Computational Spin Dynamics and Visualisation of Large Spin Systems

Gareth Trevor Patrick Charnock
Exeter College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Michaelmas 2012

This thesis is dedicated to
my parents and grandparents.

Acknowledgements

Although the text and the majority of the diagrams (a list of exceptions will follow) are entirely the author's own work, two of the chapters are based on published papers with multiple authors. Consequently, formal acknowledgement must be given to Alexander Karabanov, Ilya Kuprov, Anniek van der Drift, Luke J. Edwards, and Walter Köckenberger for aid in the publishing of "On the accuracy of the state space restriction approximation for spin dynamics simulations" [1], which was the basis for chapter 2, and to Matthew Krzystyniak and Ilya Kuprov for aid in the publishing of "Molecular structure refinement by direct fitting of atomic coordinates to experimental ESR spectra" [2], which was the basis for chapter 4.

Diagrams 2.4, 2.5 and 2.6 from chapter 2 and diagram 4.7 from chapter 5 were taken from the aforementioned papers and have been used with permission.

Acknowledgements relating to specific chapters appear at the end of those chapters.

On a more personal note, I would like to thank Dr Ilya Kuprov for help and guidance, and also for being a generally very responsive supervisor. I would also like to thank the Exeter College MCR, in particular anyone who held the position of social secretary, for being generally friendly and welcoming. Special thanks also go to Dr Helen Orchard for one well timed chat over a cup of tea, without which this thesis might not exist.

Computational Spin Dynamics and Visualisation of Large Spin Systems

Gareth Trevor Patrick Charnock

Oxford e-Research Centre and Exeter College

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2012

Abstract

The thesis commences with a detailed review of the background theory of spin dynamics simulations. State space restriction is introduced via a "top-down" approach. Common terms that make up the spin Hamiltonian are reviewed, and it is noted that the mathematical forms of these terms can be categorised in one of three ways. The review of the background theory complete, accounts are given of the following four areas of research:

1. Formal conditions are established for the validity of state space restriction via spin order pruning, based on tracking the density matrix norm through spin order subspaces. The primary predictor for success is seen to be the ratio of the largest eigenvalue to the relaxation rate. The lower this ratio, the fewer spin orders are required.

2. Software based around the Spin XML format, suitable for constructing and visualising large spin systems, is presented. Both a functional specification and a discussion of the internals are given.

3. A potential application of state space restriction, called "direct structure fitting", (DSF) is explored. In DSF, a candidate chemical structure is optimised directly by minimising the difference between its predicted spectrum and an experimental spectrum. The following examples of successful fits are provided: cyanomethyl, propargyl, and tyrosyl radicals, in the liquid state, and, tyrosyl embedded in two ribonuclease reductase proteins in the powder state.

4. A new model of the pseudocontact shift, which assumes a delocalised electron, is presented. Mathematical subtleties are resolved that would otherwise lead to the failure of numerical evaluation if left untreated. Techniques to improve efficiency are discussed, and the resulting program runs comfortably on workstation-grade hardware on protein sized datasets. E. Coli DNA Polymerase III is investigated as an example, and evidence is presented that suggests that the new model would predict significant differences in structures if used in conjunction with molecular dynamics based structural refinement.

Contents

1 Introduction **1**

1.1 Objectives, Motivations and Chapter Outlines 1

1.2 Background Theory 4

1.2.1 The Rotation Group 4

1.2.2 Spinors 5

1.2.3 The Dynamics of a System of Spins 7

1.2.4 The Density Matrix Formalism 8

1.2.4.1 The Density Matrix of a Single System 9

1.2.4.2 The Density Matrix Formalism for an Ensemble 10

1.2.5 The Initial State 11

1.2.6 Observables in Standard Apparatus 12

1.2.7 Liouville Space 12

1.2.7.1 Time Domain Calculation by Propagators 13

1.2.8 The Secular Approximation 14

1.3 Relaxation Theory 15

1.3.1 Bloch-Redfield-Wangsness Relaxation 15

1.4 State Space Restriction 20

1.4.1 The Linear Algebra Picture 20

1.4.2 Spin Order Pruning 25

1.4.3 Basis Set 26

1.5 Algebras and Spin Dynamics 27

1.5.1 The Tensor Algebra 27

1.5.2 The Product Algebra 28

1.5.3 The Lie Algebra 28

1.5.4 The Exponential Map 30

1.5.5 Lie Theory and Spin Dynamics 31

1.5.6 General n Level Quantum System 31

1.5.6.1 Real and Complex Lie Algebras 32

1.5.7 State Space Restriction in terms of Lie Algebras 34
1.5.8 Construction of Reduced Basis Operators via Left and Right Multiplication Superoperators 35
1.6 The Spin Hamiltonian Terms 37
1.6.1 Zeeman Terms 37
1.6.2 Chemical Shift 37
1.6.3 Dipole-Dipole Interactions 38
1.6.3.1 Nuclear Dipole-Dipole Couplings 39
1.6.3.2 Anisotropic Hyperfine Couplings 39
1.6.3.3 Electron-Electron Couplings 40
1.6.4 J-Couplings 40
1.6.5 Electric Couplings 41
1.6.6 Spin-Orbit Coupling and the g-Tensor 42
1.6.7 The Isotropic Hyperfine 45
1.6.8 The Pseudocontact Shift 46

2 Accuracy of the State Space Restriction Approximation 49
2.1 Statement of the Problem 49
2.2 The Flow of the Density Matrix Norm Under Unitary Transforms 50
2.3 Analysis of the Hamiltonian 51
2.3.1 Term by Term Analysis 52
2.4 Relaxation Superoperator 54
2.5 Flow Between Subspaces 55
2.6 Approximation via Partial Differential Equations 59
2.6.1 General Solutions 59
2.6.2 Interpretation of the Time-Dependent Solution 60
2.6.3 Interpretation of the Steady State 60
2.7 The Single Bucket Assumption 61
2.8 Numerical Evidence 63
2.8.1 Strychnine 64
2.8.2 Pyrene-Dicyanobenzene 65
2.8.3 Random DNP Experiment 65
2.9 Conclusions 66

3 Spin XML Graphical User Interface **67**
3.1 Introduction . . . 67
3.2 Rank 2 Tensor Representations . . . 68
3.2.1 Scalar and Matrix Representations . . . 68
3.2.2 Eigenvalue Representations . . . 68
3.2.3 Axiality and Rhombicity . . . 69
3.2.4 Span and Skew . . . 69
3.3 Rotations and their Representations . . . 71
3.3.1 Euler Angles . . . 73
3.3.2 Direction Cosine Matrices/Rotation Matrices . . . 74
3.3.3 The Angle-Axis and Quaternion Parametrisation . . . 74
3.3.4 The Conversion To Euler Angles . . . 76
3.4 The Spin XML File Format . . . 78
3.4.1 XML Schema Definition . . . 79
3.4.2 XML Schema Diagrams . . . 81
3.4.3 The Spin XML XSD . . . 83
3.4.3.1 Reference Frames . . . 83
3.4.3.2 Spin Elements . . . 85
3.4.3.3 Interactions . . . 85
3.4.3.4 Orientations . . . 86
3.4.3.5 Vectors and Matrices . . . 87
3.4.4 Relational Level Verification . . . 87
3.5 Functional Specification . . . 88
3.5.1 Application Startup . . . 88
3.5.1.1 The Current Selection and the Active Frame . . . 89
3.5.2 Main Window . . . 89
3.5.3 Tensor Visualisation Panel . . . 92
3.5.4 3D Display . . . 93
3.5.5 Grid View . . . 94
3.5.6 The Frame Tree . . . 94
3.5.7 Interaction Editor . . . 94
3.5.8 Orientation Dialogue . . . 95
3.5.9 EasySpin Export Dialogue . . . 96
3.6 Third Party Libraries . . . 99
3.7 Input and Output . . . 100
3.8 Core Data Structures . . . 102

3.8.1 The Orientation Class . . . 102
3.8.1.1 Rotation Conversations . . . 104
3.8.2 The `InteractionPayload` Class . . . 105
3.8.3 The Heap-Allocated Classes . . . 106
3.8.4 Signals and Memory Management . . . 107
3.9 The Graphical Layer . . . 109
3.9.1 The SpinachApp Compilation Unit . . . 109
3.9.2 The Spin Grid . . . 110
3.9.3 The 3D Display Classes . . . 111
3.9.3.1 Camera Management . . . 111
3.9.3.2 The `GLMode` classes . . . 112
3.9.3.3 Picking Mode . . . 112
3.9.3.4 Display Settings . . . 113
3.9.3.5 Rendering . . . 114
3.9.3.6 The `SpinsysDisplay3D` class . . . 115
3.9.3.7 `SpinsysDisplay3D` Picking . . . 117
3.9.4 Context Menus . . . 117
3.9.5 The RootFrame Class . . . 118
3.9.5.1 The `FrameTree` Class . . . 118
3.9.6 Fatal Error Handling . . . 118
3.10 Conclusions and Outlook . . . 120
3.11 Spin XML Acknowledgements . . . 120

4 Direct Structure Fitting **121**
4.1 Methodology . . . 122
4.1.1 The Direct Structure Fitting Functional . . . 122
4.1.1.1 Selection of λ . . . 125
4.1.2 The Direct Structure Fitting Functional Gradient and Minimisation 126
4.1.3 The Direct Structure Fitting Convergence Point . . . 127
4.1.4 Internal Coordinates . . . 128
4.1.5 The Effects of Function Curvature and the Initial Hessian on the Performance of BFGS . . . 129
4.1.6 Parallelism . . . 130
4.1.7 MD5 Caching . . . 131
4.1.8 Implementation Details . . . 132
4.2 Key Results . . . 133

4.2.1 Liquid State EPR . . . 133
4.2.2 Powder EPR Spectra of Embedded Tyrosyl in Proteins . . . 135
4.3 Potential Refinements of `DiStruct` . . . 138
4.3.1 Generalised Coordinate Systems . . . 138
4.3.1.1 Example: Removing Rotational and Translational Degrees of Freedom . . . 140
4.3.1.2 Example: Fixing Bond Lengths . . . 140
4.3.1.3 Example: Tyrosyl Coordinates . . . 141
4.3.2 SQL Database Layer . . . 141
4.3.3 The Micro-optimiser . . . 143
4.4 Conclusion . . . 145

5 Extended Pseudocontact Shift **146**
5.1 Introduction . . . 146
5.1.1 Magnetic Susceptibility . . . 146
5.1.2 Anisotropy in χ . . . 149
5.1.3 The Point Dipole Model . . . 151
5.2 Explicit Forms of the Point PCS Model . . . 152
5.3 The Delocalised Point Model . . . 155
5.3.1 The Integral of $\frac{1}{r^n}$ . . . 157
5.3.2 The Integral of $\frac{1}{r^3}$ Modulated by a Function that Spherically Averages to Zero . . . 159
5.3.3 Generalised Functions . . . 160
5.3.4 $\frac{Y}{r^3}$ with a Test Function . . . 161
5.4 Review of Numerical Integration . . . 165
5.4.1 Choosing Weights . . . 166
5.4.2 Error Terms . . . 167
5.4.3 Locally Adaptive vs Globally Adaptive Algorithms . . . 168
5.4.4 The Cuhre Algorithm . . . 168
5.4.5 The Integral Smoothing Function . . . 169
5.5 Analytical Derivatives . . . 170
5.5.1 Derivatives of Error Functionals . . . 170
5.5.2 Derivatives of a Convolution . . . 171
5.5.3 Derivatives of the Gaussian . . . 171
5.5.4 Derivatives of the Point Model . . . 172
5.5.5 Finite Differencing . . . 172

5.5.6 Rescaling . . . 173
5.6 Implementation Details . . . 174
5.6.1 Input File Formats . . . 174
5.6.2 `pcsfit` Self Tests . . . 174
5.6.3 The Overall Layout of `pcsfit` . . . 175
5.6.4 Models . . . 176
5.6.5 Multithreading . . . 176
5.6.6 Notes on Integral Evaluation Precision . . . 178
5.7 Point Model Verification Against Numbat . . . 179
5.8 Discussion of the Functional Form of Extended Model . . . 180
5.9 Results and Discussion . . . 182
5.9.1 Results . . . 182
5.9.2 Potential Differences in Predicted Structures . . . 186
5.9.3 The Width Parameter . . . 189
5.9.4 Conclusions . . . 190
5.10 Acknowledgements . . . 192

6 Overall Conclusions and Outlook **193**
6.1 Accuracy of the State Space Restriction Approximation . . . 193
6.2 Spin XML Graphical User Interface . . . 196
6.3 Direct Structure Fitting . . . 197
6.4 Extended Pseudocontact Shift . . . 198
6.4.1 Isosurfaces and the Poisson Equation . . . 199

A Definitions Relating to Lie Algebras **201**
A.1 Groups and their Representations . . . 201
A.2 Lie Groups . . . 203
A.3 Lie Algebras and Lie Algebra Representations . . . 203

B $SO(3)$ and its Representations **205**

C Spin XML Schema **207**

D Graphical User Interface Build Process **211**
D.1 Building on the Linux Operating System . . . 212
D.2 Building on this Windows Operating System . . . 212

E Regularisation and Ill-Posed Problems **215**

F Overview of Optimisation Techniques **217**
F.1 BFGS 217
F.2 Nelder-Mead 217

G `pcsfit` Usage **219**
G.1 Calling `pcsfit` From the Command Line 219
G.2 Examples 220

H Raw PCS Data **221**

List of Figures

1.1 State space restriction in Liouville space 21
1.2 A metal ion with an anisotropic magnetic susceptibility tensor 47

2.1 Coherence flow between spin order subspaces 51
2.2 The Liouvillian Matrix 55
2.3 Time dependent behaviour of the continuous approximation of the density matrix norm dynamics 61

3.1 Relationship between span, skew, and the powder pattern 70
3.2 Illustration of the ZYZ Euler angle rotation convention 76
3.3 An example of a “railroad” or “syntax” diagram. 82
3.4 The anonymous complex type of the spin_system element. 83
3.5 The *reference frame* type 84
3.6 Spin complex type 85
3.7 Interaction complex type 85
3.8 Orientation complex type 87
3.9 The vector and matrix named complex types 88
3.10 Screenshot of the GUI main window 90
3.11 The interaction tensor visualisation panel 92
3.12 Screenshot of the orientation dialogue 95
3.13 Screenshot of the EasySpin export dialogue 97
3.14 EasySpin line broadening tab 98
3.15 Options tabs for pepper, chili and garlic 99
3.16 The EasySpin dynamics tab 100
3.17 The “visual hash” scheme 101
3.18 Interconversions between rotational and interaction representations. 104
3.19 Applying an isometry to the position of a conceptual camera 111
3.20 The Viewing Frustum 113
3.21 The overlay coordinate system 114

3.22 The OpenGL picking name tree for the spin system 3D Display. 117

4.1 2D plots of the least squares error function between two typical spectra . . 123
4.2 Two hypothetical minimisation problems 129
4.3 Typical call graph of the `DiStruct` error functional 131
4.4 H_2CCN optimisation error functional and gradient norm vs iteration number 134
4.5 $HCCCH_2$ optimisation error functional and gradient magnitude vs iteration number 134
4.6 Tyrosyl optimisation error functional and gradient magnitude vs iteration number. 134
4.7 Powder spectra of Salmonella and E.Coli RNR 135
4.8 The signatures of forward and reverse transform functions 139
4.9 Removal of redundant coordinates relating to rotation and translation 141
4.10 Tyrosyl radical 142
4.11 A discontinuous jump in the position of the global minima 143

5.1 Schematic of the functions f, g_U and g_L 158
5.2 Effects of varying axiality and rhombicity on the point model 180
5.3 Effects of varying axiality, rhombicity, and the width parameter on the extended model 181
5.4 Point model fitting results 183
5.5 Delocalised model results 183
5.6 Scatter plots of experimental vs theoretical PCS 186
5.7 Contour plots of the predicted PCS 187
5.8 Contour plots of the predicted PCS II 188
5.9 Optimum value of the error function for fixed values of s 190

B.1 Regular representation of $SO(3)$ 206

Chapter 1

Introduction

1.1 Objectives, Motivations and Chapter Outlines

Until recently, the fully quantum mechanical simulation of spin systems was believed to be an exponentially scaling problem in the number of spins and, consequently, full quantum mechanical simulations could only be performed on small spin systems, typically of around 10 spins at the most.

Techniques are known that, in some cases, can provide good approximations of quantum mechanical systems without exponential scaling. One particularly successful example, density matrix renormalisation theory, was first introduced by White [3] in the context of of the Heisenberg model of a chain of coupled spins. It has since been applied to a variety of other strongly correlated quantum mechanical systems. Two recent reviews may be found here: [4, 5].

Recently, a new technique has emerged called the *state space restriction* (SSR) method [6] that seemed to hold the promise of opening much larger systems to full quantum mechanical treatment (at the time, it was verified only through comparisons with experiments). The first code base to implement state space restriction was the *Spinach Library*, written in `MATLAB`, and when reference to a specific implementation of state space restriction is required in this work, Spinach will often be used as an example.

Much of the work presented here became either possible or relevant as a consequence of this development. The research topics covered, broken down by chapter, are as follows:

Chapter 1. From section 1.2 onward this chapter builds up the theoretical background behind spin dynamics, beginning with the concept of spinors and working up to the density matrix formalism and Liouville Space, concepts from the mathematical bedrock upon which the entirety of spin dynamics is built. Next, relaxation to thermal equilibrium is reviewed with emphasis on Bloch-Redfield-Wangsness theory in particular, as this will prove important in chapter 2.

The chapter will then move on to review state space restriction. There are two basic pictures: the linear algebra picture and the algebra-over-a-field picture. The linear algebra picture phrases the technique in terms of vectors and matrices and it is this picture which most clearly demonstrates what constitutes state space restriction, even though some the underlying structure of the problem is obscured.

The algebra-over-a-field picture elucidates more of the underlying mathematical structure and provides a language in which both efficient restricted operator construction and the error analysis presented in chapter 2 can be discussed.

Chapter 2. The first clear deficiency of state space restriction was the lack of formal analysis and understanding of why the state space restriction technique worked; only speculative arguments existed. Although this did not prevent the use of state space restriction, formal analysis is advantageous for two reasons: firstly because by knowing the validity conditions of the SSR approximation improper use can be prevented and secondly a more thorough understanding of state space restriction might suggest refinements or improvements.

This chapter presents such an analysis of state space restriction that was recently published in the *Journal of Chemical Physics* [1]. The approximation is shown to be valid over a very broad set of criteria and a method of estimating the maximum spin order required to obtain an accurate simulation is presented. Although only spin order pruning is considered here, this is enough to conclude that spin dynamics is a polynomially scaling problem, rather than an exponentially scaling one, when reasonable assumptions about the relaxation rates are made.

Chapter 3. With state space restriction established as a valid approximation, a more practical problem presents itself: that of actually specifying the system. While providing the interaction parameters for a system of five spins in plain text by hand is feasible, doing the same for a system of 200 spins is tedious and potentially error-prone. At the same time, there is very little insight to be gained about the physical properties of a spin system from looking at the interaction parameters presented as columns of numbers.

These two problems were addressed by the development of a bare bones graphical user interface for the creation, modification and visualisation of spin systems at the interaction parameter level. The goal of this work was to develop the interface to the point where all parts of the spin system may be edited and modified in some manner, providing a solid foundation for more advanced interaction methods. A few more advanced features are also present, such as EasySpin integration.

The interface was based on the data model used by the spin XML file format, a format for the interchange of spin system interaction parameters, recently developed through consultation with the spin dynamics community (see the acknowledgements in section 3.11).

As, up until now, spin XML was only informally specified in terms of an XSD schema, so use of this format requires a full elucidation of the semantics of the format. Consequently, the chapter itself opens with a formal specification of the spin XML format and data model, followed by the functional specification of the user interface. The rest of the chapter documents the internal details of the code and is likely to be of interest to future maintainers.

Chapter 4. With the problems of using state space restriction tackled, the question of applications then arises. Of course, spin dynamics simulations already have numerous applications in the fields of both NMR and EPR and the ability to perform those simulations faster is unarguably a good thing, but more interesting examples of applications are those that were previously infeasible but become possible when the spin dynamics step becomes polynomially scaling.

One such technique is the so-called "direct structure fitting", where the experimental spectrum is fitted directly against a theoretical one computed from interaction parameters derived from a candidate structure via *ab initio* techniques, sidestepping the manual assignment step entirely. The promise of this technique is that the obtained structures are likely to be better reflections of reality than those calculated via *ab initio* techniques, such as DFT minima alone due to the injection of experimental data into the result while at the same time requiring little human intervention as manual assignment of spectral lines is not needed.

The chapter presents work on direct structure fitting based on a paper published in the *Journal of Magnetic Resonance* [2]. In addition to a description of the general technique and discussion of examples of the technique applied to molecules, this chapter also discuses a few potential pitfalls that arose during the investigation that are not mentioned in the paper.

Chapter 5. The final piece of work presented here, while not relating to state space restriction, is also pertinent to structure determination, this time in metalloproteins where the metal contains an unpaired electron. The magnetic dipole of this electron, in cases were the susceptibility tensor is not isotropic, causes an additional spectral shift on top of the chemical shift to be observed. This new shift, called the pseudocontact shift, is well known and often used to provide long range distance in structure determination.

It is usual to treat such systems as containing a point electron when calculating pseudocontact shifts, however, in reality electrons are always found delocalised to a certain extent. The ultimate objective of this work would be to investigate the differences that using a more realistic, delocalised model of the electron would bring to the final calculated metalloprotein structure, given a set of experimental data. Unfortunately because of time constraints, only the first half of this objective was achieved, namely the modelling of the pseudocontact shift of a delocalised electron and fitting the susceptibility tensor to experimental data.

A delocalised mode pseudocontact of the shift (PCS) is suggested in this chapter. The extension of the point model to the delocalised model presents a number of mathematical subtleties which are dealt with. A new program for the fitting of the susceptibility tensor using the delocalised model is presented which runs in a reasonable time on typical desktop hardware. Finally, published PCS data sets are examined and a data set for which the delocalised model offers a significant improvement over the point model is identified. The chapter ends with speculation about the effects that the use of the delocalised model would have on a structure obtained using the PCS distance constraints.

1.2 Background Theory

The background theory behind magnetic resonance is a large subject about which a great deal has already been written (for a non-exhaustive list of examples [7–9]). This section will attempt to summarise this body of theory in a concise way with an emphasis going to aspects required in computer simulations of experiments. To this end, all treatment will be quantum mechanical and will ignore analytic techniques such as product operations and the vector model that are not required for simulations.

1.2.1 The Rotation Group

Angular momentum, in both classical and quantum mechanics, is typically represented by a three-component real vector while spin, rather curiously, is represented by pair of complex numbers. This situation appears mystifying; the three-component vector seems to encode the required information more clearly, particularly in the case of the classical angular momentum vector where each component is associated with a Cartesian direction.

Theoretical justification for this form of the spin state may be found directly from the Dirac equation [10, 11] where the concept of spin emerges naturally, but the justification for this formalism ultimately comes from comparison with experiments [10], such as the Stern–Gerlach experiment, that show that particles with spin $\frac{1}{2}$ have two energy eigenvalues, suggesting that the operators that act on these states are two-dimensional.

At the heart of the theory of spin dynamics is the rotation group:

Definition 1.1. *Special orthogonal group (rotation group).*

$$SO(3) = \{R \in M_3(\mathbb{R}) : \det R = 1, RR^T = 1\} \tag{1.1}$$

It receives its name thanks to the property that each matrix from this set defines a unique rotation of the vectors in 3D space via matrix-vector multiplication. There are, however, other mathematical objects that can be rotated other than simple 3D vectors. Take smooth

functions defined over three-dimensional space, $\phi : \mathbb{R}^3 \to \mathbb{C}$. The space of all such functions is infinite-dimensional—it is a Hilbert space—but there are certain subspaces within this full space that "transform into themselves", meaning that if a function sits within one of these spaces before a rotation, then it will remain in that space after the rotation no matter which rotation is chosen.

These "invariant subspaces" are described in detail in appendix B, but the key result is that the spherical harmonics of a particular l value serve as a basis for each subspace and that a matrix corresponding to each rotation that encodes the action of the rotation in this basis set may be constructed.

In quantum mechanics, the state of a system is encoded as a vector in a Hilbert space and the dynamical variables appear as linear operators acting on those vectors. In order for the results of experiments performed at different locations and orientations to be reconciled there has to exist a set of mathematical rules for transforming the results. Under the conditions that such transformations preserve the eigenvalue spectrum and transform Hermitian operators into other Hermitian operators, it can be concluded [10, p. 104] that the dynamical variables and state vectors are transformed via unitary transformations. In the case of rotations, these transformations must come from a representation of the rotation group and, so, the subspaces that mix among themselves must have odd numbers of dimensions.

This result is exactly what is seen in the theory of angular momentum involving scalar particles (particles whose positional representation is of the form $\phi : \mathbb{R}^3 \to \mathbb{C}$) and in vector particles (particles that have a positional representation of $\phi : \mathbb{R}^3 \to \mathbb{C}^3$ such as that of a spin 1 nucleus). As there appears to be nothing "in between" a scalar field and a vector field in the language of tensor algebra, the appearance of any particle with two eigenvalues that mixed under rotations would be very surprising. Of course, this is exactly what is observed for an electron passing through a Stern–Gerlach type experiment.

1.2.2 Spinors

In the Stern–Gerlach experiment a beam of particles is passed thought a magnetic field with a gradient. The component of the magnetic moment in the direction of the magnetic field determines the magnetic force applied to the particle and the deflection angle thus provides an measurement of the magnetic moment (and thus angular momentum) along that axis. As angular momentum along a particular axis is quantized, particles emerge at discrete deflections corresponding to each possible projection of angular momentum. The Stern-Gerlach experiment was one of the first experiments to observe quantized angular momentum and that the electron was a spin $\frac{1}{2}$ particle [12, p. 487].

If a beam of electrons in a known spin state is passed through a set of Stern–Gerlach apparatus that is rotated, the results seem to indicate that the electron possesses a magnetic moment that transforms like a vector under the rotation group $SO(3)$. However, the eigenvalue spectrum contains only two values and so the vector-valued representations of $SO(3)$ are ruled out.

The solution to this dilemma comes from the observation that $SO(3)$ is not "simply connected" in the topological sense. $SO(3)$ fulfils the criteria required to possess a unique universal cover [13, p. 152]. This covering group, which is a double cover, is $SU(2)$ [14].

Definition 1.2. *Special unitary group, $SU(n)$, is a group defined as:*

$$SU(n) = \{U \in M_n(\mathbb{C}) : \det U = 1, UU^\dagger = 1\}. \tag{1.2}$$

With the group product being matrix multiplication.

The fact that the rotation group is not simply connected can be shown visually via a demonstration accredited to Dirac and often called the "Dirac belt trick" (Staley describes the Dirac belt trick in more detail in [15]). Take a belt (or any long, thin strip) and twist one end through an angle of 2π. Using only translations of the ends of the belt, try to untwist the belt. It should turn out to be impossible. Now repeat the experiment with a 4π twist. This time, untwisting should be possible. A vector which is attached to the belt and slid continuously from end to end is clearly being continuously rotated through a path in the full space of possible rotations. If the vector is rotated through 2π it travels along a closed loop in the space of possible rotations but, as the demonstration shows, this loop cannot be shrunk to a point so $SO(3)$ is not simply connected. The 4π rotation can, demonstrating that the larger $SU(2)$ space is simply connected.

$SU(2)$ possesses more vector valued representations than $SO(3)$ [14] and these additional representations may be interpreted as being representations of $SO(3)$. As, thanks to the nature of the double covering, exactly two elements of $SU(2)$ map onto each element of $SO(3)$ then for any faithful representation of $SU(2)$, two matrices from that representation have to be interpreted as representing the same rotation.

An example of this distinction can be found in section 3.3.3 where rotations are represented as normalised quaternions. The normalised quaternion group under quaternion multiplication is isomorphic to $SU(2)$ under matrix multiplication. To see this, note that a general member of $SU(2)$ may be written in terms of the identity and the Pauli matrices:

$$\alpha\mathbb{1} + \beta i\sigma_x + \gamma i\sigma_y + \delta i\sigma_z \quad \text{subject to} \quad \alpha^2 + \beta^2 + \gamma^2 + \delta^2 = 1, \text{ and } \alpha, \beta, \gamma, \delta \in \mathbb{R}, \tag{1.3}$$

but $\mathbb{1}$, $i\sigma_x$, $i\sigma_y$ and $i\sigma_z$ behave under matrix multiplication exactly like the defining relations of the multiplication of 1, i, j and k from the quaternion algebra. When representing

rotations as quaternions, when q is a normalised quaternion, q and $-q$ represent the same rotation.

When dealing with spinor valued particles, the double cover is responsible for the strange effect that rotating a spin state vector through an angle of 2π flips the sign. Under most circumstances in spin dynamics[1], the sign change in the state vector under a 2π rotation does not lead to observable effects. However, Aharonov *et. al.* [16] were able to propose an exotic hypothetical experiment where a physical consequence of 2π rotations becomes observable and, later, in 1975 Rauch *et. al.* [17] and Werner *et. al.* [18], were able to verify this property experimentally using neutron diffraction.

As with all quantum states, the dynamical variables appear as operators acting on the state vector. A measurement of angular momentum in the direction of a unit vector, $\boldsymbol{n}$, may be written in terms of the three Pauli matrices as [10] $\boldsymbol{n} \cdot \hat{\boldsymbol{\sigma}}$. This operator fixes the physical interpretation of the two-component spinor.

The total state function of a spin $\frac{1}{2}$ particle is, therefore, a function $\phi : \mathbb{R}^3 \times \{\frac{1}{2}, -\frac{1}{2}\} \to \mathbb{C}$, or equivalently, a pair of $\mathbb{R}^3 \to \mathbb{C}$ functions, as the fourth argument of ϕ takes only one of two possible values. When these two spatial functions are equal, expectation values of spatial dynamical variables are not affected by the spin and, visa versa, and the problem may be decomposed into a spin part and a spatial part. Computationally, this is a very useful assumption to make, as difficult to handle spatial degrees of freedom may be thrown out of calculations, leaving behind a simple two-component spinor.

This assumption will begin to look strained when accounting for the spin orbit coupling in section 1.6.6, but it will turn out to be possible to maintain the fiction by encoding the effects of the spatial variables to second order as a tensor.

1.2.3 The Dynamics of a System of Spins

Let $\mathcal{H}_i$ be the full vector space of single spin state vectors for spin i (this space is $\mathbb{C}^{2s+1}$ with s being the total spin of particle i). The wavefunctions of two particle quantum mechanical systems live in the space $\mathcal{H}_1 \otimes \mathcal{H}_2$ with $V = A \otimes B$ implying that vector space V contains all pairs of vectors from vector spaces A and B. Likewise, three particle wavefunctions live in $\mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \mathcal{H}_3$ and so on.

The so called "product basis" (often written as $|\alpha\alpha\rangle = |\alpha\rangle \otimes |\alpha\rangle$, $|\alpha\beta\rangle = |\alpha\rangle \otimes |\beta\rangle \ldots$ etc. with α corresponding to spin up and β corresponding to spin down) provides an easy way to calculate a basis for representing the product spaces. It should be noted that this basis does not correspond to energy eigenvectors except when there are no interactions.

[1] And certainly all circumstances relevant to this work.

The single particle spin operators need to be expanded into this new basis. The expected behaviour of the expanded operator acting on the n^{th} spin for a system in state $|\, s_0 s_1 \ldots\rangle$ is:

$$\begin{aligned}
\hat{S}_{n,z}\,|\, s_0 s_1 \ldots s_n \ldots\rangle &= \lambda\hbar\,|\, s_0 s_1 \ldots s_n \ldots\rangle \quad \text{where } \lambda = \frac{1}{2} \text{ if } s_n = \alpha \text{ or } -\frac{1}{2} \text{ if } s_n = \beta, \\
\hat{S}_{n,+}\,|\, s_0 s_1 \ldots s_n \ldots\rangle &= \lambda\hbar\,|\, s_0 s_1 \ldots \bar{s}_n \ldots\rangle \quad \text{where } \lambda = 0 \text{ if } s_n = \alpha \text{ or } \frac{1}{2} \text{ if } s_n = \beta, \\
\hat{S}_{n,-}\,|\, s_0 s_1 \ldots s_n \ldots\rangle &= \lambda\hbar\,|\, s_0 s_1 \ldots \bar{s}_n \ldots\rangle \quad \text{where } \lambda = \frac{1}{2} \text{ if } s_n = \alpha \text{ or } 0 \text{ if } s_n = \beta,
\end{aligned} \tag{1.4}$$

Where $\bar{s}_n$ implies the opposite spin to s_n. The vector operator is built as $\hat{\boldsymbol{S}}_n = (\frac{1}{2}(\hat{S}_{n,+} + \hat{S}_{n,-}), \frac{1}{2i}(\hat{S}_{n,+} - \hat{S}_{n,-}), \hat{S}_{n,z})$. In other words, each operator acts on an individual spin in the same way as the single spin operators ignoring the state of the other spins. It turns out that $\mathbb{1} \otimes \cdots \otimes \hat{\boldsymbol{S}} \otimes \cdots \otimes \mathbb{1}$ fulfils these requirements. Using these operators, the multi-spin Zeeman interaction Hamiltonian is written [7]:

$$\hat{H}_{\text{spin}} = \frac{\mu_B}{\hbar} \sum_{i=1}^{n} g_i \boldsymbol{B} \cdot \hat{\boldsymbol{S}}_i, \tag{1.5}$$

with $\hat{\boldsymbol{S}}_i$ being the vector of spin operators, defined as $(\hat{S}_x, \hat{S}_y, \hat{S}_z)$, acting on spin i and $\boldsymbol{B}$ being some vector (often the direction of the external magnetic field).

Of course, if the above were the entire spin Hamiltonian then the subject of spin dynamics would be very boring and it would not be possible to extract interesting information via magnetic resonance experiments. Fortunately, spins may both interact with each other and the external world. The general form for the spin Hamiltonian for interacting particles is:

$$H_{\text{spin}} = \sum_{i=1}^{n} \hat{\boldsymbol{S}}_{\boldsymbol{i}} \cdot A \cdot \boldsymbol{x} + \sum_{i=1, j=1, i\neq j}^{n} \hat{\boldsymbol{S}}_{\boldsymbol{i}} \cdot B \cdot \hat{\boldsymbol{S}}_{\boldsymbol{j}} + \sum_{i=1}^{n} \hat{\boldsymbol{S}}_{\boldsymbol{i}} \cdot Q \cdot \hat{\boldsymbol{S}}_{\boldsymbol{i}}. \tag{1.6}$$

This article will adopt the convention of Ernst, Bodenhausen and Wokaun's book [8], namely that terms of the form $\hat{\boldsymbol{S}}_n \cdot A \cdot \boldsymbol{x}$, $\hat{\boldsymbol{S}}_i \cdot A \cdot \hat{\boldsymbol{S}}_j|_{i\neq j}$ and $\hat{\boldsymbol{S}}_i \cdot A \cdot \hat{\boldsymbol{S}}_i$ are called linear, bilinear and quadratic terms respectively.

1.2.4 The Density Matrix Formalism

The density matrix formalism, invented by von Neumann in 1927, is an alternative method of writing down the state of a quantum mechanical system that is particularly well suited to dealing with statistical ensembles. Information on the density matrix formalism can be found in books on spin dynamics such as [7, 8, 19] but a more general account of the density matrix formalism and its uses in physics can be found in Blum's book [20]. Dirac [10] also gives a treatment of the density matrix via comparison to the phase-space density of a classical Gibbs ensemble.

A single quantum mechanical system must be described by a normalised state vector. In a particular basis set the state of the system may be a superposition (such as $\alpha\,|1\rangle + \beta\,|0\rangle$) but even then a basis may be found that corresponds to an experiment whose outcome is certain. However, when a collection of systems is involved, a distinct state vector can be assigned to each system in the ensemble and it may be no longer possible to find an experiment whose outcome is certain (mathematically: find a basis in which each state vector has only one nonzero component).

Blum [20] uses the existence of experiments whose outcome is certain to define ensembles that are in a *pure state*. Ensembles in *impure states* are then ensembles that are not in *pure states*. The physical information that can be extracted via suitable experiments from ensembles that are in impure states cannot be summarised via a single state vector. One obvious, but cumbersome, way to do this would be to have one weighted state vector per configuration in the ensemble. The result of experiment $\hat{O}$ would then be:

$$\alpha_1 \langle \Psi_1 | \hat{O} | \Psi_1 \rangle + \alpha_2 \langle \Psi_2 | \hat{O} | \Psi_2 \rangle + \ldots \quad \alpha_i \in \mathbb{R}^+. \tag{1.7}$$

Given the vast numbers of spin systems in a typical magnetic resonance experiment, the above approach is impractical. The density matrix formalism provides a much neater and more efficient way to encode the same information.

1.2.4.1 The Density Matrix of a Single System

Let $\hat{\rho} = |\psi\rangle\langle\psi|$ where $|\psi\rangle$ is the usual wavefunction of a single system. The equation of motion for $\hat{\rho}$ can be found by differentiating the definition of $\hat{\rho}$ with respect to time:

$$\frac{\partial\hat{\rho}}{\partial t} = |\psi\rangle \frac{\partial\langle\psi|}{\partial t} + \frac{\partial|\psi\rangle}{\partial t}\langle\psi|, \tag{1.8}$$

and substituting the Schrödinger equation, $i\hbar\frac{\partial|\psi\rangle}{\partial t} = \hat{H}|\psi\rangle$, and its complex conjugate $-i\hbar\frac{\partial\langle\psi|}{\partial t} = \langle\psi|\hat{H}$ resulting in the von Neumann equation:

$$i\hbar\frac{\partial\hat{\rho}}{\partial t} = \hat{H}|\psi\rangle\langle\psi| - |\psi\rangle\langle\psi|\hat{H}, \tag{1.9}$$

or alternatively,

$$i\hbar\frac{\partial\hat{\rho}}{\partial t} = [\hat{H}, \hat{\rho}], \tag{1.10}$$

where $[\,\cdot\,,\,\cdot\,]$ is the commutator.

The diagonal elements of the density matrix have a distinct physical meaning; they are the level populations of each of the basis states in $|\psi\rangle$. The off-diagonal elements are the so called "coherences" between the levels (see [7] p. 276).

The interpretation of the density matrix can be made clear by explicitly writing it out. Let $|\psi\rangle$ be a general single particle spin $\frac{1}{2}$ state:

$$|\psi\rangle = \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = c_0 |0\rangle + c_1 |1\rangle \tag{1.11}$$

The corresponding density matrix will be given by:

$$|\psi\rangle\langle\psi| = \begin{pmatrix} c_0^* c_0 & c_1^* c_0 \\ c_1 c_0^* & c_1^* c_1 \end{pmatrix}. \tag{1.12}$$

It is immediately obvious why the diagonal elements are the level populations. The Copenhagen interpretation of quantum mechanics states that the modulus squared of the wavefunction is the probability of observing a given system in a given state, exactly the quantity that appears on the diagonal.

The off-diagonal element (in this case there is only really one as $c_0^* c_1$ is the complex conjugate of $c_1^* c_0$) is harder to interpret. $|c_1^* c_0|$ is zero when the system is fully in the $|0\rangle$ or $|1\rangle$ state and nonzero when the system is in an intermediate state. The phase will be time dependent so long as $|0\rangle$ and $|1\rangle$ aren't degenerate. In fact the phase of $c_1^* c_0$ rotates at the transition frequency. The full meaning of the off-diagonal only emerges when an ensemble of systems is considered.

1.2.4.2 The Density Matrix Formalism for an Ensemble

Much of this discussion is based on [7, p. 276]. If each term in equation (1.7) is expanded into matrix and vector form, the result is:

$$\begin{aligned} \left\langle \hat{O} \right\rangle &= (c_0^* \; c_1^*) \begin{pmatrix} O_{00} & O_{01} \\ O_{10} & O_{11} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} + (d_0^* \; d_1^*) \begin{pmatrix} O_{00} & O_{01} \\ O_{10} & O_{11} \end{pmatrix} \begin{pmatrix} d_0 \\ d_1 \end{pmatrix} + \dots \\ &= (c_0^* c_0 + d_0^* d_0 + \dots) O_{00} + \\ &\quad (c_0^* c_1 + d_0^* d_1 + \dots) O_{01} + \\ &\quad (c_1^* c_0 + d_1^* d_0 + \dots) O_{10} + \\ &\quad (c_1^* c_1 + d_1^* d_1 + \dots) O_{11} \\ &= \mathrm{Tr}\left((\hat{\rho}_0 + \hat{\rho}_1 + \dots) \hat{O} \right). \end{aligned} \tag{1.13}$$

In other words, the expectation value of an operator applied to an ensemble can be found by taking the trace $\mathrm{Tr}(\hat{\rho}\hat{O})$ where $\hat{\rho}$ is the sum of all the individual density matrices in the ensemble. Thanks to the linearity of the von Neumann equation, the composite density matrix evolves in the same way as each individual density matrix of the system. The fact that only one matrix needs to be tracked rather than a very large number of wavefunctions is a great simplification.

It is in the context of ensembles that the usefulness of the cross terms becomes clear. In a large ensemble of systems there is no reason to suppose that the phase of each individual

cross term will be the same. In fact, if each system does posses a random relative phase between its basis states then the complex sum of all cross terms will be zero. The only nonzero terms will be the level populations on the diagonal and the behaviour of the bulk system will appear classical.

A nonzero term on the diagonal suggests that there is a tendency for the basis states linked by the cross term to be in phase, hence the name "coherences". Coherences are vital in the theory of magnetic resonance as coherences, rather than populations, are detectable.

In NMR and EPR the resonant system exchanges with the electromagnetic field. As each exchange results in the absorption or emission of a photon of spin $\hbar$, then only coherences between states which differ in angular momentum of $\hbar$ are allowed transitions.

Each element of the density matrix is the product of two complex constants, c_n and c_m^*. If the basis has been chosen so that each state has a well defined total angular momentum then, by convention, the angular momentum state contributing the c_n symbol subtracted from the angular momentum of the state contributing the c_m^* symbol is known as the "coherence order". In the example of the 2×2 density matrix, the terms on the diagonal are "coherences of order zero", while the upper diagonal is a (+1) quantum coherence and the lower diagonal is a (-1) quantum coherence.

1.2.5 The Initial State

The initial density matrix is usually called ρ_0. All coherences are assumed to have relaxed leaving only the diagonal population terms of the density matrix nonzero. Without loss of generality it will be assumed that the basis has been chosen to be eigenstates of the Hamiltonian. Due to the classical nature of this system, the relative populations of the states can be worked out by the application of the Boltzmann distribution [7, p. 282]:

$$\rho_{n,n} = \frac{\exp\left(-\hbar\omega_n / k_B T\right)}{\sum_i \exp\left(-\hbar\omega_i / k_B T\right)}, \tag{1.14}$$

where i runs over all states and $\hbar\omega_i$ is the energy of state i. The relative populations of two states $\frac{\rho_{n,n}}{\rho_{m,m}}$ is given by:

$$\frac{\rho_{n,n}}{\rho_{m,m}} = \exp\left(\hbar(\omega_m - \omega_n)/k_B T\right). \tag{1.15}$$

In general the transition energies between states, $\hbar(\omega_m - \omega_n)$, will be much smaller than $k_B T$, so at room temperatures the populations of the states will be very nearly equal with only a slight overabundance of the lower energy state with respect to the higher energy one. Because $\frac{\hbar(\omega_m - \omega_n)}{k_B T}$ is small, the exponentials in equation (1.14) can be expanded as Taylor series, and high order terms can be ignored. For a two spin system, equation (1.14) becomes:

$$\rho_0 = \frac{1}{2}\mathbb{1} + \frac{1}{2}\frac{\hbar\omega}{k_B T}\begin{pmatrix} \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} \end{pmatrix} \quad \text{where } \omega = \omega_1 - \omega_2. \tag{1.16}$$

The behaviour of the identity matrix is uninteresting, and through the linearity of the von Neumann equation, the identity can be treated separately (and ignored). The second term is proportional to the familiar $\hat{L}_z$ operator, which is the starting point for most NMR and EPR simulations.

1.2.6 Observables in Standard Apparatus

As the focus of this document is on simulation of magnetic resonance experiments, there will not be an extended discussion about real experimental setups, rather, only the important details will be mentioned.

In an NMR setup, the total magnetism of the sample, $\boldsymbol{M}$, is detected via two sets of coils positioned at right angles to each other. Two sets of coils are required in order to resolve the sign of M_x due to the fact that the signal must be summed with a reference frequency in order to produce a beat frequency low enough to be recordable with analog-to-digital converters [7, p. 80].

The quantum mechanical operator that corresponds to this setup is S_+ [7, p. 443], meaning that the output signal is given by:

$$M_x \propto \mathrm{Tr}(\hat{\rho}^\dagger \hat{S}_+). \tag{1.17}$$

1.2.7 Liouville Space

The space of all linear operators over a vector space forms its own linear space. It will prove useful to define an inner product over this space as $\langle A, B\rangle = \mathrm{Tr}(A^\dagger B)$. This inner product satisfies the requirements of an inner product over a complex vector space.[2] In fact, by manually expanding the expression $\mathrm{Tr}(A^\dagger B)$, it can be seen that this inner product matches the usual definition of the inner product over a complex vector space:

$$\begin{aligned}\mathrm{Tr}(A^\dagger B) &= \mathrm{Tr}\left[\begin{pmatrix} A_{00}^* & \cdots & A_{n0}^* \\ \vdots & \ddots & \vdots \\ A_{0n}^* & \cdots & A_{nn}^* \end{pmatrix}\begin{pmatrix} B_{00} & \cdots & B_{0n} \\ \vdots & \ddots & \vdots \\ B_{n0} & \cdots & B_{nn} \end{pmatrix}\right] \\ &= A_{00}^* B_{00} + A_{01}^* B_{01} + \cdots + A_{ij}^* B_{ij} + \cdots + A_{nn}^* B_{nn}.\end{aligned} \tag{1.18}$$

Because linear operators live in a vector space, there is clearly an even larger space of linear operators (the superoperators) that acts on the linear operators. These are called the superoperators and are denoted by a double hat, $\hat{\hat{O}}$. Although this linear space is simply a Hilbert space of larger dimension than the original Hilbert space, it is important enough to receive its own name.

