number of structures for which Cartesian coordinates can be stored is given by the function **maxcor**.

In a DYANA calculation, most operations (e.g. minimization) are performed on the current structure. The **structure copy** command can be used to save the current structure:

```
do i 1 10                                   Loop over 10 structures
  random                                    Create random structure
  anneal        Apply simulated annealing protocol to current structure
  structure copy 0 i          Copy current structure to structure i
end do
```

Structures can then be sorted by their target function values with the command **structure sort** and listed with **structure list**:

```
Structural statistics:

str   target  upper limits  lower limits van der Waals torsion angles
    function  #  sum   max  #  sum   max  #  sum   max   #  sum   max
  1     0.12  0  0.7  0.14  0  0.1  0.09  0  0.4  0.13   0  0.0  0.01
  2     0.49  1  1.5  0.33  0  0.0  0.03  0  1.3  0.25   0  0.1  0.06
  3     0.28  1  1.2  0.21  0  0.5  0.17  0  0.7  0.13   0  0.1  0.04
  4     0.42  0  1.3  0.18  0  0.3  0.16  0  1.2  0.15   0  0.1  0.12
  5     0.44  0  1.2  0.20  0  0.1  0.10  0  1.4  0.20   0  0.1  0.03

Ave     0.35  0  1.2  0.21  0  0.2  0.11  0  1.0  0.17   0  0.1  0.05
+/-     0.13  0  0.3  0.06  0  0.1  0.05  0  0.4  0.05   0  0.0  0.04
Min     0.12  0  0.7  0.14  0  0.0  0.03  0  0.4  0.13   0  0.0  0.01
Max     0.49  1  1.5  0.33  0  0.5  0.17  0  1.4  0.25   0  0.1  0.12
```

For each structure its number, target function value and statistical measures for restraint violations are given.

Structures can be selected or deselected using the command **structure select**. Most commands that act on groups of structures apply only to the selected structures. The function **selected(i)** can be used to check whether structure *i* is selected.

As seen in the first section of this chapter, there are several macros ending with "_all" that perform actions on a group of selected structures, e.g. "**calc_all 5**" calculates 5 structures and stores them as structures 1–5, "**write_all** *filename*" writes all selected structures to disk, "**read_all *.cor**" reads all files with the extension ".cor" and stores them as structures 1, 2, …

Handling stereospecific assignments

Even though the chemical shifts of two diastereotopic protons or methyl groups (e.g. $H^{\beta 2}$ and $H^{\beta 3}$ in Tyr) can usually be distinguished, it is not always possible to obtain stereospecific assignments. In such cases the usual strategy consists of provisionally assigning each one of the shifts to one of the diastereotopic partners. The uncertainty of the assignment is then considered by the **distance modify** command which corrects (i.e. loosens) the corresponding distance constraints to allow for both assignments (Wüthrich *et al.,* 1983; Güntert *et al.,* 1991a).

By default, the **distance modify** command assumes that none of the diastereotopic partners are stereospecificly assigned. Therefore, all ste-

reospecifically assigned atom pairs should be declared *before* **distance modify** is called. This is done with the **atom stereo** command, e.g.

```
atoms stereo HA1 22 30 38
atoms stereo HB2 2 5 6 7 14 16 19 20 24 25 35 37
atoms stereo HD2 35 40
atoms stereo QD1 24
```

A list of all diastereotopic partners with and without stereospecific assignment can be obtained with

```
atoms stereo list
```

Calibrating NOEs

Calibration, i.e. the conversion of peak intensities into distance constraints, has become very versatile in DYANA. Peaks from the peak list are selected with any criterion (command **peak select**) and then calibrated with any monotonically decreasing function (command **calibrate**). You can therefore define your own calibration classes and calibration functions.

A macro **caliba** performs a standard calibration of the current peaks using three different calibration classes: One for NOEs assigned to backbone protons, one for NOEs assigned to the more flexible side-chain protons and one for NOEs assigned to methyl groups. The calibration functions used for these three classes are $V = A/r^6$, $V = B/r^4$, $V = C/r^4$ where V is the peak volume and r is the corresponding distance. The parameters A, B and C are either given by the user or calculated automatically.

Given a proton list called “my_prot.prot” and a peak list called “my_peaks.peaks”, the peaks can be calibrated automatically:

```
read prot my_prot
read peaks my_peaks assigned integrated
caliba
```

The simple *automatic calibration* is useful if no preliminary structures are available. It sets the parameter A such that the average upper distance limit for the backbone calibration class becomes 3.6 Å. The parameters B and C are then calculated such that the calibration curves intersect at the minimally allowed upper distance limit (usually 2.4 Å). Intersection points at higher distances would not make sense as the “unphysical” calibration functions of the type “$1/r^4$” should account for flexibility and therefore always result in a higher distance limit for the same peak volume.

The calibration curves given by the automatic calibration can be refined manually: For instance, to tighten the calibration for the backbone calibration class from the automatically determined value, $A = 2.2\ 10^8$, to $A = 1.2\ 10^8$, the command

```
caliba bb=1.2E+8
```

can be used. If a peak volume and the corresponding upper distance limit are given, the peaks of the backbone calibration class can be calibrated accordingly with the **caliba** macro:

```
volume=0.6E+6
d=2.4
caliba bb=volume*d**6
```

For the calibration of multiple peak lists there are two different approaches. The first one treats every peak list separately:

```
read prot first
read peaks first assigned integrated
caliba
read prot second
read peaks second assigned integrated
caliba
...
```

In the second approach peak lists are read with different, user-defined relative weights for the peak volumes and then calibrated simultaneously:

```
read prot first
read peaks first assigned integrated
read prot second
read peaks second weight=0.3 \
           assigned integrated append
...
caliba
```

Because of the option **append** in the second **read peaks** command, the second peak list is appended to the first peak list.

This approach has the disadvantage that the weights must be specified by the user.

Experienced DYANA users may want to create their own calibration classes or use different calibration functions. Two examples illustrate this:

To use the "uniform average model" (Braun *et al.*, 1981) for all NOEs involving methyl groups, one first selects the corresponding peaks and then applies the uniform average calibration function weighed with the parameter *C* (given by the user):

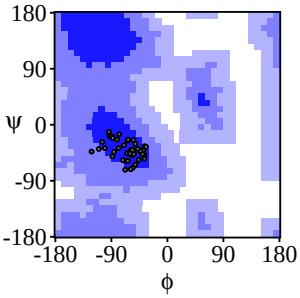
```
peaks select METHYL, *
calibrate C*(1.9**(-5)-d**(-5))/(d-1.9)
```

To calibrate all HN–HN peaks with a function A_1/d^6 with the exception of the NOEs observed in or to a long and flexible loop from residue 12 to 26 (which are calibrated with A_2/d^4), use:

```
peaks select HN 12..26, HN
calibrate A1/d**4
peaks select HN, HN xor
calibrate A2/d**6
```

The logical **xor** operator is used in the second **peaks select** command to select all HN–HN peaks except those that were already selected.

Making plots



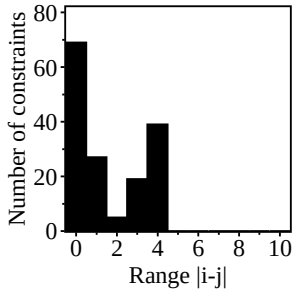
GRAF, a part of INCLAN, is a versatile tool to produce graphics both in Postscript and FrameMaker (MIF) format.

DYANA provides several commands to create standard plots. For example, the following commands create a Ramachandran plot for the group of structures calculated in section "Running DYANA" at the beginning of this manual:

```
readdata helix
read_all helix*.cor
ramachandran rama.ps
```

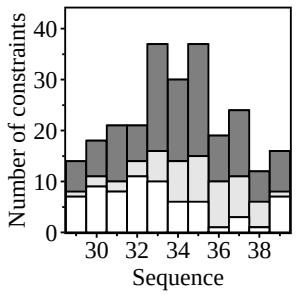
The result is a Postscript file with name "rama.ps". An additional command creates an equivalent output file "rama.mif" in FrameMaker (MIF) format:

```
ramachandran rama.mif
```



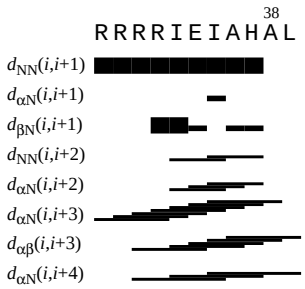
Two plots of the distribution of distance constraints as a function of their range (i. e. the residue number difference) and their residue numbers, respectively, are created by the commands:

```
readdata helix
dcostat dco
```



that create a Postscript file “dco.ps”. In the plot against the sequence, upper distance limits are classified according to their range, *R*:

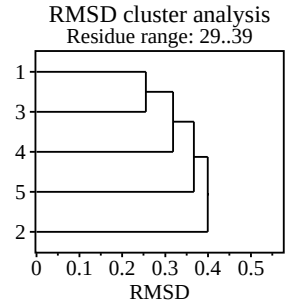
white	intraresidual constraints ($R = 0$)
light grey	sequential constraints ($R = 1$)
dark grey	medium-range ($R < 5$)
black	long-range ($R \geq 5$; not present in this example)



A FrameMaker (MIF) version of a plot of the short- and medium range upper distance limits against the sequence that is often used to identify secondary structure elements is created by

```
readdata helix
seqplot seq.mif
```

which creates a MIF file “seq.mif” that can be imported into FrameMaker in order to add other data such as amide proton exchange rates etc.



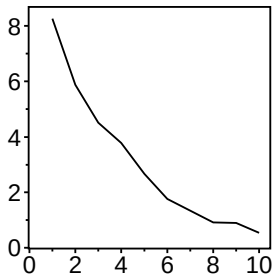
A RMSD cluster analysis that can detect whether structures are clustered in groups into distinct regions of conformation space is performed by

```
readdata helix
read_all helix*.cor
cluster cluster
```

The result is a Postscript file with name “cluster.ps”. The plot shows a clustering tree. Along the vertical axis the structures are listed, ordered according to the clustering found. On the horizontal axis the minimal RMSD between any two structures in the clusters combined so far is shown.

In addition to standard plots, GRAF can be used to produce general graphics, for instance plots made from tabulated data. Assume that we are given a table of values:

```
 1  8.27
 2  5.88
 3  4.51
 4  3.78
 5  2.66
 6  1.76
 7  1.34
 8  0.91
 9  0.89
10  0.54
```

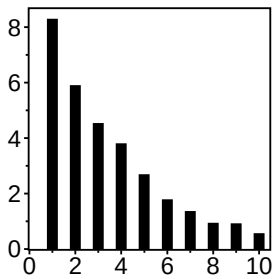


To produce a plot of this data, the table is stored in a GRAF file, i. e. a file with extension ".grf", called "curve.grf", and supplemented with the appropriate GRAF commands:

```
 1  8.27
...
10  0.54
frame
line
```

This GRAF file can be converted into in a Postscript file, "curve.ps", with the command

```
graf curve
```

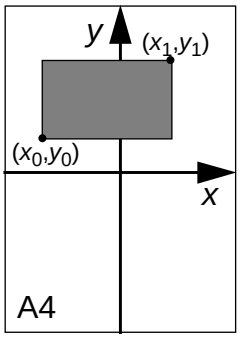


Similarly, a histogram of the same data can be produced by replacing the last line of the file "curve.grf" with

```
fill=1
rectangle x-0.2 0 x+0.2 y1
```

"fill=1" selects pattern 1 (solid) to fill the interior of polygons, and "rectangle x-0.2 0 x+0.2 y1" has the meaning: "plot for every data point (x, y) in the table of values a rectangle with lower left corner $(x - 0.2, 0)$ and upper right corner $(x + 0.2, y)$."

In the above examples, the plots had the default size and position, and the scales for the *x*- and *y*-axes have been chosen automatically by GRAF. However, the user can also specify these parameters explicitly.



The positioning of a plot on the paper is governed by the four parameters **x0**, **y0**, **x1**, and **y1** that specify the positions of two reference points, (x_0, y_0) and (x_1, y_1), in a coordinate system centered in the middle of an A4 sheet, with axes running in the (mathematically) usual directions and using typographical points (1 pt = 0.353 mm) as units. By default, the two reference points specify a large square placed in the center of an A4 sheet: $x_0 = y_0 = -250$ and $x_1 = y_1 = 250$.

The user can choose another coordinate system by specifying four parameters **X0**, **Y0**, **X1**, and **Y1** that define the coordinate values of the two reference points in the new coordinate system. By default, GRAF chooses after reading a table of data points a coordinate system such that all data points lie within the rectangle defined by the reference points.

Using INCLAN variables

In INCLAN, string variables can be used in a similar way as in a Unix shell:

```
dyana> name:=Dyana
dyana> print "My name is $name."
My name is Dyana.
```

"**:=**" performs an assignment, **$***variable* substitutes the value of a *variable* into the command line.

In addition, variables with numeric values can be used in expressions in the same way as in FORTRAN or other programming languages:

```
dyana> x=7
dyana> y=5*x
dyana> z=sqrt(y-10.0)
dyana> show x y z
    x = 7
    y = 35
    z = 5.00000
```

(**show** is an INCLAN command that displays the values of variables.) Here a different assignment sign, "**=**" instead of "**:=**", was used. Assignments with "**=**" have the meaning: "Evaluate the expression on the right hand side and assign the result value to the variable." Note the difference to a string assignment with "**:=**":

```
dyana> y:=5*x
dyana> show y
    y = 5*x
```

Expressions formed according to the rules of FORTRAN-77 may contain integer, real and complex numbers, logicals, and character strings. Within expressions character strings must be enclosed in single quotes:

```
dyana> s:=Dyana
dyana> l=lenstr(s)
*** ERROR: Illegal expression "lenstr(s)".
```

is an error because the variable s does not contain a quoted string (**lenstr** is an INCLAN function that returns the length of a string, i. e. the index of its last non-blank character). The correct use of simple, unquoted strings in an expression is:

```
dyana> l=lenstr('$s')
dyana> show l
    l = 5
```

Single quotes do not inhibit variable substitutions.

Using INCLAN control statements

INCLAN provides a full set of control statements to direct the program flow. These are used mainly in macros, i. e. in collections of INCLAN statements that form new commands which can be used in the same way as basic commands. Since control statements are not used interactively, the program prompt ("dyana>") will no longer be shown.

Commands can be executed conditionally by virtue of the **if** statement which has the same form as in FORTRAN-77:

```
if (i.gt.20) print "i is larger than 20."
if (i.lt.0) then
  print "i is negative."
else if (i.lt.10 .and. mod(i,2).eq.0) then
  print "i is less than 10 and even."
else
  print "i is none of the above."
end if
```

Alternatively, comparison and logical operators can also be given in the form of the C programming language:

```
if (i>20) print "i is larger than 20."
if (i<0) then
  print "i is negative."
else if (i<10 && mod(i,2)==0) then
  print "i is less than 10 and even."
else
  print "i is none of the above."
end if
```

Repeated execution of commands is achieved by forming loops with the **do** statement. Loops executed a predefined number of times have an integer loop variable:

```
do i 1 20
  print "i = $i"
end do
```

Here, the loop variable, **i**, runs from 1 to 20 in steps of 1.

A loop that is executed until a termination condition is met can be constructed as follows:

```
do
  ...
  if (x.gt.100.0 .or. finished) break
  ...
end do
```

The **break** statement exits from a loop.

Unconditional jumps are possible by virtue of the **go to** statement:

```
   do i 1 n
     ...
     if (err) go to cleanup
     ...
   end do
   ...
cleanup: print "Error in the loop."
   ...
```

The **go to** statement transfers the program flow to the position indicated by the label (“cleanup”).

Creating non-standard residues

Non-standard residue types can be added to the residue library as additional entries. The procedure to add a new residue type to the library is as follows (see section “File formats” for a description of the format of the residue library file):

Create Cartesian coordinates for all atoms of the residue, for example with a molecular graphics program or using the attach and insert commands of the program COFIMA. Bond lengths, bond angles, and chiralities of this structure must be correct but the conformation, i.e. the values of the dihedral angles, does not matter. The coordinates of the overlap atoms at the beginning and at the end of the residue (for example N, CA, and C in amino acids) will also be needed. If the new residue type results from a slight modification of an existing residue type, it is usually most

convenient to start from the coordinates of the existing residue type and to modify them. Order the atoms such that their order is compatible with the tree structure of dihedral angles that will be defined, i.e. such that the following two rules are fulfilled:

- A change of a dihedral angle must not affect the positions of the first, second, third, or forth atom in any preceding dihedral angle definition.
- The set of atoms whose positions will be affected by a change of a dihedral angle consists of all atoms following the third atom in the dihedral angle definition up to the fifth (last) atom in the dihedral angle definition (or the end of the main chain for backbone dihedral angles).

Convert the coordinates into the format of the library file (for example with a text editor). Add atom types, connectivities, and the information about diastereotopic partners. Add the dihedral angle definitions to the new entry. These two steps are best done using the library format in which connectivities and angle definitions are given by atom names rather than by atom numbers (see the option **convert** of the **read lib** command). Make sure that the header line starting with RESIDUE is correct. Add the new entry to (a copy of) the residue library file. Test the new entry, for example in the following way:

- Create a sequence file that contains the new residue type, preferably in the interior of the chain, i.e. not as the first or last residue.
- Using this sequence file and the new residue library in the program DYANA, create angle and coordinate files for a conformer with randomized dihedral angles.
- Start DYANA again (with the same sequence and residue library file), read the previously produced coordinate file, and write again angle and coordinate files without making any minimization.
- Check whether the angles and coordinates produced by the second run of DYANA coincide closely with those from the first run. If this test fails, then there is probably a format error in the new library entry or the ordering rules listed above are violated. However, this test does not detect errors in nomenclature, connectivities, or pointers to pseudo atoms.
- Check the coordinates produced by DYANA on a molecular graphics system, for example with the program MOLMOL (Koradi *et al.*, 1996).

Working with several molecules

DYANA allows for calculations with more than one molecule through the use of special linker residues. These “invisible” linkers consist exclusively of pseudo atoms, i.e. they can penetrate the “real” molecules without causing any steric repulsion, and thus allow the program to formally treat a system of several molecules in the same way as a single molecule.

Each of the linker residue types in the standard library has one rotatable bond. The residues are:

- **PL** (**PLM**) to link an amino acid residue to a generic linker
- **NL** (**NLM**) to link a nucleotide residue to a generic linker
- **LL**, **LL2** and **LL5** (**LLM**, **LLM2** and **LLM5**), generic linker residues with 1, 2 and 5 Å bond lengths, respectively, and 90^o bond angles
- **LP** (**LPM**) to link a generic linker to a following amino acid residue
- **LN** (**LNM**) to link a generic linker to a following nucleotide residue

There are two forms of each linker residue type: The normal form is used for minimization and in TAD calculations with spherical inertia tensors (the default). In TAD calculations with inertia tensors directly derived from the atomic masses and positions the forms given in parentheses must be used.

A sufficient number of these linker residues must be used between two molecules such that no artificial constraint on the relative positioning of the two molecules with respect to each other is introduced by the finite length and flexibility of the stretch of linker residues.

To treat, for example, a system consisting of a double-stranded DNA of residues 1–14 and 101–114 and a protein starting with residue 200, the sequence file could look like this:

```
GUA   1 ADE ADE ADE GUA CYT CYT ADE THY
        THY ADE GUA ADE GUA                1st DNA strand
NL   50 LL LL LL LL LL LL LL LL
        LL LL LL LL LL LN                      linkers
CYT 101 THY CYT THY ADE ADE THY GUA GUA
        CYT THY THY THY CYT                2nd DNA strand
NL  150 LL2 LL2 LL2 LL2 LL2 LL2 LL2
        LL2 LL2 LL2 LL2 LL2 LL2 LP             linkers
ALA 200 LEU ...                              protein
```

Automatic NOESY assignment (NOAH)

NOAH (Mumenthaler & Braun, 1995; Mumenthaler *et al.*, 1997) is an algorithm for the automatic assignment of 2D and 3D NOESY spactra. In iterative cycles, new possible assignments are identified and tested through a structure calculation. Alternative assignment possibilities for individual peaks are included simultaneously in these calculations, and peaks are unambiguously assigned after the structure calculation only if the distance constraint from one of the assignment possibilities was clearly less violated in the structures.

NOAH was implemented in DYANA at two different levels. First, new commands were implemented for NOAH specific tasks (**assign**, **create**, **filter**, **keep**, **reliability**, **write ass**) and new variables were introduced for NOAH-specific parameters (**tolerance**, **tol_una**, **tol_transp** etc.).

Second, a **noah** macro was written which contains the NOAH schedule. Two additional macros, **noahmin** and **noahanneal** are called by the **noah** macro itself, and should not be modified by the user.

The following sections give some practical advice on the use of NOAH/DYANA.

The mandatory input is a peak list containing peak positions and volumes in XEASY format, and a list of chemical shifts in XEASY format ("proton list"). NOAH can only yield good results if the resonance assignment is complete, i.e. if all or nearly all chemical shifts are assigned, and if the chemical shifts given in the proton list agree with corresponding peak positions within a small tolerance range Δ_{tol} (in general: ±0.01 ppm in 2D and ±0.02 ppm in 3D). This is best achieved by manually assigning at least one NOESY peak for every proton shift, e.g. by taking over the TOCSY peak list from the sequential assignment and overlaying it to the NOESY spectrum. All assigned peaks in the input peak list will be regarded as "safe". NOAH will include them in every structure calculation and never delete or change them, even if they give rise to large constraint violations.

Additional input will help NOAH to converge. Dihedral angle constraints may be generated using coupling constants with the **grid** commands and known disulfide bridges should be included as upper and lower limit distance constraints (use the DYANA commands **ssbond** and **write upl** to generate and save these distance constraints).

Checking input for NOAH

Before runing NOAH, the input data should be checked for obvious inconsistencies using some of the DYANA commands. First, no such warning messages should appear when loading a proton list:

```
dyana> read prot kt
*** WARNING: Inconsistency for LYS+ 21:
             QB 1.617, HB2 1.618, HB3 1.580
*** WARNING: Inconsistency for LYS+ 21:
             QG 1.774, HG2 1.424, HG3 1.741
Chemical shift list "kt.prot" read,
457 chemical shifts.
```

In the above example, the pseudo atom and the two protons it represents were assigned simultaneously. To prevent NOAH (and also the user) to assign peaks to the pseudo atom, the chemical shift of the pseudo atom should be set back to "999.000" in the proton list.

Second, a rough estimate of all missing chemical shifts can be obtained with the command **atom shifts missing**:

```
dyana> atom shifts missing
    Residue     missing shifts
    ASP-    1   HN
    THR     4   HG1
    SER    11   HB3 HG
    HIS    14   HD1
    THR    15   HG1
    MET    16   HG3
    TYR    19   HH
    GLN    21   HG3
    TYR    25   HH
    THR    27   HG1
    THR    32   HG1
    THR    33   HG1
    PRO    40   HG3
    91.7 % assigned, 14 missing chemical shifts.
```

The above listing is usual for what we call a “nearly completely assigned” proton list. Most of the missing proton shifts are those of labile sidechain protons which are not always observable.

The chemical shifts of the proton list can be checked with **atom shifts check**:

```
dyana> atom shifts check
    Atom            shift  limit1 - limit2
    HB2   TYR   11   1.123   4.100   1.620
    QE    PHE   21   7.566   7.510   5.560
    CG    LYS+  23  29.777  26.440  20.900
    CB    ALA   39  25.942  24.200  14.500
    HA    GLU-  47   5.784   5.550   2.840
    HB2   PHE   52   1.163   3.920   1.400

    Atom  Residue    Shift  Median   Spread Peaks
    HN    ASN    3   8.671   8.662   0.014      3
    HA    ILE   29   4.346   4.344   0.013     12
    HG2   GLU-  31   2.326   2.319   0.015      9
    HG2   GLN   32   1.049   1.044   0.012      9
    HG3   GLU-  35   1.819   1.822   0.014      5
    HD2   LYS+  57   1.697   1.689   0.019      7
    HE2   LYS+  57   3.121   3.114   0.016      7
    QG    PRO   63   2.124   2.120   0.018      4
    10 shifts with spread larger than tolerance.
```

If the library “dyana.lib” was used (which contains a table with statistical distributions of chemical shifts), NOAH will print a list of all shifts from the proton list which are higher or lower than the highest and lowest value ever observed for that particular proton/hetero atom. A few shifts may deviate from these values, but they should be checked carefully.

If some of the NOESY peaks are assigned, the consistency between peak and proton list can be checked. All spreads above the value of the variable **tolerance** between the peaks assigned to the same proton are listed. A large spread may indicate that a peak is assigned to the wrong proton.

The command **peaks deviation** checks the deviation between the proton shifts and the peak position of all assigned peaks. This command is of direct interest for NOAH users as a peak that deviates by more than Δ_{tol} (variable **tolerance**) from its chemical shifts can not be assigned correctly by NOAH. Consequently, the variable tolerance should be set to the value you inted to use for the NOAH calculation:

```
dyana> tolerance:=0.01,0.01
dyana> peak deviation
      Peak Dim Deviation Atom  Residue
       453  1   -0.017   QG2   ILE   75
       528  1    0.011   HN    VAL   87
      1779  1    0.013   QG1   VAL   38
      1939  1    0.010   HB    ILE   52
      2219  1    0.017   HB    ILE   75
    5 deviations larger than tolerance.
```

All these peaks should be checked and modified *before* running NOAH. If the assignment is correct, the peak position should be shifted to the intersection point of both proton shifts.

Running NOAH

A typical NOAH script ("NOAH.dya") can be found in the "noah" example directory of the DYANA distribution package:

```
dyanalib
prot_nam := "er2"
read seq $prot_nam.seq
read aco $prot_nam.aco
tolerance := 0.01,0.01,0.3
tol_una   := 0.01,0.01,0.3
tol_transp:= 0.03,0.03,0.6

seed=3771
info := normal
# nproc = 6
rmsd_range:=3..37

./ssa
atoms stereo list

noah num=24 peak_nam=er2_h2o_na,er2_d2o_na \
     rmsd=$rmsd_range protein=$prot_nam \
```

```
proton_nam=h2o,d2o \
addupl=ssbond addlol=ssbond \
peak_ref=er2_h2o,er2_d2o calibrate
```

First, the protein sequence and the predefined angle constraint files are loaded. Then, the NOAH variables **tolerance**, **tol_una** and **tol_transp** are set. The variable **tol_transp** is only used for 3D spectra to check for transposed peaks. If the script will run on a multi-processor machine, the line "**nproc**=6" may be uncommented. If some stereo-specific assignments are available, they may be included into a macro called "ssa.dya" and read before the **noah** macro is started.

The **noah** macro will perform 24 cycles (**num**=24) and assign the two unassigned peak lists "er2_h2o_na.peaks" and "er2_d2o_na.peaks". The residue range used for the RMSD calculation is 3–37 and the protein name, which is used as output file name, is "er2". The two proton lists used to assign the peaks lists are called "h2o.prot" and "d2o.prot" (**proton_nam**) and the disulfide bridges of E*r*-2 are included as upper (**addupl**) and lower (**addlol**) limits distance constraint files "ssbond.upl" and "ssbond.lol". Finally, the two assigned reference peak lists are given (**peak_ref**) and the calibration option is turned on. The reference peak lists are only used to give an overview during the NOAH run.

In principle, NOAH can be used for two different tasks: (1) Continue to assign a NOESY peak list or (2) Re-assign a peak list to check differences in the assignment made manually and automatically. In case (1), no reference peak lists may be given (parameter **peak_ref** must be deleted). In case (2), an unassigned version of the peak list must be created first, e.g.

```
read prot h2o
read peaks er2_h2o
peaks assignment delete
write peaks er2_h2o_na
```

For 3D lists, the procedure is very similar. The format of your peak lists should be included into the peak list itself with the line "#DYANAFORMAT *format*" (see command **read peaks**) or as a separate parameter **plformat** into the **noah** call (e.g. "**plformat**=hHN,hHC").

Analyzing NOAH output

NOAH will produce the following important files:

- **er2.cor**: The coordinate files of the ten best NOAH structures.
- **er2.ovw**: Overview file of these 10 conformers.
- **noah.ps**: PostScript file with summary of the NOAH run.

- **noah.grf**: Graf file for “noah.ps” containing RMSD and target function values after each cycle as well as the number of assigned peaks.
- **noah.*x*.peaks**: Assigned peak lists
- **incomp.*x*.peaks**: Peak list with peaks that are incompatible with the final structure bundle.
- **reliability.*x***: Reliability distance of every assigned peak.
- **diff.*x***: Differences in the assignment compared to the reference peak list (only if such a peak list was indicated with the parameter **peak_ref**).
- **end.upl** / **end.lol**: Upper and lower limit distance constraint files used for the final calculation.

“*x*” is the peak list number ($x = 1, \ldots, n$). The file “noah.ps” is generated by INCLAN using the “noah.grf” file which contains all numbers (RMSD, target function values and assigned peaks):

```
# Range for RMSD calculation: 3..37
# Cycle, RMSD(all), RMSD(bb), TF(1), TF(10)
  0  6.32  5.25    161.3   190.3
  1  6.18  5.11    120.0   155.8
  2  6.30  5.36    126.9   157.5
  3  6.06  5.13    127.5   160.0
  4  5.40  4.29      8.9    24.5
  5  5.93  4.82    104.4   136.7
  6  5.65  4.71    100.1   146.8
  7  5.49  4.52    108.9   146.3
  8  4.55  3.52     12.7    27.9
 ...
 20  1.94  1.34      3.0     6.1
 21  1.33  0.82      3.7     6.5
 22  1.37  0.89      4.6     7.0
 23  1.39  0.93      2.1     4.8
 25  1.46  0.92      1.6     2.9
[...]
# Number of assigned peaks
# Total, new, different
 1 372 21 0
 2 416 26 0
 3 433 29 0
 4 441 32 1
 5 449 33 1
 6 501 39 1
 7 505 40 1
...
20 705 93 4
21 703 94 5
22 715 97 5
23 719 98 5
24 720 101 5
25 708 100 4
```

In every fourth cycle (i.e. cycles 4, 8, 12, 16, 20 and 24) NOAH uses only the unambiguous and the ambiguous assignment lists, but not the test assignment list which contains most errors. Accordingly, the target function value is usually much lower in these cycles (see above) while the RMSD of the resulting structure bundle may be higher (for more information on the internal peak lists of NOAH see command **filter** and Mumenthaler & Braun (1995)).

In the example above, 708 peaks were assigned in the H_2O peak list. 100 of these peaks were not assigned in the reference peak list (and are thus "new"), and only 4 of 708 peaks were differently assigned in the reference peak list. The "noah.grf" file does also contain the corresponding numbers for the D_2O peak list (not shown here).

Another important value is found in the files "reliability.x" (where "x" is again the number of the peak list). These files contain the reliability of every individual assignment as well as a statistic on all peaks which are incompatible with the final structures, i.e. where no possible assignment within the given tolerance range is compatible (has a distance < 5 Å) with at least one structure:

```
General reliability of structures:

Peaks with no ass. possibility
  because of chemical shift  : 91
Unassigned peaks             : 242
Incompatible peaks         : 38(61 per structure)

Histogram of displacements needed
to make all peaks compatible:
  0 -  1 A :   5
  1 -  2 A :   4
  2 -  3 A :  10
  3 -  4 A :   2
  4 -  5 A :   2
  5 -  6 A :   4
  6 -  7 A :   5
  7 -  8 A :   2
  8 -  9 A :   3
  9 - 10 A :   1
```

From the above example we see that 91 peaks have no assignment possibility at all because there is no proton chemical shift within the allowed tolerance range from the peak position. Furthermore, there are 38 peaks which have some assignment possibilities, but all of them are incompatible with the current structure bundle. 5 of these peaks could be explained by a slight shift of up to 1 Å between the two protons of one assignment possibility.

These incompatible peaks are very interesting since an optimal solution

together with an ideal peak list should have no incompatible peaks. In practice, there will always be some incompatible peaks for one or several of the following reasons:

(1) Some noise and spin diffusion peaks have been picked.

(2) Some proton chemical shifts are missing in the proton list, and the NOESY peaks originating from these protons cannot be explained.

(3) The structures are partly distorted and can therefore not explain several NOEs.

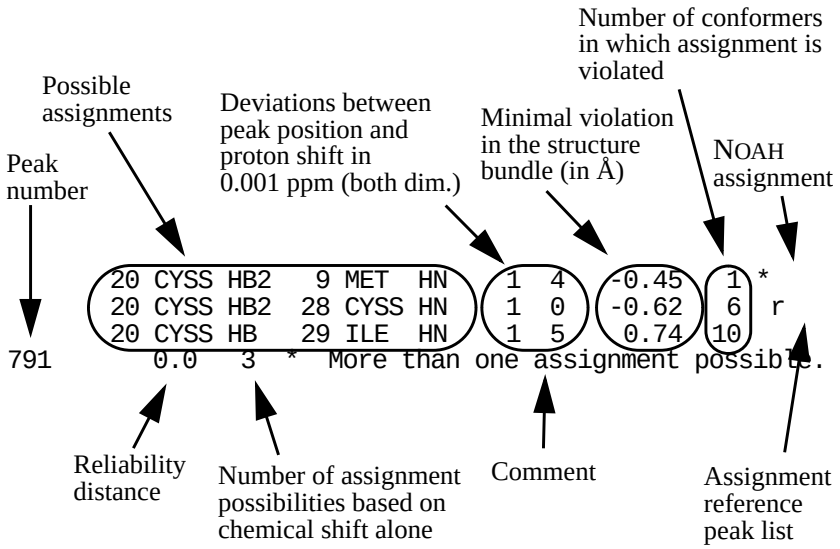
Experience has shown that the structures will often be distorted in proportion to the number of incompatible peaks with significant bias when the percentage of incompatible peaks from the whole peak list is much larger than 5%. This can be explained by the fact that NOAH tries to find an assignment for all peaks. If there are too many noise peaks, NOAH may well find a distorted structure which explains another 95% of the peaks, i.e. scores equally well than the real structures in explaining as many peaks as possible (see below).

For this reason, NOAH saves all peaks it has found incompatible into the peak lists "incomp.x.peaks". These (usually small) peak lists should be examined carefully with the spectra (with XEASY, for example). Typically, many of these peaks can be identified as noise peaks and should be eliminated from the peak list, because they disturb the NOAH calculations. For the others, the spectroscopist may search for a previously unassigned proton, specially if several incompatible peaks lie on the same shift. Once an improved peak list (and proton list) becomes available, a new NOAH calculation should be performed until the percentage of incompatible peaks reaches 1–2%.

Reliability of NOAH assignments

The files "reliability.x" contain the reliability distance of every assigned peak. All assignment possibilities which have a minimal violation of less than *r* Å in the structure bundle are listed. The parameter *r* can be given to the command **reliability** and has a default value of 1.0 Å. The signif-

icance of every entry is illustrated in the following figure:



In the above example, NOAH has assigned the peak to a different proton pair than the one in the (manually assigned) reference peak list. However, the both assignments seem to be compatible with some of the structures in the structure bundle and the peak consequently received a reliability distance of 0 Å.

An example output illustrates some different possibilities:

```
-------------------------------------------------------
   26 MET  HN   28 CYSS HN       1    0   -0.92   0 *r
   26 MET  HN   29 ILE  HN       1    5   -0.61   0
794       0.0   3     More than one assignment possible.

-------------------------------------------------------
   32 THR  HA   32 THR  HN       0    0   -2.23   0 *r
814     100.0   1
-------------------------------------------------------
    9 MET  HA   32 THR  HN       6    0   -0.58   1
   32 THR  HB   32 THR  HN       9    0   -1.51   0 *
816       0.0   2     More than one assignment possible.
-------------------------------------------------------
   34 ASP- HN   32 THR  HN       2    1   -1.28   1 *r
817       8.0   2
-------------------------------------------------------
    6 GLU- HA    8 ALA  HN       3    0   -0.68   0 *r
824       3.4   2
```

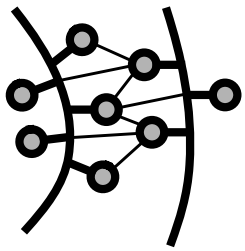
Peak 794 has two assignment possibilities, but NOAH and also the spectroscopist assigned it to a short-range NOE. Peak 814 has only 1 possible assignment based on the chemical shifts alone, and this assignment is automatically fulfilled because it is a intraresidual fixed distance. It has therefore a reliability distance of 100 Å. Peak 816 was not assigned by

the spectroscopist (the "r" is missing), probably because it is an overlap of the two possible NOEs. Peaks 817 and 824 have high reliability distances and should be safe.

Experience has shown that the reliability distance is quite efficient in identifying uncertain assignments. Over 75% of the peaks that were differently assigned by NOAH than by the spectroscopist had a reliability distance of 0 Å in a recent study (Mumenthaler et al., 1997). Therefore, you may want to use the command **keep** (after the command **reliability**) which will delete all assignments with a reliability distance equal or below a user-given threshold.

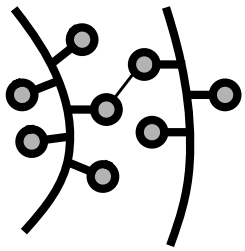
Identifying "dangerous" NOAH assignments

When analyzing the NOAH output one must keep in mind that the elimination of erroneously assigned constraints through contradiction with correct constraints will in general be less efficient in regions of low NOE density, such as chain ends, surface loops, or the periphery of long side chains than in the well defined protein core.



Well-supported long-range NOEs

The final distance constraint list should therefore be checked by the command **distance check** that calculates a score for every long-range distance constraint. High scores indicate that there are many other long-range distance constraints between the two residues (or residues close to them) that support the given distance constraint. A score of zero indicates that there is no other NOE observed between the two regions of interest. This is not only very suspect, but also quite dangerous because such a single long-range NOE may have dramatic effects on the structure. All peaks which cause such "dangerous" long-range distance constraints should be checked manually directly in the spectra.



Single long-range NOE that is not supported by other NOEs to adjacent residues

In the following example, the long-range NOE from residue 3 to 62 has a score of 0.0 because there is absolutely no other NOE supporting it, while it seems unlikely that all the NOEs observed between residue 6 and 57 are derived from wrong assignments:

```
dyana> distance check
    Distance constraint                          Score
    Upper QE    LYS+   3 - QD    LYS+  62   0.00
    Upper HA    TYR    6 - QB    ALA   53   2.00
    Upper HB2   TYR    6 - QD1   LEU   57   3.75
    Upper HB3   TYR    6 - QD1   LEU   57   3.75
    Upper QB    TYR    6 - QB    ALA   53   2.00
    Upper QB    TYR    6 - QD2   LEU   57   4.25
    Upper QD    TYR    6 - HA    ALA   53   2.50
    Upper QD    TYR    6 - QB    ALA   53   1.50
```

INCLAN

The user interface of the program is based on INCLAN, a powerful interactive command language that allows the use of variables, FORTRAN-77 mathematical and character expressions, macros, flow control (loops, conditional statements, jumps), the production of graphics etc.

When reading an input command line the command interpreter executes the following steps:

- An optional comment, i.e. text following a comment sign "#", is discarded.
- The values of variables are substituted from right to left.
- The command line is split into elements (defined as sequences of non-blank characters separated by blank characters). The first element becomes the command name, and the following elements become command parameters.
- If the command name matches a user-defined alias, the alias is expanded.
- If the command name matches a built-in command of INCLAN, it is executed by the command interpreter itself.
- Otherwise, if the command name matches a user-defined command, it is executed by the command interpreter.
- Otherwise, if the command name matches a command of the program unambiguously, it is executed by the program.
- Otherwise, the command interpreter looks for a macro with the given command name and, if it is found in the current macro search path, executes it. If no such macro is found, an error occurs.

Special characters

The following characters have a special meaning for INCLAN. To use them literally, they usually must be preceded by a backslash.

$** "**$*variable*" substitutes the value of the *variable* in the command line. Substitutions proceed from left to right. If the value of the variable or function call starts and ends with single quotes (i.e. if it is a FORTRAN-77 character string), the delimiting single quotes are removed before inserting the value.

% "**%***variable*" substitutes the value of the *variable* in the command line. Substitutions proceed from left to right. Single quotes that delimit FORTRAN-77 character strings are retained.

{ } The curly braces in "**{$***variable***}**" or "**{%***variable***}**" separate the variable name *variable* from immediately following text. "**${***expression***}**" or "**%{***expression***}**" substitute the result value of the FORTRAN-77 *expression*.

( ) "**$***variable***(***format***)**" uses the given FORTRAN-77 *format* to convert the numeric value of a variable into the string that is substituted in the command line. If the value of the *variable* is a comma-separated list, "**$***variable***(***n***)**", where *n* is an integer expression, substitutes with the *n*-th element of this list. "**$***variable***(***m***:***n***)**", where *m* and *n* are integer expressions, substitutes with the substring between positions *m* and *n* of the value of the *variable*. These three possible uses of parentheses cannot be used simultaneously.

; separates commands that stand on the same line. Note, however, that commands that form blocks (e.g. **do** . . . **end do**, **if** . . . **end if**) must always appear as the first command on a line.

: "*Label***:**" denotes a label that can be used as the target of a jump in a **goto** statement.

**** "*****c*" treats the character *c* literally and allows the use of special characters in normal text, "****" at the end of a line indicates that the statement continues on the following line.

" **"***text***"** treats *text* as a single parameter, even if it contains spaces. Variable substitutions in the *text* still occur.

' **'*text*'** treats *text* as a single parameter; the single quotes remain part of the text. Single quotes are used to delimit FORTRAN-77 character string constants. Variable substitutions in the *text* still occur.

Text between a comment sign "**#**" and the end of the line is treated as a comment and skipped by the program.

@ Commands preceded by "**@**" are only echoed if the variable **echo** has the value **full**. "**@**" has its special meaning only if it occurs as the first character of a command.

! "**!***string*" recalls the last interactive command that started with *string*. "**!**" has its special meaning only if it occurs as the first character of a command.

^ "**^***string***^***replacement***^**" executes the last interactive command again after replacing the first occurrence of *string* by *replacement*. The third caret is optional unless the *replacement* string has trailing blanks. "**^**" has its special meaning only if it occurs as the first character of a command.

Variables

The command line interpreter allows the use of variables in two different ways:

- Similar to shell-variables in the UNIX operating system as variables whose value can be substituted into the command line. In this case, the value of a variable is a general character string and has no particular type.
- As variables in FORTRAN-77 expressions. In this case, the value of a variable must be an integer, real, complex, logical or character constant, according to the rules of FORTRAN-77. In particular, character strings must be delimited with single quotes.

Variables can be used in both ways simultaneously which makes them a powerful tool of the command language.

A variable *name* consists of up to 32 letters, digits, or underscore characters "_". The *value* of a variable is always stored as a character string and only converted temporarily to an integer, real, or complex number during the evaluation of a FORTRAN-77 expression.

There are several types of variables:

Local variables exist only within the macro where they are declared, and in macros called from this macro. With the exception of the command line parameters of a macro, which are always local, local variables must be declared in **var** or **syntax** statements. They exist until they are removed with **unset** or until the end of the macro in which they are declared is reached.

