

# ARIA 1.2 Manual

Jens Linge  
Michael Nilges

Institut Pasteur  
Unité de Bioinformatique Structurale  
25 rue du Dr Roux  
F-75015 Paris  
France

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Installation</b>	<b>7</b>
2.1	Compilation of the CNS program . . . . .	8
2.2	Installation of Python . . . . .	8
2.3	Downloading ARIA . . . . .	9
2.4	Installing the cgi scripts on your server . . . . .	9
2.5	editing your shell rc files . . . . .	10
2.6	Setting up ssh to run ARIA on several machines . . . . .	11
2.7	WHATCHECK, PROCHECK, PROSA . . . . .	13
<b>3</b>	<b>Tutorial</b>	<b>14</b>
3.1	Introduction . . . . .	14
3.2	Getting started . . . . .	15
3.2.1	Setting up a new project . . . . .	15
3.2.2	Start ARIA within a UNIX shell . . . . .	15
3.2.3	Editing the parameters in run.cns . . . . .	16
3.2.4	Start ARIA within a UNIX shell . . . . .	16
3.3	Input files . . . . .	16
3.4	Output files . . . . .	18
<b>4</b>	<b>Data format</b>	<b>19</b>
4.1	ARIA NOE peak files (.list suffix) . . . . .	19
4.2	CNS files (.tbl suffix) . . . . .	23
4.3	BioMagResBank support . . . . .	25
4.4	XML support . . . . .	25
4.5	Python modules for manipulating the data . . . . .	25

<b>5</b>	<b>Description of ARIA protocols</b>	<b>30</b>
5.1	Overview . . . . .	30
5.2	setup of run directory tree . . . . .	31
5.3	run.cns . . . . .	32
5.4	generate.inp . . . . .	34
5.5	generate_template.inp . . . . .	35
5.6	stereoassign.cns . . . . .	35
5.7	Sorting of structures . . . . .	37
5.8	calib.inp . . . . .	38
5.9	merge.inp . . . . .	45
<b>6</b>	<b>Analysis protocols</b>	<b>46</b>
6.1	Overview . . . . .	46
6.2	PDB header . . . . .	47
6.3	calib.inp . . . . .	47
6.4	print_noe.inp . . . . .	48
6.5	print_dih.inp . . . . .	48
6.6	print_coup.inp . . . . .	49
6.7	print_geom.inp . . . . .	49
6.8	cop.inp . . . . .	50
6.9	rmsave.inp . . . . .	50
6.10	wellordered.inp . . . . .	51
<b>7</b>	<b>Appendix – CNS scripting language</b>	<b>53</b>
7.1	General features . . . . .	53
7.2	Symbol definitions and the EVALuate statement . . . . .	54
7.3	Parameter definition with DEFIne . . . . .	55
7.4	Arithmetic operations . . . . .	56
7.5	Special symbols . . . . .	56
7.6	IF statements . . . . .	57
7.7	Loops . . . . .	58
7.8	Atom selection . . . . .	58
7.9	Selection by atom name . . . . .	59
7.10	Wildcards and ranges . . . . .	59
7.11	Selection by atom property . . . . .	60
7.12	BYREsidue and BYGRp . . . . .	60
7.13	STOREi and RECALLi . . . . .	61
7.14	Do, show and identity statements . . . . .	61

7.15 3-D vectors and matrices . . . . .	61
7.16 Output files . . . . .	62
7.17 modules . . . . .	63
7.18 Examples . . . . .	63

# Chapter 1

## Introduction

In this manual, we are focussing on the practical aspects of automated NOE assignment and NMR structure calculation with ARIA 1.2 (Ambiguous Restraints for Iterative Assignment). For a detailed introduction of the concept of ambiguous distance restraints (ADRs), Cartesian and torsion angle dynamics simulated annealing refinement with ADRs, multimer calculations, force fields for NMR structure determination, refinement of the final structures in a shell of explicit water and the spin diffusion correction in ARIA, we refer to the literature [1, 2, 3, 4, 5, 6, 7].

First, we will explain how to install ARIA (compilation of the CNS executable, Python, setup of the cgi scripts for the HTML interface etc., see chapter 2).

A quick tutorial (chapter 3) will show the beginners how a typical ARIA project proceeds. We will use the data of the werner HRDC domain [8, 9].

The data formats for the input and output files are described in chapter 4. We describe shortly how ARIA converts ANSIG, NMRView, PIPP and XEASY data. We also explain shortly the BioMagResBank and XML files (NOE peak lists and chemical shift lists) which are generated after the last iteration.

In chapter 5 we explain some of the ARIA CNS protocols together with a more thorough description of the parameters the user can specify before starting the calculations (the file `run.cns` which can be edited via the HTML interface).

In chapter 6 we will shortly describe some of the CNS analysis protocols. These protocols are used after the final iteration of ARIA. The Appendix in chapter 7 explains some of the features of the CNS scripting language. This chapter is interesting for advanced users who want to change or extend the program.

# Chapter 2

## Installation

ARIA 1.2 was tested on several architectures, most importantly on SGI and linux machines. We changed some of the CNS protocols to circumvent a problem in the original CNS code which caused some trouble when running the ARIA 1.0 version with a g77-compiled cns-executable.

Please have a look at the Nilges group homepage for updates and more information on the installation:

<http://www.pasteur.fr/recherche/unites/Binfs/>

ARIA consists of four main parts:

1. **Python scripts** to convert the data from different file formats, create the directory tree, start the CNS protocols and control the overall program flow. You need to install a Python interpreter together with the Numerical Python module in order to use ARIA. All the Python files can be found in the *aria1.2/Aria* subdirectory (Aria is a Python package).

2. **Perl cgi scripts** for saving the data from the browser interface (mainly using scripts from the CNS distribution). There scripts have to be moved from *aria1.2/cgi* to the cgi-bin directory of your www server. Then you can access them from within the ARIA html setup. You need a Perl executable running on your www server.

Alternatively, you can use the cgi scripts which are installed at the Institut

Pasteur in Paris.

3. **CNS protocols** for the calibration, merging, Cartesian or Torsion MD, waterrefinement, analysis protocols, etc. These protocols can be edited for your purposes. They are designed for the automated approach within ARIA. If you want to use them as stand-alone CNS protocols, you have to edit them slightly. All the CNS protocols can be found in *aria1.2/protocols* subdirectory. They are copied to each run of your project automatically.

4. The **CNS executable** (compiled from the CNS 1.2 and ARIA 1.2 fortran source code).

## 2.1 Compilation of the CNS program

- Academic users can download `cns 1.1` from:  
<http://cns.csb.yale.edu/v1.1/>  
Commercial users have to purchase a CNX license from Accelrys (MSI).
- The ARIA fortran sources are included in the subdirectory *aria1.2/cns*. Please copy all the files in *aria1.2/cns* to the directory *cns\_solve\_1.1/source* of the unpacked `cns_solve` directory (it will overwrite some files, that is ok)
- edit the `cns_solve_env` in the directory *cns\_solve\_1.1* to define the location of the CNSsolve directory.
- type `make install` in the directory *cns\_solve\_1.1*

## 2.2 Installation of Python

Download and install python with the Numeric module from

<http://www.python.org>

and:

<http://numpy.sourceforge.net>

Please check the readme files for installing Numeric. The compilation should be straightforward.

## 2.3 Downloading ARIA

download `aria1.2.tar.gz` from our homepage:

```
http://www.pasteur.fr/recherche/unites/Binfs/
```

Unzip and untar in a place like `/usr/local` or your home directory.

## 2.4 Installing the cgi scripts on your server

You need two PERL cgi scripts:

`start_project.pl` is used for generating the file `new.html`, `cns_inp2form_0.3.cgi` is used for editing the module `run.cns`.

At EMBL, the cgi scripts have the URLs:

```
http://www.embl-heidelberg.de/nmr/bin/start_project.cgi
```

```
http://www.embl-heidelberg.de/nmr/bin/cns_inp2form_0.3.cgi
```

At the Institut Pasteur, the cgi scripts have the URLs:

```
http://www.pasteur.fr/cgi-bin/Binfs/start_project.pl
```

```
http://www.pasteur.fr/cgi-bin/Binfs/cns_inp2form_0.3.pl
```

If you intend to use the cgi scripts at EMBL or at the Institut Pasteur, you can skip the following paragraph. For a local installation, please follow the following steps':

- copy the cgi files from the directory `/aria1.2/cgi` to the cgi directory of your web server.
- edit the first line of each cgi script to match the path of your perl executable, e.g.:  
`#!/usr/local/bin/perl`  
Perl ([www.perl.org](http://www.perl.org)) is installed on most servers

- in all the `.html` files in the `aria1.2/html` and `aria1.2/html/start` directories, you have to edit the URL addresses of the cgi scripts (according to your server configuration). Thus edit the lines:

```
<form action=
"http://www.nmr.embl-heidelberg.de/bin/start\_project.cgi"
method="POST">
```

in the `.html` documents in `aria1.2/html` and `aria1.2/html/start`. You can also use our server at the EMBL in Heidelberg or the Institut Pasteur in Paris for the cgi scripts. However, it's a good idea to be independent from our www server (in case of downtimes or changes of our server). Ask your system administrator about the `cgi-bin` directory of your server.