[2] These are conjugate symmetry, $\langle A, B\rangle = \langle B, A\rangle^*$; linearity, $\langle A + B, C\rangle = \langle A, C\rangle + \langle B, C\rangle$ and positive definiteness, $\langle A, B\rangle \geq 0$

Definition 1.3. *Let $\mathcal{H}$ be a Hilbert space of quantum mechanical state vectors. The Liouville space generated from $\mathcal{H}$ is the vector space of linear operators acting on $\mathcal{H}$.*

As the left hand side of the von Neumann equation, $[\hat{H}, \hat{\rho}]$, is linear, a commutation superoperator should exist. It turns out that the correct form of this superoperator is $\hat{H} \otimes \mathbb{1} - \mathbb{1} \otimes \hat{H}^{\mathrm{T}}$ [8, p. 23]. This superoperator is called the "Liouvillian" and is conventionally designated $\hat{\hat{L}}$ or $\hat{\hat{H}}$. In this work, the convention will be adopted that $\hat{\hat{H}}$ is the promotion of the system Hamiltonian to Liouville space and $\hat{\hat{L}}$ is the total evolution operator including any relaxation (see section 1.3) or chemical kinetics.

In Liouville Space, the von Neumann equation begins to look very familiar:

$$i\hbar \frac{\partial \hat{\rho}}{\partial t} = \hat{\hat{H}} \hat{\rho}. \tag{1.19}$$

It has the exact same form as the Schrödinger equation and, so, has the same solution:

$$\hat{\rho}(t) = \exp(-i\hat{\hat{H}}t/\hbar)\hat{\rho}_0. \tag{1.20}$$

Using Liouville Space looks to be very wasteful from a simulation perspective as the the size of the Liouvillian is the square of the size of the Hamiltonian. However, there are cases that can be solved in Liouville space that cannot in Hilbert space. Notably, any calculations involving relaxation needs to use Liouvillian space because the Redfield matrix is a superoperator [8, p. 50]. The effects of chemical kinetics can also be introduced via a superoperator [8, p. 67] although other treatments are possible, such as coupling sets of Bloch equations for each species [8, p. 60].

1.2.7.1 Time Domain Calculation by Propagators

Now that the initial state and the equation of motion is known, it is, in principle, possible to solve the dynamics problem for the entire system. One approach would be to simply calculate the matrix exponential in equation (1.20) but this would give a unitary operator that moves the system from its initial to final state. This is fine for simulating very short events such as π-pulses but current matrix exponentiation algorithms working on finite precision hardware may become unstable if t is too great. Furthermore, we are interested in the evolution rather than the final state so it would be nice to be able to calculate intermediate states without having to perform multiple matrix exponentiations.

For these reasons, an new unitary operator is calculated instead, $\exp(-i\hat{\hat{H}}\Delta t/\hbar)$, where $\Delta t = \frac{t}{n}$ where t is the total time and n is a number at least big enough so that Δt is sufficiently small for the matrix exponential to be stable [21]. Furthermore, n must also be big enough that $\frac{1}{2\Delta t}$ is larger than the biggest frequency in the FID to satisfy the Nyquist sampling criterion.

This operator is typically called the propagator and given the symbol $\hat{\hat{P}}$. The density matrix, $\hat{\rho}_0$, is repeatedly multiplied by $\hat{\hat{P}}$ to generate a series of density matrices representing the history of the system: $\rho(\hat{\Delta}t), \rho(2\hat{\Delta}t), \ldots$

The principles that are reviewed so far in this chapter are all that are needed to simulate any spin system without relaxation if the Hamiltonian is known; indeed, such a simulation program can be written in just a few lines of `MATLAB` code. However, the sizes of the matrices involved ($\hat{\hat{L}}$, $\hat{\hat{P}}$) are $4^n \times 4^n$ (or bigger if spins higher than $\frac{1}{2}$ are involved) and so the simulation's run time and memory requirements grow exponentially, and calculation using these direct methods rapidly becomes impractical.

1.2.8 The Secular Approximation

A critical approximation that is made in spin dynamics is the so called "secular approximation." In particular, it will be important in chapter 2 where disregarding the so called "non-secular" terms will simplify the analysis greatly.

Let the Hamiltonian be composed of two parts, $\hat{H} = \hat{A} + \hat{B}$, where $\hat{A}$ and $\hat{B}$ are individually Hermitian and $\hat{A}$ is "large" while $\hat{B}$ is "small". A fairly typical example of the terms that end up in $\hat{A}$ is the Zeeman interaction, as the external static field in magnetic resonance experiments are typically much stronger than the radio-frequency fields.

We will begin with the 2×2 case. It will be convenient to work in an explicit basis, and we chose the eigenbasis of $\hat{A}$. In this basis, the that the components $\hat{A}$ and $\hat{B}$ are:

$$\hat{A} = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \end{pmatrix} \quad \hat{B} = \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{1,2}^* & b_{2,2} \end{pmatrix}. \tag{1.21}$$

Let $\hat{U}$ be the unitary operator makes $\hat{U}(\hat{A}+\hat{B})\hat{U}^\dagger$ diagonal. The components of this operator are:

$$\hat{U} = \begin{pmatrix} \alpha & -\beta^* \\ \beta & \alpha^* \end{pmatrix} \tag{1.22}$$

with the condition that $\alpha\alpha^* + \beta\beta^* = 1$.

In order to diagonalise $\hat{A} + \hat{B}$, α and β must be chosen so the off diagonal elements of $\hat{U}(\hat{A} + \hat{B})\hat{U}^\dagger$ are zero. By explicitly multiplying this expression out, we find the condition:

$$\langle a_1 | \hat{U}(\hat{A} + \hat{B})\hat{U}^\dagger | a_2 \rangle = \alpha\beta^*(a_1 - a_2 + b_{1,1} - b_{2,2}) + \alpha^2 b_{1,2} + (\beta^2 b_{1,2})^* = 0 \tag{1.23}$$

If $a_1 - a_2$ is large in comparison to $b_{1,2}$ then the last two terms may be neglected as being first order. Then either α or β^* must be very nearly zero (but not both, remembering $\alpha\alpha^* + \beta\beta^* = 1$). We choose β to be close to zero to avoid shuffling the basis in a manner that has no physical effect.

The corrections to the diagonal are given by:

$$\Delta a_1 = \alpha\alpha^* b_{1,1} + \beta\beta^*(a_2 + b_{2,2}) - \alpha\beta b_{1,2} - (\alpha\beta b_{1,2})^* \tag{1.24}$$

$$\Delta a_2 = \alpha\alpha^* b_{2,2} + \beta\beta^*(a_1 + b_{1,1}) - \alpha\beta b_{1,2} + (\alpha\beta b_{1,2})^* \tag{1.25}$$

However, with β and $b_{1,2}$ both assumed small, the second and third terms may be neglected as being second and the $\alpha\alpha^*$ factor may be dropped because it is close to 1. Thus, when the off diagonal elements of $\hat{B}$ are small, they may be neglected and the eigenvalues of $\hat{A} + \hat{B}$ can simply be taken as the diagonal of $\hat{A} + \hat{B}$. Operators of higher dimension can treated analogously finding a series of unitary operators that zero each off-diagonal element in turn.

This justifies the condition given by Levitt [7] that terms in $\hat{B}$ for which $|b_{m,n}| \ll |a_m - a_n|$ may be neglected in the secular approximation.

1.3 Relaxation Theory

In section 1.2, all the dynamical behaviours of spin systems considered were completely reversible processes and involved no losses of energy or coherence to the environment. Clearly at least one type of irreversible process must take place during the time evolution of a spin system as detection must involve the spin system radiating photons. However, Abragam estimates [19] the timescales for spontaneous and stimulated emission through the coupling of a dipole with the electromagnetic field as 10^{25} and 10^{10} seconds respectively and concludes that these processes are negligible. In real spin systems, the primary source of relaxation comes from the coupling of the spins with the spatial degrees of freedom[3] of the system rather than the coupling to the electromagnetic field.

There are a multitude of different models for spin-lattice relaxation in spin dynamics that make varying trade-offs between accuracy and ease of computation. Of these, the Bloch-Redfield-Wangsness model of relaxation is particularly important to the Spinach[4] code base as it is used there to model relaxation [22, 23] and will be the model of relaxation assumed in chapter 2. Consequently, it will be reviewed in detail in section 1.3.1.

1.3.1 Bloch-Redfield-Wangsness Relaxation

The Bloch-Redfield-Wangsness relaxation model, also known as simply "Redfield theory", grew out of a theory first proposed by Wangsness and Bloch [24] in an effort to identify the regions of validity of Bloch's phenomenological model of relaxation [25]. Redfield, in a paper published 1957 [26], improved on this model to properly deal with non-diagonal Hamiltonians.

[3] Often confusingly called the "lattice" even when no crystalline structure is implied.
[4] Software package using state space restriction. Covered in more detail in section 1.4.

The Redfield model approximates the effects of an ensemble of quantum mechanical systems interacting with a heat bath up to second order in perturbation theory. A derivation of the theory is presented here that will closely follow Goldman's 2000 review paper on Spin-Lattice Relaxation [27] but where appropriate Liouville space methods will used which, in the author's opinion, clarifies some of Goldman's points. In particular, the proof of the existence of Goldman's $\{\hat{V}_\alpha\}$ basis can be skipped entirely.

We will consider an ensemble of identical systems evolving under Hamiltonians of the form $\hat{H} = \hat{H}_0 + \hat{H}_1(t)$ where $\hat{H}_0$ is a static operator common to all the systems while $\hat{H}_1$ is a time dependent, stochastic operator that averages to zero over the ensemble and is different for every member system.

Theorem 1.4. *The decomposition $\hat{H} = \hat{H}_0 + \hat{H}_1(t)$ is always possible.*

Proof. If a proposed $\hat{H}_1$ were to fail to average to zero then the time average value, $\overline{\hat{H}}_1$, could be absorbed by $\hat{H}_0$ because this value does not depend on time. □

We will begin considering a single system within the ensemble. The dynamics are governed by the Liouville von Neumann equation:

$$\frac{\partial \hat{\rho}}{\partial t} = -\frac{i}{\hbar}[\hat{H}_0 + \hat{H}_1(t), \hat{\rho}]. \tag{1.26}$$

It is useful to construct the problem in the *interaction picture*. The *interaction picture* is a formulation of quantum mechanics somewhere between the Heisenberg and the Schrödinger pictures. In the Schrödinger picture, the wavefunction evolves with time while operators remain static, while the opposite is true of the Heisenberg picture. In the interaction picture, operators evolve under the $\hat{H}_0$ part of the Hamiltonian while, the wavefunction evolves under the $\hat{H}_1$ part. The advantage of this approach is that the effects of $\hat{H}_0$ become implicit and are completely handled by the formalism.

To convert operators from the Schrödinger picture to the interaction picture, they are conjugated with $\exp(i\hat{H}_0 t/\hbar)$:

$$\tilde{O} = \exp(i\hat{H}_0 t/\hbar)\hat{O}\exp(-i\hat{H}_0 t/\hbar). \tag{1.27}$$

Interaction picture operators will appear marked with a tilde accent, as $\tilde{O}$ (the usual "hat" will be implied by the tilde). In the following derivation, $\exp(-i\hat{H}_0 t/\hbar)$ will be written as $\hat{U}(t)$ in order to save space.

The interaction picture Liouville-von Neumann equation is derived by applying $\hat{U}(t)$ and $\hat{U}^\dagger(t)$ to the right and left hand sides of the Schrödinger picture Liouville-von Neumann

equation respectively:

$$\begin{aligned}\hat{U}^{\dagger}(t)\frac{d\hat{\rho}}{dt}\hat{U}(t) &= -\frac{i}{\hbar}\hat{U}^{\dagger}(t)[\hat{H}_0+\hat{H}_1(t),\rho]\hat{U}(t)\\ \hat{U}^{\dagger}(t)\frac{d\hat{\rho}}{dt}\hat{U}(t) &= -\frac{i}{\hbar}[\hat{H}_0,\tilde{\rho}]-\frac{i}{\hbar}[\tilde{H}_1(t),\tilde{\rho}], \quad \text{remembering that } \hat{H}_0=\tilde{H}_0.\end{aligned} \tag{1.28}$$

The identity $\hat{U}^{\dagger}[A,B]\hat{U}=[\hat{U}^{\dagger}\hat{A}\hat{U},\hat{U}^{\dagger}\hat{B}\hat{U}]$ followed by the observation that $\hat{H}_0$ commutes with $\hat{U}$ and $\hat{U}^{\dagger}$ was used to get from the first line to the second.

The relationship between $\frac{d\hat{\rho}}{dt}$ and $\frac{d\tilde{\rho}}{dt}$ may be found by applying the product rule:

$$\begin{aligned}\frac{d\tilde{\rho}}{dt} &= \frac{d\hat{U}^{\dagger}(t)}{dt}\hat{\rho}\hat{U}(t)+\hat{U}^{\dagger}(t)\frac{d\hat{\rho}}{dt}\hat{U}(t)+\hat{U}^{\dagger}(t)\hat{\rho}\frac{d\hat{U}(t)}{dt}\\ &= -\frac{i}{\hbar}\hat{H}_0\hat{U}^{\dagger}(t)\hat{\rho}\hat{U}(t)+\hat{U}^{\dagger}(t)\frac{d\hat{\rho}}{dt}\hat{U}(t)+\hat{U}^{\dagger}(t)\hat{\rho}\frac{i}{\hbar}\hat{H}_0\hat{U}(t)\\ &= \hat{U}^{\dagger}(t)\left[\frac{i}{\hbar}\hat{H}_0\hat{\rho}+\frac{d\hat{\rho}}{dt}-\frac{\hbar}{i}\hat{\rho}\hat{H}_0\right]\hat{U}(t)\\ &= \hat{U}^{\dagger}(t)\frac{d\hat{\rho}}{dt}\hat{U}(t)+\frac{i}{\hbar}[\hat{H}_0,\tilde{\rho}],\end{aligned} \tag{1.29}$$

and by substituting equation (1.28) this gives the interaction picture Liouville-von Neumann equation:

$$\frac{d\tilde{\rho}}{dt}=-\frac{i}{\hbar}[\tilde{H}_1(t),\tilde{\rho}]. \tag{1.30}$$

The explicit dependence of the density matrix on $\hat{H}_0$ has been removed, leaving an equation similar in form to the original Liouville-von Neumann equation. For a pure Zeeman interaction on a single spin, this manipulation is equivalent to constructing the spin dynamics problem in a rotating frame.

Equation (1.30) can be formally integrated and back-substituted into itself to obtain:

$$\begin{aligned}\frac{d\tilde{\rho}}{dt} &= -\frac{i}{\hbar}[\tilde{H}_1(t),\tilde{\rho}(0)-\frac{i}{\hbar}\int_0^t dt'[\tilde{H}_1(t'),\tilde{\rho}(t')]]\\ &= -\frac{i}{\hbar}[\tilde{H}_1(t),\tilde{\rho}(0)]-\frac{1}{\hbar^2}\int_0^t dt'[\tilde{H}_1(t),[\tilde{H}_1(t'),\tilde{\rho}(t')]],\end{aligned} \tag{1.31}$$

which corresponds to equation (10) of Goldman's paper.

Real spin dynamics experiments are not performed on single spin systems, rather, they are performed on ensembles, so equation (1.31) is more properly a set of equations; one for each system in the ensemble. If state vector notation were being used, averaging would be extremely complicated. Thankfully, the density matrix formalism allows a summary of the behaviour of all the systems in question, including correlations, to be constructed by simply averaging the density matrix of each system.

Because of the linearity of the commutator, the expression $[\hat{H}_1(t),\tilde{\rho}(0)]$ in equation (1.31) averages to zero when taken over the ensemble, and can be removed:

$$\frac{d\overline{\tilde{\rho}(t)}}{dt}=-\hbar^2\int_0^t dt'\overline{[\tilde{H}_1(t),[\tilde{H}_1(t'),(\tilde{\rho}(t')-\tilde{\rho}_{\mathrm{eq}})]]}. \tag{1.32}$$

To account for finite lattice temperature, we have also replaced $\tilde{\rho}$ with $\tilde{\rho} - \tilde{\rho}_{\text{eq}}$ where $\tilde{\rho}_{\text{eq}}$ is the density matrix at thermal equilibrium. $\tilde{\rho}_{\text{eq}}$ primarily a phenomenological parameter that must be determined by experiment, although Goldman justifies the form of $\tilde{\rho}_{\text{eq}}$ in his paper [27] (see also [19]). It is often assumed that $\tilde{\rho}_{\text{eq}}$ corresponds to thermal equilibrium:

$$\tilde{\rho}_{\text{eq}} = \frac{\exp(-\tilde{H}_0/k_B T)}{\text{Tr}(\exp(-H_0/k_B T))}. \tag{1.33}$$

Again, following Goldman, $\hat{H}_1(t)$ will now be written in terms of an explicit basis set, $\{\hat{V}_\alpha\}$:

$$\hat{H}_1(t) = \sum \hat{V}_\alpha F_\alpha(t) = \sum \hat{V}_\alpha^\dagger F_\alpha^*(t), \tag{1.34}$$

where $\{F_\alpha\}$ is a set of functions of time that encode the time dependence of $\hat{H}_1$. The equality of $\sum \hat{V}_\alpha F_\alpha = \sum \hat{V}_\alpha^\dagger F_\alpha^*$ arises because $\hat{H}_1$ is Hermitian. The set $\{\hat{V}_\alpha\}$ are chosen such that $[\hat{H}_0, \hat{V}_\alpha] = \omega \hat{V}_\alpha$, or in other words, they are eigenvectors of $\hat{\hat{H}}_0$:

$$\hat{\hat{H}}_0 \hat{V}_\alpha = \omega \hat{V}_\alpha. \tag{1.35}$$

Thanks to the Liouville Space formalism, it was possible to replace Goldman's proof of existence of $\{\hat{V}_\alpha\}$ with the simple statement that Hermitian matrices have eigendecompositions. Because this basis set are eigenvectors, the transformation into the interaction picture frame is particularly simple:

$$\tilde{V}_\alpha(t) = \exp(-i\omega_\alpha t)\hat{V}_\alpha. \tag{1.36}$$

In practical calculations on large spin systems, this basis would be would be expensive to compute because it would involve diagonalising the Liouvillian.

Under expansion of $\hat{H}_1$, equation (1.32) becomes:

$$\frac{d\bar{\tilde{\rho}}}{dt} = -\frac{1}{\hbar^2} \sum_{\alpha,\beta} \int_0^t dt' \overline{[\tilde{V}_\alpha(t), [\tilde{V}_\beta^\dagger(t'), \tilde{\rho}(t') - \tilde{\rho}_{\text{eq}}]] F_\alpha(t) F_\beta^*(t')}. \tag{1.37}$$

Let the characteristic time of fluctuations in $\hat{H}_1$ be τ_c (the exact definition of τ_c depends on the system being modelled, but for now, only the approximate order of magnitude of τ_c matters). Three important assumptions will be made: 1) the fluctuations of $\hat{H}_1$ are fast enough that on the timescale of τ_c, $\tilde{\rho}$ can be treated as being static; 2) The statistical properties of the dynamics remain constant in time; 3) the system is left to evolve for much longer than τ_c (in other words $\tau_c \ll t$). As the product $\overline{F_\alpha(t) F_\beta^*(t')}$ decays very quickly, the integrand is only of significant size when t' is within a few τ_c of t.

Assumption (3) allows $\overline{\tilde{\rho}}(t')$ to be replaced with $\tilde{\rho}(t')$ as the system is assumed to have been left evolving for sufficient time for differences in the individual $\hat{H}_1$ dynamics to have averaged out, leaving each system with approximately the same density matrix:

$$\frac{d\overline{\tilde{\rho}}}{dt} = -\frac{1}{\hbar^2}\sum_{\alpha,\beta}\int_0^t dt'[\tilde{V}_\alpha(t),[\tilde{V}_\beta^\dagger(t'),\tilde{\rho}(t')-\tilde{\rho}_{\mathrm{eq}}]]\overline{F_\alpha(t)F_\beta^*(t')}. \tag{1.38}$$

The substitution $\overline{F_\alpha(t)F_\beta^*(t')} = G_{\alpha,\beta}(|t-t'|)$ is made, corresponding to the assumption (2) that the statistical properties of the random fluctuations do not change:

$$\frac{d\overline{\tilde{\rho}}}{dt} = -\frac{1}{\hbar^2}\sum_{\alpha,\beta}\int_0^t dt'[\tilde{V}_\alpha(t),[\tilde{V}_\beta^\dagger(t'),\tilde{\rho}(t')-\tilde{\rho}_{\mathrm{eq}}]]G_{\alpha\beta}(t-t'). \tag{1.39}$$

Using assumptions (1) and (3), $\tilde{\rho}(t')$ can be replaced with $\tilde{\rho}(t)$ as only when $t \approx t'$ is the integrand significant. Also using (1) it is seen that because $G_{\alpha,\beta}$ decays rapidly on the time scale τ_c, the integral can be extended to infinity:

$$\frac{d\overline{\tilde{\rho}}}{dt} = -\frac{1}{\hbar^2}\sum_{\alpha,\beta}\int_0^\infty dt' \underbrace{[\tilde{V}_\alpha(t),[\tilde{V}_\beta^\dagger(t'),\tilde{\rho}(t)-\tilde{\rho}_{\mathrm{eq}}]]}\, G_{\alpha\beta}(t-t'). \tag{1.40}$$

To obtain measurable quantities, equation (1.40) needs to be transformed back into the Schrödinger picture by sandwiching it between exponentials. The $\hat{U}$ and $\hat{U}^\dagger$ matrix exponentials may be moved within the integral as they commute with everything apart from the section marked with the underbrace. Using the identity $[\hat{U}\hat{A}\hat{U}^\dagger,\hat{U}\hat{B}\hat{U}^\dagger] = \hat{U}[\hat{A},\hat{B}]\hat{U}^\dagger$ a second time, equation (1.40) becomes:

$$\hat{U}\frac{d\overline{\tilde{\rho}}}{dt}\hat{U}^\dagger = -\frac{1}{\hbar^2}\sum_{\alpha,\beta}\int_0^\infty dt'[\hat{V}_\alpha(t),[\hat{V}_\beta^\dagger(t'),\hat{\rho}(t)-\hat{\rho}_{\mathrm{eq}}]]G_{\alpha\beta}(t-t'). \tag{1.41}$$

The differential term $\frac{d\overline{\rho}}{dt}$ is converted back by substituting in equation (1.29):

$$\frac{d\overline{\hat{\rho}}(t)}{dt} = -\frac{i}{\hbar}[\hat{H}_0,\hat{\rho}(t)] - \frac{1}{\hbar^2}\sum_{\alpha,\beta}\int_0^\infty dt'[\hat{V}_\alpha,[\tilde{V}_\beta^\dagger(t-t'),\hat{\rho}(t)-\hat{\rho}_{\mathrm{eq}}]]G_{\alpha,\beta}(t-t'). \tag{1.42}$$

The time dependence of $\tilde{V}_\beta^\dagger$ may be extracted by making the substitution $\tilde{V}_\beta^\dagger(t-t') \rightarrow \hat{V}_\beta^\dagger\exp(-i\omega_\beta(t-t'))$:

$$\frac{d\hat{\rho}(t)}{dt} = -\frac{i}{\hbar}[\hat{H}_0,\hat{\rho}(t)] - \frac{1}{\hbar^2}\sum_{\alpha,\beta}\int_0^\infty dt'[\hat{V}_\alpha,[\hat{V}_\beta^\dagger,\hat{\rho}(t)-\hat{\rho}_{\mathrm{eq}}]]\exp(-i\omega_\beta(t-t'))G_{\alpha,\beta}(t-t'). \tag{1.43}$$

It is common to rewrite $\int_0^\infty \exp(-i\omega_\beta(t))G_{\alpha,\beta}(t)dt$ as $J_{\alpha,\beta}$:

$$\frac{d\hat{\rho}(t)}{dt} = -\frac{i}{\hbar}[\hat{H}_0, \hat{\rho}(t)] - \frac{1}{\hbar^2}\sum_{\alpha,\beta}[\hat{V}_\alpha, [\hat{V}_\beta^\dagger, \hat{\rho}(t) - \hat{\rho}_{\text{eq}}]]J_{\alpha,\beta}. \tag{1.44}$$

The set $\{G_{\alpha,\beta}\}$ are phenomenological and need to be found from a physical model of the classical dynamics of the system. There are a wide variety of such dynamic models and a full review would be outside of the scope of this document. Two important examples are those modelling molecular rotation and diffusion.

Equation (1.44) may be tidied up by writing it in terms of Liouville space:

$$\frac{d\hat{\rho}(t)}{dt} = -\frac{i}{\hbar}\hat{\hat{H}}_0\hat{\rho}(t) - \frac{1}{\hbar^2}\left[\sum_{\alpha,\beta} J_{\alpha,\beta}\hat{\hat{V}}_\alpha\hat{\hat{V}}_\beta^\dagger\right](\hat{\rho}(t) - \hat{\rho}_{\text{eq}}). \tag{1.45}$$

The part in the square brackets has the form of a superoperator and is usually called the Redfield relaxation superoperator and given the symbol $\hat{\hat{R}}$. Equation (1.45) is given the name "master equation".

1.4 State Space Restriction

The basic theory discussed so far is in principle sufficient to model any spin system of any number of spins, so long as the Hamiltonian and relaxation parameters are known. However, in practise, these techniques can only be used to model small spin systems due to the appearance of the Kronecker products when building operators for multi-spin systems. The resulting matrices are at least $2^n \times 2^n$ in size if the simulation is done in Hilbert space and $2^{2n} \times 2^{2n}$ in size in Liouville space (and they will be bigger still if spins greater than one half are involved) meaning that the problem is exponential in both time and memory and cannot be solved in any reasonable amount of time for large spin systems.

Recently, a new method of achieving polynomial scaling, the state space restriction technique, was published [6] that promised to cut the problem of exponential scaling to polynomial scaling. Conceptually, state space restriction is a very straightforward technique, although difficulties lie in the details. For pedagogical purposes, this review will begin with the simplest picture of state space restriction, one cast purely in the language of linear algebra, rather than trying to track the historical development of this still young subject.

These ideas have been implemented in the Spinach library introduced here [22].

1.4.1 The Linear Algebra Picture

The time evolution of a spin system is governed by the master equation:

$$i\hbar\frac{d\hat{\rho}}{dt} = \hat{\hat{H}}\rho + \hat{\hat{R}}(\hat{\rho} - \hat{\rho}_{\text{eq}}) \tag{1.46}$$

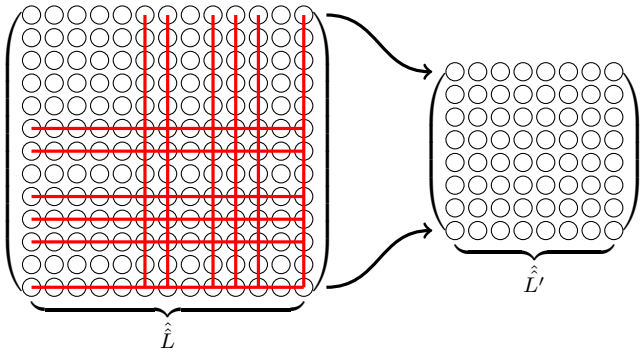


Figure 1.1 – Conceptually, state space restriction in Liouville space simply removes the rows and columns of the full Liouvillian, $\hat{\hat{L}}$, corresponding to the set of states that are to be pruned. The new, smaller, Liouvillian, $\hat{\hat{L}}'$, remains Hermitian and thus preserves the norm of the reduced density matrix. In practise the $\hat{\hat{L}}$ is rarely ever built explicitly, because its size is exponential in the number of spins present. Instead, $\hat{\hat{L}}'$ is constructed directly.

which was defined in section 1.3. For the moment, we will assume these operators have been constructed in memory. Suppose we knew in advance that there were certain states in ρ that would remain zero, or very close to zero, for the entire propagation of the system forward in time, we would be able to remove them from the system without changing the result. There does not appear to be a straightforward method of doing this in Hilbert space (this problem is discussed in more detail in the original paper [6]) but this is very simple to do in Liouville space (see figure 1.1). Due to the importance of these assertions, the definition of state space restriction will now be made formally.

Definition 1.5. *Given a space of superoperators, $\mathcal{L}$, written in basis set $\mathcal{B}$ and a subset of the basis set to restrict over, $\mathcal{B}_R \subset \mathcal{B}$, then a state space restriction is a map from $\mathcal{L}$ to the restricted space, $\mathcal{L}_R$, that for each $b_i \in \mathcal{B}_R$ zeros the i^{th} row and i^{th} column of each superoperator in $\mathcal{L}$ (see figure 1.1).*

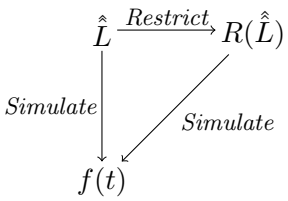
This definition could have been made equally well in terms of vector space quotients; the choice of emphasising the basis set was deliberate as, in practise, choosing a good basis set if vital for the success of the state space restriction technique.

Definition 1.6. *A "good" state space restriction, R, with respect to an operator, $\hat{\hat{L}}$, and initial state $\hat{\rho}_0$, is a restriction scheme such that for every possible operator, $\hat{O}$:*[5]

$$\langle \exp(-i\hat{\hat{L}}t/\hbar)\hat{\rho}_0, \hat{O}\rangle = \langle \exp(-iR(\hat{\hat{L}})t/\hbar)\hat{\rho}_0, \hat{O}\rangle, \tag{1.47}$$

for all $t \in \mathbb{R}$. In other words, the path taken through the following diagram does not affect the result (i.e. the diagram commutes):

[5]The inner product implied here is that defined in section 1.2.7



where $f(t)$ is the value of the observable over time.

The following theorem establishes the conditions for validity of state space restriction:

Theorem 1.7. *If $\exp(-i\hat{\hat{L}}t/\hbar)\rho_0 \cdot b_i = 0$ for all $t \in \mathbb{R}$ and for all $b_i \in \mathcal{B}_R$ then $\mathcal{B}_R$ is a good state space restriction.*

Proof. The vertical row that is removed shifts coherence from other parts of the density matrix into the removed state, but since that state remains zero, that removed row had no effect. The removed column shifts coherence from the removed state to the rest of the density matrix, but as the removed state is always zero, this column has no effect. Finally, observables are measured by taking the dot product of an operator, but because the removed state was zero, the dot product is unaffected. □

Theorem 1.8. *$\mathcal{B}_R$ is a good state space restriction with respect to operator $\hat{\hat{L}}$ and initial state $\hat{\rho}_0$ with $\hat{\hat{L}} \neq 0$ if and only if $\langle \hat{\hat{L}}^n \hat{\rho}_0, \hat{b}_i \rangle = 0$ for all $\hat{b}_i \in \mathcal{B}_R$ and for all $n \in \mathbb{N}$.*

Proof. From the definition of the matrix exponential, the expression $\exp(-i\hat{\hat{L}}t/\hbar)\rho_0 \cdot b_i$ becomes $\sum_{n=0}^{\infty} \frac{(-it)^n \hat{\hat{L}}^n \hat{\rho}_0}{n!\hbar^n} \cdot \hat{b}_i = 0$. As each $\hat{\hat{L}}^n \hat{\rho}_0 \cdot \hat{b}_i$ is a unique vector, the only way to satisfy the definition for all t is if each $\hat{\hat{L}}^n \cdot \hat{b}_i$ is the zero vector individually. The proof in the reverse direction is trivial. □

Theorem 1.9. *If $\mathcal{B}_i$ is a good state space restriction for operator $\hat{\hat{A}}$ then it is a good state space restriction for $\alpha\hat{\hat{A}}$ with $\alpha \in \mathbb{R}$.*

Proof. Apply theorem 1.8 and use the linearity of the inner product. □

Theorem 1.9 shows that knowledge of a good restriction scheme for one system implies good restriction schemes can automatically be found for other systems in the limited case when those systems are directly proportional. If a restriction scheme is known that works for two operators $\hat{\hat{L}}_1$ and $\hat{\hat{L}}_2$ then it is not, in general, the case that the scheme is automatically good for $\hat{\hat{L}}_1 + \hat{\hat{L}}_2$ unless $\hat{\hat{L}}_1$ and $\hat{\hat{L}}_2$ commute. Consequently, there is no obvious method of building a working restriction scheme from restriction schemes tailored to subsystems.[6]

[6]These arguments do not rule out such a method existing.

Data: t_1, t_2, N, M, $\hat{\hat{L}}_{\text{free}}$, $\hat{\hat{L}}_{\text{pulse}}$, $\hat{\rho}_{\text{initial}}$
Result: $f_{i,j}$, a two-dimensional array of real numbers representing the FID.
$\Delta t_1 \leftarrow \frac{t_1}{N}$; $\Delta t_2 \leftarrow \frac{t_2}{M}$;
$\hat{\hat{P}}_1 \leftarrow \exp(-i\hat{\hat{L}}_{\text{free}}\Delta t_1/\hbar)$; $\hat{\hat{P}}_2 \leftarrow \exp(-i\hat{\hat{L}}_{\text{free}}\Delta t_2/\hbar)$;
$\hat{\hat{P}}_{\text{pulse}} \leftarrow \exp(-i\hat{\hat{L}}_{\text{pulse}}\Delta t_{\text{pulse}}/\hbar)$;
$\hat{\sigma} \leftarrow \hat{\hat{P}}_{\text{pulse}}\hat{\rho}_{\text{initial}}$;
for i *in* $1 \ldots N$ **do**
  $\hat{\rho} \leftarrow \hat{\hat{P}}_{\text{pulse}}\hat{\sigma}$;
  for j *in* $1 \ldots M-1$ **do**
    $f_{i,j} = \text{Tr}(\hat{\rho}^{\text{T}}\hat{S}_+)$;
    $\hat{\rho} \leftarrow \hat{\hat{P}}_2\hat{\rho}$;
  end
  if $i \neq N$ **then**
    $\hat{\sigma} \leftarrow \hat{\hat{P}}_1\hat{\sigma}$;
  end
end

Algorithm 1: Pseudocode algorithm for the simulation a COSY experiment. The free variables t_1 and t_2 are the maximum direct and indirect propagation times respectively, N and M are the number of points in the direct and indirect dimension respectively, $\hat{\hat{L}}_{\text{free}}$ is the Liouvillian when the radio frequency field is off, $\hat{\hat{L}}_{\text{pulse}}$ is the Liouvillian during the pulse. Notice that no explicit indexing of the matrices involved is required—in this, COSY is typical of many spin dynamics experiments. As a consequence, $\hat{\hat{L}}_{\text{free}}$, $\hat{\hat{L}}_{\text{pulse}}$, $\hat{S}_+$, and $\hat{\rho}_{\text{initial}}$ may be replaced with state space restricted versions without modification of the algorithm.

The simplest type of state space restriction, zero track elimination (introduced in the second paper on state space restriction [28]), works essentially as suggested by figure 1.1. The system is propagated forward for a few time steps, and states that remain zero (the "zero tracks") are assumed[7] to be states that will continue to remain zero. These states are then eliminated from the operators in the simulation and propagation continues using the smaller operators.

Propagation with the reduced operators and density matrix is obviously faster as they are smaller in size (all other things being equal, such as the choice between Hilbert and Liouvillian space or sparse or dense matrix representation). In fact, it will turn out (chapter 2) that even though the total number of states increases exponentially with the size of the system, the total number of states that matter only increases polynomially, making the propagation step cost polynomial.

This picture completely characterises how state space restriction works at a high level. The situation is very rosy from a user point of view as each of these reduced operators may

[7] A justification for this assumption is given in the paper.

be used in place of the full operators in most simulation code written in terms of high level matrix operations. For example, the COSY experiment requires three Liouvillians: the free evolution Liouvillian (used twice) and the $\frac{\pi}{2}$ pulse Liouvillian (see algorithm 1). As no explicit indexing was required, code based on (1) would not need to be changed to use restricted state space Liouvillians.

Unfortunately two critical details need to be addressed:

- How can the unneeded states be identified in advance?
- Constructing the reduced operators from the full operators is completely infeasible as the full operators take exponential time and storage space. Even writing down a list of states to eliminate to achieve polynomially scaling propagation would take exponential time. A method of constructing the reduced operators without reference to the full operators needs to be found.

Kuprov et al. provided a technique for reduced operator construction in the original paper [6]. A second technique was proposed by *Hogben et al.* in [29] that is currently used in Spinach. It will be discussed in section 1.5.8.

An idea that will be used a great deal with respect to state space restriction, especially in chapter 2, is that of spin orders.

Definition 1.10. *Each basis state in the product basis is assigned a specific spin order equal to the number of non-identity operators in the product.*

Multiple techniques for identifying unneeded basis vectors have been proposed and are implemented in Spinach. They are:

Spin Order Pruning States of a spin order greater than k are pruned. The resulting dimension of the new state space under this scheme is $4^k\binom{n}{k}$. This scheme was proposed in Kuprov's original paper justified on the grounds that states of higher coherence order relax faster than lower order states.

Interaction Topography, or "clusterization", was also proposed in Kuprov's original paper, and was based on removing states describing coherences between spins that only interact via many intermediate spins.

Zero Track Elimination States that are observed to remain zero or close to zero for the first few propagation steps are assumed to remain zero the for rest of the simulation.

Invariant Subspace Techniques Careful analysis of the Liouvillian [29] may reveal that it can be block diagonalised, meaning that the density matrix may be decomposed into invariant subspaces. Coherence that begins in an invariant subspace remains in that subspace while the system evolves under the Liouvillian. The currently available techniques for identifying the invariant subspaces are:

1. **Symmetry** States within a different irreducible representation of some symmetry group live in different invariant subspaces [30].
2. **Path Tracing** Let each state in the density matrix be a node in a graph and let each nonzero term in the Liouvillian connecting to states be an edge then it may be the case that this graph decomposes into separate sub-graphs in which case, the nodes (states) in each sub-graph form distinct invariant subspaces.

Once the structure of the invariant subspaces is known, if a particular subspace does not start populated it can never become populated and it may be pruned from the simulation. Invariant spaces that are inherently unobservable may also be pruned under the last Liouvillian in a particular system [31].

Of these, the invariant subspace techniques are mathematically exact and are beyond questioning, while, unfortunately, zero track elimination, interaction topography and spin order pruning are inherently approximate rather than exact techniques.

Justification for zero track elimination was given in the paper it was proposed in [28], no formal justification of interaction topography is yet known (although it has strong intuitive appeal). Formal justification of spin order pruning is particularly important to establish as this technique eliminates all but a set of states whose count is polynomial in the number of spins, so establishing spin dynamics as a problem which scales, at least for practical purposes, polynomially and not exponentially in the number of spins. The formal treatment of spin order pruning will be the subject of chapter 2 but it can be motivated by arguments made in the following section.

1.4.2 Spin Order Pruning

Spin order pruning can be motivated by the expansion of the solution to the Liouville-von Neumann equation, ignoring relaxation:

$$\exp(-it\hat{\hat{H}}/\hbar)\hat{\rho}_0 = \hat{\rho}_0 - \frac{i}{\hbar}t\hat{\hat{H}}\hat{\rho}_0 - \frac{t^2}{2!\hbar^2}\hat{\hat{H}}^2\hat{\rho}_0 + \frac{it^3}{3!\hbar^3}\hat{\hat{H}}^3\hat{\rho}_0 + \ldots \tag{1.48}$$

In the majority of spin dynamics problems, the system starts with $\hat{\rho}_0$ having nonzero elements only in the first spin order. A dense $\hat{\hat{H}}$ would imply that the vector $\hat{\hat{H}}\hat{\rho}_0$ might have nonzero

elements anywhere but, assuming a Hamiltonian of the form given in equation (1.6), $\hat{\hat{H}}\hat{\rho}_0$ only has nonzero elements in the first and second spin order subspaces.

In fact, if some vector ρ has nonzero elements only in spin order k then $\hat{\hat{H}}\hat{\rho}$ has, in general, nonzero elements in spin orders $k-1$, k and $k+1$ only so long as the Hamiltonian is of the form given in equation (1.6). Proof of this claim is deferred until section 2.3 in chapter 2.

As a consequence, we see that the population of spin order k at short times grows in proportion to t^{k-1}, and so we see that for short times, higher spin orders can be neglected. This reasoning seems to imply that spin order pruning can only be regarded as an approximation that only works on short timescales but, as will be seen in chapter 2, when relaxation is considered, spin order pruning will be shown to be applicable at all reasonable time scales so long as the pruning cutoff is suitably chosen.

1.4.3 Basis Set

Because state space restriction requires physically deleting rows and columns of the Liouvillian, the success of the procedure depends on the basis set chosen. In the Spinach library the *irreducible spherical tensor* basis set is employed. Irreducible spherical tensors have the advantage of being eigenstates of the Zeeman interaction, which usually dominates.

One way to define the irreducible spherical tensors is in terms of their commutation relations with the angular momentum operators $\hat{J}_z$ and $\hat{J}_\pm$ (see *Sakurai* [32]). Let j be an integer or half integer where $j \geq 0$. If a set of $(2j+1)$ tensors for tensors indexed by index m running from, $-j$ to j in steps of 1 obey the same commutation relations as the $\hat{T}^j_m$ tensors in the following:

$$\begin{aligned}[\hat{J}_z, \hat{T}^j_m] &= m\hat{T}^j_m \\ [\hat{J}_\pm, \hat{T}^j_m] &= \sqrt{(j \mp m)(j \pm m + 1)}T^j_{m\pm 1},\end{aligned} \tag{1.49}$$

they are the irreducible spherical tensors for spin j.

Although written in terms of matrices and commutators, this same definition cast in Liouville space looks like:

$$\begin{aligned}\hat{\hat{J}}_z\hat{T}^j_m &= m\hat{T}^j_m \\ \hat{\hat{J}}_\pm\hat{T}^j_m &= \sqrt{(j \mp m)(j \pm m + 1)}\hat{T}^j_{m\pm 1},\end{aligned} \tag{1.50}$$

so the irreducible spherical tensors are seen to be essentially Liouville space equivalents of state vectors written in the usual j, m_z basis.

1.5 Algebras and Spin Dynamics

Although the theoretical description of spin dynamics and state space restriction may be viewed purely in terms of linear algebra, in doing so much of the structure of the problem is lost. For example, using linear algebra, the Hamiltonian of an n spin system corresponds to a $2^n \times 2^n$ Hermitian matrix, suggesting that it would require $2^n \times 2^n$ real parameters to specify, yet from equation (1.6) it can be seen that many fewer ($O(n^2)$) real parameters are required. It is therefore enlightening to go beyond the linear algebra picture of spin dynamics and study the problem in terms of abstract algebra and, in particular, algebras over a field.

Definition 1.11. *An algebra, $\mathcal{A}$, over the field $\mathbb{K}$ is a vector space over the field $\mathbb{K}$ equipped with an additional bilinear product, $\diamond$, which fulfils the following requirements:*

- ***Closure*** $a_1 \diamond a_2 \in \mathcal{A}$
- ***Bilinearity*** $(\alpha a_1 + \beta a_2) \diamond a_3 = \alpha(a_1 \diamond a_3) + \beta(a_2 \diamond a_3)$ *and* $a_1 \diamond (\alpha a_2 + \beta a_3) = \alpha(a_1 \diamond a_2) + \beta(a_1 \diamond a_3)$

With $a_1, a_2, a_3 \in \mathcal{A}$, $\alpha, \beta \in \mathbb{K}$ and $\mathcal{A}$ being the algebra in question.

The field that an algebra is defined over is usually called the "ground" field and the field determines the type of the algebra. For example, "real algebra" implies the ground field is $\mathbb{R}$.

An idea that will prove useful is the algebra "generated" by a set of elements.

Definition 1.12. *Let $\mathcal{A}_0 = a_i$ be a set of vectors and let $\diamond$ be a bilinear product over these vectors. Let $\mathcal{A}_n$ be the span of the space spanned by the application of $\diamond$ to all possible pairs in $\mathcal{A}_{n-1}$, then the generated algebra is* $\mathrm{span}\{\mathcal{A}_0 \cup \mathcal{A}_1 \cup \dots\}$.

In spin dynamics the three most important algebras are the product algebra, the tensor algebra and the commutator algebra of the quantum mechanical operators. Together, these three algebras model many of the calculations in spin dynamics that would otherwise need to be performed with explicit, and potentially expensive, matrix operations. These ideas are used in section 1.5.8 and again in chapter 2.

1.5.1 The Tensor Algebra

In section 1.2.3, operators representing systems of many spins were constructed via the Kronecker Product on matrices. From a computation point of view, this product is unfavourable as the matrix size of a series of n matrices combined this way grows exponentially in n.

The tensor algebra of a vector space provides a model Kronecker product.

