relax

Version respository checkout

A program for NMR relaxation data analysis

Edward d'Auvergne

January 11, 2006

Acknowledgements

Firstly, I am grateful and would like to thank Paul Gooley for giving me the opportunity to create this program as part of my PhD. The duration of development has been considerable and I thank Paul for allowing me the time to learn Python and to implement many data analysis concepts into relax.

I would also like to thank Andrew James Perry for innumerable, thoughtful discussions on both the science and programming aspects behind relax. Many of the important design features of relax originated as ideas of his and much of the usability and core layout came from these discussions. Examples include the prompt based interface with the help system and tab completion, the automatic generation of the documentation on the user functions by parsing their docstrings, details of the interaction with and control of other programs, and the powerful Python scripting. Without Andrew, this program would not be what it is today.

Finally I would like to thank Haydyn D.T. Mertens and James D. Swarbrick for many discussions and feedback.

Contents

List of Figures

Abbreviations

AIC Akaike's Information Criteria

AICc small sample size corrected AIC

BIC Bayesian Information Criteria

$C(\tau)$ correlation function

χ^2 chi-squared function

CSA chemical shift anisotropy

$\mathfrak{D}$ the set of diffusion tensor parameters

$\mathfrak{D}_{\parallel}$ the eigenvalue of the spheroid diffusion tensor corresponding to the unique axis of the tensor

$\mathfrak{D}_{\perp}$ the eigenvalue of the spheroid diffusion tensor corresponding to the two axes perpendicular to the unique axis

$\mathfrak{D}_a$ the anisotropic component of the Brownian rotational diffusion tensor

$\mathfrak{D}_{iso}$ the isotropic component of the Brownian rotational diffusion tensor

$\mathfrak{D}_r$ the rhombic component of the Brownian rotational diffusion tensor

$\mathfrak{D}_{ratio}$ the ratio of $\mathfrak{D}_{\parallel}$ to $\mathfrak{D}_{\perp}$

$\mathfrak{D}_x$ the eigenvalue of the Brownian rotational diffusion tensor in which the corresponding eigenvector defines the x-axis of the tensor

$\mathfrak{D}_y$ the eigenvalue of the Brownian rotational diffusion tensor in which the corresponding eigenvector defines the y-axis of the tensor

$\mathfrak{D}_z$ the eigenvalue of the Brownian rotational diffusion tensor in which the corresponding eigenvector defines the z-axis of the tensor

ϵ_i elimination value

$J(\omega)$ spectral density function

NOE nuclear Overhauser effect

pdf probability distribution function

r bond length

$\mathbf{R}_1$ spin-lattice relaxation rate

$\mathbf{R}_2$ spin-spin relaxation rate

R_{ex} chemical exchange relaxation rate

S^2, S_f^2, **and** S_s^2 model-free generalised order parameters

τ_e, τ_f, **and** τ_s model-free effective internal correlation times

τ_m global rotational correlation time

Chapter 1

Installation instructions

1.1 Dependencies

The following packages need to be installed before using relax:

Python: Version 2.2 or higher .

Numeric: Version 21 or higher .

ScientificPython: Version 2.2 or higher .

Optik: Version 1.4 or higher. This is only needed if running python 2.2 .

Older versions of these packages may work, use them at your own risk.

1.2 Installation

1.2.1 Linux precompiled distribution

To install the program relax on a GNU/Linux system, download the precompiled distribution labelled `relax-x.x.x.Linux.i386.tar.bz2`. This tar file has had the C code precompiled into shared objects files `*.so` which are loaded directly into relax. Hence compilation is not necessary. Untar and decompress the file using the command

```
$ tar jxvf relax-x.x.x.Linux.i386.tar.bz2
```

Then in the base directory `relax`, switch user to root and type the command

```
$ make install
```

This will create a directory in `/usr/local/` called `relax`, copy all the untarred files into this directory, create a symbolic link in `/usr/local/bin` to the file `/usr/local/relax/relax`, and then finally run relax to create the byte-compiled Python `*.pyc` files to speed up the start time of relax.

To change the installation path, modify the file `Makefile` and change the variable `INSTALL_PATH` to point to the desired location.

1.2.2 Source distribution

The source distribution should be named `relax-x.x.x.src.tar.bz2`. The make utility and a C compiler are required for compilation of the C code into shared objects which are loaded as modules into relax. To build these modules, type

```
$ make
```

Then to install the program, type

```
$ make install
```

The default installation path is currently `/usr/local/`. To change this, edit the `Makefile` and modify the variable `INSTALL_PATH`.

1.2.3 Running a non-compiled version

Compilation of the C code is not essential for running relax, however, certain features of the program will be disabled. One example is relaxation curve fitting. This approach may be necessary for systems, such as MS Windows, where C compilers are not readily available (although the cygwin environment may provide the tools required for compilation).

To run relax without compilation, install the dependencies detailed above, download the source distribution which should be named `relax-x.x.x.src.tar.bz2`, extract the files, and then run the file called `relax` in the base directory.

1.3 Optional programs

The following is a list of programs which can be used by relax, although they are not essential for normal use.

1.3.1 Grace

Grace is a program for plotting two dimensional data sets in a professional looking manner. It is used to visualise parameter values. It can be downloaded from `http://plasma-gate.weizmann.ac.il/Grace/`.

1.3.2 OpenDX

Version 4.1.3 or compatible. OpenDX is used for viewing the output of the space mapping function, and is executed by passing the command `dx` to the command line with various options. The program is designed for visualising multidimensional data and can be found at `http://www.opendx.org/`.

1.3.3 Molmol

Molmol is used for viewing the PDB structures loaded into the program and to display parameter values mapped onto the structure.

1.3.4 Dasha

Dasha is a program used for model-free analysis of NMR relaxation data. It can be used as an optimisation engine to replace the minimisation algorithms implemented within relax.

1.3.5 Modelfree4

Art Palmer's Modelfree4 program is also designed for model-free analysis and can be used as an optimisation engine to replace relax's high precision minimisation algorithms.

Chapter 2

How to use relax

2.1 The prompt

The primary interface of relax is the prompt. After typing `'relax'` within a terminal, you will be presented with

```
relax>
```

This is the Python prompt which has been tailored specifically for relax. You will hence have full access, if necessary, to the power of the Python programing language to manipulate your data. You can for instance type

```
relax> print "Hello World"
```

the result being

```
relax> print "Hello World"
Hello World
relax>
```

Or, using relax as a calculator

```
relax> (1.0 + (2 * 3))/10
0.69999999999999996
relax>
```

2.2 Python

relax has been designed such that knowledge about Python is not required to be able to fully use the program. A few basics, though, will aid in understanding relax.

A number of simple programming axioms includes that of strings , integers , floating point numbers , and lists . A string is text and within Python (as well as relax) this is delimited by either single or double quotes. An integer is a number with no decimal point while

a float is a number with a decimal point. A list in Python (called an array in other languages) is a list of anything separated by commas and delimited by square brackets, an example is [0, 1, 2, 'a', 1.2143235].

Probably the most important detail is that functions in Python require brackets around their arguments. For example

```
relax> minimise()
```

will commence minimisation however

```
relax> minimise
```

will do nothing.

The arguments to a function are simply a comma separated list within the brackets of the function. For example to save the program's current state, type

```
relax> state.save('save', force=1)
```

Two types of arguments exist in Python , standard arguments and keyword arguments. The majority of arguments you will encounter within relax are keyword arguments however you may, in rare cases, encounter a non-keyword argument. For these standard arguments, just type the values in, although they must be in the correct order. Keyword arguments consist of two parts, the key and the value. For example the key may be `file` while the value you would like to supply is `'R1.out'`. Various methods exist for supplying this argument. Firstly you could simply type `'R1.out'` into the correct position in the argument list. Secondly you can type `file='R1.out'`. The power of this second option is that argument order is unimportant. Therefore if you would like to change the default value of the very last argument, you don't have to supply values for all other arguments. The only catch is that standard arguments must come before the keyword arguments.

2.3 User functions

For standard data analysis, a large number of specially tailored functions called 'user functions' have been implemented. These are accessible from the relax prompt by simply typing the name of the function. An example is `'help()'` . An alphabetical listing of all accessible user functions together with full descriptions is presented later within this manual.

A few special objects which are available within the prompt are not actually functions. These objects do not require brackets at their end for them to function. For example to exit relax type

```
relax> exit
```

Another special object is that of the function class . This object is simply a container which holds a number of user functions. You can access the user function within the class by typing the name of the class, then a dot `'.'`, followed by the name of the user function. An example is the user function for reading relaxation data out of a file and loading the data into relax. The function is called `'read'` while the class is called `'relax_data'`. To execute the function, type something like

```
relax> relax_data.read(name, 'R1', '600', 600.0 * 1e6, 'r1.600.out')
```

The relax prompt, on first usage, can be quite daunting. Two features exist, however, to increase the usability of the prompt – the help system and tab completion

2.4 The help system

For assistance in using a function, simply type

```
help(function)
```

In addition to functions, if

```
help(object)
```

is typed, the help for the python object is returned. This system is similar to the help function built into the python interpreter, which has been renamed to `help_python`, with the interactive component removed. For the interactive python help system, type

```
help_python()
```

2.5 Tab completion

Tab completion is implemented to prevent insanity as the function names can be quite long – a deliberate feature to improve usability. The behaviour of the tab completion is very similar to that of the bash prompt.

Not only is tab completion useful for preventing RSI, but it can also be used for finding out what functions are available. To begin with if you hit the [TAB] key without typing any text, all available functions will be listed (along with function classes and other python objects). This extends to the exploration of user functions within a function class . For example, to list the user functions within the function class `'model_free'`, type

```
relax> model_free.
```

The dot character at the end is essential. After hitting the [TAB] key, you should see something like

```
relax> model_free.
model_free.__class__
model_free.__doc__
model_free.__init__
model_free.__module__
model_free.__relax__
model_free.__relax_help__
model_free.copy
model_free.create_model
model_free.delete
model_free.remove_tm
model_free.select_model
```

```
relax> model_free.
```

All the objects beginning with an underscore are "hidden", they contain information about the function class and should be ignored. From the listing the user functions 'copy', 'create_model', 'delete', 'remove_tm', and 'select_model' contained within 'model_free' are all visible.