Global variables are always visible, except when they are hidden by local variables with the same name. Variables that are not local are global. The user can introduce new global variables simply by using a variable with a new name. Global variables exist until they are removed with **unset**.

Special variables are variables that can be created and used by the user but have also a special meaning to the command interpreter.

System variables are variables that are used and, possibly, set by the program (not exclusively by the user with **eval**, **set** etc.). System variables are always global.

There are several ways to insert the value of a variable or the result value of an expression into the command line:

Basic substitutions Substitutions of the form **$***variable* or **%***variable* insert the complete value of the variable (without trailing blanks) into the command line. Substitutions with "**$**" differ from those with "**%**" only if the value of the variable starts and ends with single quotes, i.e. if it is a FORTRAN-77 character constant: with "**%**" the delimiting single quotes are retained in the substitution, with "**$**" they are removed. A variable name that is immediately followed by a letter, digit, or underscore character must be enclosed in curly braces: "**{$***variable***}**".

```
x:=4.6; y:=2.0; sum=x+y; t:=a sum          Set variables
print "This is $t: $x + $y = $sum"         Substitute values
This is a sum: 4.6 + 2.0 = 6.60000

s:='$t'          Create a FORTRAN-77 string from a normal variable
print "\$s = $s; \%s = %s"                 With and without single quotes
$s = a sum, %s = 'a sum'

print "{$t}mer"
a summer
```

Fortran format Substitutions of the form **$***variable***(***format***)** or **%***variable***(***format***)** are used to format integer or real values of variables according to a FORTRAN-77 format. A *format* that contains the letter "I" or "i" applies to in-

teger numbers, all other *formats* to real numbers.

```
x:=4.6; y:=2.0; sum=x+y
print "$x + $y = $sum(E12.3)"
4.6 + 2.0 =    0.660E+01
```

Substring

Substitutions of the form **$***variable***(***n***:***m***)** or **%***variable***(***n***:***m***)**, where *n* and *m* are positive integer expressions, are used to substitute with the substring between character positions *n* and *m* of the value of a *variable*. Substring expressions can also appear on the left hand side of assignment statements.

```
t:=a sum
print "another $t(3:5)"
another sum
t(3:):=program                    Assignment to a substring
print "$t"
a program
```

List element

If the value of a *variable* is a comma-separated list, "**$***variable***(***n***)**" or "**%***variable***(***n***)**", where *n* is a positive integer expression, substitute with the *n*-th element of this list.

```
s:=17,28,,56,"This is the end"
do i 1 length('s')       length returns the number of elments
  print "Element $i: $s(i)"
end do
Element 1: 17
Element 2: 28
Element 3:
Element 4: 56
Element 5: This is the end
```

Function call

"**$***function*" or "**%***function*" substitute with the result value of a *function* without parameters, "**$***function***(***parameters***)**" or "**%***function***(***parameters***)**" substitute with the result value of a *function* with *parameters.* If there are several *parameters*, they are separated by commas.

```
x=2.5; print "log(x)= $log(x)"
log(x) = 0.916291
```

Expression

"**${***expression***}**" or "**%{***expression***}**" substitute with the result value of an *expression*.

```
x=2.5; y=10.0; print "x/y = ${x/y}"
x/y = 0.250000
```

All substitutions in the command line proceed from right to left. This allows, for example, to compose a variable name from the values of other variables before it is used in a substitution.

```
command list_param          User-defined command list_param
  do i 1 nparam
    print "Parameter $i: $p$i"
                    $p$i inserts the value of the i-th command line
                    parameter.
  end do
end

list_param 17 second last                    Call list_param
Parameter 1: 17
Parameter 2: second
Parameter 3: last
```

Special variables

The following variables have a special meaning for the command interpreter:

echo determines which commands are echoed, i.e. copied to standard output before execution. The possible settings are:

- **NULL** (or not set at all) In macros, all commands except those built into the command line interpreter are echoed; interactive commands are not echoed.
- **off** Commands are not echoed.
- **on** Both in macros and interactively, all commands except those built into the command line interpreter are echoed.
- **large** Same as **on**, except that the echo is surrounded by blank lines.
- **full** All commands are echoed, and the corresponding line numbers in macros are given.
- **OFF** Same as **off**, except that this setting can only be overridden by another value written in capital letters.

ON Same as **on**, except that this setting can only be overridden by another value written in capital letters.

LARGE Same as **large**, except that this setting can only be overridden by another value written in capital letters.

FULL Same as **full**, except that this setting can only be overridden by another value written in capital letters. This setting is particularly useful for debugging macros in which the echo is suppressed.

Labels are not included in the echo, but variable substitutions are. Statements preceded by "@" are only echoed if **echo** has the value **full** or **FULL**.

erract is a variable for error handling in macros. If an error occurs within a macro, the value of **erract** is executed as a command. By default the **exit** command is executed, i.e. the program returns to interactive input. Errors that occur interactively are displayed and the program continues with the execution of the next statement.

```
set erract="show; quit"
```

In case of an error in a macro a listing of all global variables is given, and the program is stopped. Such error handling can be useful if the program is used non-interactively.

info determines which messages are written to standard output and into the protocol file. The possible settings are:

none No messages are written.

minimal A minimal set of messages is written, in general a single line for each command that is executed.

normal The "normal" amount of messages is written.

full The "full" amount of messages is written.

debug The "full" amount of messages and additional undocumented messages for debug purposes are written.

Optionally, this variable may have two of the above values, separated by a comma. In this case, the first value applies to standard output, the second to the protocol file.

nparam denotes the number of command line parameters of the current macro.

nproc denotes the maximal number of processors that is used for parallel do-loops.

p1, **p2**, . . . are the default names for the command line parameters of a macro. These names may be changed at the beginning of the macro.

path denotes the search path for macro files in the form of a comma-separated list of directories.

prompt denotes the prompt for interactive input. If this variable is not defined or blank, no prompt is written but multiple blank lines of input and the end of the execution of a macro are indicated by the word “Ready” on a separate line.

protocol denotes the name of the protocol file into which standard output is duplicated under the control of the variable **info**. If this variable is not defined or blank, no protocol file is written.

timing is a system variable to control the reporting of CPU times. CPU times are given for all commands (except for those that are built into the command line interpreter) that need more seconds of CPU time than the value of **timing** indicates.

Expressions

The command interpreter can evaluate general FORTRAN-77 integer, real, complex, logical and character expressions. Expressions can appear in **eval** statements, as conditions of **if** statements, as command parameters when a numeric value is expected, and as substring and element index expressions.

An expression is built according to the rules of FORTRAN-77 from constants, variables, and function calls. These basic items can be combined by operators (“**+**”, “**–**”, “*****”, “**/**”, “******”, “**.eq.**”, “**.ne.**”, “**.lt.**”, “**.le.**”, “**.ge.**”, “**.gt.**”, “**.and.**”, “**.or.**”, “**.not.**”, “**.eqv.**”, “**.neqv.**”, “**==**”, “**!=**”, “**<**”, “**<=**”, “**>=**”, “**>**”) and grouped by parentheses.

There are the following differences to the rules of FORTRAN-77:

- The data type “double precision” is not supported.
- The data type “logical” is represented by the integer values 0 (false) and 1 (true). Any integer expression can be used in place of a logical expression, with 0 representing “false”, and all other values representing “true”.
- Variable, function and operator names are case sensitive. The names of logical operators and intrinsic functions must be written in lower case.
- The logical operators “**==**”, “**!=**”, “**<**”, “**<=**”, “**>=**”, “**>**”, “**&&**”, “**||**”, and “**!**” can be used in place of its respective FORTRAN-77 equiva-

lents "**.eq.**", "**.ne.**", "**.lt.**", "**.le.**", "**.ge.**", "**.gt.**", "**.and.**", "**.or.**", and "**.not.**".

- All FORTRAN-77 intrinsic functions (except "dble", "dprod", "lge", "lgt", "lle" and "llt") are available by their generic names but not under special names. For example, the absolute value function is known by the name "**abs**" but not by the special names "iabs" or "cabs".
- There are additional intrinsic functions (see below).
- Blanks can only appear at "reasonable" places but not inside of numbers, variable names etc.

Intrinsic functions

In the following list of all INCLAN intrinsic functions, arguments are denoted by

n	integer
r	real
c	complex
s	string
x	integer or real, unless types are given explicitly
z	real or complex

The result type of an intrinsic function is only given explicitly, if it differs from the type of the argument(s).

abs(*x*) Absolute value; the argument *x* is of any numeric type, for complex arguments the result is real.

acos(*r*) Arc cosine; $|r| \le 1$, $0 \le \mathrm{acos}(r) \le \pi$.

aimag(*c*) Real function that returns the imaginary part of *c*.

aint(*r*) Discard fractional part; the result if of type real.

anint(*r*) Closest integer; the result if of type real.

asin(*r*) Arc sine; $|r| \le 1$, $-\pi/2 \le \mathrm{asin}(r) \le \pi/2$.

atan(*r*) Arc tangent; $-\pi/2 \le \mathrm{atan}(r) \le \pi/2$.

atan2(r_1**,**r_2**)** Argument of the complex number $r_2 + ir_1$ (not $r_1 + ir_2$!); r_1 and r_2 must not both be zero, $-\pi \le \text{atan2}(r_1, r_2) \le \pi$.

char(*n***)** Character function that returns the character with number *n*.

cmplx(x_1**,**x_2**)** Complex function that returns $x_1 + ix_2$; both arguments must have the same type.

conjg(*c***)** Complex conjugate.

cos(*z***)** Cosine.

cosh(*r***)** Hyperbolic cosine.

cputime Real function that returns the CPU time (in seconds) since the start of the program.

date Character function that returns the current date in the form *dd–mm–yy*.

def(*s***)** Logical function that returns 1 if a variable with name *s* exists and has a value different from **NULL**, or 0 otherwise.

dim(x_1**,**x_2**)** Positive difference; $\dim(x_1, x_2) = \max(x_1 - x_2, 0)$.

exist(*s***)** Logical function that returns 1 if a variable with name *s* exists, or 0 otherwise.

existfile(*s***)** Logical function that returns 1 if a file with name *s* exists, or 0 otherwise.

exp(*z***)** Exponential function.

external(*s***)** Character function that returns the value of the external (i.e. non-local) variable with name *s* (even if it is hidden by a local variable with the same name), or a blank string if no external variable with this name exists.

external(s_1**,**s_2**)** Character function that returns the value of the external (i.e. non-local) variable with name s_1 (even if it is hidden by a local variable with the same name), or s_2 if no external variable with the name s_1 exists.

fitchisq Real function that returns the χ^2 value of the last linear least-squares fit (see plot subcommand **fit**).

fiterr(*n*) Real function that returns the standard deviation of the *n*-th fit parameter of the last linear least-squares fit (see plot subcommand **fit**).

fitpar(*n*) Real function that returns the optimal value of the *n*-th fit parameter of the last linear least-squares fit (see plot subcommand **fit**).

fitprob Real function that returns the probability that the χ^2 value of the last linear least-squares fit would be exceeded by chance (see plot subcommand **fit**).

getenv(*s*) Character function that returns the value of the environment variable with name *s*.

getpid Integer function that returns the UNIX process identification number of the current process.

global(*s*) Character function that returns the value of the global variable with name *s* (even if it is hidden by another variable with the same name), or a blank string if no global variable with this name exists.

global(s_1,s_2) Character function that returns the value of the global variable with name s_1 (even if it is hidden by another variable with the same name), or s_2 if no global variable with the name s_1 exists.

ichar(*s*) Integer function that returns the number of the character *s*.

if(n,x_1,x_2) Function that returns the argument x_1 if $n \neq 0$, or x_2 otherwise. The arguments x_1 and x_2 can have any type.

index(s_1,s_2) Integer function that returns the starting position of the first occurence of the string s_2 in the string s_1, or zero if s_2 does not occur as a substring in s_1.

indexr(s_1,s_2) Integer function that returns the starting position of the last occurence of the string s_2 in the string s_1, or zero if s_2 does not occur as a substring in s_1.

int(*z*) Integer function that returns the integer part of the real or complex number z.

len(*s*) Integer function that returns the number of characters in *s*.

length(*s*) Integer function that returns the number of elements in the array stored in a variable with name *s*.

lenstr(*s*) Integer function that returns the index of the last non-blank character in *s*.

log(*z*) Natural logarithm; $z \neq 0$, if *z* is real it must be positive, for complex *z* the result has $-\pi < \text{Im}\ \log(z) \leq \pi$.

log10(*z*) Logarithm to base 10; $z \neq 0$, if *z* is real it must be positive, for complex *z* the result is in the range $-\pi < \text{Im}\ \log10(z) \leq \pi$.

macro(*s*) Logical function that returns 1 if a macro with name *s* is available, or 0 otherwise.

match(s_1,s_2) Wildcard match; logical function that returns 1 if the string s_2 matches the string s_1, or 0 otherwise. The string s_2 may contain wildcards: an asterisk matches zero or more characters, and a question mark matches exactly one character.

max(x_1,x_2,...) Maximum.

min(x_1,x_2,...) Minimum.

mod(x_1,x_2) Remainder of x_1 modulo x_2; $\text{mod}(x_1, x_2) = x_1 - x_2 \cdot \text{int}(x_1 / x_2)$, both arguments must have the same type, $x_2 \neq 0$.

mtime(*s*) Integer function that returns the time of last modification (in seconds since a reference date) of the file with name *s*.

nint(*r*) Integer function that returns the integer closest to *r*.

opened(*s*) Logical function that returns 1 if a file with name *s* is currently open, or 0 otherwise.

plotx0, **ploty0**, **plotx1**, **ploty1** Real functions that return the coordinates of the two reference points (X_0, Y_0) and (X_1, Y_1) in the user coordinate system used for graphics (see plot parameters **X0**, **Y0**, **X1**, **Y1**).

rand Real function that returns a pseudo-random number; pseudo-random numbers are uniformly distributed between 0 and 1.

rand(*n*) Real function that returns a pseudo-random number; pseudo-random numbers are uniformly distributed between 0 and 1. The random number generator is initialized with the seed *n*.

rand(n_1,n_2) Real function that returns a pseudo-random number; pseudo-random numbers are uniformly distributed between 0 and 1. The random number generator is initialized with the seed n_1, and the result is the n_2-th random number generated from this seed.

real(*x*) Conversion to real type; the argument *x* must be of type integer or complex, for complex *x* the real part is returned.

sign(x_1,x_2) Returns the absolute value of x_1 times the sign of x_2; if $x_2 = 0$, its sign is taken as positive, both arguments must have the same type.

sin(*z*) Sine.

sinh(*r*) Hyperbolic sine.

sqrt(*z*) Square root; if *z* is real, it must be non-negative.

tan(*z*) Tangent.

tanh(*r*) Hyperbolic tangent.

time Character function that returns the current time in the form *hh*:*mm*:*ss*.

val(*s*) Character function that returns the value of the variable with name *s*, or a blank string if no variable with this name exists.

val(s_1,s_2) Character function that returns the value of the variable with name s_1, or s_2 if no variable with the name s_1 exists.

walltime Integer function that returns the number of seconds since the start of the program.

Macros

Macros are files containing INCLAN statements. A macro is called by its

name that is identical to its filename except for the extension ".dya" that is required for macro files. INCLAN looks for macro files in the directories given by the special variable **path**, or in the explicitly given directory. Command line parameters may be passed into a macro. Within the macro, they are available as local variables that are by default called **p1**, **p2**, ... These variable names can be changed with the **parameter** statement. The local variable **nparam** denotes the number of command line parameters. Macros can be called from within other macros. On-line help information may be included into a macro as lines that start with two comment signs "##". Such lines are copied to standard output when one requests help about a macro with the command **help** *macro*.

The special macro **init** is an initialization macro that is automatically executed when the program starts. Typically, this macro sets the system variable **path** that defines the search path for macro files.

Standard output

This section explain the ways by which commands can write output to the standard output device (in the following simply called "screen") and/or to disk files by using the protocol mechanism or output redirection. The concepts of this section do not apply to output that is written to explicitly named disk files by specific output commands.

Information level

All output has an importance level, and only output that is "important enough" is actually written. The definition of what is "important enough" is given by the special variable **info** that can, in its simple form, take one of five *information level* values:

none	no output at all, except for error messages
minimal	minimal output, in general a one line confirmation
normal	the "normal" amount of output
full	detailed output
debug	additional undocumented debugging output

Protocol file

The output can be duplicated into a protocol file. In fact, different **info** values might be used for output to the screen and to the protocol file. In this case, the info value consists of two simple info values, separated by a comma. A protocol file is written if the **protocol** variable is defined and has a non-blank value that is the name of the protocol file. If the file does not exist when the **protocol** variable is set to the corresponding name, it is created; otherwise the output is appended to an existing pro-

tocol file.

```
protocol:=logfile          Open protocol file "logfile"
info:=minimal,full         Minimal screen output, full protocol
...
protocol:=                 Close protocol file
```

Output redirection

Output from a command is redirected to a given file if the last parameter of the command is

- *>file* Redirect to a new *file*, or overwrite existing *file*. After writing the output, the file remains open.
- *>file.* Redirect to a new *file*, or overwrite existing *file*. After writing the output, the file is closed.
- *>>file* Append to an existing *file*, or create new *file*. After writing the output, the file remains open.
- *>>file.* Append to an existing *file*, or create new *file*. After writing the output, the file is closed.

Blanks between **>** and *file* are not allowed and that the file name must not end with "**.**". The file name is optional; if it is omitted, the output will be redirected to the previously used *file*. When redirection is used, all output that would otherwise be sent to the screen is written to the given *file*. Standard output and the protocol file are not used.

Built-in commands

The following commands are built into the command interpreter. Their names cannot be abbreviated.

alias

[*name statement*]

Defines a new alias *name*, i.e. an abbreviation, for the given *statement*. The *statement* may contain an asterisk "*" to indicate where the command line parameters are to be inserted. Without parameters, **alias** gives a list of all currently defined aliases.

```
alias ? "print \"\%{*}\""      Simulate a pocket calculator
? 5*7
35
```

ask

prompt variable . . .

Writes the string *prompt* to standard output, reads one line from standard input, and assigns from this line strings separated by blanks to the given variables. The command is usually used for interactive input within macros. A *prompt* that contains blanks must be enclosed in double quotes.

```
ask "First and last point:" begin end
First and last point:
12 45
print "range = $begin...$end"
range = 12...45
```

break

Breaks a do-loop and is only allowed in macros. The execution of the macro is continued with the first statement following the loop.

command

[*name*]

Defines a new globally visible user-defined command within a macro, i.e. a macro within a macro. User-defined commands defined by **command** statements are called by their *name*, possibly followed by parameters, in exactly the same way as macros. Within a macro, a user-defined command can only be called after it was defined. The statement *command* without parameters gives a list of all user-defined commands, and indicates where they are defined.

do

(without parameters) Executes a loop within a macro. The loop is executed unconditionally, i.e. until one of the statements **break, exit, quit** or **return** is encountered.

```
do
  if (filename.eq.' ') break
  ...
end do
```

do

variable start end [*step*] [**parallel** [**continue**]]

Executes a FORTRAN-77 do-loop within a macro. The loop counter *variable* and the integer expressions *start, end,* and *step* have the usual meaning. **Parallel** loops are executed in parallel on **nproc** processors. If the keyword **continue** is present, the program continues immediately with the execution of the next statement after the parallel loop. Otherwise, the

next statement after the loop is executed when the parallel loop is finished.

```
do i 1 10
  print "Iteration $i."
end do
```

else

Starts an else clause of a block if-statement.

else if

(*condition***) then**

Starts an else-if clause of a block if-statement.

end

Ends a user-defined command or subroutine.

end do

Ends a do-loop.

end if

Ends a block if-statement.

error

text

Writes the *text* to standard output or into the file with the given *filename* and calls the error handler. This statement is suitable to treat errors that occur during the execution of a macro. If the *text* contains blanks it must be enclosed in double quotes.

eval

variable **=** *expression*

Evaluates the arithmetic or string *expression* according to the rules of FORTRAN-77 and assigns the result to the *variable.* The keyword **eval** can be omitted. In contrast to FORTRAN-77 function names must be given in lowercase letters.

```
eval i = 7
sentence = 'A flexible program!'
j = mod(i,4)**2
l = len(sentence)
show i sentence j l
... Variables:
    i        = 7
    sentence = 'A flexible program!'
```

```
j       = 9
l       = 19
```

external

variable **=** *expression*

or

variable **:=** *value*

assigns a *value* (i. e. a string) or the result of an *expression* to an external (non-local) *variable* even if a local variable with the same name exists. This command can be used to return values from a macro to the calling macro.

```
command swap a b                    Command to swap two variables
  var x y                           Declare two local variables, x and y
  x=$external('$a')                 Get value of external variable with name $a
  y=$external('$b')                 Get value of external variable with name $b
  external $a=y                     Assignment to external variable with name $a
  external $b=x                     Assignment to external variable with name $b
end

x=10; y=5
print "Before swap: x = $x, y = $y"
Before swap: x = 10, y = 5
swap x y
print "After swap : x = $x, y = $y"
After swap : x = 5, y = 10
```

exit

Returns from a macro to interactive input. Given interactively, it exits from the program.

go to

label

continues execution of a macro at the first line that begins with the *label*. Jumps into loops (**do** . . . **end do**) or conditionally executed statements (**if** . . . **else** . . . **end if**) are not allowed and can lead to unpredictable results. A *label* may consist of letters, digits, and underscore characters "_". A label must be followed by a colon.

```
go to cont
  ...
cont: print "Now at label cont."
```

help [*topic*]

Gives on-line help for a given *topic*. With no *topic* given, a list of all available help topics is displayed. On-line help for macros can be included in the macro: **help** *macro* shows all lines of the *macro* that start with "##".

if **(***condition***)** *statement*

Executes a logical "if" statement as in FORTRAN-77, i. e. the *statement* is executed if the logical expression *condition* is true. A line with a logical "if" statement must not end with the word **then**.

```
i=-56
if (i.lt.0) print "$i is negative."
-56 is negative.
```

if **(***condition***) then**

Executes a block-"if" statement, as in FORTRAN-77.

```
if (mod(i,2).eq.1) then
  print "$i is an odd number."
else if (def('x') .and. exist('y')) then
  print "x is defined, and y exists."
else if (s.eq.' ') then
  print "The variable s is blank."
end if
```

parameter *variable* . . .

Changes the names of the parameters that are passed to a macro; i. e. the parameters **p1**, **p2**, . . . get the names given in the **parameter** statement. The **parameter** statement must precede all other statements in a macro (except **var)** and cannot be used interactively.

plot *subcommand* [*parameter* . . .]

Performs a plot subcommand. Plot commands are described separately in the "Graphics" section of this chapter.

print

text [**level=***level*]

Writes the *text* to standard output or into the file with the given *filename.* If the *text* contains blanks it must be enclosed in double quotes. Optionally, the importance **level** of the output can be defined. By default, the importance level is **normal**.

quit

Exits from the program.

readline

file variable [**close**]

Reads one line from a *file* and assigns it to a *variable*. If the file is not yet open, it is opened and the first line is read. If the file is already open, the next line is read. If the end of the file is reached, the variable is set to **EOF** and the file is closed. Optionally, the file can be **close**d after reading a line.

remove

file . . .

Removes one or more disk files.

return

exits from the current macro and returns to the calling macro or, if the macro was called interactively, to interactive input. Given interactively, **return** exits from the program.

set

variable **=** *value*

or, if the keyword set is omitted

variable **:=** *value*

assigns a *value* (i. e. a string) to a *variable.*

```
set i=456
j := 2 + i          Short form of set assigns a string value
k = 2 + i           Short form of eval evaluates an expression
set i j k
i = 456
j = 2 + i
k = 458
```

set

variable . . .

Displays values of *variables*. If no *variable* is specified, all variables that have values different from **NULL** are displayed. If the names of one or several *variables* are given, the values of these variables are displayed.

show

variable . . .

Displays the values of all or selected *global* variables. If no *variable* is specified, all global variables that have values different from **NULL** are displayed. If the names of one or several global *variables* are given, the values of these variables are displayed.

sleep

t

Waits for *t* seconds.

subroutine

name

Defines a new user-defined command within a macro, i.e. a macro within a macro. User-defined commands defined by **subroutine** statements are called by their *name*, possibly followed by parameters, in exactly the same way as macros. User-defined commands defined by a **subroutine** statement are local to the current macro (or macros called through it). Within a macro, a user-defined command can only be called after it was defined.

syntax

format . . .

Analyzes the command line parameters of the current macro. This statement can only be called within a macro. Command line parameters that match with one of the *format* specifications are removed from the list of command line parameters and assigned to a new local variable.

The possible *format* items are:

*name***=[=]***type*[**=**default]

Declares a named parameter with the given *name*, *type* and, optionally, *default* value. If the *default* value is ab-

sent, the parameter is required, and an error will occur if the parameter is not specified in the macro call.

The optional second "**=**" sign after the *name* indicates that a parameter that matches *name* but does not contain an "**=**" sign is not recognized, otherwise (with only one "**=**" sign after *name*), an error occurs in this situation.

A local variable with the given *name* is created, and either the value specified by the user, or, in its absence, the *default* value is assigned to it. The value must be compatible with the given *type* (see below).

In a macro call, a named parameter can either be specified anywhere in the parameter list in the form "*name***=***value*" or as a positional parameter of the form "*value*" at the same position in the parameter list as the corresponding *format* in the **syntax** statement. Only parameters that appear before "*****" or "******" (see below) can be specified as positional parameters without giving their *name*.

A *name* may contain an asterisk "*" to indicate how much it can be abbreviated. By default, all unambiguous abbreviations are allowed. If a *name* starts with an asterisk, then the corresponding parameter is a positional parameter that cannot be given in the form "*name***=***value*".

name Declares a literal option with the *name*. A local variable with the given *name* is created. If the option *name* is present in the macro call this variable is set to 1 (i.e. the logical value "true"), otherwise it is set to 0.

$name_1$**|**$name_2$ . . .
Declares a set of mutually exclusive literal options with the names $name_1$, $name_2$, etc. Local variables with the given *names* are created. If one of the option names is present in the macro call, the corresponding variable is set to 1 (i.e. the logical value "true") and the other variables are set to 0.

****** Allows for additional parameters that do not match with one of the *formats.*

***** Has the same meaning as "******" except that additional parameters must not contain an "**=**" sign.

Formats must not contain blanks.

A *type* can be one of the following:

* Any character string.

@i Integer expression.

@r Real expression.

[*l*<[=]]@**i**[<[=]*u*]

[*l*<[=]]@**r**[<[=]*u*]

Integer or real expression in the given range.

@ii Integer range, i.e. one of the following:

m	a single integer expression
m..*n*	two integer expressions
m..	using the default value for *n*
..*n*	using the default value for *m*

$name_1$|$name_2$ . . .

List of mutually exclusive literals.

@f.*extension* Filename that will be extended with the given *extension*, if necessary (*extension* can also be **$***name* to denote the value of a preceding parameter).

```
command read_file
  syntax format=asc|bin file=@f.$format \
         weight=@r=1.0
```

The command **read_file** has three parameters. The first parameter (**format**) is required and can either be **asc** or **bin**, the second parameter (**file**) is also required and is a filename that will be given the extension **.asc** or **.bin**, depending on the chosen format, and the third parameter (**weight**) is an optional real number with default value 1.0.

```
  ...
end

read_file asc test
```

Positional parameters and default value for **weight**. Equivalent to setting **format=asc**, **file=test.asc** and **weight**=1.0.

```
read_file file=test format=asc weight=2.0
```

Named parameters in any order.

system

[*UNIX-command*]

Executes a *UNIX-command* by invoking a shell. If no command is specified, an interactive shell is started.

type

macro

displays the macro or user-defined command with the given *name*. Macros in the current path can be listed without giving a path; otherwise the path has to be specified.

unset *variable . . .*

Removes one or more variables.

var *variable . . .*

declares *variables* as local variables of the current macro. In contrast to normal (global) variables, local variables are only visible within the macro where they are declared and within macros that are called via that macro (except when such a macro declares itself a local variable with the same name). The **var** command must precede any other commands in a macro (except the **parameter** command) and cannot be used interactively.

Graphics

With INCLAN it is possible to produce graphical output in either Postscript of FrameMaker (MIF) format. Graphics is created with the built-in command **plot**. The **plot** command can either be invoked directly, or plot subcommands can be combined with list data in graphics files that can be read with the **plot file** command.

A graphics file can contain one or several blocks of *list data*, i.e. matrices of integer or real numbers in free format. Each row (line) of a list data block must have the same number of entries. The columns of a list data block form vectors called $\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2$,... If a list data block consists of a single column with n numbers, this column is called $\boldsymbol{y}_1$ and an $\boldsymbol{x}$-column with values $1, 2, \ldots, n$ is added implicitly. After reading a block of list data, the graphics system is in *list mode*, and various plot subcommands can be applied to vector expressions formed from the column vectors of the list data block. These vector expressions are general FORTRAN-77 expressions that are evaluated for all vector elements and where the column vectors $\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2$,... are denoted by **x**, **y1**, **y2**,...

Besides list data, a graphics file can contain plot subcommands (and comments starting with **#**) but not other commands; it is not an INCLAN macro.

The following alphabetical list contains all plot subcommands. They are called from INCLAN in the form

plot *subcommand parameters*

and in graphics files in the form

subcommand parameters

Some of the plot subcommands have different parameters in normal and list mode as indicated by "(normal mode)" or "(list mode)" at the right margin.

arc

x y a [*b* [ϕ_1 ϕ_2]]

draws a circle, an ellipse, or part of a circle or ellipse with the center at (x, y), and half axes *a* and *b*. If *b* is omitted, a circle with radius *a* (measured in the *x*-direction) is drawn. Optionally, only the part of the ellipse starting and ending with phase angles ϕ_1 and ϕ_2, respectively, is drawn. The phase angle is 0° on the positive *x*-axis and increases counterclockwise. This command can also be used in list mode, where the parameters are vector expressions.

caro

See section ***mark***.

clip

x_1 y_1 x_2 y_2

draws a rectangle with corners (x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2) and sets the current clipping path to its border. Subsequent drawing commands will only draw within this rectangular area.

clip

off

resets the clipping path. After this command, graphics will no longer be confined to the rectangular area specified in a previous **clip** command.

close

closes the current output plot file.

comment *text*

writes *text* as a comment into the output plot file.

cross See section ***mark***.

curve $x_1\, y_1\ \ x_2\, y_2\ \ x_3\, y_3\, x_4\, y_4 \ldots$ (normal mode)

draws a Bézier spline curve defined by the points (x_i, y_i) . The total number of points must be $3n + 1$, with integer $n \geq 1$. The resulting curve passes through the points 1, 4, 7,...; the other points guide the curve. Four points define the shape of each segment of the curve: The curve segment leaves (x_1, y_1) along the direction of the straight line connecting (x_1, y_1) with (x_2, y_2) and reaches (x_4, y_4) along the direction of the straight line connecting (x_3, y_3) with (x_4, y_4) . The lengths of the lines connecting (x_1, y_1) with (x_2, y_2) and (x_3, y_3) with (x_4, y_4) represent, in a sense, the "velocity" of the path at the endpoints. The curve segment is always enclosed by the convex quadrilateral defined by the four points.

curve [[***x***] ***y***$_1$. . .]. (list mode)

draws Bézier spline curves through the points of the given vector expressions ***x***, ***y***$_1$,... If no vector expressions are specified, splines are drawn through the points of all list columns. If the ***x***-expression is omitted (i.e. if only a single expression, ***y***$_1$, is given), the *x*-coordinates are taken from the *x*-column of the list. The number of list points must be $3n + 1$, with integer *n*.

dot See section ***mark***.

errorbar $x\ \ y_1\, y_2$

draws an errorbar defined by the given *x*- and *y*-coordinates. This command can also be used in list mode, where *x*, y_1 and y_2 are three vector expressions.

file *file*

reads an input graphics *file* (default extension: **.grf**) containing list data and plot commands and executes the plot commands in the graphics file. Graphics files cannot be nested. If no output plot file is open when the **file** command is executed, and if the first plot command in the graphics file does not open an output plot file explicitly, a new Postscript output plot file with the name *file***.ps** is opened implicitly. An implicitly opened output plot file will be closed when the end of the graphics file is reached.

fit ***x y*** σ f_1. . . (list mode)

performs a linear least-squares fit of the basis functions given by the vector expressions f_1,... to the data points with x-coordinates, y-coordinates and errors given by the vector expressions ***x***, ***y*** and σ, respectively. For *m* basis functions, $f_1, \ldots, f_m$ the optimal linear combination,

$$y(x) = a_1 f_1(x) + \ldots + a_m f_m(x) \ , \qquad [1]$$

is determined by minimizing

$$\chi^2(a_1, \ldots, a_m) = \sum_i \left(\frac{y_i - y(x_i)}{\sigma_i} \right)^2 \ , \qquad [2]$$

where *i* runs over the list data points. The optimal fit function $y(x)$ is added as another column to the list data. This command does not draw anything. The fit parameters, $a_1, \ldots, a_m$, their standard deviations, χ^2, and the probability that this value of χ^2 would be exceeded by chance are available through the intrinsic functions **fitpar**, **fiterr**, **fitchisq** and **fitprob**, respectively. If the errors σ_i of the data points are unkown, this can be indicated by setting σ to zero in the **fit** command.

```
dot x y1                          Plot original data points
fit x log(y1) 0 1 x               Logarithmic fit of y = a1 exp(–a2x)
spline x exp(y2)                  Plot fitted curve
```

frame

xtext=*xtext*	—
ytext=*ytext*	—
tics=*ticaxes*	x,y
labels=*labelaxes*	x,y
grid zero	

draws a rectangular frame with corners (X_0, Y_0) , (X_0, Y_1) , (X_1, Y_0) and (X_1, Y_1) . Subsequently produced graphics is clipped on the borders of the frame. The *x*- and *y*-axes are labeled with the titles *xtext* and *ytext*, respectively. The parameter **tics** and **labels** determines whether tics and numeric labels are drawn. The possible values for *ticaxes* and *labelaxes* are:

off	No labels or tics.
x	Labels or tics only on the *x*-axis.
y	Labels or tics only on the *y*-axis.
x,y	Label or tics on both axes (default).

If the option **grid** is present, a fine grid is drawn. If the option **zero** is present, fine lines will be drawn along $x = 0$ and $y = 0$ (if they fall within the frame).

function

*f*1. . .

plots the functions given by the expressions $f_1(x)$,...

label

axis position text

labels the given *axis* by placing a tic and the *text* at the given *position*. The parameter *axis* can have the following values:

x or **bottom**	Label the *x*-axis, i.e. the horizontal line at *y*-position Y_0.
y or **left**	Label the *y*-axis, i.e. the vertical line at *x*-position X_0.
top	Label the horizontal line at *y*-position Y_1.
right	Label the vertical line at *x*-position X_1.

If *text* is blank, only a tic is set.

line

$x_1\, y_1\ \ x_2\, y_2 \ldots$ (normal mode)

draws a line that connects the points (x_1, y_1), (x_2, y_2),... by straight line segments.

line [[***x***] ***y***$_1$. . .]. (list mode)

draws straight lines through the points of the given vector expressions ***x***, ***y***$_1$,... If no vector expressions are specified, straight lines are drawn through the points of all list columns. If the ***x***-expression is omitted (i.e. if only a single expression, ***y***$_1$, is given), the *x*-coordinates are taken from the *x*-column of the list.

mark *x y* (normal mode)

where *mark* stands for either **dot**, **square**, **caro**, **plus**, **cross** or **triangle**, marks the position (x, y) with the corresponding symbol. The size of the symbol is determined by the current value of the plot parameter **marksize**.

mark [[***x***] ***y***$_1$. . .]. (list mode)

where *mark* stands for either **dot**, **square**, **caro**, **plus**, **cross** or **triangle**, marks the positions given by the vector expressions ***x***, ***y***$_1$,... with the corresponding symbol. If no vector expressions are specified, all points of the list columns are marked. If the ***x***-expression is omitted (i.e. if only a single expression, ***y***$_1$, is given), the *x*-coordinates are taken from the *x*-column of the list.

mif *file*

opens and initializes an output plot *file* in FrameMaker (MIF) format. If another plot file is open when the **mif** command is executed, it is closed.

plus See section ***mark***.

polygon $x_1\ y_1\ \ x_2\ y_2\ \ x_3\ y_3$. . . (normal mode)

draws a polygon with the edges (x_i, y_i) . At least three points must be specified.

polygon [[***x***] y_1. . .]. (list mode)

draws polygons with the edges given by the vector expressions ***x***, y_1,... If no vector expressions are specified, polygons are drawn through the points of all list columns. If the ***x***-expression is omitted (i.e. if only a single expression, y_1, is given), the *x*-coordinates are taken from the *x*-column of the list. The number of list points must be three or more.

ps *file*

opens and initializes an output plot *file* in Postscript format. If another plot file is open when the **ps** command is executed, it is closed.

rectangle $x_1\, y_1\ \ x_2\, y_2$

draws a rectangle with corners (x_1, y_1), (x_1, y_2), (x_2, y_1) and (x_2, y_2). This command can also be used in list mode, where x_1, y_1, x_2 and y_2 are four vector expressions. In list mode, the command can also be used without parameters. In this case a rectangle with corners $((x_{i-1} + x_i)/2, 0)$, $((x_i + x_{i+1})/2, 0)$, $((x_{i-1} + x_i)/2, y_i)$ and $((x_i + x_{i+1})/2, y_i)$, i.e. a histogram bar, is drawn for each point (x_i, y_i) in the list columns (for the first and last point, x_{i-1} and x_{i+1} are replaced by the minimal and maximal *x*-values, X_0 and X_1, respectively).

scale *axis* ***f***$_1$. . . **exact** *(list mode)*

performs scaling of the given *axis* (**x** or **y**) on the basis of the vector expressions ***f***$_1$,... Scaling sets the coordinates of the reference points in the user coordinate system (X_0 and X_1 for the *x*-axis, and Y_0 and Y_1 for the *y*-axis) such that they include all values of the vector expressions ***f***$_1$,... If the option **exact** is present, then the new coordinates of the reference points will correspond exactly to the minimum and maximum of the vector expressions ***f***$_1$,...; otherwise a small margin will be added in order to avoid that points lie exactly on the boundary.

set *parameter*=*value* . . .

sets one or several plot *parameters* to the given *values*. The keyword **set** is optional.

shape $x_1\, y_1 \;\; x_2\, y_2 \;\; x_3\, y_3 \;\; x_4\, y_4 \;\; x_5\, y_5 \;\; x_6\, y_6 \ldots$ (normal mode)

draws a shape enclosed by a closed Bézier spline curve that is defined by the points (x_i, y_i) . The total number of points must be 3*n*, with integer $n \geq 2$.

shape [[***x***] ***y***$_1$. . .]. (list mode)

draws shapes enclosed by Bézier spline curves through the points of the given vector expressions ***x***, ***y***$_1$,... If no vector expressions are specified, shapes are drawn for all list columns. If the ***x***-expression is omitted (i.e. if only a single expression, ***y***$_1$, is given), the *x*-coordinates are taken from the *x*-column of the list. The number of list points must be 3*n*, with integer *n*.

spline $x_1\, y_1 \;\; x_2\, y_2 \ldots$ (normal mode)

draws a cubic spline through the points (x_1, y_1), (x_2, y_2),... The spline starts at the first point and ends at the last point with vanishing second derivative. The *x*-values must be increasing: $x_i < x_{i+1}$, for all *i*.

spline [[***x***] ***y***$_1$. . .]. (list mode)

draws cubic spline curves through the points of the given vector expressions ***x***, ***y***$_1$,... If no vector expressions are specified, splines are drawn through the points of all list columns. If the ***x***-expression is omitted (i.e. if only a single expression, ***y***$_1$, is given), the *x*-coordinates are taken from the *x*-column of the list.

square See section ***mark***.

text *x y text*

print *text* at position (x, y). The alignment of the text with respect to the reference position (x, y) depends on the current values of the plot parameters **align** and **rotate**. The current values of the plot parameters **font**, **textsize**, **weight** and **angle** define the font used to write the *text*. In addition, the *text* may contain the following embedded text commands:

@T	Change font type to Times.
@H	Change font type to **Helvetica**.
@C	Change font type to `Courier`.
@S	Change font type to Symbol.
@b	Change to **boldface**.
@i	Change to *italics*.
@^	Start a $^{\text{superscript}}$.
@v	Start a $_{\text{subscript}}$.
@N	Return to standard font, end sub- or superscript.

If the text contains multiple blanks, it must be enclosed in double quotes. Double quotes that are part of the text must be preceded by a backslash.

triangle See section ***mark***.

write *text*

writes *text* into the output plot file.

Plot parameters are used to define the positioning and appearance of graphics objects. They are set by the plot subcommand **set**:

align determines how text is aligned with respect to its reference position. Possible values are:

left	The horizontal reference position is at the left margin of the text.
center	The horizontal reference position is in the center of the text.
right	The horizontal reference position is at the right margin of the text.
bottom	The vertical reference position is at the bottom margin of

the text.

middle The vertical reference position is in the middle of the text.

top The vertical reference position is at the top margin of the text.

Horizontal and vertical alignment specifications can be separated by a comma, e.g. **align=center,top.**

Initial value: **left,bottom**.

angle

defines a font property with the possible values:

regular Regular; not italics.

italics Italics or oblique.

The Symbol font is only available as **regular**.

Initial value: **regular**.

autoscale

determines whether the user coordinate system is automatically rescaled after reading list data. The possible values are:

off No automatic scaling.

x Automatic scaling of the *x*-dimension only.

y Automatic scaling of the *y*-dimension only.

x,y or **on** Automatic scaling of both dimensions.

If autoscaling of the *x*-dimension is on, then the values of X_0 and X_1 (plot parameters **X0** and **X1**) are reset after reading list data such that all values in the *x*-column of the list data are in the range between X_0 and X_1. If autoscaling of the *y*-dimension is on, then the values of Y_0 and Y_1 (plot parameters **Y0** and **Y1**) are reset to include all values in the *y*-columns of the list data. In general, the limits are extended slightly with respect to the exact minimum and maximum in order to avoid that data points lie exactly on the margin.

Initial value: **on**.

border

determines whether the border of a closed figure (a rectangle, a circle, an ellipse, a polygon, a closed Bézier curve, or certain types of marks) will be drawn as a line:

off Border lines are not drawn.

on Border lines are drawn.

Initial value: **on**.

color

defines the color, and can have the value **black**, **white**, **red**, **green**, **blue**, **cyan**, **magenta**, or **yellow**. All text and graphics that follows has the given color.

Initial value: **black**.

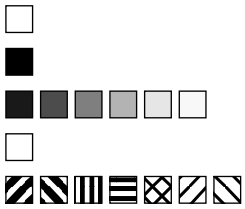
dash defines the dash pattern used to draw lines. Its value is either blank (which is equivalent to **solid**), or a comma separated list of numbers, or one of the following literals:

- **solid** Solid lines.
- **dotted** Dotted lines; equivalent to **1**.
- **dashed** Dashed lines; equivalent to **5,4**.
- **dot-dashed** Dot-dashed lines; equivalent to **5,2,1,2**.