## 2.5 editing your shell rc files

Edit your `.cshrc` or `.bashrc` file (depends on the shell you are using, try `'echo $SHELL'` to find out) like this:

- in `.cshrc` or `.tcshrc`:
 

```
setenv PYTHONPATH /home/linge/aria1.2
alias aria '/usr/pub/bin/python /home/linge/aria1.2/Aria/RunAria.py'
```
- in `.zshrc` or `.bashrc`:
 

```
export PYTHONPATH='/home/linge/aria1.2'
alias aria='/usr/pub/bin/python /home/linge/aria1.2/Aria/RunAria.py'
```

If the `PYTHONPATH` system variable is not set properly, you will get a `Traceback` with an `ImportError` when starting ARIA.

## 2.6 Setting up ssh to run ARIA on several machines

If you want to use ARIA on multiple machines, the easiest way is to use ssh to connect and run the job on another machine. In run.cns you can use the queue command:

“ssh other\_computer nice csh” (with or without nice) instead of just “csh”

However, you have to prevent the shell from asking you the password every time you log in. Here comes a short tutorial how to set up ssh accordingly.

How to ssh from A to B without providing a password (in 3 steps):

- I. generate a identity file on A
- II. copy it over to B
- III. add this file to the list of authorized login

I. on A:

you just need to press return after all questions asked (3 times)

- using ssh version 1

```
ssh-keygen
```

Generating public/private rsa1 key pair.

Enter file in which to save the key (/home/tru/.ssh/identity): (press return)

Enter passphrase (empty for no passphrase): (press return)

Enter same passphrase again: (press return)

Your identification has been saved in /home/tru/.ssh/identity.

Your public key has been saved in /home/tru/.ssh/identity.pub.

- using ssh2 (default on our machines redhat 7.1)

There are two crypt algorithms RSA and DSA, do it for both!

I think one is enough, just is case...

```
ssh-keygen -t rsa
```

Generating public/private rsa key pair.

Enter file in which to save the key (/home/tru/.ssh/id\_rsa): (press return)

```

Created directory '/home/tru/.ssh'.
Enter passphrase (empty for no passphrase): (press return)
Enter same passphrase again: (press return)
Your identification has been saved in /home/tru/.ssh/id_rsa.
Your public key has been saved in /home/tru/.ssh/id_rsa.pub.
ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/tru/.ssh/id_dsa): (press return)
Enter passphrase (empty for no passphrase): (press return)
Enter same passphrase again: (press return)
Your identification has been saved in /home/tru/.ssh/id_dsa.
Your public key has been saved in /home/tru/.ssh/id_dsa.pub.

```

```

ls -l
total 24
-rw----- 1 tru Genopole 668 Sep 26 18:37 id_dsa
-rw-r--r-- 1 tru Genopole 614 Sep 26 18:37 id_dsa.pub
-rw----- 1 tru Genopole 539 Sep 26 18:41 identity
-rw-r--r-- 1 tru Genopole 343 Sep 26 18:41 identity.pub
-rw----- 1 tru Genopole 883 Sep 26 18:36 id_rsa
-rw-r--r-- 1 tru Genopole 234 Sep 26 18:36 id_rsa.pub

```

II. on A copying to B the required files  
since you \$HOME is nfs mounted you can use cp otherwise use scp  
to transfer the \*.pub files (ATTN don't overwrite existing ones)

```

cp id_dsa.pub id_dsa.pub-Bis
cp identity.pub identity.pub-Bis
cp id_rsa.pub id_rsa.pub-Bis

```

III. on B "authorize" login from A  
since you \$HOME is nfs mounted you can skip the ssh step :)

```

ssh to B
cd $HOME/.ssh (you might need to create it )

```

```

(on machine B)
cat identity.pub-Bis && authorized_keys

```

```
cat id_rsa.pub-Bis id_dsa.pub-Bis && authorized_keys2
```

That is it!

you can now ssh FROM A to B without password

If you have a shared nfs mounted \$HOME, this means that you can freely ssh from/to any workstation.

## **2.7 WHATCHECK, PROCHECK, PROSA**

I plugged in some code to perform WHATCHECK, PROCHECK and PROSA checks automatically after the iteration 8 and the water refinement. There are some output files QUALITYCHECKS\_REPORT.ascii and QUALITYCHECKS\_REPORT.tex with a summary and statistics over the whole ensemble. You need to install WHATIF, PROCHECK and PROSA in order to use it. Please make sure that you set the system variables PROSA\_BASE and PRODIR correctly (check installation notes of PROSA\_BASE, PRODIR). For testing, you may have a look at the QualityChecks python module which can be used as stand-a-lone package as well (please read the documentation in QualityChecks.py).

# Chapter 3

## Tutorial

### 3.1 Introduction

The aim of this tutorial is to show you how to set up and run a project with ARIA 1.2. As a test case, we use NOE, dihedral, jcoupling, h-bond restraints of the werner HRDC domain [8, 9]. The NOE peaks and chemical shifts are in XEASY format (.peaks and .prot files).

For using ARIA 1.2, one has to use an www browser (e.g. Netscape) two times:

1. to specify where the input files can be found and where the directory for the new project has to be created.
2. to edit all the parameters for the NOE assignment and structure calculation process.

The program ARIA is started from the UNIX command line. It reads in the parameters (as they were specified in and saved from the browser window, see files new.html and run.cns) and starts the automated NOE assignment and structure calculation process. Different tasks are performed automatically: data are converted from different data formats (e.g. XEASY, ANSIG), spectra are calibrated (with an optional NOE backcalculation step) and merged into ambiguous and unambiguous NOE lists and structures are calculated with an Cartesian or torsion angle MD protocol. The quality of the structures can be improved by refining the final structures in a waterbox using a new OPLS/CSDX hybrid force-field [4].

ARIA can be run in parallel (either on a multi-processor machine or on different machines with a queuing system like dqs). This speeds up the NOE

assignment tremendously, e.g. eight iterations (with 20 structures of a 80 residue protein) can be calculated within three hours on 10 cpus (e.g. with Pentium III or R10000 processors).

When ARIA is restarted (e.g. after a system crash), it checks which structures were already calculated and continues with the next structures.

## 3.2 Getting started

Please have a look at the file `/home/aria/aria1.2/html/aria.html` in your browser. You can see which file formats are supported.

Also bookmark our homepage and the ARIA homepage at the Institut Pasteur:

<http://www.pasteur.fr/recherche/unites/Binfs>

<http://www.pasteur.fr/recherche/unites/Binfs/aria>

### 3.2.1 Setting up a new project

You can click on 'start a new project with XEASY data'. Edit the filenames. Save the **new.html** file with the button at the bottom. Please don't use 'save as' from the *file* menu of your browser. It doesn't matter where you save the new.html file. You just have to remember where you saved it. This file contains all the data as html text in a format which can be read by ARIA.

### 3.2.2 Start ARIA within a UNIX shell

Start ARIA in the directory where you saved the new.html file. ARIA will create the directory tree and will convert the data. Have a look at the messages on stdout.

Make sure that the PYTHONPATH system variable contains the path of the ARIA installation. Furthermore, you should set an alias like:

```
alias aria '/usr/bin/python /home/linge/aria1.2/Aria/RunAria.py'
```

If Python errors occur, there are probably due to typos in the specified directory and filenames (please check the Python tracebacks).

### 3.2.3 Editing the parameters in run.cns

Go back to your browser and load the file `/aria1.2/html/aria.html` again. Please read section 5.3 for a description of the `run.cns` file. There is a box where you should write the absolute path of the **run.cns** file which was created in your project directory, e.g.:

`/home/linge/werner/run1/run.cns`.

Press 'edit'. The cgi script will create a mask with which you can edit all the parameters for the calculations. A new window will pop up. After editing the `run.cns` file, save it again as `run.cns` (overwriting the old one). For saving, please use the button at the bottom of the page. Don't use 'save as' from the *file* menu of your browser.

### 3.2.4 Start ARIA within a UNIX shell

Start ARIA in the directory where you saved the `run.cns` file. ARIA will create a `.psf` file, a template `.pdb` file with an extended chain conformation (in the `/begin` directory) and will calibrate all the spectra (`.tbl` and `.list` files in `/structures/it0/`). The merged data will appear in the files `unambig.tbl` and `ambig.tbl` (in `/structures/it0/`). These are the files ARIA uses to calculate the structures of the corresponding iteration. ARIA will automatically calculate the structures for all eight iterations. After the last iteration, standard analysis protocols write their output to `/structures/it8/analysis`. Another calibration and merging step is done with the final structure ensemble. These files are written to `/structures/it8/finaldata`. The waterrefined structures are in `/structures/it8/water`. The BioMagResBank STAR files are written to `/structures/it8/bmr`. The XML files appear in `/structures/it8/xml`. Please have a look at the water refined structures in `/structures/it8/water` and the analysis in `/structures/it8/water/analysis`.

## 3.3 Input files

We are using NMR data from the HRDC domain. All the files can be found in the directory `/aria1.2/example/werner`. The  $^{13}\text{C}$  HMQC and  $^{15}\text{N}$  HSQC spectra were analyzed with XEASY. The original XEASY peak and chemical shift tables are:

- noehsqc.peaks:  
A  $^{15}\text{N}$  HSQC crosspeaks file in XEASY .peaks format.
- hmqcnoe.peaks  
A  $^{13}\text{C}$  HMQC crosspeaks file in XEASY .peaks format.
- noehsqc.prot:  
Chemical shift list of the  $^{15}\text{N}$  HSQC spectrum in XEASY .prot format.
- hmqcnoe.prot:  
Chemical shift list of the  $^{13}\text{C}$  HSQC spectrum in XEASY .prot format.