2.6 The 'run'

Within relax, the majority of operations are assigned to a special construct called a 'run'. For example, to load relaxation data into the program it must be committed to a pre-created 'run'. Within one instance of relax, multiple runs can be created and various operations performed in sequence on these runs. This is useful for operations such as model selection whereby the function 'model_selection' can operate on a number of runs corresponding to different models and then assign the results to a newly created run.

The flow of data through relax can be thought of as travelling through pipes – each pipe is synonymous with a run. User functions exist to transfer data between these pipes, while other functions combine data from multiple pipes into one or vice-verse. The simplest invocation of relax would be the creation of a single run and with the data being processed as it is passing through this pipe.

The primary method for creating a run is through the user function 'run.create'. For example

```
relax> run.create('m1', 'mf')
```

will create a run called 'm1'. The run is also associated with a type which, in this case, is model-free analysis. The following is a table of all the types which can be assigned to a run.

Run type	Description
'jw'	Reduced spectral density mapping
'mf'	Model-free data analysis
'noe'	Steady state NOE calculation
'relax_fit'	Relaxation curve fitting
'srls'	SRLS analysis

Currently only the NOE calculation, model-free analysis, and reduced spectral density mapping features of relax are implemented (if this documentation is out of date, then you may be able to do a lot more).

2.7 Scripting

What ever is done within the prompt is also accessible through scripting. Just type your commands into a text file and then at the terminal type

```
$ relax your_script
```

An example of a simple script which will minimise the model-free model ‘m4’ after loading six relaxation data sets is

```
# Create the run.
name = ‘m4’
run.create(name, ‘mf’)

# Nuclei type
nuclei(‘N’)

# Load the sequence.
sequence.read(name, ‘noe.500.out’)

# Load the relaxation data.
relax_data.read(name, ‘R1’, ‘600’, 600.0 * 1e6, ‘r1.600.out’)
relax_data.read(name, ‘R2’, ‘600’, 600.0 * 1e6, ‘r2.600.out’)
relax_data.read(name, ‘NOE’, ‘600’, 600.0 * 1e6, ‘noe.600.out’)
relax_data.read(name, ‘R1’, ‘500’, 500.0 * 1e6, ‘r1.500.out’)
relax_data.read(name, ‘R2’, ‘500’, 500.0 * 1e6, ‘r2.500.out’)
relax_data.read(name, ‘NOE’, ‘500’, 500.0 * 1e6, ‘noe.500.out’)

# Setup other values.
diffusion_tensor.set(name, (2e-8, 1.3, 60, 290), param_types=1, axial_type=‘prolate’,
fixed=1)
value.set(name, 1.02 * 1e-10, ‘bond_length’)
value.set(name, -160 * 1e-6, ‘csa’)

# Select a preset model-free model.
model_free.select_model(run=name, model=name)

# Grid search.
grid_search(name, inc=11)

# Minimise.
minimise(‘newton’, run=name)

# Finish.
results.write(run=name, file=‘results’, force=1)
state.save(‘save’, force=1)
```

Scripting is much more powerful than the prompt as advanced Python programming can be employed (see the file ‘full_analysis.py’ in the ‘sample_scripts’ directory for an example).

2.8 Sample scripts

A few sample scripts have been provided in the directory 'sample_scripts'. These can be used as a good starting point for using relax.

2.9 The GUI

relax has been designed primarily for scripting and, as such, no graphical user interface (GUI) currently exists. The internal structure of the program has been specifically designed so any type of control mechanism can be easily added, including a GUI, therefore in the future one may be written. A GUI will, however, detract from the power and flexibility inherent in the control by scripting.

2.10 Access to the internals of relax

For highly advanced Python scripting or control of relax, almost every part of relax has been designed in an object oriented fashion. If you would like to play with internals of the program, the entirety of relax is accessible within the object called `'self.relax'`. To access the raw objects within relax which contain the program data, all data is stored within the object called `'self.relax.data'`.

2.11 Usage of the name relax

The program relax is so relaxed that the first letter should always be in lower case!

Chapter 3

Data analysis

3.1 Introduction

This chapter aims to explain not only the steps involved in the data analysis of relaxation data but also how to use the program relax to achieve this. Although a work in progress, it covers how to calculate the NOE, how to optimise and find the R_1 and R_2 relaxation rates, and how to implement model-free analysis.

3.2 Calculating the NOE

The calculation of NOE values is a straight forward and quick procedure which involves two components, the calculation of the value itself and the calculation of the errors. To understand the steps involved, we will follow in detail the execution of a sample NOE calculation script.

The sample script

```
# Script for calculating NOEs.

# Create the run
name = ‘noe’
run.create(name, ‘noe’)

# Load the sequence from a PDB file.
pdb(name, ‘Ap4Aase_new_3.pdb’, load_seq=1)

# Load the reference spectrum and saturated spectrum peak intensities.
noe.read(name, file=‘ref.list’, spectrum_type=‘ref’)
noe.read(name, file=‘sat.list’, spectrum_type=‘sat’)

# Set the errors.
noe.error(name, error=3600, spectrum_type=‘ref’)
```

```
noe.error(name, error=3000, spectrum_type='sat')

# Individual residue errors.
noe.error(name, error=122000, spectrum_type='ref', res_num=114)
noe.error(name, error=8500, spectrum_type='sat', res_num=114)

# Unselect unresolved residues.
unselect.read(name, file='unresolved')

# Calculate the NOEs.
calc(name)

# Save the NOEs.
value.write(name, param='noe', file='noe.out', force=1)

# Create grace files.
grace.write(name, y_data_type='ref', file='ref.agr', force=1)
grace.write(name, y_data_type='sat', file='sat.agr', force=1)
grace.write(name, y_data_type='noe', file='noe.agr', force=1)

# View the grace files.
grace.view(file='ref.agr')
grace.view(file='sat.agr')
grace.view(file='noe.agr')

# Write the results.
results.write(name, file='results', dir=None, force=1)

# Save the program state.
state.save('save', force=1)
```

Initialisation of the run

Firstly to simplify referencing of the run name in the relevent functions, the name `'noe'` is assigned to to the object `name` by the command

```
name = 'noe'
```

Therefore instead of typing `'noe'` each time the run needs to be referenced, `name` can be used instead. The run is created by the command

```
run.create(name, 'noe')
```

This user function will then create a run which is named `'noe'`, the second argument setting the run type to that of calculating the NOE. Setting the run type is important so that the program knows which user functions are compatible with the run, for example the function `minimise()` is meaningless in this sample script as the NOE values are computed by direct calculation rather than through optimisation.

Loading the data

The first thing which need to be done prior to any residue specific command is to load the sequence. In this case, the command

```
pdb(name, 'Ap4Aase_new_3.pdb', load_seq=1)
```

will extract the sequence from the PDB file 'Ap4Aase_new_3.pdb'. The first argument specifies the run into which the sequence will be loaded, the second specifies the file name, while the third causes the function to extract the sequence rather than just load the PDB into relax. Although the PDB coordinates have been loaded into the program, the structure serves no purpose when calculating NOE values.

The next two commands

```
noe.read(name, file='ref.list', spectrum_type='ref')
noe.read(name, file='sat.list', spectrum_type='sat')
```

load the peak heights of the reference and saturated NOE experiments (although the volume could be used instead). The keyword argument `format` has not been specified, hence the default format of a Sparky peak list (saved after typing `'lt'`) is assumed. If the program XEasy was used to analyse the spectra, the argument `format='xeasy'` is necessary. The first column of the file should be the Sparky assignment string, while it is assumed that the 4^{th} column contains either the peak height.

Setting the errors

In this example, the errors where measured from the base plain noise. The Sparky RMSD function was used to estimate the maximal noise levels across the spectrum in regions containing no peaks. For the reference spectrum the RMSD was approximately 3600 while in the saturated spectrum the RMSD was 3000. These errors are set by the commands

```
noe.error(name, error=3600, spectrum_type='ref')
noe.error(name, error=3000, spectrum_type='sat')
```

For the residue G114, the noise levels are significantly increased compared to the rest of the protein as the peak is located close to the water signal. The higher errors for this residue are specified by the commands

```
noe.error(name, error=122000, spectrum_type='ref', res_num=114)
noe.error(name, error=8500, spectrum_type='sat', res_num=114)
```

Unresolved residues

As the peaks of certain residues overlaps to such an extent that the heights of both cannot be resolved, a simple text file was created called `unresolved` in which each line consists of a single residue number. By using the command

```
unselect.read(name, file='unresolved')
```

all residues in the file `unresolved` are excluded from the analysis.