General dash patterns are specified by a comma separated list of numbers that define the lengths (measured in points) of alternating solid and invisible stretches.

Initial value: **solid**.

fill defines the fill pattern used to draw areas. Its value is an integer between 0 and 15 with the following meaning:



- 0 Empty; do not fill areas.
- 1 Full color.
- 2–7 Progressively less saturated shading or color.
- 8 White; covers other graphics.
- 9–15 Different types of hatching.

Initial value: 0.

font defines the font type and can have the following values:

- **Times** Times.
- **Helvetica** Helvetica.
- **Courier** Courier.
- **Symbol** Symbol.

Initial value: **Helvetica**.

linewidth defines the current linewidth in points (1 pt = 0.353 mm).

Initial value: 1.

marksize defines the mark size in points (1 pt = 0.353 mm). If the mark is a circle, the mark size corresponds to the diameter. For other types of marks, similar conventions apply.

Initial value: 6.

mode defines the input mode to line and area drawing commands and can have the following values:

- **normal** Coordinates are specified explicitly on the command line.

list Coordinates are taken from vector expressions, and the corresponding command is applied to all points in the list.

The input mode is automatically set to **list** when a graphics file with list data is read.

Initial value: **normal**.

rotate defines the direction in which text is written and can have the following values:

off Text is written horizontally, from left to right.

on Text is written vertically, from bottom to top.

Initial value: **off**.

textsize defines the font size in points (1 pt = 0.353 mm).

Initial value: 12.

weight defines a font property with the possible values:

regular Regular; not bold.

bold Bold.

The Symbol font is only available as **regular**.

Initial value: **regular**.

x0, y0, x1, y1 define the positions of the two reference points (x_0, y_0) and (x_1, y_1) in the standard coordinate system. The standard coordinate system has its origin in the center of an A4 sheet and uses points (1 pt = 0.353 mm) to measure distances in both dimensions. The *x*-axis points to the right, and the *y*-axis points up.

Initial values: $x_0 = -250$, $y_0 = -375$, $x_1 = 250$, $y_1 = 375$.

X0, Y0, X1, Y1 define the positions of the two reference points (X_0, Y_0) and (X_1, Y_1) in the user coordinate system. All positions and distances are measured in the user coordinate system except for linewidth, text size, mark size, and dash patterns, which are always specified in points. These plot parameters are changed implicitly by the **scale** command or if autoscaling is enabled. The values of these plot parameters are available in INCLAN as intrinsic functions: **plotx0**, **ploty0**, **plotx1** and **ploty1**.

Initial values: $X_0 = -250$, $Y_0 = -375$, $X_1 = 250$, $Y_1 = 375$.

Commands

There are two kinds of commands in the program DYANA: general built-in commands of the command line interpreter, INCLAN, that are not specific to the program DYANA (see chapter INCLAN), and specific DYANA commands. This chapter gives an alphabetical list of the DYANA commands and the standard macros of DYANA. Macros can be found in the "macro" directory. They can be used exactly like ordinary commands and are marked with the  symbol. The names of DYANA-specific commands (but not of INCLAN commands or macros) can be abbreviated as long as there is no ambiguity.

The syntax, parameters, and options of a command are given according to the following scheme:

command name

parameter	*default value*
name=*value*	*default value*
option1 option2	
option3 | **option4**	

command name is the name of the command, which may consist of more than one word. Parameters and options are given in the form of a table in which the left column gives parameters and options, and the right column indicates *default values* for optional parameters, or "—" for required parameters. In the table above, the first row shows a positional *parameter*, the second row shows a **name**d parameter, the third row shows **option**s that may be given simultaneously, and the last row shows mutually exclusive **option**s (see INCLAN command **syntax**). Sometimes optional items are given in square brackets, and "..." indicates that the preceding item may be repeated several times.

When executing a command, the parameters and options must all be giv-

en on one line (or on continuation lines); the tabular form is used only in the manual for clarity.

angle fix

angle selection — *all angles*

All selected angles are fixed, i.e. become non-rotatable and cannot be changed during minimization or dynamics. By default, all peptide angles ω are fixed and set to 180^o (see also "Residue sequence" in chapter "File formats").

angle flip

angle selection — *all angles*

All selected angles of the selected memory structures are analyzed to find the most frequent angle value. The angles are then flipped by 180^o if the new angle is nearer to this most frequently found angle value. This command is used by the **flip** macro.

angle free

angle selection — *all angles*

All selected angles become free angles, i.e. rotatable angles that may be changed during minimization or dynamics (see also "Residue sequence" in chapter "File formats").

angle list

angle selection — *all angles*

Lists all selected angles (see chapter Selections). The names of fixed angles are enclosed in parentheses.

angle rename

name —
angle selection —
on | **off** | **clear**

Defines an external *name* for the selected angles. Not more than one angle may be selected per residue. External angle names are used in place of the corresponding internal angle name from the residue library when reading input files and writing output files. Initially, or after the command **angle rename clear**, external and internal angle names are iden-

tical. With **angle rename off**, renaming may be turned off temporarily, until it is turned on again by **angle rename on,** or by a new external angle name definition.

angle set

value=*r*	—
angle selection	*all angles*

All selected angles are set to the value *r*, given in degrees.

angstat clear

Clears the angle statistics that is used to create redundant angle constraints in the REDAC strategy (Güntert & Wüthrich, 1991; see **angstat make**).

angstat list

Lists the current angle statistics (see **angstat make**).

angstat make

ang_cut=*T*	*taken from variable* **ang_cut**

Adds the current structure to the angle statistics that is used to create redundant angle constraints in the REDAC strategy (Güntert & Wüthrich, 1991; see macro **redac**). First, the local target function value of every single residue is calculated by summing up all contributions to the target function from constraints that involve a given residue. If a given residue and its closest neighbors have a local target function value below *T*, then all dihedral angles of this residue are added to the angle statistics.

anneal

thigh=*T_{high}*	8.0
tend=*T_{end}*	0.0
steps=*N*	4000
highsteps=*N_{high}*	*N*/5
minsteps=*n*	1000
relax	

Performs simulated annealing on the current structure with a total of N MD steps, starting with N_{high} MD steps at temperature T_{high} followed by slow cooling during $N - N_{high}$ MD steps to a final temperature of T_{end}. Finally, n steps of conjugate gradient minimization are added. The temperature is measured in target function units per degree of freedom. Optionally, more minimization can be performed in order to **relax** strong overlaps and constraint violations prior to the start of the MD calcula-

tion. The **relax** option can be useful for larger (above 200 residues) proteins if otherwise the maximal length of the pair list would be exceeded.

asno

distance=d_{max}	5.5
structure=*N*	1
assignfile=*file*	
peakfile=*file*	
sortdistance color	

Determines assignment possibilities for NOESY cross peaks on the basis of chemical shift agreement and short corresponding ^{1}H-^{1}H distances in a bundle of conformers. This command provides the functionality of the former ASNO program (Güntert *et al.*, 1993). An assignment of a NOESY cross peak at position (ω_1, ω_2) to a proton pair (α, β) with chemical shifts ω^α and ω^β is possible if the condition

$$\left(\frac{\omega_1 - \omega^\alpha}{\Delta\omega_1}\right)^2 + \left(\frac{\omega_2 - \omega^\beta}{\Delta\omega_2}\right)^2 \leq 1 \qquad [3]$$

is fulfilled ($\Delta\omega_1$ and $\Delta\omega_2$ are the first and second component of the system variable **tolerance**), and if the distance between the two protons is shorter than d_{max} in at least *N* conformers (Güntert *et al.*, 1993). Three-dimensional spectra are treated analogously.

The option **assignfile** generates a *file* containing all assignments which are allowed by *asno* that can be displayed in the assignment window of the program XEASY (Bartels *et al.*, 1995). Additionally, a new peak list for XEASY containing the old peak list and all assignment possibilities found by **asno** is produced by using the option **peakfile**. An assignment possibility to a proton pair (α, β) leads to a new peak at position (ω^α, ω^β).

Assignment possibilities for individual peaks are by default sorted according to the chemical shift deviations or, if the option **sortdistance** is set, by ^{1}H-^{1}H distance values.

If the option **color** is set, the peaks of the input peak list get a XEASY color code according to the following criteria:

- color 1 — The assignment given by the user is found by **asno** as the best assignment possibility.
- color 2 — The assignment given by the user is found by **asno** but not as the best assignment possibility.
- color 3 — The assignment given by the user is not found by **asno**.
- color 4 — No assignment was given by the user but **asno** found one or several assignment possibilities

color 5 An assignment was found neither by the user nor by **asno**.

The color codes 1–5 are applied only to peaks of the input peak list. All additional **asno** assignment possibilities get the color code 6, except those which are already assigned by the user.

assign

dist=d_{tol}	5.0
transposed=Δr	10000
oneass	

Finds new possible assignments using the actual peak and proton list and stores them into a *test assignment list* (see command **filter** for information on different internal peak lists used by NOAH). For all unassigned peaks a list of possible proton pairs that have chemical shifts within $\pm\Delta_{tol}$ from the peak position is made. The value for Δ_{tol} is taken from the variable **tolerance** if at least one peak in the input peak list was assigned to the corresponding proton, and from the variable **tol_una** if the proton was never assigned to any peak in the input peak list. This allows to differentiate between proton shifts whose position is precisely determined in the spectrum and those which were determined in another spectrum and may be shifted in the actual spectrum.

The selected structures are used to reduce the list of possible assignments in the following way: For each proton pair the corresponding upper distance limit (**obsdis** + d_{pseud}, where d_{pseud} is the pseudo atom correction, if appropriate) is determined and a tolerance distance d_{tol} is added (Mumenthaler & Braun, 1995). If none of the structures can fulfil this enlarged distance limit, the assignment is discarded.

In 3D peak lists, the absence of expected transposed peaks may be used to eliminate wrong assignments if both protons are attached to the same hetero atom type (which must correspond to the spectrum type) and if both hetero atom shifts are known. Pseudo atoms like QD in Phe and Tyr which represent protons attached to different hetero atoms cannot be used for this check, because the position of the transposed peak is not determined.

The check for transposed peaks is only performed if both residues are at least Δr positions apart in the sequence (the default value of the parameter **transposed** means that no check for transposed peaks is done). The position of the transposed peak is calculated and the peak list is screened for peaks that are positioned within ±**tol_transp** ppm of this position. The assignment possibility is discarded if no such peak is found.

Optionally, only peaks which are already assigned in one proton dimension are assigned by NOAH (**oneass**). This can be useful in 3D lists where the assignment of one dimension is known and where NOAH is asked to find the assignment of the other dimension.

Only peaks with less than **maxamb** (see variables) possible assignments are taken into the *test assignment list*.

With **info**=*full*, one output line is written for every unassigned peak:

```
708   2 *  1 -  2 =  0    !      ALL ELIMINATED
734   1 *  1 -  1 =  0           ALL ELIMINATED
735   1 *  1      =  1    *      UNAMBIGUOUS
736   4 *  5 - 10 = 10    *
737                               No possible proton in dimension 1
762   1 *  2 -  1 =  1    *      unambiguous
783   4 *  3 -  7 =  5    *      ==> test al
```

The data are: peak number, number of assignment possibilities in the first and in the second proton dimension, number of assignment possibilities that were discarded because of structures or transposed peaks and the resulting number of assignments. In 3D lists, dimension 2 is always the one coupled to the hetero atom (dimension 3), regardless of what dimension 2 was in the input peak list (see **read peaks**). If a reference peak list has been loaded and the peak in consideration is assigned in this reference list, NOAH will indicate that the reference assignment is either still present ('*') in the remaining assignment possibilities or that it has been discarded ('!') by NOAH. The following comments may be printed at the end of each line:

- **UNAMBIGUOUS** – The assignment is unambiguous based on chemical shifts alone.
- **unambiguous** – The assignment is unambiguous only because some assignment possibilities could be discarded because of incompatibility with the selected structures or the absence of a transposed peak.
- **==> test al** – The peak has less or equal **maxamb** assignment possibilities and was therefore taken over in the *test assignment list*.
- **ELIMINATED** – All assignment possibilities a peak had based on chemical shifts were eliminated because of incompatibility with the selected structures or the absence of a transposed peak.
- **No possible proton in dimension *x*** – No proton chemical shift exists within the given tolerance range from the peak position in dimension *x*. In 3D peak lists the hetero atom dimension (dimension 3) is coupled to its proton (dimension 2) and the message means that the problem occurred in one of both dimensions.

No comment means that there are more than **maxamb** assignment possibilities left and that the peak was therefore not considered at this stage. At the end of this output, the number of peaks belonging to every one of the above categories is given.

atom glomsa

atom selection	*all atoms*
cutoff=*c*	0.4
threshold=*t*	0.4
fraction=*f*	100

Function of the previously separate program GLOMSA ("Global method for stereospecific assignments," Güntert *et al.*, 1991a). The selected structures are searched for possible stereo-specific assignments of the selected atoms. To be taken into account, the difference between two constraints going from a stereo-specific atom pair β1 and β2 to another atom α must be at least *c* Å, the corresponding average distance difference in the structures must be at least *t* Å, and the minimal percentage of structures in which the sign of the distance difference must be consistent must be larger than *f* percent.

atom list

atom selection	*all atoms*

Lists all selected atoms (see chapter Selections). This command is useful to test whether a certain atom selection does select the desired atoms.

atom mass

value=*m*	1.0
cluster	
atom selection	*all atoms*

Sets, if the **cluster** is not set, the mass of all selected atoms to *m*. In this case *all* inertia tensors are calculated from the masses and positions of their constituting atoms. If the **cluster** option is set, the inertia tensors of *all* rigid units are set as if the rigid units were spheres of radius 5 Å with mass $\sqrt{M}$, where *M* denotes the sum of the atomic masses within the rigid unit. Inertia tensors are initialized in this way when the program starts. Atomic masses are initialized to unity. The mass does only influence the MD calculations.

atom rename

name	—
atom selection	—
on | **off** | **clear**	

Defines an external *name* for the selected atoms. Not more than one atom may be selected per residue. External atom names are used in place of the corresponding internal atom name from the residue library when reading input files and writing output files. Initially, or after the command **atom rename clear**, external and internal atom names are identical. With **atom rename off**, renaming may be turned off temporarily, until it is turned on again by **atom rename on** or by a new external atom name definition.

```
atom rename HB1 HB2 - @ALA
atom rename HB2 HB3 - @ALA
```

These two statements allow reading input files or writing output files in which diastereotopic β-protons are called HB1/HB2 instead of HB2/HB3 (as they are called in the standard residue library).

atom shift

atom selection
missing | **adapt** | **d2o** | **check**

all atoms

One of the following actions is performed on the selected chemical shifts:

missing Lists all expected chemical shifts that are not present in the chemical shifts list. If no atom selection is given only proton shifts are reported.

adapt Uses the positions of the assigned peaks to adapt the proton shifts. The new shift is the average chemical shift position of all peaks assigned to the same proton.

d2o Deletes all NH shifts from the chemical shift list. This command is useful for preparing an chemical shift list for the automatic NOESY-spectrum assignment of a spectrum recorded in D_2O.

check Checks the current chemical shifts in two ways: First, the shifts are compared to the corresponding minimal and maximal values in the statistics of expected chemical shifts (which is stored in the standard library file, “dyana.lib”). Chemical shifts that lie outside of this range will be printed. They are not necessarily wrong, but should be checked with care. In a second test, this command uses the positions of the assigned peaks to check the chemical shifts for inconsistencies. For every proton shift the median and the spread of the peak positions assigned to this proton are calculated and printed if the spread is larger than the corresponding tolerance value (see system vari-

able **tolerance**). The number of peaks assigned to the proton is also printed.

atom stereo

atom selection	*all atoms*
list | **delete**	

Defines selected atoms as stereoassigned. It is sufficient to select one atom of a diastereotopic pair to define both diastereotopic partners as stereoassigned. Optionally, all stereo partners may be **list**ed, or the stereospecific assignments of selected atoms may be **delete**d.

atom swap

atom selection	*all atoms*
optimal	

Swaps diastereotopic partners in peaks, distance constraints, coupling constants and chemical shifts (but not in the structure itself). It is sufficient to select one atom of a diastereotopic pair to swap both diastereotopic partners.

Optionally, diastereotopic pairs which are not already stereoassigned may be swapped **optimal**ly in order to achieve the lowest possible target function value.

atom vdw

atom selection	—
scale=*s*	1.0
increment=Δr	0.0
hincrement=Δr_h	0.0

Selects atoms which are included into the van der Waals check. The optional parameters **scale** all selected atom radii by a factor *s*, **increment** them by Δr or increment only the radii of heavy atoms with directly bound hydrogen atoms by Δr_h (**hincrement**). The latter parameter is used in the DYANA standard minimization procedure to compensate for the exclusion of hydrogen atoms in the lower minimization levels (see macro **vtfmin**).

bmrblist

file	—

Writes a chemical shift list in the format of the BioMagResBank (for details, see http://www.bmrb.wisc.edu).

calc_all

structures=*n*	*all selected structures*
command=*command*	anneal
parameters	

Calculates a group of structures using the given *command* (with optional *parameters*) for each individual conformer. If the number of structures *n* is specified, the calculation will be performed starting from *n* random start conformers; otherwise the calculation is performed for all selected structures. Structure calculations are performed in parallel, if possible.

caliba

dmin=d_{min}	2.4
dmax=d_{max}	5.5
vmin=V_{min}	0.0
bb=*A*	*calculated automatically*
sc=*B*	A/d_{min}^2
methyl=*C*	*B/3*
weight=*w*	1.0
avedis=$\bar{d}$	3.4
peaklist=*filename*	*all peaks*
plot=*file*	—

Calibrates a peak list, i.e. derives upper limit distance constraints from all assigned peaks and adds them to the list of current distance constraints. The values d_{min} and d_{max} give the minimal and the maximal value in Å for a distance constraints before possible pseudo atom corrections are added. Optionally, only peaks with volume larger than V_{min} or from a peak list with given *filename* (without extension) may be considered. Peaks are classified into three calibration classes:

class	peaks/constraints	function
backbone	all HN/H^{α} — HN/H^{α}, and HN/H^{α} — H^{β} between residues (i, j) with $\lvert i - j \rvert < 5$	$V = A/d^6$

class	peaks/constraints	function
sidechain	not “backbone” and not “methyl”	$V = B/d^4$
methyl	all involving methyl groups	$V = C/d^4$

The parameters *A*, *B* and *C* are either given by the user or calculated automatically as follows:

The function **calsca** is used to calculate *A* by assuming an average distance of $\bar{d}$ Å for all constraints from the class “backbone”. By default, the scalar *B* is set to $B = A/d_{min}^2$ in order to intersect the backbone calibration curve at d_{min}, and *C* is set to *B*/3 (see also Mumenthaler et al., 1997).

Optionally, the resulting distance constraints may be given the relative weight *w*. Also optionally, a logarithmic **plot** of volumes *versus* corresponding minimal distances in the selected structures can be created.

calibrate

f(*d*)	—
d_{min} [d_2 ...] d_{max}	2.4 5.5
weight=*w*	1.0
plot=*file* **log minimal**	*none*

Derives upper distance limits from all selected peaks using a monotonically decreasing calibration function *f*(*d*), where *d* represents the distance and *f*(*d*) the corresponding volume (e.g. “1/d**6”). The minimal and maximal upper limit (before possible pseudo atom corrections are applied) are given by d_{min} and d_{max}. If additional values d_2... are given, then these discrete values are used for upper limits; otherwise, a continuous calibration curve is used. Optionally, the resulting distance constraints may be given the relative weight *w*. Also optionally, a linear or **log**arithmic **plot** of volumes *versus* corresponding average or **minimal** distances in the selected structures can be created.

Before calibration, the volumes of peaks assigned to pseudo atoms are divided by the number of protons they represent. For instance, the volume of a cross peak between a Leu QQD pseudo atom and a Tyr QB pseudo atom is divided by a 6 x 2 = 12.

cashifts

offset=Δω	0.0

Generates constraints for the backbone dihedral angles ϕ and ψ in proteins by comparing the C^α chemical shifts with the corresponding ran-

dom coil values of Spera & Bax (1991). Angle constraints are derived according to the rules of Luginbühl *et al*. (1995). The C^{α} random coil shifts are relative to internal TSP. Optionally, an **offset** $\Delta\omega$ is added to the chemical shifts in the proton list. A warning is printed for C^{α} chemical shifts that deviate by more than 15 ppm from their random coil value.

cluster

file=*name*	*cluster.ps*
range=*residue range*	*all residues*

Calculates the backbone RMSD of the selected structures and performs a cluster analysis on the resulting RMSD matrix. The resulting graphics is written into the graphics output file with given file *name* (a GRAF file if the extension is ".grf", a MIF file if the extension is ".mif", or a Postscript file otherwise). A specific *residue range* may be specified for the RMSD calculation.

The *y*-axis of the plot gives the structure numbers and the *x*-axis shows the RMSD with which a structure or a structure cluster "joins" another cluster. This RMSD is the minimal RMSD that any of the structures in the first cluster have to any of the structures in the second cluster. Currently, up to 20 structures can be analyzed.

create

list=*string*	1,2,3
w1=w_1	5.0
w2=w_2	10.0

Creates upper limit distance constraints of **obsdis** + d_{pseud} Å (d_{pseud} is the pseudo atom correction, if appropriate) on the basis of the three different assignment lists of NOAH (see command **filter** for information on the NOAH peak lists). The parameter **list** is a string containing the numbers of the lists that should be considered. (list=1: Unambiguous assignment list (UAL), list=2: Ambiguous assignment list (AAL), list=3: Test assignment list (TAL)). Constraints in the UAL are weighted with the factor w_1 by default or with the factor w_2 if they are unambiguous based on the chemical shift alone (see command **assign**). Constraints from the AAL are weighted with 1.0 and constraints from the TAL with 1.0/ N_{ass} (where N_{ass} is the number of possible assignments a peak has in the **assign** command).

The AAL only exists after the **filter** command and the TAL only exists after the **assign** command.

dcostat

file=*name* — *dcostat.ps*

Produces a graphics output file with the given *name* (a GRAF file if the extension is ".grf", a MIF file if the extension is ".mif", or a Postscript file otherwise) containing two plots which show the distribution of distance constraints. The first plot shows the number of distance constraints plotted against the residue index difference of the corresponding atoms. The second plot shows for every residue the number of intra-residual (white), short-range (vertically hatched), medium-range (horizontally hatched) and long-range (black) constraints.

differences

file
intersection
notdiff

Lists all differences in the assignments between the current peak list and an external peak list *file*. Corresponding peaks must have the same peak numbers in both lists:

```
-------------------------------------
peak  791 :   20  HB2      9  HN     NOAH - RelDis:   0.0
              20  HB2     28  HN     File - MinVio:   0.0
-------------------------------------
peak  979 :    7  HA      10  HN     NOAH - RelDis:   3.2
               8  HA      10  HN     File - MinVio:   0.0
-------------------------------------
peak  985 :   13  HA      12  HN     NOAH - RelDis:   0.0
              13  HA      15  HN     File - MinVio:   0.5
-------------------------------------
Number of equal assignments    : 608
Number of different assignments: 3
```

The first line of each differently assigned peak contains the NOAH assignment and its reliability distance (RelDis), provided the latter was previously calculated with the command **reliability**. The second line contains the assignment given in the *file* together with the minimal violation (MinVio) this assignment would have in the selected structures. This violation is calculated on the basis of a distance limit of 5.0 Å plus pseudo atom correction, if appropriate, and does therefore not consider the peak volume.

For the interpretation of the RelDis/MinVio combinations three sub-categories can be made (Mumenthaler et al., 1997):

- RelDis = 0.0 Å / MinVio = 0.0 Å: Both assignments are satisfied in the structures. Assuming that the conformers are correct solutions, such peaks must be superpositions of two NOE signals, so that both

assignments are correct.

- RelDis > 0.0 Å / MinVio = 0.0 Å: Here, the assignment from the peak list *file* lies outside of the given **tolerance** range from the peak position and was therefore not considered during the calculation of the reliability distance. Unless the current proton shifts are not well adapted to the peak list or the tolerance range was too small, the current assignment seems more appropriate for the peak under consideration.
- RelDis ≥ 0.0 Å / MinVio > 0.0 Å: These are the most relevant differences since the assignment from the peak list *file* is violated by the current structures. Thus, the different assignment will also have an impact on the structures.

If the assignment in the peak *file* has a "-" sign in the integration method field, the comment "Peak not used in structure calculation" is written.

Optionally, an **intersection** is made between both peak lists and the assignments are kept only if the peak is assigned to the same proton pair in both lists. If the peak is assigned only in one of the lists it is therefore also unassigned.

The option **notdiff** is less stringent since only peaks which are differently assigned in both lists are unassigned.

dinucleotide

range=*residue range*	*all amino acid residues*
tfcut=f_{max}	0.0
continue	

Performs grid searches for all dinucleotide fragments in the given **range**. If the cutoff value for the local, fragment-based target function, f_{max}, is positive, then all conformations with a local target function value below f_{max} will be considered as allowed. Otherwise, i.e. if f_{max} = 0.0, a conformation will be allowed if no single restraint violation exceeds the corresponding cutoff value defined by the variables **soft_upl**, **soft_lol**, etc. Unless the **continue** option is set, the allowed ranges of dihedral angles will be initialized to allow all possible angle values before the grid searches are started.

The results include dihedral angle restraints and, if possible, stereospecific assignments for the diastereotopic groups in the fragment.

distance check

Checks how well long range distance constraints are supported by other constraints. A low score indicates "lonely" and therefore "dangerous" constraints with a high impact on the calculated 3D structure.

For the distance constraint i going from residue number r_i^1 to r_i^2 (with $r_i^1 < r_i^2$), the score $s(i)$ is defined as a sum over all other distance constraints j:

$$s(i) = \sum_{j=1}^{N} \frac{1}{(1 + |r_i^1 - r_j^1|) \cdot (1 + |r_i^2 - r_j^2|)} \qquad [4]$$

A high score means that the distance constraint is supported by other constraints while a score of 0 means that the constraint is isolated.

distance clear Deletes all distance constraints.

distance compare Compares distance constraints. For every selected distance constraint other selected constraints to the same atom pair are searched. If the information level is **full**, a line is written for each comparison containing the two distances and the atom names. At the end, a histogram is printed with the number of constraints that were found for each difference interval. This command is useful for the comparison of two differently calibrated distance constraints files (the second one must be loaded with "**read upl** *file* **append**").

distance delete Deletes all selected distance constraints.

distance keep Keeps only the selected distance constraints.

distance list Lists all selected distance constraints.

distance correct Adds pseudo atom corrections to all selected upper limit distance constraints. Pseudo atom corrections are only added to constraints that involve pseudo atoms. The correction is given by the distance between the pseudo atom and the hydrogen atoms that it represents.

distance make

d	—
first atom selection,	—
second atom selection	—
weight=*w*	1.0
lol	

Creates a new distance constraint (an upper limit unless the option **lol** is set) of *d* Å between all atoms matching the *first atom selection* and those matching the *second atom selection*. A **weight** might be specified.

distance modify

Modifies distance constraints. Redundant and meaningless distance constraints are removed. Distance limits with diastereotopic groups are adjusted and/or pseudo atoms are inserted if no stereospecific assignment is available (Güntert *et al.*, 1991a).

If the information level is **full**, a detailed listing of all modifications that have been done to upper distance limit constraints such that they allow for both possible stereospecific assignments simultaneously (unless a stereospecific assignment is available for a given diastereotopic pair) is given. For example:

```
Modifications for floating stereospecific assignments:

      Atom(s) A          Atom(s) B          Input constraint(s)      modified to
                                            A1-B1 A1-B2 A2-B1 A2-B2 Ai-Bj QA-QB
Upper HA     ASP-   1 - HD2/3  PRO    2          3.39                3.55
Upper HB2/3  ASP-   1 - HD2/3  PRO    2    5.50  5.50  3.89  5.41  5.50  4.97
Upper HB2    PRO    2 - HG2/3  MET    3    5.50                            6.38
Upper HD2/3  PRO    2 - QE     TYR   19    7.63                            8.51
Upper HN     MET    3 - HB2/3  MET    3    3.95  3.33                3.83  3.45
Upper HB2/3  MET    3 - QE     MET    3    6.53        6.31        6.53
Upper HB2/3  MET    3 - HN     THR    4    5.38        5.50        5.50
Upper HB2/3  MET    3 - HA     THR    4    4.69                            5.54
Upper HG2/3  MET    3 - HN     THR    4    4.14                            5.01
Upper HG2/3  MET    3 - HB     THR    4               5.50                 6.38
Upper HG2/3  MET    3 - QB     ALA    8               6.53                 7.40
Upper HG2/3  MET    3 - QD     TYR   19    7.64                            8.52
Upper HG2/3  MET    3 - QE     TYR   19    7.63                            8.51
Upper QE     MET    3 - HG2/3  MET   16    5.47                            6.35
Upper QG2    THR    4 - HE21/2 GLN    7    6.53                            7.39
Upper HN     GLU-   6 - HG2/3  GLU-   6    5.04  5.50                5.50  5.14
Upper HA     GLU-   6 - HG2/3  GLU-   6    4.23
```

Each line in the listing of distance constraint modifications treats a pair of distance constraints in case one diastereotopic pair (without stereospecific assignment) is involved, or a quartet of distance constraints in case two diastereotopic pairs are involved. Not all two or four distance constraints need to be present in the input, of course. In case a distance constraint is available from one atom to the first diastereotopic substituent of a prochiral centre, it is listed in the column below the header A1-B1, a constraint to the second diastereotopic substituent is listed below the header A1-B2, a constraint between the second diastereotopic substituent of one and the first diastereotopic substituent of another prochiral centre appears under the header A2-B1 etc. The four columns entitled A1-B1, A1-B2, A2-B1 and A2-B2 therefore list the input distance constraints with (presumably) arbitrary stereospecific assignment. These will then be replaced by the distance constraints listed in the two columns Ai-Bj and QA-QB: the distance limits below the heading Ai-Bj will apply for the individual distances involving the diastereotopic substituents, in case one diastereotopic pair is involved there will be two such distance constraints, in case of two diastereotopic pair there will be four such distance constraints; the final column indicates the limits that are imposed on the distances involving pseudo atoms located centrally with respect to the diastereotopic substituents. No distance limit is indi-

cated if none will be imposed because the modified distance limit(s) would be meaningless.

In addition to the modifications done to account for the absence of stereospecific assignments, the command **distance modify** detects and removes meaningless constraints in the input. A table is given if the information level is **full**. For example:

```
Meaningless distance constraints:
                                                 limit  diagnosis
Upper HA    ASP-   1 - HB3   ASP-   1   3.21  duplicate constraint
Upper HA    ASP-   1 - HB3   ASP-   1   3.21  no restriction
Upper HA    ASP-   1 - HD2   PRO    2   4.20  duplicate constraint
Upper HA    ASP-   1 - HD2   PRO    2   4.20  no restriction
Upper HA    ASP-   1 - HD3   PRO    2   3.21  duplicate constraint
Upper HB2   ASP-   1 - HB3   ASP-   1   2.40  duplicate constraint
Upper HB2   ASP-   1 - HB3   ASP-   1   2.40  fixed distance
Upper HB2   ASP-   1 - HD2   PRO    2   5.50  duplicate constraint
...
Number of modified constraints: 597
```

Distance constraints can be meaningless for one of the following reasons:

fixed distance The constraint concerns an interatomic distance that cannot be varied by changing the rotatable torsion angles. Examples of this sort are constraints between geminal hydrogen atoms, or constraints between atoms of the same aromatic ring.

no restriction The constraint is such that there exists no conformation that would violate it. The program can detect this only if the constrained distance depends on one or two dihedral angles. Many meaningless intraresidual peaks can thus be eliminated.

duplicate constraint The same constraint occurs more than once in the input, for example because transposed peaks were present in the peak list.

The number of upper distance limits after doing modifications is given at the end of the table; depending on the number of stereospecific assignments, modification may increase or decrease the number of constraints.

distance scale

factor=*f* —

Scales the distance bounds of the selected distance restraints by the factor *f*.

distance select

distance constraint selection —

Selects all distance constraints that match the given *distance constraint selection* (see chapter Selections).

distance set

tolerance=Δ	0.0

Set the distance bounds of the selected distance restraints to the average distance in the selected structures plus Δ.

distance stat

Lists the total number of selected intra-residual, sequential, medium-range and long-range constraints. If the information level is **full**, these numbers are also given for each individual residue.

distance unique

Keeps only the most restrictive distance constraint if several constraints exist for the same atom pair. This command corresponds to the first part of the **distance modify** command.

distance weight

w	—

Weights all selected distance constraints with *w.*

filter

file	—
tolerance=*d*	0.5
L0=*l0*	0
L1=*l1*	50
L2=*l2*	80

Filters an assignment *file* (see command **write ass**) with respect to the selected structures (Mumenthaler & Braun, 1995). The percentage P_{vio} of structures in which every assignment is violated is counted. Violations smaller than the **tolerance** distance *d* (in Å) are not considered. **L0**, **L1** and **L2** correspond to the (percentage) thresholds mentioned in Mumenthaler & Braun (1995).

This command makes use of three internal peak lists:

- **Unambiguous assignment list (UAL)**: All peak assignments which were unambiguous at some stage of the NOAH calculation and which do not violate the structures. *This list is in fact simply the assigned peak list!*
- **Ambiguous assignment list (AAL)**: All peaks with more than one assignment which were used in structure calculations and did not

lead to large structural violations. Peaks from this list are added to the normal peak list (one entry per possible assignment) with negative peak numbers. They are visible with the command **peak list**, but are not written to disk by **write peaks**.

- **Test assignment list (TAL)**: All peak assignment which might be possible and are detected by the command **assign**. They were not used in any structure calculation yet.

The peak assignments from all three lists were stored into an assignment *file* in a previous NOAH cycle and are redistributed (if possible) into the first two lists by this command (the test assignment list is exclusively fed by the command **assign**). The AAL is cleared at the beginning of this command, but peaks already assigned in the current peak list (UAL) are not altered by this command, even if the peak is differently assigned in the assignment *file*.

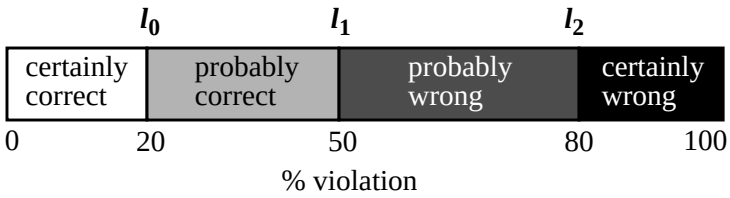
For every peak entry of the assignment *file* (which may contain several assignments per peak) the following procedure is made:

First, all assignments with $P_{vio}>l_2$ are discarded. Assignments from the UAL of the assignment *file* which pass this test are transferred to the current peak list (UAL). If the peak in the assignment file was from the AAL or the TAL, two cases are distinguished:

1) The peak has only one possible assignment and is either transferred to the unambiguous assignment list (if $P_{vio}\leq l_1$) or unassigned.

2) The peak has several assignments left. If one of them is much better than the rest, i.e. it has $P_{vio}\leq l_0$ and the rest has $P_{vio}>l_1$, the peak is unambiguously assigned. If not, all remaining assignment possibilities are stored into the ambiguous assignment list.

Usually, the relation between the l thresholds should be $l_0\leq l_1\leq l_2$ in which case the following interpretation may help to understand their significance:



flip

M

Flips planes of aromatic rings of PHE and TYR residues and planar groups of ASP- and GLU- by 180^o such that there is a best fit between all selected structures. This command does not affect the three-dimensional structure. The change is limited to the nomenclature which results in a lower heavy atom RMSD.

forall

commands
save parallel

—

This macro performs a loop over all selected structures, copies them into the structure memory #0, executes the user *commands*, and copies the structure back to the structure memory. Optionally, the structures are **save**d as angle files with name d*nnnmmm*.ang (*nnn* denotes the current process number, and *mmm* the structure number) before they are copied back. The calculation may be executed in **parallel** (if the INCLAN variable **nproc** has a value larger than 1).

```
forall parallel "vtfmin steps=100,800; angstat make"
```

Minimize all selected structures using the macro **vtfmin** with the given number of steps and include the resulting structures into the angle statistics.

graf

file=*file*
replace

—

Convert a graphics file into a postscript or MIF plot file. If *file* has the extension ".grf", nothing happens. If *file* has the extension ".mif", a MIF file with this name is produced from the corresponding graphics file with extension ".grf". In all other cases a Postscript file is produced from the corresponding graphics file with extension ".grf". If the option **replace** is set, then the graphics file is removed after the MIF or Postscript file has been produced.

grid aco

add | **replace**
multiple

Creates dihedral angle restraints that include the allowed angle values stored in the standard grid memory, **A** (see command **grid memory**). Optionally, the new restraints may be **add**ed to those already present, or the "old" restraints may be discarded and **replace**d by those created on the basis of the grid memory. If neither the **add** nor the **replace** option is set, the intersection between the "old" and "new" angle restraints will be formed. It is possible to generate **multiple** restraints for one dihedral angle if there are several allowed regions for this dihedral angle present

in grid memory. By default, only one restraint that includes all allowed ranges is created per dihedral angle.

grid correlate

function=*f*(*p*)	—
angle selection	*all angles in fragment*

Defines relationships between dihedral angles in a grid search. To represent a group of dihedral angles as a single degree of freedom in a grid search, this command has to be called once for each correlated dihedral angle. If exactly one angle within the fragment is selected, then it will be related to the parameter *p* (lower case **p**), which will be the single degree of freedom during the grid search and vary from 0 to 2π, by the function *f*(*p*). If exactly two angles within the fragment are selected, one of which has occurred in a previous **angle correlate** command, then the "new" angle will be correlated by the function *f*(*p*) to the same parameter *p* as the "old" angle.

```
grid fragment DELTA NU1 NU2 4
```

Define a molecular fragment consisting of the sugar ring of nucleotide 4 (in DNA or RNA).

```
numax=40.0/rad
```

amplitude; **rad** = 180/π

```
grid correlate numax*cos(p+2*pi/5)+2*pi/3 DELTA
grid correlate numax*cos(p-2*pi/5) DELTA NU1
grid correlate numax*cos(p) DELTA NU2
```

Correlate the dihedral angles in the sugar ring to the pseudorotation angle.

grid fragment

angle selection	*all angles*
none	

Defines a fragment to be analyzed by a subsequent grid search. The *angle selection* must select a connected subset of the dihedral angles. Alternatively, the option **none** can be given to undefine the current fragment.

```
grid fragment PSI 7 + PHI PSI CHI1 8 + PHI 9
```

Defines a molecular fragment consisting of ψ of residue 7, ϕ, ψ and χ^1 of residue 8, and ϕ of residue 9.

grid memory

grid memory expression	*none*
angle selection	*all angles in fragment*

This command handles the storage of allowed dihedral angle values that have been determined by grid searches. These are stored in *grid memories* that contain for each dihedral angle in the molecule (not only in the current fragment) a fine grid of 2˚ spacing to store the allowed values. The standard grid memory, **A**, is used by the grid search commands; other grid memories with user-defined names are initialized when they are first used in *grid memory expressions*:

a=**.true.**	initialize grid memory *a*; all angle values allowed
a=**.false.**	initialize grid memory *a*; all angle values forbidden
a=**.not.***b*	not *b*
a=*b*.**and.***c*	intersection of *b* and *c*
a=*b*.**or.***c*	union of *b* and *c*
a	list contents of grid memory *a*
a=	remove grid memory *a*

Grid memory expressions must not contain blanks. If an *angle selection* is specified, then the operation will be applied to all selected angles. By default, the operation is performed for all angles in the current fragment.

The command can be given without any parameters; in this case the names of all occupied grid memories are printed.

grid search

tfcut=f_{max}	0.0
test	

Performs a grid search for the current fragment (as defined with the **grid fragment** command). The grid search will be done over all angles in the fragment and with the number of steps given by the variable **nstep**. If the cutoff value for the local, fragment-based target function, f_{max}, is positive, then all conformations with a local target function value below f_{max} will be considered as allowed. Otherwise, a conformation will be allowed if no single restraint violation exceeds the corresponding "soft" cutoff values defined by the variables **soft_upl**, **soft_lol**, etc. To avoid excessive computation times for fragments with many angles and/or few restraints, the calculation is not started if the expected number of grid points to be checked (after the evaluation of restraints that involve a single torsion angle) exceeds the value of the variable **gridpoints**. Similarly, a grid search is aborted if the estimated total computation time exceeds **gridtime** seconds.

If the number of grid points to be checked (after evaluation of the restraints that depend on a single dihedral angle) is larger than N_{max}, or if the estimated computation time for the complete grid search exceeds t_{max} seconds, the calculation will be stopped. If the **test** option is set, the grid search will not be started but the expected number of grid points to be checked (after evaluation of the restraints that depend on a single dihedral angle) will be printed.

The grid search is restricted to angle values that are allowed according to the standard grid memory, **A**. The resulting allowed angle values from the grid search will again be stored in the standard grid memory.

grid swap

atom selection — *all atoms in fragment*

Swaps the selected diastereotopic partners in distance restraints and scalar coupling constants in the current fragment. It is sufficient to select one atom of a diastereotopic pair to swap both diastereotopic partners.

gridplot

file — gridplot.ps

Produces a plot in FrameMaker MIF (if the *file* extension is ".mif") or Postscript format of the allowed dihedral angle values in the standard grid memory.

habas

range=*residue range* — *all amino acid residues*
angles=*side-chain angles* — CHI1
tfcut=f_{max} — 0.0
continue

Performs for all amino acid residues in the given **range** grid searches comprising the backbone dihedral angles ϕ, ψ and the given *side-chain angles*. To specify more than one *side-chain angle*, the names must be given, separated by blanks and enclosed in double quotes. If the cutoff value for the local, fragment-based target function, f_{max}, is positive, then all conformations with a local target function value below f_{max} will be considered as allowed. Otherwise, a conformation will be allowed if no single restraint violation exceeds the corresponding cutoff value defined by the variables **soft_upl**, **soft_lol**, etc. Unless the **continue** option is set, the allowed ranges of dihedral angles will be initialized to allow all possible angle values before the grid searches are started.