Chemical shift lists in CNS readable format are:

- 13C.ppm
- 15N.ppm

The sequence in 3-letter code is in the file:

- werner.seq

Hydrogen bonds in CNS readable format are specified in the file:

- hbonds.tbl

Karplus restraints in CNS readable format are in the files:

- dshift\_karplus.tbl
- gamma\_karplus.tbl
- phi\_karplus.tbl

## 3.4 Output files

When you start ARIA, the python part of ARIA will write its output to stdout. The CNS output is saved in .out output files:

- the protocol `generate.inp` (for the generation of the .psf topology file) will write its output to:  
`/run1/begin/generate.out`
- the protocol `generate_template.inp` will write its output to:  
`/run1/begin/generate_template.out`
- the protocol `calib.inp` (for the generation of the calibrated peak lists) will write to:  
`/run1/structures/itx/calib_spectrum.out`
- the protocol `merge.inp` (for the merging of the peak lists) will write to:  
`/run1/structures/itx/merge.out`
- the MD protocol `refine.inp` (for Cartesian or TAD MD) will write its `*refine_x.out` files to the temporary directory (see `temptrash` variable in the file `run.cns`).
- After the last iteration, analysis scripts will print their output to:  
`/structures/it8/analysis.`
- The water refined structures are written to:  
`/structures/it8/water`  
and their analysis files to:  
`/structures/it8/water/analysis`

# Chapter 4

## Data format

For every calibrated spectrum, ARIA is writing a file with the extension `.list` in the directory `/structures/itx`. After the merging of the spectra, ARIA is also saving a file called `merged.list` in `/structures/itx` which contains the NOEs from all spectra which ARIA is using as distance restraints in a particular iteration.

Unfortunately, we couldn't use an XML based format for that because we wanted to read and write the format with CNS (which is coded in Fortran). Nevertheless, we tried hard to get the data in a reasonable table format and we provide the Python module `NoeList.py` for the data conversion in different formats. We will adapt the upcoming CCPN standard as quickly as possible.

After iteration 8, ARIA writes all data in XML format to `structures/it8/xml`. These XML files are compliant with the DTDs in the directory `aria1.2/dtd` which will be used by future versions of ARIA.

### 4.1 ARIA NOE peak files (`.list` suffix)

We will give a short description of the file format. There are some general rules:

- all empty fields must have an `'-'`

- the columns are divided through (one or more) spaces or tabs
- the format is the same for 2D, 3D and 4D spectra (in a 2D file there are just more blank fields than in an 4D)
- the only allowed comment lines begin with an # and continues until the lineend
- every line has to start either with an 'p', 'a', or 'c'

The first field in each line defines the types 'peak', 'aria' or 'contribution':

- 'p' = peak (one line with the input parameters for each peak)
- 'a' = aria (one line with the aria parameters for each peak)
- 'c' = contribution (one line with the parameters of the assignment of one contribution)

Here is an example NOE peak (from a 3D  $^{13}\text{C}$  spectrum, one peak with eight contributions = eight lines with a 'c' in the beginning):

```
p 13C 2905 1 1.00 0.301E+01 - - - 25.287 - 1.094 - - - 4.136 -
a 3 - 1.00 1.90 6.00 0.603E+01 - 0.603E+01 - 8 8
c 1.00 0.41 7.00 - - - - 25 LEU CG - - HG - - - 30 GLY CA - - HA1 - -
c 1.00 0.18 8.00 - - - - 25 LEU CG - - HG - - - 22 LEU CA - - HA - -
c 1.00 0.32 7.29 - - - - 25 LEU CG - - HG - - - 27 LEU CA - - HA - -
c 1.00 0.03 10.76 - - - - 25 LEU CG - - HG - - - 16 ARG CA - - HA - -
c 1.00 0.02 11.69 - - - - 25 LEU CG - - HG - - - 34 VAL CA - - HA - -
c 1.00 0.01 12.98 - - - - 25 LEU CG - - HG - - - 79 ALA CA - - HA - -
c 1.00 0.01 12.30 - - - - 25 LEU CG - - HG - - - 38 GLY CA - - HA1 - -
c 1.00 0.02 12.08 - - - - 25 LEU CG - - HG - - - 21 ARG CA - - HA - -
```

Each **peak line** (starting with a 'p') has the fields:

- 0 'p'
- 1 spectrum name (not longer than 4 characters)
- 2 peak number
- 3 test set flag, either 1 or 0 (1=used for the calculation, 0=not used)
- 4 the initial weight of the peak as defined in the input files
- 5 NOE volume
- 6 NOE volume error
- 7 NOE intensity
- 8 NOE intensity error
- 9 heteronucleus 1 chemical shift
- 10 heteronucleus 1 chemical shift error
- 11 proton 1 chemical shift
- 12 proton 1 chemical shift error
- 13 heteronucleus 2 chemical shift
- 14 heteronucleus 2 chemical shift error
- 15 proton 2 chemical shift
- 16 proton 2 chemical shift error

Each **ARIA line** (starting with an 'a') has the fields:

- 0 'a'
- 1 colour coding in the range [1, 6] defines the peak type:  
 colour code:  
 1 NOE accepted and unambiguous  
 2 NOE accepted and ambiguous  
 3 NOE rejected and unambiguous  
 4 NOE rejected and ambiguous  
 5 NOE not used by ARIA  
 6 new peak
- 2 figure of merit of ARIA for this peak
- 3 current weight
- 4 lower distance bound
- 5 upper distance bound
- 6  $\sum_{contributions} \left( \frac{1}{assignstruc} \sum_{assignstruc} r \right)^{-\frac{1}{6}}$
- 7 standard deviation (not yet used)
- 8  $d_{backcalculated} = \left( \sum_{contributions} NOE_{backcalculated} \right)^{-\frac{1}{6}}$
- 9 standard deviation of the backcalculated volumes (not yet used)
- 10 number of possible assignments (= number of contribution lines)
- 11 number of possible non-trivial assignments (e.g. methyl NOEs only counted once)
- Each **contribution line** (starting with a 'c') has the fields:

```

0  'c'
1  figure of merit of this contribution
2  contribution to the whole NOE peak volume
3  distance (averaged over all assignstruc structures)
4  distance standard deviation
5   $d_{backcalculated} = NOE_{backcalculated}^{-\frac{1}{6}}$ 
6  standard deviation (not yet used)
7  segid 1 (not longer than 4 characters)
8  residue number 1
9  three-letter code 1
10 atomname of heteronucleus 1
11 heteronucleus 1 chemical shift
12 heteronucleus 1 chemical shift error
13 atomname of proton 1 chemical shift
14 proton 1 chemical shift
15 proton 1 chemical shift error
16 segid 2 (not longer than 4 characters)
17 residue number 2
18 three-letter code 2
19 atomname of heteronucleus 2
20 heteronucleus 2 chemical shift
21 heteronucleus 2 chemical shift error
22 atomname of proton 2
23 proton 2 chemical shift
24 proton 2 chemical shift error

```

## 4.2 CNS files (.tbl suffix)

The format of the 2D NOE .tbl files is:

```

assi ( attr store1 < 4.18 and attr store1 > 4.1 )
      ( attr store1 < 8.725 and attr store1 > 8.685 )
      6.0 0.1 0.1 peak 5 volume 0.000e+00 ppm1 4.140 ppm2 8.705
assi ( resid 26 and name HB3 )
      ( resid 26 and name HE21 )

```

```
6.0 0.1 0.1 peak 3417 volume 2.285e+04 ppm1 1.929 ppm2 6.905
```

The first assignment uses an assignment window around the NOE which is to be assigned. The second distance restraints in this example is already fully assigned. The first value is the distance followed by lower and upper bound, the peak number, volume, chemical shifts.

The format of the 3D .tbl files is:

```
assi ( attr store1 < 7.886 and attr store1 > 7.806 )
      (( attr store1 < 7.644 and attr store1 > 7.604 )
       and bondedto ( attr store1 < 112.56 and attr store1 > 111.56))
      6.0 0.1 0.1 peak 5 volume 0.000e+00 ppm1 7.846 ppm2 7.624
assi ( resid 5 and name HN )
      ( resid 5 and name HB# )
      6.0 0.1 0.1 peak 19 volume 7.142e+05 ppm1 8.328 ppm2 1.273
```

Hydrogen-bond lists have the format:

```
assign ( residue 22 and name N ) ( residue 18 and name O )
2.80 0.00 0.50
```

Dihedral restraint lists:

```
assign (resid 10 and name C ) (resid 11 and name N )
      (resid 11 and name CA) (resid 11 and name C )
      1.00 -60.00 40.00 2
```

Residual dipolar coupling lists for the SANI statement (the VEAN statement uses intervector restraints from the program DIPOCOUP):

```
assign ( resid 999 and name O0 )
      ( resid 999 and name Z )
      ( resid 999 and name X )
      ( resid 999 and name Y )
      ( resid 48 and name N )
      ( resid 48 and name HN )
9.8010 0.2000
```

The first four selections define the tensor, the last two define the vector. The two numbers are the rdc and its experimental error.

## 4.3 BioMagResBank support

Jurgen Doreleijers from the BioMagResBank provided Python code for the conversion of chemical shift and NOE peak lists to STAR format. We integrated his Python code into ARIA 1.2. After iteration 8, we automatically convert all the ppm files and the restraint files `unambig.tbl` and `ambig.tbl` of iteration 8 to BMRB format. These files appear in the directory `structures/it8/BMRB`.