The NOE

At this point, the NOE can be calculated. The user function

```
calc(name)
```

will calculate both the NOE and the errors. The NOE value will be calculated using the formula

$$NOE = \frac{I_{sat}}{I_{ref}}, \tag{3.1}$$

where I_{sat} is the intensity of the peak in the saturated spectrum while I_{ref} is that of the reference spectrum. The error is calculated by

$$\sigma_{NOE} = \sqrt{\frac{(\sigma_{sat} \cdot I_{ref})^2 + (\sigma_{ref} \cdot I_{sat})^2}{I_{ref}}}, \tag{3.2}$$

where σ_{sat} and σ_{ref} are the peak intensity errors in the saturated and reference spectra respectively. To create a file of the NOEs, the command

```
value.write(name, param='noe', file='noe.out', force=1)
```

will create a file called `noe.out` with the NOE values and errors. The force flag will cause any file with the same name to be overwritten. An example of the format of `noe.out` is

```
Num  Name  Value                 Error
1    GLY   None                  None
2    PRO   None                  None
3    LEU   None                  None
4    GLY   0.12479588727508535   0.020551827436105764
5    SER   0.42240815792914105   0.02016346825976852
6    MET   0.45281703194372114   0.026272719841642134
7    ASP   0.60727570079478255   0.032369427242382849
8    SER   0.63871921623680161   0.024695665815261791
9    PRO   None                  None
10   PRO   None                  None
11   GLU   None                  None
12   GLY   0.92927160307645906   0.059569089743604184
13   TYR   0.88832516377296256   0.044119641308479306
14   ARG   0.84945042565860407   0.060533543601110441
```

Viewing the results

Any two dimensional data set can be plotted in relax in conjunction with the program Grace (http://plasma-gate.weizmann.ac.il/Grace/). The program is also known as Xmgrace and was previously known as ACE/gr or Xmgr. The highly flexible relax user function `grace.write` is capable of producing 2D plots of any x-y data sets. The three commands

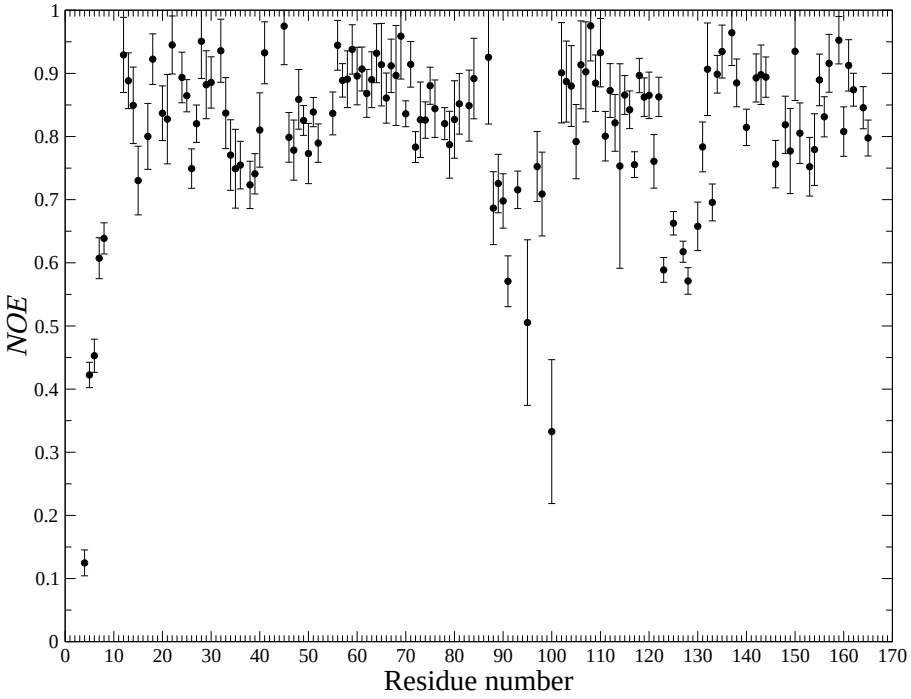

Figure 3.1: A Grace plot of the NOE value and error against the residue number. An example of the output of the user function `grace.write()`.

```
grace.write(name, y_data_type='ref', file='ref.agr', force=1)
grace.write(name, y_data_type='sat', file='sat.agr', force=1)
grace.write(name, y_data_type='noe', file='noe.agr', force=1)
```

create three separate plots of the peak intensity of the reference and saturated spectra as well as the NOE. The x-axis in all three defaults to the residue number. As the x and y-axes can be any parameter, the command

```
grace.write(name, x_data_type='ref', y_data_type='sat', file='ref_vs_sat.agr', force=1)
```

would create a plot of the reference verses the saturated intensity, with one point per residue. Returning to the sample script, three Grace data files are created, `ref.agr`, `sat.agr`, and `noe.agr` and placed in the default directory `./grace`. These can be visualised by opening the file within Grace, however relax will do that for you with the commands

```
grace.view(file='ref.agr')
grace.view(file='sat.agr')
grace.view(file='noe.agr')
```

An example of the output, after modifying the axes, is shown in figure 3.1.

3.3 The R_1 and R_2 relaxation rates - relaxation curve fitting

3.4 Model-free analysis—textbf

3.5 Reduced spectral density mapping

Chapter 4

Values, gradients, and Hessians

4.1 Introduction

A word of warning before reading this chapter, the topics covered here are quite advanced and are not necessary for understanding how to either use relax or to implement any of the data analysis techniques present within relax. The material of this chapter is intended as an in-depth explanation of the mathematics involved in the optimisation of the parameters of the model-free models. As such it contains the chi-squared equation, relaxation equations, spectral density functions, and diffusion tensor equations as well as their gradients (first partial derivatives) and Hessians (second partial derivatives).

4.1.1 Chi-squared function – $\chi^2(\theta)$

For the minimisation of models, a chain of calculations using different theories is required. At the highest level, the function which is actually minimised is the chi-squared function

$$\chi^2(\theta) = \sum_{i=1}^{n} \frac{(\mathrm{R}_i - \mathrm{R}_i(\theta))^2}{\sigma_i^2}, \tag{4.1}$$

where the index i is the summation index ranging over all the experimentally collected relaxation data, R_i, including the R_1, R_2, and NOE data at all field strengths belonging to the set data set R for an individual residue, a collection of residues, or the entire macromolecule, $\mathrm{R}_i(\theta)$ is the back calculated relaxation data belonging to the set $\mathrm{R}(\theta)$, θ is the model parameter vector which, when minimised, is denoted by the symbol $\hat{\theta}$, and σ_i are the experimental errors.

The significance of equation (4.1) is that the function returns a single value.

4.1.2 Relaxation equations – $\mathrm{R}_i(\theta)$

The chi-squared equation is itself dependent on the relaxation equations through the back calculated relaxation data $\mathrm{R}(\theta)$. These data are the spin-lattice , spin-spin , and cross-

relaxation rates of Abragam (1961) and are respectively

$$\mathrm{R}_1(\theta) = d\Big(J(\omega_H - \omega_X) + 3J(\omega_X) + 6J(\omega_H + \omega_X)\Big) + cJ(\omega_X), \tag{4.2}$$

$$\begin{aligned}\mathrm{R}_2(\theta) = \frac{d}{2}\Big(&4J(0) + J(\omega_H - \omega_X) + 3J(\omega_X) + 6J(\omega_H) \\ &+ 6J(\omega_H + \omega_X)\Big) + \frac{c}{6}\Big(4J(0) + 3J(\omega_X)\Big) + R_{ex},\end{aligned} \tag{4.3}$$

$$\sigma_{\mathrm{NOE}}(\theta) = d\Big(6J(\omega_H + \omega_X) - J(\omega_H - \omega_X)\Big), \tag{4.4}$$

where $J(\omega)$ is the power spectral density function, R_{ex} is the relaxation due to chemical exchange, and the dipolar and CSA constants are defined in SI units as

$$d = \frac{1}{4}\left(\frac{\mu_0}{4\pi}\right)^2 \frac{(\gamma_H \gamma_X \hbar)^2}{\langle r^6 \rangle}, \tag{4.5}$$

$$c = \frac{(\omega_H \Delta\sigma)^2}{3}, \tag{4.6}$$

where μ_0 is the permeability of free space, γ_H and γ_X are the gyromagnetic ratios of the H and X spins respectively, $\hbar$ is Plank's constant divided by 2π, r is the bond length, and $\Delta\sigma$ is the chemical shift anisotropy measured in ppm. The cross-relaxation rate σ_{NOE} is related to the steady state NOE by the equation

$$\mathrm{NOE}(\theta) = 1 + \frac{\gamma_H}{\gamma_X}\frac{\sigma_{\mathrm{NOE}}(\theta)}{\mathrm{R}_1(\theta)}. \tag{4.7}$$

4.1.3 Transformed relaxation equations – $\mathrm{R}'_i(\theta)$

Letting the relaxation equations, $\mathrm{R}_i(\theta)$ be the $\mathrm{R}_1(\theta)$, $\mathrm{R}_2(\theta)$, and $\mathrm{NOE}(\theta)$, an additional layer of abstraction can be used to simply the calculation of the gradients and Hessians. This involved decomposing the NOE equation into the cross relaxation rate constant $\sigma_{\mathrm{NOE}}(\theta)$ and the auto relaxation rate $\mathrm{R}_1(\theta)$. Taking equation (4.7), the relaxation equations are

$$\mathrm{R}'_1(\theta) = \mathrm{R}_1(\theta) \tag{4.8}$$

$$\mathrm{R}'_2(\theta) = \mathrm{R}_2(\theta) \tag{4.9}$$

$$\mathrm{NOE}(\theta) = 1 + \frac{\gamma_H}{\gamma_X}\frac{\sigma_{\mathrm{NOE}}(\theta)}{\mathrm{R}_1(\theta)}. \tag{4.10}$$

while the transformed relaxation equations are $\{\mathrm{R}'_1(\theta), \mathrm{R}'_2(\theta), \sigma_{\mathrm{NOE}}(\theta)\}$.

4.1.4 Spectral density functions – $J(\omega)$

The relaxation equations are themselves dependent on the calculation of the spectral density values $J(\omega)$. Within model-free analysis, these are modelled by the original model-free formula (Lipari and Szabo, 1982a,b)

$$J(\omega) = \frac{2}{5}\sum_{i=-k}^{k} c_i \cdot \tau_i \left(\frac{S^2}{1 + (\omega\tau_i)^2} + \frac{(1 - S^2)(\tau_e + \tau_i)\tau_e}{(\tau_e + \tau_i)^2 + (\omega\tau_e\tau_i)^2}\right), \tag{4.11}$$

where S^2 is the square of the Lipari and Szabo generalised order parameter and τ_e is the effective correlation time. The order parameter reflects the amplitude of the motion while the correlation time in an indication of the time scale of that motion. The theory was extended by Clore et al. (1990) by the modelling of two independent internal motions using the equation

$$J(\omega) = \frac{2}{5}\sum_{i=-k}^{k} c_i \cdot \tau_i \left(\frac{S^2}{1+(\omega\tau_i)^2} + \frac{(1-S_f^2)(\tau_f+\tau_i)\tau_f}{(\tau_f+\tau_i)^2+(\omega\tau_f\tau_i)^2} + \frac{(S_f^2-S^2)(\tau_s+\tau_i)\tau_s}{(\tau_s+\tau_i)^2+(\omega\tau_s\tau_i)^2} \right). \quad (4.12)$$

where S_f^2 and τ_f are the amplitude and timescale of the faster of the two motions while S_s^2 and τ_s are those of the slower motion. S_f^2 and S_s^2 are related by the formula $S^2 = S_f^2 \cdot S_s^2$.