Definition 1.13. *The tensor algebra over a vector space V is the algebra generated from the elements of V by taking the bilinear product, $\otimes$, to be such that $(\boldsymbol{v}_1 \otimes \boldsymbol{v}_2) \otimes \boldsymbol{v}_3 = \boldsymbol{v}_1 \otimes (\boldsymbol{v}_2 \otimes \boldsymbol{v}_3)$ for all $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$ in V.*

Examples of elements of the tensor algebra over a real vector space with basis $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{e}_3$ are 4, $\boldsymbol{e}_1$, $\boldsymbol{e}_1 \otimes \boldsymbol{e}_3$ and $2 + \boldsymbol{e}_2 + \boldsymbol{e}_1 \otimes \boldsymbol{e}_3$.

The tensor product has the exact same properties as the Kronecker product [33] and so can be used to model it without the overhead of using explicit matrices in computer memory. This idea is exploited in Spinach [22, 29]. Since a simulation has a fixed number of spins, only elements of the tensor algebra with the same number of basis elements from the initial vector space as the number of spins in the system are used in any particular simulation. These elements form a subspace, not an algebra, with respect to the tensor product.

1.5.2 The Product Algebra

The product algebra is simply the usual linear operator composition product that corresponds to matrix-matrix multiplication in linear algebra. This product has the following noteworthy properties:

- The order in which the tensor and composition products are applied doesn't matter, i.e. $(A \otimes B)(C \otimes D) = AC \otimes BD$.
- The composition product is spin-dependent. For example, $\hat{J}_+^2$ is zero for a spin half system but nonzero for higher spins.
- The composition product of two Hermitian operators is Hermitian only if they commute [10, p. 50].
- The composition product determines the commutator product via $[A, B] = AB - BA$.

Remarkably, even though the commutator product depends on the composition product (which is spin dependent), for cases that do not involve quadratic interactions the resulting commutator algebras are identical.

1.5.3 The Lie Algebra

The commutator algebra has the additional property of being a Lie algebra. Lie algebras were invented by Sophus Lie in the 19$^{\text{th}}$ century as a tool for studying infinitely differentiable groups in terms of their behaviour close to the identity and were first applied to quantum mechanics by Hermann Weyl (historical details can be found in [30]). The link between Lie

Algebras and spin dynamics (and indeed much of modern quantum physics) comes through the solution of the wave equation:

$$|\phi\rangle = \exp(-i\hat{H}t/\hbar)\,|\phi_0\rangle \tag{1.51}$$

This is also the solution to the Liouville-von Neumann equation with $\hat{H}$ replaced by the Liouvillian and $|\phi_0\rangle$ replaced by the density matrix, and so the differences between Liouville space solutions and Hilbert space solutions will be largely immaterial.

If we take the linear algebra picture, $\hat{H}$, $\exp(-i\hat{H}t/\hbar)$ are matrices while $|\phi_0\rangle$ is a vector. To solve equation (1.51) we are limited to throwing these matrices and vectors at a computer and using numerical techniques. In contrast, the use of Lie Algebra techniques reveal more about the structure of the problem, possibly permitting the identification of invariant subspaces and, consequently, allowing the problem to be block diagonalised long before any explicit matrices are constructed. Furthermore, from a computation perspective, many calculations at the Lie Algebra level may be performed via exact integer arithmetic, sidestepping the numerical uncertainty that is inherent whenever floating point numbers are used.

When treating equation (1.51) in terms of Lie Groups and Lie Algebras, the quantity $-iHt/\hbar$ is regarded as a member of a Lie Algebra, $\exp(-iHt/\hbar)$ as a member of a Lie Group and $|\phi_0\rangle$ is a vector acted on via a representation of the Lie Group. The definitions of "Lie Group", "Lie Group representation" and "Lie Algebra representation" are fairly technical and may be found in appendix A, but the definition of a Lie Algebra will be given now:

Definition 1.14. *A Lie Algebra is an algebra,* $(\mathfrak{g}, [\,\cdot\,,\cdot\,])$*, where the bilinear operation obeys the following axioms:*[8]

- ***Antisymmetry****:* $[X,Y]+[Y,X]=0$
- ***Jacobi Identity****:* $[X,[Y,Z]]+[Z,[X,Y]]+[Y,[Z,X]]=0$

with $X, Y, Z \in \mathfrak{g}$.

If the Lie Algebra elements (X, Y, Z, *etc.*) are represented as matrices then the Lie Bracket is simply the matrix commutator as suggested by the notation.

Lie Group theory is a large and complex branch of mathematics in its own right that requires background knowledge of both differential geometry and topology and so no attempt at a full treatment will be given here (see [14, 34]). The advantage of using Lie Algebras is many of the corresponding properties of Lie Groups may be inferred from the Lie Algebra and, yet, Lie Algebras are much simpler to work with. The link between the Lie Algebra

[8] Closure and bilinearity are inherited from the definition as an algebra

member and its corresponding Lie Group member is provided by the exponential map, which will be the topic of the next section.

1.5.4 The Exponential Map

Definition 1.15. *The exponential map,* exp*, on a Lie Algebra carries members of that Lie Algebra to a corresponding Lie Group according to:*

$$\exp(A) = \sum_{n=0}^{\infty} \frac{A^n}{n!}. \tag{1.52}$$

The inverse map, log*, if it exists, is defined as:*

$$\log(A) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}(A-1)^n}{n}. \tag{1.53}$$

An important consideration is whether the exp and log maps converge. As we are interested in using Lie algebras in relation to spin dynamics, only matrix Lie groups will be considered here. The exponential map can be seen to always converge as follows [35]:

$$\begin{aligned} ||\exp(A)|| &= \left|\left|\textstyle\sum_{n=0}^{\infty} \frac{A^n}{n!}\right|\right| && \text{Definition of the matrix exponential} \\ &\leq \textstyle\sum_{n=0}^{\infty} \frac{||A^n||}{n!} && ||AB|| \leq ||A||||B|| \\ &\leq \textstyle\sum_{n=0}^{\infty} \frac{||A||^n}{n!} && ||A+B|| \leq ||A|| + ||B|| \text{ Triangle Inequality} \\ &= \exp(||A||). && \text{Definition of the exponential} \end{aligned} \tag{1.54}$$

And, of course, $\exp(x)$ is known to converge for any $x \in \mathbb{R}$.

The same is not true of the log power series, as the infinite sum $\sum_{n=1}^{\infty} \frac{1}{n}||A-1||^k$ only converges when $||A-1|| < 1$. Consequently $\log(\exp(A))$ can only be regarded as a strict inverse when $||\exp(A) - 1|| < 1$; in general the inverse of exp is multivalued. This is not surprising. If θ represents an angle of rotation then $\theta + 2n\pi$ for any integer n represents the same rotation and a map back from the rotation matrix to the rotation angle has to be multivalued.

As an example of the exponential map, take:

$$\exp\left[-i\theta(\alpha\hat{J}_x + \beta\hat{J}_y + \gamma\hat{J}_z)\right] = \begin{pmatrix} \cos(\frac{1}{2}\theta) - i\gamma\sin(\frac{1}{2}\theta) & (-\beta - i\alpha)\sin(\frac{1}{2}\theta) \\ (+\beta - i\alpha)\sin(\frac{1}{2}\theta) & \cos(\frac{1}{2}\theta) + i\gamma\sin(\frac{1}{2}\theta) \end{pmatrix}, \tag{1.55}$$

where α, β and γ have been scaled so that $\alpha^2 + \beta^2 + \gamma^2 = 1$ and the magnitude has been absorbed by θ. The unitary matrix on the right hand side can be interpreted directly as member of $SU(2)$, or as a representation of $SO(3)$ by interpreting matrices that differ by 2π in their θ values as the same rotation. When dealing with spin (section 1.2.2) physics tells us to do the former, while when dealing with 3D rotations physics tells us to do the latter.[9]

[9] By noting that $i\sigma_x$, $i\sigma_y$, $i\sigma_z$ have the same multiplication algebra as the quaternions (section 1.2.2) it can be seen that equation (1.55) is a disguised version of the angle-axis to quaternion conversion formula that will appear in section 3.3.3.

1.5.5 Lie Theory and Spin Dynamics

Lie Theory touches on spin dynamics in two main places.

1. Invariant subspaces. The representation theory of Lie Algebras provides methods for identifying invariant subspaces by examining the commutation relations of the operators that make up the Hamiltonian.

2. Commutator structure. The structure of the commutators of the operators that make up the Hamiltonian can be used as a guide for determining which basis elements are important and which may be culled.

Traditionally, Lie Theory is concerned with the classification of Lie Groups, their algebras, and their representations. Indeed, the theory of the addition of angular momentum may be recast in terms of the problem of finding the invariant subspaces of product algebras [30].

As the classification of the representations of the semisimple Lie Algebras is completely solved (see Fulton and Harris' book [14] in which a good proportion of the page count is dedicated to establishing this) so it would seem that the classification theorem could be used to find all invariant subspaces from the commutator algebra alone (accidental cases arising from the particular choices of the real coefficients would be missed however) as a supplement to the current method by which Spinach identifies invariant subspaces: path tracing [29], which is basis-dependent. However, further speculation on this topic is outside of the scope of this work.

The second case, the commutator structure, will prove very relevant in chapter 2 as, according to equation (1.48), each additional commutator with the Hamiltonian required to reach a particular state raises the power of the time coefficient by one so the less likely that state is to be required for an accurate simulation. Thankfully, the study of the commutator structure is simpler and more accessible and doesn't require many of the traditional results from Lie Algebra.

1.5.6 General n Level Quantum System

The general form for a quantum Hamiltonian of an n level system, as required by the basic postulates of quantum mechanics, is an $n \times n$ Hermitian operator, which after multiplication by time and the imaginary unit and subsequent exponentiation, results in a $n \times n$ unitary operator of determinant 1, in other words, a member of $SU(n)$.

$SU(n)$ therefore physically corresponds to the set of all ways an n level quantum mechanical system may evolve from some initial time to some final time [10, p. 104]. The Hamiltonian, H, which represents the differential of the evolution of the system, is Hermitian, so iH is antihermitian and thus member of $\mathfrak{su}(n)$.

Definition 1.16. $\mathfrak{su}(n)$ *Lie Algebra:*

$$\mathfrak{su}(n) = \{X \in M_n(\mathbb{C}) : X^\dagger = -X, \mathrm{Tr}(X) = 0\}. \tag{1.56}$$

With $M_n(\mathbb{C})$ being the set of $n \times n$ matrices over the complex numbers. The Lie bracket is the matrix commutator.

The exponential of an antihermitian matrix is a unitary matrix, thus every member of $\mathfrak{su}(n)$ naturally corresponds to at least one member of $SU(n)$.

The smallest non-trivial irreducible representation $SU(n)$ is of dimension n [14, 30]. Consequently, if all that is known is that a system's Hamiltonian is a general Hamiltonian, there is little that the theory of Lie Algebras can say.

However, it is usual to have additional information, beyond the fact that the Hamiltonian is Hermitian. For example, the spin Hamiltonian for a single particle that experiences only linear interactions must be of the form:

$$H = \alpha J_z + \beta J_y + \gamma J_z \quad \alpha, \beta, \gamma \in \mathbb{R}. \tag{1.57}$$

Critically, the commutation relations between the operators $\{J_z, J_y, J_z\}$ do not depend on the magnitude of the spin in question and so we can examine them as a Lie algebra, in this case $\mathfrak{su}(2)$, without reference to the magnitude of the spin (or in other words, the size of the Lie Algebra representation).

Unfortunately, as will be seen in the next section, the situation gets more complicated when nuclear quadrupolar interactions are considered because the commutators change with the magnitude of the spin.

1.5.6.1 Real and Complex Lie Algebras

As was noted in section 1.5, the type of the ground field determines the type of the algebra. Lie Algebras appearing in physics are real because the Hamiltonian must be Hermitian. Hamiltonians are generally written as a functions of real parameters that correspond to physical quantities. In other words, they have the form:

$$\hat{H} = \sum_i \alpha_i \hat{A}_i, \tag{1.58}$$

with $\hat{A}_i$ being Hermitian operators and α_i corresponding to adjustable physical parameters that describe the system, such as the magnetic field strength or coupling constants.

Unfortunately, the real nature of these algebras are often obscured by the notation, as the imaginary unit will always appear in commutators. The reason for this is that the mathematical object that is exponentiated to produce members of the Lie Group is iH and

so the actual $\mathfrak{su}(n)$ Lie Algebra members from equation (1.58) are $i\hat{A}_i$. The relevant Lie Algebra is generated from the terms in:

$$i\hat{H} = \sum_j \alpha_j i\hat{A}_j, \tag{1.59}$$

and is thus a real Lie Algebra.

Two examples where the real nature of the Lie Algebra is obscured will be given. The first is the Lie Algebra formed from the commutation relations between the three angular momentum operators:

$$[\hat{J}_i, \hat{J}_j] = i\hbar\epsilon_{ijk}\hat{J}_k, \tag{1.60}$$

which appears to be a complex algebra due to the appearance of the imaginary unit in a scalar product.

This can be corrected by multiplying each operator by i to get:

$$[\hat{X}_i, \hat{X}_j] = -\epsilon_{ijk}\hbar\hat{X}_k \quad \text{where } \hat{X}_j = i\hat{J}_j, \tag{1.61}$$

the new set of operators is antihermitian and have imaginary eigenvalues. The two sets are still isomorphic and so both belong to the same Lie Algebra, but one obscures some of the mathematical concepts while the other obscures the physical concepts.

A second example of a real Lie Algebra disguised to look complex is found in the full quadrupolar interaction Hamiltonian as stated by Levitt [7]:

$$\frac{3eQ}{4I(2I-1)}\left[V_0(3\hat{I}_z^2 - \hat{\boldsymbol{I}}\cdot\hat{\boldsymbol{I}}) + V_1(\hat{I}_-\hat{I}_z + \hat{I}_z\hat{I}_-) + V_{-1}(\hat{I}_+\hat{I}_z + \hat{I}_z\hat{I}_+) + V_{+2}\hat{I}_-^2 + V_{-2}\hat{I}_+^2\right], \tag{1.62}$$

where $\{V_0, V_\pm, V_{2\pm}\}$ is a set of potentially complex numbers that depend on the second derivatives of the electric field. Once again, this Hamiltonian, viewed as an element of a Lie Algebra, appears to belong to a complex one as the coefficients are complex and the operators not Hermitian. But after making the following replacements:

$$\begin{aligned} V_0 &\to V_{zz} & \hat{I}_-\hat{I}_z + \hat{I}_+\hat{I}_z + \hat{I}_z\hat{I}_- + \hat{I}_z\hat{I}_+ &\to \hat{V}_{xz} \\ V_{\pm1} &\to V_{zx} \pm iV_{zy} & i(\hat{I}_+\hat{I}_z + \hat{I}_z\hat{I}_+ - \hat{I}_-\hat{I}_z - \hat{I}_z\hat{I}_-) &\to \hat{V}_{yz} \\ V_{\pm2} &\to \frac{1}{2}(V_{xx} - V_{yy}) \pm iV_{xy} & \hat{I}_-\hat{I}_- + \hat{I}_+\hat{I}_+ &\to \hat{V}_{2a} \\ & & i(\hat{I}_-\hat{I}_- - \hat{I}_+\hat{I}_+) &\to \hat{V}_{2b}. \end{aligned} \tag{1.63}$$

The operators $\hat{V}_{xz}$, $\hat{V}_{yz}$, $\hat{V}_{2a}$ and $\hat{V}_{2b}$ have become Hermitian and the quantities V_{zz}, V_{zx}, V_{zy}, $(V_{xx} - V_{yy})$ are real (by their definition as derivatives of the electric field). The new Hamiltonian is:

$$\frac{3eQ}{4I(2I-1)}\left[V_{zz}(3\hat{I}_z^2 - \hat{\boldsymbol{I}}\cdot\hat{\boldsymbol{I}}) + V_{xz}\hat{V}_{xz} + V_{yz}\hat{V}_{yz} + (V_{xx} + V_{yy})\hat{V}_{2a} + 2V_{xy}\hat{V}_{2b}\right], \tag{1.64}$$

which now looks like a member of a real Lie Algebra. There five operators, taken together with $\hat{I}_x$, $\hat{I}_y$ and $\hat{I}_z$, which may potentially appear elsewhere in the Hamiltonian, describe an algebra with at least eight basis vectors.

Unlike the case of $\hat{J}_x$, $\hat{J}_y$ and $\hat{J}_z$, where the commutators do not change when the magnitude of the spin changes, the commutators of the nuclear quadrupolar operators do, implying that the Lie Algebra depends on the magnitude of the spin. It can be shown (use of a computer algebra package is highly recommended for doing so), for example, that the spin 1 case corresponds to $\mathfrak{su}(3)$.

1.5.7 State Space Restriction in terms of Lie Algebras

In section 1.4.1 it was noted that state space restriction, understood at its simplest level, corresponds to deleting rows and columns from the Liouvillian. State space restriction can also be understood in terms of commutators. It has been established above that Hamiltonians belong to a real Lie Algebras. However, this treatment was basis-independent, whereas it is known that state space restriction is a basis-dependent technique so a new approach is needed.

Let $\{\hat{O}_i\}$ be the basis set the density matrix is written in (this will be tensor products of irreducible spherical tensors in the case of Spinach). The commutator algebra generated by this set will be the Lie Algebra in which we will be interested. This algebra may be of higher dimension than the "minimal" algebra, for example, the basis algebra of a spin 1 particle is nine-dimensional but the Hamiltonian can be written in terms of the three real coefficients of $\hat{J}_x$, $\hat{J}_y$ and $\hat{J}_z$ and, so, the minimal algebra is three dimensional.

The structure constants of general Lie Algebra are given by:

$$[\hat{O}_i, \hat{O}_j] = c_{ijk}\hat{O}_k. \tag{1.65}$$

This definition can be rearranged by taking the inner product (in the sense defined in section 1.2.7) with a basis element to give the equivalent definition:

$$[\hat{O}_i, \hat{O}_j] \cdot \hat{O}_k = c_{ijk}. \tag{1.66}$$

The structure constants are fundamentally related to the adjoint representation of a Lie Algebra. The adjoint action of an operator $\hat{H}$ written out as an explicit matrix can be seen to be:

$$[\hat{H}, \cdot\,] \rightarrow \begin{pmatrix} [\hat{H}, \hat{O}_1] \cdot O_1 & [\hat{H}, \hat{O}_2] \cdot \hat{O}_1 & \ldots & [\hat{H}, \hat{O}_n] \cdot \hat{O}_1 \\ [\hat{H}, \hat{O}_1] \cdot O_2 & [\hat{H}, \hat{O}_2] \cdot \hat{O}_2 & \ldots & [\hat{H}, \hat{O}_n] \cdot \hat{O}_2 \\ \vdots & \vdots & \ddots & \vdots \\ [\hat{H}, \hat{O}_1] \cdot O_n & [\hat{H}, \hat{O}_2] \cdot \hat{O}_n & \ldots & [\hat{H}, \hat{O}_n] \cdot \hat{O}_n \end{pmatrix}. \tag{1.67}$$

Remembering that $\hat{H} = \sum_i \alpha_i \hat{O}_i$ with $\{\alpha_i\}$ being a set of arbitrary real numbers, we can see that, deleting basis element m from the state space for all valid Hamiltonians is equivalent to setting the constants c_{ijk} to zero when any one of i, j or k is m. In short, state space restriction can also be thought of as the process of removing certain commutators from the Lie Algebra generated by the commutators of the basis set.

1.5.8 Construction of Reduced Basis Operators via Left and Right Multiplication Superoperators

While, conceptually, state space restriction can be understood in terms of deleting rows and columns from the full Liouvillian and density matrix, in practise, the full Liouvillian is too large to construct explicitly. Without a method of operator construction, state space restriction would fail to achieve the promised reduction of problem complexity from exponential to polynomial so it is worth reviewing the current operator construction technique used in Spinach. This method was first published by *Hogben et al.* in [29].

Suppose that an operator $\hat{Q}$ is required to be constructed. It will be assumed that this operator may be written in the form $\sum_i^N q_i \hat{O}_i$, where N is the size of the basis set and $\{O_i\}$ is the set of basis operators. It will be assumed that these basis operators may be written in the form:

$$\underbrace{\hat{T}_{b(1)} \otimes \cdots \otimes \hat{T}_{b(j)} \otimes \cdots \otimes \hat{T}_{b(N)}}_{N \text{ terms}}, \tag{1.68}$$

where N is the number of spins and $b(i)$ gives the l and m values of irreducible spherical tensor i or E. If all $\prod_i^N (2J_i+2)$ possible $b(i)$ are present in the basis set construction then the state space is complete. Unlike the full matrix representation of $\hat{T}_{b(1)} \otimes \cdots \otimes \hat{T}_{b(j)} \otimes \cdots \otimes \hat{T}_{b(N)}$, which is exponential in size, $b(i)$ is simply a list of pieces of information of length equal to the number of spins and so can easily be realised in computer memory. For example, in Spinach, it is stored as a list of integers [29].

Ultimately, the operator required to be built is the commutation superoperator. The action of a commutation superoperator $\hat{\hat{Q}}$ built from a given operator $\hat{Q}$ is defined as:

$$\hat{\hat{Q}} = [\hat{Q}, \cdot\,]. \tag{1.69}$$

This map can in turn be built from the left and right multiplication superoperators, defined through their actions as:

$$\hat{\hat{Q}}_L \hat{\rho} = \hat{\rho}\hat{Q} \quad \text{and} \quad \hat{\hat{Q}}_R \hat{\rho} = \hat{Q}\hat{\rho}. \tag{1.70}$$

Note that these three actions are linear.

The operator and the density matrix can be decomposed in terms of the basis set via $\hat{Q} = \sum_i q_i \hat{O}_i$ and $\hat{\rho} = \sum_i r_i \hat{O}_i$:

$$\hat{\hat{Q}}_R \hat{\rho} = \sum_{i,j} q_j r_i \hat{O}_j \hat{O}_i \qquad \hat{\hat{Q}}_L \hat{\rho} = \sum_{i,j} r_i q_j \hat{O}_i \hat{O}_j. \tag{1.71}$$

Each individual operator is built through tensor products from identity operators or the irreducible spherical tensors.

As operator multiplication is bilinear we can write $O_i O_j$ as $\sum_k c_{ijk} O_k$. Explicitly, in terms of matrices:

$$c_{ijk} = \mathrm{Tr}(\hat{O}_i \hat{O}_j \hat{O}_k^\dagger). \tag{1.72}$$

This trace operation can been seen as giving the projection of one vector onto another by treating the product $\hat{O}_i \hat{O}_j$ and $\hat{O}_k$ as vectors. $c_{i,j,k}$ is a set of real numbers.

We will examine the $\hat{\hat{Q}}_R$ operator, focusing on the action of $\hat{\hat{Q}}_R$ on the l^{th} basis operator:

$$\hat{\hat{Q}}_R \hat{O}_l = \sum_{i,j,k} q_j c_{ijk} \hat{O}_k. \tag{1.73}$$

Taking the scalar product of both sides with $\hat{O}_m$ (not operator multiplication) brings all the operators over to the left hand side:

$$\mathrm{Tr}\left(\hat{O}_m^\dagger \hat{\hat{Q}}_R \hat{O}_l\right) = \sum_{i,j} q_j c_{ijm}. \tag{1.74}$$

We can make the meaning of the left hand side clear by making the scalar product explicit (mathematical picture) or by slightly abusing Dirac notation:

$$\mathrm{Tr}(\hat{O}_m^\dagger \hat{\hat{Q}}_R \hat{O}_l) = \langle \hat{O}_m, \hat{\hat{Q}}_R \hat{O}_l \rangle = \langle \hat{O}_m | \hat{\hat{Q}}_R | \hat{O}_l \rangle. \tag{1.75}$$

In other words, we can find the matrix elements of $\hat{\hat{Q}}_R$ from the c_{ijk} and q_j arrays. The problem of evaluating c_{ijk} can be solved as follows. Remembering that $\hat{O}$ was written $\bigotimes_{i=1}^n T_{b(i)}$ then:

$$\begin{aligned}
c_{ijk} &= \mathrm{Tr}(\hat{O}_i \hat{O}_j \hat{O}_k^\dagger) \\
&= \mathrm{Tr}[(\textstyle\bigotimes_{i=1}^n T_{b(i)})(\bigotimes_{j=1}^n T_{b(j)})(\bigotimes_{k=1}^k T_{(k)})^\dagger] && \text{Expansion into ISTs} \\
&= \mathrm{Tr}[(\textstyle\bigotimes_{n=1}^N T_{b(i)} T_{b(j)} T_{b(k)}^\dagger)] && \text{Associativity of the tensor product} \\
&= \textstyle\prod \mathrm{Tr}(T_{b(i)} T_{b(j)} T_{b(k)}^\dagger) && \text{Trace of the product is the product of the traces} \\
&= \textstyle\prod_{n=1}^N f_{ijk}.
\end{aligned} \tag{1.76}$$

Where the fact that the eigenvalues of the tensor $a \otimes b$ are $\{a_i b_j : a_i \in \mathrm{eig}(a), b_j \in \mathrm{eig}(b)\}$ was used to get from line three to line four. f_{ijk} are the structure constants of the multiplication algebra of the irreducible spherical tensors. Since these arrays are small, they can be tabulated.

Interaction	Form	Exp.	Expression
Zeeman	Linear	Both	$\frac{\mu g}{\hbar}\boldsymbol{B}\cdot\hat{\boldsymbol{S}}$
Shielding	Linear	NMR	$-\sum_i\sum_j g\delta_{ij}\boldsymbol{B}_{\mathrm{ext},j}\cdot\hat{S}_i$
g-Tensor	Linear	EPR	$\frac{\mu_B}{\hbar}\hat{\boldsymbol{B}}\cdot g\cdot\hat{\boldsymbol{S}}$
Nuclear Dipolar	Bilinear	Both	$\frac{\mu_0}{4\pi}\frac{g_1 g_2 \mu_n^2}{\hbar^2}\hat{\boldsymbol{I}}_1\cdot D(\boldsymbol{r}_1-\boldsymbol{r}_2)\cdot\hat{\boldsymbol{I}}_2.$
e-*e* Dipolar	Bilinear	Both	$\frac{\mu_0}{4\pi}\frac{g_1 g_2 \mu_B^2}{\hbar^2}\hat{\boldsymbol{S}}_1\cdot\int_{\mathbb{R}^3}\int_{\mathbb{R}^3}D(\boldsymbol{r}_1-\boldsymbol{r}_2)\sigma^2(\boldsymbol{r})d\boldsymbol{r}_1 d\boldsymbol{r}_2\cdot\hat{\boldsymbol{S}}_2$
J coupling	Bilinear	NMR	$\hat{\boldsymbol{S}}_a^T J_{ab}\hat{\boldsymbol{S}}_b$
Isotropic HFC	Bilinear	EPR	$\frac{2\mu_0}{3}g\mu_e\mu_n\sigma(0)\hat{\boldsymbol{S}}\cdot\hat{\boldsymbol{I}}$
Anisotropic HFC	Bilinear	EPR	$\frac{\mu_0}{4\pi}\frac{g_e g_n \mu_e \mu_n}{\hbar^2}\hat{\boldsymbol{S}}\cdot\int_{\mathbb{R}^3}D(\boldsymbol{r}_e-\boldsymbol{r}_n)\sigma(\boldsymbol{r})d\boldsymbol{r}_e\cdot\hat{\boldsymbol{I}},$
Quadrupolar	Quadratic	NMR	$\frac{eQ}{6I(2I-1)}\sum_{\alpha,\beta}V_{\alpha,\beta}[\frac{3}{2}(\hat{I}_\alpha\hat{I}_\beta+\hat{I}_\beta\hat{I}_\alpha)-\delta_{\alpha\beta}I^2]$
ZFS	Quadratic	EPR	$\boldsymbol{S}_i D\boldsymbol{S}_j$

Table 1.1 – Summary of the common interaction terms in the spin dynamics Hamiltonian. μ in the Zeeman term is either the Bohr magneton or the nuclear magneton as appropriate.

1.6 The Spin Hamiltonian Terms

The purpose of this section is to give a brief overview of the types of terms that appear in spin Hamiltonians and their physical sources. A summary is given in table 1.1. The contents of this section help to motivate the data model used in spin XML in chapter 3 and the assertion that spin Hamiltonians take linear/bilinear/quadratic form, which will be important in chapter 2.

1.6.1 Zeeman Terms

The Hamiltonian [7, 36] for a single spin within a magnetic field is:

$$\hat{H}_{\mathrm{spin}} = -\boldsymbol{B}\cdot\hat{\boldsymbol{\mu}}, \tag{1.77}$$

where $\boldsymbol{B}$ is the applied magnetic field and $\hat{\boldsymbol{\mu}}$ is the magnetic moment operator. In spin dynamics we are usually interested in the ground states with no orbital angular momentum, so it is usually sufficient to assume that $\hat{\boldsymbol{\mu}} = -\frac{g\mu}{\hbar}\hat{\boldsymbol{S}}$, with $\hat{\boldsymbol{S}}$ being the spin angular momentum operator (but see section 1.6.6). g is the g-factor and μ is either the nuclear or Bohr magneton as appropriate.

1.6.2 Chemical Shift

The application of a strong external field induces weak internal currents within molecules, due to motion of the electrons. In effect, the real field felt by each nucleus is a sum, $\boldsymbol{B}_{\mathrm{real}} = \boldsymbol{B}_{\mathrm{ext}} + \boldsymbol{B}_{\mathrm{ind}}$, which gives the induced field in terms of the applied field. The quantity δ_{ij} is called the chemical shift tensor, and B_{ind} is given by [7, p. 193]:

$$\begin{pmatrix} B_{x,\text{induced}} \\ B_{y,\text{induced}} \\ B_{z,\text{induced}} \end{pmatrix} = \begin{pmatrix} \delta_{xx} & \delta_{xy} & \delta_{xz} \\ \delta_{yx} & \delta_{yy} & \delta_{yz} \\ \delta_{zx} & \delta_{zy} & \delta_{zz} \end{pmatrix} \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}. \tag{1.78}$$

In most situations, the coordinate axes in equation (1.78) will be set up so that the applied field falls along the z-axis.

The induced field is then fed back into equation (1.77) to give:

$$H_{\text{shift}} = -\sum_i \sum_j g\delta_{ij} B_{\text{ext},j} \hat{S}_i. \tag{1.79}$$

Clearly, chemical shift can be regarded as a linear term. There is no analytical way to find the induced field that works in all cases, but it can be found numerically (see [37] for a review). Alternatively, there are a number of rules of thumb [7, p. 197]:

- The more "electronegative" the environment of the nucleus, that is, the more the environment of the nucleus tends to pull electrons to itself, the stronger the induced field, which increases the absolute value of δ.
- Heavier atoms tend to have chemical shifts with a bigger absolute value as they possess more low-lying excited states.

1.6.3 Dipole-Dipole Interactions

Dipole-dipole interactions result from the interaction between the dipole magnetic fields of electrons and nuclei. From a classical perspective, dipole-dipole couplings are easy to understand. Spins can be thought of as acting very much like bar magnets [7, p. 203]. The lowest energy configurations are parallel when the spins are laid out end to end and anti-parallel when the spins are laid out side by side.

They are all fundamentally the same from a mathematical perspective so they will be covered all at once, but nuclear-nuclear, nuclear-electron, and electron-electron couplings have different magnitudes and are important in different situations.

The energy between two generic spins is given by:

$$\hat{H}_{\text{dipolar}} = \frac{\mu_0}{4\pi} \frac{g_1 g_2 \mu_1 \mu_2}{\hbar^2} \left(\frac{\hat{\boldsymbol{S}} \cdot \hat{\boldsymbol{L}}}{r^3} - \frac{3(\hat{\boldsymbol{S}} \cdot \boldsymbol{r})(\hat{\boldsymbol{L}} \cdot \boldsymbol{r})}{r^5} \right) \tag{1.80}$$

Where g_1 and g_2 are the g-factors of the $\hat{\boldsymbol{S}}$ and $\hat{\boldsymbol{L}}$ spins respectively, μ_1 and μ_2 are either the Bohr or nuclear magneton depending on the nature of the spins and $\boldsymbol{r}$ is a vector connecting the spins and r is that vector's magnitude. In theory, this expression is complete; $\boldsymbol{r}$ can be treated as an operator and integration over spatial coordinates is implied by quantum

mechanical formalism. However, when performing spin dynamics simulations we do not want to explicitly deal with spatial coordinates, so it is worth recasting equation (1.80) into a form suitable for use with a spin Hamiltonian.

Written in explicit coordinates, equation (1.80) reads:

$$\begin{aligned}\hat{H}_{\text{dipolar}}(\boldsymbol{r}) = \frac{\mu_0}{4\pi}\frac{g_1 g_1 \mu_1 \mu_1}{\hbar}\frac{1}{r^5}[& \\ & + (r^2 - 3x^2)\hat{S}_x\hat{L}_x \\ & + (r^2 - 3y^2)\hat{S}_y\hat{L}_y \\ & + (r^2 - 3z^2)\hat{S}_z\hat{L}_z \\ & - (3xy)(\hat{S}_x\hat{L}_y + \hat{S}_x\hat{L}_x) \\ & - (3xz)(\hat{S}_x\hat{L}_z + \hat{S}_z\hat{L}_x) \\ & - (3yz)(\hat{S}_y\hat{L}_z + \hat{S}_z\hat{L}_y) \\ &].\end{aligned} \tag{1.81}$$

Equation (1.81) may be factored:

$$\hat{H}_{\text{dipolar}} = \frac{\mu_0}{4\pi}\frac{g_1 g_1 \mu_1 \mu_2}{\hbar}\begin{pmatrix}\hat{S}_x & \hat{S}_y & \hat{S}_z\end{pmatrix}\begin{pmatrix}\frac{r^2-3x^2}{r^5} & -\frac{3xy}{r^5} & -\frac{3xz}{r^5} \\ -\frac{3xy}{r^5} & \frac{r^2-3y^2}{r^5} & -\frac{3yz}{r^5} \\ -\frac{3xz}{r^5} & -\frac{3yz}{r^5} & \frac{r^2-3z^2}{r^5}\end{pmatrix}\begin{pmatrix}\hat{L}_x \\ \hat{L}_y \\ \hat{L}_z\end{pmatrix}. \tag{1.82}$$

Let the matrix part be written $D(\boldsymbol{r})$. Notice that $D(\boldsymbol{r}) = D(-\boldsymbol{r})$; consequently, it doesn't really matter which spin is $\hat{\boldsymbol{L}}$ and which is $\hat{\boldsymbol{S}}$. Notice also that D is traceless and so averages to zero in isotropicly tumbling liquids. Consequently, dipole-dipole are unobservable in such liquids because the time scale of an NMR experiment is much greater than the time scale for molecular rotations [7, p. 209].

1.6.3.1 Nuclear Dipole-Dipole Couplings

Nuclear dipole-dipole coupling is the simplest case because nuclei are localised and no spatial integration is required. [7, p. 205]:

$$\hat{H} = \frac{\mu_0}{4\pi}\frac{g_1 g_2 \mu_n^2}{\hbar^2}\hat{\boldsymbol{I}}_1 \cdot D(\boldsymbol{r}_1 - \boldsymbol{r}_2) \cdot \hat{\boldsymbol{I}}_2. \tag{1.83}$$

Due to the r^{-3} term, nuclear dipole-dipole couplings are strongly affected by distance. Therefore, they are very useful for determining the distance between spins and, thus, can be used for determining 3D structures.

1.6.3.2 Anisotropic Hyperfine Couplings

For historical reasons, interactions between electrons and nuclei are given the descriptor "hyperfine". There are actually two separate physical sources of hyperfine interactions that

are relevant in EPR: the isotropic Fermi Contact Interaction and the dipole-dipole interaction between nuclear and electric magnetic moments. The dipole-dipole part of the interaction will be covered here while the isotropic part will left until section 1.6.7, as it does not involve dipoles.

The anisotropic part of the hyperfine coupling is due to the nuclear-electron dipole-dipole interactions. Just as with nucleus-nucleus dipole interactions, these average to zero in low viscosity fluids, but become important in rigid systems. The spatial Hamiltonian is [38, p. 120]:

$$\hat{H} = \frac{\mu_0}{4\pi}\frac{g_e g_n \mu_e \mu_n}{\hbar^2}\hat{\boldsymbol{S}} \cdot D(\boldsymbol{r}_e - \boldsymbol{r}_n) \cdot \hat{\boldsymbol{I}} \tag{1.84}$$

where the nucleus sits at $\boldsymbol{r}_n$ electron sits at $\boldsymbol{r}_e$. The situation is more complicated than the case of pure nuclear dipole-dipole coupling because the electrons are normally delocalised; integration over space must be performed:

$$\hat{H} = \frac{\mu_0}{4\pi}\frac{g_e g_n \mu_e \mu_n}{\hbar^2}\hat{\boldsymbol{S}} \cdot \int_{\mathbb{R}^3} D(\boldsymbol{r}_e - \boldsymbol{r}_n)\sigma(\boldsymbol{r})d\boldsymbol{r}_e \cdot \hat{\boldsymbol{I}}, \tag{1.85}$$

$\sigma(\boldsymbol{r})$ is the spin density.

Equation (1.85) is now in a form suitable for inclusion in a spin Hamiltonian. Conventionally, the terms other than the spin operators, that is, the constant factors and the integrated matrix, are gathered into a matrix is called a.

1.6.3.3 Electron-Electron Couplings

Just as nuclear dipoles interact with other nuclear dipoles, and nuclear dipoles interact with unpaired electrons, if a system contains more than one unpaired electron, the dipoles of each pair of electrons will interact via the now familiar dipole-dipole interaction operator. As with the anisotropic hyperfine, spatial averaging is performed but now both dipoles are delocalised, so a double integral is needed:

$$\hat{H} = \frac{\mu_0}{4\pi}\frac{g_1 g_2 \mu_B^2}{\hbar^2}\hat{\boldsymbol{S}}_1 \cdot \int_{\mathbb{R}^3}\int_{\mathbb{R}^3} D(\boldsymbol{r}_1 - \boldsymbol{r}_2)\sigma(\boldsymbol{r})^2 d\boldsymbol{r}_1 d\boldsymbol{r}_2 \cdot \hat{\boldsymbol{S}}_2 \tag{1.86}$$

$\sigma(\boldsymbol{r})$ is again the spin density.

Spin-spin couplings are the strongest of the dipole-dipole interactions. They, along with spin orbit coupling (section 1.6.6), can contribute to zero field splitting [38, 39].

1.6.4 J-Couplings

J-couplings act on nuclei in a very similar way to dipole-dipole couplings, except they link spins through covalent (as well as hydrogen [7, p. 211]) bonds rather than via through-space interaction. In a covalent single bond, two electrons are shared between nuclei. Because

of the Pauli exclusion principle, both electrons have opposite spin, with the two possible states being $|10\rangle$ and $|01\rangle$. If both nuclei had no spin, then the lowest energy state would be $|01\rangle + |10\rangle$. However, if nuclei have spins, then the wavefunctions are mixed so that it is meaningful to talk about one of the pair of electrons being closer to a nucleus [7, p 215]. Now the lowest energy state is the state where the spin of the near electron and the nucleus are aligned. When two nuclei are anti-aligned, each electron can align with its nearest nucleus. When the nuclei are aligned, only one nuclei can be aligned with an electron.

J-couplings which span more than one covalent bond are possible, but the lowest energy state could be aligned or anti-aligned depending on the circumstances [7, p. 213]. The form of the J-coupling contribution to the Hamiltonian is:

$$\hat{H}_{ab} = \hat{\boldsymbol{S}}_a^T J_{ab} \hat{\boldsymbol{S}}_b, \tag{1.87}$$

where, as in equation (1.83), a and b index all the spins and J_{ab} is a tensor represented as a matrix. Typically, J_{ab} is given in units of Hz rather than radians, so it is common to include a factor of 2π in the Hamiltonian. The form of equation (1.87) indicates that this is bilinear term. In isotropic liquids, motional averaging ensures that the J coupling is always a scalar:

$$J_{\text{iso}} = \frac{1}{3}\operatorname{Tr}(J). \tag{1.88}$$

1.6.5 Electric Couplings

The nuclear charge may also affect the spectrum as well as the nuclear magnetic moment. Reviews can be found in [7, 40] and some basic results will be stated here. The classical interaction energy of a nucleus with charge distribution ρ in potential V is:

$$\begin{aligned} E &= \int_{\mathbb{R}^3} \rho(\boldsymbol{r}) V(\boldsymbol{r}) d\boldsymbol{r} \\ &= V(0) \int_{\mathbb{R}^3} \rho(\boldsymbol{r}) d\boldsymbol{r} + \sum_{i=1}^{3} V_i \int_{\mathbb{R}^3} x_i(\boldsymbol{r}) \rho(\boldsymbol{r}) d\boldsymbol{r} + \sum_{i=1}^{3}\sum_{j=1}^{3} V_i \int_{\mathbb{R}^3} x_i(\boldsymbol{r}) x_j(\boldsymbol{r}) \rho(\boldsymbol{r}) d\boldsymbol{r} \ldots, \end{aligned} \tag{1.89}$$

where ρ has been expanded as a Taylor series. $x_i(\boldsymbol{r})$ are the "coordinate functions" that extract the i^{th} Cartesian component of the vector they are fed.

The first term of the series may be eliminated as, in a relaxed molecule nuclei always sit in a minimum of the electric potential. The spectrum of electric multipole terms that need to be considered can be cut down further, using some theorems from nuclear physics [7, p. 173]:

- No nuclei have an electric dipole moment.

- The maximum electric multipole moment possessed by a nucleus is twice its spin.

A corollary is that a spin $\frac{1}{2}$ nucleus behaves exactly like a point charge with respect to external electric fields, which is a very fortunate conclusion indeed because nuclei with higher spin can behave in a much more complicated manner, especially as the typical quadrupole action is large compared to all other other interactions except the external static field [7, p. 217].

The quadrupolar Hamiltonian term is [40, p.497]:

$$\hat{H}_{\text{quad}} = \frac{eQ}{6I(2I-1)} \sum_{\alpha,\beta} V_{\alpha,\beta}[\frac{3}{2}(\hat{I}_\alpha \hat{I}_\beta + \hat{I}_\beta \hat{I}_\alpha) - \delta_{\alpha\beta} I^2], \tag{1.90}$$

where Q is the nuclear quadrupole moment, $\hat{I}$ is the nuclear spin operator, α and β are indices ranging over x, y and z, V is the electric potential, and $V_{\alpha\beta} = \frac{\partial V}{\partial\alpha\partial\beta}$.

1.6.6 Spin-Orbit Coupling and the g-Tensor

So far, only the spin states have been considered, however, electrons do have significant spatial degrees of freedom which must be dealt with in some manner. Only non-orbitally degenerate ground states will be examined here as more complicated cases are outside of the scope of this document.

The wavefunction of the electron in the unperturbed system can be written in terms of spatial basis, $|n\rangle$, which are the eigenfunctions of some spatial Hamiltonian and the spin states, $|m_s\rangle$, which are eigenfunctions of the Zeeman Hamiltonian (equation 1.77). The Zeeman Hamiltonian commutes with the spatial Hamiltonian and so the product basis $|n, m_s\rangle$ remain eigenfunctions of the total Hamiltonian.

Treatment begins by modifying the magnetic moment operator to include the effects of orbital angular momentum:

$$\hat{\boldsymbol{\mu}} = -\frac{\mu_B}{\hbar}(\hat{\boldsymbol{L}} + g_e \hat{\boldsymbol{S}}). \tag{1.91}$$

As we are dealing with non-orbitally degenerate ground states the angular momentum must be zero and the above modification would appear to have no effect. However, due to relativistic effects, the motion of an electron couples to its spin state. This is effect is known as the spin-orbit coupling. Its functional form is [38, p. 106-108]:

$$\hat{H}_{\text{so}} = \lambda \hat{\boldsymbol{S}} \cdot \hat{\boldsymbol{L}}, \tag{1.92}$$

where λ is a constant. The full justification for equation (1.92) requires solving the Dirac equation and is of little relevance here. Like any coupling added to a Hamiltonian, the energy eigenfunctions are changed and the old good quantum numbers have to be thrown

out. In this case, spin and angular momentum quantum numbers are lost and a total angular momentum quantum number is gained. In addition, the product basis $|\, n, m_s \,\rangle$ are no longer eigenfunctions of the Hamiltonian. In may cases the spin orbit coupling is small enough to be treated as a perturbation. The total Hamiltonian now reads:

$$\hat{H}_{\text{total}} = \hat{H}_{\text{spatial}} + \lambda \hat{\boldsymbol{S}} \cdot \hat{\boldsymbol{L}} + \frac{\mu_B}{\hbar} \boldsymbol{B} \cdot (\hat{\boldsymbol{L}} + g_e \hat{\boldsymbol{S}}). \tag{1.93}$$

This Hamiltonian is awkward to deal with because it appears to require full knowledge of the spatial states of the electron; gone is the relative simplicity of being able to deal with only a few discrete spin states. Thankfully, Pryce [41] was able to show that so long as the energy gaps between spin states are much smaller than the orbital states, a new Hamiltonian may be constructed that acts purely on the subspace spanned by the orbital ground states (that is, span$\{|\, 0, m_s \,\rangle\}$) that includes the effects of Spin-Orbit coupling to second order. From this Hamiltonian, the well known g-tensor appears naturally. This derivation is based on the paper by Pryce cited above, but other derivations may be found in [39] and [42].