This macro provides the functions of the former HABAS program (Güntert *et al.*, 1989). The results include dihedral angle restraints and, if possible, stereospecific assignments for the diastereotopic groups in the fragment.

```
habas angles="CHI1 CHI2*" tfcut=0.05
```

Perform grid searches for all amino acid residues including the dihedral angles ϕ, ψ, χ^1 and χ^2. Allow conformations with local target function values up to 0.05.

hbond

atom1=*atom name*	—
residue1=*residue number*	—
atom2=*atom name*	—
residue2=*residue number*	—

Creates the standard upper and lower limit distance constraints (Williamson *et al.*, 1985) to enforce a hydrogen bond between two atoms, one of which must be a hydrogen atom. The distance between the hydrogen and the acceptor is restrained to the range 1.8–2.0 Å, and the distance between the atom covalently bound to the hydrogen and the acceptor is restrained to the range 2.7–3.0 Å.

init

Contains commands that are executed automatically at the start-up time of DYANA, e.g. the setting of important variables and the definition of some aliases. After the general **init** macro, a user-defined **init** macro in the current directory is executed, if available.

keep

dist	0.0

Keeps only assignments with a *reliability distance* > *dist*. The reliability distances must have been calculated previously with the command **reliability**.

kringle

ile=*file*	kringle.ps
delta=Δ	30°
errorbars	

Produces a graphics output file with the given *name* (a GRAF file if the extension is ".grf", a MIF file if the extension is ".mif", or a Postscript file otherwise) containing a plot of ${}^3J_{\alpha\beta 2}$ versus ${}^3J_{\alpha\beta 3}$ coupling constants (Nagayama & Wüthrich, 1981). The theoretical curve based on the Karplus equation given in the library is also drawn, both for a rigid structure (solid line) and for the situation when the χ^1 angle is uniformly distributed in the interval $\pm\Delta$ around a given value (dotted line). Optionally, **errorbars** can be shown for the coupling constant values.

longrangeplot

file=*file*	longrangeplot.ps

Plots long-range distance restraints (five or more residues apart) versus (two copies of) the sequence. Lines going from upper left to lower right represent restraints between side-chain atoms, those going from lower left to upper right represent restraints that involve backbone atoms.

md

steps=*N*	100
dt=Δt	0.05
level=*L*	*taken from variable* **level**
temperature=*T*	0.1
accuracy=ε	0.0
tau=τ	0.0
nprint=*n*	0
angdev=$\Delta\phi$	10.0^o
vdwupdate=N_{vdw}	100
tinit=t_0	0.0
estart=T_0	0.01
exact continue	

Performs *N* steps of molecular dynamics in torsion angle space with step size Δt including constraints up to minimization level *L*.

With $\tau = 0$ a molecular dynamics calculation at constant energy is performed. Otherwise, the system is weakly coupled to a heat bath of temperature *T* using time constant τ (Berendsen *et al.*, 1984). The temperature, *T*, can be a function, *T*(*s*), of the parameter *s* that varies linearly from 0 to 1 during the TAD run, i.e. in step *n* out of a total of *N* steps it has the value $s(n) = (n - 1)/(N - 1)$.

If the reference value for the accuracy of energy conservation, ε, has a positive value, the length of the integration time-step, Δt, will be adapted

during the run in the same way as the temperature such that the relative change of the total energy in successive integration steps is close to ε. In this case, the parameter **dt** specifies the only initial value of Δt.

The van der Waals interaction list is updated every N_{vdw} steps or each time a torsion angle has changed its value by more than $\Delta\phi$ degrees since the last update of the an der Waals interaction list.

The "leap-frog" algorithm is used to perform the torsion angle dynamics steps. Usually, torsional accelerations are computed on the basis torsional velocity values that are linearly extrapolated from those half a time-step earlier. Optionally, it is possible to use more **exact** values which are calculated iteratively (Mathiowetz *et al.*, 1994).

The molecular dynamics simulation starts at time t_0 with random torsional velocities, chosen as Gaussian random variables such that the initial temperature (kinetic energy per degree of freedom) is T_0, unless the **continue** option is given. When a calculation is **continue**d, the velocities from the end of the previous **md** command are used and all parameters that are not given explicitly are kept at the values of the previous **md** command. The parameters **tinit** and **estart** cannot be used together with the option **continue**.

One line of output is written every *n* time-steps, giving the current step, current time, potential energy (i.e. target function value), kinetic energy, total energy, the root-mean-square torsion angle change per time-step (in degrees; averaged over all time-steps since the last output), the maximal torsion angle change per time-step (in degrees; since the last output), the number of updates of the van der Waals interaction list (since the last output), and the number of target function evaluations (since the last output). For example:

step	time	Epot	Ekin	Etot	rmsdev	maxdev	#up	#f
0	0.000	17817.672	5776.000	23593.672			1	1
200	13.778	4367.090	7321.274	11688.363	2.842	18.576	4	204
400	28.471	2896.928	6002.219	8899.147	2.763	16.301	4	206
600	42.374	2464.380	6988.264	9452.645	2.330	13.941	4	200
800	60.234	2496.055	6167.296	8663.351	2.815	15.211	4	200
1000	76.882	1654.211	5322.900	6977.111	2.779	15.591	4	200

All energies are measured in target function units. Temperatures are measured in target function units per degree of freedom (i.e. per rotatable torsion angle).

A warning is printed if in a single time-step the value of a dihedral angle changed by more than 35°, and an error occurs if the change exceeds 90°.

minimize

steps=*N*	100
level=*L*	*taken from variable* **level**
flat=*n*	100
angdev=$\Delta\phi$	10.0^{o}
vdwupdate=N_{vdw}	100

Performs *N* conjugate gradient minimization steps including constraints up to minimization level *L*.

The **flat** parameter is used for the "flat" stop criterion of the conjugate gradient minimizer: It is stopped if within *n* minimization steps the target function cannot be reduced by at least 1%.

The van der Waals interaction list is updated every N_{vdw} steps or each time a torsion angle has changed its value by more than $\Delta\phi$ degrees since the last update of the an der Waals interaction list.

If the information level is **normal** or higher, one line of information will be printed out as in the following example from the macro **vtfmin**:

```
Minimization (standard strategy):
 lev    upper    lower        vdw   angle target funct. |grad| #up   #f stop
        # act    # act     #  act  # act  begin    end    end
   0  115  13    0   0   313   26  84   9 282.65   0.13 1.8E-2   0  150 maxit
   1  271  47    0   0   925   73  84   5 163.09   3.79   0.16  34  150 maxit
   2  299  51    0   0  1067   81  84   5  24.71   3.79   0.11  18  150 maxit
   3  381  57    0   0  1240   92  84   8 694.27   4.22   0.27  20  116 flat
   4  431  74    0   0  1335   90  84  10  21.83   4.47   0.16   9  130 flat
```

The first column gives the minimization level. Then, there are four times two columns containing each time the total number of constraints and the number of "active" constraints for the upper limit constraints, the lower limit constraints, the intrinsic van der Waals lower limit constraints and the angle constraints. "Active" constraints are those that yield non-vanishing (but often small) contributions to the target function. Following this data, the value of the target function at the beginning and at the end of the minimization step is given, accompanied by the norm of the gradient of the target function at the end of the minimization step, the number of updates of the van der Waals contact list, the number of target function evaluations, and a stop criterion code. The following stop criteria codes may occur:

gradtl The squared norm of the gradient of the target function is smaller than the value of the parameter GSQTOL in the subroutine CGMIN.

maxit The maximal number of target function evaluations has been exceeded.

linmin The maximal number of target function evaluations during the line minimization (see the parameter MAXLIN in

the subroutine CGMIN) has been exceeded without decreasing the target function value.

nostep The step size during line minimization became too small.

uphill The direction of a conjugate gradient minimization step was uphill.

const Several conjugate gradient steps did not succeed in decreasing the target function (see the parameter MAXCON in the subroutine CGMIN).

flat The target function was minimized by less than 1% during the last *n* iterations.

stuck Several attempts to restart the conjugate minimizer after an update of the list of steric constraints failed.

If 64-bit floating point precision is used, normal stop criteria are gradtl, maxit, and flat. All others should occur only rarely. With 32-bit floating point precision other stop criteria may occur due to (non-serious) numerical problems.

noah

num_cyc=*n*	24
peak_nam=*file1*[,*file2*,..]	—
plformat=*string1*[,,*string2*,..]	*determined by peak list file*
rmsd_range=*residue range*	*all residues*
protein=*name*	—
proton_nam=*file1*[,*file2*,..]	—
minimizer=*macro*	**noahmin**
peak_ref=*file1*[,*file2*,..]	
options=*string*	
addupl=*file*	
addlol=*file*	
exit=*cycle*	**num_cyc**
entry=*cycle*	1
calibrate	

Automatically assigns the peak lists given in the array **peak_nam** with corresponding proton lists in the array **proton_nam** in *n* NOAH cycles. If the peak lists do not contain a line with "#DYANAFORMAT" the format of every peak list must be given in the string **plformat**. The name given by **protein** will be used for output files, namely the final overview and coordinate files. Optionally, the **minimizer** macro may be changed and reference peak lists (**peak_ref**) containing the correct assignment may be given. Agreement between NOAH assignment and reference as-

signment will be indicated in the final “noah.grf” file. The string given in **options** is used as parameters for the command **assign** (e.g. “**options**=”transposed=0” should be used if the transposed peaks should be checked in 3D peak lists). Upper and lower limits distance constraint files may be added to each NOAH structure calculation (e.g. constraints for known disulfide bonds) with the parameters **addupl** and **addlol**. The parameters **entry** and **exit** indicate at which cycle (of the *n* cycles) NOAH actually starts and ends, which allows to split a given NOAH calculation into several smaller jobs. The parameter **entry** can only be used if all files necessary for the given cycle are present in the current directory (i.e. structures, assignment-files etc.). The peaks are calibrated with the DYANA standard macro **caliba** after the 10th cycle if the option **calibrate** is specified.

For more informations on how to use this method please refer to the tutorial section.

noahanneal

Parameters... —

This is the standard annealing protocol used by NOAH. The input parameters correspond to those of the macro **anneal**. This protocol uses different weights than **anneal** and never includes protons in the van der Waals check.

noahmin

Parameters... —

This is the standard minimization protocol used by NOAH. The input parameters correspond to those of the macro **vtfmin**. This protocol uses different weights than **vtfmin** and never includes protons in the van der Waals check.

overview

name=*name* —
structures=*n* *all selected structures*
range=*residue range* *all residues*
ang cor pdb
hbond vdw full

Sorts the selected structures with regard to their target function value and creates an overview file *name*.ovw for the first *n* of these structures. If

the **name** parameter is not specified and if a variable with name **name** is defined, then its value is used as *name*.

Pair-wise RMSDs are calculated for the given *residue range* (see command **rmsd**). The RMSD calculation can be suppressed by setting **range=–**. If the **range** parameter is not specified and if a variable with name **rmsdrange** is defined, then its value is used as *residue range.*

Optionally, output angle (**ang**), DG coordinate (**cor**) or PDB coordinate (**pdb**) files of the structures may be written with file names "*name*.ang", "*name*.cor" or "*name*.pdb", respectively.

The structures may be analyzed for hydrogen bonds (option **hbond**), or for violations of steric lower distance limits (option **vdw**).

Note: Because the target function is re-calculated, it is important that all constraints used for the calculation of the structures are present and that the same weights are used.

An overview file may contain four different tables:

- For each structure: the target function value, the numbers, sums and maxima of constraint violations (the output of the **structure list** command).
- For each violated constraint: the structures in which it is violated by more than the corresponding cutoff value (the output of the **structure violate** command). By default, violations are shown only if they occur in at least one third of the conformers. To obtain a listing of all violations larger than the cutoffs, the option **full** must be set.
- For all pairs of structures: the RMSD for the backbone and all heavy atoms (output of the **rmsd** command). By default only the average value of all pairwise comparisons is written. A table with the individual pairwise RMSD values is created only if the option **full** is set.
- For all residues: the local RMSD for tri-peptide segments, and the displacements for backbone and all heavy atoms (output of the **rmsd** command). This table is only created if the option **full** is set.

peak abs

Replace all peak volumes by their absolute value.

peak create

distance=*d*	4.0
structures=*n*	1
additional	
c13 | n15	

Deletes current peak lists and creates expected peaks using the structures from the selected structure memories. Peaks are created if the distance between two assigned proton (or pseudo atom) chemical shifts is less

than *d* in at least *n* of the selected structures. To calculate distances where pseudo atoms are involved, a r^{-6} weighted average distance is determined between all protons that are represented by the pseudo atom. With the option **additional**, the current peaks are not deleted and the expected peaks are only added if they are not already present. Per default, a 2D peak list is created, but with the options **c13** or **n15**, a 3D ^{13}C- or ^{15}N-correlated NOESY peak list is simulated.

peak delete

Deletes all selected peaks.

peak deviations

Prints out a list of peaks where the deviations between the peak position and the assigned chemical shift are larger than the tolerance value given by the variable **tolerance**. If the information level is **full**, a histogram is written at the end with the number of deviating peaks in each dimension for different deviations (in ppm).

peak distance

Lists for all selected peaks the average, standard deviation, minimum and maximum of the corresponding distance in the selected structures.

peak list

Lists all selected peaks.

peak scale **factor=***f* —

Scales the volumes of the selected peaks by the factor *f*.

peak select *peak selection* —

Selects all peaks that match the given *peak selection* (see chapter Selections).

peak unassign **dim=***d* all dimensions

Deletes the assignment of the selected peaks. Optionally, only assignments in dimension *d* may be deleted.

peak unique **average** | **maximum** average

From each group of identically assigned peaks only one peak is kept. Peaks are considered as identically assigned if they are assigned to the same proton pair. For instance, in a 2D NOESY spectrum a cross peak and its transposed peak are "identically assigned." Peak volumes may be **average**d or their **maximum** be kept. Only peaks with positive volume are considered in the averaging procedure.

ramachandran

file=*name*
nobackground label

ramachandran.ps

Produces a graphics output file with the given *name* (a GRAF file if the extension is ".grf", a MIF file if the extension is ".mif", or a Postscript file otherwise) with a Ramachandran plot of the selected structures. The background consists of three different blue tones indicating "most favored regions" (dark blue), "additional allowed regions" (medium blue) and "generously allowed regions" (light blue). It corresponds to the background found in the program PROCHECK (Laskowski *et al*., 1993).

Optionally, the background can be omitted (**nobackground**) and the residues with backbone angles that lie outside of the allowed regions are **label**ed.

random_all

n

All available structure memories

Creates *n* random structures in the structure memories.

randomize

i

Actual seed number

Creates a random structure in the structure memory #0. The target function value is automatically set to 0. A new seed number *i* for the random number generator may be specified.

read aco

file=*file*
unknown=error|warning|skip
append

—

error

Reads an angle constraint *file*. Constraints that involve **unknown** residues or angles can either cause an **error**, a **warning**, or can be **skip**ped. Optionally, the angle constraints are **append**ed to those already present.

read ang

file=*file*	—
structure=*n*	1
unknown=error|**warning**|**skip**	error

Reads an angle *file*. If there is a DYANA header in the file, the target function value will be read from the header. Optionally, only the *n*-th structure may be read from a multi-conformer file. Otherwise, all *m* structures in a multi-conformer file are read and stored as structures 1,...,*m*. The first structure read will also be stored in the default structure memory, 0. The presence of **unknown** residues or angles can either cause an **error**, a **warning**, or can be **skip**ped.

read cco

file=*file*	—
unknown=error|**warning**|**skip**	error
append	

Reads a coupling constant *file*. Coupling constants that involve **unknown** residues or atoms can either cause an **error**, a **warning**, or can be **skip**ped. Optionally, the coupling constants are **append**ed to those already present.

read cor

file=*file*	—
structure=*n*	1
unknown=error|**warning**|**skip**	error

Reads a coordinates *file* in DG format. If there is a DYANA header in the file, the target function value will be read from the header. Optionally, only the *n*-th structure may be read from a multi-conformer file. Otherwise, all *m* structures in a multi-conformer file are read and stored as structures 1,...,*m*. The first structure read will also be stored in the default structure memory, 0. The presence of **unknown** residues or atoms can either cause an **error**, a **warning**, or they can be **skip**ped.

read lib

file=*file*
convert=*file*

—

Reads a residue library. Optionally, a library file with atom pointers converted from numeric to name format or *vice versa* may be written (see chapter File Formats).

read lol

file=*file*
unknown=error|**warning**|**skip**
append

—
error

Reads a lower limit distance constraints *file*. Constraints that involve **unknown** residues or atoms can either cause an **error**, a **warning**, or can be **skip**ped. Optionally, the distance constraints are **append**ed to those already present.

read ori

file=*file*
unknown=error|**warning**|**skip**
append

—
error

Reads an orientation constraint *file*. Constraints that involve **unknown** residues or atoms can either cause an **error**, a **warning**, or can be **skip**ped. Optionally, the orientation constraints are **append**ed to those already present.

read pdb

file=*file*
structure=*n*
unknown=error|**warning**|**skip**
all

—
1
error

Reads a coordinates *file* in PDB format. If there is a DYANA header in the file, the target function value will be read from the header. Optionally, only the *n*-th structure may be read from a multi-conformer file. Otherwise, all *m* structures in a multi-conformer file are read and stored as structures 1,...,*m*. The first structure read will also be stored in the default

structure memory, 0. The presence of **unknown** residues or atoms can either cause an **error**, a **warning**, or they can be **skip**ped.

read peaks

file=*file*	—
weight=*w*	1.0
filter=*s_1,s_2,...*	*no filter*
format=*string*	hH
reference	
integrated assigned append	

Reads a XEASY (Bartels *et al.,* 1995) peak list. Volumes are scaled with the weight factor *w.* The **filter** option allows to skip peaks with comments that match one of the strings s_1,s_2,...

For 3D NOESY peak lists, the **format**, i.e. the order in which chemical shifts and assignments are given in the peak list, may be specified. The format *string* has one character per dimension that identifies the column of ^{15}N or ^{13}C atoms ("N" or "C"), the column of protons bound to ^{15}N or ^{13}C ("H"), and the column of "independent" protons ("h"). If the format parameter is absent, the program uses the format given in the peak list header line "#DYANAFORMAT *string*", or, if no such header line is present, tries to determine the format from the peak assignments (if possible). Regardless of this input order DYANA permutes these dimensions to "hHN" or "hHC" in 3D lists so that dimension 3 is always the heteroatom dimension and dimension 2 is the proton dimension coupled to it.

The option **reference** is used to read in a peak list as reference list for NOAH.

Optionally, only **integrated** peaks, i.e. those with an integration method flag different from "–", or only **assigned** peaks, i.e. those that are assigned in both proton dimensions, are read. Optionally, the peaks are **append**ed to those already present.

```
read peaks n15 format=NhH filter=overlap
```

Reads a peak list named "n15.peaks". The three columns for the chemical shifts and the corresponding assignments in the peak list file refer to ^{15}N, the "independent" proton, and the proton bound to ^{15}N. Peaks with comment "overlap" are skipped.

read prot

file=*file* —
tolerance=Δω ∞
add

Reads a XEASY chemical shift list. A warning message is printed if chemical shifts are present simultaneously for an atom and its corresponding pseudo atom. Optionally, only chemical shifts of currently unassigned atoms are **add**ed. For chemical shifts that are present in both lists and that differ by more Δω (in ppm), a warning is printed.

read seq

file=*file* —

Reads a residue sequence.

read upl

file=*file* —
unknown=error|**warning**|**skip** error
append

Reads a upper limit distance constraints *file*. Constraints that involve **unknown** residues or atoms can either cause an **error**, a **warning**, or can be **skip**ped. Optionally, the distance constraints are **append**ed to those already present.

read xplor

file=*file* —
unknown=error|**warning**|**skip** error
append

Reads a *file* with conformational constraints in XPLOR format (Brünger, 1992). A simplified version of the atom selection syntax of XPLOR is used. Constraints that involve **unknown** residues or atoms can either cause an **error**, a **warning**, or can be **skip**ped. Optionally, constraints are **append**ed to those already present.

read_all

list of files —

Reads all given angle, coordinate or PDB files and stores them into the structure memories. File name may contain asterisk and question marks to select several structures at one time (e.g. “er*.ang” or “er???.cor”). The file name extension decides on the format of the individual files: files with filename extension “.ang” are read as angle files, files with filename extension “.pdb” are read as PDB coordinate files, and other files are read as DG coordinate files.

readdata

name . . . —

Reads input data files with the given *name*. If *name* has an extension (i.e. if it contains a “.”), a file with the corresponding format (as given by the extension) is read. Otherwise, the sequence file “*name*.seq” and, if available, the upper limits distance constraints file “*name*.upl”, the lower limits distance constraints file “*name*.lol” and the angle constraints file “*name*.aco” are read. If no residue library is present, the standard DYANA library (“dyana.lib”) is read in advance.

redac

name=*name*	—
schedule=*schedule*	0.4,0.0,0.0
structures=*n*	50
steps=N_1,N_2,N_3	150,400,800
minimizer=*macro*	vtfmin

Performs REDAC cycles (Güntert & Wüthrich, 1991) with *n* structures according to the given *schedule*. Overview and angle files of every cycle are written to the files “*name**.ovw” and “*name**.ang” where the asterisk is replaced by “a”, “b”, “c” etc. for successive cycles. The *schedule* is a comma-separated list of **ang_cut** values that will be used to generate redundant dihedral angle constraints. Structures are calculated using the given *macro* for minimization. This *macro* must accept the same parameters as the standard variable target function minimization macro, **vtfmin**. A zero or negative **ang_cut** value means that no redundant angle constraints will be generated in this cycle. The next cycle will therefore use the original angle constraints to minimize the current structures on the last level during N_3 iterations. Otherwise, i.e. if **ang_cut** was positive in the previous cycle, structures will be calculated using N_1 and N_2

minimization steps at intermediate levels and at the final level, respectively.

`redac er2 schedule=1.0,1.0,0.4,0.0,0.0 50`

In the first cycle, 50 structures are calculated with the original constraints and angle constraints are generated with **ang_cut** = 1.0. After this, new structures are calculated three times using the previously generated constraints. No new angle constraints are generated the third time. Finally, the structures are minimized on the last level. In this cycle too, no angle constraints are generated.

reliability

dist — 1.0

Calculates the reliability distance (RD, Mumenthaler & Braun, 1995) of all unambiguously assigned peaks using the given tolerance range and the structures in the structure memory. If *dist* is specified, the number of assigned peaks with a RD above *dist* will be written. After this command, all peaks that cannot be explained with the current structures and the given **tolerance** range (see System Variables) are selected.

`reliability 1.0`

Calculate the reliability distance of all peaks.

`write peaks incomp.peaks selected`

Write all selected peaks into the peak file “incomp.peaks”.

rmsd

range=*residue range* — *all residues*

segment=*n* — *3*

Calculates pair-wise root-mean-square deviation (RMSD) between all pairs of selected structures (McLachlan, 1979) for the backbone atoms and for all heavy atoms. Optionally, a residue range for the superposition may be specified.

For two sets of n atoms each, $\vec{r}_1, \ldots, \vec{r}_n$ and $\vec{q}_1, \ldots, \vec{q}_n$, with $\sum_i \vec{r}_i = \sum_i \vec{q}_i = 0$, the RMSD is defined by

$$\mathrm{RMSD} = \min_{R \in SO(3)} \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left| \vec{r}_i - R\vec{q}_i \right|^2} \qquad [5]$$

R denotes a rotation matrix, and *SO*(3) the rotation group.

If the information level is **full**, local RMSDs and global displacements are calculated for each residue. Local RMSDs for residue *i* are calculated for the segment of *n* residues, $i - n,...,i,...,i + n$ (*n* odd).

seqplot

file=*file* — seqplot.ps

Analyses the upper limit distance constraints and draws a sequence plot in FrameMaker MIF (if the *file* extension is ".mif") or Postscript format.

The first three lines below the amino acid sequence represent torsion angle restraints for the backbone torsion angles ϕ and ψ, and for the side-chain torsion angle χ^1. For ϕ and ψ a triangle pointing upwards indicates a restraint that allows the torsion angle to take the values observed in an ideal α-helix ($\phi = -57^\circ$, $\psi = -47^\circ$) or 3_{10}-helix ($\phi = -60^\circ$, $\psi = -30^\circ$); a triangle pointing downwards indicates compatibility with an ideal parallel or antiparallel β-strand ($\phi = -119^\circ$, $\psi = 113^\circ$ or $\phi = -139^\circ$, $\psi = 135^\circ$, respectively; Schultz & Schirmer, 1979); a restraint represented by a star encloses conformations of both α and β secondary structure types; and a filled circle marks a restraint that excludes the torsion angle values of these regular secondary structure elements. Torsion angle restraints for χ^1 are depicted by filled squares of three different decreasing sizes, depending on whether they allow for one, two, or all three of the staggered rotamer positions $\chi^1 = -60, 60, 180^\circ$. Torsion angle restraints for χ^1 that exclude all three staggered rotamer positions are shown as filled circles. Upper distance limits for sequential and medium-range distances are shown by horizontal lines connecting the positions of the two residues involved. The thickness of the lines for the sequential distances $d_{NN}(i, i + 1)$, $d_{\alpha N}(i, i + 1)$ and $d_{\beta N}(i, i + 1)$ is inversely proportional to the squared upper distance bound.

This command should be executed before **distance modify** because many of the intra-residual and short-range distance constraints will be removed by **distance modify** because the do not effectively restrict the conformation.

ssbond

R_1–R_2 ... — —

Creates the standard upper and three lower limit distance constraints (Williamson *et al.*, 1985) to enforce disulfide bonds between pairs R_1–R_2, R_3–R_4 etc. of cystine residues. These residues must be of the type

"CYSS". For a disulfide bridge between residues *i* and *j*, three upper limits and three lower limits are generated:

$$\begin{aligned} 2.0 \le d(S_i^{\gamma}, S_j^{\gamma}) \le 2.1\text{Å} \\ 3.0 \le d(C_i^{\beta}, S_j^{\gamma}) \le 3.1\text{Å} \\ 3.0 \le d(S_i^{\gamma}, C_j^{\beta}) \le 3.1\text{Å} \end{aligned} \qquad [6]$$

stereoassign

angle selection	*angles of current fragment*
tfcut=f_{max}	0.0
conformations=*name*	*none*

Tries to find stereospecific assignments by systematic analysis of the local conformation of a molecular fragment with grid searches. If there are *n* pairs of (stereospecificly unassigned) diastereotopic substituents within the molecular fragment, 2^n grid searches will be performed, one for each possible combination of stereospecific assignments. If a (connected) angle selection is specified, then it defines the molecular fragment that will be analyzed; otherwise, the fragment set in the preceding **grid fragment** command will be used. The parameter **tfcut** has the same meaning as in the **grid search** command. Optionally, the total number of allowed **conformations** can be stored in a variable with the given *name*. Grid searches are restricted to values of the torsion angles given in the standard grid memory, **A**, on input. On output, the allowed values of the torsion angles are again stored in grid memory **A**.

structure clear

all

Deletes the selected or simply **all** structures.

structure copy

from=*n*	—
to=*m*	—
name=*name*	*none*

Copies structure *n* to structure *m*. The current structure has number 0. Optionally, the structure can be given a new *name*.

structure insert

name=*name*

Inserts the current structure into the sequence of stored structures according to its target function value. Optionally, the structure can be given a *name*.

structure list

sum | **average** | **rms**

Lists target function value and statistics of restraint violations for all selected structures. For each type of restraints (upper distance limits, lower distance limits, van der Waals lower distance limits, torsion angle restraints, coupling constants, and orientational restraints) the number of violations exceeding the cutoff value (variables **cut_upl**, **cut_lol** etc.), either the **sum**, **average** or **rms** (root-mean-square) violation, and the maximal violation will be given. Average and rms violation cannot be calculated for van der Waals restraints.

structure select

structure selection — *all structures*
first=*n* — *all*

Selects structures according to the given *structure selection* (see chapter Selections). Optionally, the selection may be restricted to the **first** *n* structures that are matched by the *structure selection.*

structure sort

Sorts the selected structures according to their target function value.

structure violate

structures=*n* — 1
delete

Lists violations of distance constraints and angle constraints violations that exceed in at least *n* of the selected structures the cutoffs given by the variables **cut_upl**, **cut_lol** and **cut_aco**. Optionally, all violated constraints that are listed may be **delete**d.

```
structures select 1..20
structure violate structures=10 delete
```

Deletes all distance and angle constraints that are violated in at least 10 out of the selected 20 structures.

sugarbond

*r***ange=***residue range*	*all nucleotides*

Defines the correlations between the dihedral angles and the pseudorotation angle *P* in DNA/RNA sugar rings:

sugarring

residue number	*residue of current fragment*

Creates 5 upper and 5 lower limit constraints to "close" the bonds between C4’ and O4’ in the ribose rings of the nucleotides in the given *residue range*.

translate

xplor | **on** | **off** | **clear**

Defines atom and angle name translations between the nomenclature used in the standard DYANA residue library and other commonly used nomenclature systems. This allows reading of input files and writing of output files according to other nomenclature systems. Currently, **xplor** nomenclature is supported. The options **on**, **off**, and **clear** have the same meaning as in the **atom rename** and **angle rename** commands. Without option, **translate** lists the currently set atom and angle name translations.

```
translate xplor                    Set XPLOR translation table
read xplor noe.tbl                 Read a XPLOR distance constraint file
read pdb in.pdb unknown=warning
                                   Read a PDB file with XPLOR nomenclature.
                                   Avoid errors if unknown atoms are encountered.
translate off                      Use standard DYANA nomenclature again
...
translate on                       Switch to XPLOR nomenclature
write pdb out.pdb                  Write a PDB file with XPLOR nomenclature.
```

vtfmin

levels=L_1,L_2	0,*number of residues*
steps=N_1,N_2	150,400
flatsteps=n_1,n_2	50,100
tf	

Performs a standard variable target function minimization (Güntert *et al.*, 1991a) starting at minimization level L_1 and ending at minimization level L_2.

At each of the lower levels (i.e. those below L_2) N_1 minimization steps are performed, the minimization is stopped if n_1 steps failed to decrease the target function by at least 1%, and the steric repulsion is considered only for the heavy atoms. The weight for steric lower limits is 0.2.

At minimization level L_2 three times N_2 minimization steps are performed, the minimization is stopped if n_2 steps failed to decrease the target function by at least 1%, and the steric repulsion is considered for all atoms. The weight for steric lower limits is 0.2 for the first N_2 minimization steps, then it is increased to 0.6 for the following N_2 minimization steps, and to 2.0 for the final N_2 minimization steps.

If the option **tf** (and none of the other parameters) is given, the final target function value is calculated, without performing any minimization.

watsoncrick

strand1=*residue range*	—
strand2=*residue range*	—
planar	

Creates restraints to enforce standard Watson-Crick-type pase pairing between two antiparallel DNA or RNA strands. The two strands must have the same length and continuous numbering. Optionally, additional **planar**ity restraints can be added that restrain the interstrand C1'–C1' distances to 10.35–10.65 Å for A-T and 10.6–10.9 Å for C-G base pairs.

write aco

file=*file*	—
structures=*N*	10
maxwidth=$\Delta\phi$	270.0°
redac append	

Writes an angle constraint *file*. Optionally, the output may be **append**ed to an existing *file*.

Optionally, redundant dihedral angle constraints for the REDAC strategy (Güntert & Wüthrich, 1991) may be derived from the current angle statistics and included in the output (option **redac**; see the commands **angstat make** and **redac**). In order to generate redundant dihedral angle constraints for a given residue, its local target function value (and that of its immediate neighbors must be below the cutoff value given by the variable **ang_cut** in at least *N* structures. Redundant dihedral angle con-

straints with an allowed range wider than $\Delta\phi$ degrees are discarded. The parameters **structures** and **maxwidth** can only be in conjunction with the **redac** option.

write ang

file=*file*
fixed all append

—

Writes an angle *file*. Optionally, the output may be **append**ed to an existing *file*.

By default, the values of all rotatable dihedral angles of the current structure conformation are written. The values of the **fixed** dihedral angles (e.g. peptide bond angles) may be written, too. Optionally, the angles of **all** selected structures may be written.

write ass

file

Write an assignment file that is used by the NOAH command **filter**. For every possible peak assignment an entry is made. Peaks from the three internal NOAH assignment lists are saved (see **filter**). Assignments from the ambiguous and unambiguous list also have the distance constraint (in Å) that was derived from the peak volume and the assignment.

write cor

file=*file*
connect all append

—

Writes a coordinate *file* in DG (Distance Geometry) format. Optionally, the output may be **append**ed to an existing *file*.

By default, the Cartesian coordinates of the current structure are written. The covalent **connect**ivities may be included, too. Optionally, the Cartesian coordinates of **all** selected structures may be written.

write lol

file=*file*
append

—

Writes a lower limit distance constraint *file*. Optionally, the output may be **append**ed to an existing *file*.

write pdb

file=*file*
all append

—

Writes a coordinate *file* in PDB (Protein Data Bank; Bernstein *et al.*, 1977) format. Optionally, the output may be **append**ed to an existing *file*.

By default, the Cartesian coordinates of the current structure are written. Optionally, the Cartesian coordinates of **all** selected structures may be written.

write peaks

file=*file*
selected append

—

Writes a peak list in XEASY format (Eccles *et al.*, 1991; Bartels *et al.*, 1995). Optionally, the output may be **append**ed to an existing *file*.

By default all peaks are written. Optionally, only the **selected** peaks are written.

write prot

file=*file*
append

—

Write a chemical shift list (traditionally called “proton list”) in XEASY format (Eccles *et al.*, 1991; Bartels *et al.*, 1995). Optionally, the output may be **append**ed to an existing *file*.

write upl

file=*file*
append

—

Writes an upper limit distance constraint *file*. Optionally, the output may be **append**ed to an existing *file*.

write_all

name=*name*
ang cor pdb

—

cor

Writes all selected structures to angle (**ang**), DG coordinate (**cor**), or PDB coordinate (**pdb**) files with names "*namennn*.ang", "*namennn*.cor", or "*namennn*.pdb", respectively. *nnn* denotes the structure number.

Variables and Functions

DYANA gives the user access to internal variables and functions of the program through system variables and functions. With system variables the user can obtain and set parameters of the program; with functions the user can obtain the value of parameters of the program but he cannot change them.

System variables

The following is an alphabetical list of all DYANA system variables.

cut_aco

Cutoff value for angle constraint violations (in degrees). Only violations larger than this value are listed in the commands **structure list** and **structure violate.**

Initial value: 5.0°.

cut_cco

Cutoff value for coupling constant restraint violations (in Hz). Only violations larger than this value are listed with the commands **structure list** and **structure violate.**

Initial value: 0.5 Hz.

cut_lol

Cutoff value for lower limit distance constraint violations (in Å). Only violations larger than this value are listed in the commands **structure list** and **structure violate.**

Initial value: 0.2 Å.

cut_ori

Cutoff value for orientation restraint violations (in Hz). Only violations larger than this value are listed with the commands **structure list** and **structure violate.**

Initial value: 0.1 Hz.

cut_tflocal

Cutoff value for the maximal target function value (in $Å^2$) that a single residue is allowed to have to be included into the angle statistics of the **angstat make** command.

Initial value: 0.2 $Å^2$.

cut_upl

Cutoff value for upper limit distance constraint violations (in Å). Only violations larger than this value are listed with the commands **structure list** and **structure violate.**

Initial value: 0.2 Å.

cut_vdw

Cutoff value for van der Waals violations (in Å). Only violations larger than this value are listed with the commands **structure list** and **structure violate.**

Initial value: 0.2 Å.

gridtime

The maximal expected computation time of a grid search. If a grid search is expected to take longer than this value it is aborted.

Initial value: 60 s.

gridpoints

The maximal expected number of grid points to be checked in a grid search. If this value is exceeded the grid search will not be started.

Initial value: 10^{20}.

hb_len

Maximal proton-acceptor distance for a hydrogen bond.

Initial value: 2.4 Å.

hb_ang

Maximal angle between the donor-proton bond and the line connecting acceptor and donor for a hydrogen bond.

Initial value: 35˚.

level

Minimization level, *L* (Güntert *et al.*, 1991). Only distance constraints between atoms not more than *L* residues apart are considered in the target function. *L* = 0: only intraresidual, *L* = 1: intraresidual and sequential constraints, etc.

Initial value: *number of residues* (i.e., use all distance constraints).

maxamb NOAH variable: Maximum number of possible assignments a peak can have to be taken into the test assignment list.
Initial value: 2.

nstep Number of steps, *n*, per dihedral angle in grid searches. The grid searches will run over the *n* angle values 0, Δ, 2Δ, ..., $(n-1)\Delta$, where $\Delta = 2\pi/n$.
Initial value: 36 (i.e. $\Delta = 10^{\circ}$).

obsdis Maximal distance for which an NOE can be observed. Used by automatic calibration (function **calscale**) and for automatic assignment.
Initial value: 5.0 Å.

ori_axial Axial component D_{ax} of the tensor that relates residual dipolar couplings to the orientation of the corresponding chemical bond:

$$\delta(\theta, \phi) = D_{ax}(3\cos^2\theta - 1) + \frac{3}{2}D_{rh}(\sin^2\theta \cos 2\phi) \qquad [7]$$

$\delta(\theta, \phi)$ is the residual dipolar coupling as a function of the polar angles θ and ϕ of the chemical bond with respect to the principal axes system of the tensor *D*.
Initial value: 1.0 Hz.

ori_rhombic Rhombic component D_{rh} of the tensor that relates residual dipolar couplings to the orientation of the corresponding chemical bond. See Eq. [7].
Initial value: 0.0 Hz.

seed Random number generator seed.
Initial value: 3771.

soft_aco Cutoff for angle restraint violations for allowed conformations in grid searches.
Initial value: 5.0°.

soft_cco Cutoff for scalar coupling constant restraint violations for allowed conformations in grid searches.
Initial value: 0.5 Hz.

soft_lol Cutoff for lower limit distance restraint violations for allowed conformations in grid searches.
Initial value: 0.1 Å.

soft_upl

Cutoff for upper limit distance restraint violations for allowed conformations in grid searches.

Initial value: 0.1 Å.

soft_vdw

Cutoff for steric lower limit distance restraint violations for allowed conformations in grid searches.

Initial value: 0.1 Å.

tf_beta

Value of the parameter β if target function type 3 or 4 used (see system variable **tf_type**).

Initial value: 1.0

tf_type

Type of target function used for distance constraints. The same functional form is used for upper limits, lower limits, and van der Waals lower limits.

type	term for a violated upper limit ($d > b$)	limiting cases	
		small violation ($d \approx b$)	large violation ($d >> b$)
1	$\left(\frac{d^2 - b^2}{2b}\right)^2$	$(d-b)^2$	$\frac{1}{4b^2}d^4$
2	$(d-b)^2$	$(d-b)^2$	d^2
3	$\frac{\beta^2}{2}\left[\sqrt{1+\left(\frac{d^2-b^2}{\beta b^2}\right)^2}-1\right]$	$\left(\frac{d-b}{b}\right)^2$	$\frac{\beta}{2}\left(\frac{d}{b}\right)^2$
4	$2\beta^2 b^2\left[\sqrt{1+\left(\frac{d-b}{\beta b}\right)^2}-1\right]$	$(d-b)^2$	$2\beta b \cdot d$

d, *b* and β denote the actual distance, the upper distance bound, and the value of the system variable **tf_beta**. The larger the value of β, the longer the functional form will be close to the limiting case for small violations.

Type 1 is the normal DIANA target function (Güntert *et al.*, 1991), and type 3 is the error-tolerant target function used by NOAH (Mumenthaler *et al.*, 1997).

Target functions of type 1, 2 and 4 have unit $Å^2$, whereas the target function of type 3 is dimensionless, i.e. target function values obtained with type 3 cannot be compared with those obtained with other types.

In all cases the contribution to the target function from a small violation is proportional to the square of the violation. For large violations, the target function types differ significantly: type 1 is proportional to d^4, types 2 and 3 are proportional to d^2, and type 4 is linear in d.

Note that distance constraints with very small upper bound b can lead to problems when the target function of type 1 is used because they get an excessive weight over other constraints. To illustrate this, assume two upper limit distance constraints that are violated by the same amount:

b = 0.1 Å, d = 2 Å: target function contribution = 398.0 $Å^2$

b = 3.1 Å, d = 5 Å: target function contribution = 6.2 $Å^2$

The first constraint gives a more than 60 times larger contribution than the second! In such cases it is advisable to use the target function of type 2, to which both constraints would contribute the same amount.

Initial value: 1.

tolerance

Tolerance ranges Δ_{tol} between peak positions and proton chemical shifts (in ppm). In automatic NOESY assignment, these values are used for atoms that are already used in peak assignments of the current peak list. The value of **tolerance** is a comma-separated list of values for the different spectral dimensions: the first and second numbers apply to protons, the third number to ^{13}C or ^{15}N. The second number is for protons that are directly bound to the corresponding ^{13}C or ^{15}N atom, the first number for other protons.

Initial value: 0.01, 0.01, 0.2 ppm.

tol_transp

Chemical shift tolerance ranges used to check for the existence of tranposed peaks in 3D peak lists, given in the same format as for the variable **tolerance**.

Initial value: 0.05, 0.05, 0.5 ppm.

tol_una

Same as variable **tolerance**, but the Δ_{tol} values given here are used in automatic NOESY assignment for all chemical shifts from atoms which are not used in any peak assignment of the current peak list.

Initial value: 0.04, 0.04, 0.4 ppm.

weight_aco

Weight value for contributions to the target function from dihedral angle constraints.

Initial value: 5.0.

weight_cco Weight value for contributions to the target function from coupling constant constraints.
Initial value: 0.5.

weight_lol Weight value for contributions to the target function from lower limit distance constraints.
Initial value: 1.0.

weight_ori Weight value for contributions to the target function from orientational constraints.
Initial value: 10.0.

weight_upl Weight value for contributions to the target function from upper limit distance constraints.
Initial value: 1.0.

weight_vdw Weight value for contributions to the target function from van der Waals lower limit distance constraints.
Initial value: 2.0.

Functions

In the following alphabetical list of all DYANA functions arguments are denoted by

n, m	integer
r	real
s	string
x	integer or real, unless types are given explicitly

The result type of a function is only given explicitly if it differs from the type of the argument(s), and if it is not obvious.

Several functions give access to internal data structures used by DYANA to store information about residues, atoms and dihedral angles. Internally, residues, atoms, and dihedral angles are numbered consecutively from 1 to **nr**, **na**, and **nd**, respectively. The "residue *n*", "atom *n*", and "dihedral angle *n*" refer to these internal numberings.

For residues this internal number is called the *residue index*. Residues have also an external *residue number*, which is used in the input and output of the program, and which can differ from the residue index. For example if a sequence starts with residue "ALA 101", this first residue has

index 1 and (external) number 101.

acoviol(*n*,*r*) Real function that returns the size of the angle restraint violation for a given value *r* (in degrees) of the torsion angle *n*. This function returns a negative result if there exists no angle restraint for angle *n*, and 0 if the restraint(s) are not violated.

anam(*n*) Character function that returns the name of atom *n*.

angle(*n*) Character function that returns a string consisting of the angle name and the residue number of dihedral angle *n*.

Angle(*n*) Character function that returns a string consisting of the angle name, the residue name, and the residue number of dihedral angle *n*.

atom(*n*) Character function that returns a string consisting of the atom name and the residue number of atom *n*.