4.1.5 Brownian diffusion

In equations (4.11) and (4.12), the generic Brownian diffusion NMR correlation function presented in **?** has been used. This function is

$$C(\tau) = \frac{1}{5}\sum_{i=-k}^{k} c_i \cdot e^{-\tau/\tau_i}, \quad (4.13)$$

where the summation index i ranges over the number of exponential terms within the correlation function. This equation is generic in that it describes the diffusion of an ellipsoid, a spheroid, and a sphere.

Ellipsoid

For the ellipsoid defined by the parameter set $\{\mathfrak{D}_{iso}, \mathfrak{D}_a, \mathfrak{D}_r, \alpha, \beta, \gamma\}$ the variable k is equal to two, therefore the index $i \in \{-2,-1,0,1,2\}$. The geometric parameters $\{\mathfrak{D}_{iso}, \mathfrak{D}_a, \mathfrak{D}_r\}$ are defined as

$$\mathfrak{D}_{iso} = \tfrac{1}{3}(\mathfrak{D}_x + \mathfrak{D}_y + \mathfrak{D}_z), \quad (4.14a)$$

$$\mathfrak{D}_a = \mathfrak{D}_z - \tfrac{1}{2}(\mathfrak{D}_x + \mathfrak{D}_y), \quad (4.14b)$$

$$\mathfrak{D}_r = \frac{\mathfrak{D}_y - \mathfrak{D}_x}{2\mathfrak{D}_a}, \quad (4.14c)$$

and are constrained by

$$0 < \mathfrak{D}_{iso} < \infty, \quad (4.15a)$$

$$0 \le \mathfrak{D}_a < \frac{\mathfrak{D}_{iso}}{\frac{1}{3}+\mathfrak{D}_r} \le 3\mathfrak{D}_{iso}, \quad (4.15b)$$

$$0 \le \mathfrak{D}_r \le 1. \quad (4.15c)$$

The orientational parameters $\{\alpha,\ \beta,\ \gamma\}$ are the Euler angles using the z-y-z rotation notation.

The five weights c_i are defined as

$$c_{-2} = \tfrac{1}{4}(d+e), \tag{4.16a}$$
$$c_{-1} = 3\delta_y^2\delta_z^2, \tag{4.16b}$$
$$c_0 = 3\delta_x^2\delta_z^2, \tag{4.16c}$$
$$c_1 = 3\delta_x^2\delta_y^2, \tag{4.16d}$$
$$c_2 = \tfrac{1}{4}(d-e), \tag{4.16e}$$

where

$$d = 3\left(\delta_x^4 + \delta_y^4 + \delta_z^4\right) - 1, \tag{4.17}$$
$$e = -\frac{1}{\mathfrak{R}}\left[(1+3\mathfrak{D}_r)\left(\delta_x^4 + 2\delta_y^2\delta_z^2\right) + (1-3\mathfrak{D}_r)\left(\delta_y^4 + 2\delta_x^2\delta_z^2\right) - 2\left(\delta_z^4 + 2\delta_x^2\delta_y^2\right)\right], \tag{4.18}$$

and where

$$\mathfrak{R} = \sqrt{1+3\mathfrak{D}_r^2}, \tag{4.19}$$

The five correlation times τ_i are

$$1/\tau_{-2} = 6\mathfrak{D}_{iso} - 2\mathfrak{D}_a\mathfrak{R}, \tag{4.20a}$$
$$1/\tau_{-1} = 6\mathfrak{D}_{iso} - \mathfrak{D}_a(1+3\mathfrak{D}_r), \tag{4.20b}$$
$$1/\tau_0 = 6\mathfrak{D}_{iso} - \mathfrak{D}_a(1-3\mathfrak{D}_r), \tag{4.20c}$$
$$1/\tau_1 = 6\mathfrak{D}_{iso} + 2\mathfrak{D}_a, \tag{4.20d}$$
$$1/\tau_2 = 6\mathfrak{D}_{iso} + 2\mathfrak{D}_a\mathfrak{R}. \tag{4.20e}$$

Spheroid

The variable k is equal to one in the case of the spheroid defined by the parameter set $\{\mathfrak{D}_{iso},\ \mathfrak{D}_a,\ \theta,\ \phi\}$, hence $i \in \{-1, 0, 1\}$. The geometric parameters $\{\mathfrak{D}_{iso},\ \mathfrak{D}_a\}$ are defined as

$$\mathfrak{D}_{iso} = \tfrac{1}{3}(\mathfrak{D}_\parallel + 2\mathfrak{D}_\perp), \tag{4.21a}$$
$$\mathfrak{D}_a = \mathfrak{D}_\parallel - \mathfrak{D}_\perp. \tag{4.21b}$$

and are constrained by

$$0 < \mathfrak{D}_{iso} < \infty, \tag{4.22a}$$
$$-\tfrac{3}{2}\mathfrak{D}_{iso} < \mathfrak{D}_a < 3\mathfrak{D}_{iso}, \tag{4.22b}$$

The orientational parameters $\{\theta,\ \phi\}$ are the spherical angles defining the orientation of the major axis of the diffusion frame within the lab frame.

The three weights c_i are

$$c_{-1} = \tfrac{1}{4}(3\delta_z^2 - 1)^2, \tag{4.23a}$$
$$c_0 = 3\delta_z^2(1 - \delta_z^2), \tag{4.23b}$$
$$c_1 = \tfrac{3}{4}(\delta_z^2 - 1)^2, \tag{4.23c}$$

The five correlation times τ_i are

$$1/\tau_{-1} = 6\mathfrak{D}_{iso} - 2\mathfrak{D}_a, \tag{4.24a}$$
$$1/\tau_0 = 6\mathfrak{D}_{iso} - \mathfrak{D}_a, \tag{4.24b}$$
$$1/\tau_1 = 6\mathfrak{D}_{iso} + 2\mathfrak{D}_a. \tag{4.24c}$$

Sphere

In the situation of a molecule diffusing as a sphere either described by the single parameter τ_m or $\mathfrak{D}_{iso}$, the variable k is equal to zero. Therefore $i \in \{0\}$. The single weight c_0 is equal to one, while the single correlation time τ_0 is equivalent to the global tumbling time τ_m given by

$$1/\tau_m = 6\mathfrak{D}_{iso}. \tag{4.25}$$

4.2 Minimisation concepts

4.2.1 The function value

At the simplest level, all minimisation techniques require at least a function which will supply a single value for different parameter values θ. For the modelling of NMR relaxation data, this is given by equation (4.1). For certain algorithms, such a Simplex minimisation , this single value suffices.

4.2.2 The gradient

The majority of minimisation algorithms, however, also require the gradient at the point of the given parameter values θ. The gradient is a vector of partial derivatives defined as

$$\nabla = \begin{pmatrix} \frac{\partial}{\partial\theta_1} \\ \frac{\partial}{\partial\theta_2} \\ \vdots \\ \frac{\partial}{\partial\theta_n} \end{pmatrix} \tag{4.26}$$

where n is the total number of parameters in the model.

An example of a powerful algorithm which requires both the value and gradient at given parameter values is BFGS quasi-Newton minimisation .

4.2.3 The Hessian

A few algorithms, including the most powerful, require in addition the Hessian at given parameter values θ. The Hessian is the matrix of second partial derivatives defined as

$$\nabla^2 = \begin{pmatrix} \frac{\partial^2}{\partial\theta_1^2} & \frac{\partial^2}{\partial\theta_1 \cdot \partial\theta_2} & \cdots & \frac{\partial^2}{\partial\theta_1 \cdot \partial\theta_n} \\ \frac{\partial^2}{\partial\theta_2 \cdot \partial\theta_1} & \frac{\partial^2}{\partial\theta_2^2} & \cdots & \frac{\partial^2}{\partial\theta_2 \cdot \partial\theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial\theta_n \cdot \partial\theta_1} & \frac{\partial^2}{\partial\theta_n \cdot \partial\theta_2} & \cdots & \frac{\partial^2}{\partial\theta_n^2} \end{pmatrix} \tag{4.27}$$

The most powerful minimisation algorithm, Newton minimisation , requires the value, gradient, and Hessian at the given parameter values.

4.3 Value, gradient, and Hessian dependency chain

The dependency chain outlined in the introduction to this chapter, that the chi-squared function is dependent on the relaxation equations which are dependent on the transformed relaxation equations which themselves are dependent on the spectral density functions, combine with the values, gradients, and Hessians to create a complex web of dependencies. The relationship between all the values, gradients, and Hessians are outlined in Figure 4.1.

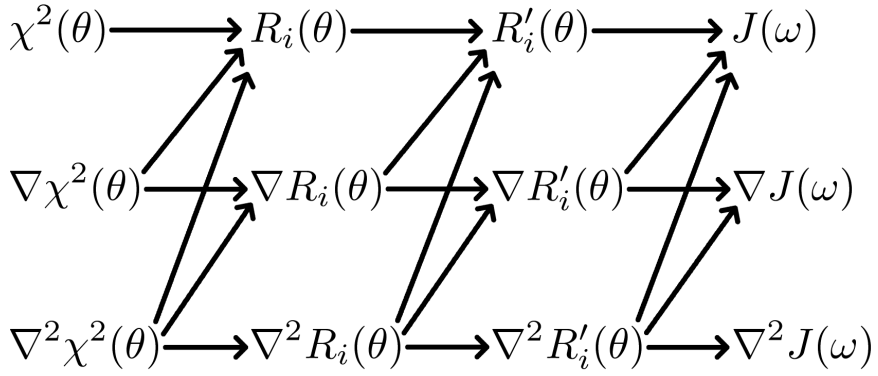

Figure 4.1: Dependencies between the χ^2, transformed relaxation, relaxation, and spectral density equations, gradients, and Hessians.