The necessary code for the conversion can be found in the module `BMRB.py` and `Nomenclature.py`.

## 4.4 XML support

We have written three DTDs for NOEs, chemical shifts and sequence data. The DTDs can be found in `aria1.2/dtd`. After the last iteration, all the data are written in XML format to the directory `/structures/it8/xml`. These files will be used in future version of ARIA.

## 4.5 Python modules for manipulating the data

Here is a short introduction how to use the Python modules:

- **NoeList.py** for reading and writing NOE peak lists
- **PpmList.py** for reading and writing chemical shift lists
- **SequenceList.py** for reading and writing sequence files

These modules provide means to convert data from the most common assignment programs (XEASY, ANSIG, NMRView, PIPP etc.) in other file formats. Feel free to change the Python sources or to add methods to the classes defined in these modules.

If you are interested in learning Python, please check out [www.python.org](http://www.python.org). Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries, as

well as to various window systems, and is extensible in C or C++. It is also usable as an extension language for applications that need a programmable interface. Finally, Python is portable: it runs on many brands of UNIX, on the Mac, and on PCs under MS-DOS, Windows, Windows NT, and OS/2.

Here are some Python lines to show you how to use these modules:

```
# XEASY files:
print 'reading some 13C data in XEASY format...'

# import the modules (you have to set the PYTHONPATH system
variable in your UNIX shell):
from Aria.DataIO import NoeList, PpmList, SequenceList

# define some variables with the file names etc.
peaksfile = '/home/linge/aria1.2/example/werner/hmqcnoe.peaks' #
XEASY NOE file
protfile = '/home/linge/aria1.2/example/werner/hmqcnoe.prot' # XEASY
ppm file
sequencefile = '/home/linge/aria1.2/example/werner/werner.seq' # con-
tains three-letter code
fileroot = '/home/linge/workshop/hmqcnoe_out' # output file name root
het1 = '1' # the first ppm column contains the heteronucleus 1
pro1 = '2' # the second ppm column contains the proton 1
het2 = 'N' # in the 3D there is no second heteronucleus
pro2 = '3' # the third ppm column contains the proton 2

# instantiate the class NoeList
NL=NoeList.NoeList()

# read the data from the XEASY .peaks and .prot files:
NL.ReadPeaksProt(peaksfile, protfile, het1, pro1, het2, pro2)

# write out a CNS readable .tbl file:
NL.WriteTbl(fileroot + '.tbl')

# for testing write out the XEASY files again:
NL.WriteXeasyAssignPeaks(fileroot, het1, pro1, het2, pro2)
```

```
# instantiate the class PpmList:
PL = PpmList.PpmList()

# read an XEASY .prot file:
PL.ReadXeasyProt(protfile)

# printing the content to STDOUT:
PL.Stdout()

# writing a CNS readable file:
PL.WriteChem(fileroot + '.chem')

# writing an XML file:
PL.WriteXML2File(fileroot + '.xml')

# instantiating the class SequenceList:
SL = SequenceList.SequenceList()

# reading the sequence in 3-letter code:
SL.ReadSeq(sequencefile)

# writing the sequence in FASTA format:
SL.WriteFasta(fileroot + '.fasta')

# ANSIG files:
print 'reading some 15N data in ANSIG format...'

# define some variables with the file names etc.
cpkfile = '/home/linge/aria1.2/example/ansig/Nnoigen.crp' # ANSIG cpk
file
sequencefile = '/home/linge/aria1.2/example/s15.seq' # ANSIG sequence
file
fileroot = '/home/linge/workshop/ansig_out' # output file name root
het1 = '2' # the second ppm column contains the heteronucleus 1
pro1 = '1' # the first ppm column contains the proton 1
het2 = 'N' # in the 3D there is no second heteronucleus
pro2 = '3' # the third ppm column contains the proton 2
```

```
print 'reading some 15N data in ANSIG format...'  
  
# instantiate the class NoeList  
NL=NoeList.NoeList()  
  
# read the data from an ANSIG crosspeaks export file:  
NL.ReadAnsigCpe(cpkfile, het1, pro1, het2, pro2)  
  
# write out a CNS readable .tbl file:  
NL.WriteTbl(fileroot + '.tbl')  
  
# instantiate the class PpmList  
PL=PpmList.PpmList()  
  
# read the ANSIG crosspeaks export file:  
PL.ReadAnsig(cpkfile)  
  
# write a CNS readable .tbl file:  
PL.WriteChem(fileroot + '.ppm')  
  
# another example: reading from pdb, converting to fast and 3-letter-code...  
pdbfilename = '/home/linge/workshop/werner.pdb'  
fastafilename = '/home/kubge/workshop/werner.fasta'  
3letterfilename = '/home/linge/workshop/werner.seq'  
SL = SequenceList.SequenceList()  
SL.ReadPdb(pdbfilename)  
SL.Stdout()  
SL.WriteFasta(fastafilename)  
SL.WriteSeq(3letterfilename)
```

The modules themselves provide some documentation in their headers which tells you the details of the attributes and methods of all these classes.

We have written some conversion routines for different broadly available programs. All these methods are tested. However, as the output file formats of the most common NMR programs are rather obscure, it's difficult to consider all the possible variations in these files. Thus, some methods might cause trouble even when the provided test cases from the directory

aria1.2/examples do work.

If you want to read, write and edit the .list file, you can use the ReadList() and WriteList() methods of NoeList.py as a starting point. Have a look at the code and adapt it to your own needs, e.g. it is easy to change the fields for every peak or every contribution of one peak.

# Chapter 5

## Description of ARIA protocols

### 5.1 Overview

1. setup of run directory tree (python)
2. data conversion and setup of the run subdirectories (python)
3. **generate.inp**: MTF file (molecular topology file, a.k.a. as psf or protein structure file)
4. **generate\_template.inp**: template structure (pdb file of an extended chain conformation which is minimized)
5. in each iteration
  - (a) for each spectrum: **calib.inp**: assign and calibrate restraints
  - (b) **merge.inp**: merge restraints
  - (c) **refine.inp**: structures
  - (d) energy sort structures: file.list, file.nam (for MOLMOL), file.cns
  - (e) convert assigned spectra to peak list format
6. for last iteration:
  - (a) analysis (wellordered, noe violations, WHATCHECK, PROCHECK, PROSA etc)
  - (b) refinement in an explicit shell of water

- (c) analysis of water refined structures (wellordered, noe violations, WHATCHECK, PROCHECK, PROSA etc)
- (d) BioMagResBank, XML, molmol (.upl and .lol) files

## 5.2 setup of run directory tree

```
run1/  
  run.cns  
  begin/  
  data/  
    spectrum1/  
    spectrum2/  
    .../  
    hbonds/  
    ssbonds  
    sequence/  
    jcouplings/  
    rdcs/  
    .../  
  structures/  
    it0/  
    .../  
    it8/  
      water/  
      analysis/  
      bmr  
      xml  
      finaldata  
  protocols/  
  toppar/  
run2/  
.../
```

### 5.3 run.cns

The file `run.cns` all parameters for the ARIA, the simulated annealing script, and the analysis are stored in the following compound parameters:

**Spectrum** the name of the spectrum currently worked on

**Iteration** the iteration currently worked on

**Filenames** various filenames (e.g., psf file etc)

**Spectra** parameter for spectra (name, mixing time etc)

**Data** parameter for data

**Iterations** aria assignment parameters

**Saprotoocol** simulated annealing protocol parameters

**Refine** refinement parameters (e.g., h2o refinement)

**Relax** relaxation matrix calculation

**Toppar** topology and parameter information

**Analysis** analysis parameters

The following variables are important for the NOE assignment process:

**Qshift** use re-assignment based on chemical shifts. If true, shifts file is read in and used for the assignments. If false, only fully assigned peaks will be used.

**Qrelax** switch spin diffusion correction off or on (if true, you need to set spectrometer frequency, rotation correlation time, mixing time)

**Qcalib** use automated calibration to re-derive target distances from volumes. If true, calibration is switched on. If false, ARIA does not calibrate the peaks.

**Qerrset** re-define error bounds based on target distance or volume.

**Qmove** re-“calibrate” violated restraints by setting them to 0...6 (if true, moving of upper bounds to 6.0 Å possible)

**Qexclude** remove violated restraints (exclusion of peaks possible/impossible)

**qfadjust** not yet implemented

**errmod** DIST or VOLUME (at the moment, we use DIST as default)

**solvent** not yet implemented

Please have a look at the atomname nomenclature of your peak lists (use the atomname nomenclature switch in *run.cns*). ARIA writes *pdb* files in IUPAC format (thus e.g. LEU HB2 and HB3 instead of HB2 and HB3). The termini are H1, H2, H3 and OXT, O. Please have a look at *topallhdg5.2.pro* in the *toppar* directory to make sure that you are using the same conventions.

You can use CSI or TALOS restraints with our setup, but also be aware of the empirical content of these restraints. You can switch them on or off in *run.cns*.

You may want to play around with the maximum number of assignment possibilities *maxn*, the violation tolerance, the ambiguous cutoff and the violation ratio (see [7] for a discussion of these values).

Usually, we calculate 20 structures, take the 10 best structures (regarding total energy) as template structures for the next iteration and the 7 best structures for the statistics upon which we accept or reject peaks.