Atom(*n*) Character function that returns a string consisting of the atom name, the residue name, and the residue number of atom *n*.

calscale(*s*,r_1,r_2) This function is used for the automatic calibration procedure included in the **caliba** macro. The string *s* contains the calibration function *f(d)* relating the peak volume *f* with a corresponding distance *d*. Using this function, **calscale** determines a scaling factor *A* such that the average distance bound of all peaks calibrated with the function *A f*(*d*) becomes r_1. Only peaks with volume greater that r_2 are taken into account.

cco(*n*) Coupling constant value of the *n*-th coupling constant restraint.

coord(*m*,*n*) Cartesian coordinate *m* (= 1, 2, 3) of atom *n*.

derms Real function that returns the RMS total energy change per timestep, averaged over all timesteps of the most recently executed **md** command.

diastereotopic(*n*) Number of the *n*-th diastereotopic atom in the current grid search fragment. In a pair of diastereotopic atoms, e.g. HB2/HB3, only the first one will be counted. For values of *n* larger than the number of diastereotopic pairs in the fragment, the function returns 0.

dmax Real function that returns the maximal dihedral angle change (in degrees) per timestep, averaged over all timesteps of the most recently executed **md** command.

dnam(*n*) Character function that returns the name of dihedral angle *n*.

drms Real function that returns the RMS dihedral angle change (in degrees) per timestep, averaged over all timesteps of the most recently executed **md** command.

dval(*n*) Real function that returns the value (in degrees) of dihedral angle *n*.

ekin Real function that returns the current kinetic energy (in target function units).

ekmean Real function that returns the mean kinetic energy, averaged over all timesteps of the most recently executed **md** command.

ekrms Real function that returns the standard deviation of the kinetic energy, averaged over all timesteps of the most recently executed **md** command.

element(*n*) Ordinal number of the atom *n*, e.g. 1 for hydrogen, 6 for carbon atoms.

emean Real function that returns the mean total energy, averaged over all timesteps of the most recently executed **md** command.

erms Real function that returns the standard deviation of the total energy, averaged over all timesteps of the most recently executed **md** command.

heavyatom(*n*) Number of the heavy atom associated with (hydrogen or pseudo) atom *n*.

iar(*n*) Residue index of atom *n*.

iacod(*n*) Number of the dihedral angle that is restrained by the *n*-th dihedral angle restraint.

iangle(*s*) Internal number of the dihedral angle with name *s*. The string *s* consists of the angle name followed by the residue number. The function returns 0 if *s* does not identify an existing angle.

iatom(*s*) Internal number of the atom with name *s*. The string *s* consists of the atom name followed by the residue number. The function returns 0 if *s* does not identify an existing atom.

iaunit(*n*) Rigid unit number of a atom *n*, i.e. the number of the dihedral angle immediately preceding atom *n*.

ibond(*n*,*m*) Number of the *m*-th atom that is covalently bound to atom *n* (*m* = 1, ..., 4). The function returns 0 if less than *m* atoms are bound to atom *n*.

iccoa(*m*,*n*) Numbers of the two atoms (*m* = 1, 2) involved in the *n*-th coupling constant restraint.

ida(*n*,*m*) Number of the *m*-th defining atom of dihedral angle *n* (*m* = 1, ..., 4).

idcoa(*m*,*n*) Numbers of the two atoms (*m* = 1, 2) involved in the *n*-th distance restraint.

idord(*n*) Index of dihedral angle *n* with respect to the original order of dihedral angles.

idr(*n*) Residue index of dihedral angle *n*.

ifira(*n*) Number of the first atom belonging to residue *n*.

ifird(*n*) Number of the first dihedral angle belonging to the residue *n*.

interval(*i*,*j*,*n*) Lower (*j* = 1) or upper (*j* = 2) bound of the *i*-th allowed interval for angle *n* found by grid searches.

intervals(*n*) Number of allowed intervals for angle *n* found by grid searches.

iprev(*n*) Number of the dihedral angle that precedes dihedral angle *n* in the tree structure of dihedral angles.

irnum(*n*) Index of the residue with external residue number *n*.

istruct(*n*) Number of the *n*-th selected structure.

lda(*n*) Number of the last atom that is affected by a change of dihedral angle *n*.

libdir Character function that returns the current library directory. The library directory name is taken from the environment variable **DYANALIB** when the program starts.

maxang Maximal number of structures for which the dihedral angles can be stored. This value depends on the size of the protein.

maxcor Maximal number of structures for which the Cartesian coordinates can be stored. This value depends on the size of the protein.

na Number of atoms.

naco Number of dihedral angle constraints.

nassign Number of assigned peaks.

nbond(*n*) Number of atoms that are covalently bound to atom *n*.

ncco Number of coupling constant constraints.

nconf Number of allowed conformations in the most recent grid search.

nd Total number of dihedral angles, free and fixed.

ndcdis(*n*) Number of distance constraints between residues that are exactly *n* positions apart in the primary sequence.

ndco Number of distance constraints.

ndcres(*n*,*m*) Number of distance constraints that involve the residue *n* and span a distance of at least *m* positions in the sequence.

ndfree Number of free (i.e. rotatable) dihedral angles.

nlevel Number of distance constraints between residues that are less than *n* positions apart in the primary sequence.

nlol Number of lower limit distance constraints.

np_ass NOAH variable: Number of peaks in the unambiguous assignment list after the command **filter**.

np_corr NOAH variable: Number of peaks in the unambiguous assignment list that have the same assignment as in the reference peak list after the command **filter**.

np_inc NOAH variable: Number of peaks that are incompatible with the current structures after the command **assign**.

np_new NOAH variable: Number of peaks in the unambiguous assignment list that have no assignment in the reference peak list after the command **filter**.

np_out NOAH variable: Number of peaks that have no possible assignment based on peak positions and proton chemical shifts after the command **assign**.

np_wrg NOAH variable: Number of peaks in the unambiguous assignment list that have a different assignment as in the reference peak list after the command **filter**.

npeaks Number of peaks.

nplist Number of peak lists.

nr Number of residues.

nseldis Number of selected distance constraints.

nstruct Number of selected structures.

numpro(*n*) Number of atoms that are associated with the pseudoato *n*.

nupl Number of upper limit distance constraints.

pi Numerical constant $\pi = 3.14159$.

pseudoatom(*n*) Number of the pseudoatom associated with atom *n*. If no pseudoatom is associated with atom *n*, the function returns 0.

rad Numerical constant $180/\pi = 57.2958$.

rmsd_bb Average of the pair-wise backbone RMSD calculated with the command **rmsd**.

rmsd_bbdev Standard deviation of the pair-wise backbone RMSD calculated with the command **rmsd**.

rmsd_hv Average of the pair-wise heavy atom RMSD calculated with the command **rmsd**.

rmsd_hvdev Standard deviation of the pair-wise heavy atom RMSD calculated with the command **rmsd**.

rnam(*n*) Character function that returns the name of the residue *n*.

rnum(*n*) (External) residue number of the residue *n*.

seldis Average distance bound of all selected distance constraints.

selected(*n*) Logical function that returns 1 structure *n* is selected, or 0 otherwise.

shift(*n*) Real function that returns the chemical shift of atom *n*, or 999.0 if the chemical shift is unknown.

stereopartner(*n*) Number of the diastereotopic partner of atom *n*. If atom *n* has no stereopartner, the function returns 0. If the atom (and its stereopartner) are stereospecifically assigned (see command **atom stereo**), the function returns the number of the diastereotopic partner with a negative sign.

tf Target function value obtained from the most recent target function evaluation. If the current structure was read from a file that contained the target function value in its header, and if the target function was not re-evaulated after reading the file, then the value from the file header is returned.

tf(*n*) Target function value of structure *n*.

timestep Length of the time-step used in the last integration step of a previous **md** command.

tfcalc Target function value of the current structure, obtained by evaluation of the target function with the current constraints, weights etc.

tfmin Minimal local target function value in the most recent grid search.

tfmax Maximal local target function value in the most recent grid search.

tfres(*n*) Local target function value of residue *n* in the current structure, obtained by evaluation of the target function with the current constraints, weights etc.

tolcco(*n*) Tolerance for the coupling constant value of the *n*-th coupling constant restraint.

Selections

The DYANA command groups **atom**, **angle**, **peak**, **distance**, and **structure** apply to sets of *selected* atoms, angles, peaks, distance constraints, and structures, respectively. This chapter describes the syntax of the various selections used by DYANA.

Residue range

A residue range consists of one or several of the following elements, separated by commas:

m	a residue number
*m***..***n*	a range of residue numbers
*m***..**	from the residue with number *m* onwards
..*n*	from the first up to the residue *n*

Atom selection

Atom selections have the following general form (items in square brackets are optional, and items in curly braces can occur zero or more times):

[**!**] {*atom*} [*residue*] {*operator* {*atom*} [*residue*]}

where

!	denotes negation
atom	denotes an atom name, possibly containing wildcards ("?" or "*" replace exactly one or any number of characters, respectively), or the word **METHYL** to denote all atoms in methyl groups, or the word **AMIDE** to denote all atoms in amide groups.
residue	denotes a residue selection

A residue selection consists of one or several of the following elements:

@*name*	a residue *name*, possibly containing wildcards
@**FIRST**	the first residue

@LAST	the last residue,
@first	the first residue of every fragment with contiguous residue numbers
@last	the last residue of every fragment with contiguous residue numbers
m	a residue number
m..*n*	a range of residue numbers
m..	from the residue with number *m* onwards
..*n*	from the first up to the residue *n*

Atom selections can be combined using the following *operators*:

+	atoms in the current set or in the new set
–	atoms in the current set but not in the new set
/	atoms in the current set and in the new set

Operators are always evaluated from left to right. The current atom set is the set of atoms defined by what precedes the *operator.* The new atom set is the set of atoms defined by what follows the *operator.*

An empty atom selection selects all atoms.

`HA`	atoms called HA
`HA HB*`	all atoms called HA or HB...
`HA @ALA 10..20`	HA in ALA of residues 10–20
`HA @ALA - 10..20`	HA in ALA except in residues 10–20
`N CA C + 15 17 - H* Q*`	all backbone atoms and the sidechain heavy atoms of residues 15 and 17

The command **atom list** can be used to check atom selections.

Angle selection

Angle selections follow the same syntax as atom selections, except that angle names instead of atom names are specified. The command **angle list** can be used to check whether angle selections.

Peak selection

Peak selections are made with the **peak select** command and consist of two atom selections that are separated by a comma. The comma may be omitted if both atom selections consist of a single atom name. In addition, peak selections may contain one or several of the following conditions:

levels=*m*..*n*
: Select only peaks between residues that are between *m* and *n* residues apart.

volume=V_{min}..V_{max}
: Select only peaks with volume between V_{min} and V_{max}.

fraction=*p*

Select randomly only the fraction *p* of all peaks that would normally be selected. This option is useful to simulate peak lists.

variable Select only peaks that correspond to a variable distance.

peaklist=*filename* Select only peaks that were read from the peak list with the given *filename* (without extension).

The current peak selection may be combined with the previously made peak selection using one of the operators:

union Select peaks that are selected by any of the two selections.

intersection Select peaks that are selected by both selections.

xor Select peaks that are selected in exactly one of the two selections.

By default, previously selected peaks are not considered. The command **peak list** can be used to check peak selections.

Distance constraint selection

Distance constraint selections are made with the **distance select** command and follow the same syntax as peak selections, except that the conditions **volume=**V_{min}**..**V_{max} and **peaklist=***filename* cannot be used. The command **distance list** can be used to check distance constraint selections.

Structure selection

A structure selection consists of one or several of the following elements, separated by blanks:

m	a structure number
m..*n*	a range of structure numbers
m..	from the structure with number *m* onwards
..*n*	from the first up to the structure *n*

An empty structure selection selects all structures.

File Formats

This chapter describes the format of the input files to DYANA. Most input With the exception of residue library, angle and coordinate files all input files have “free format”, i.e. the exact positioning of the individual entries on a line is not important; subsequent input fields are separated by one or more blanks. In all input files except residue library and Cartesian coordinate files the character “#” indicates that the rest of the line is a comment that will be skipped by the program.

The program DYANA uses for all types of input and output data files default file name extensions if no extension is specified by the user:

file type	format	default extension
dihedral angle constraints	DIANA	.aco
dihedral angles	DIANA	.ang
coupling constants	HABAS	.cco
Cartesian coordinates	DG	.cor
residue library	DYANA	.lib
lower limit distance constraints	DIANA	.lol
orientation constraints	DYANA	.ori
Cartesian coordinates	PDB	.pdb
peak list	XEASY	.peaks
chemical shift list	XEASY	.prot
residue sequence	DIANA	.seq
upper limit distance constraints	DIANA	.upl
XPLOR distance and angle constraints	XPLOR	.xplor

Formats of most files are those of already existing programs: DIANA

(Güntert *et al.*, 1991a), HABAS (Güntert *et al.*, 1989), XEASY (Eccles *et al.*, 1991; Bartels *et al.*, 1995), and XPLOR (Brünger, 1992). DG is the coordinate file format used by DIANA and other distance geometry programs. PDB is the format used by the Protein Data Bank (Bernstein *et al.*, 1977). Some features of PDB and XPLOR format are not supported by DYANA.

Residue library

The residue library input file declares the atom types, the nomenclature, the dihedral angle definitions, the covalent connectivities and the standard geometry. The standard library, "dyana.lib", uses the standard geometry of the ECEPP/2 force field (Momany *et al.*, 1975; Némethy *et al.*, 1983) for all amino acid residue types. The covalent geometry of the nucleotides is based on the AMBER force field (Cornell *et al.*, 1995). For reasons of compatibility with other programs, the residue library used for DYANA contains more information than is actually read by the program; the following description treats only data that is relevant for DYANA. First of all, the present version of DYANA does not allow for special endgroups at the N- or C-terminus of the polypeptide chain. Therefore only the entries marked with the keywords ATOMTYPES or RESIDUE are considered.

The atom types entry starts with a header line with the Fortran format (A10,I5) containing the word ATOMTYPES and the number of atom type declarations that will follow. The following lines contain atom type declarations in the Fortran format (5X,A5,F10.2,2I5): the atom type, the repulsive core radius that will be assigned to atoms of this type, a code for hydrogen bond capabilities (1 for hydrogen atoms that can form hydrogen bonds, for hydrogen bond acceptors (e.g. oxygens), and 0 for atoms that cannot be involved in hydrogen bonds), and the order number of the chemical element (0 for pseudo atoms, 1 for hydrogen, 6 for carbon, 7 for nitrogen etc.). The atom types entry must precede the residue entries.

A residue entry starts with a header line with the Fortran format (A10,A5,4I5) and containing the word RESIDUE, the residue name, the numbers of rotatable dihedral angle and atom declarations that will follow, respectively, and the numbers of the first and last atom in the list of atom declarations that belong to the residue (not to the preceding or following one in the polypeptide chain). The next lines contain dihedral angle declarations in the format (5X,A5,20X,5I5): the dihedral angle name, the numbers of the four atoms that define the dihedral angle, and

the number of the last atom that will be affected by a rotation of the dihedral angle (for backbone dihedral angles this number is set to 0). Atom numbers correspond to the running numbers in the first column of the atom declarations. The atom declarations must be ordered such that the set of atoms affected by a change of a dihedral angle consists of all atoms following the third atom in the dihedral angle definition up to the last atom (or the C-terminus for backbone dihedral angles) that is affected. Finally, there are lines containing atom declarations: the format is (5X,2A5,15X,3F10.4,5I5), the data are the atom name, the atom type (used to set the repulsive core radii), the x-, y- and z-coordinates in for an arbitrary conformation, four atom numbers indicating covalent connectivities (if there are less than four connectivities, the corresponding numbers are set to 0) and the atom number of the corresponding pseudo atom (or 0 if there is no corresponding pseudo atom).

The nomenclature of atoms in amino acid residues closely follows the IUPAC recommendations. The only exception is the backbone amide proton which is called HN instead of H. In addition to real atoms the residue library may contain pseudo atoms identified by the atom type PSEUD that are used in DYANA as dimensionless reference points for distance constraints.

To avoid nomenclature confusion all atom types and residue entries of the standard residue library file are listed in the following (for compactness two numbers that are not used by DYANA are not printed in the atom lines between the atom type and the *x*-coordinate):

Atom types

```
ATOMTYPES    18
   1 PSEUD    -10.00    0    0
   2 H_ALI      1.00    0    1
   3 H_AMI      0.95    1    1
   4 H_ARO      1.00    0    1
   5 H_SUL      1.00    0    1
   6 H_OXY      1.00    1    1
   7 C_ALI      1.40    0    6
   8 C_BYL      1.40    0    6
   9 C_ARO      1.35    0    6
  10 C_VIN      1.40    0    6
  11 N_AMI      1.30   -1    7
  12 N_AMO      1.30    0    7
  13 O_BYL      1.20   -1    8
  14 O_HYD      1.20   -1    8
  15 O_EST      1.20   -1    8
  16 S_OXY      1.60    0   16
  17 S_RED      1.60    0   16
  18 P_ALI      1.60    0   15
```

ALA

```
RESIDUE   ALA      4   14    3   13
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   12    0
   3 CHI1     1    3    1.3500    3    5    8    9   11
   4 PSI      0    0    0.0000    3    5   12   14    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    8   12    0
   6 HA   H_ALI    1.7849   -0.4925    0.9140    5    0    0    0    0
```

```
  7 QB   PSEUD    2.0850   -0.9858   -1.4878    0    0    0    0    0
  8 CB   C_ALI    1.9637   -0.7966   -1.2023    5    9   10   11    0
  9 HB1  H_ALI    3.0537   -0.7963   -1.2019    8    0    0    0    7
 10 HB2  H_ALI    1.6006   -1.8224   -1.1392    8    0    0    0    7
 11 HB3  H_ALI    1.6006   -0.3386   -2.1223    8    0    0    0    7
 12 C    C_BYL    1.9587    1.4440    0.0000    5   13   14    0    0
 13 O    O_BYL    1.1648    2.3835    0.0000   12    0    0    0    0
 14 N    N_AMI    3.2772    1.5756    0.0000   12    0    0    0    0
```

ARG

```
RESIDUE   ARG      7   30    3   29
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   28    0
   3 CHI1     1    3    1.3500    3    5    7   11   27
   4 CHI2     1    3    1.3500    5    7   11   15   27
   5 CHI3     1    3    1.3500    7   11   15   19   27
   6 CHI4     0    0    0.0000   11   15   19   21   27
   7 PSI      0    0    0.0000    3    5   28   30    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   28    0
   6 HA   H_ALI    1.7317   -0.5213    0.9158    5    0    0    0    0
   7 CB   C_ALI    2.0038   -0.7402   -1.2205    5    8    9   11    0
   8 HB2  H_ALI    1.6375   -1.7668   -1.2235    7    0    0    0   10
   9 HB3  H_ALI    1.6375   -0.2685   -2.1323    7    0    0    0   10
  10 QB   PSEUD    1.6375   -1.0177   -1.6779    0    0    0    0    0
  11 CG   C_ALI    3.5338   -0.7388   -1.2182    7   12   13   15    0
  12 HG2  H_ALI    3.9001    0.2878   -1.2152   11    0    0    0   14
  13 HG3  H_ALI    3.9001   -1.2106   -0.3064   11    0    0    0   14
  14 QG   PSEUD    3.9001   -0.4614   -0.7608    0    0    0    0    0
  15 CD   C_ALI    4.0846   -1.4791   -2.4387   11   16   17   19    0
  16 HD2  H_ALI    3.7230   -2.5073   -2.4444   15    0    0    0   18
  17 HD3  H_ALI    3.7230   -1.0090   -3.3532   15    0    0    0   18
  18 QD   PSEUD    3.7230   -1.7581   -2.8988    0    0    0    0    0
  19 NE   N_AMI    5.5643   -1.4643   -2.4144   15   20   21    0    0
  20 HE   H_AMI    6.0160   -1.0017   -1.6515   19    0    0    0    0
  21 CZ   C_VIN    6.3425   -2.0364   -3.3576   19   22   24    0    0
  22 NH1  N_AMI    7.6548   -1.9625   -3.2357   21   23    0    0    0
  23 HH1  H_AMI    8.3002   -2.3586   -3.8888   22    0    0    0    0
  24 NH2  N_AMI    5.7870   -2.6745   -4.4096   21   25   26    0    0
  25 HH21 H_AMI    6.3705   -3.0956   -5.1040   24    0    0    0   27
  26 HH22 H_AMI    4.7919   -2.7260   -4.4945   24    0    0    0   27
  27 QH2  PSEUD    5.5812   -2.9108   -4.7993    0    0    0    0    0
  28 C    C_BYL    1.9838    1.4350    0.0000    5   29   30    0    0
  29 O    O_BYL    1.2064    2.3881    0.0000   28    0    0    0    0
  30 N    N_AMI    3.3043    1.5436    0.0000   28    0    0    0    0
```

ARG+

```
RESIDUE   ARG+     7   32    3   31
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   30    0
   3 CHI1     1    3    1.3500    3    5    7   11   29
   4 CHI2     1    3    1.3500    5    7   11   15   29
   5 CHI3     1    3    1.3500    7   11   15   19   29
   6 CHI4     0    0    0.0000   11   15   19   21   29
   7 PSI      0    0    0.0000    3    5   30   32    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   30    0
   6 HA   H_ALI    1.7317   -0.5213    0.9158    5    0    0    0    0
   7 CB   C_ALI    2.0038   -0.7402   -1.2205    5    8    9   11    0
   8 HB2  H_ALI    1.6375   -1.7668   -1.2235    7    0    0    0   10
   9 HB3  H_ALI    1.6375   -0.2685   -2.1323    7    0    0    0   10
  10 QB   PSEUD    1.6375   -1.0177   -1.6779    0    0    0    0    0
  11 CG   C_ALI    3.5338   -0.7388   -1.2182    7   12   13   15    0
  12 HG2  H_ALI    3.9001    0.2878   -1.2152   11    0    0    0   14
  13 HG3  H_ALI    3.9001   -1.2106   -0.3064   11    0    0    0   14
  14 QG   PSEUD    3.9001   -0.4614   -0.7608    0    0    0    0    0
  15 CD   C_ALI    4.0846   -1.4791   -2.4387   11   16   17   19    0
  16 HD2  H_ALI    3.7230   -2.5073   -2.4444   15    0    0    0   18
  17 HD3  H_ALI    3.7230   -1.0090   -3.3532   15    0    0    0   18
  18 QD   PSEUD    3.7230   -1.7581   -2.8988    0    0    0    0    0
  19 NE   N_AMI    5.5643   -1.4643   -2.4144   15   20   21    0    0
```

```
  20 HE   H_AMI    6.0160   -1.0017   -1.6515   19    0    0    0    0
  21 CZ   C_VIN    6.3367   -2.0321   -3.3506   19   22   26    0    0
  22 NH1  N_AMO    7.6712   -1.9691   -3.2468   21   23   24    0    0
  23 HH11 H_AMI    8.2477   -2.3929   -3.9454   22    0    0    0   25
  24 HH12 H_AMI    8.0907   -1.4983   -2.4706   22    0    0    0   25
  25 QH1  PSEUD    8.1692   -1.9456   -3.2080    0    0    0    0    0
  26 NH2  N_AMO    5.7747   -2.6630   -4.3906   21   27   28    0    0
  27 HH21 H_AMI    6.3512   -3.0868   -5.0893   26    0    0    0   29
  28 HH22 H_AMI    4.7788   -2.7101   -4.4681   26    0    0    0   29
  29 QH2  PSEUD    5.5650   -2.8985   -4.7787    0    0    0    0    0
  30 C    C_BYL    1.9838    1.4350    0.0000    5   31   32    0    0
  31 O    O_BYL    1.2064    2.3881    0.0000   30    0    0    0    0
  32 N    N_AMI    3.3043    1.5436    0.0000   30    0    0    0    0
```

ASN

```
RESIDUE   ASN      5   19    3   18
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   17    0
   3 CHI1     1    3    1.3500    3    5    7   11   16
   4 CHI2     0    0    0.0000    5    7   11   12   16
   5 PSI      0    0    0.0000    3    5   17   19    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   17    0
   6 HA   H_ALI    1.7394   -0.5422    0.9011    5    0    0    0    0
   7 CB   C_ALI    2.0038   -0.6938   -1.2474    5    8    9   11    0
   8 HB2  H_ALI    1.6375   -1.7196   -1.2891    7    0    0    0   10
   9 HB3  H_ALI    1.6375   -0.1881   -2.1409    7    0    0    0   10
  10 QB   PSEUD    1.6375   -0.9538   -1.7150    0    0    0    0    0
  11 CG   C_BYL    3.5338   -0.6925   -1.2451    7   12   13    0    0
  12 OD1  O_BYL    4.1821   -1.1949   -2.1483   11    0    0    0    0
  13 ND2  N_AMI    4.0726   -0.1015   -0.1825   11   14   15    0    0
  14 HD21 H_AMI    3.4834    0.2913    0.5236   13    0    0    0   16
  15 HD22 H_AMI    5.0670   -0.0498   -0.0896   13    0    0    0   16
  16 QD2  PSEUD    4.2752    0.1208    0.2170    0    0    0    0    0
  17 C    C_BYL    1.9587    1.4440    0.0000    5   18   19    0    0
  18 O    O_BYL    1.1648    2.3835    0.0000   17    0    0    0    0
  19 N    N_AMI    3.2772    1.5756    0.0000   17    0    0    0    0
```

ASP

```
RESIDUE   ASP      6   17    3   16
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   15    0
   3 CHI1     1    3    1.3500    3    5    7   11   14
   4 CHI2     0    0    0.0000    5    7   11   12   14
   5 CHI32   -1    2    4.0000    7   11   13   14   14
   6 PSI      0    0    0.0000    3    5   15   17    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   15    0
   6 HA   H_ALI    1.7394   -0.5422    0.9011    5    0    0    0    0
   7 CB   C_ALI    2.0038   -0.6938   -1.2474    5    8    9   11    0
   8 HB2  H_ALI    1.6190   -1.7132   -1.2776    7    0    0    0   10
   9 HB3  H_ALI    1.6190   -0.1818   -2.1294    7    0    0    0   10
  10 QB   PSEUD    1.6190   -0.9475   -1.7035    0    0    0    0    0
  11 CG   C_BYL    3.5302   -0.7444   -1.3384    7   12   13    0    0
  12 OD1  O_BYL    4.0951   -1.2810   -2.3031   11    0    0    0    0
  13 OD2  O_HYD    4.1537   -0.1955   -0.3514   11   14    0    0    0
  14 HD2  H_OXY    5.1416   -0.2707   -0.4866   13    0    0    0    0
  15 C    C_BYL    1.9587    1.4440    0.0000    5   16   17    0    0
  16 O    O_BYL    1.1648    2.3835    0.0000   15    0    0    0    0
  17 N    N_AMI    3.2772    1.5756    0.0000   15    0    0    0    0
```

ASP-

```
RESIDUE   ASP-     5   16    3   15
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   14    0
   3 CHI1     1    3    1.3500    3    5    7   11   13
   4 CHI2     0    0    0.0000    5    7   11   12   13
   5 PSI      0    0    0.0000    3    5   14   16    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
```

```
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   14    0
   6 HA   H_ALI    1.7394   -0.5422    0.9011    5    0    0    0    0
   7 CB   C_ALI    2.0038   -0.6938   -1.2474    5    8    9   11    0
   8 HB2  H_ALI    1.6190   -1.7132   -1.2776    7    0    0    0   10
   9 HB3  H_ALI    1.6190   -0.1818   -2.1294    7    0    0    0   10
  10 QB   PSEUD    1.6190   -0.9475   -1.7035    0    0    0    0    0
  11 CG   C_BYL    3.5302   -0.7444   -1.3384    7   12   13    0    0
  12 OD1  O_BYL    4.0206   -1.3033   -2.3432   11    0    0    0    0
  13 OD2  O_BYL    4.1722   -0.2231   -0.4011   11    0    0    0    0
  14 C    C_BYL    1.9587    1.4440    0.0000    5   15   16    0    0
  15 O    O_BYL    1.1648    2.3835    0.0000   14    0    0    0    0
  16 N    N_AMI    3.2772    1.5756    0.0000   14    0    0    0    0
```

CYS

```
RESIDUE   CYS      5   15    3   14
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   13    0
   3 CHI1     1    3    1.3500    3    5    7   11   12
   4 CHI2     1    3    0.7500    5    7   11   12   12
   5 PSI      0    0    0.0000    3    5   13   15    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   13    0
   6 HA   H_ALI    1.7661   -0.5112    0.9103    5    0    0    0    0
   7 CB   C_ALI    2.0187   -0.7882   -1.1831    5    8    9   11    0
   8 HB2  H_ALI    1.6627   -1.8178   -1.1483    7    0    0    0   10
   9 HB3  H_ALI    1.6627   -0.3594   -2.1199    7    0    0    0   10
  10 QB   PSEUD    1.6627   -1.0886   -1.6341    0    0    0    0    0
  11 SG   S_RED    3.8479   -0.7581   -1.1379    7   12    0    0    0
  12 HG   H_SUL    4.0261   -1.4888   -2.2348   11    0    0    0    0
  13 C    C_BYL    1.9334    1.4526    0.0000    5   14   15    0    0
  14 O    O_BYL    1.1232    2.3781    0.0000   13    0    0    0    0
  15 N    N_AMI    3.2494    1.6072    0.0000   13    0    0    0    0
```

CYSS

```
RESIDUE   CYSS     4   14    3   13
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   12    0
   3 CHI1     1    3    1.3500    3    5    7   11   11
   4 PSI      0    0    0.0000    3    5   12   14    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   12    0
   6 HA   H_ALI    1.7661   -0.5112    0.9103    5    0    0    0    0
   7 CB   C_ALI    2.0187   -0.7882   -1.1831    5    8    9   11    0
   8 HB2  H_ALI    1.6404   -1.8093   -1.1355    7    0    0    0   10
   9 HB3  H_ALI    1.6404   -0.3509   -2.1071    7    0    0    0   10
  10 QB   PSEUD    1.6404   -1.0801   -1.6213    0    0    0    0    0
  11 SG   S_OXY    3.8460   -0.8430   -1.2654    7    0    0    0    0
  12 C    C_BYL    1.9334    1.4526    0.0000    5   13   14    0    0
  13 O    O_BYL    1.1232    2.3781    0.0000   12    0    0    0    0
  14 N    N_AMI    3.2494    1.6072    0.0000   12    0    0    0    0
```

GLN

```
RESIDUE   GLN      6   23    3   22
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   21    0
   3 CHI1     1    3    1.3500    3    5    7   11   20
   4 CHI2     1    3    1.3500    5    7   11   15   20
   5 CHI3     0    0    0.0000    7   11   15   16   20
   6 PSI      0    0    0.0000    3    5   21   23    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   21    0
   6 HA   H_ALI    1.7427   -0.5017    0.9233    5    0    0    0    0
   7 CB   C_ALI    2.0013   -0.7874   -1.1917    5    8    9   11    0
   8 HB2  H_ALI    1.6341   -1.8130   -1.1545    7    0    0    0   10
   9 HB3  H_ALI    1.6341   -0.3510   -2.1206    7    0    0    0   10
```

```
  10 QB   PSEUD    1.6341   -1.0820   -1.6375    0    0    0    0    0
  11 CG   C_ALI    3.5313   -0.7874   -1.1917    7   12   13   15    0
  12 HG2  H_ALI    3.8985    0.2382   -1.2290   11    0    0    0   14
  13 HG3  H_ALI    3.8985   -1.2239   -0.2629   11    0    0    0   14
  14 QG   PSEUD    3.8985   -0.4928   -0.7459    0    0    0    0    0
  15 CD   C_BYL    4.0796   -1.5749   -2.3835   11   16   17    0    0
  16 OE1  O_BYL    5.2772   -1.7032   -2.5777   15    0    0    0    0
  17 NE2  N_AMI    3.1391   -2.0933   -3.1681   15   18   19    0    0
  18 HE21 H_AMI    2.1732   -1.9506   -2.9521   17    0    0    0   20
  19 HE22 H_AMI    3.3980   -2.6258   -3.9740   17    0    0    0   20
  20 QE2  PSEUD    2.7856   -2.2882   -3.4631    0    0    0    0    0
  21 C    C_BYL    1.9838    1.4350    0.0000    5   22   23    0    0
  22 O    O_BYL    1.2064    2.3882    0.0000   21    0    0    0    0
  23 N    N_AMI    3.3043    1.5436    0.0000   21    0    0    0    0
```

GLU

```
RESIDUE   GLU      7   21    3   20
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   19    0
   3 CHI1     1    3    1.3500    3    5    7   11   18
   4 CHI2     1    3    1.3500    5    7   11   15   18
   5 CHI3     0    0    0.0000    7   11   15   16   18
   6 CHI42   -1    2    4.0000   11   15   17   18   18
   7 PSI      0    0    0.0000    3    5   19   21    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   19    0
   6 HA   H_ALI    1.7427   -0.5017    0.9233    5    0    0    0    0
   7 CB   C_ALI    2.0013   -0.7874   -1.1917    5    8    9   11    0
   8 HB2  H_ALI    1.6341   -1.8130   -1.1545    7    0    0    0   10
   9 HB3  H_ALI    1.6341   -0.3510   -2.1206    7    0    0    0   10
  10 QB   PSEUD    1.6341   -1.0820   -1.6375    0    0    0    0    0
  11 CG   C_ALI    3.5313   -0.7874   -1.1917    7   12   13   15    0
  12 HG2  H_ALI    3.8985    0.2382   -1.2290   11    0    0    0   14
  13 HG3  H_ALI    3.8985   -1.2239   -0.2629   11    0    0    0   14
  14 QG   PSEUD    3.8985   -0.4928   -0.7459    0    0    0    0    0
  15 CD   C_BYL    4.0796   -1.5749   -2.3835   11   16   17    0    0
  16 OE1  O_BYL    5.3043   -1.6818   -2.5453   15    0    0    0    0
  17 OE2  O_HYD    3.1835   -2.0873   -3.1571   15   18    0    0    0
  18 HE2  H_OXY    3.6219   -2.5827   -3.9070   17    0    0    0    0
  19 C    C_BYL    1.9838    1.4350    0.0000    5   20   21    0    0
  20 O    O_BYL    1.2064    2.3882    0.0000   19    0    0    0    0
  21 N    N_AMI    3.3043    1.5436    0.0000   19    0    0    0    0
```

GLU-

```
RESIDUE   GLU-     6   20    3   19
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   18    0
   3 CHI1     1    3    1.3500    3    5    7   11   17
   4 CHI2     1    3    1.3500    5    7   11   15   17
   5 CHI3     0    0    0.0000    7   11   15   16   17
   6 PSI      0    0    0.0000    3    5   18   20    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   18    0
   6 HA   H_ALI    1.7427   -0.5017    0.9233    5    0    0    0    0
   7 CB   C_ALI    2.0013   -0.7874   -1.1917    5    8    9   11    0
   8 HB2  H_ALI    1.6341   -1.8130   -1.1545    7    0    0    0   10
   9 HB3  H_ALI    1.6341   -0.3510   -2.1206    7    0    0    0   10
  10 QB   PSEUD    1.6341   -1.0820   -1.6375    0    0    0    0    0
  11 CG   C_ALI    3.5313   -0.7874   -1.1917    7   12   13   15    0
  12 HG2  H_ALI    3.8985    0.2382   -1.2290   11    0    0    0   14
  13 HG3  H_ALI    3.8985   -1.2239   -0.2629   11    0    0    0   14
  14 QG   PSEUD    3.8985   -0.4928   -0.7459    0    0    0    0    0
  15 CD   C_BYL    4.0796   -1.5749   -2.3835   11   16   17    0    0
  16 OE1  O_BYL    5.3227   -1.6469   -2.4925   15    0    0    0    0
  17 OE2  O_BYL    3.2432   -2.0870   -3.1585   15    0    0    0    0
  18 C    C_BYL    1.9838    1.4350    0.0000    5   19   20    0    0
  19 O    O_BYL    1.2064    2.3882    0.0000   18    0    0    0    0
  20 N    N_AMI    3.3043    1.5436    0.0000   18    0    0    0    0
```

GLY

```
RESIDUE   GLY      3   11    3   10
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5    9    0
   3 PSI      0    0    0.0000    3    5    9   11    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7    9    0
   6 HA1  H_ALI    1.8202   -0.5343    0.8762    5    0    0    0    8
   7 HA2  H_ALI    1.8202   -0.5343   -0.8762    5    0    0    0    8
   8 QA   PSEUD    1.8202   -0.5343    0.0000    0    0    0    0    0
   9 C    C_BYL    2.0013    1.4284    0.0000    5   10   11    0    0
  10 O    O_BYL    1.2356    2.3910    0.0000    9    0    0    0    0
  11 N    N_AMI    3.3231    1.5208    0.0000    9    0    0    0    0
```

HIS

```
RESIDUE   HIS      5   21    3   20
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   19    0
   3 CHI1     1    3    1.3500    3    5    7   11   18
   4 CHI2     0    0    0.0000    5    7   11   12   18
   5 PSI      0    0    0.0000    3    5   19   21    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   19    0
   6 HA   H_ALI    1.7658   -0.4880    0.9231    5    0    0    0    0
   7 CB   C_ALI    1.9963   -0.8228   -1.1699    5    8    9   11    0
   8 HB2  H_ALI    1.6134   -1.8405   -1.0938    7    0    0    0   10
   9 HB3  H_ALI    1.6134   -0.4071   -2.1019    7    0    0    0   10
  10 QB   PSEUD    1.6134   -1.1238   -1.5979    0    0    0    0    0
  11 CG   C_VIN    3.5040   -0.8713   -1.2388    7   12   13    0    0
  12 ND1  N_AMI    4.1876   -1.5636   -2.2231   11   14   15    0    0
  13 CD2  C_ARO    4.4515   -0.3061   -0.4368   11   16   17    0    0
  14 HD1  H_AMI    3.7713   -2.0867   -2.9668   12    0    0    0    0
  15 CE1  C_ARO    5.4873   -1.4156   -2.0126   12   16   18    0    0
  16 NE2  N_AMI    5.6488   -0.6364   -0.9048   13   15    0    0    0
  17 HD2  H_ARO    4.2578    0.3120    0.4399   13    0    0    0    0
  18 HE1  H_ARO    6.2871   -1.8416   -2.6184   15    0    0    0    0
  19 C    C_BYL    1.9662    1.4413    0.0000    5   20   21    0    0
  20 O    O_BYL    1.1773    2.3850    0.0000   19    0    0    0    0
  21 N    N_AMI    3.2853    1.5659    0.0000   19    0    0    0    0
```

HIST

```
RESIDUE   HIST     5   21    3   20
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   19    0
   3 CHI1     1    3    1.3500    3    5    7   11   18
   4 CHI2     0    0    0.0000    5    7   11   12   18
   5 PSI      0    0    0.0000    3    5   19   21    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   19    0
   6 HA   H_ALI    1.7658   -0.4880    0.9231    5    0    0    0    0
   7 CB   C_ALI    1.9963   -0.8228   -1.1699    5    8    9   11    0
   8 HB2  H_ALI    1.6134   -1.8405   -1.0938    7    0    0    0   10
   9 HB3  H_ALI    1.6134   -0.4071   -2.1019    7    0    0    0   10
  10 QB   PSEUD    1.6134   -1.1238   -1.5979    0    0    0    0    0
  11 CG   C_VIN    3.5040   -0.8713   -1.2388    7   12   13    0    0
  12 ND1  N_AMI    4.1876   -1.5636   -2.2231   11   16    0    0    0
  13 CD2  C_ARO    4.4515   -0.3061   -0.4368   11   14   15    0    0
  14 HD2  H_ARO    4.2578    0.3120    0.4399   13    0    0    0    0
  15 NE2  N_AMI    5.6488   -0.6364   -0.9048   13   16   18    0    0
  16 CE1  C_ARO    5.4873   -1.4156   -2.0126   12   15   17    0    0
  17 HE1  H_ARO    6.2871   -1.8416   -2.6184   16    0    0    0    0
  18 HE2  H_AMI    6.5236   -0.3132   -0.5438   15    0    0    0    0
  19 C    C_BYL    1.9662    1.4413    0.0000    5   20   21    0    0
  20 O    O_BYL    1.1773    2.3850    0.0000   19    0    0    0    0
  21 N    N_AMI    3.2853    1.5659    0.0000   19    0    0    0    0
```

HIST+

```
RESIDUE   HIS+     5   22    3   21
```

```
  1 OMEGA   -1    2   10.0000    2    1    3    4    0
  2 PHI      0    0    0.0000    1    3    5   20    0
  3 CHI1     1    3    1.3500    3    5    7   11   19
  4 CHI2     0    0    0.0000    5    7   11   12   19
  5 PSI      0    0    0.0000    3    5   20   22    0
  1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
  2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
  3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
  4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
  5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   20    0
  6 HA   H_ALI    1.7658   -0.4880    0.9231    5    0    0    0    0
  7 CB   C_ALI    1.9963   -0.8228   -1.1699    5    8    9   11    0
  8 HB2  H_ALI    1.6134   -1.8405   -1.0938    7    0    0    0   10
  9 HB3  H_ALI    1.6134   -0.4071   -2.1019    7    0    0    0   10
 10 QB   PSEUD    1.6134   -1.1238   -1.5979    0    0    0    0    0
 11 CG   C_VIN    3.5040   -0.8713   -1.2388    7   12   13    0    0
 12 ND1  N_AMO    4.1876   -1.5636   -2.2231   11   14   15    0    0
 13 CD2  C_ARO    4.4515   -0.3061   -0.4368   11   16   17    0    0
 14 HD1  H_AMI    3.7713   -2.0867   -2.9668   12    0    0    0    0
 15 CE1  C_ARO    5.4873   -1.4156   -2.0126   12   16   18    0    0
 16 NE2  N_AMO    5.6488   -0.6364   -0.9048   13   15   19    0    0
 17 HD2  H_ARO    4.2578    0.3120    0.4399   13    0    0    0    0
 18 HE1  H_ARO    6.2871   -1.8416   -2.6184   15    0    0    0    0
 19 HE2  H_AMI    6.5236   -0.3132   -0.5438   16    0    0    0    0
 20 C    C_BYL    1.9662    1.4413    0.0000    5   21   22    0    0
 21 O    O_BYL    1.1773    2.3850    0.0000   20    0    0    0    0
 22 N    N_AMI    3.2853    1.5659    0.0000   20    0    0    0    0
```