4.4 χ^2 values, gradients, and Hessians

4.4.1 χ^2 values

The χ^2 value is

$$\chi^2(\theta) = \sum_{i=1}^{n} \frac{(\mathrm{R}_i - \mathrm{R}_i(\theta))^2}{\sigma_i^2}, \tag{4.28}$$

which is the same as equation (4.1) on page 19.

4.4.2 χ^2 gradients

The χ^2 gradient is

$$\nabla\chi^2(\theta) = 2\sum_{i=1}^{n} \frac{(\mathrm{R}_i - \mathrm{R}_i(\theta))^2}{\sigma_i^2} \nabla\mathrm{R}_i(\theta). \tag{4.29}$$

4.4.3 χ^2 Hessians

The χ^2 Hessian is

$$\nabla^2\chi^2(\theta) = 2\sum_{i=1}^{n} \frac{1}{\sigma_i^2} \left(\nabla\mathrm{R}_i(\theta) \cdot \nabla\mathrm{R}_i(\theta)^T - (\mathrm{R}_i - \mathrm{R}_i(\theta))\nabla^2\mathrm{R}_i(\theta)\right). \tag{4.30}$$

4.5 $\mathrm{R}_i(\theta)$ values, gradients, and Hessians

4.5.1 $\mathrm{R}_i(\theta)$ values

The $\mathrm{R}_i(\theta)$ values are given by

$$\mathrm{R}_1(\theta) = \mathrm{R}'_1(\theta), \tag{4.31}$$
$$\mathrm{R}_2(\theta) = \mathrm{R}'_2(\theta), \tag{4.32}$$
$$\mathrm{NOE}(\theta) = 1 + \frac{\gamma_H}{\gamma_X}\frac{\sigma_{\mathrm{NOE}}(\theta)}{\mathrm{R}_1(\theta)}. \tag{4.33}$$

4.5.2 $\mathrm{R}_i(\theta)$ gradients

The $\mathrm{R}_i(\theta)$ gradients are

$$\nabla\mathrm{R}_1(\theta) = \nabla\mathrm{R}'_1(\theta), \tag{4.34}$$
$$\nabla\mathrm{R}_2(\theta) = \nabla\mathrm{R}'_2(\theta), \tag{4.35}$$
$$\nabla\mathrm{NOE}(\theta) = \frac{\gamma_H}{\gamma_X}\frac{1}{\mathrm{R}_1(\theta)^2}\Big(\mathrm{R}_1(\theta)\nabla\sigma_{\mathrm{NOE}}(\theta) - \sigma_{\mathrm{NOE}}(\theta)\nabla\mathrm{R}_1(\theta)\Big). \tag{4.36}$$

4.5.3 $\mathrm{R}_i(\theta)$ Hessians

The $\mathrm{R}_i(\theta)$ Hessians are

$$\nabla^2\mathrm{R}_1(\theta) = \nabla^2\mathrm{R}'_1(\theta), \tag{4.37}$$
$$\nabla^2\mathrm{R}_2(\theta) = \nabla^2\mathrm{R}'_2(\theta), \tag{4.38}$$
$$\begin{aligned}\nabla^2\mathrm{NOE}(\theta) = \frac{\gamma_H}{\gamma_X}\frac{1}{\mathrm{R}_1(\theta)^3}\Big[&\sigma_{\mathrm{NOE}}(\theta)\Big(2\nabla\mathrm{R}_1(\theta)\cdot\nabla\mathrm{R}_1(\theta)^T - \mathrm{R}_1(\theta)\nabla^2\mathrm{R}_1(\theta)\Big)\\ &- \mathrm{R}_1(\theta)\Big(\nabla\sigma_{\mathrm{NOE}}(\theta)\cdot\nabla\mathrm{R}_1(\theta)^T - \mathrm{R}_1(\theta)\nabla^2\sigma_{\mathrm{NOE}}(\theta)\Big)\Big].\end{aligned} \tag{4.39}$$

4.6 $\mathrm{R}_i'(\theta)$ values, gradients, and Hessians

The partial and second partial derivatives of the transformed relaxation equations, $\mathrm{R}_i'(\theta)$, are different for each parameter of the vector θ. The vector representation of the gradient, $\nabla \mathrm{R}_i'(\theta)$, and the matrix representation of the Hessian, $\nabla^2 \mathrm{R}_i'(\theta)$, can be reconstructed from the individual elements presented below.

4.6.1 Components of the $\mathrm{R}_i'(\theta)$ equations

To simplify the calculations of the gradients and Hessians, the $\mathrm{R}_i'(\theta)$ equations have been broken down into their various components. These include the dipolar and CSA constants as well as the dipolar and CSA spectral density terms for each of the three transformed relaxation data types, R_1, R_2, and σ_{NOE}. The segregation of these components simplifies the maths as many partial derivatives of the components are zero.

Dipolar constant

The dipolar constant is defined as

$$d = \frac{1}{4}\left(\frac{\mu_0}{4\pi}\right)^2 \frac{\left(\gamma_H \gamma_X \hbar\right)^2}{< r^6 >}. \tag{4.40}$$

This component of the relaxation equations is independent of the parameter of the spectral density function, θ_j, the chemical exchange parameter, σ_{ex}, and the CSA parameter, $\Delta\sigma$. Therefore, the partial and second partial derivatives with respect to these parameters is zero. Only the partial derivative with respect to the bond length, r, is non-zero, being

$$d' \equiv \frac{\mathrm{d}d}{\mathrm{d}r} = -\frac{3}{2}\left(\frac{\mu_0}{4\pi}\right)^2 \frac{\left(\gamma_H \gamma_X \hbar\right)^2}{< r^7 >}. \tag{4.41}$$

The double partial derivative with respect to the bond length twice is

$$d'' \equiv \frac{\mathrm{d}^2 d}{\mathrm{d}r^2} = \frac{21}{2}\left(\frac{\mu_0}{4\pi}\right)^2 \frac{\left(\gamma_H \gamma_X \hbar\right)^2}{< r^8 >}. \tag{4.42}$$

CSA constant

The CSA constant is defined as

$$c = \frac{\left(\omega_X \cdot \Delta\sigma\right)^2}{3}. \tag{4.43}$$

The partial derivative of this component with respect to all parameters but the CSA parameter $\Delta\sigma$ is zero. This partial derivative is

$$c' \equiv \frac{\mathrm{d}c}{\mathrm{d}\Delta\sigma} = \frac{2\omega_X^2 \cdot \Delta\sigma}{3}. \tag{4.44}$$

The CSA constant double partial derivative with respect to $\Delta\sigma$ is

$$c'' \equiv \frac{\mathrm{d}^2 c}{\mathrm{d}\Delta\sigma^2} = \frac{2\omega_X^2}{3}. \tag{4.45}$$

Spectral density terms of the R_1 dipolar component

For the dipolar component of the R_1 equation, the spectral density terms are

$$J_d^{R_1} = J(\omega_H - \omega_X) + 3J(\omega_X) + 6J(\omega_H + \omega_X). \tag{4.46}$$

The partial derivative of these terms with respect to the parameter of the spectral density function θ_j is

$$J_d^{R_1\prime} \equiv \frac{\partial J_d^{R_1}}{\partial \theta_j} = \frac{\partial J(\omega_H - \omega_X)}{\partial \theta_j} + 3\frac{\partial J(\omega_X)}{\partial \theta_j} + 6\frac{\partial J(\omega_H + \omega_X)}{\partial \theta_j}. \tag{4.47}$$

The second partial derivative with respect to the parameter of the spectral density function θ_j and θ_k is

$$J_d^{R_1\prime\prime} \equiv \frac{\partial^2 J_d^{R_1}}{\partial \theta_j \cdot \partial \theta_k} = \frac{\partial^2 J(\omega_H - \omega_X)}{\partial \theta_j \cdot \partial \theta_k} + 3\frac{\partial^2 J(\omega_X)}{\partial \theta_j \cdot \partial \theta_k} + 6\frac{\partial^2 J(\omega_H + \omega_X)}{\partial \theta_j \cdot \partial \theta_k}. \tag{4.48}$$

Spectral density terms of the R_1 CSA component

For the CSA component of the R_1 equation, the spectral density terms are

$$J_c^{R_1} = J(\omega_X). \tag{4.49}$$

The partial derivative of these terms with respect to the parameter of the spectral density function θ_j is

$$J_c^{R_1\prime} \equiv \frac{\partial J_c^{R_1}}{\partial \theta_j} = \frac{\partial J(\omega_X)}{\partial \theta_j}. \tag{4.50}$$

The second partial derivative with respect to the parameter of the spectral density function θ_j and θ_k is

$$J_c^{R_1\prime\prime} \equiv \frac{\partial^2 J_c^{R_1}}{\partial \theta_j . \partial \theta_k} = \frac{\partial^2 J(\omega_X)}{\partial \theta_j \cdot \partial \theta_k}. \tag{4.51}$$

Spectral density terms of the R_2 dipolar component

For the dipolar component of the R_2 equation, the spectral density terms are

$$J_d^{R_2} = 4J(0) + J(\omega_H - \omega_X) + 3J(\omega_X) + 6J(\omega_H) + 6J(\omega_H + \omega_X). \tag{4.52}$$

The partial derivative of these terms with respect to the parameter of the spectral density function θ_j is

$$J_d^{R_2\prime} \equiv \frac{\partial J_d^{R_2}}{\partial \theta_j} = 4\frac{\partial J(0)}{\partial \theta_j} + \frac{\partial J(\omega_H - \omega_X)}{\partial \theta_j} + 3\frac{\partial J(\omega_X)}{\partial \theta_j} + 6\frac{\partial J(\omega_H)}{\partial \theta_j} + 6\frac{\partial J(\omega_H + \omega_X)}{\partial \theta_j}. \tag{4.53}$$