For setting up *aria* on several machines or with a queuing system, please specify all the *cns* executables for all the machines you are using (e.g. a SGI and a linux executable) and use something like *cs* or *rsh* or *ssh* or *dqs* to distribute them to the other machines. The *cns* jobs are distributed using some *.job* files (see temporary directory) which are started with the specified queue command. The Python jobs will always run on the machine on which the main job is running.

You may want to change the number of steps of your simulated annealing protocol. To improve convergence, please multiply the number of steps by 2,

4 or even 8. The default values are pretty good, but you may change them as you like. Water refinement is switched on by default. It should be done before submission of your structures to the pdb.

You can specify procheck, whatcheck and prosa executables in the run.cns files (it will produce some nice statistics in ascii and LaTeX format).

ARIA sets some default values automatically. Thus, a first try would be to calculate some structures without touching the run.cns file at all. More experienced users may want to edit the assignment parameters and the simulated annealing parameters at will. You can also change the CNS protocols, e.g. refine.inp.

## 5.4 generate.inp

like standard MTF file generation. in addition,

- generate table of equivalent protons with the file define\_methyls.ini.cns

```
display do (store1 = 0) (all)
ident (store9) (bondedt (name h*))
for $loopid in id (store9) loop meth
  coor select (bondedt(id $loopid) and name h*) end
  if ($select eq 3) then
    display do (store1 = 1) (id $loopid)
  end if
end loop meth
```

The result is in begin/methyls.tbl.

- set up floating assignment table with setup\_swap\_init.cns. Stereospecific assignments need to be defined in data/sequence/stereoassign.cns  
The result is in begin/setup\_swap\_list.tbl
- switch to IUPAC notation for protons by applying (internally!) protocols/xplortodiana3.inp

## 5.5 generate\_template.inp

generation of template structure by minimization of random structure  
 result: begin/template.pdb

## 5.6 stereoassign.cns

- define stereospecific assignments
- lines to be edited are marked by

```
{===>}
```

- swapped assignments:

```
! methylene protons
for $id in id
(
{===>} (name ca and (resid 8 or resid 9))
{===>}or (name cb and (resid 10 or resid 11))
{===>}or (name cg and (resid 12 or resid 13))
{===>}or (name cg1 and (resid 14))
)
loop Lmethy
  display aria revert (bondedto (id $id)
    and name h*) end
  display do (store1=0) (bondedto (id $id)
    and name h*)
end loop Lmethy

! isopropyle groups
for $id in id (name ca and
(
{===>}resid 84
))
loop Vrev
  display do (store1=0) (byresid (id $id)
    and (name cg1 or name cd1))
```

```

show (resn) (id $id)
if ($result eq VAL) then
  evaluate ($name3 = cg1)
  evaluate ($name4 = cg2)
elseif ($result eq LEU) then
  evaluate ($name3 = cd1)
  evaluate ($name4 = cd2)
end if
display aria revert (bondedto
  (byresid(id $id)
  display and (name $name3 or name $name4))
  and name h*) end
end loop Vrev

```

- assignments “as is”

```

! methylene protons
for $id in id
(
{===>} (name ca and (resid 8 or resid 9))
{===>}or (name cb and (resid 10 or resid 11))
{===>}or (name cg and (resid 12 or resid 13))
{===>}or (name cg1 and (resid 14))
)
loop Lmethy
  display do (store1=0) (bondedto (id $id)
    and name h*)
end loop Lalph

```

```

!isopropyle groups
for $id in id (name ca and
(
{===>} resid 27
{===>}or resid 29
{===>}or resid 46
{===>}or resid 63
{===>}or resid 37
{===>}or resid 66

```

```

))
loop Vrev
  display do (store1=0) (byresid (id $id) and
    (name cg1 or name cd1))
end loop Vrev

```

## 5.7 Sorting of structures

- sort coordinate files by total energy: python script, analysing line

```

REMARK FILENAME="/home/linge/werner1.2/run11/structures/it8/werner1.2_32.pdb"
REMARK =====
REMARK          overall,bonds,angles,improper,dihe,vdw,elec,noe,cdih,coup,sani
REMARK energies: 1838.58, 36.4741, 191.084, 37.6665, 337.142, 388.097, 0, 848.12
0, 0, 0, 0
REMARK =====
REMARK          bonds,angles,impropers,dihe,noe,cdih,coup,sani,vean
REMARK rms-dev.: 4.877767E-03,0.672506,0.55669,17.6079,0.223152,0,0, 0, 0
REMARK =====
REMARK          noe,cdih,coup,sani,vean
REMARK          >0.5,>5,>1,>0,>5
REMARK violations.: 21, 0, 0, 0, 0
REMARK =====
REMARK DATE:10-Nov-2001  10:22:13          created by user: linge
REMARK VERSION:1.2
%

```

- result in file.list, file.nam and file.cns

```

% "PREVIT:bptitest18.pdb" { 6193.75, }
% "PREVIT:bptitest1.pdb" { 6479.29, }
% "PREVIT:bptitest14.pdb" { 6493.26, }
% "PREVIT:bptitest19.pdb" { 6556.19, }
% "PREVIT:bptitest11.pdb" { 6779.57, }
% "PREVIT:bptitest6.pdb" { 6783.73, }
% "PREVIT:bptitest4.pdb" { 6799.72, }
% "PREVIT:bptitest10.pdb" { 6801.73, }

```

```
% "PREVIT:bptitest3.pdb" { 7123.47, }  
% ...  
%
```

- in iteration 0: file.list contains
  - name of template file
  - name of homology model
  - list of sorted pre-calculated structures

## 5.8 calib.inp

calibration, violation analysis, and assignment.

### Operations

1. read in structures from file.list and store *average* distances
2. back-calculate NOE spectrum
3. calibrate calculated vs. experimental NOEs
4. set error bounds
5. check for violated restraints
6. exclude violated restraints
7. re-calibrate
8. re-set error bounds
9. check for violated restraints
10. move bounds for violated restraints
11. check for violated restraints
12. exclude violated restraints
13. assign and write out restraints

**Governed by the flags** **Qshift** use re-assignment based on chemical shifts. If true, shifts file is read in.

**Qrelax** use back-calculation.

**Qcalib** use automated calibration to re-derive target distances from volumes.

**Qerrset** re-define error bounds based on target distance or volume.

**Qmove** re-“calibrate” violated restraints by setting them to 0...6

**Qexclude** remove violated restraints

**reading in data** : chemical shifts are stored in store1 and rmsd

```
do (store1 = -999.0)
if ($Spectra.Qshifts eq TRUE) then
  evaluate ($ShiftsFile = "SPECTRUM:" + $Spectra.Shifts)
  @@$ShiftsFile
end if
evaluate ($PeaklistFile = "SPECTRUM:" + $Spectra.Peaklist)
do (rmsd = store1) (all)
noe
...
  @@$PeaklistFile
...
```

**equivalent protons**

```
@protocols/define_methyls_all.cns
@begin/setup_swap_list.tbl
```

distance averaging options:

- *aver* calculates arithmetic average  $\langle r \rangle$ ;
- *accumulate*,  $\langle r^{-6} \rangle^{-1/6}$ ;
- *minimum*.
- Result is an internal distance matrix  $\hat{d}_{ij}$

```
aria analyse_restraints reset end end
evaluate ($count = 0)
for $file in ( @@PREVIT:file.list ) loop main
  evaluate ($count = $count + 1)
  coor init end
```

```

    coor @@$file
    aria analyse_restraints
        aver
    end end
    if ($count ge $Iterations.AssignStruct) then
        exit loop main
    end if
end loop main

```

### back-calculation

- command:  
*aria back\_calculate <class> ... end end*
- for each spectrum: mixing time, correlation time, frequency, solvent
- general parameters: distance cutoff for setup of the relaxation matrix, number of “layers”, number of matrix squares
- command:

```

if ($Spectra.Qrelax eq true) then
    aria analyse
        back_calculate DIST
        frequency=$Spectra.frequency
        tcorrel=$Spectra.tcorrel
        tmix=$Spectra.tmix
        nlayers=$relax.nlayers
        cutoff=$relax.cutoff
        mdouble=$relax.mdouble
    end end end
end if

```

- relaxation matrix calculation calibrates experimental volumes such that  
 $NOE_k \rightarrow \min(d_i^{-6} NOE_k / \max(NOE_l))$

### calibration @protocols/calibrate.cns

- explicit (with reference volume and distance) or automatic

- different proton-proton vectors can be calibrated differently
- mode isotrop:  $C_{IJ} = \sqrt{C_{II}C_{JJ}}$
- errmod: error estimate from distance or volume<sup>-1/6</sup>

```

if ($Spectra.Qcalib eq TRUE) then
! define groups of protons for the calibration
! store1 exchanging protons; store2 aromatic protons
! store3 aliphatic, store4 alpha, store5 methyl
ident (store1) (name hn or name ht#
              or ... or (resn arg and name hh#))
ident (store2) ...
! calibrate. this changes the values in the NOEDIS array

aria calibrate
  reset    mode isotrop  cutoff 6.0
  errmode &Spectra.errmod
  !automatic calibration with two different groups
  vector (store3 or store4 or store5) (store3 or store4
    or store5) auto
  vector (store1 or store2) (store3 or store4 or store5)
    auto
  vector (store1 or store2) (store1 or store2) auto
end end
end if