ILE

```
RESIDUE   ILE      7   25    3   24
  1 OMEGA   -1    2   10.0000    2    1    3    4    0
  2 PHI      0    0    0.0000    1    3    5   23    0
  3 CHI1     1    3    1.3500    3    5    7   14   22
  4 CHI22    1    3    1.3500    5    7   10   11   13
  5 CHI21    1    3    1.3500    5    7   14   19   22
  6 CHI31    1    3    1.3500    7   14   19   20   22
  7 PSI      0    0    0.0000    3    5   23   25    0
  1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
  2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
  3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
  4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
  5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   23    0
  6 HA   H_ALI    1.7797   -0.4805    0.9222    5    0    0    0    0
  7 CB   C_ALI    1.9888   -0.8392   -1.1617    5    8   10   14    0
  8 HB   H_ALI    1.6625   -1.8692   -1.0179    7    0    0    0    0
  9 QG2  PSEUD    1.2708   -0.2506   -2.8117    0    0    0    0    0
 10 CG2  C_ALI    1.4086   -0.3635   -2.4951    7   11   12   13    0
 11 HG21 H_ALI    1.8059   -0.9770   -3.3037   10    0    0    0    9
 12 HG22 H_ALI    0.3225   -0.4528   -2.4713   10    0    0    0    9
 13 HG23 H_ALI    1.6840    0.6781   -2.6602   10    0    0    0    9
 14 CG1  C_ALI    3.5188   -0.8471   -1.1725    7   15   16   19    0
 15 HG12 H_ALI    3.8906    0.1742   -1.2551   14    0    0    0   17
 16 HG13 H_ALI    3.8906   -1.2463   -0.2289   14    0    0    0   17
 17 QG1  PSEUD    3.8906   -0.5361   -0.7420    0    0    0    0    0
 18 QD1  PSEUD    4.1818   -1.8856   -2.6101    0    0    0    0    0
 19 CD1  C_ALI    4.0546   -1.6863   -2.3342   14   20   21   22    0
 20 HD11 H_ALI    5.1444   -1.6749   -2.3183   19    0    0    0   18
 21 HD12 H_ALI    3.7005   -2.7124   -2.2348   19    0    0    0   18
 22 HD13 H_ALI    3.7005   -1.2695   -3.2771   19    0    0    0   18
 23 C    C_BYL    1.9587    1.4440    0.0000    5   24   25    0    0
 24 O    O_BYL    1.1648    2.3835    0.0000   23    0    0    0    0
 25 N    N_AMI    3.2772    1.5756    0.0000   23    0    0    0    0
```

LEU

```
RESIDUE   LEU      7   26    3   25
  1 OMEGA   -1    2   10.0000    2    1    3    4    0
  2 PHI      0    0    0.0000    1    3    5   24    0
  3 CHI1     1    3    1.3500    3    5    7   11   23
  4 CHI2     1    3    1.3500    5    7   11   15   23
  5 CHI31    1    3    1.3500    7   11   15   16   18
  6 CHI32    1    3    1.3500    7   11   19   20   22
  7 PSI      0    0    0.0000    3    5   24   26    0
  1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
  2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
  3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
  4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
  5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   24    0
```

```
  6 HA   H_ALI    1.7797   -0.4805    0.9222    5    0    0    0    0
  7 CB   C_ALI    1.9888   -0.8392   -1.1617    5    8    9   11    0
  8 HB2  H_ALI    1.5941   -1.8507   -1.0656    7    0    0    0   10
  9 HB3  H_ALI    1.5941   -0.4301   -2.0917    7    0    0    0   10
 10 QB   PSEUD    1.5941   -1.1404   -1.5787    0    0    0    0    0
 11 CG   C_ALI    3.5118   -0.9251   -1.2806    7   12   15   19    0
 12 HG   H_ALI    3.8975   -1.4120   -0.3849   11    0    0    0    0
 13 QD1  PSEUD    4.0177   -1.9934   -2.7595    0    0    0    0   23
 14 QD2  PSEUD    4.2850    0.8015   -1.3553    0    0    0    0   23
 15 CD1  C_ALI    3.9206   -1.7884   -2.4757   11   16   17   18    0
 16 HD11 H_ALI    5.0080   -1.8326   -2.5369   15    0    0    0   13
 17 HD12 H_ALI    3.5225   -2.7953   -2.3497   15    0    0    0   13
 18 HD13 H_ALI    3.5225   -1.3524   -3.3920   15    0    0    0   13
 19 CD2  C_ALI    4.1366    0.4702   -1.3410   11   20   21   22    0
 20 HD21 H_ALI    5.2196    0.3800   -1.4253   19    0    0    0   14
 21 HD22 H_ALI    3.7480    1.0051   -2.2076   19    0    0    0   14
 22 HD23 H_ALI    3.8873    1.0195   -0.4331   19    0    0    0   14
 23 QQD  PSEUD    4.1513   -0.5960   -2.0574    0    0    0    0    0
 24 C    C_BYL    1.9587    1.4440    0.0000    5   25   26    0    0
 25 O    O_BYL    1.1648    2.3835    0.0000   24    0    0    0    0
 26 N    N_AMI    3.2772    1.5756    0.0000   24    0    0    0    0
```

LYS

```
RESIDUE   LYS      8   29    3   28
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   27    0
   3 CHI1     1    3    1.3500    3    5    7   11   26
   4 CHI2     1    3    1.3500    5    7   11   15   26
   5 CHI3     1    3    1.3500    7   11   15   19   26
   6 CHI4     1    3    1.3500   11   15   19   23   26
   7 CHI5     1    3    0.9000   15   19   23   24   26
   8 PSI      0    0    0.0000    3    5   27   29    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   27    0
   6 HA   H_ALI    1.7797   -0.4805    0.9222    5    0    0    0    0
   7 CB   C_ALI    1.9888   -0.8392   -1.1617    5    8    9   11    0
   8 HB2  H_ALI    1.6169   -1.8605   -1.0798    7    0    0    0   10
   9 HB3  H_ALI    1.6169   -0.4400   -2.1053    7    0    0    0   10
  10 QB   PSEUD    1.6169   -1.1502   -1.5926    0    0    0    0    0
  11 CG   C_ALI    3.5188   -0.8471   -1.1725    7   12   13   15    0
  12 HG2  H_ALI    3.8906    0.1742   -1.2551   11    0    0    0   14
  13 HG3  H_ALI    3.8906   -1.2463   -0.2289   11    0    0    0   14
  14 QG   PSEUD    3.8906   -0.5361   -0.7420    0    0    0    0    0
  15 CD   C_ALI    4.0546   -1.6863   -2.3342   11   16   17   19    0
  16 HD2  H_ALI    3.6827   -2.7076   -2.2516   15    0    0    0   18
  17 HD3  H_ALI    3.6827   -1.2871   -3.2778   15    0    0    0   18
  18 QD   PSEUD    3.6827   -1.9974   -2.7647    0    0    0    0    0
  19 CE   C_ALI    5.5845   -1.6941   -2.3450   15   20   21   23    0
  20 HE2  H_ALI    5.9587   -0.6738   -2.4289   19    0    0    0   22
  21 HE3  H_ALI    5.9587   -2.0943   -1.4027   19    0    0    0   22
  22 QE   PSEUD    5.9587   -1.3841   -1.9158    0    0    0    0    0
  23 NZ   N_AMI    6.0907   -2.5086   -3.4723   19   24   25    0    0
  24 HZ1  H_AMI    7.1041   -2.5293   -3.5010   23    0    0    0   26
  25 HZ2  H_AMI    5.7789   -3.4718   -3.4151   23    0    0    0   26
  26 QZ   PSEUD    6.4415   -3.0006   -3.4581    0    0    0    0    0
  27 C    C_BYL    1.9587    1.4440    0.0000    5   28   29    0    0
  28 O    O_BYL    1.1648    2.3835    0.0000   27    0    0    0    0
  29 N    N_AMI    3.2772    1.5756    0.0000   27    0    0    0    0
```

LYS+

```
RESIDUE   LYS+     8   30    3   29
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   28    0
   3 CHI1     1    3    1.3500    3    5    7   11   27
   4 CHI2     1    3    1.3500    5    7   11   15   27
   5 CHI3     1    3    1.3500    7   11   15   19   27
   6 CHI4     1    3    1.3500   11   15   19   23   27
   7 CHI5     1    3    0.9000   15   19   23   24   27
   8 PSI      0    0    0.0000    3    5   28   30    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   28    0
```

```
   6 HA   H_ALI    1.7797   -0.4805    0.9222    5    0    0    0    0
   7 CB   C_ALI    1.9888   -0.8392   -1.1617    5    8    9   11    0
   8 HB2  H_ALI    1.6169   -1.8605   -1.0798    7    0    0    0   10
   9 HB3  H_ALI    1.6169   -0.4400   -2.1053    7    0    0    0   10
  10 QB   PSEUD    1.6169   -1.1502   -1.5926    0    0    0    0    0
  11 CG   C_ALI    3.5188   -0.8471   -1.1725    7   12   13   15    0
  12 HG2  H_ALI    3.8906    0.1742   -1.2551   11    0    0    0   14
  13 HG3  H_ALI    3.8906   -1.2463   -0.2289   11    0    0    0   14
  14 QG   PSEUD    3.8906   -0.5361   -0.7420    0    0    0    0    0
  15 CD   C_ALI    4.0546   -1.6863   -2.3342   11   16   17   19    0
  16 HD2  H_ALI    3.6827   -2.7076   -2.2516   15    0    0    0   18
  17 HD3  H_ALI    3.6827   -1.2871   -3.2778   15    0    0    0   18
  18 QD   PSEUD    3.6827   -1.9974   -2.7647    0    0    0    0    0
  19 CE   C_ALI    5.5845   -1.6941   -2.3450   15   20   21   23    0
  20 HE2  H_ALI    5.9587   -0.6738   -2.4289   19    0    0    0   22
  21 HE3  H_ALI    5.9587   -2.0943   -1.4027   19    0    0    0   22
  22 QE   PSEUD    5.9587   -1.3841   -1.9158    0    0    0    0    0
  23 NZ   N_AMO    6.0907   -2.5086   -3.4723   19   24   25   26    0
  24 HZ1  H_AMI    7.0907   -2.5032   -3.4648   23    0    0    0   27
  25 HZ2  H_AMI    5.7617   -3.4483   -3.3787   23    0    0    0   27
  26 HZ3  H_AMI    5.7617   -2.1246   -4.3351   23    0    0    0   27
  27 QZ   PSEUD    6.2047   -2.6920   -3.7262    0    0    0    0    0
  28 C    C_BYL    1.9587    1.4440    0.0000    5   29   30    0    0
  29 O    O_BYL    1.1648    2.3835    0.0000   28    0    0    0    0
  30 N    N_AMI    3.2772    1.5756    0.0000   28    0    0    0    0
```

MET

```
RESIDUE   MET      7   23    3   22
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   21    0
   3 CHI1     1    3    1.3500    3    5    7   11   20
   4 CHI2     1    3    1.3500    5    7   11   15   20
   5 CHI3     1    3    1.0000    7   11   15   17   20
   6 CHI4     1    3    1.0000   11   15   17   18   20
   7 PSI      0    0    0.0000    3    5   21   23    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   21    0
   6 HA   H_ALI    1.7457   -0.5091    0.9183    5    0    0    0    0
   7 CB   C_ALI    1.9637   -0.7431   -1.2361    5    8    9   11    0
   8 HB2  H_ALI    1.5826   -1.7643   -1.2341    7    0    0    0   10
   9 HB3  H_ALI    1.5826   -0.2624   -2.1371    7    0    0    0   10
  10 QB   PSEUD    1.5826   -1.0134   -1.6856    0    0    0    0    0
  11 CG   C_ALI    3.4932   -0.7637   -1.2704    7   12   13   15    0
  12 HG2  H_ALI    3.8789    0.2557   -1.2752   11    0    0    0   14
  13 HG3  H_ALI    3.8789   -1.2461   -0.3722   11    0    0    0   14
  14 QG   PSEUD    3.8789   -0.4952   -0.8237    0    0    0    0    0
  15 SD   S_RED    4.0596   -1.6359   -2.7211   11   17    0    0    0
  16 QE   PSEUD    6.1844   -1.4722   -2.4490    0    0    0    0    0
  17 CE   C_ALI    5.8250   -1.4999   -2.4950   15   18   19   20    0
  18 HE1  H_ALI    6.3363   -1.9959   -3.3200   17    0    0    0   16
  19 HE2  H_ALI    6.1084   -0.4476   -2.4720   17    0    0    0   16
  20 HE3  H_ALI    6.1084   -1.9731   -1.5549   17    0    0    0   16
  21 C    C_BYL    2.0013    1.4284    0.0000    5   22   23    0    0
  22 O    O_BYL    1.2356    2.3910    0.0000   21    0    0    0    0
  23 N    N_AMI    3.3231    1.5208    0.0000   21    0    0    0    0
```

PHE

```
RESIDUE   PHE      5   27    3   26
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   25    0
   3 CHI1     1    3    1.3500    3    5    7   14   24
   4 CHI2     0    0    0.0000    5    7   14   15   24
   5 PSI      0    0    0.0000    3    5   25   27    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   25    0
   6 HA   H_ALI    1.7765   -0.5114    0.9066    5    0    0    0    0
   7 CB   C_ALI    1.9003   -0.7053   -1.2819    5    8    9   14    0
   8 HB2  H_ALI    1.4836   -1.7124   -1.2948    7    0    0    0   10
   9 HB3  H_ALI    1.4836   -0.1770   -2.1395    7    0    0    0   10
  10 QB   PSEUD    1.4836   -0.9447   -1.7171    0    0    0    0    0
  11 QD   PSEUD    3.5529   -0.8031   -1.4598    0    0    0    0   13
```

```
  12 QE   PSEUD    6.0442   -0.9505   -1.7278    0    0    0    0   13
  13 QR   PSEUD    4.7986   -0.8768   -1.5938    0    0    0    0    0
  14 CG   C_VIN    3.4189   -0.7951   -1.4453    7   15   23    0    0
  15 CD1  C_ARO    3.9503   -1.3986   -2.5422   14   16   17    0    0
  16 HD1  H_ARO    3.2943   -1.8182   -3.3049   15    0    0    0   11
  17 CE1  C_ARO    5.3597   -1.4820   -2.6938   15   18   19    0    0
  18 HE1  H_ARO    5.7856   -1.9656   -3.5729   17    0    0    0   12
  19 CZ   C_ARO    6.1782   -0.9585   -1.7422   17   20   21    0    0
  20 HZ   H_ARO    7.2601   -1.0225   -1.8585   19    0    0    0   13
  21 CE2  C_ARO    5.6468   -0.3550   -0.6453   19   22   23    0    0
  22 HE2  H_ARO    6.3028    0.0646    0.1174   21    0    0    0   12
  23 CD2  C_ARO    4.2374   -0.2716   -0.4937   14   21   24    0    0
  24 HD2  H_ARO    3.8115    0.2120    0.3854   23    0    0    0   11
  25 C    C_BYL    2.0013    1.4284    0.0000    5   26   27    0    0
  26 O    O_BYL    1.2356    2.3910    0.0000   25    0    0    0    0
  27 N    N_AMI    3.3231    1.5208    0.0000   25    0    0    0    0
```

PRO

```
RESIDUE   PRO      2   20    3   19
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PSI      0    0    0.0000    3    5   18   20    0
   1 C    C_BYL   -0.7005    0.3017    1.1260    2    3    0    0    0
   2 O    O_BYL   -0.1904    0.5914    2.2071    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 CD   C_ALI   -0.5705   -0.3478   -1.2981    3   11   15   16    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   18    0
   6 HA   H_ALI    1.8001   -0.4260    0.8355    5    0    0    0    0
   7 CB   C_ALI    1.8464   -0.7872   -1.2393    5    8    9   11    0
   8 HB2  H_ALI    2.6920   -0.3217   -1.7456    7    0    0    0   10
   9 HB3  H_ALI    2.1502   -1.8009   -0.9780    7    0    0    0   10
  10 QB   PSEUD    2.4211   -1.0613   -1.3618    0    0    0    0    0
  11 CG   C_ALI    0.6166   -0.8003   -2.1326    4    7   12   13    0
  12 HG2  H_ALI    0.7579   -0.1376   -2.9864   11    0    0    0   14
  13 HG3  H_ALI    0.4445   -1.8004   -2.5305   11    0    0    0   14
  14 QG   PSEUD    0.6012   -0.9690   -2.7585    0    0    0    0    0
  15 HD2  H_ALI   -1.0711    0.5072   -1.7525    4    0    0    0   17
  16 HD3  H_ALI   -1.3137   -1.1396   -1.2047    4    0    0    0   17
  17 QD   PSEUD   -1.1924   -0.3162   -1.4786    0    0    0    0    0
  18 C    C_BYL    2.0013    1.4284    0.0000    5   19   20    0    0
  19 O    O_BYL    1.2356    2.3910    0.0000   18    0    0    0    0
  20 N    N_AMI    3.3231    1.5208    0.0000   18    0    0    0    0
```

SER

```
RESIDUE   SER      5   15    3   14
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   13    0
   3 CHI1     1    3    1.3500    3    5    7   11   12
   4 CHI2     1    3    0.3000    5    7   11   12   12
   5 PSI      0    0    0.0000    3    5   13   15    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   13    0
   6 HA   H_ALI    1.7416   -0.5122    0.9178    5    0    0    0    0
   7 CB   C_ALI    2.0038   -0.7653   -1.2049    5    8    9   11    0
   8 HB2  H_ALI    1.6328   -1.7901   -1.1839    7    0    0    0   10
   9 HB3  H_ALI    1.6328   -0.3109   -2.1235    7    0    0    0   10
  10 QB   PSEUD    1.6328   -1.0505   -1.6537    0    0    0    0    0
  11 OG   O_HYD    3.4286   -0.7774   -1.2238    7   12    0    0    0
  12 HG   H_OXY    3.7558   -1.2840   -2.0214   11    0    0    0    0
  13 C    C_BYL    1.9763    1.4377    0.0000    5   14   15    0    0
  14 O    O_BYL    1.1939    2.3868    0.0000   13    0    0    0    0
  15 N    N_AMI    3.2963    1.5532    0.0000   13    0    0    0    0
```

THR

```
RESIDUE   THR      6   18    3   17
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   16    0
   3 CHI1     1    3    1.3500    3    5    7   10   15
   4 CHI21    1    3    0.3000    5    7   10   11   11
   5 CHI22    1    3    1.3500    5    7   12   13   15
   6 PSI      0    0    0.0000    3    5   16   18    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
```

```
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   16    0
   6 HA   H_ALI    1.7966   -0.4689    0.9220    5    0    0    0    0
   7 CB   C_ALI    1.9258   -0.8241   -1.1992    5    8   10   12    0
   8 HB   H_ALI    1.5187   -1.8345   -1.1619    7    0    0    0    0
   9 QG2  PSEUD    1.5244    0.0064   -2.8527    0    0    0    0    0
  10 OG1  O_HYD    3.3475   -0.7692   -1.1194    7   11    0    0    0
  11 HG1  H_OXY    3.7526   -1.2870   -1.8728   10    0    0    0    0
  12 CG2  C_ALI    1.6014   -0.1530   -2.5354    7   13   14   15    0
  13 HG21 H_ALI    1.9584   -0.7795   -3.3528   12    0    0    0    9
  14 HG22 H_ALI    0.5230   -0.0208   -2.6232   12    0    0    0    9
  15 HG23 H_ALI    2.0917    0.8194   -2.5821   12    0    0    0    9
  16 C    C_BYL    1.9863    1.4340    0.0000    5   17   18    0    0
  17 O    O_BYL    1.2106    2.3886    0.0000   16    0    0    0    0
  18 N    N_AMI    3.3070    1.5402    0.0000   16    0    0    0    0
```

TRP

```
RESIDUE   TRP      5   28    3   27
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   26    0
   3 CHI1     1    3    1.3500    3    5    7   11   25
   4 CHI2     0    0    0.0000    5    7   11   12   25
   5 PSI      0    0    0.0000    3    5   26   28    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   26    0
   6 HA   H_ALI    1.7862   -0.5002    0.9093    5    0    0    0    0
   7 CB   C_ALI    2.0013   -0.8067   -1.1787    5    8    9   11    0
   8 HB2  H_ALI    1.6202   -1.8261   -1.1167    7    0    0    0   10
   9 HB3  H_ALI    1.6202   -0.3799   -2.1065    7    0    0    0   10
  10 QB   PSEUD    1.6202   -1.1030   -1.6116    0    0    0    0    0
  11 CG   C_VIN    3.5292   -0.8520   -1.2448    7   12   13    0    0
  12 CD1  C_ARO    4.2978   -1.4719   -2.1507   11   16   17    0    0
  13 CD2  C_VIN    4.4485   -0.2236   -0.3267   11   14   15    0    0
  14 CE3  C_ARO    4.2018    0.5582    0.8155   13   18   19    0    0
  15 CE2  C_VIN    5.7342   -0.5067   -0.7404   13   16   20    0    0
  16 NE1  N_AMI    5.6397   -1.2900   -1.8848   12   15   21    0    0
  17 HD1  H_ARO    3.9116   -2.0476   -2.9918   12    0    0    0    0
  18 HE3  H_ARO    3.1962    0.7957    1.1626   14    0    0    0    0
  19 CZ3  C_ARO    5.3514    1.0094    1.4749   14   22   23    0    0
  20 CZ2  C_ARO    6.8740   -0.0475   -0.0695   15   23   24    0    0
  21 HE1  H_AMI    6.4722   -1.6873   -2.4654   16    0    0    0    0
  22 HZ3  H_ARO    5.2167    1.6203    2.3675   19    0    0    0    0
  23 CH2  C_ARO    6.6525    0.7332    1.0713   19   20   25    0    0
  24 HZ2  H_ARO    7.8795   -0.2851   -0.4166   20    0    0    0    0
  25 HH2  H_ARO    7.4966    1.1227    1.6404   23    0    0    0    0
  26 C    C_BYL    1.9258    1.4551    0.0000    5   27   28    0    0
  27 O    O_BYL    1.1108    2.3763    0.0000   26    0    0    0    0
  28 N    N_AMI    3.2409    1.6166    0.0000   26    0    0    0    0
```

TYR

```
RESIDUE   TYR      6   28    3   27
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   26    0
   3 CHI1     1    3    1.3500    3    5    7   14   25
   4 CHI2     0    0    0.0000    5    7   14   15   25
   5 CHI6    -1    2    1.7500   17   19   24   25   25
   6 PSI      0    0    0.0000    3    5   26   28    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   26    0
   6 HA   H_ALI    1.7894   -0.5104    0.9024    5    0    0    0    0
   7 CB   C_ALI    1.8747   -0.6954   -1.2959    5    8    9   14    0
   8 HB2  H_ALI    1.4494   -1.6989   -1.3128    7    0    0    0   10
   9 HB3  H_ALI    1.4494   -0.1548   -2.1414    7    0    0    0   10
  10 QB   PSEUD    1.4494   -0.9269   -1.7271    0    0    0    0    0
  11 QD   PSEUD    3.5334   -0.8057   -1.5014    0    0    0    0   13
  12 QE   PSEUD    6.0510   -0.9730   -1.8132    0    0    0    0   13
  13 QR   PSEUD    4.7922   -0.8893   -1.6572    0    0    0    0    0
  14 CG   C_VIN    3.3898   -0.7961   -1.4835    7   15   22    0    0
  15 CD1  C_ARO    3.9068   -1.4012   -2.6111   14   16   17    0    0
  16 HD1  H_ARO    3.2356   -1.8074   -3.3679   15    0    0    0   11
  17 CE1  C_ARO    5.3328   -1.4960   -2.7877   15   18   19    0    0
```

```
  18 HE1  H_ARO    5.7543   -1.9713   -3.6734   17    0    0    0   12
  19 CZ   C_VIN    6.1417   -0.9790   -1.8244   17   20   24    0    0
  20 CE2  C_ARO    5.6654   -0.3767   -0.7019   19   21   22    0    0
  21 HE2  H_ARO    6.3477    0.0253    0.0471   20    0    0    0   12
  22 CD2  C_ARO    4.2394   -0.2819   -0.5253   14   20   23    0    0
  23 HD2  H_ARO    3.8311    0.1960    0.3652   22    0    0    0   11
  24 OH   O_HYD    7.4885   -1.0685   -1.9912   19   25    0    0    0
  25 HH   H_OXY    7.9579   -0.6510   -1.2132   24    0    0    0    0
  26 C    C_BYL    2.0013    1.4284    0.0000    5   27   28    0    0
  27 O    O_BYL    1.2356    2.3910    0.0000   26    0    0    0    0
  28 N    N_AMI    3.3231    1.5208    0.0000   26    0    0    0    0
```

VAL

```
RESIDUE   VAL      6   22    3   21
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5   20    0
   3 CHI1     1    3    1.3500    3    5    7   11   19
   4 CHI21    1    3    1.3500    5    7   11   12   14
   5 CHI22    1    3    1.3500    5    7   15   16   18
   6 PSI      0    0    0.0000    3    5   20   22    0
   1 C    C_BYL   -0.6824   -1.1357    0.0000    2    3    0    0    0
   2 O    O_BYL   -0.1723   -2.2550    0.0000    1    0    0    0    0
   3 N    N_AMI    0.0000    0.0000    0.0000    1    4    5    0    0
   4 HN   H_AMI   -0.4226    0.9063    0.0000    3    0    0    0    0
   5 CA   C_ALI    1.4530    0.0000    0.0000    3    6    7   20    0
   6 HA   H_ALI    1.7819   -0.4870    0.9180    5    0    0    0    0
   7 CB   C_ALI    1.9763   -0.8167   -1.1833    5    8   11   15    0
   8 HB   H_ALI    1.6329   -0.3367   -2.0997    7    0    0    0    0
   9 QG1  PSEUD    3.8693   -0.8354   -1.2105    0    0    0    0   19
  10 QG2  PSEUD    1.2807   -2.5773   -1.1531    0    0    0    0   19
  11 CG1  C_ALI    3.5061   -0.8318   -1.2052    7   12   13   14    0
  12 HG11 H_ALI    3.8514   -1.4191   -2.0561   11    0    0    0    9
  13 HG12 H_ALI    3.8783    0.1889   -1.2932   11    0    0    0    9
  14 HG13 H_ALI    3.8783   -1.2761   -0.2821   11    0    0    0    9
  15 CG2  C_ALI    1.4141   -2.2394   -1.1589    7   16   17   18    0
  16 HG21 H_ALI    1.8021   -2.7980   -2.0107   15    0    0    0   10
  17 HG22 H_ALI    1.7137   -2.7326   -0.2341   15    0    0    0   10
  18 HG23 H_ALI    0.3262   -2.2013   -1.2144   15    0    0    0   10
  19 QQG  PSEUD    2.5750   -1.7064   -1.1818    0    0    0    0    0
  20 C    C_BYL    1.9587    1.4440    0.0000    5   21   22    0    0
  21 O    O_BYL    1.1648    2.3835    0.0000   20    0    0    0    0
  22 N    N_AMI    3.2772    1.5756    0.0000   20    0    0    0    0
```

ADE

```
RESIDUE   ADE      9   38    3   37
   1 ZETA     0    0    0.0000    1    2    3    6    0
   2 ALPHA    0    0    0.0000    2    3    6    7    0
   3 BETA     0    0    0.0000    3    6    7   11    0
   4 GAMMA    0    0    0.0000    6    7   11   13    0
   5 DELTA    0    0    0.0000    7   11   13   37    0
   6 NU2      0    0    0.0000   11   13   15   19   36
   7 NU1      0    0    0.0000   13   15   19   21   36
   8 CHI      0    0    0.0000   21   19   22   23   36
   9 EPSI     0    0    0.0000   11   13   37   38    0
   1 C3′  C_ALI   -0.9681   -5.8551    2.5577    2    0    0    0    0
   2 O3′  O_EST   -0.6348   -4.7127    1.7719    1    3    0    0    0
   3 P    P_ALI    0.4817   -3.6875    2.2842    2    4    5    6    0
   4 OP1  O_BYL    1.7976   -4.3635    2.3259    3    0    0    0    0
   5 OP2  O_BYL    0.0167   -3.0215    3.5214    3    0    0    0    0
   6 O5′  O_EST    0.5255   -2.5916    1.1193    3    7    0    0    0
   7 C5′  C_ALI    1.4216   -1.4863    1.2135    6    8    9   11    0
   8 H5′  H_ALI    2.4568   -1.8260    1.2707    7    0    0    0   10
   9 H5″  H_ALI    1.2157   -0.8907    2.1041    7    0    0    0   10
  10 Q5′  PSEUD    1.8362   -1.3584    1.6874    0    0    0    0    0
  11 C4′  C_ALI    1.2779   -0.5959    0.0000    7   12   21   13    0
  12 H4′  H_ALI    1.5099   -1.1747   -0.8952   11    0    0    0    0
  13 C3′  C_ALI    2.1963    0.6228    0.0000   11   15   14   37    0
  14 H3′  H_ALI    3.1027    0.4293    0.5756   13    0    0    0    0
  15 C2′  C_ALI    1.3731    1.7403    0.5780   19   16   17   13    0
  16 H2′  H_ALI    1.4019    1.6718    1.6635   15    0    0    0   18
  17 H2″  H_ALI    1.7912    2.6897    0.2499   15    0    0    0   18
  18 Q2′  PSEUD    1.5965    2.1808    0.9567    0    0    0    0    0
  19 C1′  C_ALI    0.0000    1.4100    0.0000   21   20   22   15    0
  20 H1′  H_ALI   -0.0659    1.7932   -1.0183   19    0    0    0    0
  21 O4′  O_EST    0.0000    0.0000    0.0000   11   19    0    0    0
  22 N9   N_AMI   -1.1209    1.9441    0.8017   19   23   35    0    0
  23 C4   C_ARO   -2.3784    2.2371    0.3436   22   24   33    0    0
```

```
  24 N3   N_AMI   -2.8170    2.0867   -0.9370   23   25    0    0    0
  25 C2   C_ARO   -4.1036    2.4736   -1.0196   24   26   27    0    0
  26 H2   H_ARO   -4.6227    2.4240   -1.9654   25    0    0    0    0
  27 N1   N_AMI   -4.9152    2.9484   -0.0482   25   28    0    0    0
  28 C6   C_ARO   -4.4155    3.0749    1.2114   27   29   33    0    0
  29 N6   N_AMI   -5.2378    3.5526    2.1808   28   30   31    0    0
  30 H61  H_AMI   -6.1905    3.8024    1.9570   29    0    0    0   32
  31 H62  H_AMI   -4.8967    3.6581    3.1256   29    0    0    0   32
  32 Q6   PSEUD   -5.5436    3.7303    2.5413    0    0    0    0    0
  33 C5   C_ARO   -3.0716    2.6978    1.4177   23   28   34    0    0
  34 N7   N_AMI   -2.2897    2.7047    2.5414   33   35    0    0    0
  35 C8   C_ARO   -1.1186    2.2402    2.1184   22   34   36    0    0
  36 H8   H_ARO   -0.2357    2.0985    2.7240   35    0    0    0    0
  37 O3′  O_EST    2.6047    0.9094   -1.3466   13   38    0    0    0
  38 P    P_ALI    3.5778    2.1447   -1.6417   37    0    0    0    0
```

RADE

```
RESIDUE   RADE    10   38    3   37
   1 ZETA     0    0    0.00      1    2    3    6    0
   2 ALPHA    0    0    0.00      2    3    6    7    0
   3 BETA     0    0    0.00      3    6    7   11    0
   4 GAMMA    0    0    0.00      6    7   11   13    0
   5 DELTA    0    0    0.00      7   11   13   37    0
   6 NU2      0    0    0.00     11   13   15   19   36
   7 HOXI     0    0    0.00     13   15   17   18   18
   8 NU1      0    0    0.00     13   15   19   21   36
   9 CHI      0    0    0.00     21   19   22   23   36
  10 EPSI     0    0    0.00     11   13   37   38    0
   1 C3′  C_ALI   -0.9681   -5.8551    2.5577    2    0    0    0    0
   2 O3′  O_EST   -0.6348   -4.7127    1.7719    1    3    0    0    0
   3 P    P_ALI    0.4817   -3.6875    2.2842    2    4    5    6    0
   4 OP1  O_BYL    1.7976   -4.3635    2.3259    3    0    0    0    0
   5 OP2  O_BYL    0.0167   -3.0215    3.5214    3    0    0    0    0
   6 O5′  O_EST    0.5255   -2.5916    1.1193    3    7    0    0    0
   7 C5′  C_ALI    1.4216   -1.4863    1.2135    6    8    9   11    0
   8 H5′  H_ALI    2.4568   -1.8260    1.2707    7    0    0    0   10
   9 H5″  H_ALI    1.2157   -0.8907    2.1041    7    0    0    0   10
  10 Q5′  PSEUD    1.8362   -1.3584    1.6874    0    0    0    0    0
  11 C4′  C_ALI    1.2779   -0.5959    0.0000    7   12   13   21    0
  12 H4′  H_ALI    1.5099   -1.1747   -0.8952   11    0    0    0    0
  13 C3′  C_ALI    2.1963    0.6228    0.0000   11   14   15   37    0
  14 H3′  H_ALI    3.1027    0.4293    0.5756   13    0    0    0    0
  15 C2′  C_ALI    1.3731    1.7403    0.5780   13   16   17   19    0
  16 H2′  H_ALI    1.4019    1.6718    1.6635   15    0    0    0    0
  17 O2′  O_HYD    1.9936    2.8894    0.0464   15   18    0    0    0
  18 HO2′ H_OXY    2.1370    3.5157    0.7597   17    0    0    0    0
  19 C1′  C_ALI    0.0000    1.4100    0.0000   15   20   21   22    0
  20 H1′  H_ALI   -0.0659    1.7932   -1.0183   19    0    0    0    0
  21 O4′  O_EST    0.0000    0.0000    0.0000   11   19    0    0    0
  22 N9   N_AMI   -1.1209    1.9441    0.8017   19   23   35    0    0
  23 C4   C_ARO   -2.3784    2.2371    0.3436   22   24   33    0    0
  24 N3   N_AMI   -2.8170    2.0867   -0.9370   23   25    0    0    0
  25 C2   C_ARO   -4.1036    2.4736   -1.0196   24   26   27    0    0
  26 H2   H_ARO   -4.6227    2.4240   -1.9654   25    0    0    0    0
  27 N1   N_AMI   -4.9152    2.9484   -0.0482   25   28    0    0    0
  28 C6   C_ARO   -4.4155    3.0749    1.2114   27   29   33    0    0
  29 N6   N_AMI   -5.2378    3.5526    2.1808   28   30   31    0    0
  30 H61  H_AMI   -6.1905    3.8024    1.9570   29    0    0    0   32
  31 H62  H_AMI   -4.8967    3.6581    3.1256   29    0    0    0   32
  32 Q6   PSEUD   -5.5436    3.7303    2.5413    0    0    0    0    0
  33 C5   C_ARO   -3.0716    2.6978    1.4177   23   28   34    0    0
  34 N7   N_AMI   -2.2897    2.7047    2.5414   33   35    0    0    0
  35 C8   C_ARO   -1.1186    2.2402    2.1184   22   34   36    0    0
  36 H8   H_ARO   -0.2357    2.0985    2.7240   35    0    0    0    0
  37 O3′  O_EST    2.6047    0.9094   -1.3466   13   38    0    0    0
  38 P    P_ALI    3.5778    2.1447   -1.6417   37    0    0    0    0
```

CYT

```
RESIDUE   CYT      9   36    3   35
   1 ZETA     0    0    0.00      1    2    3    6    0
   2 ALPHA    0    0    0.00      2    3    6    7    0
   3 BETA     0    0    0.00      3    6    7   11    0
   4 GAMMA    0    0    0.00      6    7   11   13    0
   5 DELTA    0    0    0.0000    7   11   13   35    0
   6 NU2      0    0    0.0000   11   13   15   19   34
   7 NU1      0    0    0.0000   13   15   19   21   34
   8 CHI      0    0    0.0000   21   19   22   23   34
```

```
  9 EPSI     0    0    0.0000   11   13   35   36    0
  1 C3′   C_ALI   -0.9681   -5.8551    2.5577    2    0    0    0    0
  2 O3′   O_EST   -0.6348   -4.7127    1.7719    1    3    0    0    0
  3 P     P_ALI    0.4817   -3.6875    2.2842    2    4    5    6    0
  4 OP1   O_BYL    1.7976   -4.3635    2.3259    3    0    0    0    0
  5 OP2   O_BYL    0.0167   -3.0215    3.5214    3    0    0    0    0
  6 O5′   O_EST    0.5255   -2.5916    1.1193    3    7    0    0    0
  7 C5′   C_ALI    1.4216   -1.4863    1.2135    6    8    9   11    0
  8 H5′   H_ALI    2.4568   -1.8260    1.2707    7    0    0    0   10
  9 H5″   H_ALI    1.2157   -0.8907    2.1041    7    0    0    0   10
 10 Q5′   PSEUD    1.8362   -1.3584    1.6874    0    0    0    0    0
 11 C4′   C_ALI    1.2779   -0.5959    0.0000    7   12   21   13    0
 12 H4′   H_ALI    1.5099   -1.1747   -0.8952   11    0    0    0    0
 13 C3′   C_ALI    2.1963    0.6228    0.0000   11   15   14   35    0
 14 H3′   H_ALI    3.1027    0.4293    0.5756   13    0    0    0    0
 15 C2′   C_ALI    1.3731    1.7403    0.5780   19   16   17   13    0
 16 H2′   H_ALI    1.4019    1.6718    1.6635   15    0    0    0   18
 17 H2″   H_ALI    1.7912    2.6897    0.2499   15    0    0    0   18
 18 Q2′   PSEUD    1.5965    2.1808    0.9567    0    0    0    0    0
 19 C1′   C_ALI    0.0000    1.4100    0.0000   21   20   22   15    0
 20 H1′   H_ALI   -0.0659    1.7932   -1.0183   19    0    0    0    0
 21 O4′   O_EST    0.0000    0.0000    0.0000   11   19    0    0    0
 22 N1    N_AMI   -1.1209    1.9441    0.8017   19   23   33    0    0
 23 C2    C_ARO   -2.3304    2.1782    0.1493   22   24   25    0    0
 24 O2    O_BYL   -2.4085    1.9304   -1.0590   23    0    0    0    0
 25 N3    N_AMI   -3.3653    2.6642    0.8578   23   26    0    0    0
 26 C4    C_ARO   -3.2644    2.9228    2.1462   25   27   31    0    0
 27 N4    N_AMI   -4.2972    3.4021    2.8279   26   28   29    0    0
 28 H41   H_AMI   -5.1756    3.5735    2.3597   27    0    0    0   30
 29 H42   H_AMI   -4.2049    3.5955    3.8149   27    0    0    0   30
 30 Q4    PSEUD   -4.6903    3.5845    3.0873    0    0    0    0    0
 31 C5    C_ARO   -2.0221    2.6890    2.8462   26   32   33    0    0
 32 H5    H_ARO   -1.9484    2.9039    3.9020   31    0    0    0    0
 33 C6    C_ARO   -0.9996    2.2045    2.1264   22   31   34    0    0
 34 H6    H_ARO   -0.0488    2.0117    2.6008   33    0    0    0    0
 35 O3′   O_EST    2.6047    0.9094   -1.3466   13   36    0    0    0
 36 P     P_ALI    3.5778    2.1447   -1.6417   35    0    0    0    0
```

RCYT

```
RESIDUE   RCYT    10   36    3   35
  1 ZETA     0    0    0.00     1    2    3    6    0
  2 ALPHA    0    0    0.00     2    3    6    7    0
  3 BETA     0    0    0.00     3    6    7   11    0
  4 GAMMA    0    0    0.00     6    7   11   13    0
  5 DELTA    0    0    0.00     7   11   13   35    0
  6 NU2      0    0    0.00    11   13   15   19   34
  7 HOXI     0    0    0.00    13   15   17   18   18
  8 NU1      0    0    0.00    13   15   19   21   34
  9 CHI      0    0    0.00    21   19   22   23   34
 10 EPSI     0    0    0.00    11   13   35   36    0
  1 C3′   C_ALI   -0.9681   -5.8551    2.5577    2    0    0    0    0
  2 O3′   O_EST   -0.6348   -4.7127    1.7719    1    3    0    0    0
  3 P     P_ALI    0.4817   -3.6875    2.2842    2    4    5    6    0
  4 OP1   O_BYL    1.7976   -4.3635    2.3259    3    0    0    0    0
  5 OP2   O_BYL    0.0167   -3.0215    3.5214    3    0    0    0    0
  6 O5′   O_EST    0.5255   -2.5916    1.1193    3    7    0    0    0
  7 C5′   C_ALI    1.4216   -1.4863    1.2135    6    8    9   11    0
  8 H5′   H_ALI    2.4568   -1.8260    1.2707    7    0    0    0   10
  9 H5″   H_ALI    1.2157   -0.8907    2.1041    7    0    0    0   10
 10 Q5′   PSEUD    1.8362   -1.3584    1.6874    0    0    0    0    0
 11 C4′   C_ALI    1.2779   -0.5959    0.0000    7   12   13   21    0
 12 H4′   H_ALI    1.5099   -1.1747   -0.8952   11    0    0    0    0
 13 C3′   C_ALI    2.1963    0.6228    0.0000   11   14   15   35    0
 14 H3′   H_ALI    3.1027    0.4293    0.5756   13    0    0    0    0
 15 C2′   C_ALI    1.3731    1.7403    0.5780   13   16   17   19    0
 16 H2′   H_ALI    1.4019    1.6718    1.6635   15    0    0    0    0
 17 O2′   O_HYD    1.9936    2.8894    0.0464   15   18    0    0    0
 18 HO2′  H_OXY    2.1370    3.5157    0.7597   17    0    0    0    0
 19 C1′   C_ALI    0.0000    1.4100    0.0000   15   20   21   22    0
 20 H1′   H_ALI   -0.0659    1.7932   -1.0183   19    0    0    0    0
 21 O4′   O_EST    0.0000    0.0000    0.0000   11   19    0    0    0
 22 N1    N_AMI   -1.1209    1.9441    0.8017   19   23   33    0    0
 23 C2    C_ARO   -2.3304    2.1782    0.1493   22   24   25    0    0
 24 O2    O_BYL   -2.4085    1.9304   -1.0590   23    0    0    0    0
 25 N3    N_AMI   -3.3653    2.6642    0.8578   23   26    0    0    0
 26 C4    C_ARO   -3.2644    2.9228    2.1462   25   27   31    0    0
 27 N4    N_AMI   -4.2972    3.4021    2.8279   26   28   29    0    0
```

```
  28 H41  H_AMI   -5.1756    3.5735    2.3597   27    0    0    0   30
  29 H42  H_AMI   -4.2049    3.5955    3.8149   27    0    0    0   30
  30 Q4   PSEUD   -4.6903    3.5845    3.0873    0    0    0    0    0
  31 C5   C_ARO   -2.0221    2.6890    2.8462   26   32   33    0    0
  32 H5   H_ARO   -1.9484    2.9039    3.9020   31    0    0    0    0
  33 C6   C_ARO   -0.9996    2.2045    2.1264   22   31   34    0    0
  34 H6   H_ARO   -0.0488    2.0117    2.6008   33    0    0    0    0
  35 O3'  O_EST    2.6047    0.9094   -1.3466   13   36    0    0    0
  36 P    P_ALI    3.5778    2.1447   -1.6417   35    0    0    0    0
```