The second partial derivative with respect to the parameter of the spectral density function θ_j and θ_k is

$$J_d^{\mathrm{R_2}\prime\prime} \equiv \frac{\partial^2 J_d^{\mathrm{R_2}}}{\partial\theta_j \cdot \partial\theta_k} = 4\frac{\partial^2 J(0)}{\partial\theta_j \cdot \partial\theta_k} + \frac{\partial^2 J(\omega_H - \omega_X)}{\partial\theta_j \cdot \partial\theta_k} + 3\frac{\partial^2 J(\omega_X)}{\partial\theta_j \cdot \partial\theta_k} + 6\frac{\partial^2 J(\omega_H)}{\partial\theta_j \cdot \partial\theta_k} + 6\frac{\partial^2 J(\omega_H + \omega_X)}{\partial\theta_j \cdot \partial\theta_k}. \quad (4.54)$$

Spectral density terms of the $\mathrm{R_2}$ CSA component

For the CSA component of the $\mathrm{R_2}$ equation, the spectral density terms are

$$J_c^{\mathrm{R_2}} = 4J(0) + 3J(\omega_X). \quad (4.55)$$

The partial derivative of these terms with respect to the parameter of the spectral density function θ_j is

$$J_c^{\mathrm{R_2}\prime} \equiv \frac{\partial J_c^{\mathrm{R_2}}}{\partial\theta_j} = 4\frac{\partial J(0)}{\partial\theta_j} + 3\frac{\partial J(\omega_X)}{\partial\theta_j}. \quad (4.56)$$

The second partial derivative with respect to the parameter of the spectral density function θ_j and θ_k is

$$J_c^{\mathrm{R_2}\prime\prime} \equiv \frac{\partial^2 J_c^{\mathrm{R_2}}}{\partial\theta_j \cdot \partial\theta_k} = 4\frac{\partial^2 J(0)}{\partial\theta_j \cdot \partial\theta_k} + 3\frac{\partial^2 J(\omega_X)}{\partial\theta_j \cdot \partial\theta_k}. \quad (4.57)$$

Spectral density terms of the σ_{NOE} dipolar component

For the dipolar component of the σ_{NOE} equation, the spectral density terms are

$$J_d^{\sigma_{\mathrm{NOE}}} = 6J(\omega_H + \omega_X) - 6J(\omega_H - \omega_X). \quad (4.58)$$

The partial derivative of these terms with respect to the parameter of the spectral density function θ_j is

$$J_d^{\sigma_{\mathrm{NOE}}\prime} \equiv \frac{\partial J_d^{\sigma_{\mathrm{NOE}}}}{\partial\theta_j} = 6\frac{\partial J(\omega_H + \omega_X)}{\partial\theta_j} - 6\frac{\partial J(\omega_H - \omega_X)}{\partial\theta_j}. \quad (4.59)$$

The second partial derivative with respect to the parameter of the spectral density function θ_j and θ_k is

$$J_d^{\sigma_{\mathrm{NOE}}\prime\prime} \equiv \frac{\partial^2 J_d^{\sigma_{\mathrm{NOE}}}}{\partial\theta_j \cdot \partial\theta_k} = 6\frac{\partial^2 J(\omega_H + \omega_X)}{\partial\theta_j \cdot \partial\theta_k} - 6\frac{\partial^2 J(\omega_H - \omega_X)}{\partial\theta_j \cdot \partial\theta_k}. \quad (4.60)$$

4.6.2 $\mathrm{R}'_i(\theta)$ values

The three relaxation equations, utilising the above components, can be expressed as

$$\mathrm{R}_1(\theta) = dJ_d^{\mathrm{R}_1} + cJ_c^{\mathrm{R}_1}, \tag{4.61}$$

$$\mathrm{R}_2(\theta) = \frac{d}{2}J_d^{\mathrm{R}_2} + \frac{c}{6}J_c^{\mathrm{R}_2}, \tag{4.62}$$

$$\sigma_{\mathrm{NOE}}(\theta) = dJ_d^{\sigma_{\mathrm{NOE}}}. \tag{4.63}$$

4.6.3 $\mathrm{R}'_i(\theta)$ gradients

The partial derivatives with respect to the parameter of the spectral density functions, the chemical exchange parameter, CSA parameter, and bond length parameters are all different and are presented below.

Spectral density function parameter

The partial derivatives of the relaxation equations with respect to the parameter of the spectral density function θ_j are

$$\frac{\partial \mathrm{R}_1(\theta)}{\partial \theta_j} = dJ_d^{\mathrm{R}_1\prime} + cJ_c^{\mathrm{R}_1\prime}, \tag{4.64}$$

$$\frac{\partial \mathrm{R}_2(\theta)}{\partial \theta_j} = \frac{d}{2}J_d^{\mathrm{R}_2\prime} + \frac{c}{6}J_c^{\mathrm{R}_2\prime}, \tag{4.65}$$

$$\frac{\partial \sigma_{\mathrm{NOE}}(\theta)}{\partial \theta_j} = dJ_d^{\sigma_{\mathrm{NOE}}\prime}. \tag{4.66}$$

Chemical exchange parameter

The partial derivatives of the relaxation equations with respect to the chemical exchange parameter R_{ex} are

$$\frac{\partial \mathrm{R}_1(\theta)}{\partial R_{ex}} = 0, \tag{4.67}$$

$$\frac{\partial \mathrm{R}_2(\theta)}{\partial R_{ex}} = 1, \tag{4.68}$$

$$\frac{\partial \sigma_{\mathrm{NOE}}(\theta)}{\partial R_{ex}} = 0. \tag{4.69}$$

CSA parameter

The partial derivatives of the relaxation equations with respect to the CSA parameter $\Delta\sigma$ are

$$\frac{\partial \mathrm{R}_1(\theta)}{\partial \Delta\sigma} = c' J_c^{\mathrm{R}_1}, \tag{4.70}$$

$$\frac{\partial \mathrm{R}_2(\theta)}{\partial \Delta\sigma} = \frac{c'}{6} J_c^{\mathrm{R}_2}, \tag{4.71}$$

$$\frac{\partial \sigma_{\mathrm{NOE}}(\theta)}{\partial \Delta\sigma} = 0. \tag{4.72}$$

Bond length parameter

The partial derivatives of the relaxation equations with respect to the bond length parameter r are

$$\frac{\partial \mathrm{R}_1(\theta)}{\partial r} = d' J_d^{\mathrm{R}_1}, \tag{4.73}$$

$$\frac{\partial \mathrm{R}_2(\theta)}{\partial r} = \frac{d'}{2} J_d^{\mathrm{R}_2}, \tag{4.74}$$

$$\frac{\partial \sigma_{\mathrm{NOE}}(\theta)}{\partial r} = d' J_d^{\sigma_{\mathrm{NOE}}}. \tag{4.75}$$

4.6.4 $\mathrm{R}_i'(\theta)$ Hessians

The second partial derivatives with respect to the parameter of the spectral density functions, the chemical exchange parameter, CSA parameter, and bond length parameters are presented below.

Spectral density function parameter - Spectral density function parameter

The second partial derivatives of the relaxation equations with respect to the parameter of the spectral density functions θ_j and θ_k are

$$\frac{\partial^2 \mathrm{R}_1(\theta)}{\partial \theta_j \cdot \partial \theta_k} = d J_d^{\mathrm{R}_1\prime\prime} + c J_c^{\mathrm{R}_1\prime\prime}, \tag{4.76}$$

$$\frac{\partial^2 \mathrm{R}_2(\theta)}{\partial \theta_j \cdot \partial \theta_k} = \frac{d}{2} J_d^{\mathrm{R}_2\prime\prime} + \frac{c}{6} J_c^{\mathrm{R}_2\prime\prime}, \tag{4.77}$$

$$\frac{\partial^2 \sigma_{\mathrm{NOE}}(\theta)}{\partial \theta_j \cdot \partial \theta_k} = d J_d^{\sigma_{\mathrm{NOE}}\prime\prime}. \tag{4.78}$$

Spectral density function parameter - Chemical exchange parameter

The second partial derivatives of the relaxation equations with respect to the parameter of the spectral density function θ_j and the chemical exchange parameter R_{ex} are

$$\frac{\partial^2 \mathrm{R}_1(\theta)}{\partial\theta_j \cdot \partial R_{ex}} = 0, \tag{4.79}$$

$$\frac{\partial^2 \mathrm{R}_2(\theta)}{\partial\theta_j \cdot \partial R_{ex}} = 0, \tag{4.80}$$

$$\frac{\partial^2 \sigma_{\mathrm{NOE}}(\theta)}{\partial\theta_j \cdot \partial R_{ex}} = 0. \tag{4.81}$$

Spectral density function parameter - CSA parameter

The second partial derivatives of the relaxation equations with respect to the parameter of the spectral density function θ_j and the CSA parameter $\Delta\sigma$ are

$$\frac{\partial^2 \mathrm{R}_1(\theta)}{\partial\theta_j \cdot \partial \Delta\sigma} = c' J_c^{\mathrm{R}_1\prime}, \tag{4.82}$$

$$\frac{\partial^2 \mathrm{R}_2(\theta)}{\partial\theta_j \cdot \partial \Delta\sigma} = \frac{c'}{6} J_c^{\mathrm{R}_2\prime}, \tag{4.83}$$

$$\frac{\partial^2 \sigma_{\mathrm{NOE}}(\theta)}{\partial\theta_j \cdot \partial \Delta\sigma} = 0. \tag{4.84}$$

Spectral density function parameter - Bond length parameter

The second partial derivatives of the relaxation equations with respect to the parameter of the spectral density function θ_j and the bond length parameter r are

$$\frac{\partial^2 \mathrm{R}_1(\theta)}{\partial\theta_j \cdot \partial r} = d' J_d^{\mathrm{R}_1\prime}, \tag{4.85}$$

$$\frac{\partial^2 \mathrm{R}_2(\theta)}{\partial\theta_j \cdot \partial r} = \frac{d'}{2} J_d^{\mathrm{R}_2\prime}, \tag{4.86}$$

$$\frac{\partial^2 \sigma_{\mathrm{NOE}}(\theta)}{\partial\theta_j \cdot \partial r} = d' J_d^{\sigma_{\mathrm{NOE}}\prime}. \tag{4.87}$$