Firstly, let:

$$\hat{H}_{\text{mag}} = \lambda \hat{\boldsymbol{S}} \cdot \hat{\boldsymbol{L}} + \frac{\mu_B}{\hbar} \boldsymbol{B} \cdot (\hat{\boldsymbol{L}} + g_e \hat{\boldsymbol{S}}), \tag{1.94}$$

and let E_n be the energy eigenvalues of the $|\, n, m_s \,\rangle$ basis w.r.t. $\hat{H}_{\text{spatial}}$. Now consider an eigenfunction, $|\, \Psi \,\rangle$, of $\hat{H}_{\text{total}}$:

$$\hat{H}_{\text{total}} \,|\, \Psi \,\rangle = E \,|\, \Psi \,\rangle\,, \tag{1.95}$$

where E is the eigenvalue. As the basis $\{|\, n, m_s \,\rangle\}$ is complete, $|\, \Psi \,\rangle$ may be expanded in this basis:

$$|\, \Psi \,\rangle = \sum_{n,s} \alpha_{n,s} \,|\, n, m_s \,\rangle\,. \tag{1.96}$$

Furthermore, we demand E is close to E_0 so that $E - E_0$ is first order. Operating from the left with $\hat{H}_{\text{spatial}} + \hat{H}_{\text{mag}}$ results in:

$$\begin{aligned} E \,|\, \Psi \,\rangle &= \sum_{n,s} \alpha_{n,s} (E_n + \hat{H}_{\text{mag}}) \,|\, n, m_s \,\rangle \\ 0 &= \sum_{n,s} \alpha_{n,s} (E_n - E + \hat{H}_{\text{mag}}) \,|\, n, m_s \,\rangle \end{aligned} \tag{1.97}$$

And operating again from the left with a projection operator, $\hat{P}_k$, which projects into the k^{th} spatial subspace, gives:

$$\sum_s \alpha_{0,s} (E_0 - E + \hat{P}_0 \hat{H}_{\text{mag}}) \,|\, 0, m_s \,\rangle + \sum_{n \neq 0, s} \alpha_{n,s} \hat{P}_0 \hat{H}_{\text{mag}} \,|\, n, m_s \,\rangle = 0 \tag{1.98}$$

$$\sum_s \alpha_{0,s} \hat{P}_k \hat{H}_{\text{mag}} \,|\, 0, m_s \,\rangle + \sum_{n \neq 0, n \neq k, s} \alpha_{n,s} \hat{P}_k \hat{H}_{\text{mag}} \,|\, n, m_s \,\rangle + \sum_s \alpha_{k,s} (E_k - E) \,|\, k, m_s \,\rangle = 0. \tag{1.99}$$

The distinction between the orbital ground state subspace and the excited states has been deliberately emphasised although the ground state has not yet received special treatment.

Now, as we are dealing with an eigenstate close to the ground states, $\alpha_{n,s}$ with $n > 0$ will be small, as is $\hat{H}_{\text{mag}}$ as it is being treated as a perturbation. Consequently the middle term of equation (1.99) may be dropped as being second order and the replacement $E_n - E \to E_n - E_0$ may be made. After rearrangement:

$$\sum_s \alpha_{n,s} \left| n, m_s \right\rangle = -\frac{\sum_s \alpha_{0,s} \hat{P}_n \hat{H}_{\text{mag}}}{E_n - E_0} \left| 0, m_s \right\rangle . \tag{1.100}$$

This may be substituted into the equation (1.98) in order to write it entirely in terms of the orbital ground state subspace:

$$\sum_s \alpha_{0,s} (E_0 - E + \hat{P}_0 \hat{H}_{\text{mag}}) \left| 0, m_s \right\rangle - \sum_{n \neq 0} \hat{P}_0 \hat{H}_{\text{mag}} \frac{\sum_s \alpha_{0,s} \hat{P}_n \hat{H}_{\text{mag}}}{E_n - E_0} \left| 0, m_s \right\rangle = 0. \tag{1.101}$$

After some more rearrangement and operating from right with $\hat{P}_0$:

$$\left[E_0 + \hat{P}_0 \hat{H}_{\text{mag}} \hat{P}_0 - \sum_{n \neq 0} \frac{\hat{P}_0 \hat{H}_{\text{mag}} \hat{P}_n \hat{H}_{\text{mag}} \hat{P}_0}{E_n - E_0} \right] \sum_s \alpha_{0,s} \left| 0, m_s \right\rangle = E \sum_s \alpha_{0,s} \left| 0, m_s \right\rangle . \tag{1.102}$$

Notice that the bracketed expression is an operator that acts entirely on the orbital ground state subspace; we will call this the spin Hamiltonian. It is also independent of our choice of E and $\left| \Psi \right\rangle$ and has the property that if $\left| \Psi \right\rangle$ is an eigenstate of $\hat{H}_{\text{total}}$ then $\hat{P}_0 \left| \Psi \right\rangle$ is an eigenstate of the spin Hamiltonian with the same eigenvalue (up to the level of approximation used in the derivation).

The only remaining task is to substitute the explicit from of $\hat{H}_{\text{mag}}$. Firstly, note that:

$$\hat{P}_n = \sum_s \left| n, m_s \right\rangle \left\langle n, m_s \right| = \left| n \right\rangle \sum_s \left| m_s \right\rangle \left\langle m_s \right| \left\langle n \right| = \left| n \right\rangle \left\langle n \right| . \tag{1.103}$$

The E_0 term of the spin Hamiltonian is trivial. As were are interested in orbital ground states (with $\left\langle 0 \right| \hat{\boldsymbol{L}} \left| 0 \right\rangle = 0$), the second is reduced to the free electron Zeeman term:

$$\hat{P}_0 \hat{H}_{\text{mag}} \hat{P}_0 = \left| 0 \right\rangle \left\langle 0 \right| \frac{\mu_B}{\hbar} \boldsymbol{B} \cdot (\hat{\boldsymbol{L}} + g_e \hat{\boldsymbol{S}}) + \lambda \hat{\boldsymbol{L}} \cdot \hat{\boldsymbol{S}} \left| 0 \right\rangle \left\langle 0 \right| = \left\langle 0 \right| \frac{g_e \mu_B}{\hbar} \boldsymbol{B} \cdot \hat{\boldsymbol{S}} \left| 0 \right\rangle \left| 0 \right\rangle \left\langle 0 \right| . \tag{1.104}$$

The third term is more involved, but still straightforward:

$$\begin{aligned} &\hat{P}_0 \hat{H}_{\text{mag}} \hat{P}_n \hat{H}_{\text{mag}} \hat{P}_0 = \\ &\quad \left| 0 \right\rangle \left\langle 0 \right| \frac{\mu_B}{\hbar} \boldsymbol{B} \cdot (\hat{\boldsymbol{L}} + g_e \hat{\boldsymbol{S}}) + \lambda \hat{\boldsymbol{L}} \cdot \hat{\boldsymbol{S}} \left| n \right\rangle \left\langle n \right| \frac{\mu_B}{\hbar} \boldsymbol{B} \cdot (\hat{\boldsymbol{L}} + g_e \hat{\boldsymbol{S}}) + \lambda \hat{\boldsymbol{L}} \cdot \hat{\boldsymbol{S}} \left| 0 \right\rangle \left\langle 0 \right| \\ &\quad \left\langle 0 \right| (\frac{\mu_B}{\hbar} \boldsymbol{B} + \lambda \hat{\boldsymbol{S}}) \cdot \hat{\boldsymbol{L}} \left| n \right\rangle \left\langle n \right| \hat{\boldsymbol{L}} \cdot (\frac{\mu_B}{\hbar} \boldsymbol{B} + \lambda \hat{\boldsymbol{S}}) \left| 0 \right\rangle \left| 0 \right\rangle \left\langle 0 \right| \\ &\quad (\frac{\mu_B}{\hbar} \boldsymbol{B} + \lambda \hat{\boldsymbol{S}}) \cdot \left\langle 0 \right| \hat{\boldsymbol{L}} \left| n \right\rangle \left\langle n \right| \hat{\boldsymbol{L}} \left| 0 \right\rangle \cdot (\frac{\mu_B}{\hbar} \boldsymbol{B} + \lambda \hat{\boldsymbol{S}}) \left| 0 \right\rangle \left\langle 0 \right| \end{aligned} \tag{1.105}$$

Dropping factors of $|0\rangle\langle 0|$ on the understanding that the spin Hamiltonian operates within the orbital ground state, and ignoring E_0 as the ground state energy is not spectroscopically, interesting the spin Hamiltonian is:

$$\hat{H}_{\text{spin}} = \frac{g_e \mu_B}{\hbar} \boldsymbol{B} \cdot \hat{\boldsymbol{S}} - \sum_{n \neq 0} \frac{(\frac{\mu_B}{\hbar} \boldsymbol{B} + \lambda \hat{\boldsymbol{S}}) \cdot \langle 0 | \hat{\boldsymbol{L}} | n \rangle \langle n | \hat{\boldsymbol{L}} | 0 \rangle \cdot (\frac{\mu_B}{\hbar} \boldsymbol{B} + \lambda \hat{\boldsymbol{S}})}{E_n - E_0} \tag{1.106}$$

This expression can be tided up by introducing a new tensor:

$$\Lambda = -\sum_{n \neq 0} \frac{\langle 0 | \hat{\boldsymbol{L}} | n \rangle \langle n | \hat{\boldsymbol{L}} | 0 \rangle}{E_n - E_0} \tag{1.107}$$

$$\hat{H}_{\text{spin}} = \frac{g_e \mu_B}{\hbar} \boldsymbol{B} \cdot \hat{\boldsymbol{S}} + \frac{\mu_B^2}{\hbar^2} \boldsymbol{B}^{\mathrm{T}} \cdot \Lambda \cdot \boldsymbol{B} + 2\lambda \frac{\mu_B}{\hbar} \boldsymbol{B} \cdot \Lambda \cdot \hat{\boldsymbol{S}} + \lambda^2 \hat{\boldsymbol{S}} \cdot \Lambda \cdot \hat{\boldsymbol{S}}. \tag{1.108}$$

The second term is uninteresting from a spectroscopic point of view because it is unaffected by spin flips and will be ignored. The full Hamiltonian obtained so far is:

$$\hat{H}_{\text{spin}} = \frac{g_e \mu_B}{\hbar} \boldsymbol{B} \cdot \hat{\boldsymbol{S}} + 2\frac{\lambda \mu_B}{\hbar} \boldsymbol{B} \cdot \Lambda \cdot \hat{\boldsymbol{S}} + \lambda^2 \hat{\boldsymbol{S}} \cdot \Lambda \cdot \hat{\boldsymbol{S}}. \tag{1.109}$$

The Zeeman splitting (the first term) is conventionally combined with the second term by defining a new tensor, the g-tensor:

$$g = g_e \mathbb{1} + 2\lambda\Lambda. \tag{1.110}$$

The third term is known as the *zero field splitting* term because it causes splitting even when there is no applied field. Conventionally, the zero field splitting tensor, D, is defined as follows:

$$D = \lambda^2 \Lambda. \tag{1.111}$$

1.6.7 The Isotropic Hyperfine

To the first approximation, it is not expected that any kind of isotropic hyperfine interaction is found between an electron for a spherically symmetric orbital and a point nucleus. As discussed in section 1.6.3.1, there is no residual dipole-dipole interaction after spherical averaging. Indeed, dipole-dipole interactions are not the cause of the isotropic hyperfine coupling [38, p. 39]. It actually arises because the nucleus is not a point and, at some level, the point-dipole approximation breaks down. Fermi first wrote a semi-classical expression for the energy shift due to the time that the electron spends in the nucleus [43]:

$$E_{\text{iso}} = -\frac{2\mu_0}{3} |\phi(0)|^2 \mu_{ez} \mu_{nz}. \tag{1.112}$$

In isolated atoms, only electrons in s orbitals have a nonzero probability of spending time in the nucleus (p, d and f orbitals have nodes there) and so only s electrons have isotropic Fermi contact shifts.

The fully quantum mechanical Fermi contact equation Hamiltonian can be obtained by substituting the classical dynamical variables with their quantum mechanical operator equivalents [38, p. 46]:

$$\hat{H}_{\text{iso}} = \frac{2\mu_0}{3}\frac{g\mu_e\mu_n}{\hbar^2}\sigma(0)\hat{\boldsymbol{S}}\cdot\hat{\boldsymbol{I}}, \tag{1.113}$$

where $\sigma(0)$ here is the spin density at the nucleus. Spin density is found from an N particle wavefunction, Ψ, via:

$$\sigma(r) = N\int\ldots\int|\Psi(\boldsymbol{r_1}\ldots\boldsymbol{r_N},\alpha)|^2 d^3\boldsymbol{r}_2\ldots d^3\boldsymbol{r_N} - N\int\ldots\int|\Psi(\boldsymbol{r_1}\ldots\boldsymbol{r_N},\beta)|^2 d^3\boldsymbol{r}_2\ldots d^3\boldsymbol{r_N} \tag{1.114}$$

where α represents spin up and β represents spin down.

In magnetic resonance, the coefficient before $\hat{\boldsymbol{S}}\cdot\hat{\boldsymbol{I}}$ is conventionally labelled A_{iso}; the isotropic hyperfine coupling constant.

Putting together the isotropic and anisotropic hyperfine couplings, the total coupling matrix, A, is may therefore be written $A = A_{\text{iso}}\mathbb{1} + a$.

1.6.8 The Pseudocontact Shift

An external field, $\boldsymbol{B}_0$, can induce a dipole moment on a spin:

$$\boldsymbol{\mu} = X\boldsymbol{B_0}, \tag{1.115}$$

where X is the magnetic susceptibility, a quantity which arises through thermodynamics [44, ch. 1]. The resulting dipole can then take part in dipole-dipole interactions as usual.

Magnetic susceptibility can take on tensor character [44, 45]. In this regime, the induced dipoles do not necessarily line up with the external field and can vary in both magnitude and direction under tumbling. Because of these variations, the dipole-dipole interaction no longer averages to zero, there is a residual field that is responsible for the pseudocontact shift.

Figure 1.2 illustrates the situation in the point-dipole approximation, where it is assumed that both dipoles can be localised to a point. The central dipole will be referred to as the paramagnetic metal atom, although the model presented here would apply to any pair of dipoles. The magnetic moment of the metal is given by [44, p. 39]:

$$\langle\boldsymbol{\mu}\rangle = \frac{B_0}{\mu_0}(\chi_{\parallel}\cos\alpha\boldsymbol{\lambda} - \chi_{\perp}\sin\alpha\boldsymbol{\nu}), \tag{1.116}$$

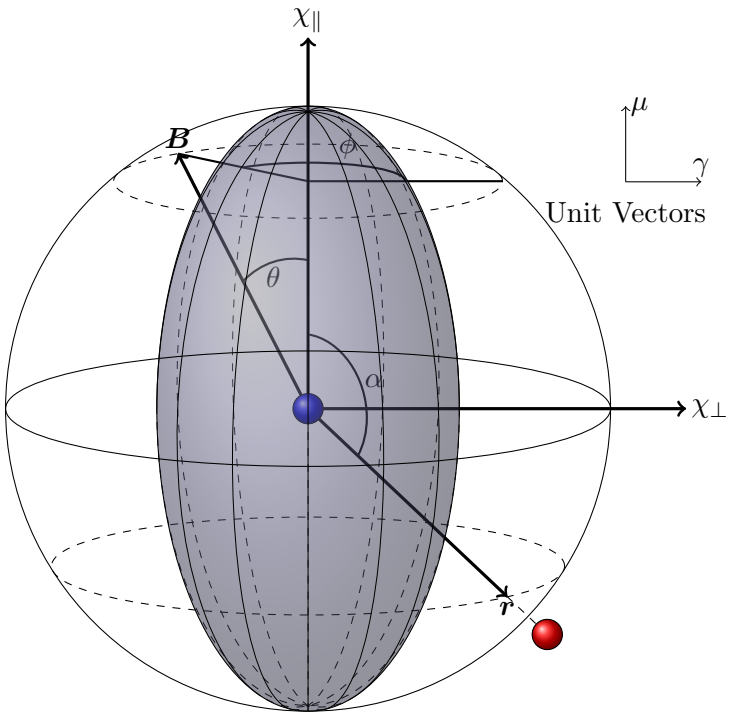


Figure 1.2 – A metal with an anisotropic magnetic susceptibility tensor with an attached ligand sitting at $\boldsymbol{r}$ in a magnetic field. Under rapid tumbling the magnetic field sweeps out the surface of the sphere.

where $\chi_\perp$ and $\chi_\parallel$ are the perpendicular and parallel components of the susceptibility tensor, and $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ are unit vectors along those axes. Note that axial symmetry of χ has been assumed. A more general derivation of the pseudocontact shift can be found in chapter 5. The above equation can now be substituted into the dipole-dipole energy equation (equation 1.83) to obtain:

$$\begin{aligned}\frac{\Delta E}{\hbar\gamma B_0} = \delta_{dipolar} = \frac{1}{4\pi r^3}[& \\ & + \chi_\parallel \cos^2\theta(3\cos^2\alpha - 1) \\ & + \chi_\perp \sin^2\theta(3\sin^2\alpha\cos^2\phi - 1) \\ & + \frac{3}{4}(\chi_\parallel + \chi_\perp)\sin 2\theta \sin 2\alpha \cos\phi \\ ]. &\end{aligned} \tag{1.117}$$

To obtain the mean shift under rotational averaging, the dipolar shifts need to be spherically integrated as the direction of the magnetic field varies:

$$\delta_{pcs} = \int_0^{2\pi}\int_0^{\pi} \delta_{\text{dipolar}} \sin\theta d\theta d\phi. \tag{1.118}$$

After some tedious algebra (or use of a computer algebra package) the point dipole pseudocontact shift term emerges:

$$\delta_{pcs} = \frac{1}{12\pi r^3}(\chi_{\parallel} - \chi_{\perp})(3\cos^2\alpha - 1). \tag{1.119}$$

The pseudocontact potential has proven a useful tool for constraining the structures of molecules, particularly metalloproteins [46,47], because the pseudocontact shift only depends on a ligand's displacement vector from a paramagnetic centre [47,48]. A quarter of naturally occurring proteins have a suitable paramagnetic metal ion and, even when a protein lacks such an ion, one can sometimes be inserted [46]. Because of this prevalence, PCS represents an important tool for protein structure determination.

Chapter 2

Accuracy of the State Space Restriction Approximation

When performing spin dynamics simulations using approximate state space restriction methods (such as those described in [6, 22, 28, 29] and section 1.4) it would generally be desirable to know the degree to which the results of such simulations can be trusted. This chapter presents an attempt at deriving conditions for the validity of the state space restriction method based on tracking the flow of the density matrix norm through the spin order subspaces. In addition to this chapter, details of this method have been published here [1].

2.1 Statement of the Problem

Let $\hat{\rho}(t)$ be the simulated trajectory of a spin system evolving under a Liouvillian, $\hat{\hat{H}}(t)$, and a relaxation superoperator, $\hat{\hat{R}}$, written in terms of basis $\mathcal{B} = \{\hat{\boldsymbol{e}}_i\}$, and let $\hat{\rho}_{\mathrm{ssr}}(t)$ be the simulated trajectory under a restricted state space simulation that excludes a certain subset of $\mathcal{B}$. There are a number of questions that could be asked that seem to touch on the accuracy of the simulation.

The most direct question would be *what is the difference between* $\langle \hat{O}, \hat{\rho}(t) \rangle$ *and* $\langle \hat{O}, \hat{\rho}_{ssr}(t) \rangle$ *for a given observable,* $\hat{O}$*?* The problem with this question is that observables that are interesting in spin dynamics tend to have oscillatory behaviour and, so, the absolute difference between the trajectories of observables might not be informative. Consequently, we might try asking questions about the Fourier transform of the trajectory of $\langle \hat{O} \rangle(t)$ but, again, this might not always be appropriate. For example, if $\hat{O}$ is the detection state operator, then the trajectory will be the FID and the Fourier Transform will be the spectra. Two physically similar spectra, however, often have misleadingly high differences in obvious measures of dissimilarity, such as the squared difference (this phenomena causes problems when trying to perform direct structure fitting and is discussed in more detail in section 4.1.1). This

problem might still be fixed by taking inspiration from direct structure fitting (see section 4.1.1) and asking questions about the integral of the Fourier transform of the trajectory of our observable.

Even though only upper bounds on errors are required, the analysis of these questions is quite complicated. To make progress, we shall instead ask questions about the flow of the square of the density operator norm through the spin order subspaces. Tracking this flow will turn out to be a much more tractable problem.

2.2 The Flow of the Density Matrix Norm Under Unitary Transforms

The specific matrix norm we are interested is the Frobenius norm, defined as $|\hat{\rho}| =: \sqrt{\mathrm{Tr}(\hat{\rho}^\dagger \hat{\rho})}$, where $\hat{\rho}$ is the density matrix. If we choose to view the density matrix as a vector in Liouville space, then this norm is simply the length of that vector. It is well known that, for a pure states, $|\hat{\rho}| = 1$, while for statistical mixtures $|\hat{\rho}| < 1$ [20].

Let $\hat{\hat{\pi}}_V$ be a projection operator into subspace V, then the compliment of $\hat{\hat{\pi}}_V$ is given by $\hat{\hat{1}} - \hat{\hat{\pi}}_V$. The following relation is generally true:

$$|\hat{\hat{\pi}}_V \hat{\rho}|^2 + |(\hat{\hat{1}} - \hat{\hat{\pi}}_V)\hat{\rho}|^2 = |\hat{\rho}|^2. \tag{2.1}$$

This can be shown by choosing an orthogonal basis where all basis vectors lie in either the subspace or its compliment and applying the Pythagorean theorem.

We will refer to the number $|\hat{\hat{\pi}}_V \hat{\hat{\rho}}|$ as the subspace norm of $\hat{\hat{\rho}}$ with respect to subspace V. Sometimes the specification of the subspace will be omitted when is clear from the context.

Under unitary propagation of a quantum mechanical system, the norm of the entire density matrix is preserved, implying the sum of squared norm across any set of subspaces is also preserved, thus any coherence disappearing from a particular subspace must reappear in the other subspaces. We can thus draw a very strong analogy between the density matrix norm and the flow of an incompressible fluid.

Armed with this analogy, we can re-cast the problem of error estimation as a problem of tracking the flow of an incompressible fluid (representing the squared norm) between a network of tanks (representing the subspaces). This approach has the advantages of being both highly intuitive and mathematically rigorous.

State space restriction corresponds to the removal of "tanks" (states) from the system and the sealing of the "pipes" that lead to to them (corresponding to the requirement, noted in the original state space restriction paper [6], that propagation remains unitary). In order to make progress with the analysis, only spin order pruning, rather than the general case, will be considered, with one "tank" per spin order.

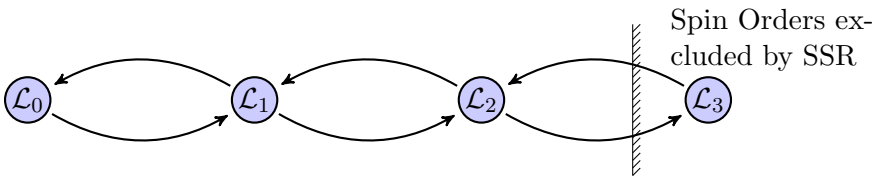


Figure 2.1 – Coherence flow between spin order subspaces. Only adjacent subspaces are directly linked so coherence starting in $\mathcal{L}_0$ requires time to reach higher spin orders.

Already the problem is looking more tractable than the general, exponentially scaling, problem as it corresponds to a system of n tanks, one for each spin order, and $O(n^2)$ pipes. The situation is further improved by the observation that each spin order will only be linked to neighbouring spin orders for all Hamiltonians of the form described by equation (2.4) (such as those described in section 1.6), producing a linear system of sparsely coupled ordinary differential equations. The proof of this assertion will be the subject of 2.3. The situation is summarised in figure 2.1.

For the vast majority of spin dynamics experiments, the only directly observable states are in the first spin order. Exceptions to this rule generally involve continuous wave experiments (some details can be found in [8]), but here we will assume the usual case of detection being performed in the first spin order. The only scope for the dynamics of the higher spin orders to affect observables is for coherence to flow to them and then return to the first spin order.

2.3 Analysis of the Hamiltonian

Consider the full state space of a system of n spins, $\mathcal{L}$. This space may be decomposed into the direct sum of spin order subspaces:

$$\hat{\rho} \in \mathcal{L} = \mathcal{L}_0 \oplus \mathcal{L}_1 \oplus \cdots \oplus \mathcal{L}_n. \tag{2.2}$$

In the product basis, a basis element belongs to spin order k, where k is the number of non-identity operators in the direct product. The dimensionality of each subspace is therefore $3^k\binom{n}{k}$.

The density matrix, $\hat{\rho} \in \mathcal{L}$, evolves under the master equation:

$$i\hbar\frac{\partial\hat{\rho}}{\partial t} = \hat{\hat{H}}\hat{\rho} + \hat{\hat{R}}(\hat{\rho} - \hat{\rho}_{\text{eq}}), \tag{2.3}$$

where $\hat{\rho} \in \mathcal{L}$ is the density matrix, $\hat{\hat{R}}$ is the relaxation superoperator, and $\hat{\hat{H}}$ is the Liouvillian superoperator, which in general takes the form:

$$\hat{\hat{H}} = \sum_i \hat{\boldsymbol{S}}_i \cdot A_i \cdot \boldsymbol{B} + \sum_{i,j} \hat{\boldsymbol{S}}_i \cdot B_{i,j} \cdot \hat{\boldsymbol{S}}_j + \sum_i \hat{\boldsymbol{S}}_i \cdot C_i \cdot \hat{\boldsymbol{S}}_i, \tag{2.4}$$

where $\{\hat{\boldsymbol{S}}_i\}$ is the set of single spin operators.

Theorem 2.1. *$\hat{\hat{H}}$ only links spin order subspaces to spin order subspaces within one spin order:*

$$\begin{aligned}
[\hat{\hat{H}}, \mathcal{L}_0] &\subset \mathcal{L}_0 \oplus \mathcal{L}_1 \\
[\hat{\hat{H}}, \mathcal{L}_k] &\subset \mathcal{L}_{k-1} \oplus \mathcal{L}_k \oplus \mathcal{L}_{k+1} \\
[\hat{\hat{H}}, \mathcal{L}_n] &\subset \mathcal{L}_{n-1} \oplus \mathcal{L}_n,
\end{aligned} \tag{2.5}$$

where k runs from 1 to $n-1$.

This theorem will be proved separately for linear, bilinear and quadratic terms in turn in section 2.3.1.

2.3.1 Term by Term Analysis

In the following section, basis elements of the full density matrix will be represented as:

$$\hat{T}_{b(1)} \otimes \cdots \otimes \hat{T}_{b(k)} \otimes \cdots \otimes \hat{T}_{b(n)}, \tag{2.6}$$

where n is the number of spins in the system and $\hat{T}$ is an irreducible spherical tensor (see section 1.4.3) or the identity. $b(k)$ either gives the l and m value for that tensor or sets the tensor to the identity $\hat{E}$.

Hamiltonian terms will be represented as:

$$\hat{E} \otimes \cdots \otimes \hat{S}_\alpha \otimes \cdots \otimes \hat{E}, \tag{2.7}$$

where $\alpha \in \{+, -, z\}$. Notice that Hamiltonian terms are the tensor products of, at most, two non-identity terms. It will be assumed that each term in the tensor product is written in the appropriate irreducible Lie algebra representation (or alternatively, they can be regarded as abstract operators, divorced of an explicit matrix representation).

Theorem 2.2. *Linear terms, such as Zeeman interaction terms, do not connect spin order subspaces.*

Proof. Through linearity of the commutator, the analysis can be restricted to basis elements. The general commutator between a general basis term in the density matrix, $\hat{T}_{b(1)} \otimes \cdots \otimes \hat{T}_{b(n)}$, and a linear term from the Hamiltonian, $E, \ldots, \hat{S}_\alpha, \ldots, E$, for a system of n spins is:

$$[E \otimes \cdots \otimes \hat{S}_\alpha \otimes \cdots \otimes E, \hat{T}_{b(1)} \otimes \cdots \otimes \hat{T}_{b(k)} \otimes \cdots \otimes \hat{T}_{b(n)}]. \tag{2.8}$$

By applying the following identity of associative algebras:

$$[A \otimes B, C \otimes D] = AC \otimes [B, D] + [A, C] \otimes DB, \tag{2.9}$$

expression (2.8) becomes:

$$\hat{T}_{b(1)} \otimes \cdots \otimes [\hat{S}_\alpha, \hat{T}_{b(i)}] \otimes \cdots \otimes \hat{T}_{b(n)}, \tag{2.10}$$

and so it can be seen that a linear term in the Hamiltonian only "sees" the part of the density matrix corresponding to the spin the linear term governs.

The important part of this string of tensor products is $[\hat{S}_\alpha, \hat{T}_{b(i)}]$. Using the defining commutation relations (equation 1.49) of the irreducible spherical tensors, $[\hat{S}_\alpha, \hat{T}_{b(i)}]$ is either zero (if $\hat{T}_{b(i)} = \hat{E}$ or $\hat{T}_{b(i)}$ is annihilated by a raising or lowering operator) or just another irreducible spherical tensor from the same invariant subspace. In either case, term (2.8) remains in the same spin order subspace as the original Hamiltonian term. □

Theorem 2.3. *Bilinear terms connect the n^{th} spin order subspaces with the n^{th}, the $(n+1)^{th}$ and the $(n-1)^{th}$ spin order subspaces only.*

Proof. The action of a general bilinear term is:

$$[\hat{S}_{\alpha,i}\hat{S}_{\beta,j}, \hat{T}_{b(1)} \otimes \cdots \otimes \hat{T}_{b(n)}] \quad i \neq j, \tag{2.11}$$

this expression can be decomposed with the identity $[AB, C] = [A, C]B + A[B, C]$, where A, B and C are elements of an associate algebra:

$$[\hat{S}_{\alpha,i}\hat{S}_{\beta,j}, \hat{T}_{b(1)} \otimes \cdots \otimes \hat{T}_{b(n)}] = \overbrace{\underbrace{[\hat{S}_{\alpha,i}, \hat{T}_{b(1)} \otimes \cdots \otimes \hat{T}_{b(n)}]}_{\text{as the linear case}} \hat{S}_{\beta,j}}^{\text{Case A}} + \overbrace{\hat{S}_{\alpha,i} \underbrace{[\hat{S}_\alpha, \hat{T}_{b(1)} \otimes \cdots \otimes \hat{T}_{b(n)}]}_{\text{as the linear case}}}^{\text{Case B}}. \tag{2.12}$$

The two commutators on the right hand side can be evaluated in the same manner as the linear terms (see theorem 2.2 above), so remain in the n^{th} spin order subspace. The first term (case A) may be written out in the general form:

$$(\hat{T}_{b(1)} \otimes \cdots \otimes \hat{T}_{b(i)} \otimes \cdots \otimes \hat{T}_{b(n)})(\hat{E} \otimes \cdots \otimes \hat{S}_\alpha \otimes \cdots \otimes \hat{E}). \tag{2.13}$$

By using the identity $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$ we can see that the product is:

$$\hat{T}_{b(1)} \otimes \cdots \otimes \hat{T}_{b(i)}\hat{S}_\alpha \otimes \cdots \otimes \hat{T}_{b(n)}. \tag{2.14}$$

Depending on the values of $b(i)$ and α, the of the following may occur:

1. $b(i) = \hat{E}$, the product is $\hat{S}_\alpha$ and so the expression (2.14) is in the $(n+1)^{\text{th}}$ spin order subspace.

2. $\hat{T}_{b(i)}\hat{S}_\alpha = \hat{E}$, so the expression (2.14) is in the $(n-1)^{\text{th}}$ spin order subspace.

3. $\hat{T}_{b(i)}\hat{S}_\alpha = \hat{S}_\beta$ with $\beta \neq \hat{E}$, so the expression (2.14) is in the n^{th} spin order subspace.

Similar logic applies to the second term. □

Theorem 2.4. *Quadratic terms connect the n^{th} spin order subspaces with the n^{th}, the $(n+1)^{th}$, and the $(n-1)^{th}$ spin order subspaces only.*

Proof. Once again, we begin by considering the general quadratic term:

$$[\hat{S}_{\alpha,i}\hat{S}_{\beta,i}, \hat{T}_{b(1)} \otimes \cdots \otimes \hat{T}_{b(n)}]. \tag{2.15}$$

As with the linear case, the expression may be simplified by applying identity (2.9):

$$\hat{T}_{b(1)} \otimes \cdots \otimes [\hat{S}_\alpha\hat{S}_\beta, \hat{T}_{b(i)}] \otimes \cdots \otimes \hat{T}_{b(n)}. \tag{2.16}$$

Once again, if $b(i) = E$, then the spin order increases, if $[\hat{S}_\alpha\hat{S}_\beta, \hat{T}_{b(i)}] = \hat{E}$ then the spin order decreases otherwise the spin order remains the same. □

2.4 Relaxation Superoperator

In section 1.3.1, the Redfield relaxation superoperator was derived and at the heart of the theory was a double commutator with the Hamiltonian. Taking equation (1.31) from the derivation and recasting without the interaction representation gives:

$$\frac{d\tilde{\rho}}{dt} = -\frac{i}{\hbar}[\hat{H}_0 + \hat{H}_1(t), \rho(0)] - \frac{1}{\hbar^2}\int_0^t dt'[\hat{H}_1(t), [\hat{H}_1(t'), \rho(t')]], \tag{2.17}$$

From here, numerous approximations and assumptions were made to get to the Redfield superoperator but only one is important: that on the timescale of the fluctuations in $\hat{H}_1$, ρ is essentially static. This let us replace $\rho(t')$ with $\rho(0)$ and move it outside the integral.

$$\frac{d\tilde{\rho}}{dt} = -\frac{i}{\hbar}[\hat{H}_0 + \hat{H}_1(t), \rho(0)] - \frac{1}{\hbar^2}\left(\int_0^t dt'[\hat{H}_1(t), [\hat{H}_1(t'), \ \cdot\ ]]\right)\rho(0), \tag{2.18}$$

The integral symbol is linear; summing over operators that only change the spin order subspace by at most ± 1 results in an operator that still only changes the spin order by ± 1. A careful reading of all the other assumptions and approximations shows that they, too, are linear (such as summing over an ensemble or changing the integration limit to infinity) or irrelevant (assuming certain statistical properties of the ensemble we are summing over; linearity was guaranteed by the summing).

As the Redfield superoperator is formed from the sum of double commutators of the perturbative parts of an ensemble of Hamiltonians and we already know the effects of a single commutators it is clear that:

$$\hat{\hat{R}}\hat{\rho}_k \subset \mathcal{L}_{k-2} \oplus \mathcal{L}_{k-1} \oplus \mathcal{L}_k \oplus \mathcal{L}_{k+1} \oplus \mathcal{L}_{k+2}, \tag{2.19}$$

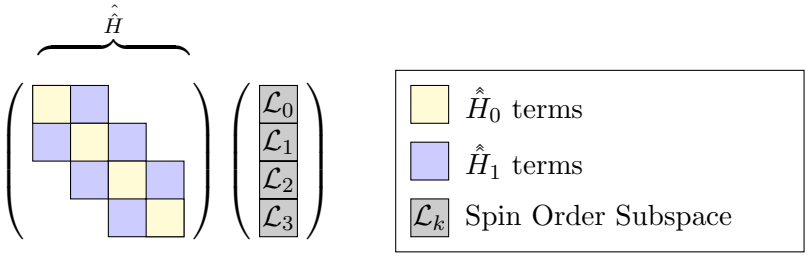


Figure 2.2 – The Liouvillian Matrix in terms of spin orders. $\mathcal{L}_0$ terms mix coherence within subspaces only, while $\mathcal{L}_1$ terms mix between adjacent subspaces.

As with the Liouvillian, the Redfield superoperator may be decomposed such that:

$$\hat{\hat{R}} = \hat{\hat{R}}_1 + \hat{\hat{R}}_2 \quad \hat{\hat{R}}_1\rho_k \in \mathcal{L}_k \quad \hat{\hat{R}}_2\rho_k \in \mathcal{L}_{k-2} \oplus \mathcal{L}_{k-1} \oplus \mathcal{L}_{k+1} \oplus \mathcal{L}_{k+2}. \tag{2.20}$$

The $\hat{\hat{R}}_2$ term, which is responsible for cross-correlated relaxation processes, will be assumed to be small and, therefore, neglected. Note that that the important result established here was that coherence cannot jump large distances up the "ladder" of spin orders via cross correlation as this would cause problems in the following analysis even for very small cross correlation terms.

2.5 Flow Between Subspaces

The dynamics of the density matrix with the Redfield-type relaxation superoperator is completely encoded in the master equation:

$$\frac{\partial\hat{\rho}}{\partial t} = -\frac{i}{\hbar}\hat{\hat{H}}\hat{\rho} - \frac{1}{\hbar^2}\hat{\hat{R}}(\hat{\rho} - \hat{\rho}_{eq}). \tag{2.21}$$

$\hat{\hat{H}}$ may be decomposed into $\hat{\hat{H}}_0$ and $\hat{\hat{H}}_1$ such that:

$$\hat{\hat{H}} = \hat{\hat{H}}_0 + \hat{\hat{H}}_1, \quad \hat{\hat{H}}_0\hat{\rho}_i \subset \mathcal{L}_i, \quad \hat{\hat{H}}_1\hat{\rho}_i \subset \mathcal{L}_{i+1} \oplus \mathcal{L}_{i-1}. \tag{2.22}$$

By the theorems proven in sections 2.3.1 and 2.4, there must be a basis set in which $\hat{\hat{H}}$ has the form given in figure 2.2. Linear terms contribute solely to the $\hat{\hat{H}}_0$ elements (yellow) while bilinear and quadratic interaction may also contribute to the $\hat{\hat{H}}_1$ elements.

Equation (2.21) may be decomposed into coupled differential equations governing each spin order subspace.

$$\begin{aligned}
\frac{\partial\hat{\rho}_1}{\partial t} &= -\frac{i}{\hbar}\hat{\hat{H}}_0\hat{\rho}_1 - \frac{i}{\hbar}\hat{\hat{\pi}}_0\hat{\hat{H}}_1(\hat{\rho}_0 + \hat{\rho}_1) + \frac{1}{\hbar^2}\hat{\hat{R}}_1(\hat{\rho}_1 - \hat{\rho}_{\mathrm{eq}}) \\
\frac{\partial\hat{\rho}_k}{\partial t} &= -\frac{i}{\hbar}\hat{\hat{H}}_0\hat{\rho}_k - \frac{i}{\hbar}\hat{\hat{\pi}}_k\hat{\hat{H}}_1(\hat{\rho}_{k-1} + \hat{\rho}_k + \hat{\rho}_{k+1}) + \frac{1}{\hbar^2}\hat{\hat{R}}_1(\hat{\rho}_k) \\
\frac{\partial\hat{\rho}_n}{\partial t} &= -\frac{i}{\hbar}\hat{\hat{H}}_0\hat{\rho}_n - \frac{i}{\hbar}\hat{\hat{\pi}}_0\hat{\hat{H}}_1(\hat{\rho}_{n-1} + \hat{\rho}_n) + \frac{1}{\hbar^2}\hat{\hat{R}}_1(\hat{\rho}_n).
\end{aligned} \tag{2.23}$$

To save having to write out three nearly identical equations, it will be understood that $\mathcal{L}_k$, where $k < 0$ or $k > n$, are trivial vector spaces containing the zero-vector only.[1] With this change of notation, the equations (2.23) may instead be written:

$$\frac{\partial \hat{\rho}_k}{\partial t} = -\frac{i}{\hbar}\hat{\hat{H}}_0\hat{\rho}_k - i\hat{\hat{\pi}}_k\hat{\hat{H}}_1(\hat{\rho}_{k-1} + \hat{\rho}_k + \hat{\rho}_{k+1}) + \frac{1}{\hbar^2}\hat{\hat{R}}_1(\hat{\rho}_k - \delta_{1,k}\hat{\rho}_{\mathrm{eq}}), \tag{2.24}$$

where δ_{ij} is the Kronecker delta symbol.

We are interested in the dynamics of each of the the subspace norms of the density matrix (section 2.2). The derivatives of these subspace norms are given by:

$$\frac{\partial |\hat{\rho}_k|^2}{\partial t} = \mathrm{Tr}\left[\frac{\partial \rho_k^\dagger}{\partial t}\rho_k + \rho_k^\dagger \frac{\partial \rho_k}{\partial t}\right] \tag{2.25}$$

We can then substitute equation (2.24) into the sub-expression within the trace symbol of equation (2.25) to get:

$$\begin{aligned}\frac{\partial \hat{\rho}_k^\dagger}{\partial t}\hat{\rho}_k + \hat{\rho}_k^\dagger\frac{\partial \hat{\rho}_k}{\partial t} =& \frac{i}{\hbar}\hat{\rho}_k^\dagger\hat{\hat{H}}_0\hat{\rho}_k - \frac{i}{\hbar}\hat{\rho}_k^\dagger\hat{\hat{H}}_0\hat{\rho}_k + \\ & \frac{i}{\hbar}(\hat{\rho}_{k-1}^\dagger + \hat{\rho}_k^\dagger + \hat{\rho}_{k+1}^\dagger)\hat{\hat{H}}_1\hat{\hat{\pi}}_k^\dagger\hat{\rho}_k - \frac{i}{\hbar}\hat{\rho}_k^\dagger\hat{\hat{\pi}}_k\hat{\hat{H}}_1(\hat{\rho}_{k-1} + \hat{\rho}_k + \hat{\rho}_{k+1}) + \cdot \\ & \frac{1}{\hbar^2}\hat{\hat{R}}_1^\dagger(\hat{\rho}_k^\dagger - \delta_1^k\hat{\rho}_{\mathrm{eq}}^\dagger)\hat{\rho}_k + \frac{1}{\hbar^2}\hat{\rho}_k^\dagger\hat{\hat{R}}_1(\hat{\rho}_k - \delta_{1,k}\hat{\rho}_{\mathrm{eq}}).\end{aligned} \tag{2.26}$$

The first line vanishes as expected, because $\hat{\hat{H}}_0$ was defined to not carry coherence across subspaces. On the second line, the $\hat{\hat{\pi}}_k$ operators have no effect because they appear acting on the terms $\hat{\rho}_k$, but $\hat{\rho}_k$ is already in the k^{th} subspace by definition. Terms of the form $\mathrm{Tr}(\hat{\rho}_p\hat{\hat{O}}\hat{\rho}_q)$ can be recognised as scalar products of vectors and are, therefore, simply complex numbers.

Using the identity $iz - iz^* = -2\,\mathrm{Im}\,z$

$$\begin{aligned}\frac{\partial |\hat{\rho}_k|^2}{\partial t} =& -\frac{2}{\hbar}\,\mathrm{Im}\,\langle\,\hat{\rho}_{k+1}\,|\,\hat{\hat{H}}_1\,|\,\hat{\rho}_k\,\rangle + \frac{2}{\hbar}\,\mathrm{Im}\,\langle\,\hat{\rho}_k\,|\,\hat{\hat{H}}_1\,|\,\hat{\rho}_{k-1}\,\rangle \\ & + \frac{2}{\hbar^2}\,\langle\,\hat{\rho}_k\,|\,\hat{\hat{R}}_1\,|\,\hat{\rho}_k\,\rangle - \frac{2\delta_{1,k}}{\hbar^2}\,\mathrm{Re}\,\langle\,\hat{\rho}_k\,|\,\hat{\hat{R}}_1\,|\,\hat{\rho}_{\mathrm{eq}}\,\rangle\end{aligned} \tag{2.27}$$

For clarity, the trace of two linear operators has been written in Dirac notation.