GUA

```
RESIDUE   GUA      9   39    3   38
   1 ZETA     0    0    0.0000    1    2    3    6    0
   2 ALPHA    0    0    0.0000    2    3    6    7    0
   3 BETA     0    0    0.0000    3    6    7   11    0
   4 GAMMA    0    0    0.0000    6    7   11   13    0
   5 DELTA    0    0    0.0000    7   11   13   38    0
   6 NU2      0    0    0.0000   11   13   15   19   37
   7 NU1      0    0    0.0000   13   15   19   21   37
   8 CHI      0    0    0.0000   21   19   22   23   37
   9 EPSI     0    0    0.0000   11   13   38   39    0
   1 C3'  C_ALI   -0.9681   -5.8551    2.5577    2    0    0    0    0
   2 O3'  O_EST   -0.6348   -4.7127    1.7719    1    3    0    0    0
   3 P    P_ALI    0.4817   -3.6875    2.2842    2    4    5    6    0
   4 OP1  O_BYL    1.7976   -4.3635    2.3259    3    0    0    0    0
   5 OP2  O_BYL    0.0167   -3.0215    3.5214    3    0    0    0    0
   6 O5'  O_EST    0.5255   -2.5916    1.1193    3    7    0    0    0
   7 C5'  C_ALI    1.4216   -1.4863    1.2135    6    8    9   11    0
   8 H5'  H_ALI    2.4568   -1.8260    1.2707    7    0    0    0   10
   9 H5"  H_ALI    1.2157   -0.8907    2.1041    7    0    0    0   10
  10 Q5'  PSEUD    1.8362   -1.3584    1.6874    0    0    0    0    0
  11 C4'  C_ALI    1.2779   -0.5959    0.0000    7   12   21   13    0
  12 H4'  H_ALI    1.5099   -1.1747   -0.8952   11    0    0    0    0
  13 C3'  C_ALI    2.1963    0.6228    0.0000   11   15   14   38    0
  14 H3'  H_ALI    3.1027    0.4293    0.5756   13    0    0    0    0
  15 C2'  C_ALI    1.3731    1.7403    0.5780   19   16   17   13    0
  16 H2'  H_ALI    1.4019    1.6718    1.6635   15    0    0    0   18
  17 H2"  H_ALI    1.7912    2.6897    0.2499   15    0    0    0   18
  18 Q2'  PSEUD    1.5965    2.1808    0.9567    0    0    0    0    0
  19 C1'  C_ALI    0.0000    1.4100    0.0000   21   20   22   15    0
  20 H1'  H_ALI   -0.0659    1.7932   -1.0183   19    0    0    0    0
  21 O4'  O_EST    0.0000    0.0000    0.0000   11   19    0    0    0
  22 N9   N_AMI   -1.1209    1.9441    0.8017   19   23   36    0    0
  23 C4   C_ARO   -2.3784    2.2371    0.3436   22   24   34    0    0
  24 N3   N_AMI   -2.8170    2.0867   -0.9370   23   25    0    0    0
  25 C2   C_ARO   -4.1036    2.4736   -1.0196   24   26   30    0    0
  26 N2   N_AMI   -4.7356    2.4092   -2.1892   25   27   28    0    0
  27 H21  H_AMI   -5.7010    2.6979   -2.2581   26    0    0    0   29
  28 H22  H_AMI   -4.2498    2.0716   -3.0078   26    0    0    0   29
  29 Q2   PSEUD   -4.9754    2.3848   -2.6330    0    0    0    0    0
  30 N1   N_AMI   -4.9152    2.9484   -0.0482   25   31   32    0    0
  31 H1   H_AMI   -5.8690    3.2001   -0.2652   30    0    0    0    0
  32 C6   C_ARO   -4.4155    3.0749    1.2114   30   33   34    0    0
  33 O6   O_BYL   -5.1745    3.5092    2.0764   32    0    0    0    0
  34 C5   C_ARO   -3.0716    2.6978    1.4177   23   32   35    0    0
  35 N7   N_AMI   -2.2897    2.7047    2.5414   34   36    0    0    0
  36 C8   C_ARO   -1.1186    2.2402    2.1184   22   35   37    0    0
  37 H8   H_ARO   -0.2357    2.0985    2.7240   36    0    0    0    0
  38 O3'  O_EST    2.6047    0.9094   -1.3466   13   39    0    0    0
  39 P    P_ALI    3.5778    2.1447   -1.6417   38    0    0    0    0
```

RGUA

```
RESIDUE   RGUA    10   39    3   38
   1 ZETA     0    0    0.00     1    2    3    6    0
   2 ALPHA    0    0    0.00     2    3    6    7    0
   3 BETA     0    0    0.00     3    6    7   11    0
   4 GAMMA    0    0    0.00     6    7   11   13    0
   5 DELTA    0    0    0.00     7   11   13   38    0
   6 NU2      0    0    0.00    11   13   15   19   37
   7 HOXI     0    0    0.00    13   15   17   18   18
   8 NU1      0    0    0.00    13   15   19   21   37
   9 CHI      0    0    0.00    21   19   22   23   37
  10 EPSI     0    0    0.00    11   13   38   39    0
   1 C3'  C_ALI   -0.9681   -5.8551    2.5577    2    0    0    0    0
   2 O3'  O_EST   -0.6348   -4.7127    1.7719    1    3    0    0    0
   3 P    P_ALI    0.4817   -3.6875    2.2842    2    4    5    6    0
   4 OP1  O_BYL    1.7976   -4.3635    2.3259    3    0    0    0    0
```

```
    5 OP2  O_BYL    0.0167   -3.0215    3.5214    3    0    0    0    0
    6 O5′  O_EST    0.5255   -2.5916    1.1193    3    7    0    0    0
    7 C5′  C_ALI    1.4216   -1.4863    1.2135    6    8    9   11    0
    8 H5′  H_ALI    2.4568   -1.8260    1.2707    7    0    0    0   10
    9 H5″  H_ALI    1.2157   -0.8907    2.1041    7    0    0    0   10
   10 Q5′  PSEUD    1.8362   -1.3584    1.6874    0    0    0    0    0
   11 C4′  C_ALI    1.2779   -0.5959    0.0000    7   12   13   21    0
   12 H4′  H_ALI    1.5099   -1.1747   -0.8952   11    0    0    0    0
   13 C3′  C_ALI    2.1963    0.6228    0.0000   11   14   15   38    0
   14 H3′  H_ALI    3.1027    0.4293    0.5756   13    0    0    0    0
   15 C2′  C_ALI    1.3731    1.7403    0.5780   13   16   17   19    0
   16 H2′  H_ALI    1.4019    1.6718    1.6635   15    0    0    0    0
   17 O2′  O_HYD    1.9936    2.8894    0.0464   15   18    0    0    0
   18 HO2′ H_OXY    2.1370    3.5157    0.7597   17    0    0    0    0
   19 C1′  C_ALI    0.0000    1.4100    0.0000   15   20   21   22    0
   20 H1′  H_ALI   -0.0659    1.7932   -1.0183   19    0    0    0    0
   21 O4′  O_EST    0.0000    0.0000    0.0000   11   19    0    0    0
   22 N9   N_AMI   -1.1209    1.9441    0.8017   19   23   36    0    0
   23 C4   C_ARO   -2.3784    2.2371    0.3436   22   24   34    0    0
   24 N3   N_AMI   -2.8170    2.0867   -0.9370   23   25    0    0    0
   25 C2   C_ARO   -4.1036    2.4736   -1.0196   24   26   30    0    0
   26 N2   N_AMI   -4.7356    2.4092   -2.1892   25   27   28    0    0
   27 H21  H_AMI   -5.7010    2.6979   -2.2581   26    0    0    0   29
   28 H22  H_AMI   -4.2498    2.0716   -3.0078   26    0    0    0   29
   29 Q2   PSEUD   -4.9754    2.3848   -2.6330    0    0    0    0    0
   30 N1   N_AMI   -4.9152    2.9484   -0.0482   25   31   32    0    0
   31 H1   H_AMI   -5.8690    3.2001   -0.2652   30    0    0    0    0
   32 C6   C_ARO   -4.4155    3.0749    1.2114   30   33   34    0    0
   33 O6   O_BYL   -5.1745    3.5092    2.0764   32    0    0    0    0
   34 C5   C_ARO   -3.0716    2.6978    1.4177   23   32   35    0    0
   35 N7   N_AMI   -2.2897    2.7047    2.5414   34   36    0    0    0
   36 C8   C_ARO   -1.1186    2.2402    2.1184   22   35   37    0    0
   37 H8   H_ARO   -0.2357    2.0985    2.7240   36    0    0    0    0
   38 O3′  O_EST    2.6047    0.9094   -1.3466   36   39    0    0    0
   39 P    P_ALI    3.5778    2.1447   -1.6417   38    0    0    0    0
```

THY

```
RESIDUE   THY     10   38    3   37
   1 ZETA     0    0    0.00      1    2    3    6    0
   2 ALPHA    0    0    0.00      2    3    6    7    0
   3 BETA     0    0    0.00      3    6    7   11    0
   4 GAMMA    0    0    0.00      6    7   11   13    0
   5 DELTA    0    0    0.0000    7   11   13   37    0
   6 NU2      0    0    0.0000   11   13   15   19   36
   7 NU1      0    0    0.0000   13   15   19   21   36
   8 CHI      0    0    0.0000   21   19   22   23   36
   9 CHI2     0    0    0.00     27   29   30   31   33
  10 EPSI     0    0    0.00     11   13   37   38    0
   1 C3′  C_ALI   -0.9681   -5.8551    2.5577    2    0    0    0    0
   2 O3′  O_EST   -0.6348   -4.7127    1.7719    1    3    0    0    0
   3 P    P_ALI    0.4817   -3.6875    2.2842    2    4    5    6    0
   4 OP1  O_BYL    1.7976   -4.3635    2.3259    3    0    0    0    0
   5 OP2  O_BYL    0.0167   -3.0215    3.5214    3    0    0    0    0
   6 O5′  O_EST    0.5255   -2.5916    1.1193    3    7    0    0    0
   7 C5′  C_ALI    1.4216   -1.4863    1.2135    6    8    9   11    0
   8 H5′  H_ALI    2.4568   -1.8260    1.2707    7    0    0    0   10
   9 H5″  H_ALI    1.2157   -0.8907    2.1041    7    0    0    0   10
  10 Q5′  PSEUD    1.8362   -1.3584    1.6874    0    0    0    0    0
  11 C4′  C_ALI    1.2779   -0.5959    0.0000    7   12   21   13    0
  12 H4′  H_ALI    1.5099   -1.1747   -0.8952   11    0    0    0    0
  13 C3′  C_ALI    2.1963    0.6228    0.0000   11   15   14   37    0
  14 H3′  H_ALI    3.1027    0.4293    0.5756   13    0    0    0    0
  15 C2′  C_ALI    1.3731    1.7403    0.5780   19   16   17   13    0
  16 H2′  H_ALI    1.4019    1.6718    1.6635   15    0    0    0   18
  17 H2″  H_ALI    1.7912    2.6897    0.2499   15    0    0    0   18
  18 Q2′  PSEUD    1.5965    2.1808    0.9567    0    0    0    0    0
  19 C1′  C_ALI    0.0000    1.4100    0.0000   21   20   22   15    0
  20 H1′  H_ALI   -0.0659    1.7932   -1.0183   19    0    0    0    0
  21 O4′  O_EST    0.0000    0.0000    0.0000   11   19    0    0    0
  22 N1   N_AMI   -1.1209    1.9441    0.8017   19   23   35    0    0
  23 C2   C_ARO   -2.2881    2.1620    0.1364   22   24   25    0    0
  24 O2   O_BYL   -2.4402    1.9414   -1.0548   23    0    0    0    0
  25 N3   N_AMI   -3.3338    2.6596    0.8814   23   26   27    0    0
  26 H3   H_AMI   -4.2250    2.8415    0.4424   25    0    0    0    0
  27 C4   C_ARO   -3.2471    2.9346    2.2228   25   28   29    0    0
  28 O4   O_BYL   -4.2464    3.3768    2.7874   27    0    0    0    0
  29 C5   C_ARO   -2.0063    2.6890    2.8681   27   30   35    0    0
```

```
  30 C7   C_ALI   -1.8531    2.9733    4.3432   29   31   32   33    0
  31 H71  H_ALI   -0.8409    2.7265    4.6636   30    0    0    0    0
  32 H72  H_ALI   -2.5641    2.3728    4.9107   30    0    0    0    0
  33 H73  H_ALI   -2.0420    4.0291    4.5373   30    0    0    0    0
  34 Q7   PSEUD   -1.8157    3.0428    4.7039    0    0    0    0    0
  35 C6   C_ARO   -1.0061    2.2067    2.1270   22   29   36    0    0
  36 H6   H_ARO   -0.0464    2.0062    2.5799   35    0    0    0    0
  37 O3'  O_EST    2.6047    0.9094   -1.3466   13   38    0    0    0
  38 P    P_ALI    3.5778    2.1447   -1.6417   37    0    0    0    0
```

URA

```
RESIDUE   URA     10   34    3   33
   1 ZETA     0    0    0.00      1    2    3    6    0
   2 ALPHA    0    0    0.00      2    3    6    7    0
   3 BETA     0    0    0.00      3    6    7   11    0
   4 GAMMA    0    0    0.00      6    7   11   13    0
   5 DELTA    0    0    0.00      7   11   13   33    0
   6 NU2      0    0    0.00     11   13   15   19   32
   7 HOXI     0    0    0.00     13   15   17   18   18
   8 NU1      0    0    0.00     13   15   19   21   32
   9 CHI      0    0    0.00     21   19   22   23   32
  10 EPSI     0    0    0.00     11   13   33   34    0
   1 C3'  C_ALI   -0.9681   -5.8551    2.5577    2    0    0    0    0
   2 O3'  O_EST   -0.6348   -4.7127    1.7719    1    3    0    0    0
   3 P    P_ALI    0.4817   -3.6875    2.2842    2    4    5    6    0
   4 OP1  O_BYL    1.7976   -4.3635    2.3259    3    0    0    0    0
   5 OP2  O_BYL    0.0167   -3.0215    3.5214    3    0    0    0    0
   6 O5'  O_EST    0.5255   -2.5916    1.1193    3    7    0    0    0
   7 C5'  C_ALI    1.4216   -1.4863    1.2135    6    8    9   11    0
   8 H5'  H_ALI    2.4568   -1.8260    1.2707    7    0    0    0   10
   9 H5"  H_ALI    1.2157   -0.8907    2.1041    7    0    0    0   10
  10 Q5'  PSEUD    1.8362   -1.3584    1.6874    0    0    0    0    0
  11 C4'  C_ALI    1.2779   -0.5959    0.0000    7   12   13   21    0
  12 H4'  H_ALI    1.5099   -1.1747   -0.8952   11    0    0    0    0
  13 C3'  C_ALI    2.1963    0.6228    0.0000   11   14   15   33    0
  14 H3'  H_ALI    3.1027    0.4293    0.5756   13    0    0    0    0
  15 C2'  C_ALI    1.3731    1.7403    0.5780   13   16   17   19    0
  16 H2'  H_ALI    1.4019    1.6718    1.6635   15    0    0    0    0
  17 O2'  O_HYD    1.9936    2.8894    0.0464   15   18    0    0    0
  18 HO2' H_OXY    2.1370    3.5157    0.7597   17    0    0    0    0
  19 C1'  C_ALI    0.0000    1.4100    0.0000   15   20   21   22    0
  20 H1'  H_ALI   -0.0659    1.7932   -1.0183   19    0    0    0    0
  21 O4'  O_EST    0.0000    0.0000    0.0000   11   19    0    0    0
  22 N1   N_AMI   -1.1209    1.9441    0.8017   19   23   31    0    0
  23 C2   C_ARO   -2.2881    2.1620    0.1364   22   24   25    0    0
  24 O2   O_BYL   -2.4402    1.9414   -1.0548   23    0    0    0    0
  25 N3   N_AMI   -3.3338    2.6596    0.8814   23   26   27    0    0
  26 H3   H_AMI   -4.2250    2.8415    0.4424   25    0    0    0    0
  27 C4   C_ARO   -3.2471    2.9346    2.2228   25   28   29    0    0
  28 O4   O_BYL   -4.2464    3.3768    2.7874   27    0    0    0    0
  29 C5   C_ARO   -2.0063    2.6890    2.8681   27   30   31    0    0
  30 H5   C_ALI   -1.8967    2.8923    3.9231   29    0    0    0    0
  31 C6   C_ARO   -1.0061    2.2067    2.1270   22   29   32    0    0
  32 H6   H_ARO   -0.0464    2.0062    2.5799   31    0    0    0    0
  33 O3'  O_EST    2.6047    0.9094   -1.3466   13   34    0    0    0
  34 P    P_ALI    3.5778    2.1447   -1.6417   33    0    0    0    0
```

PL

```
RESIDUE   PL       1    6    3    5
   1 LB       0    0    0.00      2    3    4    5    0
   1 C    PSEUD   -0.6824   -1.1357    0.0000    0    0    0    0    0
   2 O    PSEUD   -0.1723   -2.2550    0.0000    0    0    0    0    0
   3 N    PSEUD    0.0000    0.0000    0.0000    0    0    0    0    0
   4 Q1   PSEUD    0.9971   -0.0762    0.0000    0    0    0    0    0
   5 Q2   PSEUD    1.0733    0.9209    0.0000    0    0    0    0    0
   6 Q3   PSEUD    2.0704    0.8447    0.0000    0    0    0    0    0
```

NL

```
RESIDUE   NL       1    6    3    5
   1 LB       0    0    0.00      2    3    4    5    0
   1 C3'  PSEUD   -0.9681   -5.8551    2.5577    0    0    0    0    0
   2 O3'  PSEUD   -0.6348   -4.7127    1.7719    0    0    0    0    0
   3 P    PSEUD    0.4817   -3.6875    2.2842    0    0    0    0    0
   4 Q1   PSEUD    0.3487   -3.1324    1.4631    0    0    0    0    0
   5 Q2   PSEUD    1.0465   -2.4917    1.7833    0    0    0    0    0
   6 Q3   PSEUD    0.9136   -1.9366    0.9622    0    0    0    0    0
```

LL

```
RESIDUE   LL       1    6    3    5
   1 LB       0    0    0.00      2    3    4    5    0
   1 Q1    PSEUD    0.0000    0.0000    0.0000    0    0    0    0    0
   2 Q2    PSEUD    1.0000    0.0000    0.0000    0    0    0    0    0
   3 Q3    PSEUD    1.0000    1.0000    0.0000    0    0    0    0    0
   4 Q1    PSEUD    2.0000    1.0000    0.0000    0    0    0    0    0
   5 Q2    PSEUD    2.0000    0.0000    0.0000    0    0    0    0    0
   6 Q3    PSEUD    3.0000    0.0000    0.0000    0    0    0    0    0
```

LL2

```
RESIDUE   LL2      1    6    3    5
   1 LB       0    0    0.00      2    3    4    5    0
   1 Q1    PSEUD    0.0000    0.0000    0.0000    0    0    0    0    0
   2 Q2    PSEUD    2.0000    0.0000    0.0000    0    0    0    0    0
   3 Q3    PSEUD    2.0000    2.0000    0.0000    0    0    0    0    0
   4 Q1    PSEUD    4.0000    2.0000    0.0000    0    0    0    0    0
   5 Q2    PSEUD    4.0000    0.0000    0.0000    0    0    0    0    0
   6 Q3    PSEUD    6.0000    0.0000    0.0000    0    0    0    0    0
```

LL5

```
RESIDUE   LL5      1    6    3    5
   1 LB       0    0    0.00      2    3    4    5    0
   1 Q1    PSEUD    0.0000    0.0000    0.0000    0    0    0    0    0
   2 Q2    PSEUD    5.0000    0.0000    0.0000    0    0    0    0    0
   3 Q3    PSEUD    5.0000    5.0000    0.0000    0    0    0    0    0
   4 Q1    PSEUD   10.0000    5.0000    0.0000    0    0    0    0    0
   5 Q2    PSEUD   10.0000    0.0000    0.0000    0    0    0    0    0
   6 Q3    PSEUD   15.0000    0.0000    0.0000    0    0    0    0    0
```

LP

```
RESIDUE   LP       1    6    3    5
   1 LB       0    0    0.00      2    3    4    5    0
   1 Q1    PSEUD    0.0000    0.0000    0.0000    0    0    0    0    0
   2 Q2    PSEUD    1.0000    0.0000    0.0000    0    0    0    0    0
   3 Q3    PSEUD    1.0000    1.0000    0.0000    0    0    0    0    0
   4 C     PSEUD    2.0000    1.0000    0.0000    0    0    0    0    0
   5 O     PSEUD    2.0000   -0.2300    0.0000    0    0    0    0    0
   6 N     PSEUD    3.0920    1.7505    0.0000    0    0    0    0    0
```

LN

```
RESIDUE   LN       1    6    3    5
   1 LB       0    0    0.00      2    3    4    5    0
   1 Q1    PSEUD    0.0000    0.0000    0.0000    0    0    0    0    0
   2 Q2    PSEUD    1.0000    0.0000    0.0000    0    0    0    0    0
   3 Q3    PSEUD    1.0000    1.0000    0.0000    0    0    0    0    0
   4 C3'   PSEUD    2.0000    1.0000    0.0000    0    0    0    0    0
   5 O3'   PSEUD    2.0000   -0.4260    0.0000    0    0    0    0    0
   6 P     PSEUD    3.3856   -1.2260    0.0000    0    0    0    0    0
```

PLM

```
RESIDUE   PLM      1    8    3    7
   1 LB       0    0    0.00      2    3    4    6    0
   1 C     PSEUD   -0.6824   -1.1357    0.0000    0    0    0    0    0
   2 O     PSEUD   -0.1723   -2.2550    0.0000    0    0    0    0    0
   3 N     PSEUD    0.0000    0.0000    0.0000    0    0    0    0    0
   4 Q1    PSEUD    0.9971   -0.0762    0.0000    0    0    0    0    0
   5 Q1'   PSEUD    0.9971   -0.0762   10.0000    0    0    0    0    0
   6 Q2    PSEUD    1.0733    0.9209    0.0000    0    0    0    0    0
   7 Q2'   PSEUD    1.0733    0.9209   10.0000    0    0    0    0    0
   8 Q3    PSEUD    2.0704    0.8447    0.0000    0    0    0    0    0
```

NLM

```
RESIDUE   NLM      1    8    3    7
   1 LB       0    0    0.00      2    3    4    6    0
   1 C3'   PSEUD   -0.9681   -5.8551    2.5577    0    0    0    0    0
   2 O3'   PSEUD   -0.6348   -4.7127    1.7719    0    0    0    0    0
   3 P     PSEUD    0.4817   -3.6875    2.2842    0    0    0    0    0
   4 Q1    PSEUD    0.3487   -3.1324    1.4631    0    0    0    0    0
   5 Q1'   PSEUD    0.3487   -3.1324   10.0000    0    0    0    0    0
   6 Q2    PSEUD    1.0465   -2.4917    1.7833    0    0    0    0    0
   7 Q2'   PSEUD    1.0465   -2.4917   10.0000    0    0    0    0    0
   8 Q3    PSEUD    0.9136   -1.9366    0.9622    0    0    0    0    0
```

LLM

```
RESIDUE   LLM      1    9    3    8
```

```
   1 LB       0    0    0.00      2    3    5    7    0
   1 Q1   PSEUD    0.0000    0.0000    0.0000    0    0    0    0    0
   2 Q2   PSEUD    1.0000    0.0000    0.0000    0    0    0    0    0
   3 Q3   PSEUD    1.0000    1.0000    0.0000    0    0    0    0    0
   4 Q3'  PSEUD    1.0000    1.0000   10.0000    0    0    0    0    0
   5 Q1   PSEUD    2.0000    1.0000    0.0000    0    0    0    0    0
   6 Q1'  PSEUD    2.0000    1.0000   10.0000    0    0    0    0    0
   7 Q2   PSEUD    2.0000    0.0000    0.0000    0    0    0    0    0
   8 Q2'  PSEUD    2.0000    0.0000   10.0000    0    0    0    0    0
   9 Q3   PSEUD    3.0000    0.0000    0.0000    0    0    0    0    0
```

LLM2

```
RESIDUE   LLM2     1    9    3    8
   1 LB       0    0    0.00      2    3    5    7    0
   1 Q1   PSEUD    0.0000    0.0000    0.0000    0    0    0    0    0
   2 Q2   PSEUD    2.0000    0.0000    0.0000    0    0    0    0    0
   3 Q3   PSEUD    2.0000    2.0000    0.0000    0    0    0    0    0
   4 Q3'  PSEUD    2.0000    2.0000   10.0000    0    0    0    0    0
   5 Q1   PSEUD    4.0000    2.0000    0.0000    0    0    0    0    0
   6 Q1'  PSEUD    4.0000    2.0000   10.0000    0    0    0    0    0
   7 Q2   PSEUD    4.0000    0.0000    0.0000    0    0    0    0    0
   8 Q2'  PSEUD    4.0000    0.0000   10.0000    0    0    0    0    0
   9 Q3   PSEUD    6.0000    0.0000    0.0000    0    0    0    0    0
```

LLM5

```
RESIDUE   LLM5     1    9    3    8
   1 LB       0    0    0.00      2    3    5    7    0
   1 Q1   PSEUD    0.0000    0.0000    0.0000    0    0    0    0    0
   2 Q2   PSEUD    5.0000    0.0000    0.0000    0    0    0    0    0
   3 Q3   PSEUD    5.0000    5.0000    0.0000    0    0    0    0    0
   4 Q3'  PSEUD    5.0000    5.0000   10.0000    0    0    0    0    0
   5 Q1   PSEUD   10.0000    5.0000    0.0000    0    0    0    0    0
   6 Q1'  PSEUD   10.0000    5.0000   10.0000    0    0    0    0    0
   7 Q2   PSEUD   10.0000    0.0000    0.0000    0    0    0    0    0
   8 Q2'  PSEUD   10.0000    0.0000   10.0000    0    0    0    0    0
   9 Q3   PSEUD   15.0000    0.0000    0.0000    0    0    0    0    0
```

LPM

```
RESIDUE   LPM      1    7    3    6
   1 LB       0    0    0.00      2    3    5    6    0
   1 Q1   PSEUD    0.0000    0.0000    0.0000    0    0    0    0    0
   2 Q2   PSEUD    1.0000    0.0000    0.0000    0    0    0    0    0
   3 Q3   PSEUD    1.0000    1.0000    0.0000    0    0    0    0    0
   4 Q3'  PSEUD    1.0000    1.0000   10.0000    0    0    0    0    0
   5 C    PSEUD    2.0000    1.0000    0.0000    0    0    0    0    0
   6 O    PSEUD    2.0000   -0.2300    0.0000    0    0    0    0    0
   7 N    PSEUD    3.0920    1.7505    0.0000    0    0    0    0    0
```

LNM

```
RESIDUE   LNM      1    7    3    6
   1 LB       0    0    0.00      2    3    5    6    0
   1 Q1   PSEUD    0.0000    0.0000    0.0000    0    0    0    0    0
   2 Q2   PSEUD    1.0000    0.0000    0.0000    0    0    0    0    0
   3 Q3   PSEUD    1.0000    1.0000    0.0000    0    0    0    0    0
   4 Q3'  PSEUD    1.0000    1.0000   10.0000    0    0    0    0    0
   5 C3'  PSEUD    2.0000    1.0000    0.0000    0    0    0    0    0
   6 O3'  PSEUD    2.0000   -0.4260    0.0000    0    0    0    0    0
   7 P    PSEUD    3.3856   -1.2260    0.0000    0    0    0    0    0
```

LGLY

```
RESIDUE   LGLY     3   11    3   10
   1 OMEGA   -1    2   10.0000    2    1    3    4    0
   2 PHI      0    0    0.0000    1    3    5    9    0
   3 PSI      0    0    0.0000    3    5    9   11    0
   1 C    PSEUD   -0.6824   -1.1357    0.0000    0    0    0    0    0
   2 O    PSEUD   -0.1723   -2.2550    0.0000    0    0    0    0    0
   3 N    PSEUD    0.0000    0.0000    0.0000    0    0    0    0    0
   4 Q1   PSEUD   -0.4226    0.9063    0.0000    0    0    0    0    0
   5 Q2   PSEUD    1.4530    0.0000    0.0000    0    0    0    0    0
   6 Q21  PSEUD    1.8202   -0.5343    0.8762    0    0    0    0    0
   7 Q22  PSEUD    1.8202   -0.5343   -0.8762    0    0    0    0    0
   8 Q3   PSEUD    1.8202   -0.5343    0.0000    0    0    0    0    0
   9 C    PSEUD    2.0013    1.4284    0.0000    0    0    0    0    0
  10 O    PSEUD    1.2356    2.3910    0.0000    0    0    0    0    0
  11 N    PSEUD    3.3231    1.5208    0.0000    0    0    0    0    0
```

For amino acid residues, the first three characters of the name correspond to the standard three letter code, the fourth character indicates positive or negative charges (e.g. ARG+) or differentiates between cysteine (CYS) and cystine (CYSS) residues. Cystines are involved in disulfide bridges.

ADE, CYT, GUA, and THY denote the standard deoxyribonucleotides of DNA; RADE, RCYT, RGUA, and URA denote the standard nucleotides of RNA.

Linker residues used to treat more than one molecule and containing only pseudo atoms are also included: PL to link an amino acid residue to a generic linker, NL to link a nucleotide residue to a generic linker, LL, a generic linker residue with 1 Å bond lengths and bond angles, LL2 and LL5, similar linkers residue but with 2 Å and 5 Å bond lengths, LP to link a generic linker to a following amino acid residue, and LN to link a generic linker to a following nucleotide residue. There is an additional set of linker residues, PLM, NLM, LLM, LLM2, LLM5, LPM and LNM, with three instead of one rotatable angle that should only be used for torsion angle dynamics calculations with inertia tensors derived directly from atomic masses and positions. LGLY is a linker residue with the geometry of GLY but only containing pseudo atoms.

In addition to the standard residue library (dyana.lib) which is based on the ECEPP/2 force field (Momany *et al.*, 1975; Némethy *et al.*, 1983), a residue library (amber.lib) that employs the standard geometry of the AMBER force field (Cornell *et al.*, 1995) is also provided. The names of atoms and dihedral angles are the same in both libraries.

The program DYANA supports an alternative format for residue library entries that uses atom names instead of numbers to define dihedral angles, covalent connectivities and pseudo atoms. This format is particularly useful in the process of creating manually a new or modified library entry. As an example, an alternative entry for SER is given which is equivalent to the one in the standard library:

```
RESIDUE   SER      5   15    3   14
   1 OMEGA   -1    2   10.0000 -O   -C    N    HN
   2 PHI      0    0    0.0000 -C    N    CA   C
   3 CHI1     1    3    1.3500  N    CA   CB   OG   HG
   4 CHI2     1    3    0.3000  CA   CB   OG   HG   HG
   5 PSI      0    0    0.0000  N    CA   C   +N
   1 C    C_BYL   -0.6824   -1.1357    0.0000 -O    N
   2 O    O_BYL   -0.1723   -2.2550    0.0000 -C
   3 N    N_AMI    0.0000    0.0000    0.0000 -C    HN   CA
   4 HN   H_AMI   -0.4226    0.9063    0.0000  N
   5 CA   C_ALI    1.4530    0.0000    0.0000  N    HA   CB   C
   6 HA   H_ALI    1.7416   -0.5122    0.9178  CA
   7 CB   C_ALI    2.0038   -0.7653   -1.2049  CA   HB2  HB3  OG
   8 HB2  H_ALI    1.6328   -1.7901   -1.1839  CB                   QB
   9 HB3  H_ALI    1.6328   -0.3109   -2.1235  CB                   QB
  10 QB   PSEUD    1.6328   -1.0505   -1.6537
  11 OG   O_HYD    3.4286   -0.7774   -1.2238  CB   HG
  12 HG   H_OXY    3.7558   -1.2840   -2.0214  OG
  13 C    C_BYL    1.9763    1.4377    0.0000  CA   O    +N
  14 O    O_BYL    1.1939    2.3868    0.0000  C
  15 N    N_AMI    3.2963    1.5532    0.0000  C
```

Names of atoms located in the preceding or next residue are preceded by "–" or "+", respectively, and positions that correspond to a "0" in the normal format are left blank.

The standard residue library also includes a statistical data base of chemical shift values in proteins, which was compiled by Daniel Braun on the basis of 26 proteins for which ^{1}H, ^{13}C and ^{15}N assignments are available. There are no random coil values in this table. The first few lines of this data base are as follows:

```
CSTABLE   320
   1 ALA  N      144  123.04    3.67  133.90  130.40  117.60  113.70
   3 ALA  CA     184   52.59    2.19   57.30   55.60   48.96   47.15
   4 ALA  CB     182   18.78    1.96   24.20   22.70   15.70   14.50
   5 ALA  HN     159    8.16    0.76   10.14    9.29    6.73    6.19
   6 ALA  HA     169    4.32    0.53    6.16    5.24    3.54    2.94
   7 ALA  QB     167    1.36    0.24    1.77    1.67    1.01   -0.02
```

The number after the heading "CSTABLE" denotes the number of atoms for which chemical shift information is available. For each such atom one line of data with the following entries is given: a running number, the residue name, the atom name, the number of chemical shifts that were available for this atom, the average chemical shift, the standard deviation of the chemical shift, the maximal chemical shift, the upper and lower 5%-quartiles of the chemical shift, and the minimal chemical shift. The Fortran format of the data lines is (5X,2A5,I5,6F8.2).

A similar block of data, named "KARPLUS", is used to define Karplus-type relationships of the form

$$^{3}J(\theta) = A + B\cos\theta + C\cos^{2}\theta \qquad [8]$$

between vicinal scalar couplings, ^{3}J, and the intervening dihedral angle, θ:

```
KARPLUS      15
   1 *    HN   HA      1.90   -1.40    6.40  PHI
   2 *    HN   C       0.10    1.10    4.00  PHI
   3 *    HN   CB     -0.20   -1.50    4.70  PHI
   4 *    C    C      -0.30   -0.80    2.00  PHI
   5 *    C    CB     -0.10   -0.60    1.50  PHI
   6 *    C    HA     -0.80   -4.40    9.00  PHI
   7 *    HA   N      -0.27   -0.61   -0.88  PHI
   8 *    HA   HB*     1.80   -1.60    9.50  CHI1
   9 PHE  HA   CG      0.70   -1.00    7.10  CHI1
  10 TYR  HA   CG      0.70   -1.00    7.10  CHI1
  11 *    HA   CG      0.20   -1.20   10.20  CHI1
  12 *    N    HB*     0.10    1.20   -4.40  CHI1
  13 *    C    HB*     0.60   -2.04    7.20  CHI1
  14 *    HB*  HG      1.80   -1.60    9.50  CHI2
  15 *    HB*  CD      0.20   -1.30   10.20  CHI2
```

Each Karplus curve is given on one line with the following data: a running number, a residue name that may contain wildcards, the two names of the atoms that are scalar coupled, the parameters, *A*, *B* and *C* of the

Karplus curve, and an optional comment (the corresponding dihedral angle). The Fortran format of the data lines is (5X,3A5,3F8.2).

Residue sequence

The sequence input file defines the primary structure of the molecule under consideration, identifies residues following cis-peptide bonds, declares special covalent bonds, i. e. covalent bonds that are not compatible with the tree structure of the molecule, and identifies fixed and rotatable dihedral angles in the molecule.

The residue names consist of up to four characters and must, of course, match the name of a residue entry in the residue library file. If a residue name is preceded immediately by a lowercase “c” the dihedral angle of the peptide bond preceding this residue will be fixed at (cis position) instead of (trans position) throughout the calculation. Optionally, a residue name may be followed by its residue number; by default the residue number of the first (N-terminal) residue in the sequence is set to one, and for other residues the residue number will be the residue number of the preceding residue plus one. Different residue names and numbers must be separated by at least one blank or end-of-line character, otherwise the format is free. The syntax to declare fixed and rotatable dihedral angles is explained below. An example sequence input file follows:

```
# Second helix of Antennapedia Homeodomain
ARG+ 29 ARG+ ARG+ ARG+ ILE  GLU- ILE  ALA  HIS  ALA
LEU
```

This file contains the sequence of a peptide that is 11 residues long with residue numbers 29–39. Trans-peptide bonds will be assumed throughout. No special covalent bond is declared. The first line of the file is a comment line.

A special covalent bond is declared by the lowercase keyword **link** followed by the first atom name, the first residue number, the second atom name, and the second residue number, in free format. This information will only be used in DYANA to exclude the necessary atom pairs from the steric overlap check; to correctly form the special covalent bond explicit upper and lower limit distance constraints are required. Situations where special covalent bonds are needed are, for instance: proteins with disulfide bridges, cyclic peptides, or flexible proline rings. To declare, for example, a disulfide bridge between CYSS 3 and CYSS 55 of a protein the following entry is used in the sequence input file:

```
link SG 3 SG 55
```

In addition, the presence of this disulfide bond is then fixed directly with distance constraints, e.g. by imposing a range of 2.0 to 2.1 Å on the S–S distance, and of 3.0 to 3.1 Å on the S–C distances across the bridge using explicit upper and lower distance limits (Williamson *et al.*, 1985). Because disulfide bonds occur frequently in proteins, it is not necessary (but possible, of course) to declare them explicitly in the sequence file; if the sulphur atoms of CYSS residues are not explicitly linked to other atoms by link entries in the sequence file, the program allows for special covalent bonds between all such sulphur atoms of CYSS residues, i. e. in the van der Waals check it potentially allows disulfide bridges between any two CYSS residues in the molecule. Another frequent case where the program generates a special covalent bond implicitly occurs if the bond is present in the list of covalent connectivities of the atom entries in the library but not compatible with the tree structure of dihedral angles. This is the case for example in flexible sugar rings of the DNA. Nevertheless, explicit upper and lower limit distance constraints are still required to enforce correct bond lengths and angles.

By default, the program DYANA assumes that all dihedral angles declared in the residue library are rotatable, i. e. are degrees of freedom during the minimization. The only exception are angles called OMEGA which are, by default, fixed at 180˚ or 0˚. To obtain different choices of fixed and rotatable dihedral angles, angle declarations have to be inserted into the sequence file. To make a dihedral angle rotatable, use the syntax: "*angle***=free**", where *angle* stands for the angle name. To fix a dihedral angle at the value of the input conformation that will be read, use "*angle***=fixed**". This type of declaration cannot be used if the start conformations are generated randomly within the program. To fix a dihedral angle at a given value, use "*angle***=***value*". The *value* has to be given in degrees. The angle name may contain wildcards * to match any number of characters and ? to match exactly one character. If an angle declaration should apply only to part of the sequence, the declaration and the corresponding part of the sequence are enclosed in curly braces. More than one angle declaration may follow the left brace, and parts of the sequence enclosed in braces may be nested. In the following example sequence all ω angles will be fixed at 180˚ (by default), all ψ angles except the third one will be fixed at -47˚, and all other dihedral angles are rotatable (by default):

```
{PSI=-47
 ARG+ 29 ARG+ {PSI=free ARG+}
 ARG+ ILE  GLU- ILE  ALA  HIS  ALA  LEU}
```

Chemical shift list

Chemical shift lists (traditionally called “proton lists”) follow the format used by the program XEASY (Bartels *et al.*, 1995). For each chemical shift, the list contains a line with the following data in free format: the atom number, the chemical shift, the error of the chemical shift (currently not used by DYANA), the atom name, and the residue number. The atom number is referenced by peak assignments in peak lists (see next section) and can be different from the atom number in coordinate files. Chemical shifts are measured in ppm and entries with a magnitude larger than 900 ppm are skipped. An example, containing proton, ^{15}N and ^{13}C chemical shifts follows:

```
622 119.770 0.000 N       3
 21   8.635 0.000 HN      3
 22  52.530 0.000 CA      3
 23   4.738 0.000 HA      3
 24  38.080 0.000 CB      3
 25   3.024 0.000 HB2     3
 26   2.956 0.000 HB3     3
623 109.960 0.000 ND2     3
 28   7.400 0.000 HD21    3
 29   6.660 0.000 HD22    3
624 121.460 0.000 N       4
 31   7.940 0.000 HN      4
 32  54.370 0.000 CA      4
 33   5.261 0.000 HA      4
 34  43.970 0.000 CB      4
 35   1.732 0.000 HB2     4
 36   1.454 0.000 HB3     4
 37  27.300 0.000 CG      4
 38   1.522 0.000 HG      4
 39   0.890 0.000 QD1     4
 40   0.870 0.000 QD2     4
```

Peak list

Peak lists follow the format used by the program XEASY (Bartels *et al.*, 1995). The program DYANA can handle two-dimensional homonuclear and three-dimensional heteronuclear peak lists. A peak list file starts with a line “# Number of dimensions *n*”, where *n* is either 2 or 3, possibly

followed by additional comment lines starting with "#". For each peak, there is a data line, possibly followed by a comment line that contains the user-defined comment for the given peak.

Each peak data line contains the following data: the peak number, *n* chemical shifts, the peak color code (integer), the spectrum type (a string; not used by DYANA), the peak volume, the error of the peak volume (not used by DYANA), the integration method code (a character), an integer (not used by DYANA), *n* atom numbers that identify atoms in the corresponding chemical shift list (a zero atom number indicates a missing assignment), and, possibly, additional data that is not used by DYANA. An example of a two-dimensional peak list is:

```
# Number of dimensions 2
   4   3.339  10.048 1 U  1.183e+05  0.00e+00 e   0   28   23
       # overlap
   5   2.791  10.048 1 U  3.090e+05  0.00e+00 e   0   27   23
       # transposed
   6   6.307   9.858 1 U  1.810e+05  0.00e+00 e   0   46   44
   7   3.179   9.858 1 U  3.506e+04  0.00e+00 e   0   49   44
   8   4.570   9.939 1 U  0.000e+00  0.00e+00 -   0   67   65
   9   4.361   9.939 1 U  2.429e+03  0.00e+00 e   0 2420   65
  10   1.226   9.939 1 U  1.793e+05  7.51e-01 d   0 2421   65
```

The first two peaks carry comments ("overlap" and "transposed", respectively). The integration method code is either "e" for peaks that have been integrated, or "-" for peaks that have not been integrated.