Chemical exchange parameter - Chemical exchange parameter

The second partial derivatives of the relaxation equations with respect to the chemical exchange parameter R_{ex} twice are

$$\frac{\partial^2 \mathrm{R}_1(\theta)}{\partial {R_{ex}}^2} = 0, \tag{4.88}$$

$$\frac{\partial^2 \mathrm{R}_2(\theta)}{\partial {R_{ex}}^2} = 0, \tag{4.89}$$

$$\frac{\partial^2 \sigma_{\mathrm{NOE}}(\theta)}{\partial {R_{ex}}^2} = 0. \tag{4.90}$$

Chemical exchange parameter - CSA parameter

The second partial derivatives of the relaxation equations with respect to the chemical exchange parameter R_{ex} and the CSA parameter $\Delta\sigma$ are

$$\frac{\partial^2 \mathrm{R}_1(\theta)}{\partial R_{ex} \cdot \partial \Delta\sigma} = 0, \tag{4.91}$$

$$\frac{\partial^2 \mathrm{R}_2(\theta)}{\partial R_{ex} \cdot \partial \Delta\sigma} = 0, \tag{4.92}$$

$$\frac{\partial^2 \sigma_{\mathrm{NOE}}(\theta)}{\partial R_{ex} \cdot \partial \Delta\sigma} = 0. \tag{4.93}$$

Chemical exchange parameter - Bond length parameter

The second partial derivatives of the relaxation equations with respect to the chemical exchange parameter R_{ex} and the bond length parameter r are

$$\frac{\partial^2 \mathrm{R}_1(\theta)}{\partial R_{ex} \cdot \partial r} = 0, \tag{4.94}$$

$$\frac{\partial^2 \mathrm{R}_2(\theta)}{\partial R_{ex} \cdot \partial r} = 0, \tag{4.95}$$

$$\frac{\partial^2 \sigma_{\mathrm{NOE}}(\theta)}{\partial R_{ex} \cdot \partial r} = 0. \tag{4.96}$$

CSA parameter - CSA parameter

The second partial derivatives of the relaxation equations with respect to the CSA parameter $\Delta\sigma$ twice are

$$\frac{\partial^2 \mathrm{R}_1(\theta)}{\partial \Delta\sigma^2} = c'' J_c^{\mathrm{R}_1}, \tag{4.97}$$

$$\frac{\partial^2 \mathrm{R}_2(\theta)}{\partial \Delta\sigma^2} = \frac{c''}{6} J_c^{\mathrm{R}_2}, \tag{4.98}$$

$$\frac{\partial^2 \sigma_{\mathrm{NOE}}(\theta)}{\partial \Delta\sigma^2} = 0. \tag{4.99}$$

CSA parameter - Bond length parameter

The second partial derivatives of the relaxation equations with respect to the CSA parameter $\Delta\sigma$ and the bond length parameter r are

$$\frac{\partial^2 \mathrm{R}_1(\theta)}{\partial \Delta\sigma \cdot \partial r} = 0, \tag{4.100}$$

$$\frac{\partial^2 \mathrm{R}_2(\theta)}{\partial \Delta\sigma \cdot \partial r} = 0, \tag{4.101}$$

$$\frac{\partial^2 \sigma_{\mathrm{NOE}}(\theta)}{\partial \Delta\sigma \cdot \partial r} = 0. \tag{4.102}$$

Bond length parameter - Bond length parameter

The second partial derivatives of the relaxation equations with respect to the bond length parameter r twice are

$$\frac{\partial^2 \mathrm{R}_1(\theta)}{\partial r^2} = d'' J_d^{\mathrm{R}_1}, \tag{4.103}$$

$$\frac{\partial^2 \mathrm{R}_2(\theta)}{\partial r^2} = \frac{d''}{2} J_d^{\mathrm{R}_2}, \tag{4.104}$$

$$\frac{\partial^2 \sigma_{\mathrm{NOE}}(\theta)}{\partial r^2} = d'' J_d^{\sigma_{\mathrm{NOE}}}. \tag{4.105}$$

4.7 Ellipsoidal diffusion tensor

4.7.1 Ellipsoid weight derivatives

Chapter 5

Development of relax

This chapter is written for developers or those who would like to extend the functionality of relax. It is not required for using relax.

5.1 Coding conventions

5.1.1 Indentation

Indentation should be set to four spaces rather than a tab character. This is the recommendation given in the python style guide found at `'http://www.python.org/doc/essays/styleguide.html'`. For vi, add the following lines to `'~/.vimrc'`:

```
set tabstop=4
set shiftwidth=4
set expandtab
```

Certain versions of vim, those within the 6.2 series, contain a bug where the tabstop value cannot be changed using the `'~/.vimrc'` file (although typing `':set tabstop=4'` in vim will fix it). One solution is to edit the file `'python.vim'` which is located in the path `'/usr/share/vim/ftplugin/'` or equivalent. It contains the two lines:

```
" Python always uses a 'tabstop' of 8.
setlocal ts=8
```

If these lines are deleted, the bug will be removed. Another way to fix the problem is to install newer versions of the run-time files (which will pretty much do the same thing).

5.1.2 Doc strings

These should be set to no more than 100 characters long including all leading white space. The standard Python convention of a one line description separated from a detailed description by an empty line should be adhered to.

5.2 The make system

For UNIX like systems, including GNU/Linux, various components of the program relax can be created using the Unix make utility. This includes C module compilation, manual creation, distribution creation, and cleaning up and removing certain files. Makefiles exist in the various directories to facilitate this process.

5.2.1 C module compilation

As described in the installation chapter, typing `'make'` in the base directory will create the shared objects which are imported into Python as modules.

5.2.2 Creation of the PDF manual

To create the PDF version of the relax manual, type

```
make manual
```

in the base directory. Make will then shift to the `'docs/latex'` directory and run a series of shell commands to create the manual from the LaTeX sources. This is dependent on the programs `'rm'`, `'latex'`, `'makeindex'`, `'dvips'`, and `'ps2pdf'` being located in the environment's path.

5.2.3 Creation of the HTML manual

The HTML version of the relax manual is made by typing

```
make manual_html
```

in the base directory. Again make will then shift to the `'docs/latex'` directory. One command calling the program `'latex2html'` will be executed which will create HTML pages from the LaTeX sources.

5.2.4 Making distribution archives

This section is incomplete. Creation of the HTML manual

5.2.5 Cleaning up

If the command

```
make clean
```

is run in the base directory, all Python byte compiled files '*.pyc', all C object files '*.o', all C shared object files '*.so', and any backup files with the extension '*.bak' are removed from all subdirectoies. In addition, any temporary LaTeX compilation files are removed from the `'docs/latex'` directory.

Chapter 6

Alphabetical listing of user functions

The following is a listing with descriptions of all the user functions available within the relax prompt and scripting environments. These are simply an alphabetical list of the docstrings which can normally be viewed in prompt mode by typing '`help(function)`' .

6.1 A warning about the formatting

The following documentation of the user functions has been automatically generated by a script which extracts and formats the docstring associated with each function. There may therefore be instances where the formatting has failed or where there are inconsistencies.

6.2 The list of functions

Each user function is presented within it's own subsection with the documentation broken into multiple parts, the synopsis, the default arguments, and the sections from the function's docstring.

6.2.1 The synopsis

The synopsis presents a brief description of the function. It is taken as the first line of the docstring when browsing the help system .

6.2.2 Defaults

This section lists all the arguments taken by the function and their default values. To invoke the function, type the function name then in brackets type a comma separated list of arguments.

The first argument printed is always 'self' but you can safely ignore it. 'self' is part of the object oriented programming within Python and is automatically prefixed to the list of arguments you supply. Therefore you can't provide 'self' as the first argument, even if you do try.

6.2.3 Docstring sectioning

All other sections are created from the sectioning within the actual docstring.

6.2.4 minimise

Synopsis

Minimisation function.

Defaults

minimise(self, *args, **keywords)

Arguments

The arguments, which should all be strings, specify the minimiser as well as its options. A minimum of one argument is required. As this calls the function '`generic_minimise`' the full list of allowed arguments is shown below in the reproduced '`generic_minimise`' docstring. Ignore all sections except those labelled as minimisation algorithms and minimisation options. Also do not select the Method of Multipliers constraint algorithm as this is used in combination with the given minimisation algorithm if the keyword argument '`constraints`' is set to 1. The grid search algorithm should also not be selected as this is accessed using the '`grid`' function instead. The first argument passed will be set to the minimisation algorithm while all other arguments will be set to the minimisation options.

Keyword arguments differ from normal arguments having the form '`keyword = value`'. All arguments must precede keyword arguments in python. For more information see the examples section below or the python tutorial.

Keyword Arguments

run: The name of the run.

func_tol: The function tolerance. This is used to terminate minimisation once the function value between iterations is less than the tolerance. The default value is 1e-25.

grad_tol: The gradient tolerance. Minimisation is terminated if the current gradient value is less than the tolerance. The default value is None.

max_iterations: The maximum number of iterations. The default value is 1e7.

constraints: A flag specifying whether the parameters should be constrained. The default is to turn constraints on (constraints=1).

scaling: The diagonal scaling flag. The default that scaling is on (scaling=1).

print_flag: The amount of information to print to screen. Zero corresponds to minimal output while higher values increase the amount of output. The default value is 1.

Diagonal scaling

Diagonal scaling is the transformation of parameter values such that each value has a similar order of magnitude. Certain minimisation techniques, for example the trust region methods, perform extremely poorly with badly scaled problems. In addition, methods which are insensitive to scaling such as Newton minimisation may still benefit due to the minimisation of round off errors.

In Model-free analysis for example, if $S^2 = 0.5$, $\tau_e = 200$ ps, and $R_{ex} = 15$ 1/s at 600 MHz, the unscaled parameter vector would be [0.5, 2.0e-10, 1.055e-18]. R_{ex} is divided by (2 * π * 600,000,000)**2 to make it field strength independent. The scaling vector for this model may be something like [1.0, 1e-9, 1/(2 * π * 6e8)**2]. By dividing the unscaled parameter vector by the scaling vector the scaled parameter vector is [0.5, 0.2, 15.0]. To revert to the original unscaled parameter vector, the scaled parameter vector and scaling vector are multiplied.