Making the substitutions:

[1] This roughly corresponds to notation used with ladder operators. Without adding zero dimensional vector spaces, expressions such as $J_+\,|\,l,m\,\rangle = \hbar\sqrt{(l-m)(l+m+1)}\,|\,l,m+1\,\rangle$ would need to be treated as special cases because $|\,l,m+1\,\rangle$ with $m > l$ would be formally undefined.

$$\frac{1}{\hbar}\,\mathrm{Im}\,\langle\,\hat{\rho}_{k+1}\,|\,\hat{\hat{H}}_1\,|\,\hat{\rho}_k\,\rangle \to h_k(t)|\hat{\rho}_k||\hat{\rho}_{k+1}|, \tag{2.28}$$

$$\frac{1}{\hbar^2}\,\langle\,\hat{\rho}_k\,|\,\hat{\hat{R}}_1\,|\,\hat{\rho}_k\,\rangle \to r_k(t)|\hat{\rho}_k|^2, \quad \text{and} \tag{2.29}$$

$$\frac{1}{\hbar^2}\,\mathrm{Re}\,\langle\,\hat{\rho}_0\,|\,\hat{\hat{R}}_1\,|\,\hat{\rho}_{\mathrm{eq}}\,\rangle \to g(t)|\hat{\rho}_k||\hat{\rho}_{\mathrm{eq}}|, \tag{2.30}$$

we can write equation (2.27) as:

$$\frac{\partial|\hat{\rho}_k|^2}{\partial t} = -2h_k(t)|\hat{\rho}_{k+1}||\hat{\rho}_k| + 2h_{k-1}(t)|\hat{\rho}_k||\hat{\rho}_{k-1}| + 2r_k(t)|\hat{\rho}_k|^2 - 2\delta_{1,k}g(t)|\hat{\rho}_k||\hat{\rho}_{\mathrm{eq}}| \tag{2.31}$$

Noting that $\frac{\partial|\hat{\rho}_k|^2}{\partial t} = 2\frac{\partial|\hat{\rho}_k|}{\partial t}|\hat{\rho}_k|$, the above simplifies to:

$$\frac{\partial|\hat{\rho}_k|}{\partial t} = -h_k(t)|\hat{\rho}_{k+1}| + h_{k-1}(t)|\hat{\rho}_{k-1}| + r_k(t)|\hat{\rho}_k| - \delta_{1,k}g(t)|\hat{\rho}_{\mathrm{eq}}| \tag{2.32}$$

This equation is still formally exact, but, in general, the h_k, r_k and g coefficients depend on time in a complicated manner. However, it will be possible to find an upper bound on the absolute value using the following theorem:

Theorem 2.5. *If A is a linear operator over a complex vector space and $\boldsymbol{u}, \boldsymbol{v}$ are vectors, then $\boldsymbol{u}^\dagger A\boldsymbol{v} = \alpha|\boldsymbol{u}||\boldsymbol{v}|$ where $|\alpha| \leq \lambda_{max}$ and where λ_{max} is the maximum absolute value of the eigenvalues of A.*

Proof. Firstly, note that $\frac{\boldsymbol{u}^\dagger A\boldsymbol{v}}{|\boldsymbol{u}||\boldsymbol{v}|} = \frac{\boldsymbol{u}^\dagger}{|\boldsymbol{u}|}A\frac{\boldsymbol{v}}{|\boldsymbol{v}|} = \alpha$, so we can restrict our attentions to unit vectors. Writing $\boldsymbol{u}^\dagger A\boldsymbol{v}$ in a basis where A is diagonal:

$$\sum_j u_j v_j \lambda_j, \tag{2.33}$$

with u_i and v_i being the complex components of each vector. Writing this in polar form:

$$\sum_j |u_j||v_j||\lambda_j| \exp(i\theta_j), \tag{2.34}$$

where θ_j gives the total complex phase of each term. The absolute value of this expression as the θ_js vary is maximised when all $\exp(i\theta_j)$s are in phase:

$$\sum_j |u_j||v_j||\lambda_j|. \tag{2.35}$$

The total expression is maximised when $|u_j| = 1$ and $|v_j| = 1$ for the maximum of $\{|\lambda_j|\}$, providing the upper bound on α. □

As a consequence of the above theorem, h_k must be bounded. As we are interested in in the worst case scenario, we can replace h_k its maximum value:

$$h_k \to h = \frac{1}{\hbar}\,\text{absmax}\,\text{eig}(\hat{\hat{H}}_1). \tag{2.36}$$

The assumption is made that $r_1, g > \frac{1}{\hbar^2}\,\text{absmin}\,\text{eig}\,\hat{\hat{R}}$.[2] This assumption could be extended to all r_k, but this would be overly conservative because higher spin orders do relax faster than lower spin orders (see the papers of *Pama et al* [49] and *Unruh* [50]), so we instead make the assumption that only the lowest order subspace relaxes at the rate $\text{absmin}\,\text{eig}(\hat{\hat{R}}_1)$, and relaxation in higher order subspaces increase in proportion to the spin order:

$$r_k \to kr, \quad g \to r \quad \text{where } r = \text{absmin}\,\text{eig}(\hat{\hat{R}}_1). \tag{2.37}$$

Carravetta *et al.* [51] and Pileio *et al.* [52] recently found systems where relaxation is proportional to the square root of the spin order, rather than to the spin order itself, so we will also analyse the case where the substitution:

$$r_k \to \sqrt{k}r = \frac{\sqrt{k}}{\hbar^2}\,\text{absmin}\,\text{eig}(\hat{\hat{R}}_1), \tag{2.38}$$

is made.

In summary, the worst case norm dynamics are given by the following coupled set of ordinary differential equations:

$$\frac{\partial|\hat{\rho}_k|}{\partial t} = -h|\hat{\rho}_{k+1}| + h|\hat{\rho}_{k-1}| + rf(k)|\hat{\rho}_k| - \delta_{1,k}r|\hat{\rho}_{\text{eq}}| \quad f(k) = k \text{ or } \sqrt{k} \tag{2.39}$$

Written in matrix form, this system of equations reads:

$$\frac{\partial|\rho|}{\partial t} = D|\rho| + r|\rho_{\text{eq}}|\boldsymbol{e}_1 \quad \text{where } D = \begin{pmatrix} r & -h & 0 & 0 & \cdots & 0 \\ h & 2r & -h & 0 & & \vdots \\ 0 & h & 3r & -h & & \\ 0 & 0 & h & 4r & & \\ \vdots & & & & \ddots & \vdots \\ 0 & \cdots & & & \cdots & nr \end{pmatrix}, \tag{2.40}$$

where we are abusing the terminology slightly by treating $|\rho|$ as a vector of subspace norms, i.e. $|\rho| = (|\hat{\rho}_1|, |\hat{\rho}_2|, \dots)^{\text{T}}$.

When relaxation is neglected, this system of equations has the formal solution $|\rho| = \exp(Dt)|\rho|_0$. Notice that D is traceless and skew symmetric, so $\exp(D)$ is an orthogonal matrix and the total squared norm of the density matrix is preserved, implying that our incompressible fluid analogy still holds.

[2] Read the operations absmax and absmin as returning the maximum and minimum absolute values respectively of a set.

Equation (2.40) gives an error estimate that is in a from that would be suitable for performing practical estimates very quickly. The dimension of D grows linearly with the size of the spin system so explicit diagonalisation is completely feasible, especially given it is tridiagonal in form. It is easy to picture the norm dynamics being computed from (2.40) automatically at the beginning of a simulation and some kind of warning or similar being printed if too much coherence is predicted to flow into spin order subspaces that are not being simulated. Despite this, it is worth persuading further simplifications, if only to build up an intuition for how the coherence behaves with respect to the spin order subspaces. This will be the subject of the next section.

2.6 Approximation via Partial Differential Equations

As noted, the behaviour of the coupled equations (2.27) could easily be integrated numerically as a set of n coupled ordinary differential equations. However, it is illustrative to assume that the spin system being considered is so large that there is effectively no upper bound on the spin order index, k. We will also make k a continuous variable in order to convert the problem into a single partial differential equation (this step is discussed in more detail in the JCP paper [1]).

The following replacements are made:

$$\begin{aligned} rf(k)|\hat{\rho}_k| &\rightarrow rf(k)\rho(k,t) \\ -|\hat{\rho}_{k+1}| + |\hat{\rho}_{k-1}| &\rightarrow 2\frac{\partial \rho(k,t)}{\partial k} \\ \delta_{1,k} r|\hat{\rho}_{\mathrm{eq}}| &\rightarrow r_0 \delta(k-1), \end{aligned} \tag{2.41}$$

where ρ is a continuous real valued function of two variables and δ is the Dirac delta function. The factor $|\hat{\rho}_{\mathrm{eq}}|$ is constant in time, so has been absorbed into r_0. After making these substitutions, the following differential equation is found:

$$\frac{\partial \rho(k,t)}{\partial t} = -2h\frac{\partial \rho(k,t)}{\partial k} + f(k) r \rho(k,t) + r_0 \delta(k-1). \tag{2.42}$$

These substitutions roughly correspond to the inverse of the procedure to find a numerical solution to a differential equation via finite differencing.

2.6.1 General Solutions

To keep the notation relatively free of clutter, in this section we will write $\beta = \frac{r}{2h(1+\alpha)}$.

For the case where $f(k) = k^\alpha$, the general steady state ($\frac{\partial \rho}{\partial t} = 0$) solution is:

$$\rho(k) = C\exp(-\beta k^{1+\alpha}) + \exp\left(\beta(1-k^{1+\alpha})\right)\frac{r_0}{2h}\Theta(k-1), \tag{2.43}$$

where Θ is a step function with $\Theta(k) = 1$ for $k > 0$ and $\Theta(k) = 0$ for $k < 0$ and C is a constant of integration. The condition that $\rho(k) = 0$ for all $k < 1$ implies that $C = 0$.

The fully general, time-dependent, solution is:

$$\rho(k,t) = \exp(\beta(1-k^{1+\alpha}))\frac{r_0}{2h}\Theta(k-1) + C\left(t - \frac{k}{2h}\right)\exp(-\beta k^{1+\alpha}), \tag{2.44}$$

where C is now an arbitrary function to be found by examining the boundary conditions. It would appear that the step function, Θ, may be simplified away through the assumption that $k \geq 1$, but manipulation of equation (2.44) to obtain particular solutions will reveal that differing particular solutions are found depending on the presence or absence of Θ.

The particular solution for the boundary condition $\rho(k,0) = F(k)$ is:

$$\rho(k,t) = \exp(\beta(1-k^{1+\alpha}))\frac{r_0}{2h}\chi_{[1,2ht+1]}(k) + F(k-2ht)\exp(\beta[(k-2ht)^{1+\alpha} - k^{1+\alpha}]). \tag{2.45}$$

The notation has been simplified by borrowing the χ function commonly used in real analysis. $\chi_{[a,b]}(x) = 1$ whenever $x \in [a,b]$ and is zero otherwise.

2.6.2 Interpretation of the Time-Dependent Solution

The time-dependent solution for the dynamics of the of the norm is applicable in cases where simulation time is short. As most spin dynamics simulations begin with the norm concentrated in the lowest spin orders, an appropriate choice of $F(k)$ in equation (2.45) is $\delta(k-1)$. Examining equation (2.45) closely, we see that if F is set to $\delta(k-1)$ the situation illustrated in figure 2.3 emerges. Initially all the area under the graph is concentrated in the delta function. As time progresses, the delta function slides to higher spin orders, loosing magnitude as it travels. As it moves, it leaves a static, exponentially decaying, envelope it it's wake. Physically, higher spin orders are not required in the simulation because the coherence does not have sufficient time to reach them.

2.6.3 Interpretation of the Steady State

For cases where the simulation time is long, the delta function slides to higher and higher spin orders and rapidly diminishes in magnitude, so the distribution of density matrix norm becomes effectively static:

$$\lim_{t\to\infty}\rho(k,t) = \exp(\beta(1-k^{1+\alpha}))\frac{r_0}{2h}\Theta(k-1). \tag{2.46}$$

The spin order pruning method of state space restriction corresponds to throwing away high spin order states, thus a reasonable condition to test would be:

$$\frac{\int_k^\infty \rho(k)dk}{\int_1^\infty \rho(k)dk} < \xi, \tag{2.47}$$

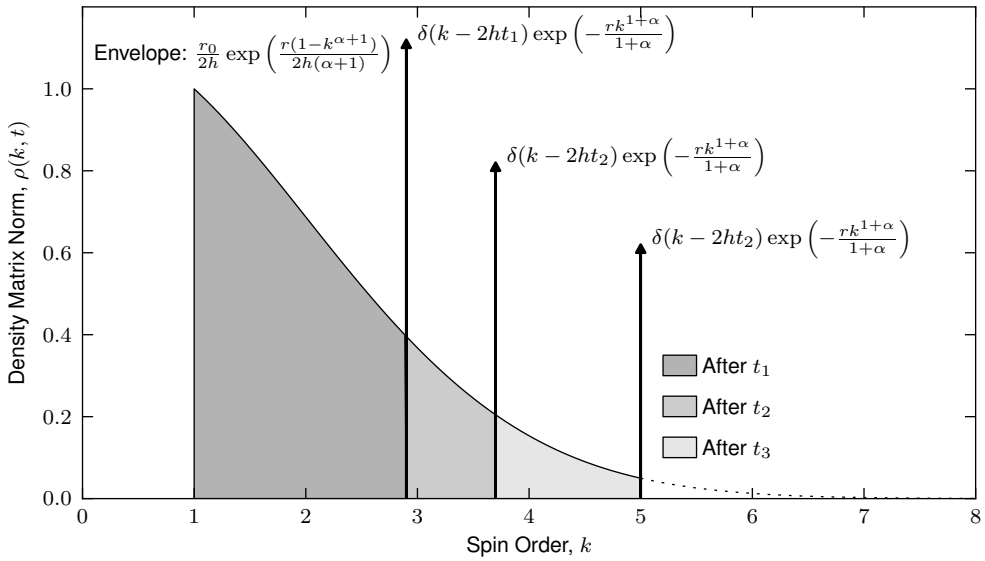


Figure 2.3 – Time dependent behaviour of the continuous approximation of the density matrix norm dynamics with $F(k-2ht) = \delta(k-2ht)$ shown at a particular values of t. Coherence begins concentrated into a delta function sitting at $k=1$. As time passes the delta function is swept towards higher spin orders, leaving behind a distribution corresponding to the steady state solution.

where ξ is a user-defined tolerance level between zero and one that corresponds to the fraction of coherence of which the simulation is allowed to lose track.

The integrals in equation (2.47) do not appear to be expressible in terms of well known elementary, or special functions for arbitrary α, but specific solutions can be found for the values of α that are relevant. For $\alpha = 1$:

$$\xi = \frac{\operatorname{erfc}\left(\frac{1}{2}\sqrt{\frac{r}{h}}\right)}{\operatorname{erfc}\left(\frac{1}{2}k\sqrt{\frac{r}{h}}\right)}, \tag{2.48}$$

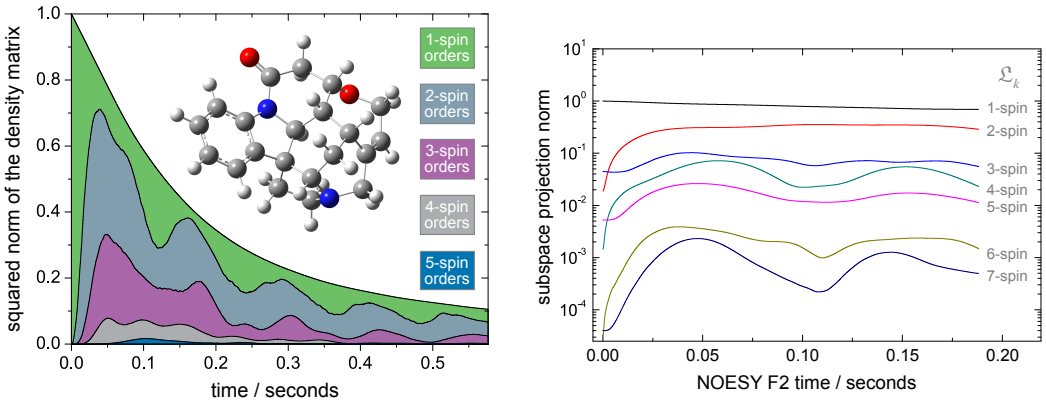
and for $\alpha = \frac{1}{2}$:

$$\xi = \frac{k\,\mathrm{Ei}_{1/3}\left(\frac{r}{3h}\right)}{\mathrm{Ei}_{1/3}\left(\frac{rk^{\frac{3}{2}}}{3h}\right)}, \tag{2.49}$$

where Ei is the exponential integral function, defined as $\mathrm{Ei}_n(x) = \int_1^x t^{-n}\exp(-xy)dx$. These expressions can be used to select a suitable k given a ξ.

2.7 The Single Bucket Assumption

An assumption that has been implicit in the above analysis is that, so long as a sufficiently small fraction of the total coherence sits in spin orders that are truncated, the simulation will be accurate. In reality, coherence is continually flowing up to higher spin orders and



(a) Pulse Acquire **(b)** NOSEY

Figure 2.4 – Dynamics of the squared density matrix norm of low spin order subspaces during two simulations performed on strychnine. In both cases distances and magnetic parameters were obtained using GIAO B3LYP/EPR-II [53–55] implemented in Gaussian 03. A full Bloch-Redfield-Wangsness relaxation superoperator which included the dipole-dipole, chemical shift anisotropy, and cross-correlation terms was used with an isotropic rotational diffusion correlation time of 200 ps. Plot (a) is a simulation of a pulse acquire experiment. The x-axis represents time since the application of the π pulse. The plot is stacked so that the sum of each colour adds up to the total density matrix norm. Plot (b) is a NOESY. "F2 time" is the time after the last pulse in which the FID is recorded.

back again, so the equilibrium is actually dynamic. In other words, we are assuming that, if coherence is returned to a lower spin order, it is returned in such a way that it does not cause errors in the observables (we are treating each spin order like a single homogeneous bucket). In reality, coherence may flow up to a higher spin order, undergo evolution, then flow back to a different part of the lower spin order—such an effect would be missed by a restricted simulation. Eventually, this backflow of "tainted" coherence may produce an observable effect.

This assumption is likely to be justified in all cases where the fraction of coherence above the spin order pruning cut-off is sufficiently small, because the rate at which coherence flows down from a higher spin order to a lower one is proportional to the amount of coherence at that level (equation 2.32) and, so, a small amount of coherence above the cut-off implies a slow flow of tainted coherence back to lower levels.

Formal analysis of this assumption has been left as further work. In addition, a method is suggested that is not susceptible to the single bucket assumption, but, unfortunately, it can only be applied during a simulation rather than before. Further details can be found in section 6.1.

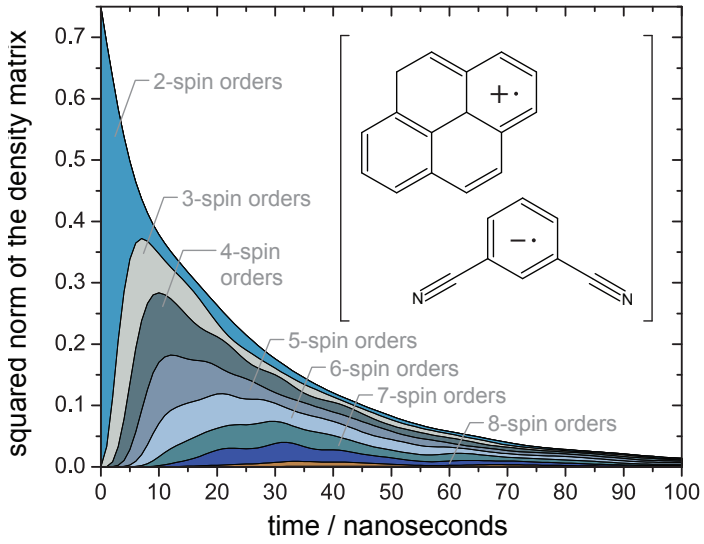


Figure 2.5 – Squared density matrix norm dynamics of the spin order subspaces during a simulation of a singlet-born pyrene-dicyanobenzene radical pair. See the supplementary information included with the JCP paper [1] for more technical details of the simulation. The Hore-Jones radical recombination kinetics superoperator was used with a singlet recombination rate of $4 \times 10^7 \mathrm{s}^{-1}$. All magnetic parameters were imported from a GIAO DFT B3LYP/EPR-II calculation; Bloch-Redfield-Wangsness relaxation theory (with H_0 containing the isotropic Zeeman and hyperfine interactions) was used to obtain the relaxation superoperator.

2.8 Numerical Evidence

There are already numerous examples of the Spinach library reproducing experimental spectra and the results of exact simulations as early as the original paper [6] so two or three further examples of correctly reproduced spectra would not appear to add much to the debate. Instead, a different approach was taken.

The situation is under consideration is essentially the modelling of a complicated mathematical system (exact simulations) with a more computationally favourable mathematical system (the restricted state space simulation). The validity of the exact simulation need not be called into question because if for any reason the exact simulation does not accurately reflect nature (for example, if Redfield theory is used in a situation where it does not apply) then both the exact and restricted simulation are invalid. The mistake was in selecting an incorrect model of reality, which restricted state space simulations can say nothing about.

If the analyses performed in this chapter were entirely mathematically sound and no assumptions were made, this section would be unnecessary. However, certain assumptions were indeed made (such as the decision to neglect cross-relaxation terms and assume certain properties are fulfilled by relaxation at higher spin orders). It is believed that the assumptions made are justified for systems that arise in practice, but it is still worthwhile to verify that spin systems do still behave as described in this chapter when the assumptions are removed. These assumptions are not made in either exact or restricted simulations, they were simply

made so that we were able to try and reason about what was going on in exact simulation. This suggests a methodology where we observe the dynamics of an exact (or as close to exact as feasible on current hardware) simulation to check that exact simulations behave as predicted. Coherence remaining in the low spin order subspaces in these simulations, as predicted, would provide supporting evidence that the mathematical analysis was on the correct track.

To this end, Spinach was modified to track the populations at the spin order subspaces through the time evolution of the system for several example simulations. A selection of systems are presented, simulated to the highest level of accuracy practical. All calculations were performed in Spinach version 1.0 [22]. Relaxation was modeled using Redfield theory. In Spinach, this is implemented using a method that avoids the diagonalisation step inherent in computing the eigenoperators (see equation 1.34) and is thus friendlier to state space restriction [23].

2.8.1 Strychnine

The first system chosen as an example was strychnine, a molecule containing 22 hydrogens. Zeeman, dipolar, and J-couplings were included in the Hamiltonian. These magnetic parameters were computed using Gaussian 03 [56] using density functional theory. The basis set was EPR-II [55] and the level of theory was B3LYP [53, 54] using gauge invariant atomic orbitals. The system was modeled with a field strength of 5.9 Tesla. Relaxation was modeled using Redfield theory assuming isotropic tumbling. This model is expected to be appropriate because strychnine is roughly spherical with no elongated axes (its structure can be seen in figure 2.4). This model of the dynamics contains only one adjustable parameter, the rotational diffusion correlation time, which specifies how quickly the molecules tumble. This was set to 200ps.

Both a simple pulse acquire and a nuclear Overhauser effect (NOSEY) experiment were simulated. The norm dynamics of both of these simulations during the free induction decay periods are shown in figure 2.4. Note that the pulse acquire plot is a stacked plot. The height of the top edge gives the total coherence in all the spin order subspaces and the smooth, approximately exponential decay results from the longitudinal and transverse relaxation of the system. In both the pulse acquire and the NOESY there is a general trend that the fraction of coherence in each spin order subspace drops as the index increases. This is particularly clear in the NOSEY; although there are some fluctuation, the 4^{th} and 5^{th} spin orders contain something in the order of magnitude of a hundredth of the total coherence and the 7^{th} spin order contains in the order of magnitude of a thousandth. Of the 22 spin

order subspaces, these results indicate that only the first 5 need to be used to track the dynamics of 99% of the coherence.

2.8.2 Pyrene-Dicyanobenzene

The next example chosen was the EPR radical pair pyrene-dicyanobenzene, simulated using the Hore-Jones radical recombination kinetics superoperator [57]. In this type of system, a pair of radicals are generated in the singlet state, separate, and after some time passes they may re-encounter each other. In the intervening time they may evolve from the singlet to the triplet state and the chances of a reaction occurring depend on whether the radical pair is in a singlet or triplet state at this time with the singlet state being more reactive. Among other factors, the ambient magnetic field affects this singlet-triplet mixing rate resulting in magnetically altered reaction yields (MARY effect). It is believed that such reactions play a role the ability of some birds to navigate by the Earth's magnetic field [57, 58]. The pyrene-dicyanobenzene radical pair was chosen for this simulation as the MARY effect was noted to be particularly strong.

Once again the magnetic parameters were computed using Gaussian 03 using density functional theory. The basis set was EPR-II and the level of theory was B3LYP using gauge invariant atomic orbitals. The system was modeled with a field strength of 0.33 Tesla.

The system relaxes relatively slowly, resulting in a large h/r ratio and, consequently, it would be expected that an accurate simulation would require more spin orders. The norm dynamics, shown in figure 2.5, illustrate that this is the case; acceptable accuracy requires 8 spin orders. This result approximately matches the prediction of the steady state model that 7 spin orders would be required (section 2.6.3), assuming typical organic radical values of 10 Gauss for hyperfine coupling, 10MHz for relaxation rate, and that no more than 1% of the norm should be lost. The pyrene-dicyanobenzene is an 11 spin system so a simulation restricted to 8 spin orders requires only 2285053 states[3] in the density matrix (assuming no other symmetries or simplifications were used) which is a significant improvement over the $2^{11} = 4194304$ states required for an exact simulation.

2.8.3 Random DNP Experiment

The final example selected was a solid state dynamic nuclear polarisation (DNP) experiment. Here, relaxation parameters were set manually based on the experience of the authors of [1]. Details and the resultant norm dynamics are given in the caption of figure 2.6.

[3]Note that this number is for spin order pruning only. For practical calculations, Spinach uses other techniques to shrink the state space. These were detailed in section 1.4.1.

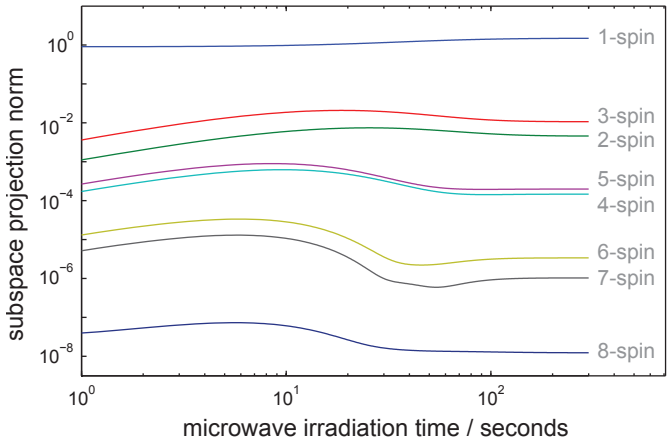


Figure 2.6 – Solid state DNP. 1 electron and 7 nuclei placed at random locations between 12Å and 18Å from the electron. The longitudinal relaxation rates were selected as 0.01Hz and the transverse was 10Hz.

Notice that this simulation is exact; it contained all 8 spin orders. Qualitatively, the distribution appears similar to figure 2.4; once again the ratios of the occupancy of the spin orders remain constant for much of the duration of the simulation, as predicted by the steady state model. Observing that the factional occupancy of 2^{nd} and 3^{rd} spin orders hovers around 10^{-2}, we observe that of the 65536 states in the simulation, 1789 from the 1^{st}, 2^{nd}, and 3^{rd} spin orders hold about 99.9% of the coherence.

2.9 Conclusions

When the original paper describing state space restriction was published [6], it potentially represented an important advance in the state of the art of computational spin dynamics. However, no formal analysis of why the algorithm worked was known, although qualitative arguments were presented. This situation has now changed; a mathematically rigorous analysis of the state space restriction algorithm has been presented here and in the JCP paper [1], and a condition, the h/r ratio, has been presented that is predictive of the minimum number of spin orders that need to be included for an accurate simulation.

Note that equation (2.45) contains an decaying exponential, which implies that, given a certain h/r, there will always be a sufficiently large spin order k such that the highest spin orders remain effectively unoccupied. This observation shows that, so long as there is some degree of spin relaxation present, spin dynamics should be regarded as an asymptotically, polynomially scaling problem unless perfect accuracy is required (and, if perfect accuracy is required, floating point rounding errors are likely to pose a significant problem first).

Chapter 3

Spin XML Graphical User Interface

3.1 Introduction

Perhaps because spin dynamics simulation packages had previously been limited to dealing with small numbers of spins, the problem of constructing and visualising these spin systems has received little interest. Common spin dynamics simulation programs such as EasySpin [59] in EPR, and Spinevolution [60] in NMR, use inputs based on text files. While constructing a system of five spins by hand in a plain text file is feasible, doing so for a two hundred spin system is likely to be both tedious and error prone, especially when no visual feedback is present. There have been previous attempts to tackle these problems, such as the SIMMOL visualisation package [61], associated with SIMPSON [62], but that package is limited to visualisation, with editing data still requiring the use of text files.

A new graphical user interface written in C++ for editing and visualising spin systems is presented in this chapter. This user interface is based around spin XML; a standard for storing the parameters making up the the spin Hamiltonian of spin systems that has recently been proposed based on extensive consultation with the spin dynamics community.

A key problem that needs to be solved by any potential format for storing spin Hamiltonians is finding a way to describe rank 2 tensors, and so methods to do so will be reviewed in section 3.2. The important subproblem of rotational representations is discussed in section 3.3. In particular, an algorithm is presented for the conversation of quaternions to Euler angles that appears stable even for small values of the β angle. The proposed spin XML format will then be described in depth in section 3.4. Section 3.5 covers the function specification of the GUI code; in other words, what the intended behaviour of the final program should be, without touching on how this behaviour is to be achieved. This section could potentially be adapted into user documentation.

The rest of the chapter is dedicated to technical details of the code and the decisions taken during development. The intended audience is, primarily, future maintainers of the GUI. Section 3.6 introduces and credits the main third party libraries upon which the software is built. Section 3.7 tackles input and output, especially the factorisation of the input problem into a stage specific to each format and a common stage. Section 3.8 describes the core data structures used to model the spin system in memory. Finally, section 3.9 describes the workings of the graphical layer. Build instructions for the software may be found in appendix D.

3.2 Rank 2 Tensor Representations

A fundamental problem in the representation of spin systems is that of the representation of rank-2 tensors. In section 1.2.3, it was noted that, for most spin systems, the Hamiltonian may be written in the following form:

$$\hat{H}_{\text{spin}} = \sum_{i=1}^{n} \hat{\boldsymbol{S}}_{\boldsymbol{i}} \cdot A_i \cdot \boldsymbol{x} + \sum_{i=1,j=1,i\neq j}^{n} \hat{\boldsymbol{S}}_{\boldsymbol{i}} \cdot B_{i,j} \cdot \hat{\boldsymbol{S}}_{\boldsymbol{j}} + \sum_{i=1}^{n} \hat{\boldsymbol{S}}_{\boldsymbol{i}} \cdot Q_i \cdot \hat{\boldsymbol{S}}_{\boldsymbol{i}}. \tag{3.1}$$

Notice that the complete, exact, spin Hamiltonian from a system of n spins may be specified in principle by specifying n linear interaction tensors (the A tensors), $\frac{n^2-n}{2}$ bilinear interaction tensors (the B tensors), and n quadratic interaction tensors (the Q tensors). The problem of representing the Hamiltonian has, therefore, been reduced to the problem of specifying rank 2, dimension 3 tensors.

Unfortunately, there are a multitude of conventions used in spin dynamics for the specification of such tensors, and a consensus does not seem likely to be reached in the foreseeable future. Five common conventions will be reviewed below.

3.2.1 Scalar and Matrix Representations

The most straightforward representation of a rank 2 tensor in 3 dimensions is a 3×3 matrix. An important special case is that of isotropic tensors (for example, J coupling tensors are usually taken to be isotropic), in which case the matrix is diagonal proportional to the identity. In this case, it may be summarised via a single real number: the constant of proportionality.

3.2.2 Eigenvalue Representations

If the matrix representing an interaction tensor is diagonalisable, then it may factorised via an eigendecomposition into the form $R^{-1}DR$, where R is invertible and D is diagonal. Spin XML's eigenvalue representation imposes the additional restriction that R be a special

orthogonal matrix, so that it can be represented as a rotation. This restriction is equivalent to the restriction that the tensor is symmetric.

The eigenvalue representation of a tensor is not unique, even in the cases where the eigenvalues are non-degenerate. Let $R^{-1}DR$ be an eigendecomposition of a matrix; $R^{-1}S^{-1}SDS^{-1}SR$ is another eigendecomposition so long as SDS^{-1} is diagonal. Using the result that the eigenvalues of a matrix are unique and, assuming they are not degenerate,[1] we see that the effects of S must be at most a permutation of the diagonal elements of D and thus S must be a member of the following set:

$$\left\{ \begin{pmatrix} a & & \\ & b & \\ & & c \end{pmatrix}, \begin{pmatrix} & a & \\ b & & \\ & & c \end{pmatrix}, \begin{pmatrix} a & & \\ & & b \\ & c & \end{pmatrix} \begin{pmatrix} & & a \\ & b & \\ c & & \end{pmatrix}, \begin{pmatrix} & a & \\ & & b \\ c & & \end{pmatrix}, \begin{pmatrix} b & & \\ & & a \\ & c & \end{pmatrix} \right\}. \quad (3.2)$$

for some arbitrary constants $a \neq 0$, $b \neq 0$, and $c \neq 0$. Furthermore, the requirement that R be orthogonal restricts each of the constants to ± 1. Thus, the eigenvalue representation is 48-fold degenerate. When computing an eigendecomposition from a matrix, there is no guarantee about which of the 48 possibilities the eigensolver will pick. The situation can be improved by correcting the determinant of R so that it is $+1$, but beyond this, either an arbitrary convention has to be invented or the degeneracy must be accepted.

3.2.3 Axiality and Rhombicity

Axiality and Rhombicity are an alternate method of presenting the eigenvalues of an interaction tensor that emphasise the deviation of the tensor from being isotropic. The representation of a general symmetric tensor requires a rotation and three parameters, A_{ax}, A_{rh}, and A_{iso}, although the isotropic part is often not quoted.

There are many Axiality and Rhombicity conventions in use in the literature. The definitions used in spin XML are as follows: the three eigenvalues of the tensor are rearranged in the following order:

$$A_{xx} \leq A_{yy} \leq A_{zz}, \quad (3.3)$$

and then the tensor rotation is adjusted correspondingly.

This done, the values of A_{ax}, A_{rh}, and A_{iso} are given by:

$$A_{\text{ax}} = 2A_{zz} - (A_{xx} + A_{yy}), \quad A_{\text{rh}} = A_{xx} - A_{yy}, \quad A_{\text{iso}} = \frac{1}{3}(A_{xx} + A_{yy} + A_{zz}). \quad (3.4)$$

3.2.4 Span and Skew

Span and skew, sometimes also called the Herzfeld-Berger convention (which appears to have been introduced in [63]), are similar to Axiality and Rhombicity in that they are another

[1] In the case of degenerate eigenvalues, there are an uncountably infinite number of eigendecompositions.

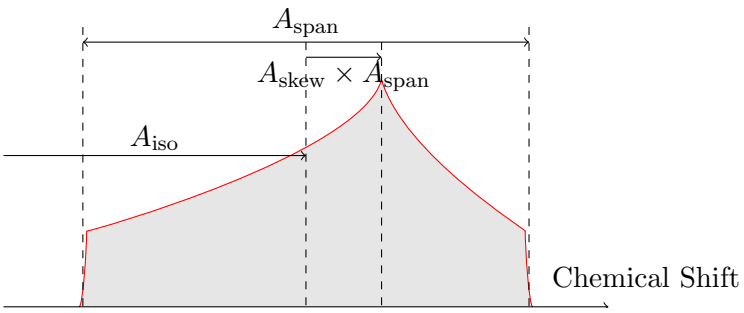


Figure 3.1 – Relationship between span, skew, and the powder pattern.

presentation of the eigenvalues of a tensor that, when paired with the isotropic value and the tensor orientation, allow the recovery of the full tensor.

The ordering of the eigenvalues is as follows:

$$|A_{xx}| < |A_{yy}| < |A_{zz}|. \tag{3.5}$$

Span and skew are defined as:

$$\begin{aligned} A_{\text{span}} &= A_{zz} - A_{xx} \\ A_{\text{skew}} &= \frac{3}{2}\frac{A_{\text{iso}} - A_{yy}}{A_{\text{span}}} \\ A_{\text{iso}} &= \frac{1}{3}(A_{xx} + A_{yy} + A_{zz}). \end{aligned} \tag{3.6}$$

Span and skew are of particular interest in relation to NMR spectra of powders because they relate directly to the shape of the observed spectrum.

In a powder, individual molecules do not rotate, so no averaging takes place. Instead the spectrum of a powder is simply the superposition of single crystal spectra taken over all orientations of the crystal (see figure 3.1). The Hamiltonian of a linear interaction described by the tensor A in a single crystal is given by:

$$H = \begin{pmatrix} 0 \\ 0 \\ B_0 \end{pmatrix} \begin{pmatrix} a_{xx} & a_{xy} & a_{xz} \\ a_{yx} & a_{yy} & a_{yz} \\ a_{zx} & a_{zy} & a_{zz} \end{pmatrix} \begin{pmatrix} \hat{S}_x \\ \hat{S}_y \\ \hat{S}_z \end{pmatrix}, \tag{3.7}$$

where B_0 is the external field, assumed to point in the z direction. When expanded and solved, the energy eigenvalues are found to be $-\frac{1}{2}B_0\sqrt{a_{zx}^2 + a_{zy}^2 + a_{zz}^2}$ and $\frac{1}{2}B_0\sqrt{a_{zx}^2 + a_{zy}^2 + a_{zz}^2}$, giving a resonance at $B_0\sqrt{a_{zx}^2 + a_{zy}^2 + a_{zz}^2}$. Different orientations of the crystal give different values of a_{zx}, a_{zy}, and a_{zz} and, hence, different resonances. The width of the powder pattern is the difference between the maximum and minimum resonances.

Let the tensor be written as $R(\alpha, \beta, \gamma)AR^{-1}(\alpha, \beta, \gamma)$, with R being rotation matrices parameterised over three Euler Angles. The resonance can be shown to be:

$$B_0\sqrt{A_{xx}^2 \sin^2\beta\cos^2\alpha + A_{yy}^2 \sin^2\alpha\sin^2\beta + A_{zz}^2\cos^2\beta}. \tag{3.8}$$

Notice that the γ rotation has disappeared. Its disappearance makes physical sense as the magnetic field appears as a vector quantity, so the initial rotation about the z-axis leaves it invariant. Expression (3.8) can be maximised and minimised as α and β vary by inspection, or by basic calculus. The maximum is at $\alpha = 0$, $\beta = 0$, and the minimum is at $\alpha = \frac{\pi}{2}$, $\beta = 0$ assuming $|A_{xx}| < |A_{yy}| < |A_{zz}|$, which shows that span is indeed the width of the powder pattern.

Proof that A_{skew} takes the role suggested by figure 3.1 is significantly harder and is outside of the scope of this document. See, for example, [64] for one treatment of this problem. In any case, span and skew only take these roles exactly for spins that do not interact. When other interactions are present, the shape of the spectrum changes, so, in practice, powder patterns are rarely assumed to have analytical shapes, and are instead calculated via integration over grids [65].

3.3 Rotations and their Representations

Many methods of representing rotations in 3D space in terms of real numbers have been invented over the years, each with advantages and disadvantages. There are a number of attributes that are favourable in rotational representations, but no one representation presented here possesses all of them.

Before looking at specific rotational representations, we will begin by describing the properties of an abstract rotation by considering its action on a 3D vector. All properties of rotations follow from this definition.

Definition 3.1. *The set of rotations, which will be called $SO(3)$, is the set whose members have an action on the space of 3D vectors that preserves length, $|v| = |Rv|$; the angles between vectors, $v_1 \cdot v_2 = (Rv_1) \cdot (Rv_2)$; and the exterior triple product, $v_1 \wedge v_2 \wedge v_3 = (Rv_1) \wedge (Rv_2) \wedge (Rv_3)$ (geometric algebra picture) or has determinant +1 (linear algebra picture) for all $v, v_1, v_2, v_3 \in \mathbb{R}^3$ and for all $R \in SO(3)$.*

From this definition, it will be possible to show that the rotations form a group. To see this, we begin by proving the following:

Theorem 3.2. *The action of a rotation is bijective.*

Proof. This can be shown by deriving a contradiction. Suppose there exist two distinct vectors, v_1 and v_2, for which $Rv_1 = Rv_2$. First, note that $|v_1| = |v_2|$, as rotations preserve the magnitude of vectors. As Rv_1 and Rv_2 are the same vector:

$$Rv_1 \cdot Rv_2 = |Rv_1||Rv_2| = |v_1||v_2| = |v_1|^2, \tag{3.9}$$

but, by definition, rotations preserve the dot product so that:

$$Rv_1 \cdot Rv_2 = v_1 \cdot v_2 = |v_1||v_2| \cos\theta = |v_1|^2 \cos\theta, \tag{3.10}$$

where θ is the angle between the vectors.

Equations (3.9) and (3.10) imply together that θ must be zero, but for v_1 and v_2 to be distinct and yet have the same magnitude, θ must be nonzero, which is a contradiction, thus $v_1 = v_2$. Hence rotations are injective.

By swapping each occurrence of v_1 with Rv_1, and v_2 with Rv_2, we can also establish the surjectivity property of rotations. Together, these imply bijectivity. □

Theorem 3.3. *The rotations form a group with the composition product, defined as:*

$$(R_1 R_2)v = R_1(R_2 v), \tag{3.11}$$

being the group product.

Proof. To show this, each of four properties of groups needs to be proven in turn.

1. **Closure** The composition of the actions of two rotations on $\mathbb{R}^3$ is again a rotation, because $|R_1 v| = |R_2(R_1 v)|$ and $R_1 v_1 \cdot R_1 v_2 = (R_2(R_1 v_1)) \cdot (R_2(R_1 v_2))$, as $R_1 v$, $R_1 v_2$ are vectors.

2. **Identity** The identity map on vectors, I, over $\mathbb{R}^3$ is a rotation and, as $(IR)v = I(Rv) = Rv$, the composition product has an identity.

3. **Associativity** The composition of functions is inherently associative.

4. **Inverse** This follows as a consequence of the action of rotations having an inverse (theorem 3.2).

□

These considerations of the properties of abstract rotations were used to inform decisions about the public interface of the `Orientation` class (section 3.8.1). While having a good picture of an abstract rotation is important, at some point representations in terms of real numbers (to be approximated via floating point numbers) need to be considered. The four stipulated by spin XML will now be reviewed.

In section 3.3.1, a method of recovering the Euler Angles describing a rotation is described that uses only the action of abstract rotations on vectors, rather than relying on any specific property of the format in which the rotation is specified. Consequently, this method could be implemented generically by only using the rotation-vector product of the `Orientation`

class (although an optimised version, dependent on the quaternion representation, is also presented).

Having worked out the properties of abstract rotations, we can now look at ways to represent them in terms of real numbers (and, ultimately, approximated via floating point arithmetic, so calculations may be performed on a computer). The following attributes are favourable:

Natural Products A good rotational representation has natural products that accurately represent the action and composition maps for abstract rotations. For example, the matrix-matrix product accurately represents composition.

Singularity Free Let μ be a metric over the rotations. Then, if $\mu(R_1, R_2) < \epsilon$, and if δ is the biggest difference in the corresponding real number parameters in the representations of R_1 and R_2, then $\epsilon \to 0$ implies $\delta \to 0$. Euler Angles provide a familiar example of a rotational parametrisation with a singularity, as will be seen in section 3.3.1.

Intuitive Although subjective, it is arguable that, for some rotational representations, it is easier to mentally picture the effect of the rotation based on the rotation parameters than for others representations.

3.3.1 Euler Angles

Euler angles are a set of three parameters corresponding to successive rotations about coordinate axes. Any ordered choice of three axes is a valid Euler angle convention, so long as no axis is immediately repeated. Consequently, there are six conventions involving all axes (the 3! permutations of XYZ) and six involving two axes (all of the form XYX, generated by choosing two distinct axes from three possibilities then assigning one axis to appear once and one axis to appear twice, giving 2 × (3 choose 2) possibilities). Furthermore each rotation might be specified to be left handed or right handed. Sensible conventions specify all left or all right handed rotations, but there is no mathematical reason why a mixed specification cannot exist, thus the number of possible conventions must be multiplied by 2^3 giving 72 Euler angle conventions. Finally, each convention may then be interpreted in the "active" sense or the "passive" sense, bringing the total count of Euler angle conventions up to 144.

The Euler angle convention used in spin XML is sometimes called *Varshalovich Convention B*, due to its appearance in *Varshalovich's* book [66, p. 21-22]. It is:

- Rotation through an angle of α about z-axis in the range $[0, 2\pi)$
- Rotation through an angle of β about y-axis in the range $[0, \pi)$

- Rotation through an angle of γ about z-axis in the range $[0, 2\pi)$

The rotations are active, that is they should be applied to the molecule and not the coordinate system.

3.3.2 Direction Cosine Matrices/Rotation Matrices

As the application of a rotation to a vector space is a linear operation in the sense that $R(\boldsymbol{v}_1 + \boldsymbol{v}_1) = R\boldsymbol{v}_1 + R\boldsymbol{v}_2$, it should come as no surprise that rotations have a matrix representation, as every linear map on a vector space has a matrix representation when a basis has been chosen [35].

Rotation matrices have very favourable computational properties: the composition product is modelled via the matrix-matrix product and the rotation action on vectors is modelled via the matrix-vector product. They are, however, somewhat unwieldy as they require the specification of nine real numbers that must conform to six constraint equations.

3.3.3 The Angle-Axis and Quaternion Parametrisation

The angle-axis and quaternion parametrisations of rotations are two closely related conventions for rotations, both of which require the storage of only a single, additional, real number over the Euler angles. Quaternions have the additional advantage of being easily composable via their multiplication operation and come equipped with a natural action on three-dimensional vectors. In contrast, the angle-axis parametrisation has the advantage of being highly intuitive, but lacks a both a straightforward composition product and a vector action. As the conversion between these two conventions is particularly simple, the angle-axis parametrisation can be thought of as the "public face" of the quaternion parametrisation.

The angle-axis parametrisation requires two parameters: $\boldsymbol{v}$, a normalised vector, and a, a scalar. The object is simply rotated by angle a around axis $\boldsymbol{v}$.

The set of quaternions are often introduced as extension of the complex numbers. In much the same way as the normalised complex number $e^{i\theta}$ encodes a 2D rotation, quaternions encode 3D rotations.

Mathematically, the quaternions are a real vector space over the basis elements 1, i, j, and k, where 1 is the multiplicative identity from the real number line with an additional bilinear product that is completely defined by:

$$i^2 = j^2 = k^2 = ijk = -1. \tag{3.12}$$

Remembering the definition of an algebra over a field from section 1.5, it is seen that the quaternions are yet another example of such an algebraic structure. The quaternion product is associative but not commutative [67, ch. 5].

Normalised quaternions are closely linked to the angle-axis parametrisation and conversion between the two conventions is particularly simple. These formula are relatively well-known, see for example [68, ch. 14]. They can also be derived from equation (1.55), which is noted in section 1.5.4.

An angle-axis parametrisation may be converted to a quaternion via the following conversion:

$$\begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos(a) \\ v_x \sin(a) \\ v_y \sin(a) \\ v_z \sin(a) \end{pmatrix}, \tag{3.13}$$

where a is the angle; $\boldsymbol{v}$ is the axis; and w, x, y, and z are the *real*, i, j, and k^{th} components of the quaternion respectively.