```

**error bounds** : @protocols/errset.cns

- aria calibrate defines the data arrays noe\_lower and noe\_higher as target distance or exp. volume<sup>-1/6</sup>
- evaluate (\$lowUp = 2.2)  
evaluate (\$highUp = 6.0)

```

if (&Spectra.Qerrset eq TRUE) then
  aria
    do (noe_distance = int(noe_distance*10+0.5)/10)
    do (noe_distance = min(noe_distance, $highUp))
    do (noe_higher = &Spectra.err0
      + &Spectra.err1 *noe_higher

```

```

+ &Spectra.err2 *noe_higher**2
+ &Spectra.err3 *noe_higher**3)
do (noe_lower = &Spectra.err0
+ &Spectra.err1 *noe_lower
+ &Spectra.err2 *noe_lower**2
+ &Spectra.err3 *noe_lower**3)

do (noe_higher=max(noe_higher,$lowUp-noe_distance))
do (noe_higher=min(noe_higher,$highUp-noe_distance))
do (noe_lower =min(noe_lower, noe_distance))

do (noe_higher=int(noe_higher*10.0+0.5)/10)
do (noe_lower =int(noe_lower*10.0+0.5)/10)
end
end if

```

**check violations :**

```

do (store2=store9) (all)
aria countvio reset end end
evaluate ($count = 0)
for $file in ( @@PREVIT:file.list ) loop main
evaluate ($count = $count + 1)
coord init end
coord @$file
@protocols/swap.cns(swap=1.0001)
aria
countvio threshold $Iterations.violtoler end
end
if ($count ge $Iterations.assignstruct)
then exit loop main end if
end loop main

```

**exclude violations :**

```

if ($Spectra.qexclude eq TRUE) then
aria countvio
exclude DIST $Iterations.violratio

```

```
    end end  
  end if
```

re-calibrate

re-set error bounds

re-check violations

**move bounds** for violated restraints

```
  if ($Spectra.qmove eq TRUE) then  
    aria countvio  
      set DIST $Iterations.violratio 0.0 6.0  
    end end  
  end if
```

re-check for violations

exclude violations

list violated and excluded restraints

```
  aria countvio  
    list DIST $Iterations.violratio  
  end end
```

**assign** and write out restraints

```
  evaluate ($restraintfile = "NEWIT:" + $Spectra.restraintfile)  
  set print $restraintfile end  
  aria analyse_restraints  
    mode intra  
    minn 1 maxn $Iterations.maxn  
    cutoff 10000  
    level $Iterations.ambigcutoff  
    or-restraint DIST  
  end end
```

```

! list them again in the .list format
! this lists all restraints (also excluded ones)
evaluate ($listfile = "NEWIT:" + $Spectra.listfile)
set print $listfile end
aria analyse_restraints
    minn 1 maxn 200
    cutoff 10000
    level $Iterations.ambigcutoff
    sort DIST
end end

```

result file: table file (ambiguous and unambiguous mixed)

```

ASSI { 32}
  (( segid "    " and resid 2    and name HB3 ))
  (( segid "    " and resid 2    and name HA  ))
  6.0 3.6 0.0 peak 32 weight 0.10E+01 volume 0.00E+00 ppm1 1.849 ppm2 4.432 CV 1
ASSI { 36}
  (( segid "    " and resid 2    and name HB3 ))
  (( segid "    " and resid 2    and name HD3 ))
  6.0 3.6 0.0 peak 36 weight 0.10E+01 volume 0.00E+00 ppm1 1.849 ppm2 3.143 CV 1
OR { 36}
  (( segid "    " and resid 2    and name HB3 ))
  (( segid "    " and resid 2    and name HD2 ))
OR { 36}
  (( segid "    " and resid 48   and name HB3 ))
  (( segid "    " and resid 48   and name HD3 ))

```

and list file:

```

p DIST 12 1 1.00 - - - - 0.000 2.137 0.000 - 0.000 2.387 0.000
a 4 0.00 1.00 2.40 6.00 0.186E+01 0.000E+00 0.178E+01 0.000E+00 8 2
c 1.00 0.16 2.43 0.00 0.00 0.00 - 8 GLU CB -999.000 0.000 HB2 2.130 0.000 - 8 GLU CG
c 1.00 0.16 2.44 0.00 0.00 0.00 - 8 GLU CB -999.000 0.000 HB3 2.130 0.000 - 8 GLU CG
c 1.00 0.15 2.45 0.00 0.00 0.00 - 1 MET CB -999.000 0.000 HB2 2.137 0.000 - 1 MET CG
c 1.00 0.13 2.53 0.00 0.00 0.00 - 1 MET CB -999.000 0.000 HB3 2.137 0.000 - 1 MET CG
c 1.00 0.11 2.57 0.00 0.00 0.00 - 1 MET CB -999.000 0.000 HB3 2.137 0.000 - 1 MET CG
c 1.00 0.11 2.58 0.00 0.00 0.00 - 1 MET CB -999.000 0.000 HB2 2.137 0.000 - 1 MET CG
c 1.00 0.10 2.64 0.00 0.00 0.00 - 8 GLU CB -999.000 0.000 HB2 2.130 0.000 - 8 GLU CG
c 1.00 0.10 2.64 0.00 0.00 0.00 - 8 GLU CB -999.000 0.000 HB3 2.130 0.000 - 8 GLU CG

```

## 5.9 merge.inp

- merges several restraint files
- command *aria analyse check <class> end end*
- *level* (“p”) and other parameters have same effect as inparameters have the s

```
aria analyse_restraints
  cutoff 1000
  level 1.01
  check *
end end
```

- result: files unambig.tbl and ambig.tbl

```
ASSI { 5}
  (( segid " " and resid 8 and name HA ))
  (( segid " " and resid 8 and name HN ))
  6.0 3.6 0.0 peak 5 weight 0.10E+01 volume 0.00E+00 ppm1 4.140 ppm2 8.705 CV 1
OR { 5}
  (( segid " " and resid 34 and name HA ))
  (( segid " " and resid 34 and name HN ))
OR { 5}
  (( segid " " and resid 7 and name HA ))
  (( segid " " and resid 8 and name HN ))
```

# Chapter 6

## Analysis protocols

### 6.1 Overview

In each iteration:

- energies and rms differences in coordinate header
- analysis of restraint violations in calib.out file and in .list file
- conversion of assigned peak lists to interactive format (XEASY, ANSIG, NMRView)

For the last iteration:

- conversion of restraints to MOLMOL
- print\_noe.inp: noe rms/violations
- print\_dih.inp: dihedral angle rms/violations
- print\_coup.inp: coupling constant rms/violations
- print\_geom.inp: geometrical analysis (bonds, angles...)
- cop.inp: Ramachandran plot and circular order parameter
- wellordered.inp:
  - ordered region

- average structure
  - sequential rmsd
  - fitted ensemble
- rmsave.inp: average rms difference

## 6.2 PDB header

energies of structures in each iteration

```
REMARK FILENAME="/home/linge/werner1.2/run11/structures/it8/werner1.2_32.pdb"
REMARK =====
REMARK          overall,bonds,angles,improper,dihe,vdw,elec,noe,cdih,coup,sani,vean
REMARK energies: 1838.58, 36.4741, 191.084, 37.6665, 337.142, 388.097, 0, 848.121,
0, 0, 0, 0
REMARK =====
REMARK          bonds,angles,impropers,dihe,noe,cdih,coup,sani,vean
REMARK rms-dev.: 4.877767E-03,0.672506,0.55669,17.6079,0.223152,0,0, 0, 0
REMARK =====
REMARK          noe,cdih,coup,sani,vean
REMARK          >0.5,>5,>1,>0,>5
REMARK violations.: 21, 0, 0, 0, 0
REMARK =====
REMARK DATE:10-Nov-2001  10:22:13          created by user: linge
REMARK VERSION:1.2
```

## 6.3 calib.inp

analysis of restraint violations in calib.out file

```
ARIA>      countvio threshold $Iterations.violtoler end
ARIEXC: peak, lower, upper, distance  24  0.25000E+01  0.60000E+01  0.23107E+01
ARIEXC: peak, lower, upper, distance  58  0.24000E+01  0.60000E+01  0.20178E+01
ARIEXC: peak, lower, upper, distance 115  0.24000E+01  0.60000E+01  0.20228E+01
ARIEXC: peak, lower, upper, distance 184  0.23000E+01  0.49000E+01  0.51916E+01
ARIEXC: peak, lower, upper, distance 282  0.24000E+01  0.60000E+01  0.21938E+01

CNSsolve>aria countvio list DIST $Iterations.violratio {%/100} end end
===== restraint    58 =====
set-i-atoms
      2    ARG  HD3
      2    ARG  HD2
```

```

          14  PHE  HB2
          29  ILE  HA
          48  ARG  HD3
set-j-atoms
          2   ARG  HE
          29  ILE  HN
          38  PHE  HE1
          38  PHE  HE2
          53  PHE  HE1
          53  PHE  HE2
          59  ARG  HE

```

4 Violations Above Threshold, Rms Violation= 0.27 Target distance= 6.0 (- 3.6/+ 0.0)

## 6.4 print\_noe.inp

- result: NEWIT:analysis/noe.disp
- results for each file and overall statistics

```

all distance restraints
PREVIT:argRN_2.pdb 6.347854E-02
PREVIT:argRN_13.pdb 8.196656E-02
PREVIT:argRN_7.pdb 7.379967E-02
PREVIT:argRN_4.pdb 7.458125E-02
PREVIT:argRN_15.pdb 6.125104E-02
PREVIT:argRN_16.pdb 7.90166E-02
PREVIT:argRN_3.pdb 7.892201E-02
mean values
noe 7.328795E-02 (+/- 7.396184E-03 )
...