Examples

To minimise the model-free run '`m4`' using Newton minimisation together with the GMW81 Hessian modification algorithm, the More and Thuente line search algorithm, a function tolerance of 1e-25, no gradient tolerance, a maximum of 10,000,000 iterations, constraints turned on to limit parameter values, and have normal printout, type any combination of:

```
relax> minimise('newton', run='m4')

relax> minimise('Newton', run='m4')

relax> minimise('newton', 'gmw', run='m4')

relax> minimise('newton', 'mt', run='m4')

relax> minimise('newton', 'gmw', 'mt', run='m4')

relax> minimise('newton', 'mt', 'gmw', run='m4')

relax> minimise('newton', run='m4', func_tol=1e-25)

relax> minimise('newton', run='m4', func_tol=1e-25, grad_tol=None)

relax> minimise('newton', run='m4', max_iter=1e7)

relax> minimise('newton', run=name, constraints=1, max_iter=1e7)

relax> minimise('newton', run='m4', print_flag=1)
```

To minimise the model-free run '`m5`' using constrained Simplex minimisation with a maximum of 5000 iterations, type:

```
relax> minimise('simplex', run='m5', constraints=1, max_iter=5000)
```

Note

```
All the text which follows is a reproduction of the docstring of the generic_minimise
function.  Only take note of the minimisation algorithms and minimisation options sections,
the other sections are not relevant for this function.  The Grid search and Method of
```

```
Multipliers algorithms CANNOT be selected as minimisation algorithms for this function.

The section entitled Keyword Arguments is also completely inaccessible therefore please
ignore that text.
```

Generic minimisation function.

This is a generic function which can be used to access all minimisers using the same set of function arguments. These are the function tolerance value for convergence tests, the maximum number of iterations, a flag specifying which data structures should be returned, and a flag specifying the amount of detail to print to screen.

Keyword Arguments

func: The function which returns the value.

dfunc: The function which returns the gradient.

d2func: The function which returns the Hessian.

args: The tuple of arguments to supply to the functions func, dfunc, and d2func.

x0: The vector of initial parameter value estimates (as an array).

min_algor: A string specifying which minimisation technique to use.

min_options: A tuple to pass to the minimisation function as the min_options keyword.

func_tol: The function tolerance value. Once the function value between iterations decreases below this value, minimisation is terminated.

grad_tol: The gradient tolerance value.

maxiter: The maximum number of iterations.

A: Linear constraint matrix m*n (A.x $\geq b$).

b: Linear constraint scalar vector (A.x $\geq b$).

l: Lower bound constraint vector ($l \leq x \leq u$).

u: Upper bound constraint vector ($l \leq x \leq u$).

c: User supplied constraint function.

dc: User supplied constraint gradient function.

d2c: User supplied constraint Hessian function.

full_output: A flag specifying which data structures should be returned.

print_flag: A flag specifying how much information should be printed to standard output during minimisation. 0 means no output, 1 means minimal output, and values above 1 increase the amount of output printed.

Minimisation output

The following values of the '`full_output`' flag will return, in tuple form, the following data

0 – '`xk`',

1 – '`(xk, fk,` k`, f_count, g_count, h_count, warning)`',

where the data names correspond to

'`xk`' – The array of minimised parameter values,

'`fk`' – The minimised function value,

'`k`' – The number of iterations,

'`f_count`' – The number of function calls,

'`g_count`' – The number of gradient calls,

'`h_count`' – The number of Hessian calls,

'`warning`' – The warning string.

Minimisation algorithms

A minimisation function is selected if the minimisation algorithm argument, which should be a string, matches a certain pattern. Because the python regular expression '`match`' statement is used, various strings can be supplied to select the same minimisation algorithm. Below is a list of the minimisation algorithms available together with the corresponding patterns.

This is a short description of python regular expression, for more information, see the regular expression syntax section of the Python Library Reference. Some of the regular expression syntax used in this function is:

'`[]`' – A sequence or set of characters to match to a single character. For example, '`[Nn]ewton`' will match both '`Newton`' and '`newton`'.

'`^`' – Match the start of the string.

'`$`' – Match the end of the string. For example, '`^[Ll][Mm]$`' will match '`lm`' and '`LM`' but will not match if characters are placed either before or after these strings.

To select a minimisation algorithm, set the argument to a string which matches the given pattern.

Parameter initialisation methods:

Minimisation algorithm	Patterns
Grid search	'^[Gg]rid'

Unconstrained line search methods:

Minimisation algorithm	Patterns
Back-and-forth coordinate descent	'^[Cc][Dd]$' or '^[Cc]oordinate[_-][Dd]escent$'
Steepest descent	'^[Ss][Dd]$' or '^[Ss]teepest[_-][Dd]escent$'
Quasi-Newton BFGS	'^[Bb][Ff][Gg][Ss]$'
Newton	'^[Nn]ewton$'
Newton-CG	'^[Nn]ewton[_-][Cc][Gg]$' or '^[Nn][Cc][Gg]$'

Unconstrained trust-region methods:

Minimisation algorithm	Patterns
Cauchy point	'^[Cc]auchy'
Dogleg	'^[Dd]ogleg'
CG-Steihaug	'^[Cc][Gg][-_][Ss]teihaug' or '^[Ss]teihaug'
Exact trust region	'^[Ee]xact'

Unconstrained conjugate gradient methods:

Minimisation algorithm	Patterns
Fletcher-Reeves	'^[Ff][Rr]$' or '^[Ff]letcher[-_][Rr]eeves$'
Polak-Ribière	'^[Pp][Rr]$' or '^[Pp]olak[-_][Rr]ibiere$'
Polak-Ribière +	'^[Pp][Rr]\+$' or '^[Pp]olak[-_][Rr]ibiere\+$'
Hestenes-Stiefel	'^[Hh][Ss]$' or '^[Hh]estenes[-_][Ss]tiefel$'

Miscellaneous unconstrained methods:

Minimisation algorithm	Patterns
Simplex	'^[Ss]implex$'
Levenberg-Marquardt	'^[Ll][Mm]$' or '^[Ll]evenburg-[Mm]arquardt$'

Constrained methods:

Minimisation algorithm	Patterns
Method of Multipliers	'`^[Mm][Oo][Mm]$`' or '`[Mm]ethod of [Mm]ultipliers$`'

Minimisation options

The minimisation options can be given in any order.

Line search algorithms. These are used in the line search methods and the conjugate gradient methods. The default is the Backtracking line search.

Line search algorithm	Patterns
Backtracking line search	'`^[Bb]ack`'
Nocedal and Wright interpolation based line search	'`^[Nn][Ww][Ii]`' or '`^[Nn]ocedal[ _][Ww]right[ _][Ii]nt`'
Nocedal and Wright line search for the Wolfe conditions	'`^[Nn][Ww][Ww]`' or '`^[Nn]ocedal[ _][Ww]right[ _][Ww]olfe`'
More and Thuente line search	'`^[Mm][Tt]`' or '`^[Mm]ore[ _][Tt]huente$`'
No line search	'`^[Nn]one$`'

Hessian modifications. These are used in the Newton, Dogleg, and Exact trust region algorithms.

Hessian modification	Patterns
Unmodified Hessian	'`[Nn]one`'
Eigenvalue modification	'`^[Ee]igen`'
Cholesky with added multiple of the identity	'`^[Cc]hol`'
The Gill, Murray, and Wright modified Cholesky algorithm	'`^[Gg][Mm][Ww]$`'
The Schnabel and Eskow 1999 algorithm	'`^[Ss][Ee]99`'

Hessian type, these are used in a few of the trust region methods including the Dogleg and Exact trust region algorithms. In these cases, when the Hessian type is set to Newton, a Hessian modification can also be supplied as above. The default Hessian type is Newton, and the default Hessian modification when Newton is selected is the GMW algorithm.

Hessian type	Patterns
Quasi-Newton BFGS	'`^[Bb][Ff][Gg][Ss]$`'
Newton	'`^[Nn]ewton$`'

For Newton minimisation, the default line search algorithm is the More and Thuente line search, while the default Hessian modification is the GMW algorithm.

Chapter 7

Licence

7.1 Copying, modification, sublicencing, and distribution of relax

To ensure that the program relax, including all future versions, will remain legally available for perpetuity to anyone who wishes to use the program, it has been decided to release the code under the GNU General Public Licence. The freedom of relax is guaranteed by the GPL. This is a licence which has been very carefully crafted to protect both the author of the program as well as the public by means of copyright law. If the licence is violated by improper copying, modification, sublicencing, or distribution then the licence terminates – hence the violator is copying, modifying, sublicencing, or distributing the program illegally in full violation of copyright law. For a better understanding of the protections afforded by the GPL, the licence is reprinted in whole within the next section.

7.2 The GPL

The following is a verbatim copy of the GNU General Public Licence. A text version is available in the relax 'docs' directory within the file 'COPYING'.

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those

sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published

by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
```

```
Copyright (C) <year>  <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year  name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands ‘`show w`’ and ‘`show c`’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘`show w`’ and ‘`show c`’; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Bibliography

Anatole Abragam. *The Principles of Nuclear Magnetism*. Clarendon Press, Oxford, 1961.

G.M. Clore, A. Szabo, A. Bax, L.E. Kay, P.C. Driscoll, and A.M. Gronenborn. Deviations from the simple 2-parameter model-free approach to the interpretation of N-15 nuclear magnetic-relaxation of proteins. *Journal of the American Chemical Society*, 112(12): 4989 – 4991, JUN 6 1990. ISSN 0002-7863.

G. Lipari and A. Szabo. Model-free approach to the interpretation of nuclear magnetic-resonance relaxation in macromolecules I. Theory and range of validity. *Journal of the American Chemical Society*, 104(17):4546 – 4559, 1982a. ISSN 0002-7863.

G. Lipari and A. Szabo. Model-free approach to the interpretation of nuclear magnetic-resonance relaxation in macromolecules II. Analysis of experimental results. *Journal of the American Chemical Society*, 104(17):4559 – 4570, 1982b. ISSN 0002-7863.

Index