The axis can be easily recovered:

$$\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} x/\sin(a) \\ y/\sin(a) \\ z/\sin(a) \end{pmatrix}, \tag{3.14}$$

with the angle simply being $\arccos(w)$. As only normalised quaternions are used as rotation parametrisation, $\arccos(w)$ is always defined.

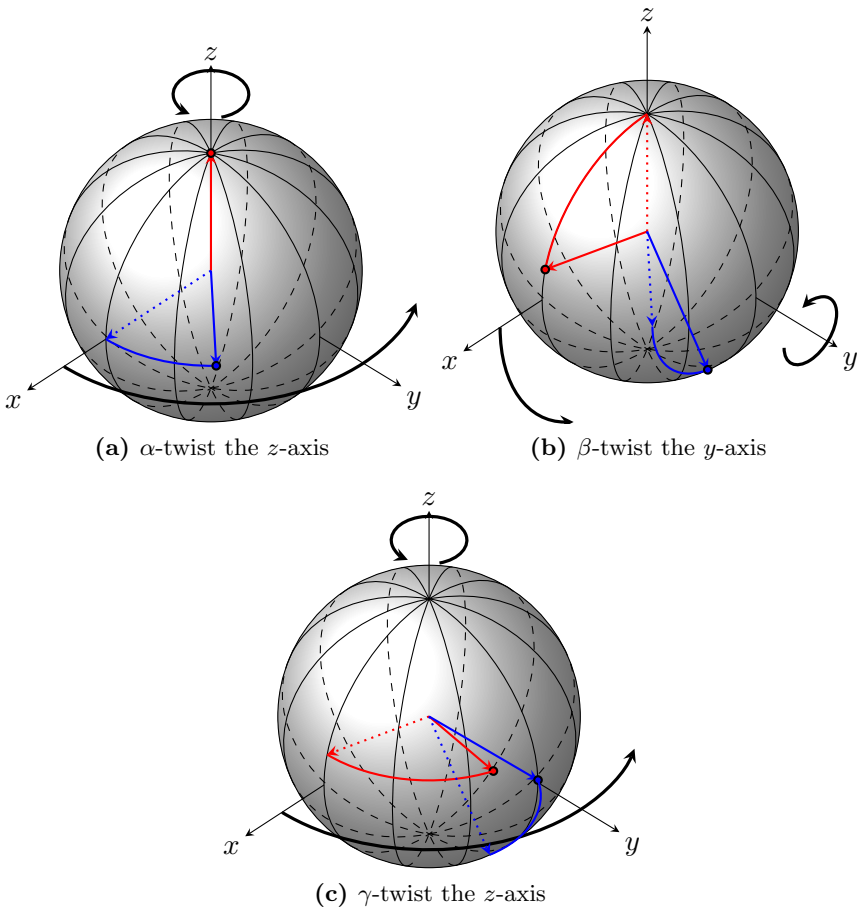
Quaternions are particularly useful whenever rotations need to be smoothly interpolated [69], as the linear interpolation between two quaternions, q_a and q_b, as a parameter t is varied from 0 to 1, is given by:

$$q_{\text{inter}} = q_a(q_a^{-1}q_b)^t. \tag{3.15}$$

The link between quaternions and the angle axis parametrisation becomes even more fundamental when looked at in the language of geometric algebra. Geometric algebra is yet another example of an algebra over a field with the bilinear product defined as $\boldsymbol{ab} = \boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \wedge \boldsymbol{b}$, where the "$\cdot$", or "inner product" is symmetric, and the "$\wedge$", or "exterior product" is antisymmetric.

The geometric algebra extends vector algebra by admitting the existence of additional mathematical objects beyond scalars and vectors, such as bivectors, trivectors, and so on. A bivector is formed via the exterior product of two vectors and can be interpreted as a directed area. There is only one basis bivector in two dimensions, which can be seen to behave isomorphically to the imaginary unit. The exponentiation of this unit gives the two dimensional rotor, $e^{i\theta}$, which is capable of rotating points via familiar complex multiplication.

Quaternions appear naturally as the exponentiation of a three-dimensional bivector (the magnitude of the bivector determines the angle, and the direction determines the axis of the rotation).



(a) α-twist the z-axis **(b)** β-twist the y-axis **(c)** γ-twist the z-axis

Figure 3.2 – Illustration of the ZYZ Euler angle rotation convention. By rotating special points beginning on the z- and x-axes (shown in red and blue respectively), the effects of the α, β and γ twists can be isolated. By (c), the entire rotation has been applied, but points that began on the z-axis (example shown in red) have only been affected by the β and γ angles. Consequently these can be read off from the final position of a test point. Using knowledge of these angles, the effects of the γ and β twists on the final position of the second special point (shown in blue) can be reversed, leaving it in the position it was in after the α twist was applied (a).

Unfortunately, a full discussion of geometric algebra is far outside of the scope of this document. Good reviews of this subject can be found in Vince's book [67] and the book of Dorst *et al.* [70].

3.3.4 The Conversion To Euler Angles

Let R be a rotation, $\boldsymbol{e}_x$ and $\boldsymbol{e}_z$ be unit vectors pointing in the x and z directions, and let R_α, R_β and R_γ be the three components of rotation as a ZYZ Euler Angle set.

Consider $\boldsymbol{e}_z$ under $R_\gamma R_\beta R_\alpha$. $\boldsymbol{e}_z$ is left invariant under R_α, rotated on the x-z plane by the R_β rotation, and rotated along a circle of constant latitude by the R_γ rotation. Clearly, the angles β and γ correspond to the spherical coordinates of $R_\gamma R_\beta \cdot \boldsymbol{e}_z$.

Now consider the action of the rotation on $\boldsymbol{e}_x$. R_α and R_β alone are sufficient to place $\boldsymbol{e}_x$ on an arbitrary point of the unit sphere, masking the effects of R_γ, but since R_β and R_γ are known, α must be the angular separation of the points $R_\alpha R_\beta R_\gamma \cdot \boldsymbol{e}_x$ and $R_\beta R_\gamma \cdot \boldsymbol{e}_x$. This process is illustrated in 3.2. In listing 3.1, the inverses of R_β and R_γ are computed and are used to carry $R_\alpha R_\beta R_\gamma \cdot \boldsymbol{e}_x$ back to $R_\alpha \cdot \boldsymbol{e}_x$ at which point α is found with the `atan2` function. Here, `atan2(y,x)` is taken to follow the conventions of the `atan2` from the C standard library.

```
1  input: rotation R
2  output: real alpha, real beta, real gamma
3
4  vector x_axis = (1,0,0)
5  vector z_axis = (0,0,1)
6
7  x_axis = transform(R,x_axis)
8  z_axis = transform(R,z_axis)
9
10 real gamma = atan2(z_axis[1],z_axis[0])
11 real beta  = atan2(sqrt(z_axis[0]^2 + z_axis[1]^2),z_axis[2])
12
13 rotation betaTwistInv  =
14         rotation_from_angle_axis(-beta ,vector(0,1,0))
15 rotation gammaTwistInv =
16          rotation_from_angle_axis(-gamma,vector(0,0,1))
17
18 x_axis = transform(betaTwistInv * gammaTwistInv,x_axis)
19
20 real alpha = atan2(x_axis[1],x_axis[0])
```

Listing 3.1 – C-like pseudocode for transforming a rotation into a set of Euler Angles

Listing 3.1 is written in a form in which its physical interpretation is clear, however, it is far from efficient. For example, if R is represented as a quaternion, then a total of 4 quaternion multiplications, 2 quaternion constructions, 3 additional trigonometric functions, and 2 additional multiplications must be performed, giving a total of 76 multiplications and 11 trigonometric function evaluations.[2]

Some of these multiplications are wasteful. Lines 7 and 8 will clearly involve multiplications by 0 and 1. Listing 3.1 may be simplified to the following set of equations:

[2]Quaternion multiplication requires 16 multiplications, while constructing a quaternion from an angle and an axis requires 4 more multiplications and 4 trigonometric functions to be evaluated.

$$\alpha = \arctan2(p, q) \tag{3.16}$$

$$\beta = \arctan2(2\sqrt{(a^2+d^2)(b^2+c^2)}, a^2 - b^2 - c^2 + d^2) \tag{3.17}$$

$$\gamma = \arctan2(-ab + cd, ac + bd) \tag{3.18}$$

where:

$$r = a^2 + b^2 - c^2 - d^2$$

$$s = 2(ad + bc)$$

$$p = s\cos\gamma - r\sin\gamma$$

$$q = r\cos\gamma\cos\beta + s\cos\beta\sin\gamma + (ac - bd)\sin\beta,$$

by evaluating listing 3.1 into a symbolic algebra package to obtain three expressions in terms of the components of the quaternion, $a + bi + cj + dk$ and then eliminating the repeated sub-expressions: r, s, p, and q. The resulting expression is given in equation (3.16). Only 7 trigonometric functions, 20 multiplications, and a square root are required in total.

There are also clearly no division operators in equation (3.16) and so no floating point exceptions are expected unless all of a, b, c, and d are simultaneously 0.

3.4 The Spin XML File Format

The aim of the spin XML file format was to provide a standard means with which to specify spin systems. The two main components of this problem were the specifications of the structure and the spin Hamiltonian, while a third, somewhat specialist, feature was the specification of reference frames. Of these, the specification of the spin Hamiltonian is perhaps the trickiest part. As was discussed in section 1.6.1, spin interactions have only one of three forms: linear ($\boldsymbol{B} \cdot \mathrm{A} \cdot \hat{\boldsymbol{S}}$), bilinear ($\hat{\boldsymbol{S}}_1 \cdot \mathrm{A} \cdot \hat{\boldsymbol{S}}_2$), and quadratic ($\hat{\boldsymbol{S}} \cdot \mathrm{A} \cdot \hat{\boldsymbol{S}}$). This classification of interaction terms is used to reduce the problem of specifying the general 3×3 A tensor with only a few more annotations being required. In Spinach GUI, this gross property of an interaction (whether the interaction is linear, bilinear, or quadratic) is known as the `form`.

There are two more properties of interactions that will be used when discussing the source code of Spinach GUI: `kind` and `storage`.[3] The `kind` property refers to the physical source of the interaction: for example, hyperfine couplings, J-couplings, dipolar couplings *etc*, all of which are bilinear from the point of view of the simulation. The `storage` property implies which particular in-memory representation of the interaction tensor is used (matrix,

[3]Because the words "`form`", "`storage`", and "`kind`" are commonly used in a general sense, they will by typeset in a `monospace` font when they are being used in the specific senses defined above.

eigenvalues, *etc*). Interactions can have differing `storage` properties while maintaining the same physical meaning.[4]

Spinach GUI is written to fully support the spin XML file format, where "fully support" means:

1. When a spin system is saved and later restored from that save, nothing relating to the spin system should be lost. In contrast, saving as, for example, an XYZ file format would result in all magnetic interaction parameters being lost.

2. If an arbitrary, valid spin XML file is loaded and then saved, then the resulting saved file should be identical up to an isomorphism.

3. It should be possible to generate (up to an isomorphism) every valid spin XML file through some sequence of commands in the GUI.

Two spin XML files are regarded as isomorphic if the only differences between them are: differences in whitespace;[5] contents of comment sections; or the order that spin, interaction, or reference frames appear; and the values of the indexes used for specifying relationships between them.

3.4.1 XML Schema Definition

Roughly speaking, the *Extensible Markup Language (XML) 1.1 (Second Edition)* published by the *World Wide Web Consortium* specifies a format for storing a hierarchy of elements and text nodes as well as other details, such as namespaces. Historically, XML was developed as a successor to *standard generalised markup language* (SGML, an earlier standard for text markup published by the *International Organisation for Standardisation*). The XML format is quite general; from the point of view of an XML parser, a document representing a spin system and a document representing a set of employee records are both equally valid so long as they correspond to the basic syntax of XML.

There are several standard schemes for further specifying the expected contents of an XML document. The first is called the *document type definition* (DTD). DTD is an older format that was initially developed for SGML and then adapted for use with XML. However, the current World Wide Web Consortium recommendation for the specification of schema is the *XML schema definition* (XSD). XSD serves a similar purpose for XML documents as BNF grammars (see [71, 72] for a discussion on BNF grammars) serve for character streams.

[4] The names `form`, `kind` and `storage` come from the names of the enumerations defined in `interaction.hpp`.

[5] XML, probably due to its history as a markup language, regards files that differ in whitespace as being semantically different.

A full description of the XSD specification is completely out of the scope of this document; interested readers are referred to the official World Wide Web Consortium recommendation, which is available on the their website at www.w3.org. However, spin XML only uses a small fraction of the feature set of the XSD specification and, so, a few of the important features will be reviewed here.

Consider the following example of an XML document:

```
1  <?xml version="1.1" encoding="UTF-8"?>
2  <spin_system>
3    <spin number="0" element="0" isotope="0" label="electron">
4        <coordinates x="0" y="0" z="0" reference_frame="0" />
5    </spin>
6    <interaction kind="g-tenser" units="unitless" spin_1="0">
7        <eigenvalues XX="1" YY="1" ZZ="3" />
8        <orientation>
9            <quaternion re="1" i="0" j="0" k="0" reference_frame="0" />
10       </orientation>
11   </interaction>
12 </spin_system>
```

The first line is the XML declaration that simply asserts that the document should indeed be parsed as an XML document and specifies a few other details, such as the encoding.

The XML declaration is followed by a hierarchy of XML nodes that encode the document. From the point of view of an XSD schema, each element has a type against which its attributes and child nodes are validated. In XSD, types may be simple or complex, where elements with simple types should contain, at most, a single text node in the data section, while a complex type may possess attributes and a sub-hierarchy of child nodes. For example `<scalar>0.1</scalar>` is of simple type, but the `orientation` element in:

```
1 <orientation>
2     <quaternion re="1" i="0" j="0" k="0" reference_frame="0" />
3 </orientation>
```

is of complex type because it has a child element. The `quaternion` element is also of complex type because it has attributes. Simple types must be:

- one of the build-in types listed in the W3C XSD recommendation,
- a restriction of an existing simple type,
- a whitespace delimited list of simple types or,
- a union of simple types.

Complex types are built from simple types in a relatively straightforward manner, for example, part of the specification of the `orientation` complex type is:

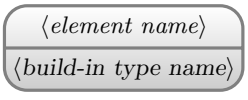
```
1 ...
2 <xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org
    /2001/XMLSchema">
3 ...
4 <xs:complexType name="orientation">
5   <xs:choice minOccurs="1" maxOccurs="1">
6     ...
7     <xs:element name="dcm" type="matrix" />
8
9     <xs:element name="quaternion">
10      <xs:complexType>
11        <xs:attribute name="re" type="xs:double" use="required" />
12        <xs:attribute name="i" type="xs:double" use="required" />
13        <xs:attribute name="j" type="xs:double" use="required" />
14        <xs:attribute name="k" type="xs:double" use="required" />
15        <xs:attribute name="reference_frame" type="xs:integer" use="
              required" />
16      </xs:complexType>
17    </xs:element>
18
19  </xs:choice>
20 </xs:complexType>
```

The specification declares a new complex type on line 4 with the name "`orientation`." Line 5 states that exactly one element from the following list of types should appear. Line 7 states that one of the options has the tag name `dcm` and is of type `matrix`, which is a complex type that is defined elsewhere. Like line 7, line 9 declares a new element with the name `quaternion`, but here the complex type is an anonymous type, as the type of the element `quaternion` is not needed elsewhere and so is unnamed.

3.4.2 XML Schema Diagrams

In this chapter, the descriptions of the overall structure of the spin XML schema will be supplemented by visual representations. These diagrams will have a precise and unambiguous meaning that can be related back to the spin XML schema, but the canonical source of information is still the actual XSD file, which can be found in appendix C.

An element whose type is a built-in simple type will be represented by a grey box:

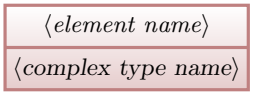


In spin XML, the majority of elements have an anonymous type. For the purposes of this document, it will be assumed that the *type name* of an `element` with an anonymous type is the same as the element name.[6] An element with an anonymous complex type will be

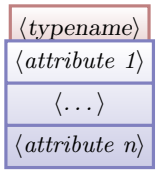
[6] To distinguish between the name of the element and the name of the type, *slanted font* will be used when talking about the type and `monospace` font will be used when talking about the name.

represented as:

Elements with a named complex type will be represented as:



with the element name in the upper part of the box and the type name in the lower part of the box. Attributes of a complex type will be listed below by the definition of the complex type:



In XML, a tag may contain a hierarchy of tags that appear between the opening tag and the closing tag. XSD provides a means for valid hierarchies to be specified for a given type of tag. As each inner tag on the first layer of the hierarchy has its own definite XSD type that determines the hierarchy are deeper levels, the only information required to determine all valid hierarchies is a specification of the types of the tags on the first layer. This specification problem corresponds to the classic problem of the specification of a formal language. The role of the alphabet is taken by the set of all XSD types that are declared in the XSD specification (see [72] for more information on formal languages).

There are several standard conventions for representing formal languages, for example, BNF grammars [71, 72]. In this work, a second, more visual, method will be used that is sometimes called the "railroad diagram" or "syntax diagram". An example of such a diagram is given here:

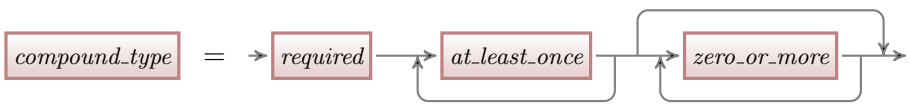


Figure 3.3 – An example of a "railroad" or "syntax" diagram.

For brevity, the type ***r**equired* will be called R, *at_least_**o**nce* will be called O, and ***z**ero_or_more* will be called Z. Strings that might potentially belong to the language can be formed from this alphabet. For example, $RROZOR$, $ZZOR$ and $ROZZZ$ are strings (the first two are not members of the language but the last one is). A particular string can be established to belong to the language described by a railroad diagram via following steps:

- Begin at the left hand side. Take the first letter of the string as the "current letter."
- If the current letter is reachable from the current position by following an arrow, then that letter is accepted, otherwise the string is rejected. Assuming the string wasn't rejected, move to where the arrow points and take the next letter of the string to be the current letter. Repeat this step until all letters in the string have been accepted.
- When every letter in the string is accepted, the string is accepted if the right hand side is reachable by an arrow.

The example language described in figure 3.3 can be seen to be the set of all strings beginning with exactly one R, followed by one or more O, and followed by zero or more Z. R is required because it is not possible to go from the left hand side to the right hand side without passing through R. After passing through R, the only place to go next is through O and, so, all strings in the language begin RO. After leaving O, there is a choice: go back and pass through O a second time, move onto Z, or skip to the end. There is a similar choice after Z has been passed through.

The question of whether these diagrams would be sufficient to describe the expected sub-tags of all possible XSD types will not be considered, as these diagrams are sufficient to describe the XSD types appearing in spin XML.

3.4.3 The Spin XML XSD

The top level element of spin XML is always of the type *spin_system* (see figure 3.4), which is composed of a sequence of zero or more spin elements, zero or more interaction elements, and zero or more reference frame elements.

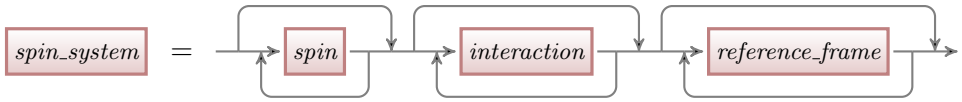


Figure 3.4 – The anonymous complex type of the spin_system element.

3.4.3.1 Reference Frames

Because all components of vectors and tensors may appear in the spin XML file transformed into reference frames, reference frames will be covered first. Formally, the spin system described by spin XML corresponds to a physical molecule that resides in $\mathbb{R}^3$. A frame is chosen as canonical; this will be called the "molecular frame."[7] In spin XML, the molecular

[7]Components of vectors and tensors quoted in this reference frame will be written unprimed (like $\boldsymbol{x}$), while vectors and tensors in other frames will be written primed (like $\boldsymbol{x}'$).

frame is implicitly defined as frame 0. It is an error to explicitly define a frame 0 in a spin XML file.

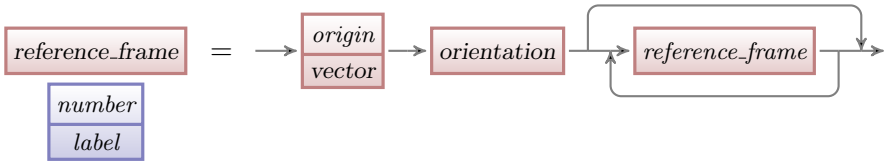


Figure 3.5 – The *reference frame* type

The *reference_frame* type is illustrated in 3.5. The origin and orientation elements represent a vector, $\boldsymbol{v}$, and a rotation, R, that defines the affine transformation as the action: "rotate a copy of the coordinate axes by R then translate them by $\boldsymbol{v}$ *in the old reference frame*." In other words, the change-of-frame isometry should be viewed as a *passive transformation*.

A subtle point to remember is that direction vectors and position vectors transform differently under affine transformations. If $\boldsymbol{x}'_t$ is a vector in the transformed frame (as would be found in a spin XML file), then its molecular frame components are recovered by:

$$\boldsymbol{x}_t = R^{-1}\boldsymbol{x}'_t, \tag{3.19}$$

under the affine transformation $(R, \boldsymbol{v})$, because vectors correspond to a quantity with a magnitude and direction, but are not attached anywhere. On the other hand, if $\boldsymbol{x}'_p$ is a position vector, then it transforms differently under the same affine transformation:

$$\boldsymbol{x}_p = R^{-1}(\boldsymbol{x}'_p - \boldsymbol{v}), \tag{3.20}$$

because position vectors can be thought of as being attached to the origin.

Note that the definition of *reference_frame* is recursive, in that each reference frame may have zero or more child frames. If a *reference_frame* element appears within another reference frame, then the two isometries are composed and any quantities specified in the child reference frame are mapped to the top level molecular frame by the composed transformation.

Suppose $(R_1, \boldsymbol{v}_1)$ is the parent reference frame and $(R_2, \boldsymbol{v}_2)$ is the child reference frame in the XML file, then the composed frame is $(R_2R_1, \boldsymbol{v}_2 + R_2\boldsymbol{v}_1)$, which can be seen by observing its effects on a primed position vector:

$$\begin{aligned} \boldsymbol{x}_p &= {R_1}^{-1}({R_2}^{-1}(\boldsymbol{x}'_p - \boldsymbol{v}_2) - \boldsymbol{v}_1) \\ &= {R_1}^{-1}{R_2}^{-1}\boldsymbol{x}'_p - {R_1}^{-1}{R_2}^{-1}\boldsymbol{v}_2 - {R_1}^{-1}\boldsymbol{v}_1 \\ &= (R_2R_1)^{-1}(\boldsymbol{x}'_p - \boldsymbol{v}_2 - R_2\boldsymbol{v}_1). \end{aligned} \tag{3.21}$$

3.4.3.2 Spin Elements

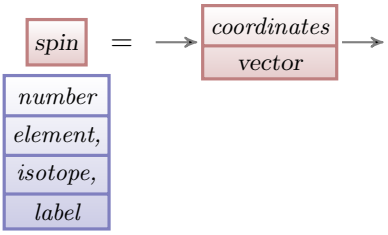


Figure 3.6 – Spin complex type.

The type of `spin` elements is relatively simple (figure 3.6). `element` is of type *nonNegativeInteger*[8] corresponding to the atomic number of the element. `element` $= 0$ represents an electron.[9] `isotope` is also a *nonNegativeInteger* corresponding to the mass number of the isotope. `number` is of type *integer* is simply a numeric identifier and `label` is a human-readable name for the spin. The coordinates of *vector* should be interpreted as a position vector and transformed according to equation (3.20) when its `reference_frame` attribute is set $\neq 0$.

3.4.3.3 Interactions

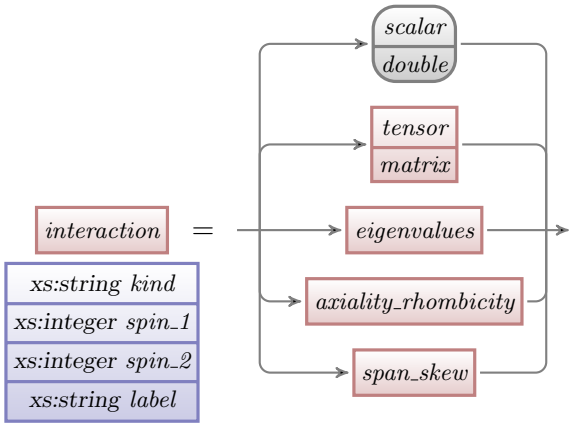


Figure 3.7 – Interaction complex type.

The *interaction* type (shown in figure 3.7) essentially represents a rank 2 tensor; it is built from the union of five different representation types of tensors: *double* (attribute name `scalar`), *tensor*, *eigenvalues*, *axiality_rhombicity* and *span_skew*. Of these, only the last

[8] *nonNegativeInteger* and *integer* are simple XSD types specified as part of the XSD standard.

[9] The format currently does not support spin systems containing particles other than protons, neutrons and electrons. Such systems have been the subject of magnetic resonance studies in the past. See in particular muon spin resonance (μSR) where muonium (an isotope of hydrogen with a μ^+ particle in place of a proton) is inserted into a chemical compound and then subjected to a resonance experiment. μSR was first observed by Roduner *et al.* [73].

token	Meaning	spin_2	See section	Unit
hfc	Hyper-fine coupling	required, $\neq$ spin_1	1.6.3.2	MHz
shielding	Chemical shielding	illegal	1.6.2	Unitless
quadrupolar	Quadrupolar	required, = spin_1	1.6.5	MHz
scalar	J-coupling	required, $\neq$ spin_1	1.6.4	Hz
dipolar	Dipolar	required, $\neq$ spin_1		MHz
g-tensor	g-tensor	illegal	1.6.6	Unitless
zfs	Zero Field Splitting	required, = spin_1	1.6.6	MHz
exchange	exchange	required, $\neq$ spin_1		MHz
custom	N/A	optional	N/A	MHz

Table 3.1 – Interaction `kind`s and their representations in spin XML. The "spin_2" column shows what the status of that XML attribute should be depending on the value of `kind`. "Required" indicates it should be present, "illegal" indicates it should be absent. Also indicated is if should be or must not be set equal to `spin_2` (the case `spin_1` = `spin_2` indicates a quadratic interaction, `spin_1` $\neq$ `spin_2` indicates a bilinear interaction).

four are general enough that they can represent arbitrary symmetric tensors, while the scalar representation is restricted to representing isotropic tensors. In addition to its children, the *interaction* type has the following attached attributes:

`kind` This attribute, corresponding to the `kind` enum within Spinach GUI's code, provides a hint to the physical origin of the interaction. Valid kinds are given in first column of table 3.1.

`label` A human-readable name for the interaction.

`spin_1` and `spin_2` These are keys corresponding to spins. `spin_2` is listed as optional in the schema and it is used to determine the `form` of the interaction. If it is absent, then a linear interaction is indicated. If it is present, and equal to `spin_1`, then the interaction is quadrupolar, otherwise the interaction is bilinear and links `spin_1` and `spin_2`. The `form` must match the `kind` as indicated in table 3.1, otherwise the file is not valid.

3.4.3.4 Orientations

The *orientation* type is illustrated in figure 3.8. Like the *vector* type, it has a context-dependent mathematical interpretation. It may represent a physical orientation in space, or a rotation (when used in reference frames). When interpreted as an orientation, an *orientation* type should be thought of as a physical, frame-independent quantity, while the rotation interpretation is better thought of as a process.

The *orientation* type is defined as a union of the *euler_angles*, *angle_axis*, *quaternion* and *matrix* types, all of which carry reference frame information, but specify their rotation parts

in four different manners. The vector contained within the *angle_axis* type is a plain vector and transforms according to equation (3.19). In the spin XML format, Euler angles are specified in the ZYZ active convention commonly used in spin dynamics (this convention is often called *Varshalovich B* because of Euler Angles are specified in this way in Varshalovich's book [66]). Such a rotation may be applied to a vector as follows: 1) rotate the vector about the z-axis by angle of α 2) rotate the vector about the y-axis by an angle of beta β 3) rotate the vector about the z-axis again by an angle of γ. All rotations are right handed.

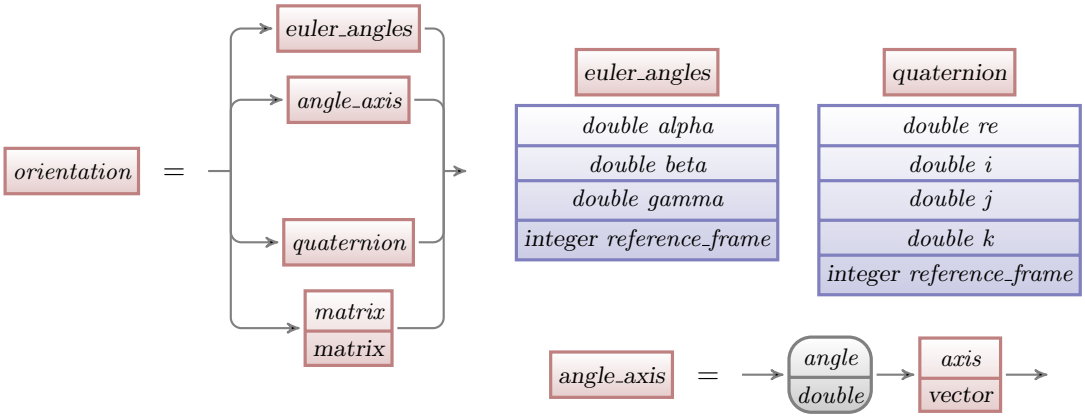


Figure 3.8 – Orientation complex type.

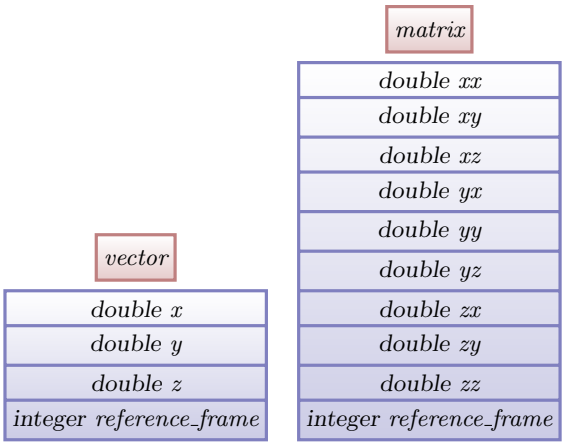
3.4.3.5 Vectors and Matrices

The *vector* type, illustrated in figure 3.9, carries a 3-tuple of *double*s, `x`, `y`, and `z`, and an *integer* `reference_frame`. Depending on the context where *vector* appears, this type may be interpreted as a vector, or as a position vector. When interpreted as a position vector, the attributes simply correspond to the three components of the vector part, while the position part of the tangent vector sits at the origin of the reference frame. The matrix type is similar, though there is no position-vector/pure vector ambiguity. If a matrix appears with a `reference_frame` that is not zero, then it is assumed to be written in the specified reference frame and needs to be rotated back into the lab frame before use.

3.4.4 Relational Level Verification

An XML file conforming to the spin XML schema may still fail to be a valid spin XML file. The following additional consistency checks need to be performed:

- The `number` attribute of every `spin` element must be unique.
- The `spin_1` and (if present) `spin_2` attributes of every `interaction` element must represent valid spins.



(a) The *vector* complex type

(b) The *matrix* complex type

Figure 3.9 – The vector and matrix named complex types.

- All `number` attributes of all `reference_frame` elements must be unique and nonzero.
- All `reference_frame` attributes must represent valid reference frames or be 0.
- Every interaction, unless it is of custom `kind` interaction, should have its `form` checked against its `kind` according to table 3.1.

3.5 Functional Specification

The following section is the *functional specification* of the GUI, that is, it will describe the intended behaviour of the GUI without explaining how this described behaviour will be achieved. That discussion will be deferred until sections 3.7, 3.8, and 3.9.

3.5.1 Application Startup

When the Spinach GUI binary is executed, the user is presented with the main window (section 3.5.2). Spinach GUI checks for the existence of the file `data/isotopes.dat` in the current working directory and tries to parse it. If it fails to parse the file or the file is missing, an error message is shown and Spinach GUI exits.

The `isotopes.dat` is expected to be laid out in simple, column-based, formation. All input between a `#` character and the end of a line is treated as a comment. Tokens are separated by white space, with the exception of the new line which is interpreted as a ⟨end of line⟩ token. Once comments are removed, the file is expected to conform to a

grammar which has the following BNF-style production rules:[10]

$$\begin{aligned} \textit{file} &::= (\textit{line}\,\langle\text{end of line}\rangle)^* \\ \textit{line} &::= \langle\text{blank}\rangle \mid \langle\text{integer}\rangle\,\langle\text{integer}\rangle\,\langle\text{integer}\rangle\,\langle\text{float}\rangle \end{aligned} \tag{3.22}$$

with ⟨integer⟩, ⟨float⟩, and ⟨end of line⟩ being the terminals and *file* being the start symbol. Details of BNF grammars may be found in [71, p. 42].

⟨integer⟩ is formatted as a standard base ten number; ⟨float⟩ is a floating point double precision number, formatted as expected by the function `scanf` from the C standard library [74]; and ⟨blank⟩ is empty production that produces nothing (this rule permits the file to contain blank lines). The three integers are interpreted as atomic number, mass number, and multiplicity, respectively. The floating point number is interpreted as the isotope's gyromagnetic ratio (units $10^6 s^{-1} T^{-1}$). The gyromagnetic ratios in the default `isotopes.dat` file provided with Spinach GUI are taken from Stone's paper [75].

3.5.1.1 The Current Selection and the Active Frame

In line with countless other graphical applications, Spinach GUI has a concept of an application wide selection upon which certain actions act. At any time, zero or more spins may be in the selection. In addition to the selection, at any time, exactly one reference frame is the active reference frame. All vector and tensor components are shown in terms of the active reference frame.

3.5.2 Main Window

If initialisation of Spinach GUI is successful, the first thing the user sees is the main window (a screenshot of which is given in figure 3.10). The main window contains a menu bar, a toolbar, and space for several panels. These panels can be shown, hidden, rearranged, and detached from the main window. Each panel serves a separate purpose: the 3D display panel visualises the spin system (section 3.5.4), the tensor visualisation panel displays options for changing the way interactions are visualised (section 3.5.3), and the spin grid panel displays the spins in the spin system in a tabular format for easier editing of the parameters in bulk (section 3.5.5). Furthermore the interaction editor (section 3.5.7) allows interactions to be edited and the reference frame tree (section 3.5.6) displays the hierarchy of defined reference frames that allows the user to change the active reference frame.

The user can use the menu and toolbar to interact with the Spinach GUI in various manners. Every tool in the toolbar has a corresponding entry in the menu (through due to

[10] The convention adopted here is to write terminals in angle brackets and nonterminals in slanted font. The "*" symbol is the zero-or-more repetition symbol. Parentheses have their usual meaning of determining precedence.

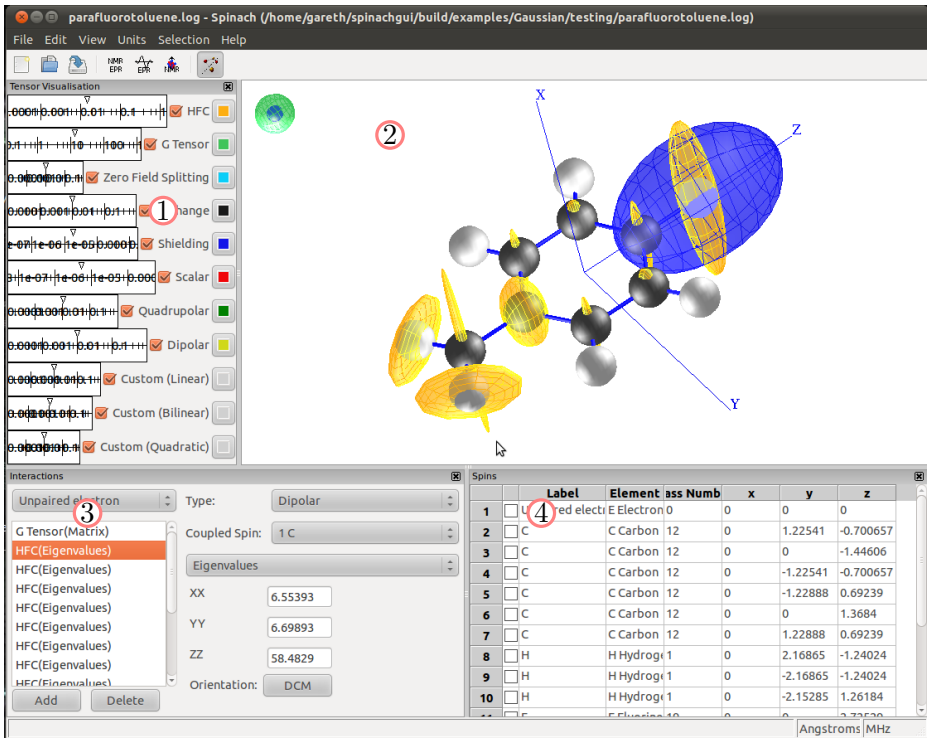


Figure 3.10 – Screenshot of the GUI main window with an example file loaded. ① Panel controlling details of the visualisation of the interactions, such as their colour and size; ② 3D display in which the view may be rotated, translated and zoomed with the left, right mouse buttons and wheel respectively; ③ interaction editor; ④ table of spins in the spin system in which all fields are editable.

limited space in the toolbar, the converse is not true), so only the actions in the menu bar will be documented here. They are:

File→New Clears all spins and interactions. If there are unsaved changes then the user is asked if they want to save those changes.

File→Open An operating system-dependent file open dialogue appears. If the user selects a valid file, then all spins and interactions are cleared from memory and the selected file is loaded. If the user has unsaved changes, they are given a chance to cancel.

File→Save If there is an open file, saves the current spin system to the open file without prompting. If there is no opened file, has the same effect as **Save As**.

File→Save As An operating system-dependent save dialogue appears. If the user enters a valid file name, the current spin system is saved to that file in the specified format. The current file is set to the file name of the file specified. If saving would overwrite an existing file, the user is given a chance to cancel.

File→Export to EasySpin... Opens the EasySpin export dialogue (see section 3.5.9).

File→Exit Quits the program. If the user has unsaved data, they are given a chance to save or cancel the quit.

Edit→Make Selection Isotropic Makes the interaction tensors of any selected spins isotropic. The representation is set to scalar and the value is set to the average of the tensor eigenvalues. This operation will most likely be useful for preparing input for rotationally averaged spectra, such as liquid state NMR and EPR.

View→Wire Frame Mode A wire frame representation of the 3D model is displayed. The 3D display updates immediately.

View→Solid A solid representation of the 3D model is displayed. The display updates immediately. This mode is the default.

View→Set EPR/NMR/General Mode Sets which interaction `kind`s from table 3.1 are shown or hidden. "EPR mode" shows the following `kind`s: HFC, g-tensor, zero field splitting, exchange, quadrupolar, dipolar, and custom interactions and hides others. "NMR mode" shows the shielding, scalar, quadrupolar, dipolar and custom interactions and hides others. "Mixed mode" shows all interaction `kind`s.

View→Toggle bonds Shows/hides bonds.

View→"Panel Name" A group of four similar options, which "Panel Name" being one of "Grid", "Interaction Tensor Visualisation", "Interaction Editor" or "Reference Frames." In each case, shows or Hides the appropriate panel. A tick is shown next to the menu item when the panel is visible.

View→Hide Selected Interactions The current selection is added to the group of spins for which interactions are not shown.

View→Unhide All All interactions are shown again.

View→Draw Interactions as Ellipsoids Interactions are shown as ellipsoids. This is the default option.

View→Draw Interactions as Axes Interactions are drawn as sets of axes drawn lined up with the tensor eigenframe and scaled by the eigenvalues.

In addition to the above fixed menus, there is a "selection" menu. Its content is dynamic and contains one entry per element present in the molecule. Clicking an element in this menu selects all nuclei of that element. For example, to suppress interactions on carbon atoms, a user could click "Selection→Carbon," followed by clicking "View→Hide Selected Interactions".

3.5.3 Tensor Visualisation Panel

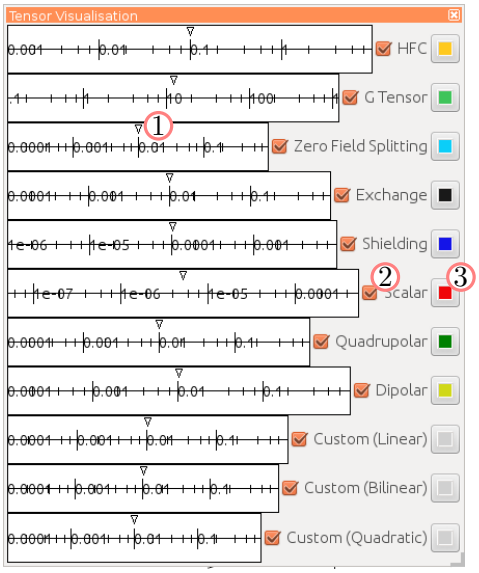


Figure 3.11 – The interaction tensor visualisation panel. The size of the displayed tensor, (linear/quadratic) or the cut-off of the conjoining line (bilinear) may be adjusted using the logarithmic sliders (marked ①). The checkboxes (marked ②) can be used to switch visualisation of each type of interaction on or off. The colour used to display each interaction can be set using the colour selectors (marked ③).

The tensor visualisation panel (illustrated in figure 3.11) is located by default to the right. It can be seen marked as ① in figure 3.10. It provides fine-grained control over the visualisation of interaction tensors in the 3D display. For each `kind` of interaction, three controls are displayed: a check box, a slider, and a coloured button. The check box controls whether interactions of a particular `kind` are shown or hidden, while the button controls the colour used to display them. The slider controls the scaling of the interactions in a manner dependent on the interaction's `form`.

Even interactions of the same `kind` may vary greatly in strength from molecule to molecule, so a slider is provided for adjusting the visualisation of those interactions so that they can be displayed comfortably. The number on the slider is simply a scaling factor applied during the drawing phase and has no physical significance or unit.

For interactions of linear and quadratic `form`, the slider controls the sizes of the ellipses drawn, but for bilinear interactions it controls the width of the interaction drawn. Two cutoff values, `minCutoff` and `maxCutoff`, are adjusted. For each bilinear interaction, the norm is calculated and used as a parameter to represent the strength of that interaction. If the norm is smaller than `minCutoff`, then the interaction is not drawn. If it is larger than `maxCutoff` then the interaction is drawn at a visually appealing maximum width, otherwise the interaction is scaled by the following factor:

$$\frac{\texttt{parameters} - \texttt{minCutoff}}{\texttt{maxCutoff} - \texttt{minCutoff}} \tag{3.23}$$

which can takes values between 0 and 1.

3.5.4 3D Display

A representation of the molecule being edited is displayed in the 3D display (marked as ② in figure 3.10). Left clicking a spin sets the selection to that spin. Left clicking while holding *shift* adds that spin to the currently selected group of spins if the spin is not selected, otherwise it or removes it from the currently selected group. Right clicking brings up the context menu for that spin (the context menu is covered in section 3.5.1.1). Moving the mouse over a spin will highlight that spin, suggesting to the user that the spin is clickable.

The display can be zoomed in or out by moving the mouse wheel upwards or downwards respectively. Left clicking and dragging rotates the molecule while right clicking and dragging translates the molecule in a direction perpendicular to the direction of the camera vector. Once a drag operation is detected, lifting the mouse button will not change the current selection.

Linear or quadratic interaction tensors are displayed as either ellipsoids or a set of coordinate axes that line up with the tensor eigenvectors and have lengths proportional to the tensor eigenvectors. The ellipsoids should be interpreted as a result of stretching a unit sphere in the x, y and z directions by one of the tensor eigenvalues followed by a rotation of the resulting ellipsoid into the tensor frame. In either case, the resulting object is then translated so that its centre is over the corresponding spin.

Frames are displayed by drawing lines along the positive directions of the coordinate axes. The active frame is displayed in blue while inactive frames are grey.

3.5.5 Grid View

The purpose of the grid view (marked as ④ in figure 3.10) is to display the spins in the current spin system in table format, so that bulk editing operations may be performed easily. The table has seven columns, a row for each spin in the spin system, and a final, blank, row.

The first column contains a widget controlling the selection status of the spin. Clicking once enables the check box widget for editing and further clicking toggles the check. Clicking outside of the cell hides the widget and sets the current selection appropriately. This behaviour is not the optimal behaviour, which would be that clicking anywhere within the cell toggles the selection status of that spin. The reason for this deficiency is due to the limitations of the `wxGridCellBoolEditor` class of the wxWidgets toolkit.

The other columns correspond directly to the parameters associated with spins. They can be edited by clicking once to enable editing operations; after performing the edit, clicking elsewhere disables the editing operation. The in-memory update is performed when the cell is deselected. Editing the "element" field also changes the mass number to the default value for that element, e.g. 1 for hydrogen and 13 for carbon.

The last row in the grid is always blank. Double clicking on this row creates a new hydrogen spin, positioned at the origin, and a new blank row is added to the end of the table. Spins can be rapidly added by double clicking repeatedly.

3.5.6 The Frame Tree

The frame tree displays a hierarchy of reference frames that are available using the `wxTreeCtrl` widget. The active frame is displayed in bold. Right clicking on a frame brings up a context menu with the additional option labelled: "Activate Frame." Clicking on this option sets the clicked frame to be the active frame.