```

- separate statistics for
  - all distance restraints
  - only unambiguous
  - only ambiguous
  - all noe distance restraints
  - only hydrogen bond

## 6.5 print\_dih.inp

- prints/ counts violations larger than 5 degrees

- result: NEWIT:analysis/dihedrals.disp
- rdb format

```
# DIH PREVIT:werner_3.pdb rmsd: 0.678812    vio: 0
# DIH PREVIT:werner_4.pdb rmsd: 0.515913    vio: 0
# DIH PREVIT:werner_1.pdb rmsd: 1.36146     vio: 1
# mean values data/dihedrals/dihedrals.tbl
# DIH rmsd (deg) 0.852061 (0.366285)
# DIH vio > 5 deg: 0.333333 (0.471405)
m_rms_dih      sd_rms_dih      cutdih  m_vio_dih      sd_vio_dih
0.852061      0.366285      5       0.333333      0.471405
```

## 6.6 print\_coup.inp

- prints/ counts violations larger than 0.5 Hz
- result: NEWIT:analysis/couplings.disp
- statistics for up to 5 classes separately

```
# c1: PREVIT:werner_3.pdb 1.91216
# c1: PREVIT:werner_4.pdb 1.92177
# c1: PREVIT:werner_1.pdb 1.98139
# mean values couplings class c1
# coup 1.93844 (+/- 3.062418E-02 )
```

## 6.7 print\_geom.inp

- counts violations over 0.025 Å for bonds, 2.5 degrees for angles and improper
- calculates repel vdW energy
- result: NEWIT:analysis/geom.disp

```
# geometry analysis
# filename      rms_bond      rms_angle      rms_impr
# PREVIT:werner_3.pdb 1.704921E-03  0.340385      0.202822
```

```

# PREVIT:werner_4.pdb  1.868641E-03  0.353714  0.212289
# PREVIT:werner_1.pdb  2.1967E-03  0.399033  0.264723
# mean values
# bond 1.923421E-03 (+/- 2.044706E-04 )
# angle 0.364377 (+/- 2.510233E-02 )
# impr 0.226611 (+/- 2.722499E-02 )
# vdw 62.2524 (+/- 3.48706 )
mean_rms_bond stdev_rms_bond mean_rms_angle stdev_rms_angle mean_rms_impr
  stdev_rms_impr mean_vdw stdev_vdw
1.923421E-03 2.044706E-04 0.364377 2.510233E-02 0.226611 2.722499E-02
62.2524 3.48706

```

## 6.8 cop.inp

Ramachandran plot and circular order parameter

- result: NEWIT:analysis/cop.disp

```

display residue phi    psi    avphi  avpsi
# circular order parameter
residue phi    psi    avphi  avpsi
1      0      0.485031    0      70.9808
2      0.914459    0.936289    -61.9632    159.277
3      0.334463    0.335955    17.1335 71.8634
4      0.959342    0.139638    -48.5661    87.9373

```

- result: NEWIT:analysis/ramachandran.disp

```

# ramachandran plot
reskey result phi    psi
1      0      360    21.2894
1      0.914459    -137.793    -178.708
1      0.334463    57.973    -168.37
1      0.139638    -105.424    78.0368

```

## 6.9 rmsave.inp

- result: NEWIT:analysis/rmsave.disp
- uses average structure from wellordered

- and secondary structure defined in data/sequence/secondarystruct
- rdb format

```
# NEWIT:analysis/wernerfit_1.pdb 2.7021 3.33567 8.01771 9.12359
# NEWIT:analysis/wernerfit_2.pdb 1.72461 2.28108 5.22951 5.88154
# NEWIT:analysis/wernerfit_3.pdb 0.876135 0.914621 1.17966 1.24068
# mean values
# backbone, 2nd struct      : 1.76761 (+/- 0.746066 )
# heavy atoms, 2nd struct  : 2.17713 (+/- 0.99112 )
# backbone, all residues   : 4.80896 (+/- 2.80742 )
# heavy atoms, all residues: 5.41527 (+/- 3.23503 )
mean_rms_wb stdev_rms_wb mean_rms_wa stdev_rms_wa mean_rms_ab
  stdev_rms_ab mean_rms_aa stdev_rms_aa
1.76761 0.746066 2.17713 0.99112 4.80896 2.80742 5.41527 3.23503
```

## 6.10 wellordered.inp

- searches for welldefined regions by iteratively excluding all residues  $i$  for which  $rms_i > \overline{rms} + 2\sigma$
- parameters:

```
!starting cutoff level:
evaluate ($nsigma = 2)
!convergence criterion:
evaluate ($mean_toler = 0.5)
evaluate ($stdev_toler = 0.05)
stop
```

- result: “outroot”.prt: results of iterations

```
! all atoms with Rms > RmsLimit are excluded
! RmsLimit = MeanRms + SigmaLimit*Stdev
! Iteration SigmaLimit RmsLimit MeanRms Stdev nFittedAtoms
1 2 0 3.19202 2.38972 273
2 2 7.97146 2.58186 1.08085 252
```

```

3 2 4.74355 2.46305 0.856696 244
4 2 4.17645 2.40302 0.793234 237
5 2 3.98949 2.39573 0.786939 236

```

- result: average structure in:  
NEWIT:analysis/“outroot”\_ave.pdb  
contains definition of wellordered region in Q column

```

ATOM      19  CA  MET      1    -23.940  -1.132 -17.032  0.00 11.98
ATOM      22  CA  LYS      2    -22.172   2.039 -18.175  0.00 10.46
ATOM      44  CA  HIS      3    -22.759   4.435 -21.084  0.00 11.16
ATOM      62  CA  HIS      4    -25.867   5.509 -19.166  0.00 11.25
ATOM      80  CA  HIS      5    -24.596   6.816 -15.823  0.00 10.65
ATOM      98  CA  HIS      6    -27.787   5.466 -14.255  0.00  9.69
ATOM     116  CA  HIS      7    -26.303   3.414 -11.419  0.00  9.01
ATOM     134  CA  HIS      8    -24.634   5.299  -8.568  0.00  6.83
ATOM     151  CA  PRO      9    -21.077   3.959  -8.000  0.00  4.40
ATOM     166  CA  MET     10    -19.158   4.353  -4.739  1.00  3.64
ATOM     183  CA  GLU     11    -16.445   6.840  -5.705  1.00  3.33
ATOM     198  CA  LEU     12    -15.802   7.382  -1.995  1.00  2.60
ATOM     217  CA  ASN     13    -12.993   4.863  -1.526  1.00  3.50
ATOM     231  CA  ASN     14    -11.013   4.642   1.715  1.00  3.87

```

- result: NEWIT:analysis/rmsdseq.disp  
sequential rms-differences

```

# sequential rms differences
resid  rmsback3      rmsback5      rmsside rmsall
1      12.0123 11.6756 11.2303 11.5107
2      10.7728 10.7832 11.3349 11.0318
3      10.9388 11.1164 13.9294 12.6016

```

- result: fitted ensemble  
NEWIT:analysis/“outroot”fit\_i.pdb

# Chapter 7

## Appendix – CNS scripting language

### 7.1 General features

- variables (symbols) (marked by \$)
- parameters (marked by &)
- if-statements
- loops
- atom selection
- data structure manipulations
- many application statements
- mathematical functions  
for variable and data structure manipulations
- include files
- modules

## 7.2 Symbol definitions and the EVALuate statement

- recognized by \$ sign
- symbols are defined *and* manipulated by EVALuate

```
evaluate ($count = 0)
evaluate ($filename = "dg.pdb")
```

- Symbols can be
  - real numbers
  - strings
- the type definition is implicit by usage
- type conversion by encode and decode

```
evaluate ($name = encode($count))
evaluate ($number = decode($name))
```

- variables can be indexed:

```
evaluate ($item_1 = 1)
evaluate ($item_2 = 2)
evaluate ($sum = 0)
for $i in (1 2) loop lsum
  evaluate ($sum = $sum + $item_$i)
end loop lsum
```

- formatting:

```
display $result[F8.3]
```

- variables have scope: global variables are defined in a module level above present
- \$? produces list of all defined symbols

## 7.3 Parameter definition with DEFINE

- in module define section:

```
define (
  parameter1 = 3 ;
  parameter2 = 15 ;
)
```

- are referenced within module with &

```
evaluate ($newvariable = &parameter1)
```

- Parameters can hold very different things:

- numbers
- strings
- commands
- atom selections

- can be compound:

```
define (
  SaProtocol.iniseed=88779 ;
  SaProtocol.carinit_t=2000 ;
  SaProtocol.tadinit_t=10000 ;
  SaProtocol.final1_t=1000 ;
  SaProtocol.final2_t=50 ;
  ...
)
```

this can be transferred to a module as one single parameter:

```
@RUN:protocols/sa_ls_cool2.cns (SaProtocol=
  &SaProtocol; ... ;)
```

## 7.4 Arithmetic operations

- Standard operations

```
+ - * / ** ^ ( )
evaluate($number = (5*$count)^(3+$count))
```