An example of a three-dimensional peak list is:

```
# Number of dimensions 3
  45  52.530   4.738   4.738 1 ? 1.903e+05  6.88e+03 a 0   22   23   23
  46  52.530   3.024   4.738 1 ? 1.842e+04  8.93e+02 a 0   22   25   23
  47  52.530   2.956   4.738 1 ? 3.620e+04  1.02e+03 a 0   22   26   23
  51  38.080   8.635   3.024 1 ? 6.100e+04  2.70e+02 a 0   24   21   25
  52  38.080   4.738   3.024 1 ? 1.872e+04  2.73e+02 a 0   24   23   25
  53  38.080   3.024   3.024 1 ? 2.776e+06  1.25e+04 a 0   24   25   25
  54  38.080   2.956   3.024 1 ? 2.922e+06  1.45e+04 a 0   24   26   25
  55  38.080   7.400   3.024 1 ? 1.155e+05  2.89e+02 a 0   24   28   25
  56  38.080   6.660   3.024 1 ? 2.673e+04  3.01e+02 a 0   24   29   25
```

Upper and lower distance limits

The upper and lower distance limit files are used to enter distance constraints into the program DYANA. For each distance constraint there is a line with the following data: residue number, residue name and atom name of the first and second atom, respectively, the distance limit in Å, and, optionally, the relative weight of the constraint. The default relative weight is 1. Relative weights should be positive. The weight of a constraint in the target function equals the relative weight times the weight-

ing factor for the corresponding type of constraints. An example file follows:

```
29 ARG+ HN     29 ARG+ HB2     2.90
29 ARG+ HN     29 ARG+ HB3     3.00
29 ARG+ HN     29 ARG+ QG      4.33
29 ARG+ HN     30 ARG+ HN      3.30
29 ARG+ HA     29 ARG+ QG      3.87
29 ARG+ HA     32 ARG+ HN      4.00
29 ARG+ QD     33 ILE  QD1     6.80
30 ARG+ HN     30 ARG+ QB      2.99
30 ARG+ HN     31 ARG+ HN      3.40
30 ARG+ HN     33 ILE  CB      9.10
30 ARG+ HA     30 ARG+ QB      2.72
30 ARG+ HA     30 ARG+ QD      5.80
30 ARG+ HA     33 ILE  HN      3.80  5.00E+00
```

In this example, the last constraint has a relative weight of 5, all others have the default relative weight of 1. If on an input line the first residue number and name are absent, the corresponding data from the previous data line is used. On the other hand, the first residue number and name may stand alone on a line such that the following is an equivalent form of the above example distance constraint file:

```
29 ARG+
        HN     29 ARG+ HB2     2.90
        HN     29 ARG+ HB3     3.00
        HN     29 ARG+ QG      4.33
        HN     30 ARG+ HN      3.30
        HA     29 ARG+ QG      3.87
        HA     32 ARG+ HN      4.00
        QD     33 ILE  QD1     6.80
30 ARG+
        HN     30 ARG+ QB      2.99
        HN     31 ARG+ HN      3.40
        HN     33 ILE  CB      9.10
        HA     30 ARG+ QB      2.72
        HA     30 ARG+ QD      5.80
        HA     33 ILE  HN      3.80  5.00E+00
```

Dihedral angle constraints

Dihedral angle constraint files contain direct constraints on individual

dihedral angles in the form of an allowed interval $[\phi_1, \phi_2]$ with $\phi_1 < \phi_2 < \phi_1 + 360^\circ$. This implies that the allowed interval must not degenerate to a point. A data line contains the residue number, the residue name, the dihedral angle name, the lower and upper bounds of the allowed interval in degrees, and, optionally, the relative weight of the constraint. The default relative weight is 1. Relative weights should be positive. The weight of a constraint in the target function equals the relative weight times the weighting factor for the corresponding type of constraints. See the following example file:

```
32 ARG+  PHI     -55.0   -35.0
32 ARG+  PSI     -75.0   -15.0 1.00E-01
32 ARG+  CHI1   -155.0  -125.0
33 ILE   PHI     -65.0   -35.0
33 ILE   PSI     -85.0   -15.0
33 ILE   CHI1   -105.0   -35.0
34 GLU-  PHI     -65.0   -45.0
34 GLU-  PSI     -85.0   -25.0
34 GLU-  CHI1     -5.0   125.0
```

In this example, the second constraint has a relative weight of 0.1, all others have the default relative weight of 1. As for distance constraint files, the residue number and name need not be repeated on each data line: if they are missing the corresponding data of the previous data line is assumed.

XPLOR distance and angle constraints

As an alternative to its native format, DYANA can also read distance and angle constraint files in XPLOR format (Brünger, 1992). Both types of constraints are specified with "assign" statements followed by two or four XPLOR atom selections, respectively, given in free format. Other statements in the input file are ignored. DYANA uses a simplified version of XPLOR atom selections that supports only the "resid" and "name" expressions. Other selection expressions and logical operators are skipped. The XPLOR wildcard "#" is converted to "*", and the XPLOR wildcards "%" and "+" are converted to "?". In contrast to other input files, comments are started with an exclamation mark. An example of an XPLOR distance constraint file is (only the part printed in bold is interpreted by DYANA):

```
set echo=false end
set wrnlev=0 end
assign (resid  1 and name HA  )(resid  1 and name HG* ) 0.00 0.00 3.96
assign (resid  1 and name HB* )(resid  1 and name HE  ) 0.00 0.00 5.50
assign (resid  3 and name HN  )(resid  3 and name HA  ) 0.00 0.00 2.80
assign (resid  3 and name HN  )(resid  3 and name HB* ) 0.00 0.00 3.60
assign (resid  4 and name HN  )(resid  4 and name HB2 ) 0.00 0.00 3.60
assign (resid  4 and name HN  )(resid  4 and name HB1 ) 0.00 0.00 2.80
assign (resid  4 and name HA  )(resid  4 and name HB2 ) 0.00 0.00 2.60
assign (resid  4 and name HA  )(resid  4 and name HB1 ) 0.00 0.00 2.70
assign (resid  5 and name HN  )(resid  5 and name HB2 ) 0.00 0.00 4.10
assign (resid  5 and name HN  )(resid  5 and name HB1 ) 0.00 0.00 4.10
assign (resid  5 and name HN  )(resid  5 and name HB* ) 0.00 0.00 3.36
assign (resid  6 and name HN  )(resid  6 and name HB* ) 0.00 0.00 3.60
assign (resid  6 and name HN  )(resid  6 and name HG  ) 0.00 0.00 3.10
assign (resid  6 and name HN  )(resid  6 and name HD1*) 0.00 0.00 5.30
assign (resid  6 and name HN  )(resid  6 and name HD2*) 0.00 0.00 5.50
! total constraints:      642
set echo=true end
set wrnlev=5 end
```

An atom selection must select either exactly one atom or a group of atoms that is represented in DYANA by a pseudo atom. Each "assign" statement can define an upper limit, $d + d_+$ and a lower limit $d - d_-$ for the corresponding distance, where d, d_-, and d_+ denote the three real numbers at the end of an "assign" statement. Upper limits with $d + d_+ \geq 900$ Å and lower limits with $d - d_- \leq 0.001$ Å are not considered.

An example of an XPLOR angle constraint file is (only the part printed in bold is interpreted by DYANA):

```
set message=off echo=off end
restraints dihedral reset
assign (resid  2 and name C )(resid  3 and name N )
       (resid  3 and name CA)(resid  3 and name C )
       1.00000    240.000    85.000  2
assign (resid  3 and name N )(resid  3 and name CA)
       (resid  3 and name C )(resid  4 and name N )
       1.00000    5.00000   100.000  2
assign (resid  3 and name C )(resid  4 and name N )
       (resid  4 and name CA)(resid  4 and name C )
       1.00000   -50.0000    15.000  2
assign (resid  4 and name N )(resid  4 and name CA)
       (resid  4 and name C )(resid  5 and name N )
       1.00000   -60.0000    45.000  2
assign (resid  4 and name N )(resid  4 and name CA)
       (resid  4 and name CB)(resid  4 and name CG)
       1.00000    60.0000    15.000  2
end
set message=on echo=on end
```

Each of the four atom selections for an angle constraint must match exactly one atom. In the above example the first "assign" statement constrains the ϕ dihedral angle of residue 3, the second constrains ψ of residue 3 etc. The allowed interval of a dihedral angle constraint is $[\phi - \Delta\phi, \phi + \Delta\phi]$ where ϕ and $\Delta\phi$ are the second and third real number in the "assign" statement, respectively.

Scalar coupling constants

Scalar coupling constant files specify values, and, optionally, tolerance ranges and weighting factors for vicinal scalar coupling constants:

```
1  ASP- HA   HB2   5.4
1  ASP- HA   HB3   4.1
2  GLU- HA   HB2  12.3   2.0   5.0
2  GLU- HA   HB3   4.1   2.0   5.0
2  GLU- HN   HA    5.1   1.0
3  CYSS HN   HA    6.2   1.0
```

Each line specifies, in this order, the following data: residue number, residue name, first atom name, second atom name, value, J, of the coupling constant (in Hertz), tolerance, ΔJ, of the coupling constant (default value: 2.0 Hz), and relative weight (default value: 1.0). The allowed interval of a coupling constant is $[J - \Delta J, J + \Delta J]$.

Orientation constraints

Orientation constraint files specify values of residual dipolar couplings. These are related to the orientation of the corresponding chemical bond according to Eq. [7]. An example file with constraints for the orientation of N–HN bonds:

```
 1 ARG+ HN      0.68
 3 ASP- HN      0.14
 4 PHE  HN      0.81
 5 CYSS HN      0.94   0.05
 6 LEU  HN      0.42   0.05
 7 GLU- HN      0.52   0.05  5.0
10 TYR  HN      0.95   0.05  5.0
```

Each line specifies, in this order, the following data: residue number, residue name, atom name, residual dipolar coupling value, the tolerance for the residual dipolar coupling value (default value: 0.1 Hz), and relative weight (default value: 1.0). The atom that is specified must have exactly one covalent bond.

Dihedral angles

Dihedral angle files are used by DYANA to store conformations in a more compact way than by storing Cartesian coordinates. The format used is: (I3,1X,A5,4(1X,A5,F9.3)) corresponding to the residue number and residue name, and up to four dihedral angle names and values (in degrees). As it is shown in the following example output dihedral angle file from DYANA, the residue number and name need not to be repeated on each data line if the line corresponds to the same residue as the previous one:

```
# Structure from DYANA, f = 2.50927E-01
 29 ARG+  PHI    -51.817 CHI1  -171.357 CHI2  -160.460 CHI3   -87.384
          CHI4    85.465 PSI    -56.588
 30 ARG+  PHI    -41.708 CHI1  -141.979 CHI2    71.718 CHI3    88.493
          CHI4    83.498 PSI    -50.974
 31 ARG+  PHI    -57.541 CHI1  -154.468 CHI2    72.871 CHI3  -174.205
          CHI4  -166.353 PSI    -68.960
 32 ARG+  PHI    -39.181 CHI1  -153.419 CHI2  -140.783 CHI3    56.936
          CHI4  -158.340 PSI    -37.495
 33 ILE   PHI    -64.011 CHI1   -81.197 CHI22   61.594 CHI21 -135.938
          CHI31   57.983 PSI    -50.837
 34 GLU-  PHI    -54.036 CHI1    98.207 CHI2  -172.555 CHI3     5.197
          PSI    -41.856
 35 ILE   PHI    -79.325 CHI1   -83.405 CHI22 -171.980 CHI21  -47.870
          CHI31 -145.152 PSI    -16.608
 36 ALA   PHI    -83.603 CHI1  -178.076 PSI    -21.708
 37 HIS   PHI   -113.998 CHI1   110.027 CHI2   120.049 PSI     -7.492
 38 ALA   PHI   -114.971 CHI1  -166.685 PSI    -12.535
 39 LEU   PHI   -124.389 CHI1  -134.399 CHI2    39.785 CHI31  151.866
          CHI32  -61.206  PSI   -41.470
```

Output dihedral angle files from DYANA start with a comment line that indicates the final value of its target function.

Cartesian coordinates

Cartesian coordinate files in DG format are used by DYANA for the input and output of conformations. The format of the data lines is: (6X,A5,I6,1X,A5,3F11.4) corresponding to the atom name, the residue number and name, and the x-, y- and z-coordinates of the atom in . For compatibility with other programs, the first three lines are always comment lines even if they do not start with “#”; further comment lines are not allowed. Optionally, the Cartesian atomic coordinates may be followed by the covalent connectivities, in this case the format is (6X,A5,I6,1X,A5,3F11.4,4I6).

On input Cartesian coordinates are only used to calculate all dihedral angles; the structure will be rebuilt in DYANA according to the standard ge-

ometry obtained from the residue library file. Therefore, conformations may be significantly changed if the Cartesian coordinates do not imply exactly the bond lengths, bond angles and chiralities of the standard geometry! The same applies for Cartesian coordinates where the dihedral angle of the peptide bonds are not exactly in the or conformation as defined in the sequence input file. An example output Cartesian coordinate file from DYANA follows:

```
Structure from DYANA, f = 2.50927E-01
DYANA 1.5 (sgi), 22-11-96
Number of residues:    11 Number of atoms:     240
    1 N        29 ARG+      1.3249     0.0000     0.0000
    2 HN       29 ARG+      1.8841     0.0000     0.8290
    3 CA       29 ARG+      2.0733     0.0000    -1.2455
    4 HA       29 ARG+      1.8760     0.9759    -1.6891
    5 CB       29 ARG+      3.5715    -0.1727    -0.9878
    6 HB2      29 ARG+      3.8976     0.5374    -0.2278
    7 HB3      29 ARG+      3.7640    -1.1709    -0.5948
    8 QB       29 ARG+      3.8308    -0.3167    -0.4113
    9 CG       29 ARG+      4.3767     0.0400    -2.2713
   10 HG2      29 ARG+      3.8459    -0.4008    -3.1151
   11 HG3      29 ARG+      4.4684     1.1070    -2.4747
   12 QG       29 ARG+      4.1572     0.3531    -2.7949
   13 CD       29 ARG+      5.7688    -0.5840    -2.1550
   14 HD2      29 ARG+      5.7220    -1.4866    -1.5457
   15 HD3      29 ARG+      6.1258    -0.8829    -3.1405
   16 QD       29 ARG+      5.9239    -1.1847    -2.3432
   17 NE       29 ARG+      6.7092     0.3855    -1.5500
   18 HE       29 ARG+      7.2613     0.9490    -2.1646
   19 CZ       29 ARG+      6.8670     0.5557    -0.2303
   20 NH1      29 ARG+      7.7438     1.4591     0.2287
   21 HH11     29 ARG+      7.8615     1.5862     1.2136
   22 HH12     29 ARG+      8.2804     2.0063    -0.4137
   23 QH1      29 ARG+      8.0710     1.7963     0.3999
   24 NH2      29 ARG+      6.1479    -0.1775     0.6305
   25 HH21     29 ARG+      6.2657    -0.0504     1.6154
   26 HH22     29 ARG+      5.4935    -0.8516     0.2880
   27 QH2      29 ARG+      5.8796    -0.4510     0.9517
   28 C        29 ARG+      1.5863    -1.1280    -2.1573
   29 O        29 ARG+      1.1807    -0.8822    -3.2923
```

Output Cartesian coordinate files from DYANA start with three comment lines that indicate the target function value, the program version used, and the number of residues and atoms listed in the coordinate file, respectively.

Optionally, the program DYANA can also output Cartesian coordinates in the format of the Protein Data Bank (Bernstein *et al.*, 1977).

COFIMA

The program COFIMA (coordinate file manipulation) is a versatile program to make simple manipulations on Cartesian coordinate, distance constraint and dihedral angle constraint files. The program works interactively and allows for a variety of commands. Some of the operations that can be performed with COFIMA are:

- Conversion between different data file formats
- Renaming of atoms, residues and dihedral angles
- Deletion of atoms, distance or angle constraints
- Listing of specific atoms, distance or angle constraints
- Measurement of distances and dihedral angles
- Attaching of atoms (e.g. hydrogens)
- Insertion of pseudo atoms or pseudo atom constraints
- Generation of covalent connectivities
- Sorting of atoms, distance or angle constraints

The program consists of three parts: COFIMA for coordinate file manipulations, DIFIMA for distance constraint file manipulations, and ANCOMA for angle constraints file manipulations. The prompts "`cofima>`", "`difima>`", and "`ancoma>`" indicate the part of the program that is currently active. Many commands can be used for all three types of data files, but there are also commands that are specific for certain types of data files. Commands can be abbreviated as long as the abbreviation remains unambiguous.

In the following description of the individual commands, A, A_1 etc. denote atom or angle names, R, R_1 etc. denote residue names, and r, r_1 etc. denote residue numbers. Atom, angle, and residue names must start with a letter and may (except in some cases) contain wildcards: "*****" stands for zero or more arbitrary characters, "**?**" stands for exactly one arbitrary character. No blanks are allowed within names. Residue numbers are integers. Atom, angle, residue names, and residue numbers may be preced-

ed by an exclamation mark "**!**" which acts as a "not operator." The special residue names **FIRST** and **LAST** can be used to denote the first and the last residue in the coordinate file, respectively. The special residue names **first** and **last** can be used to denote the first and last residue of every fragment with contiguous residue numbers in the coordinate file, respectively. Atom, angle, and residue names (but not command words) are case-sensitive.

Many commands allow for the specification of a residue range, denoted by *range*, which consists of one or more of the elements "*r*", "*r***..**", "**..***r*", "r_1**..**r_2" or "**@***R*" (separated by at least one blank).

12	Residue 12
12 20..25	Residues 12, 20, 21, 22, 23, 24, 25
@THR	All residues with name "THR"
20..25 @THR	All residues with name "THR" and numbers 20–25
!@CY*	All residues with names that do not start with "CY"

The default residue range that will be used if no residue range is specified includes all residues.

For many commands all selected atoms must be in the same residue. This convention can be circumvented by preceding certain atom names with a tilde "**~**". In this case, atoms are searched through the list of covalent connectivities When using "**~**", covalent connectivities must of course be present; either they can be read from a DG coordinate file or they can be generated using the **connect**, **bind** or **link** commands.

The output of those commands that give interesting output can be redirected to disk files. To do this, the last parameter on the command line must be "**>**[*file*]" (here and in the following, items given in brackets are optional) which writes the output to a new file, or "**>>**[*file*]" which appends the output to an existing file. Note that no space is allowed between the **>** sign and the output file specification. If the output file specification is omitted, the previously used output file is used.

Sequences of commands that are often used may be stored in macros (different from INCLAN macros) with file name extension ".cfm" in order to facilitate routine applications of the program.

Macros can be called from within a macro. When executing a command, the program can detect two different types of problems, warnings which cause only the current command to be skipped, and errors which cause the whole rest of the macro to be skipped. Macros can be commented; text between the comment sign **#** and the end of a line is considered as a comment. A set of standard macros is provided with the program:

am_di	Change from AMBER to DYANA nomenclature.
am_fm	Change from AMBER to FANTOM nomenclature.

am_op Change from AMBER to OPAL nomenclature.

attach_am Attach hydrogens to amino acids and DNA, AMBER conventions.

backbone Keep only backbone atoms N, CA, C.

di_am Change from DYANA to AMBER nomenclature.

di_fm Change from DYANA to FANTOM nomenclature.

di_op Change from DYANA to OPAL nomenclature.

di_xp Change from DYANA to XPLOR nomenclature.

fm_am Change from FANTOM to AMBER nomenclature.

fm_di Change from FANTOM to DYANA nomenclature.

fm_pdb Rename residue names and last atom from FANTOM to PDB.

fm_xp Change from FANTOM to XPLOR nomenclature.

heavy Keep only heavy atoms.

norm_residues Achieve standard three letter code for amino acid residues starting from AMBER, DYANA, FANTOM or other reasonable residue names.

op_am Change from OPAL to AMBER nomenclature.

op_di Change from OPAL to DYANA nomenclature.

plimits Change upper limit distance constraints from real to pseudo atoms; DYANA nomenclature.

pseudo Insert pseudo atoms, DYANA nomenclature.

sort Sort atoms in amino acid residues.

If a command should only be applied to a certain type of data files, the command word may be followed (with no intervening spaces) by the qualifiers **/cofima** (to apply the command only to Cartesian coordinates), **/difima** (to apply the command only to distance constraints), **/ancoma** (to apply the command only to angle constraints), **/!cofima** (to not apply the command to Cartesian coordinates), **/!difima** (to not apply the command to distance constraints), or **/!ancoma** (to not apply the command to angle constraints).

The following, alphabetically ordered list of commands includes all commands that can be used for coordinate, distance constraint, and angle constraint files.

angles

[~]A_1 A_2 [~]A_3 [~]A_4 [[~]A_5] [*range*]

List bond angles, dihedral angles, or relative dihedral angles. This command can only be used with coordinate files. If three atom names are giv-

en, the bond angle A_1–A_2–A_3 is calculated. If four atom names are given, the dihedral angle A_1–A_2–A_3–A_4 is calculated. If five atom names are given, the difference between the dihedral angle A_1–A_2–A_3–A_4 and the dihedral angle A_1–A_2–A_3–A_5 is calculated.

For instance, the dihedral angles in a polypeptide can be calculated with the following command:

`angles CA C ~N ~CA` Calculate ω dihedral angles

ancoma

Switch to ANCOMA, the part of the program for the manipulation of angle constraint files.

attach

A [~]A_1 A_2 A_3 [[~]A_4] *b* τ θ [*range*]

Attach atoms to a structure. This command can only be used with coordinate files. The atom *A* is attached to the atom A_3 such that the bond length A_3–*A* equals *b*, the bond angle A_2–A_3–*A* equals τ, and the dihedral angle A_1–A_2–A_3–*A* (if A_4 is omitted) or the difference between the dihedral angles A_1–A_2–A_3–*A* and A_1–A_2–A_3–A_4 (if A_4 is present) equals θ (in this case it is not important which atom is specified by A_1). Note that *b*, τ and θ must be given as real numbers with a period to avoid confusion with the following *range* specification.

Normally, all atoms must be in the same residue. This convention can be circumvented by preceding atom names with a tilde "**~**". In this case, atoms are searched through the list of covalent connectivities which allows to use the **attach** command also if not all atoms lie within one residue.

`attach HB N CA CB OG1 1.09 110.9 123.0 @THR`
Attach the β-proton HB of threonine if the heavy atom positions are known.

bind

A_1 r_1 A_2 r_2

Insert a specific covalent connectivity between the atom A_1 of residue r_1 and the atom A_2 of residue r_2. This command can only be used with coordinate files.

break

A_1 r_1 A_2 r_2

Remove a specific covalent connectivity between the atom A_1 of residue r_1 and the atom A_2 of residue r_2. This command can only be used with coordinate files.

change

$A_1 \ldots A_2$ [*range*] @*R*|*r*|=*r*|+*r*|-*r*

Change residue names or residue numbers. If the last parameter is @*R*, the residue names of the specified atoms are set to *R*. If the last parameter is *r* or =*r*, the residue numbers of the specified atoms are set to *r*. If the last parameter is +*r* or -*r*, the residue numbers of the specified atoms are incremented or decremented by *r*.

cofima

Switch to COFIMA, the part of the program for the manipulation of Cartesian coordinate files.

connect

[A_1=$b_1 \ldots A_2$=b_2]

Generate covalent connectivities on the basis of bond length criteria. This command can only be used with coordinate files. Incorrect results may occur if there are large steric overlaps. The command without any parameters is equivalent to the following command:

```
connect H*=0.4 C*=0.85 N*=0.8 O*=0.7 S*=1.3
        P*=1.2 Q*=-999 LP*=-999 *=0.85
```

Usually, the command can be used with these default parameters. Covalent connectivities are generated for those atom pairs with the interatomic distance smaller than the sum of the two bond radii. The bond radius of an atom is given by b_i Å if A_i is the leftmost atom type on the command line that matches the atom name. Covalent connectivities are only generated between atoms that are in the same or in sequentially neighboring residues. To generate other connectivities, the commands **bind** and **link** can be used.

constraints

$A_1 \ldots A_2$ [*range*]

List the distance or angle constraints involving the specified atoms or angles. This command can only be used for distance constraint or angle constraint files.

coordinates

$A_1 \ldots A_2$ [*range*]

List the atom names, residue names and numbers, Cartesian coordinates, and, if present, covalent connectivities of the specified atoms. This com-

mand can only be used with coordinate files.

copy

A_1 r_1 [A_2] r_2 [A_3]

Copy the atom A_1 of residue r_1, i. e. its Cartesian coordinates, to atom A_2 of residue r_2. This command can only be used with coordinate files. It adds a new atom to residue r_2. If A_2 is omitted, the name of the new atom will be A_1. If A_3 is given, the new atom will be inserted after A_3 in r_2, otherwise as the last atom of the residue.

delete

$A_1 \ldots A_2$ [*range*]

Delete the specified atoms or constraints. When working with Cartesian coordinate files, all atoms whose name matches one of the atom specifications on the command line are deleted. When working with distance constraints, all distance constraints for which one or both atom names match an atom specification on the command line are deleted. When working with angle constraints, all constraints for angles whose name matches one of the angle specifications on the command line are deleted.

difima

Switch to DIFIMA, the part of the program for the manipulation of distance constraint files.

directory

[*macro*]

Give a directory of all standard macro files and all macro files in the current working directory. If a *macro* specification is given, the directory will only contain those macro files with names that match the given *macro* specification. A *macro* specification is a macro file name, possibly containing wildcard characters, but excluding the extension ".cfm". For every macro, its name and the comment lines that precede the first command line are listed.

disconnect

$A_1 \ldots A_2$ [*range*]

Remove the covalent connectivities of the specified atoms. This command can only be used with coordinate files. The default is to remove all covalent connectivities.

distances

A_1 [$range_1$] A_2 [$range_2$] [*condition*]

When working with Cartesian coordinates, calculates the distances between atoms specified by A_1 $range_1$ and atoms specified by A_2 $range_2$. When working with distance constraints, list constraints for distances between atoms specified by A_1 $range_1$ and atoms specified by A_2 $range_2$. Optionally, only distances or constraints that fulfill one or several of the following *conditions* are listed:

d<*value*	distance less than *value*
d>*value*	distance greater than *value*
r<*value*	residue number difference less than *value*
r>*value*	residue number difference greater than *value*
r=*value*	residue number difference equal to *value*

Note that no spaces are allowed within a *condition*. For example, the command

```
distances HB% HN r=1 d<5
```

lists all sequential distances shorter than 5 Å between β and amide protons. The command cannot be used for angle constraints. end Terminate the program.

extract

A_1 [$range_1$] A_2 [$range_2$] [*condition*] [*limit*]

Extract constraints for the distances between atoms specified by A_1 $range_1$ and atoms specified by A_2 $range_2$. The extracted distance constraints are appended to the current list of distance constraints. Optionally, only constraints that fulfill one or several *conditions* are extracted. The format of a *condition* is the same as for the command **distances**. The distance limit is set according to an optional *limit* specification. It is possible to set the distance limit to the actual distance plus an offset by using the expression **l=***offset*, or to set the distance limit to the smallest possible value from a list of limits by using the expression **l<**l_1, l_2,..., l_n, or to set the distance limit to the largest possible value from a list of limits by using the expression **l>**l_1, l_2,..., l_n. Note that no spaces are allowed within these *limit* expressions. This command can only be used with coordinate files.

help

Display help information. Instead of help a question mark "**?**" may be used.

insert

A $A_1 \ldots A_2$ [*range*]

Insert pseudo atoms. This command can only be used with coordinate files. The command inserts a new atom with the name *A* in the centre of the atoms A_1,..., A_2. For example, the command

```
insert QB HB%
```

inserts a pseudo atom QB in the centre of the β-protons.

keep

$A_1 \ldots A_2$ [*range*]

Keep only those atoms, distance constraints, or angle constraints that match the specification given on the command line. As an example,

```
keep N CA C
```

deletes all atoms except the backbone atoms N, CA, and C' in amino acid residues. When working with distance constraints, distance constraints with one or both atom names matching an atom specification on the command line are kept.

link

[*b*] A_1 [$range_1$] A_2 [$range_2$]

Generates covalent connectivities between atoms called A_1 in the residue range $range_1$ and atoms called A_2 in the residue range $range_2$ if they are less than *b* Å apart. The default for bond length is *b* = 2.5 Å. This command can only be used with coordinate files. For example, the command

```
link SG SG
```

inserts covalent connectivities between atoms called SG which are less than 2.5 Å apart from each other, and can thus be used to generate the connectivities that correspond to disulphide bridges.

list

[*range*]

Gives a summary listing of the atoms, distance constraints or angle constraints in the given residue *range* (by default including all residues). The number of atoms, distance constraints or angle constraints, the number of residues, and lists of the occuring atom, angle, and residue names are given.

pseudo

$A\ A_1 \ldots A_2\ [A_3{=}c_3 \ldots]$ ***=**c_4 [*range*]

Modifies distance constraints from real to pseudo atoms (Wüthrich *et al.*, 1983). This command can only be used with distance constraints files that contain upper distance bounds. Distance constraints involving atoms that match one of the atom specifications A_1,..., A_2 are changed in order to refer to the pseudo atom A, and the upper distance bound is increased by a correction. Usually, this correction is given by c_4 Å; if the distance constraint is an intraresidual constraint that involves one of the atoms A_3,... the specific correction given for this atom is used.

quit

Terminate the program.

read

file

Read an input *file* with Cartesian coordinates, distance constraints, or angle constraints. The program determines automatically which format the input data file has. The allowed formats for Cartesian coordinate files are:

DG	The format used by DYANA
PDB	The format used by the Brookhaven Protein Data Bank (Bernstein *et al.*, 1977) with some restrictions.
AMBER	The format used by the molecular dynamics program AMBER (Singh *et al.*, 1986; very similar to PDB).

Distance constraints and angle constraints are read in the format used by DYANA.

remove

A_1 [$range_1$] A_2 [$range_2$] [*condition*]

Remove constraints for distances between atoms specified by A_1 $range_1$ and atoms specified by A_2 $range_2$. Optionally, only constraints that fulfill one or several *conditions* are removed. The format of a *condition* is the same as for the command **distances**. The command can only be used for distance constraints.

rename

$A_1\ A_2$ [*range*]

Change the name of atoms or angles A_1 into A_2. As an example, the three commands

```
rename HB2 XXX
rename HB3 HB2
rename XXX HB3
```

exchange the names of the atoms HB2 and HB3.

retain

A_1 [*range*$_1$] A_2 [*range*$_2$] [*condition*]

Retain only constraints for distances between atoms specified A_1 *range*$_1$ and atoms specified by A_2 *range*$_2$. Optionally, only constraints that fulfill one or several *conditions* are retained. The format of a *condition* is the same as for the command **distances**. The command can only be used for distance constraints.

save

Write an output Cartesian coordinate, distance constraint, or angle constraint file with the same name and the same format as the input file from which the data was read.

sort

[$A_1 \ldots A_2$ ***** $A_3 \ldots A_4$] [*range*]

Sort atoms, distance constraints, or angle constraints. If there are no atom or angle specifications on the command line, the items are sorted according to a default order. Otherwise, the items are sorted into the order given by the atom or angle specifications on the command line. The asterisk "*****" represents all atoms or angles which are not explicitly given.

type

[*macro*]

List the contents of the macro file(s) that match the given *macro* specification. A *macro* specification is a macro file name, possibly containing wildcard characters, but excluding the extension ".cfm".

writeaco

file

Write an angle constraint output file in the format used by the program DYANA (see above). This command can only be used with angle constraint files.

writeamber *file*

Write a Cartesian coordinate output file in AMBER format, the format used by the molecular dynamics program AMBER. This command can only be used with coordinate files.

writedco *file*

Write a distance constraint output file in the format used by the program DYANA. The residue name and number of the first atom of the constraints are not repeated if they are the same as for the previous constraint. This command can only be used with distance constraint files.

writedg *file*

Write a Cartesian coordinate output file in DG format, the format used, for example, by the program DYANA. This command can only be used with coordinate files.

writelongdco *file*

Write a distance constraint output file in the format used by the program DYANA. The residue name and number of the first atom of all constraints are written out. This command can only be used with distance constraint files.

writepdb *file*

Write a Cartesian coordinate output file in PDB format, the format used by the Brookhaven Protein Data Bank. This command can only be used with coordinate files.

@macro Execute a *macro*, i. e. a file containing COFIMA commands. A *macro* specification is the file specification of the macro file excluding the extension ".cfm".

!*string* Repeat the last command that started with *string*. The *string* must not

contain spaces.

Installation

The program is delivered as a tar-file (**dyana-1.5.tar**), possibly gzip-compressed (**dyana-1.5.tar.gz**) or compressed (**dyana-1.5.tar.Z**). To uncompress, the approriate commands are:

gunzip dyana-1.5.tar.gz
or
uncompress dyana-1.5.tar.Z

Unpacking with the command

tar xf dyana-1.5.tar

creates in the current directory a subdirectory called **dyana-1.5** that contains all files of the program package. The different subdirectories contain the following data:

dyana	DYANA source files
inclan	INCLAN source and help files
macro	DYANA standard macros
lib	residue libraries
help	on-line help files
example	example files used in the tutorial
cofima	COFIMA source, help, and macro files
scripts	installation scripts

The program is configured for a particular computer system by the shell script **configure** using the UNIX command

./configure [*options*] [*install-directory*]

that automatically recognizes many UNIX computer systems. The optional parameter *install-directory* denotes the directory where libraries, macros, and on-line help files will be installed. The default *install-directory* is $HOME/lib where $HOME is the value of the corresponding UNIX environment variable. The *install-directory* should be different from the directory in which the tar file was unpacked. *Options* include:

-d	Use double precision (64 bit) for real numbers (default).
-f	Use the Fortran 90 compiler (instead of Fortran 77).
-g	Prepare executables for debugging.
-h	Print a summary of these options.
-q	"Quick". Compile without optimization.
-s	Use single precision (32 bit) for real numbers.
-t *type*	Configure for a given computer *type*. Possible *types* are listed in the file "scripts/identify" which is used to determine automatically the type of the current computer system.

The script **configure** assumes that the name of the directory where the program resides is of the form **dyana-***version*. All parameters set by the configuration script are listed and stored in the file "make.config". Execution of the **configure** script has no other effect than creating the file "make.config".

The program package is then built by the UNIX command

make

and installed in the directory *install-directory***/dyana-1.5** by

make install

The directory in which the tar file was unpacked can be removed after this step.

Executable shell scripts to start the programs will be created and installed in the directory *bin-directory* by

cd *install-directory***/dyana-1.5**
./setup [*bin-directory*]

The default *bin-directory* is $HOME/bin where $HOME is the value of the corresponding UNIX environment variable.

To have easy access to the program DYANA, the *bin-directory* should be included in the search path of the UNIX shell. If DYANA was correctly installed and the directory containing the executables is included in the UNIX search path, then the programs can be started simply by typing their name.

If the program DYANA can be started but does not display the prompt (“dyana>”), the installation is not correct and should be repeated.

To remove all files created by **make**, the command

```
make clean
```

may be used. After **make clean** it will be necessary to run the configuration script again. After copying the program from one computer to another, it is important to perform **make clean** before running the configuration script. Similarly,

```
make uninstall
```

removes all files created by **make install**, and

```
make recompile
```

completely recompiles the programs but preserves executables for other computer types that are already present.

References

Bartels, C., Xia, T., Güntert, P., Billeter, M. & Wüthrich, K. (1995). The program XEASY for computer-supported NMR spectral analysis. *J. Biomol. NMR* **5**, 1–10.

Berendsen, J. H. C., Postma, J. P. M., van Gunsteren, W. F., DiNola, A. and Haak, J. R. (1984). Molecular dynamics with coupling to an external bath. *J. Chem. Phys.* **81**, 3684–3690.

Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Meyer, E. F., Jr., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T., Tasumi, M. (1977). The Protein Data Bank: A computer-based archival file for macromolecular structures. *J. Mol. Biol.* **112,** 535–542.

Braun, W. & Gō, N. (1985). Calculation of protein conformations by proton-proton distance constraints. A new efficient algorithm. *J. Mol. Biol.* **186,** 611–626.

Braun, W., Bösch, C., Brown, L. R., Gō, N. & Wüthrich, K. (1981). Combined use of proton-proton overhauser enhancements and a distance geometry algorithm for determination of polypeptide conformations. application to micelle-bound glucagon. *Biochim. Biophys. Acta*, **667**, 377–396.

Brünger, A. T. (1992). *X-PLOR version 3.1. A system for X-ray crystallography and NMR*, Yale University Press, New Haven.

Bystrov, V. F. (1976). Spin-spin coupling and the conformational states of peptide systems. *Prog. NMR. Spectrosc.* **10**, 41–81.

Cornell, W. D., Cieplak, P., Bayly, C. I., Gould, I. R., Merz Jr., K. M., Ferguson, D. M., Dpellmeyer, D. C., Xox, T., Caldwell, J. W. & Kollman, P. A. (1995). A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *J. Am. Chem. Soc.* **117**, 5179–5197.

DeMarco, A., Llinás, M. and Wüthrich, K. (1978a) Analysis of the ^{1}H-NMR spectra of ferrichrome peptides. I. The non-amide protons. *Biopolymers*, **17,** 617–636.

DeMarco, A., Llinás, M. and Wüthrich, K. (1978b) ^{1}H-^{15}N spin-spin couplings in alumichrome. *Biopolymers*, **17,** 2727–2742.

Eccles, C., Güntert, P., Billeter, M. & Wüthrich, K. (1991). Efficient analysis of protein 2D NMR spectra using the software package EASY. *J. Biomol. NMR*, **1,** 111–130.

Ernst, R. R., Bodenhausen, G. & Wokaun, A. (1987). *The principles of nuclear magnetic resonance in one and two dimensions,* Clarendon, Oxford.

Güntert, P. (1998). Structure calculation of biological macromolecules from NMR data. *Q. Rev. Biophys.*, in press.

Güntert, P., & Wüthrich, K. (1991). Improved efficiency of protein structure calculations from NMR data using the program DIANA with redundant dihedral angle constraints. *J. Biomol. NMR.*, **1,** 447–456.

Güntert, P., Berndt, K D & Wüthrich, K. (1993). The program ASNO for computer-supported collection of NOE upper distance constraints as input for protein structure determination. *J. Biomol. NMR* **3,** 601–606.

Güntert, P., Braun, W., Billeter, M. & Wüthrich, K. (1989). Automated stereospecific ^{1}H NMR assignments and their impact on the precision of protein structure determinations in solution. *J. Amer. Chem. Soc.* **111,** 3997–4004.

Güntert, P., Braun, W. & Wüthrich, K. (1991a). Efficient computation of three-dimensional protein structures in solution from nuclear magnetic resonance data using the program DIANA and the supporting programs CALIBA, HABAS and GLOMSA. *J. Mol. Biol*. **217,** 517–530.

Güntert, P., Qian, Y. Q., Otting, G., Müller, M., Gehring, W. J. & Wüthrich K. (1991b). Structure determination of the *Antp(C39→S)* homeodomain from nuclear magnetic resonance data in solution using a novel strategy for the structure calculation with the programs DIANA, CALIBA, HABAS and GLOMSA. *J. Mol. Biol.* **217,** 531–540.

Güntert, P., Mumenthaler, C. & Wüthrich, K. (1997). Torsion angle dynamics for NMR structure calculation with the new program DYANA. *J. Mol. Biol.*, **273**, 283–298.

Güntert, P., Billeter, M., Ohlenschläger, O., Brown, L. R. & Wüthrich, K. (1998). Conformational analysisof protein and nucleic acid fragments with the new grid search algorithm FOUND. *J. Biomol. NMR*, in press.

Jain, A., Vaidehi, N. & Rodriguez, G. (1993). A fast recursive algorithm for molecular dynamics simulation. *J. Comp. Phys.* **106,** 258–268.

Kießling, I. & Lowes, M. (1987). *Programmierung mit FORTRAN-77*, Teubner, Stuttgart.

Koradi, R., Billeter, M. & Wüthrich, K. (1996). MOLMOL: A program for display and analysis of macromolecular structures. *J. Mol. Graph.* **14**, 51–55.

Laskowski R A, MacArthur M W, Moss D S & Thornton J M. (1993). PROCHECK: A program to check the stereochemical quality of protein structures. *J. Appl. Cryst.*, **26,** 283–291.

Luginühl, P., Szyperski, T. & Wüthrich, K. (1995). Statistical basis for the use of $^{13}C^{\alpha}$ chemical shifts in protein structure determination. *J. Magn. Reson. B*, **109,** 229–233.

Mazur, A. K., Dorofeev, V. E. & Abagyan, R. A. (1991). Derivation and testing of explicit equations of motion for polymers described by internal coordinates. *J. Comp. Phys.* **92**, 261–272.

Mathiowetz, A. M., Jain, A., Karasawa, N. & Goddard III, W. A. (1994). Protein simulations using techniques suitable for very large systems: The cell multipole method for nonbond interactions and the Newton-Euler inverse mass operator method for internal coordinate dynamics. *Proteins* **20**, 227–224.

McLachlan, A. D. (1979). Gene duplication in the structural evolution of chymotrypsin. *J. Mol. Biol.* **128,** 49–79.

Momany, F. A., McGuire, R. F., Burgess, A. W. & Scheraga, H. A. (1975). Energy parameters in polypeptides. VII. Geometric parameters, partial atomic charges, nonbonded interactions, hydrogen bond interactions, and intrinsic torsional potentials for the naturally occuring amino acids. *J. Phys. Chem.* **79,** 2361–2381.

Mumenthaler, C. & Braun, W. (1995). Automated assignment of simulated and experimental NOESY spectra of proteins by feedback filtering and self-correcting distance geometry. *J. Mol. Biol.* **254**, 465–480.

Mumenthaler, C., Güntert, P., Braun, W. & Wüthrich, K. (1997). Automated combined assignment of NOESY spectra and three-dimensional protein structure determination. *J. Biomol. NMR* **10**, 351–362.

Nagayama, K. & Wüthrich, K. (1981). Structural interpretation of vincinal proton-proton coupling constants $^3J_{\alpha\beta}$ in the basic pancreatic trypsin inhibitor measured by two-dimensional J-resolved NMR spectroscopy. *Eur. J. Biochem.* **115**, 653–657.

Némethy, G., Pottle, M. S. & Scheraga, H. A. (1983). Energy parameters in polypeptides. 9. Updating of geometrical parameters, nonbonded interactions, and hydrogen bond interactions for the naturally occuring amino acids. *J. Phys. Chem.* **87,** 1883–1887.

Ottiger, M., Szyperski, T., Luginbühl, L., Ortenzi, C., Luporini, P., Bradshaw, R. A. & Wüthrich, K. (1994). The NMR solution structure of the pheromone Er-2 from the ciliated protozoan *Euplotes raikovi*. *Protein Science* **3**, 1515–1526.

Pardi, A., Billeter, M. and Wüthrich, K. (1984) Calibration of the angular dependence of the amide proton–C^α proton coupling constants, $^3J_{HN\alpha}$, in a globular protein. Use of $^3J_{HN\alpha}$ for identification of helical secondary structure. *J. Mol. Biol.* **180,** 741–751.

Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T. (1986). *Numerical Recipes. The art of scientific computing*, Cambridge University Press, Cambridge.

Singh, U. C., Weiner, P. K., Caldwell, J. W. & Kollman, P. A. (1986). AMBER 3.0, University of California at San Francisco.

Spera, S. & Bax, A. (1991) Empirical correlation between protein backbone conformation and C^α and C^β ^{13}C nuclear magnetic resonance chemical shifts. *J. Amer. Chem. Soc.* **113**, 5490–5492.

Stein, E. G., Rice, L. M. & Brünger, A. T. (1997). Torsion-angle molecular dynamics as a new efficient tool for NMR structure calculation. *J. Magn. Reson.* **124**, 154–164.

Williamson, M. P., Havel, T. F. & Wüthrich, K. (1985). Solution conformation of proteinase inhibitor IIa from bull seminal plasma by ^1H nuclear magnetic resonance and distance geometry. *J. Mol. Biol.* **182,** 295–315.

Wüthrich, K. (1986). *NMR of Proteins and Nucleic Acids*, Wiley, New York.

Wüthrich, K., Billeter, M. & Braun, W. (1983). Pseudo-structures for the 20 common amino acids for use in studies of protein conformations by measurements of intramolecular proton-proton distance constraints with nuclear magnetic resonance. *J. Mol. Biol.* **169**, 949–961.

Index

A

B

C

D

F

G

H

I

K

L

M

N

O

P

R

S

T

V

W