3.5.7 Interaction Editor

The interaction editor (marked as ③ in figure 3.10) permits editing of the numerical values of interactions associated with a particular spin that is indicated by the drop-down box in the top left corner. The user can change the spin using the drop-down box, but the spin is also changed automatically whenever a spin is clicked in the 3D display or the grid.

Once a spin is selected, all interactions associated with that spin are displayed in the list-box in the format "⟨`kind`⟩ (⟨`storage`⟩)." Clicking on an interaction in the list box makes it editable (see below). If a bilinear interaction `kind` is selected, the drop-down menu marked as "coupled spin" is ungreyed, allowing a second spin to be selected.

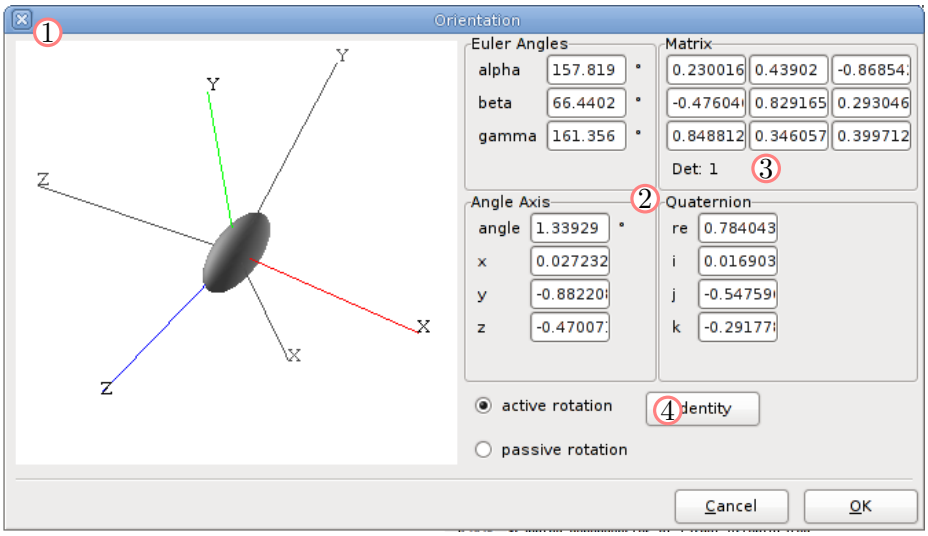


Figure 3.12 – The orientation dialogue. (1) 3D display visualising the rotation acting on a set of coordinate axes. The display may be rotated with the left mouse button. Rotating the display updates the numbers displayed on the right panel in real time. (2) The rotation is displayed in all four supported conventions simultaneously. Editing any text box updates the other three conventions. (3) The determinant of the rotation matrix. (4) A button is provided to reset the rotation to the identity.

Below the list box there are two buttons: "Add" and "Delete." Pressing "Add" creates a new interaction associated with the current spin, while pressing "Delete" removes the currently selected interaction.

The values shown in the left section of the interaction editor directly correspond to the values stored in the interaction data-type, and the in-memory values typically update when a widget loses focus.

3.5.8 Orientation Dialogue

The orientation dialogue is a generic dialogue for editing rotations and orientations and is used in several places within Spinach GUI. It is also available as a standalone program, which may be built separately (see appendix D for instructions on building the standalone orientation dialogue).

The dialogue (see figure 3.12) is split into two sections: a 3D display (marked as (1)) and the numerical panel (marked as (2)). Two sets of coordinate axes are displayed in the 3D display, a set of fixed axes that act as a reference, and a second set of coloured axes. The current rotation is the rotation that carries the reference coordinates onto the coloured coordinates. Dragging the mouse over the 3D display rotates the coordinate system in the same manner as the 3D display in the main window, but right clicking and dragging does not

trigger translation. Rotating the coordinates with the mouse updates the current rotation and the numerical values in the panel on the right.

The right-panel displays the explicit numerical values of the four different representations of the current rotation. Editing these values updates the rotation. However, there are some complications because, for all representations apart from Euler Angles, the user may enter invalid values. For example, entering a quaternion that fails to be normalised. Furthermore, immediately trying to fix the user's input will lead to problems. Suppose the user wants to enter the quaternion $0.7071 + 0.7071i + 0j + 0k$. If the quaternion were immediately normalised, then upon entering "0.7071" in "re" box and clicking elsewhere, the value would be corrected to "1" before the user had a chance to enter "0.7071" in the "i" box.

Instead, the orientations dialogue follows the principle of never editing user input unless the user directly requests that action. It is assumed that this should be done when the user begins editing one of the other conventions. For example, if the user begins editing the rotation as a quaternion and changes their mind and, instead, decides to enter the rotation as a set of Euler Angles, then only when the values of the Euler Angles text boxes are changed will any kind of automatic update of the quaternion text boxes be performed.

The behaviour of the "Ok" and "Cancel" buttons are determined by the context in which the dialogue is called, but both dismiss the dialogue.

3.5.9 EasySpin Export Dialogue

The EasySpin dialogue is brought up by selecting *File→Export to EasySpin...* from the menu. The dialogue is divided into two sections, the input panel and the preview panel. The user edits values in the input panel and the preview panel is updated with valid EasySpin code reflecting the options selected in the input panel, so long as the *Automatically Generate* check box (marked as ⑤ in 3.13) is checked. The user may edit the text in the preview panel, which causes the check box to be unchecked, preventing any changes from being automatically overwritten.

Two buttons, "Copy" and "Save", are provided. Pressing "Copy" copies the contents of the preview panel into the system clipboard, while "Save" brings up the operating system-dependent save file dialogue. The contents of the preview panel are written as a text file to the path the user selected.

A principle that the EasySpin dialogue attempts to follow is that the text in the preview panel should be minimal. EasySpin has many options, yet most of these have sensible default values and can be omitted. As it would not be user-friendly to output EasySpin code with all options set explicitly, several other schemes were considered that could have been used to implement this behaviour:

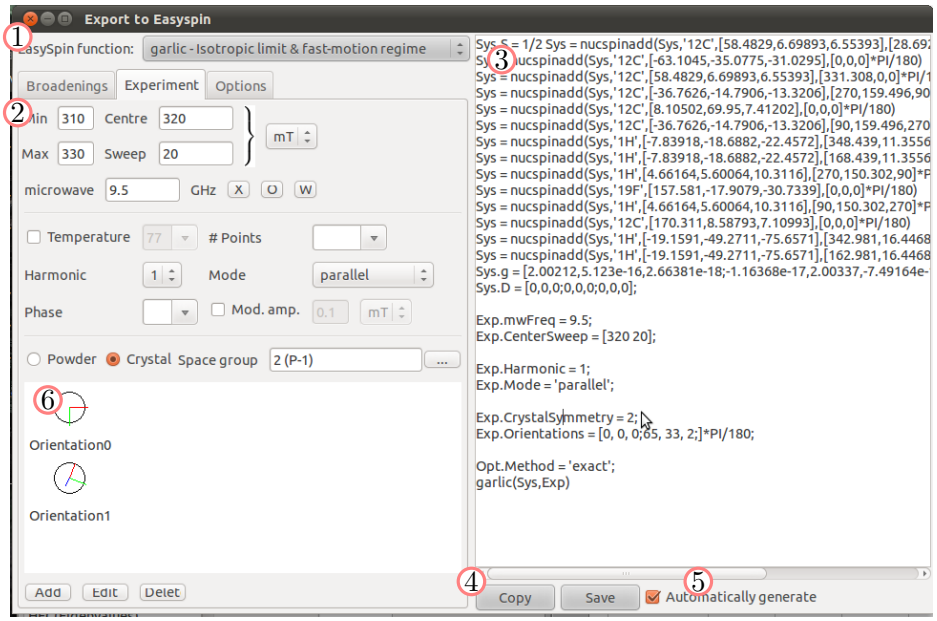


Figure 3.13 – Screenshot of the EasySpin export dialogue. ① The three most popular EasySpin modules, `garlic`, `chili` and `pepper` are supported. ② Most basic options may be entered. ③ output is generated in real time. ④ Output may be copied to the clipboard or saved as a `MATLAB` m-file. ⑤ The preview text is updated automatically if this check box is ticked. Editing the preview pane automatically unchecks this check box. ⑥ Orientations are drawn as transformed sets of axes.

- **Detect the Default** All fields start initialised to their default EasySpin values. If the user happens to enter a value that is the EasySpin default, then no code for that option is generated.
- **Generate after Edit** All fields carry an additional Boolean flag which is set on the first edit. Once the flag is set, code related to that field is generated.
- **Explicit Switch** Each field has an additional check box to indicate whether or not the field is to be left specified.
- **Blank String Represents the Default** When using text boxes, entering any text indicates that code related to that text box's setting should be generated, but blanking the text indicates that no code should be generated, which lets EasySpin use its default value.

Generally, the "generate after edit" scheme was inappropriate because it involves giving the dialogue state that is not visually represented to the user, as well as preventing the user from going back to the default state without dismissing the dialogue.

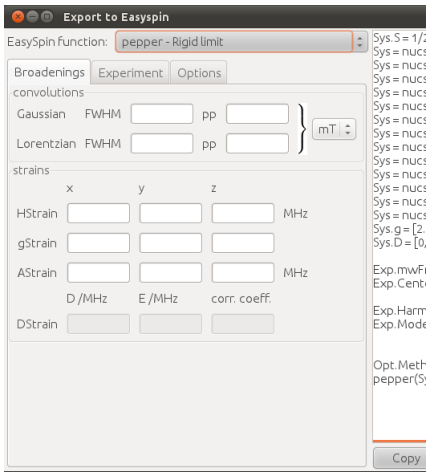


Figure 3.14 – EasySpin line broadening tab

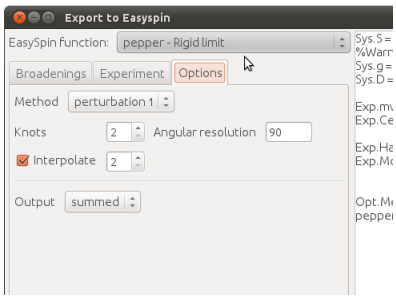
The "explicit switch" scheme is essentially the "generate after edit" scheme with the additional flag made explicit. This scheme would have been excessive; a checkbox for every field would have taken up too much screen space. However, it was used for setting the temperature and the modulation amplitude because EasySpin uses different mathematical models depending on whether a temperature is specified or not, so it was judged to be appropriate to keep these settings explicit as possible.

The "detect the default" scheme requires the GUI to know the defaults used by EasySpin, which creates a potential maintenance problem. In addition, the behaviour is confusing, in that the user could enter a value which would cause the corresponding line in the output to disappear. "Detect the default" was used for the "harmonic" and "mode" options as they do not allow free text, preventing use of the "blank string represents unset" scheme. Generally, as these have defaults that are very unlikely to change, this scheme is acceptable here.

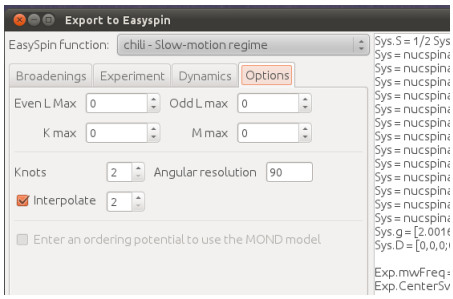
For most options, the "blank string represents unset" scheme was judged the most appropriate, as its behaviour is very natural for the user, the code does not need to be aware of the EasySpin defaults, and no additional screen space is needed for a checkbox.

There are too many options to describe in any great detail, but they largely correspond to the EasySpin options which are covered in the EasySpin documentation [59]. The interfaces for setting these options is shown in figures 3.14, 3.15, and 3.16.

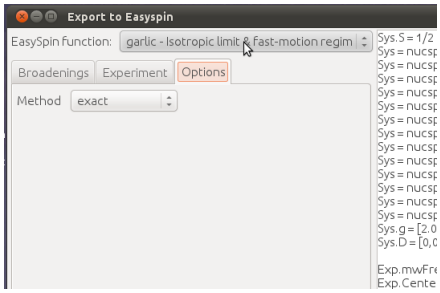
The two types of spectra available are power and crystal spectra. In the case of crystal spectra, the crystal space group and a list of crystal orientations must be provided. An interface for entering this additional information appears when "Crystal" is selected. This interface contains "New" "Edit" and "Delete" buttons for editing the list of orientations, along with a list-box (shown as ⑥ in figure 3.13) for containing the on-screen representations of the orientations. In addition, there is a text box for entering the space group.



(a) Pepper Options



(b) Chili Options



(c) Garlic Options

Figure 3.15 – Options tabs for pepper, chili, and garlic. If the user changes the EasySpin function with the drop down menu, the options are automatically updated.

The orientations are labelled with numbers ("orientation 0", "orientation 1", *etc.*) because there is no obvious way to generate meaningful names for orientations. Usability is improved by a small visual representation of the orientation that is drawn next to the label. The interpretation of this visual aid is explained in figure 3.17.

This concludes the functional specification section, where the intended behaviour was described at a high level. The following sections form the technical specification, and, so, focus on the questions of how the described behaviour is achieved.

3.6 Third Party Libraries

The software described here makes use of the following open-source, third party libraries to achieve the functionality described. Generally these libraries are not associated with an academic paper or book that may be cited in a traditional manner, but nevertheless the development of Spinach GUI has been aided by their existence and so they are acknowledged here.

wxWidgets This library was used to create the GUI elements in a cross-platform manner

Eigen 3 This library provides a convenient interface for linear algebra operations. It also includes a number of algorithms, such as eigendecomposition, and has build in support

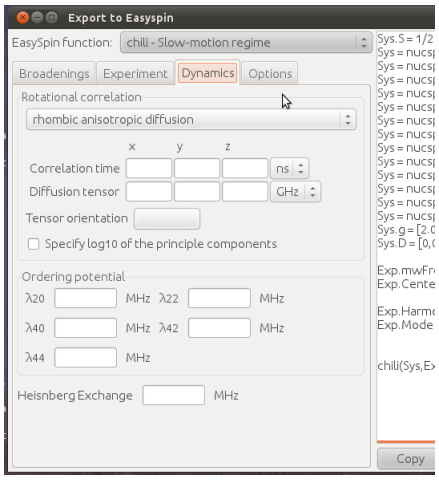


Figure 3.16 – The EasySpin dynamics tab.

for three of the four rotational conventions used by spin XML.

sigc++ This library was used to allow various subsystems of the software to communicate in a convenient manner.

OpenGL OpenGL is technically not a library at all, but a API specification maintained by the Khronos Group. Its purpose is to provide a common and system-independent interface for issuing rendering instructions to graphics hardware. Specific implementations are provided by operating system and graphics card vendors.

3.7 Input and Output

Before any interaction or visualisation can take place, a spin system must either be specified or read in from a file. In Spinach GUI, the file input/output (I/O) has been organised into a very modular, loosely coupled subsystem. Files relating to I/O may be found in the `shared/formats` directory.

I/O filters are derived from the abstract class `ISpinSystemLoader` in `spinsys.hpp`. `ISpinSystemLoader` defines the five functions that must be overridden by a derived class. Of these, three are trivial; `GetFilterType` should indicate through its return value whether the derived class supports only input, only output or both. `GetFormatName` simply returns the human-readable name of the format, and `GetFilter` returns an expression that is used to filter files by their extensions in the `file→save` dialogue. The format of the filter string is documented in the wxWidgets documentation of the `wxFileDialog` class.

A careful examination of several existing file formats suggests that a degree of redundancy in processing their contents is present that may be factored out. A fairly typical pattern

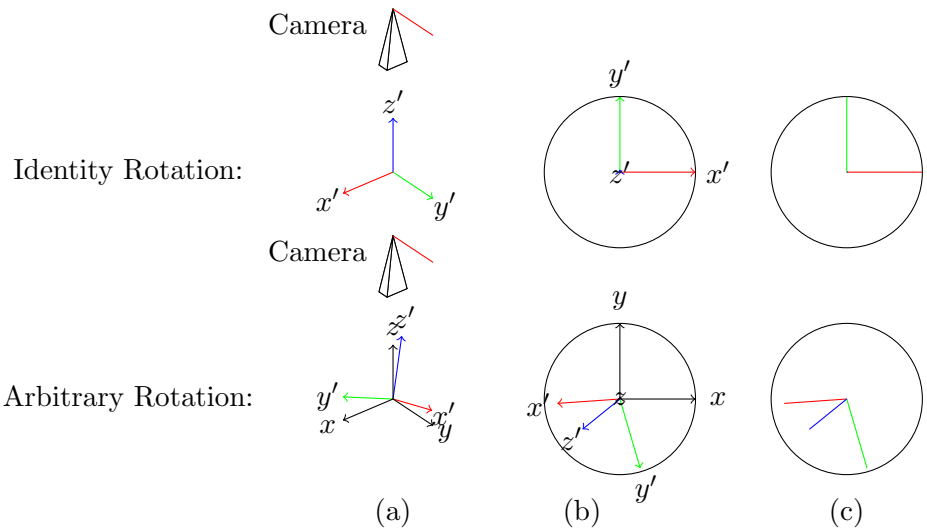


Figure 3.17 – The "visual hash" scheme used in the EasySpin dialogue for crystal orientation specification demonstrated for the identity rotation and a arbitrary rotation. Rotations are displayed as circle containing three lines of different colours (column c). These correspond to set of three coloured axes pictured through a virtual camera (column a, camera indicated by the pyramid. The single red line indicates the direction of the top of the view). Column (b) show the view from the camera before axis labels are removed. While recovering the original rotation from the picture is not intuitive, distinguishing between rotations is.

in an input file is to have a list of nuclei that are identified by a number, while, in another part of the file, there is a list of interaction tensors that identify the associated spins by those numbers. Examples of file formats that conform to this pattern are spin XML and Gaussian 03 output logs. One strategy would be to create each entity in full in-memory as soon as it is recognised, but a problem arises if that entity relies on the existence of another entity that has not yet been created. For example, a dipolar interaction could reference spins that appear further down the file. This problem is particularly prevalent in spin XML because reference frames appear at the end of the file and, so, interpretation cannot begin until everything has been read into memory.

The solution adopted by the g03 and spin XML input filters is to only partially construct the spin system during the first pass, based on local information only, and then hand these jigsaw pieces to the shared routine `Assemble` in `proto.cpp`, found in the `shared/formats` directory, for assembly. Classes for holding the partially constructed pieces are defined in `proto.hpp` and are named with the prefix "`Proto`".

Each of the so called proto-entities defines a `selfCheck` that performs consistency checks, based solely on local information within the object.

- `ProtoSpin::selfCheck` tests that each coordinate of the position vector was not NaN, the element ≥ 0, and that isotope (mass number) ≥ 0.

- `ProtoInteraction::selfCheck` That no floating point numbers are set to NaN and that the stated interaction `kind` matches the `form`.
- `ProtoFrame::selfCheck` Check that none of the floating point numbers are set to NaN.

Once these local checks are performed, the relational checks (see section 3.4.4) are performed and, finally, a new `SpinSystem` object is constructed.

3.8 Core Data Structures

Once the input data has been verified by `Assemble`, information is assembled into the final representation of the spin system in memory. Because of the constraints imposed by Spinach GUI's aim to support spin XML (section 3.4), the information stored by Spinach GUI is largely determined by spin XML.

The core classes used for in-memory representation of a spin system are the `SpinSystem`, `Frame Spin`, `InteractionPayload`, and `Orientation` classes. Of these, `Orientation` is generally used as a value-type that is allocated on the stack, while the other classes should be allocated on the heap.

3.8.1 The Orientation Class

The orientation class is designed to represent an element of $SO(3)$, to be interpreted as a rotation,[11] in a uniform manner. The public interface of the orientation class was informed by the ideas presented in section 3.3. The user of the class need not be aware of the actual convention that the rotation is stored in and can simply call `Orientation::Apply` on a vector to compute the effect on a vector. If a particular representation of a rotation is required, the user may obtain it by using the set of member functions:

```
1  EulerAngles GetAsEuler() const;
2  Eigen::Matrix3d    GetAsDCM() const;
3  Eigen::AngleAxisd  GetAsAngleAxis() const;
4  Eigen::Quaterniond GetAsQuaternion() const;
```

and the rotation class will perform the necessary conversion required to obtain the rotation in the requested convention. With the exception of the quaternion to Euler angle conversion (see section 3.3.4), these conversions are well known (see a good textbook on mathematics for 3D graphics such as [76]) and, in fact, those not involving Euler angles are implemented as thin wrappers around the Eigen library.

[11] Note that, unlike the spin XML complex type of the same name, the `Orientation` class does not encode a full orientation its own, only a rotation.

Unlike the other core classes, the `Orientation` class should generally be used as a value-type rather than being allocated on the heap. This decision was made because `Orientations`, unlike the other core classes, do not reference other core objects and are typically not "watched" by GUI elements (except indirectly when they are members of other core classes) and, so, they can be copied without causing aliasing problems that would require schemes such as reference counting to solve. The ability to treat `Orientations` as value-type significantly simplifies the code where they are used.

Internally, `Orientations` are implemented as an object with four orientation members and a `mType` variable to indicate the `kind`:

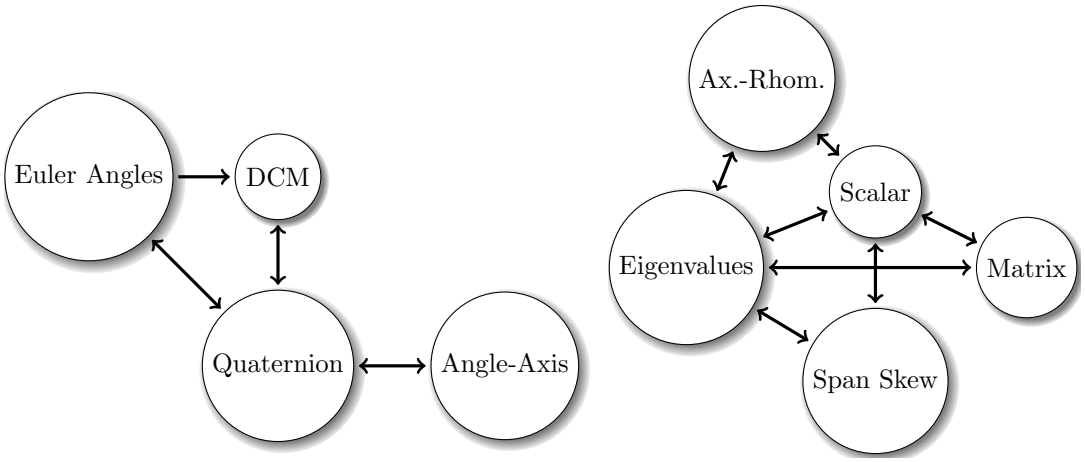
```
1  class Orientation {
2  public:
3      enum Type {
4          EULER,
5          ANGLE_AXIS,
6          QUATERNION,
7          DCM
8      };
9  private:
10     Type mType;
11     EulerAngles mEuler;
12     Eigen::AngleAxisd mAngleAxis;
13     Eigen::Quaterniond mQuaternion;
14     Eigen::Matrix3d mMatrix;
15 };
```

Whenever the value of an `Orientation` object is updated, `mType` is set to the appropriate value and the corresponding member is written to. The other three members then become invalid and it would be an error to read from them. As these details are hidden behind the public interface, there should be no way for the user code to violate these usage rules.

This implementation is, currently, sub-optimal. Even though only one of the four members is "active" at any one time, memory is allocated for all four. Although, with current generation hardware, RAM is not in short supply, using `Orientation` as a value type means it will be copied when being passed as an argument to a function, so every additional byte adds an overhead to function calls involving `Orientations`[12] and increases the amount of cache memory they use.

One way to solve this problem would be with a **`union`**, however, the C++03 standard [77] disallows members of `unions` to have constructors but all the members of `Orientation` do have constructors and, so, will not fit in a **`union`**. In particular, the `AngleAxisd`, `Quaterniond`, and `Matrix3d` classes are all from the third party Eigen library, so modifying them is not feasible.

[12] Unless the optimiser successfully inlines the function call.



(a) Rotational Representations **(b)** Interaction Representations

Figure 3.18 – Interconversions between rotational and interaction representations. Each circle represents a representation and each arrow represents an algorithm to directly convert from one representation to another. Note that converting to the scalar representation loses information.

A second solution, used in the `Interaction` class, would be to use the 3rd party template, `boost::variant`, which implements a tagged union of general types as well as providing safe utilities for working with them. Unfortunately, at the time of writing, there is a known bug that surfaces when one `boost::variant` is nested within another.[13]

However, the most recent C++ standard (informally known as "*C++11*") allows unions of more general types, including types with constructors. In light of this, the current implementation of `Orientation` should be regarded as a temporary solution until *C++11* is sufficiently well supported by compilers to be usable.

3.8.1.1 Rotation Conversations

The actual rotation conversions are performed by the following four overloaded function sets (making 16 functions in total):

```
1 EulerAngles         ConvertToEuler     (const T& rot);
2 Eigen::Matrix3d     ConvertToDCM       (const T& rot);
3 Eigen::Quaterniond  ConvertToQuaternion(const T& rot);
4 Eigen::AngleAxisd   ConvertToAngleAxis (const T& rot);
```

where `T` is a type from the set:

{`EulerAngles`, `Eigen::Matrix3d`, `Eigen::Quaterniond`, `Eigen::AngleAxisd`}

[13] At the time of writing, information relating to this bug can be found in the boost bug database, at: https://svn.boost.org/trac/boost/ticket/5146

Not every function in the set is implemented independently; some call others (figure 3.18). For example, to get to an angle axis representation from an Euler angle representation, `ConvertToAngleAxis(const EulerAngles&)` would call `ConvertToAngleAxis(const EulerAngles&)`.

3.8.2 The `InteractionPayload` Class

The `InteractionPayload` class is designed to store a pure tensor without any extra information, such as the associated spins that might be coupled by the tensor or the `kind` of the interaction it might represent. `InteractionPayload` is used by the core class `Interaction` and, also, by the `ProtoInteraction` class that is used in spin system assembly (see Section 3.7).

```
1 class InteractionPayload {
2 public:
3      void ToScalar();
4      void ToMatrix();
5      void ToEigenvalues();
6      void ToAxRhom();
7      void ToSpanSkew();
8
9      energy           AsScalar() const;
10     Eigen::Matrix3d  AsMatrix() const;
11     Eigenvalues      AsEigenvalues() const;
12     AxRhom           AsAxRhom() const;
13     SpanSkew         AsSpanSkew() const;
14 protected:
15     typedef boost::variant<energy,
16                            Eigen::Matrix3d,
17                            Eigenvalues,
18                            AxRhom,
19                            SpanSkew> var_t;
20     var_t mData;
21 };
```

Because the `Orientation` class does not contain a `boost::variant`, `boost::variant` can and has been used to implement the `Interaction` class. The three types: `Eigenvalues`, `AxRhom`, and `SpanSkew` are three very light-weight datatypes that are defined as follows:

```
1 struct Eigenvalues {
2     Eigenvalues(const double _XX,const double _YY,const double _ZZ,
          const Orientation& o);
3     double xx;
4     double yy;
5     double zz;
6     Orientation mOrient;
7 };
8
9 struct AxRhom {
```

```
10      AxRhom(const double _iso,const double _ax,const double _rh, const
            Orientation& o);
11      double iso;
12      double ax;
13      double rh;
14      Orientation mOrient;
15  };
16
17  struct SpanSkew {
18      SpanSkew(const double _iso,const double _span,const double _skew,
            const Orientation& o);
19      double iso;
20      double span;
21      double skew;
22      Orientation mOrient;
23  };
```

3.8.3 The Heap-Allocated Classes

The heap-allocated core classes are the `Spin`, `Interaction`, `SpinSystem`, and `Frame` classes. These represent spins, interactions, spin systems, and reference frames, respectively. All the heap allocated classes, except for the `Frame` class, are derived from the template `SpinXMLBase<T>`. The *curiously recurring template pattern* (this is a well known C++ idiom, details can be found in textbooks such as [78]) is employed to make this class aware of the classes that will be derived from it. The ability to treat the derived classes polymorphically at run time is lost when this pattern is used, but in this case as the derived classes are used for very different purposes, so the lack of polymorphism is a small loss.

One of the primary purposes of the `SpinXMLBase<T>` template is to provide a set of standard signals that are used to cause user interface elements to update and to aid memory management (section 3.8.4). Due to the *curiously recurring template pattern* these signals can be written in terms of the derived class, even though they appear in the base class. The following standard signals are provided:

`sigChange` Emitted whenever the information in the object changes. It is the responsibility of the inheriting object to send this signal.

`sigDying` Emitted just before the memory allocated by the object is freed. This signal is called from the base class destructor, so derived objects do not need to worry about emitting this signal.

`sigAnyChange` This signal is a static member of the base class, and, so, potential listeners do not need a specific instance of a class to subscribe to this signal. The base class automatically relays all `sigChange` signals through this global object and, so, the

derived class does not need to worry about emitting this signal as long as it emits `sigChange` signals correctly.

In addition, `SpinXMLBase<T>` is responsible for managing the spin XML reference frame associated with each object. As the `SpinSystem` class (covered next) does not need an associated reference frame, `GetPreferredFrame` and `SetPreferredFrame` should not be called on a `SpinSystem` object. Simplified definitions of the `SpinSystem`, `Spin`, and `Interaction` classes follow:

```
1 class SpinSystem {
2   std::vector<Interaction*> interactions;
3   std::vector<Spin*> spins;
4   Frame* rootFrame;
5 };
6 class Spin {
7   Vector3d position;
8   string label;
9   long element;
10  long isotope;
11 };
12 class Interaction {
13  Interaction::Kind mKind;
14  InteractionPayload mPayload;
15  Spin* mSpin1;
16  Spin* mSpin2;
17 }
```

3.8.4 Signals and Memory Management

The memory management scheme used by the core layer generally follows the scheme used by wxWidgets. Objects are organised in a hierarchy with parent classes being responsible for managing the memory of child classes. For example, the code listed here creates a new spin and hands over control of that spin to a `SpinSystem` object:

```
1 vector3d pos = vector3d(0.0,0.0,0.0); //The position of the Spin
2 string label = "Hydrogen #42"; //A human readable name
3 atomicNumber = 1;
4 massNumber = 1;
5 mySpinSystem->InsertSpin(new Spin(pos,label,atomicNumber,massNumber));
```

From now on, the spin system object is considered the owner of the spin object; if this spin system is deleted, the spin object will be destroyed along with it. When a spin being held by a spin system is to be deleted, the following must happen:

1. Memory held by the spin object must be freed.
2. Any interactions that reference the spin must be either updated to no longer reference the spin, or deleted outright. In Spinach GUI, the second approach is taken.

3. If the interaction class is deleted, the reference held by the spin system object must also be deleted.

4. External objects that may hold references to the spin must be informed that they should get rid of those references.

Because every object has a single, well-defined, owner, the usual garbage collection problem of freeing objects is solved, but this does not solve the inverse problem of getting rid of all references to an object in response to that object being deleted. There appears to be no general solution that would work without the cooperation of foreign objects that hold references, as such objects may need to perform arbitrary actions in addition to removing their references, for example, graphical objects might need to redraw themselves to reflect a deleted spin.

The solution adopted by Spinach GUI is based on signals. In object orientated programming, a signal object is an object that, conceptually, gives out a signal. A metaphor that could be used here would be a radio broadcasting tower. The broadcasting tower sends out signals without having to know the details of where they are received or who they are received by. In the same way, a signal object does not need to know where messages it sends go, which decouples the signal object from the rest of the program. Contrast to the situation where signals are not used. The object would need to know how to inform every interested object about changes, meaning that whenever details of the rest of the program changed, the signalling object would need to be updated to reflect those changes. The signalling system used in Spinach GUI is implemented by the open source sigc++ library.

In Spinach GUI each spin system object sends out a signal when deleted and it is the responsibility of any objects holding references to subscribe to these signals and remove any references they might hold, along with doing other necessary cleanup.

Continuing from the aforementioned example, suppose a spin is deleted through the following actions:

```
1 //In user code
2 Spin* spin = spinsystem->GetSpin(0);
3  delete spin;
```

then a series of events is set into motion that fulfil the above requirements:

1. The spin emits a `Spin::sigDying` event, which the `SpinSystem` subscribed to when the spin was added to the spin system. This signal is received by the `RemoveSpin` function, which scans through the `mSpins` member vector until it finds the pointer to the spin and removes that pointer. This accomplishes point (3) of the above list.

2. When interaction objects are assigned spins, they also subscribe to the spin's `sigDying` signal and will delete themselves when receiving such a signal, accomplishing point (2).

3. Any external objects that reference spin system objects are expected to subscribe to the `sigDying` event themselves and respond appropriately when that signal is received. Assuming well written external objects, point (4) is accomplished.

4. Point (1) is accomplished by the **`delete`** operator.

3.9 The Graphical Layer

The graphical layer is built on top of the core layer. Care has been taken to ensure that no part of the core layer depends on the graphical layer. All source files that belong to the graphical layer are kept in the `gui/` directory, except for code dealing with pure OpenGL, which is kept in the `3d/` directory.

Cross-platform graphical widgets are provided by the open-source library wxWidgets, which abstracts away the differences between operating systems with regards to creating and managing windows and the graphical user interface components (often called "widgets") they contain. wxWidgets also presents a higher level interface to this functionality than would be available using standard operating system calls. wxWidgets was chosen because it:

1. has an open-source licence,

2. wraps the native widgets of the operating system to provide as seamless an experience as possible, and

3. has a long development history and has been successfully used in many real world scenarios: both commercial, open source, and academic.[14]

3.9.1 The SpinachApp Compilation Unit

Execution of Spinach GUI begins in the `SpinachApp` compilation unit, where the `main` (or `WinMain` on Windows) function may be found. After a few initial bookkeeping operations, including setting up the graphical reporting of assertion errors (see section 3.9.6), control is passed to wxWidgets via wxEntry. All wxWidgets applications have a single global object that inherits from the wxApp class; for Spinach GUI, this class is `SpinachApp`. Control is passed back to Spinach GUI code in the `SpinachApp::OnInit` function. `OnInit`

[14] Three examples of significant pieces of software built on wxWidgets from these three domains are AVG anti-virus, Audacity, BOINC.

creates a new SpinSystem object that will last the lifetime of the application, creates a new RootFrame, and shows it (see Section 3.9.5).

SpinachApp also holds all global state (for example: the current selection, the spin which the mouse is currently hovering over, *etc*), except for graphical settings. These are found in the displaySettings compilation unit (section 3.9.5). Generally, global state is stored in variables that are only in scope in their respective compilation units and accessible only via a pair of getter and setter functions. The setter function emits a signal, informing subscribers that a particular global variable has been set. For example, the API for selection management is found in SpinachApp.hpp and consists of the functions and signals: IsSelected(SpinXML::Spin*), GetSelectedCount(), and GetSelection().

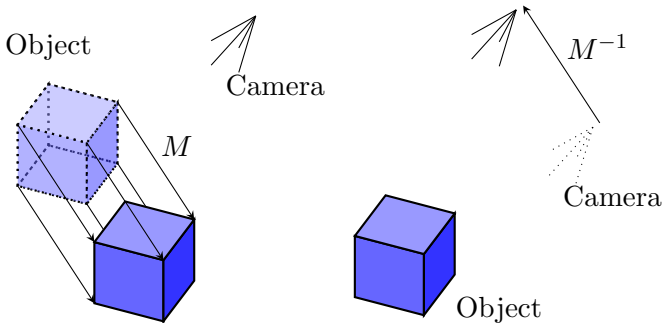
3.9.2 The Spin Grid

The spin grid is derived from the wxWidgets class wxGrid. During the construction of this object, columns are set up (names, widths, *etc.* are set) and then SpinGrid::RefreshFromSpinSystem is called. RefreshFromSpinSystem is responsible for copying the actual spin data from the SpinSystem to the grid. RefreshFromSpinSystem makes use of the SetupRow and UpdateRow for setting up and copying the spin data to individual rows.

The following signals are subscribed to automatically by SpinGrid:

- SpinSystem::sigReloaded – Refreshes the contents of the grid from the SpinSystem object by calling SpinGrid::RefereshFromSpinSystem.
- SpinSystem::sigNewSpin – Appends a row containing the new spin to the end of the grid.
- SpinSystem::sigAnySpinDying – Removes a row from the grid.
- sigHover – Changes the colour of the relevant row in response to a user moving the mouse over a spin in the 3D display to give visual feedback. The colour that the spin changes to is determined by the value of the hoverC variable.
- sigSelectChange – The current selected set of spins is indicated on the grid via colouring. The SpinGrid receives information about the selected set via this signal.

The SpinGrid class exports sigc::signal<**void,** SpinXML::Spin*> sigSelect, a signal which is emitted whenever the user clicks on a row. On start up, the RootFrame class corrects this signal to the InterEditPanel class so that clicking on a row of the table changes the spin that is currently being edited. Because of the use of signals, SpinGrid



(a) Moving the object with M

(b) Moving the camera with M^{-1}

Figure 3.19 – Applying the isometry, M, to the position of a conceptual camera produces identical results to applying the inverse isometry, M^{-1}, to the scene.

does not need to explicitly know of the existence of `InterEditPanel`, achieving decoupling of the two objects.

3.9.3 The 3D Display Classes

Generalised 3D drawing capacity is based on the `Display3D` abstract base class, which is itself derived from the wxWidgets class wxGLCanvas. The purpose of `Display3D` is to provide OpenGL initialisation, basic user interaction, camera management (with the help of the `Camera` class), and a higher level interface to OpenGL picking mode, while being agnostic about the geometry that will be drawn. By default, the `Display3D` class will rotate the scene whenever the left mouse button is clicked and dragged, and will pan the scene on a right-click-drag.

3.9.3.1 Camera Management

As OpenGL is designed as a low level interface to graphics hardware, there is no way to define an explicit camera position. This is because moving an object away from the viewer produces identical results to moving the viewer towards the object (figure 3.19), so OpenGL chooses not to differentiate between these two cases. Generally, the camera position and orientation can be regarded as coming from the first matrix to be placed on the model-view matrix stack. However, it is often very useful to think of a camera with a physical position in space. The purpose of the `Camera` class is to track the location of the imaginary camera and provide a matrix to place on the model-view matrix stack. In general, the public interface of the camera object is relatively straightforward, with separate member functions that perform translations, rotations, and zoom operations on the camera.

Objects of classes deriving from `Display3D` keep pointers to two camera objects: one for the 3D view and one for any 2D overlay. `Display3D` handles the panning and zooming behaviour of the camera automatically; derived classes do not need to react to these events.

3.9.3.2 The `GLMode` classes

In order to better model the low-level hardware and increase the efficiency, OpenGL is a highly stateful API. State changes in the API usually map directly onto state changes in the graphics card hardware. Several hardware state changes might need to be combined to create a desired effect. While state changes can be mixed with the geometry drawing code, Spinach GUI chooses to separate them where possible for increased flexibility and code clarity. The `GlMode` class provides a simple interface for switching effects on and off:

```
1 class GLMode {
2 public:
3   virtual void On()  = 0;
4   virtual void Off() = 0;
5 };
```

The following derived classes are defined in `glmode.hpp`:

GLLighting Turns lighting on and off. Also creates some default lights.

GLWire While switched on, solid objects are drawn in wire frame.

GLTranslucent While switched on, objects drawn are translucent.

GLPicking While switched on, nothing is drawn to scene. Instead, the `GL_PICKING` mode is activated. Picking mode will be covered in more detail below.

3.9.3.3 Picking Mode

A useful feature of OpenGL is picking mode, which is used to detect which 3D objects are being drawn within the viewing frustum.[15] This concept is illustrated in figure 3.20. This facility is often used to detect objects under the mouse for user interaction. While flexible, it is somewhat awkward to use and is too low level. For the purposes of Spinach GUI, the following always hold: the picking frustum should always be a narrow approximation of a line, its position should be determined by the position of the mouse, and only the closest object to the camera needs to be reported.

The Display3D class provides a picking mode based on this use-case. To start and stop picking mode, `Display3D::StartPicking` and `Display3D::StopPicking` should be called respectively. The abstract virtual function:

[15] in the context of OpenGL, the viewing frustum is the visible region of space between the near and far clipping planes that is shaped like a pyramid with the top shaved off

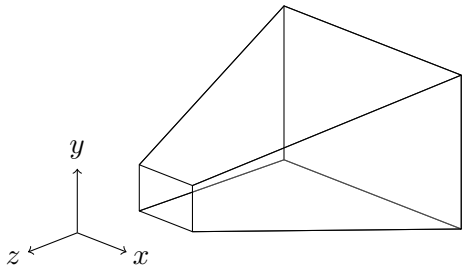


Figure 3.20 – The model-view matrix transforms objects specified in world coordinates into "eye coordinates". In these coordinates it can be imagined that a camera sits at the origin, facing in the negative z direction. Objects are then transformed by the projection matrix and any vertices that do not end up in a $1 \times 1 \times 1$ cube, centred on the origin, are culled. The preimage of this cube in the viewing "fustrum" (shown in the figure), whose dimensions determine both the field of view, and the near and far clipping planes. By severely narrowing the viewing frustum and lining it up with the mouse, only objects sitting directly under the mouse are detected and reported by OpenGL picking mode. Setting up this narrow frustum and reporting the detected objects is the responsibility of the base class, `Display3D`, while acting on this information is the responsibility of any derived classes.

```
Display3D::OnMouseOver3D(int stackLength, const GLuint* ClosestName)
```

should be overridden by derived classes. It will be called by the base class whenever `StopPicking` is called and will return a sequence of OpenGL picking names corresponding to the object over which the mouse is hovering.

3.9.3.4 Display Settings

The display settings compilation unit, found in `gui/displaySettings.cpp`, contains global state relating to the specifics of how the scene is rendered. This state is accessed via getter and setter functions. The exact state that is kept is as follows:

- For each `Interaction::Type`, a **`bool`** representing if that interaction should be drawn.
- For each `Interaction::Type`, a `Colour` representing the colour that that type of interaction should be drawn.
- For each `Interaction::Type`, a **`double`** representing the magnitude of the interaction. For a single spin interaction this is a quantity by which the ellipsoid representing the interaction is scaled. For two spin interactions, this value is the threshold value below which the interaction will not be drawn.
- A `GLUquadric` structure that is fed to OpenGL utility library functions such as `gluSphere`. For the purposes of the Spinach GUI this determines if objects are drawn as solids or as wire-frames.

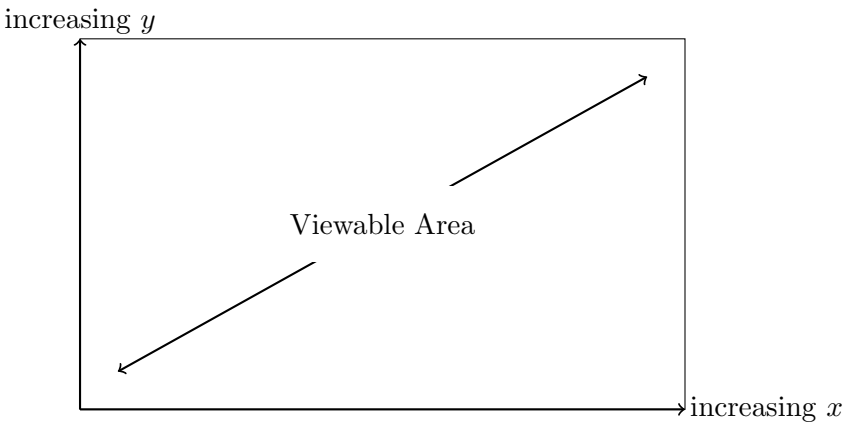


Figure 3.21 – The overlay coordinate system set up after `Display3D::Set2DView` is called. The z axis is perpendicular to the page.

- A **`bool`** representing whether bonds should be drawn or not.
- A `std::set<Spin*>` whose members are spins that will not be drawn.
- A `MONO_DRAW_MODE` (defined as a **`enum`** of the two constants `MONO_ELLIPSOID` and `MONO_AXES`), that determines whether interaction tensors should be drawn as full ellipsoids or just principle axes.

In addition to the getters and setters, the public interface of this compilation unit provides four signals: `sig3DChange`, `sigInterVisible`, `sigInterColour`, and `sigInterSize`. Of these, `sig3DChange` is emitted whenever any display setting changes and it is typically used to cause the display to be updated. The other three are emitted when an interaction changes visibility, scaling, or colour respectively and are typically subscribed to by the display settings panel in order to update them.

3.9.3.5 Rendering

Actual rendering begins when the `Display3D::OnPaint` member function is called. wxWidgets will call this function automatically whenever the scene needs to be re-drawn (for example, the render window is partially covered by another window that is then withdrawn – the bits of the canvas that are newly uncovered require redrawing). To explicitly trigger a redrawing, the `Refresh` member function of the `wxCanvas` base class should be called. In fact, `OnPaint` should never be explicitly called because wxWidgets needs to execute code to set up the `glCanvas` for drawing and this would not be run if `OnPaint` were called directly.

The first time `OnPaint` is called, OpenGL is initialised via the `Display3D::EnableGL` member function and the initial OpenGL state (such as the background colour) is set. During subsequent calls, `EnableGL` is skipped. The `Camera` object is used to set the model-view

matrix, the projection matrix, and the viewing transform based on the current size of the canvas. At this point, the first of the two pure virtual functions are called:

```
virtual void DrawScene()= 0;
```

The derived class should override this function and use it to draw any 3D objects that are required. `OnPaint` then sets up a coordinate system suitable for drawing overlays (see figure 3.21). The second pure virtual function is then called:

```
virtual void DrawForeground()= 0;
```

`DrawForeground` is ideal for drawing objects that appear in fixed positions on the scene, regardless of the position of the camera. Once any overlay is drawn, the front and back buffers are swapped and the OpenGL error state is checked.