- mathematical functions

```
cos sin ran ...
evaluate($number = sin( $count*ran() ))
```

## 7.5 Special symbols

- Fundamental constants

```
$pi $kboltz
```

- Results of certain operations (incomplete list)

- PRINT statements define \$result

```
print angle
evaluate ($rms_angle = $result)
```

- SHOW statements define \$result

```
show average (x) (all)
evaluate ($x_ave = $result)
```

- ENERgy, MINImiz and DYNAmics define energy terms

```
energy end
display $ener $bond $angl
```

## 7.6 IF statements

- basic structures:
  - IF ( condition ) THEN commands END IF
  - IF ( condition ) THEN commands  
ELSE commands END IF
  - “case” statement  
IF ( condition ) THEN commands  
ELSEIF (condition) THEN commands  
...  
END IF

- can be nested

- note: ELSEIF is *not* ELSE IF

- two “end if” necessary

```
if ( $count eq 1 )
then
  coor copy end
else
  if ( $count eq 2 )
  then
    coor fit end
  end if
end if
```

- one “end if” necessary

```
if ( $count eq 1 )
then
  coor copy end
elseif ( $count eq 2 )
then
  coor fit end
end if
```

## 7.7 Loops

- WHILE loop:  
WHILE (condition) LOOP loop-name  
  commands  
END LOOP loop-name  
  
  evaluate (\$count = 1)  
  while (\$count le 10) loop main  
    evaluate (\$count = \$count + 1)  
  end loop main
- FOR loop:  
FOR variable IN (set)  
  
  for \$filename in ( "sa\_1.pdb" "sa\_3.pdb" )  
  loop main  
    coord @@\$filename  
  end loop main
- FOR loop:  
FOR variable IN ID (selection)  
  
  for \$loopid in id (all) loop main  
    show element (x) (id \$loopid)  
  end loop main

## 7.8 Atom selection

- select atoms for certain operations
- selection by atom name
- wildcards and ranges
- selection by atom property
- different “queries” can be connected by AND / OR
- “queries” can be negated by NOT
- parantheses necessary for combinations of AND, OR, NOT

## 7.9 Selection by atom name

- The atom name consists of
  - SEGId, segment name defined by SEGMENT
  - RESId, residue “number” (also 48b etc!)
  - RESName, residue name (ALA, VAL...)
  - NAME, atom name (N, CA...)

```
coor select (resid 5 and name hn)
...
end
coor select ((resid 5 or resid 7) and name hn)
...
end
coor select (resid 5:7 and not name h*)
...
end
```
- Atoms can also be selected by
  - CHEM (atom type defined in topology)
  - ID (internal number)

## 7.10 Wildcards and ranges

- wildcards and ranges can be used for
  - SEGId
  - RESId
  - RESName
  - NAME
  - CHEM
- ranges are lexicographical order
- indicated by “:”

```
coor sele (name ha:hg#) ... end
```

```
selects ha, hb1, hb2, hg1, hg2, hd1, hd2, he1, he2
```

- wildcard hierarchy
  - “\*” any string (abcd, 78, 8u)
  - “#” any number (2, 43, 39987)
  - “%” any character (a, 6, j)
  - “+” any digit (0, 1, ... 9)

## 7.11 Selection by atom property

- ATTRibute selects on any atom property  
(coordinates, derivatives, mass, charge, ...)

```
coor sele (attribute charge > 0) ... end
```

- AROUnd, SAROund select atoms within cutoff of specified atoms

```
coor sele ((resid 1 and name ca) around 5.0)
...
end
```

SAROund selects atoms also in symmetry mates

- POINt ... CUT selects atoms around point

```
coor sele (point (3.0 4.0 5.0) cut 5.0) ... end
```

## 7.12 BYREsidue and BYGRp

- BYREsidue (selection)  
selects all atoms in a residue

```
coor sele= (byres( point (0 0 0) cut 5.0) )
```

## 7.13 STOREi and RECALLi

- Atom selections can be stored and used later

```
iden (store1) (name ca)

coor sele= (store1) ...
coor sele= (recall1) ...
```

## 7.14 Do, show and identity statements

These statements allow analysis and manipulation of atom properties and names.

- SHOW ELEMent (AtomArray) (selection)  
lists elements and defines \$result
- SHOW AVERAge (AtomArray) (selection)  
SHOW RMS (AtomArray) (selection)  
SHOW SUM (AtomArray) (selection)  
SHOW NORM (AtomArray) (selection)

```
show element (resid)
    (name ca and (resid 5 and name ca) around 5.0)

show average (x) (name ca)
```

- DO (expression) (selection)  
  
do (b = b + x<sup>2</sup> + y<sup>2</sup> + z<sup>2</sup>) (all)
- IDEN (STOREi) (selection)  
defines a STORE to be used in atom selection later

## 7.15 3-D vectors and matrices

- 3D vectors can be defined explicitly, or through atom selections

```

coor translate vector= (1 0 0) end
coor translate
    vector= (head=(resid 1 and name cb)
            tail=(resid 1 and name ca))
    distance= 5.0

```

- 3x3 matrices can be defined by

```

coor rotate
    center= (0 0 0)
    matrix= AXIS (head=(resid 1 and name cb)
                tail=(resid 1 and name ca))
    90.0

```

- or by Euler angles, Lattman angles, Quaternions, Spherical angles
- or explicitly

```

coor rotate
    center= (0 0 0)
    matrix= (1 0 0) (0 1 0) (0 0 1)

```

## 7.16 Output files

- DISPLAY files  
for DISPlay statements  
open with SET DISPlay filename END
- PRINT files  
for info from PRINt statements (e.g. PRINt ANGLes)  
open with SET PRINt filename END
- coordinate, structure, parameter files  
with WRITE COOR (structure...) OUTPut= filename end
- trajectory files

## 7.17 modules

modules are include files (subroutines) that perform a certain task.

```
module { restraintnumber }
(
  &num=num;
)
{- NOE restraint number -}
  evaluate ($number=0.)
  noe print threshold = 500.0 end
  evaluate (&num.noe = $number)
```

## 7.18 Examples

```
set display rmsd.disp end

evaluate ($maxcount = 10)

evaluate ($count = 1)
for $filename in ( @@file.list ) loop fit

  coor @$filename
  if ($count eq 1) then
    coor copy end
  end if

  coor sele (name ca) fit end
  coor sele (name ca) rms end

  display structure $count $filename diff $result A

  if ($count ge $maxcount) then
    exit loop fit
  end if

  evaluate ($count = $count + 1)
end loop fit
```

file.list contains a list of files, for example ordered by energy

```
"sa_1.pdb"  
"sa_6.pdb"  
...  
"sa_67.pdb"
```

The display file rmsd.disp will look like this

```
structure 1 sa_1.pdb rms difference 0 A  
structure 2 sa_6.pdb rms difference 1.245 A  
...  
structure 10 sa_67.pdb rms difference 1.87 A  
  
set display rmsfluc.disp end  
  
for $loopid in id (name ca) loop rms  
  
    show element (resid) (id $loopid)  
    evaluate ($resid = $result)  
    show element (resn) (id $loopid)  
    evaluate ($resn = $result)  
    show norm (b) (byresidue(id $id) and not hydro)  
    evaluate ($rmsfluc = $result)  
  
    display $resn $resid $rmsfluc  
  
end loop rms
```

# Bibliography

- [1] M. Nilges. A calculation strategy for the structure determination of symmetric dimers by  $^1\text{H}$ . *Proteins*, 17:297–309, 1993.
- [2] M. Nilges. Calculation of protein structures with ambiguous distance restraints. Automated assignment of ambiguous NOE crosspeaks and disulfide connectivities. *J. Mol. Biol.*, 245:645–660, 1995.
- [3] M. Nilges, M. J. Macias, S. I. O’Donoghue, and H. Oschkinat. Automated NOESY interpretation with ambiguous distance restraints: the refined NMR solution structure of the pleckstrin homology domain from  $\beta$ -spectrin. *J. Mol. Biol.*, 269:408–422, 1997.
- [4] J. P. Linge and M. Nilges. Influence of non-bonded parameters on the quality of NMR structures: a new force-field for NMR structure calculation. *J. Biomol. NMR*, 13:51–59, 1999.
- [5] S. I. O’Donoghue and M. Nilges. Calculation of symmetric oligomer structures from NMR data. In R. Krishna and J. L. Berliner, editors, *Structure computation and dynamics in protein NMR*, volume 17 of *Biological Magnetic Resonance*, pages 131–161. Kluwer Academic/ Plenum, New York, 1999.
- [6] J. P. Linge. *New methods for automated NOE assignment and NMR structure calculation*. Book on demand Verlag, Norderstedt, Germany, 2001.
- [7] J. P. Linge, S. I. O. O’Donoghue, and M. Nilges. Assigning ambiguous NOEs with ARIA., 2001.
- [8] Z. Liu, M. J. Macias, M. J. Bottomley, G. Stier, J. P. Linge, M. Nilges, P. Bork, and M. Sattler. The three-dimensional structure of the HRDC

domain and implications for the Werner and Bloom syndrome proteins. *Folding & Design*, 7(12):1557–1566, Dec 15 1999.

- [9] R. Sprangers, M. J. Bottomley, J. P. Linge, J. Schultz, M. Nilges, and M. Sattler. Refinement of the protein backbone angle  $\psi$  in structure calculations. *J. Biomol. NMR*, 16(1):47–58, Jan 2000.