3.9.3.6 The `SpinsysDisplay3D` class

The most important class that inherits from `Display3D` is the `SpinsysDisplay3D`, which handles the drawing of the main spin system display (described in section 3.5.4). The routines for generating the geometry drawn by the derived class `DrawScene` are kept in the `glgeometry` compilation unit as a series of classes that derive from the base class `Renderer`. The purpose of the `Renderer` base class is solely to encapsulate pushing and popping of a name to the OpenGL name stack for use with OpenGL picking (see section 3.9.3.3). The name is provided when the class is constructed. A `Renderer` may be made to draw its contents by calling its `Draw` member function. `Draw` pushes the name to the stack (unless the name is `NAME_NONE`), calls the pure virtual member function `Geometry`, that has the signature:

```
virtual void Geometry()const = 0;
```

and pops the name from the stack.

Derived classes should override `Geometry` and will usually provide a OpenGL name for their base `Renderer` class. The following geometry drawing classes are provided in the `glgeometry` compilation unit:

SpinDrawer Draws one sphere per spin centred at the x, y, and z of that spin. These spheres are given the OpenGL name `NAME_SPIN`$\rightarrow\langle$spin number$\rangle$. The colour depends on the element of the nucleus.

BondDrawer Draws cylinders between nuclei that are within a certain distance of each other to suggest bonding. Although this technique does not accurately represent chemical bonding, it is often sufficient to provide a guide to the eye.

MonoInterDrawer For every linear or quadratic interaction (*i.e.* interactions involving only one spin), draws either an interaction ellipsoid or a scaled set of axes along the interaction eigenvectors. Hyperfine interactions are also drawn centred around the spin that is not the electron, while g-tensors are not drawn by this class (see `ElectronInterDrawer`).

BilinearInterDrawer Draws bilinear interactions as cylinders between spins in a similar manner to `BondDrawer`. The algorithm for determining the width and visibility of the cylinder was described in 3.5.3.

FrameDrawer Draws a set of coordinate axes per reference frame and labels the end of each axis with "x", "y", or "z" appropriately. The active reference frame is drawn with a distinct colour. `DrawFrame` makes use of the `DrawFrameRecursive` function which, when passed a frame, adds the frame's transform matrix to the matrix stack, draws a set of the axes at the origin, calls `DrawFrameRecursive` for every child reference frame, and, finally, pops the transform matrix from the OpenGL matrix stack.

SpinSysScene is simply a convenience renderer that combines `SpinDrawer`, `BondDrawer`, and `FrameDrawer`.

InteractionScene is likewise a convenience renderer that groups the single-spin interaction renderer and the binary-interaction renderers together.

ElectronScene Draws a sphere representing an electron in the foreground coordinate system.

ElectronInterDrawer Draws the g-tensor's interaction around an electron. This drawer is also meant to draw in the foreground.

Electrons are treated specially because they often have no well defined position and are, instead, drawn in the top left-hand corner of the scene using the foreground coordinate system (section 3.9.3.5). While their position remains fixed, they do rotate in order to keep their orientation synchronised with the main molecule. This behaviour requires setting up a completely different coordinate system that is not affected by panning and scaling operations of the main 3D display, but is sensitive to rotations.

The foreground coordinate system is set up in `Display3D::Set2DView`. Here, the projection matrix is set to an orthographic projection running from the point $(0, 0)$ to the point (w, h), where w and h are the height and width of the drawable area. The near and far clipping planes are set to -100 and 100 OpenGL units, which should be sufficient for

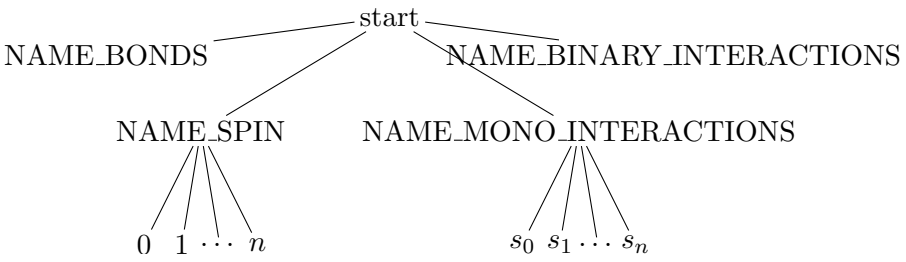


Figure 3.22 – The OpenGL picking name tree for the spin system 3D Display. The constants are defined in the `GLName` enum in `glgeometry.hpp`.

displaying most foreground objects without clipping. The model-view matrix is also reset to the identity by this function.

During `Display3D`'s `OnPaint` routine, the pure-virtual function `DrawForeground()` is called immediately after `Set2DView`. This function is overridden in the derived class `SpinsysDisplay3D` for drawing the electrons. A translation matrix corresponding to 40 pixels down and 40 pixels across from the top-left hand corner is pushed onto the matrix stack and control is passed to `ElectronScene`. `ElectronScene` extracts the pure rotational part of the 3D camera transform and uses the resulting rotation to rotate the electron, resulting in the interaction tensor rotating with the molecule and the other interactions.

3.9.3.7 `SpinsysDisplay3D` Picking

The picking name tree is shown in figure 3.22. The numbers $0 \ldots n$ correspond to the numbers of each spin, and $s_0 \ldots s_m$ are the spin numbers of the spin associated with each of the m interactions. For example, if the user hovered their mouse over spin p, `OnMouseOver3D` would be called with arguments `stackLength`$= 2$ and `ClosestName` set to a pointer to the values `NAME_SPIN`, followed by that spin's number.

3.9.4 Context Menus

Context menus, or right click menus, are a user interface feature similar to the menus associated with the menu bar. However, they are typically brought up in response to the user clicking the right mouse button rather than clicking on the menu bar. In wxWidgets, context menus are simply wxMenus, making them identical to menus from the menu bar. Each menu entry is associated with a numerical ID obtained via calling the `GetId` member of the `wxCommandEvent` passed to the on-click handler. It is the responsibility of the user of wxWidgets to interpret this ID.

The class `RightClickMenu` hides the details of dealing with IDs and provides an interface for building the context menu from actions. New actions are defined by deriving a class

from `RightClickAction` and providing definitions for the following pure virtual functions:

```
1  virtual void Exec(wxCommandEvent& e) = 0;
2  virtual bool Visible() const = 0;
```

`Exec` is executed whenever an action from the context menu is clicked. `Visible` is a user-defined predicate that should return **`true`** if the action should be visible in the context menu (for example, actions of the group of selected spins should not be shown if no spins are selected). `Visible` is called during the construction of the menu after the user has clicked.

3.9.5 The RootFrame Class

The `RootFrame` class ties the other display elements together along with menu and tool bars. The majority of its member functions are actually event handlers.

3.9.5.1 The `FrameTree` Class

In addition to the root frame itself, the `RootFrame.cpp` compilation unit also contains the `FrameTree` class. This class is responsible for setting up the hierarchical list of frames using a `wxTreeCtrl` widget, with the active frame bolded. `FrameTree` listens to the global signal `sigFrameChange` and responds by shifting the bolded element of the tree when the active frame changes.

`FrameTree` does not directly change the active frame; it builds a `RightClickMenu` containing the action `RCActionActivateFrame`, which will change the current frame when executed. The act of changing the reference frame triggers a `sigFrameChange` signal (signals were covered in section 3.8.4), updating the `FrameTree`.

3.9.6 Fatal Error Handling

All programs contain bugs when first written, and Spinach GUI is no exception. When debugging, it is good practice to try and catch errors at the point where they occur, rather than at the point where they cause a visible problem or crash. Sometimes the effects of errors become apparent as soon the errors occur; a null pointer might be received from a API function and be dereferenced immediately, causing a crash. If a debugger is attached, then execution will stop near that erroneous API call. But suppose that pointer was stored and was dereferenced much later. In this case, the source of the error would not be immediately apparent.

A very common method of catching such errors early with a good effort-to-effectiveness ratio is to test assumptions explicitly with an assert statement as often as is practical. C++ has a standard macro defined in the `<assert>` header for this purpose. An important, but subtle, point is that an assert should never be tripped by a correct program during program

execution under any circumstances for any input. If the program can be made to trip an assert under any circumstances, that is a proof that the program is incorrect and requires debugging. The use of asserts does not prove the inverse; even if a program runs without triggering and assert, that is not a proof of correctness, but, for non-critical systems such as graphical user interfaces, rigorous correctness-proofs are seldom needed.

The reason the requirement that no correct program can trip an assert is made is because asserts are generally implemented as macros, which expand to nothing on production builds. If an assert were used to catch an error in the input, then the production build would lack that error checking feature, leading to incorrect behaviour.

Asserts are distinct from regular run-time errors that might be caused by program misuse by the user, or the environment that program runs in (such as an expected external file being missing). Errors that come from environmental conditions do not indicate an incorrect program and, so, they should not trigger an assert. Instead, a well written program should catch such an error and act appropriately, perhaps by informing the user about the problem and, perhaps, provide hints as to how the problem can be fixed.

Spinach GUI defines a number of macros in `panic.hpp`. These macros are more flexible and better adapted for use in Spinach GUI than the general purpose `assert` macro of C++. They represent an improvement in four ways: 1) the `__file__`, `__line__`, and `__function__` macros are employed to provide the programmer with additional information about the file, line, and function that the error occurred in; 2) a short message may optionally be passed to provide reasoning about why triggering a particular assert represents an incorrect program; 3) a set of macros with the prefix "NaN" are defined for detecting invalid floating point values;[16] 4) the behaviour when an assert is tripped may be customised. The list of defined macros in `panic.hpp` are:

- `PANIC()` — Calling this causes a fatal error to be reported.
- `PANICMSG(msg)` — As `PANIC`, but a diagnostic message may be supplied.
- `MAYBEPANIC(p)` — As `PANIC`, but only if `p` is false, otherwise no action is taken.
- `MAYBEPANICMSG(p,msg)` — As `MAYBEPANIC`, but a message can be provided.
- `NaNPANIC(n)` — As `PANIC` if `n` has the value NaN or a floating point infinity. `n` should be of type **`double`**, `Eigen::Vector3d`, or `Eigen::Matrix3d`.

[16] Interestingly, the `NaN`—macros have turned out to detect some memory errors as well as floating point exceptions. This behaviour is unexpected, as, if `n` was a random collection of bits, then there would only be a 1/2024 chance of triggering this panic. However, in practice, invalid floating point values arising from uninitialised memory seemed to occur more frequently than could be expected if memory was random (perhaps because any pattern of memory containing all 1s is an invalid floating point number), making this macro somewhat useful for detecting memory errors in addition to bugs with floating point calculations.

- `NaNPANICMSG(n,msg)` — As `NaNPANIC` but a message can be provided.

The default behaviour after `PANIC` is called is to quit and print information about the error to the standard error stream. Sometime during initialisation, after wxWidgets is initialised sufficiently to display graphical messages, Spinach GUI changes this behaviour by calling `SetPanicHandler`. After this point, the user will be given a graphical prompt displaying information contained in the panic and giving the user a chance to continue or quit.

3.10 Conclusions and Outlook

A new graphical user interface has been introduced, supporting both editing and visualising the parameters required by the spin XML format. This GUI has now reached a state near feature completeness, in the sense that all parts of the spin XML standard can be read and understood by the GUI and all parts may be edited except for reference frames (they can still be visualised). As reference frames are a somewhat more exotic feature, it is acceptable to leave full editing capacity out of an initial release.

In addition to editing and visualising the spin Hamiltonian parameters, the GUI supports importing atomic coordinates from XYZ files and importing simulation parameters from Gaussian log files (most testing so far has been done using Gaussian03). In addition, simulation and experimental parameters may be exported to EasySpin.

There is still work left to be done, mostly with regards to integration with the Spinach library, which was recently released publicly and, thus, is likely to now have a sufficiently stable API for integration with the GUI to be feasible. The EasySpin integration will likely prove to be a good model for how Spinach integration should work. Nevertheless, the spin XML data model of spin systems and GUI as presented here will hopefully serve as a solid base upon which an interactive environment for manipulating large spin systems may be built.

3.11 Spin XML Acknowledgements

The Spin XML format was developed in consultation with Malcolm Levitt, Paul Hodgkinson, Konstantin Pervushin and Peter Hore. I would also like to acknowledge Stefan Stoll for his input and advice on the EasySpin dialogue.

Chapter 4

Direct Structure Fitting

Magnetic resonance experiments can provide a large amount of information about the structures of molecules. In particular, it has become possible, even routine, to use NMR to determine structures of small proteins [79–83]. In biochemistry, the principal advantages of NMR based structures over X-ray crystallography are that NMR techniques do not require the proteins to be crystallised, and the obtained structures are solution structures, which more accurately reflect the biological context in which they naturally occur.

However, the extraction of this information is a laborious process. Standard procedures usually involve manually assigning each line to a particular part of the structure and using various well known rules (e.g. line splitting gives information about the J-coupled groups) to obtain the relevant information. Once an assignment has been made, distance constraints can then be read off directly and the protein's structure can be optimised against these constraints. Common sources of constraints are the nuclear Overhauser effect, which provides short range constraints, or pseudocontact shifts and residual dipolar couplings, which provide longer range constraints but require the presence of a paramagnetic ion (see the review of *Bertini et al.* [45]).

Some progress was made recently by the group of Lyndon Emsley, who presented a procedure which recovered the structure of thymol from high resolution hydrogen and carbon spectra [84]. However, their work required the NMR parameters to be recovered from the spectra (thus still requiring an assignment step) before fitting took place, as well as requiring a molecular dynamics stage.

There has been some speculation [85–90] in the literature that the assignment step may be unnecessary and that obtaining structures might be possible by optimising the calculated spectrum of a test structure against an experimental structure directly. Previous attempts have been hampered by difficulties in calculating the magnetic parameters, particularly those that cannot be obtained from the ground state and the exponential scaling of spin dynamics simulations.

Progress has been made towards solving these two limiting problems for EPR experiments: sufficiently accurate quantum chemical methods for calculating the EPR parameters of radicals are now known [91–96]. *Barone and Cimino* [97] provide a recent review of the state of the art in which they compare *ab initio* g-tensor calculations with experimental results for 60 organic molecules and conclude that, in general, they "seem accurate enough to allow for quantitative studies." With regards to the second problem, it has become possible to accurately predict the spectra of large spin systems in polynomial time with respect to the number of spins [6, 22].

A new method, "direct structure fitting", of obtaining a structure directly from the magnetic resonance spectrum without a molecular dynamics stage or explicit line assignment is presented here [2]. This method has been made feasible by the aforementioned progress in the *ab initio* calculation of the spectral parameters and, more significantly, the availability of a polynomially scaling spin dynamics algorithm [6, 22, 28, 29]. In this work, all spin dynamics simulations were performed with Spinach [22] and magnetic parameters were calculated with Gaussian 03 [56].

The direct structure fitting problem essentially amounts to that of function minimisation—a very well studied branch of applied mathematics [98]—however, there are a number of details that require careful consideration if a working and practical implementation is to be produced, and the purpose of section 4.1 will be to discuss these details in depth. The results obtained using this methodology, which were also presented in our JMR paper, will be summarised in section 4.2. The methodology was the result of a large amount of experimentation. Some techniques investigated, but not included in the implementation used to obtain the results, will be chronicled in section 4.3, in the hope that they might provide inspiration for further refinements.

4.1 Methodology

Direct structure fitting is currently a very computationally expensive procedure. The results from the JMR paper [2] alone required 50,000 hours of CPU time. While this situation would change if analytical gradients of the shielding parameters were to become available, the reality is that care needs to be taken to ensure that the process is efficient. There are numerous other details that need consideration for a successful fit to be obtained. The purpose of this section will be to describe these details.

4.1.1 The Direct Structure Fitting Functional

The most obvious way to define a fitting functional is as the sum of squared differences between the theoretical spectra and experimental spectra. Let $\{B_i\}$ be the set of magnetic

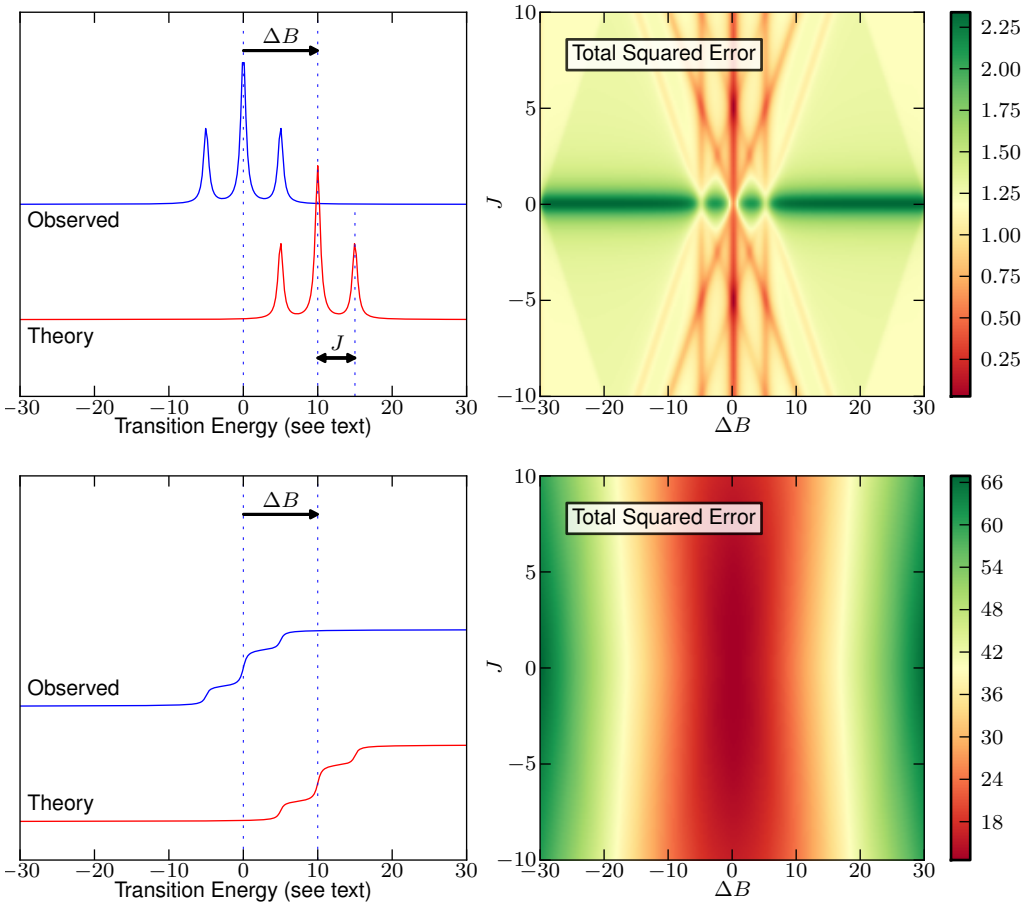


Figure 4.1 – Shown in the top left corner is an function ("observed") consisting of a triplet of Lorentzians which is meant to be representative of the kinds of spectrum found in magnetic resonance. Also shown is an attempt to model it with a two parameter model ("theory"). The resulting error as a function of the two model parameters is shown in the top right. This function displays deep, narrow valleys and many local minima and it would be difficult to find the global minima with a hill climbing algorithm. However, when the two spectra integrated (bottom left) the pathological features of the surface vanish (bottom right).

field strengths at which an experimental spectrum is measured, and let $S_{\text{theo}}(B)$ and $S_{\text{expt}}(B)$ be the values of the simulated and experimental spectra respectively at magnetic field B. Then this simple fitting function is:

$$\Omega_{\text{simple}} = \sum_{i=0}^{\text{No. Points}} (S_{\text{theo}}(B_i) - S_{\text{expt}}(B_i))^2. \tag{4.1}$$

Unfortunately, Ω_{simple} has a number of problems. Spectra in EPR and especially NMR typically contain many sharp peaks. This causes the comparison of a candidate spectrum against an with an experimental spectrum to be problematic because it results in an objective function with many local minima. Generally one will occur whenever whenever two spectral lines happen to be superimposed. This effect is illustrated in figure 4.1. Even for a relatively

simple three peak spectra and two parameter model, many local minima appear. Also note that in regions of parameter space where no lines coincide the optimisation surface is uniformly flat. Methods that do not rely on gradients, such as the simplex method (described in appendix F), grid searches, or genetic algorithms, can sometimes help but a solution to the fundamental problem of the existence of excessive numbers of local minima would be preferable.

This problem is greatly alleviated by fitting the cumulative integral of the spectra, instead of the actual spectra (this idea was recently used successfully by Husein *et al.* [99]). The fitting function is:

$$\Omega_{\text{integral}} = \sum_{i=0}^{\text{No. Points}} \left[\int_{B_0}^{B_i} (S_{\text{theo}}(B) - S_{\text{expt}}(B)) dB \right]^2 . \tag{4.2}$$

Since $S(B) > 0$ for a properly baseline-corrected magnitude-mode spectra, the integral of the spectra is a monotonically increasing function. These functions no longer exhibit the pathological properties of raw spectra. The effects of performing this integration on the two parameter model from figure 4.1 are also shown in the lower half of that figure. The pathological features of the spectrum vanish and the only remaining minima is the global one. Experience indicates that this is typical behaviour of the fitting function. The existence of local minima in the has not been completely removed, but in practice, the number of local minima is greatly reduced. The possibility of local minima in equation (4.2) still exists, it is just vastly reduced.

One last problem with Ω_{integral} is the possibility that configurations of the atomic coordinates exist that are non-physical but result in the same spectra. A good example would be a large radical with EPR-active and EPR-inert ends. The EPR-silent end of the molecule could conceivably take any configuration without affecting the value of the fitting function. During preliminary work, genuine examples of non-physical structures, even in radicals as simple as methyl, were found that could mimic the spectrum of the true structure.

To keep the candidate structure physical, an extra energy term is added to equation (4.2):

$$\Omega = E\lambda + \sum_{i=0}^{\text{No. Points}} \left[\int_{B_0}^{B_i} A S_{\text{theo}}(B + \Delta B) - S_{\text{expt}}(B) dB \right]^2 , \tag{4.3}$$

where E is the molecular energy of the configuration and λ is an adjustable parameter which is chosen to select how important the energy is to the optimiser. If λ is set too high and the optimiser will simply find the energy minimum, so care must be taken in making a selection.

Two extra fitting variables, A and ΔB have been added to allow the height and offset of the theoretical spectrum to vary as these tend to depend of the details of the experimental

setup in unimportant ways (though if a sharp calibration line, such as from TMS, is available, the need for fitting to ΔB would be diminished).

4.1.1.1 Selection of λ

Mathematically, the direct structure fitting problem is ill-posed (via at least a violation of the condition that a unique solution exists). The definitions of "well-posed" and "ill-posed" problems are covered in appendix E. Ill-posed problems are generally regarded as unsolvable in isolation; indeed Hadamard claimed that, for a mathematical problem to be a valid model of physical reality, it had to be well-posed (historical details on the development of inverse problem theory can be found in Kirsch's book [100]). However, if additional information about the solution to an ill-posed problem can be assumed, it is often possible to reformulate the problem as well-posed. The addition of the $E\lambda$ term in equation (4.3) is one such example of additional information being injected into the problem (the form of equation (4.3) matches the form given for regularisation of an unconstrained optimisation problem in Chen and Haykin's review paper [101]).

Notice that, strictly speaking, the units in equation (4.3) are ambiguous as the absolute vertical height of the lines in a magnetic resonance spectra (as opposed to the relative heights of different lines) are generally regarded as meaningless, while the magnitude of E has a very definite meaning. This necessitates the inclusion of a constant of proportionality, λ.

Due care needs to be taken when selecting a λ as too large and the system will simply converge to the DFT minimum, too small and an unphysical solution may be found and unfortunately some guesswork is required in making the selection. A guiding principle for selecting a good λ may be obtained using a quantity called the "full spectral mismatch", which is defined as:

$$\sum_{i=0}^{\text{No. Points}} \left[\int_{B_0}^{B_i} S_{\text{expt}}(B) dB \right]^2, \tag{4.4}$$

that is, the size that the error would be if S_{theo} were zero everywhere. This quantity has no meaningful unit, so one is assigned in terms of energy. By making this assignment, we are saying, in terms of energy, how costly completely failing to match the two spectra should be. By making a full spectral mismatch worth very little in terms of energy, the optimiser is given leave to effectively ignore the spectrum and will most likely just find the energy minimum. For example, if the full spectral mismatch were to integrate to 10.0 (remembering that this has no meaningful unit) and it was decided that such a mismatch should be worth an energy penalty of 100kJ/mol, then λ would be 0.1mol/kJ. In this work, energy penalties for full spectral mismatches were assigned based on intuition and many of

the results presented in section 4.2 were repeated at several different values of λ to illustrate the effects of changing this parameter.

At room temperature, approximately 2.5 kJ/mol per vibrational mode (each with a kinetic and potential degree of freedom) is available, and this observation was used as a guide for retrospectively judging a choice of λ: a direct structure fitting minimum should be thermally accessible from the DFT minimum. For proof of principle work, this *ad hoc* method was deemed acceptable; however, it is possible that more systematic methods of picking this parameter, based on the theory of inverse problems exist. Discussion of these techniques is given in section 6.3 on further work.

Two additional parameters, relative magnitude, A, and whole spectrum shift, ΔB, have been added as additional adjustable parameters to vary and a third, the line broadening, is implicit. These three parameters are unique in that they can be adjusted without requiring an additional quantum chemistry step; an attempt was made to exploit this fact in order to reduce the dimensionality of the full problem, but this attempt was met with only limited success, and, in the opinion of the author, should not be recommended (see section 4.3.3 for a full discussion).

4.1.2 The Direct Structure Fitting Functional Gradient and Minimisation

The BFGS algorithm (which is reviewed in appendix F) was used for fitting. This algorithm requires computation of gradients. While analytical gradients of spin dynamics simulations with respect to the magnetic parameters do exist [102], no such gradients are as yet available for *ab initio* magnetic parameters and, so, finite differencing was necessary. It was found that fourth order finite differencing provided sufficiently accurate derivatives for the calculations presented in [2]. The four point formula is given by:

$$f'(x) = \frac{f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)}{12h} + O(h^4), \tag{4.5}$$

where f is a smooth function, h is the step size and x is the point at which the derivative is required (this formula is derived in [103]).

Naive analysis of finite differencing schemes suggests that decreasing h increases the accuracy. As using a four point formula doubles the computation cost (which is very significant) of the gradient over a minimal two point formula, why not just reduce h until sufficient accuracy is obtained? The reason is that the results of the DFT calculations as functions of the atomic coordinates are often not perfectly smooth to machine precision.

Speculation about the inner workings of Gaussian is beyond the scope of this document, but an example of how such small scale noise might arise comes from the observation that the DFT process uses its own inner function minimisation schemes, which are not "exact

arithmetic algorithms" (an "exact arithmetic algorithm" is an algorithm that would return the exact result if it were implemented on the real numbers, rather than a floating point approximation of them. See Lyness' paper [104] for a discussion of the dangers inherent in algorithms that are not exact in this sense). When h is shrunk down too far, these small errors spoil the gradient, so we cannot gain arbitrary extra precision by shrinking h. The two point scheme proved insufficient precision even for the optimum h, but the four point scheme was able to provide sufficient precision for convergence.[1]

4.1.3 The Direct Structure Fitting Convergence Point

An apparent difficulty direct structure fitting seems to face is that magnetic parameters often vary rapidly with the atomic coordinates [95, 105, 106]. As the atomic coordinates and magnetic parameters have different units, care must be taken when making this comparison. What is meant by "very rapid" is that adjustments to atomic coordinates that are small enough to be thermally accessible produce changes to the magnetic parameters large enough to change the appearance of the spectrum. This rapid variance makes accurate *a priori* calculations of the spectrum of a molecule from its structure difficult, often requiring full vibrational averaging. Counter intuitively, this difficulty of the forward problem actually turns out to be beneficial to the reverse problem of fitting a structure to a spectrum. In particular, vibrational averaging does not appear to be needed.

Firstly, we consider the effects of uncertainties in the independent variable on the dependant variable in each direction. In the forward problem of magnetic parameter calculation, a small uncertainty in structural information is scaled up to a large uncertainty in the magnetic parameters, but in the reverse direction a large uncertainty in the magnetic parameters is scaled down to a small uncertainty in the atomic coordinates of the direct structure fitting minimum. Mathematically speaking, let Δm be a magnetic parameter uncertainty and Δr be a spatial parameter uncertainty.

$$\Delta m \approx \left.\frac{dm}{dr}\right|_{r_0} \Delta r \qquad \Delta r \approx \left(\left.\frac{dm}{dr}\right|_{m_0}\right)^{-1} \Delta m. \tag{4.6}$$

The equation to the left represents forward problem and the right equation the reverse problem. The approximate equality is used as we are neglecting higher order terms of the Taylor series expansion of $m(r)$ and $r(m)$ at r_0 and m_0 respectively. If $\frac{dm}{dr}$ is "large" (in the sense described above) small uncertainties in r translate to large uncertainties in m for

[1] As an aside: differentiation is a formally an ill-posed problem. In fact, h takes the role of regularisation parameter and shrinking h amounts to turning off regularisation [100]. For a given level of noise, there is an optimum h which can be found using tools of inverse problem theory.

the forward problem, but when considering the reverse problem, large uncertainties in m translate into small uncertainties in r.

Returning to the question of vibrational averaging, let us consider what the direct structure fitting minimum would converge to if no vibrational averaging were used. This may be answered using the mean value theorem. The mean value theorem for functions of many variables implies that, if m and p are some continuous functions integrated over some convex volume V, then:

$$\text{there exists } \boldsymbol{x} \in V \text{ such that } \int_V m(\boldsymbol{r})p(\boldsymbol{r})d\boldsymbol{r} = m(\boldsymbol{x}) \int_V p(\boldsymbol{r})d\boldsymbol{r}. \tag{4.7}$$

Let V be the thermally acceptable configurations of a particular radical and let m give a particular magnetic parameter as a function of the structure, $\boldsymbol{r}$, and let p be the probability density function of the configurations at a given temperature, then, as $\int_V p(\boldsymbol{r})d\boldsymbol{r} = 1$, the mean value theorem implies:

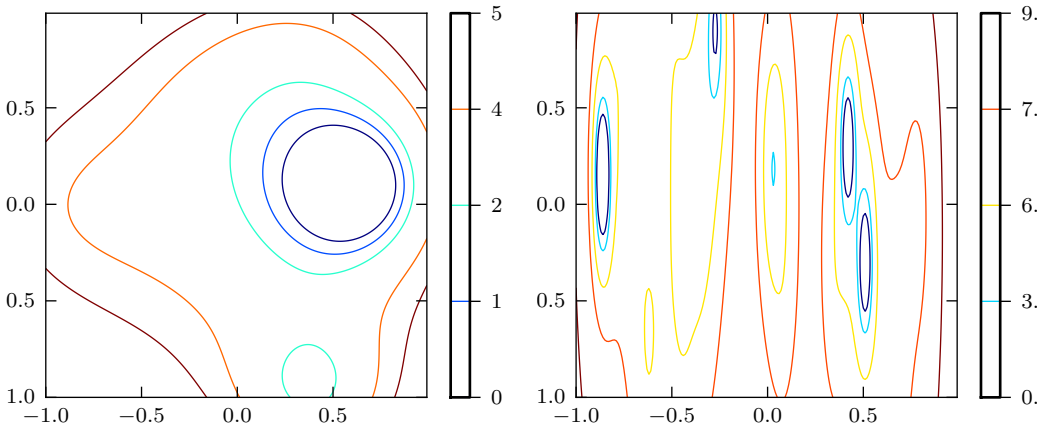
$$\text{there exists } \boldsymbol{x} \in V \text{ such that } m(\boldsymbol{x}) = \int_V m(\boldsymbol{r})p(\boldsymbol{r})d\boldsymbol{r}. \tag{4.8}$$

In other words, there exists a thermally accessible configuration that has the same magnetic parameters as the full vibrational average, and this is the point to which the direct structure fitting procedure will converge.

4.1.4 Internal Coordinates

Gaussian 03 allows input coordinates to be specified in one of two possible ways: as Cartesian coordinates or as internal coordinates (also known as Z-matrix coordinates). It is believed that internal coordinates are, in many cases, a better choice than Cartesian coordinates when performing geometry optimisation to find the energy minimum. *Pulay and Fogarasi* [107] attribute this to internal coordinates reducing the magnitude of some of the cubic or higher terms of the optimisation surface, thus improving the quadratic approximation used by quasi-Newton methods, such as BFGS.

It is, therefore, not a huge stretch to imagine that internal coordinates would be more appropriate than Cartesian for the problem of fitting geometry to a spectrum. The fitting problem is, after all, still partly an energy minimisation problem (see equation 4.3). Consequently, `DiStruct` was written to use structures specified in both types of coordinate representations, and the production runs presented in section 4.2 used internal coordinate representations.



(a) Problem with approximately equivalent curvature in each direction

(b) Problem with higher curvature in the x direction

Figure 4.2 – The error functions for two hypothetical minimisation problems (axis units are arbitrary). The left example has the same approximate curvature in both coordinates and the optimum initial step size that doesn't depend on the direction. However, in the right example, curvature dependent is much higher in the x direction than the y direction. This situation might arise if, say, the x-axis represented displacement in Ångströms but the y-axis represented a bond angle. Information about these differences are contained in the Hessian but by default, BFGS and similar Quasi-Newton methods generally assume the starting Hessian matrix is the identity.

4.1.5 The Effects of Function Curvature and the Initial Hessian on the Performance of BFGS

The performance (especially during the initial steps) of gradient based algorithms such as BFGS is affected by the curvature of the target function in each coordinate. If a function has high curvature in a particular coordinate then only short steps should be taken whereas for low curvature, longer steps can be safely attempted. As the gradient alone contains no information about curvature, gradient based algorithms begin blind to the curvature. Curvature may also be different for different coordinates (see figure 4.2)—this is often the case when units are mixed (for example, in a Z-matrix parameterisation, some coordinates represent bond angles and others represent bond lengths).

Ideally the optimiser needs to take the longest steps it can as this saves on function evaluations and to do this, it needs to know the curvature in each coordinate. After sufficient initial steps, the BFGS algorithm is able to estimate the Hessian and so gains knowledge of the curvature but before that estimate is obtained, computing time may be needlessly wasted.

This problem was also encountered during the development of `pcsfit` (see section 5.5.6), where the problem was rescaled based on the magnitudes of initial input vector. This solution was selected due to its simplicity; even though it is formally incorrect—there is no

reason to suppose that the initial values of model parameters have anything to do with the derivatives of the model—but it was found to be adequate in practice for `pcsfit` where the computation cost of each iteration was orders of magnitude lower.

Scaling in `DiStruct` required a more sophisticated treatment, as: a) far more (expensive) CPU time was at stake and b) the initial steps taken by the optimiser often proved too large, resulting in the atoms in the candidate structure being moved far apart. In an ideal world, these "exploded" structures should merely result in very bad spectra, in turn causing the optimiser to shrink the trust region and try smaller steps. In reality, the "exploded" structures tended to cause Gaussian to either crash or otherwise fail, because they are not chemically reasonable. For this reason, the correction of bad scaling, which could be considered an optimisation in the case of `pcsfit`, must be considered both an optimisation and a vital part of obtaining a solution in the case of `DiStruct`.

Two methods of dealing with bad scaling were investigated:

- The first was to artificially stretch the target function based on intuition on what a sensible step size would be. For example, a user might decide that for bond angles, a change on 1 degree counted as "small" but for a bond length a change of 0.1 Ångströms was "small". `DiStruct` would then use this information to perform the stretching and the optimiser would see a function for which the default identity Hessian was a more reasonable guess.

- Provide an initial guess Hessian based on the Hessian of the DFT energy, which Gaussian is capable of computing at a reasonable cost [108].

Of these two methods, the second was found to be superior, which is perhaps not surprising in retrospect as the problem is already partly an energy minimisation problem due to the regularisation. The DFT Hessian also has the advantage that it can be computed automatically and efficiently without additional user input, whereas the first method would have required the user to provide hints as to good initial step sizes.

4.1.6 Parallelism

Due to the four point order finite differencing used, each optimiser step that requires a gradient evaluation results in the creation of $4n + 1$ individual, independent Gaussian jobs that may be evaluated in parallel, where n is the number of degrees of freedom. As each job is independent, the problem is embarrassingly parallel and the production code takes full advantage of this to speed the calculation.

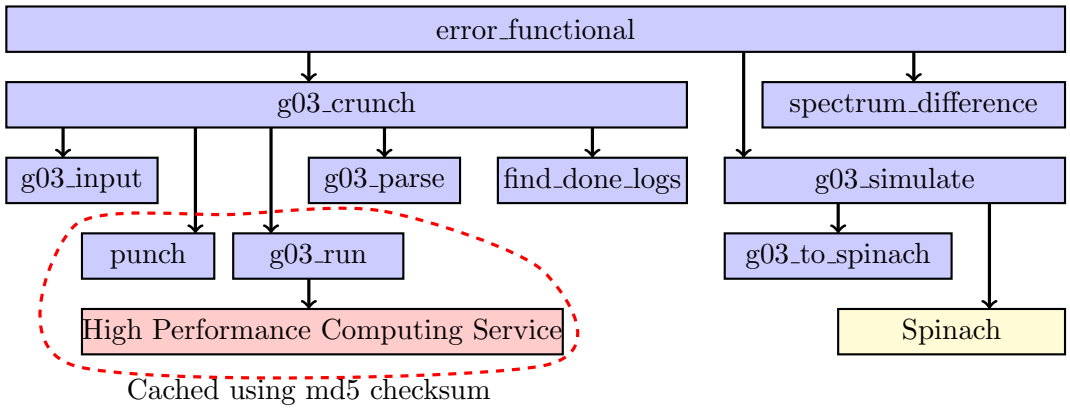


Figure 4.3 – Typical call graph of the `DiStruct` error functional. Time proceeds from left to right. Each box represents the lifetime of a particular function, and child boxes represent calls made from that function.

4.1.7 MD5 Caching

During development, and even during the production runs described in the JMR paper [2], a frequent problem encountered was the need to restart an optimisation from the beginning. There were many reasons for this, ranging from the discovery of a new programming error (generally during development) to the random failure of one of the high performance computing (HPC) processes (generally during production). The very large numbers of HPC processes spawned by a single production run on a large molecule meant that even a very low failure probability for a single HPC process could translate into a unacceptable failure probability for the entire run. Consequently, the ability to restart a calculation without needlessly recomputing every result was extremely important.

A critical condition for the restart capacity to work is that the optimisation algorithm is deterministic. Here "deterministic" means that an optimisation algorithm, if given the same starting point on two individual runs, will choose to investigate the same point, x_1. If both invocations are then supplied with the same value and gradient, f_1 and v_1 respectively, then both invocations will choose to investigate the same point x_2 next and so on. Both the Nelder Mead simplex algorithm and BFGS are deterministic while, for example, an algorithm based on simulated annealing would not be (unless the random number generator were to be provided with the same seed on each invocation).

During development, a relational database was used to associate hashes of Gaussian input files with their outputs but the production runs described in the JMR paper [2] simply used the operating system file system for simplicity.

4.1.8 Implementation Details

These ideas were implemented in MATLAB code which ran on a workstation. The BFGS implementation was provided by MATLAB's fminunc function. Jobs were dispatched to a remote, high performance computing service. The call graph of a typical invocation of the DiStruct error functional is shown in figure 4.3. This error functional is called by the MATLAB optimiser which is, in turn, evoked by a user script that will typically be different for each candidate molecule.

In figure 4.3, execution begins from the top left in error_functional and proceeds from left to right, hopping down to called functions when they are encountered. So error_functional calls g03_crunch which in turn calls g03_input followed by punch followed by g03_run and so on so that figure 4.3 gives a rough idea of how execution proceeds and which functions depend on other functions. Descriptions of the purposes of each called function, in the order they are encountered, are as follows:

error_functional Is responsible for computing the total error and its gradient.

g03_crunch Is responsible for calling Gaussian and returning the results in a structured from.

g03_input Automatically constructs an input file in memory for Gaussian from atomic coordinates. Called repeatedly if a gradient is required.

punch Writes out the Gaussian input file(s) to disk.

g03_run Sends the Gaussian input file(s) over the network to a high performance computing service and waits for the result.

g03_parse Reads the Gaussian output files and automatically extracts the details of the computed magnetic tensors.

find_done_logs Checks that all returned Gaussian logs files are present and well formed.

g03_simulate Sets up and performs a Spinach spin dynamics simulation and calls the Spinach library. g03_to_spinach marshals the data from the Gaussian log file into a format suitable for Spinach.

spectrum_difference Handles subtracting one spectrum from another. This may involve interpolation as the values of the heights of the spectra are only available at discrete points that may not line up.

Once control returns to `error_functional`, the subtracted spectrum is integrated and the final error calculated. If a gradient is required, more rounds of `g03_simulate` and `spectrum_difference` are performed, four for each coordinate (due to the use of four point finite differencing, section 4.1.2) and four for each of broadening, scaling and offset.

The section enclosed in the dotted line is cached using MD5 hashing on the input (section 4.1.7). When a Gaussian output file is returned, it is renamed to the MD5 hash of the input file that generated it. If, at a later time Gaussian output file is found to have the same name as the MD5 hash of an input file about to be submitted to the high performance computing service, `punch` and `g03_run` are skipped over and the found output file is used instead.

4.2 Key Results

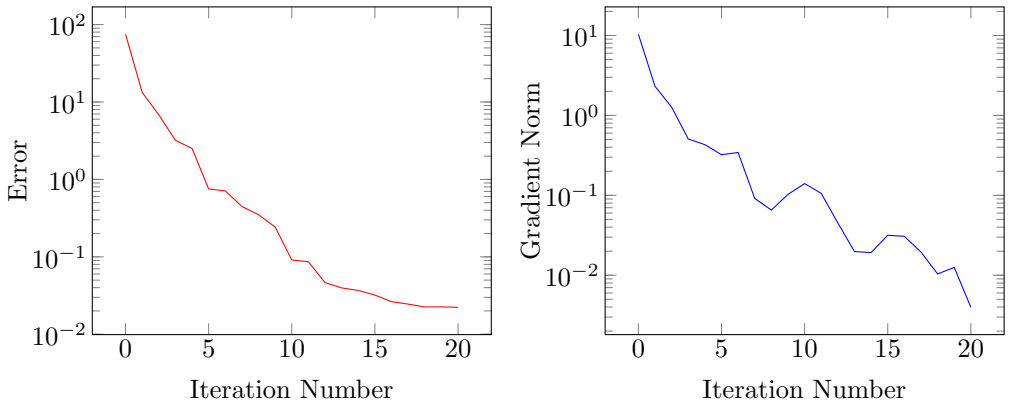
4.2.1 Liquid State EPR

Examples of successful direct structure fitting using liquid state EPR may be found in the JMR paper [2], and will be described in brief here. Spectra for cyanomethyl and propargyl were taken from the SDBS-ESR database. Quantum mechanical calculations were done using the GIAO B3LYP/EPR-II [53–55], implemented in Gaussian 03 [56], while magnetic resonance calculations were done in Spinach v1.0.950 with a large enough basis set to make the resulting spectra essentially exact. The radicals studied were cyanomethyl, propargyl, and tyrosyl. Graphs of the fitting of the error functional and gradient norm are given in figures 4.4, 4.5, and 4.6 respectively.

The cyanomethyl DFT minimum matches the direct structure fitting minimum to within 10^{-3} kJ/mol per degree of freedom and, so, deliberately distorted initial geometry was selected as the starting point for this example. As a consequence, the direct structure fitting procedure does not provide any new structural information, however, this example did illustrate a case of the direct structure fitting procedure working when only three significant figures of accuracy in the gradient could be reliably computed.

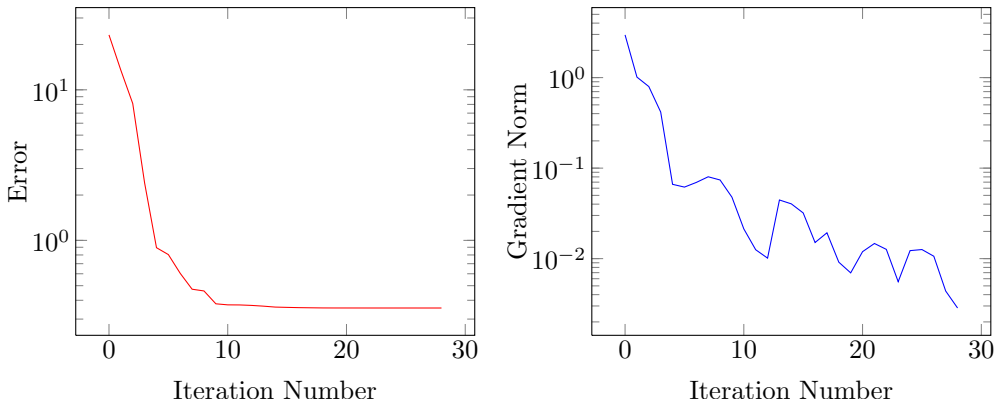
In contrast to the cyanomethyl case, the DFT minimum of propargyl does not yield a good spectrum. However, the direct structure fitting minimum does give a good spectral fit with only a very small perturbation in the geometry. The RMSD displacement was 0.053 Å, at the additional energy cost of 2.78 kJ/mol per degree of freedom. This example serves as a good illustration of the point made in section 4.1.3: the magnetic parameters can shift very rapidly with respect to changes in the geometry, but this rapid rate of change can be beneficial to the direct structure fitting process.

One more complicated example was the tyrosyl radical, (further discussion of tyrosyl can be found in section 4.2.2). Once again, the cumulative integral technique was needed for



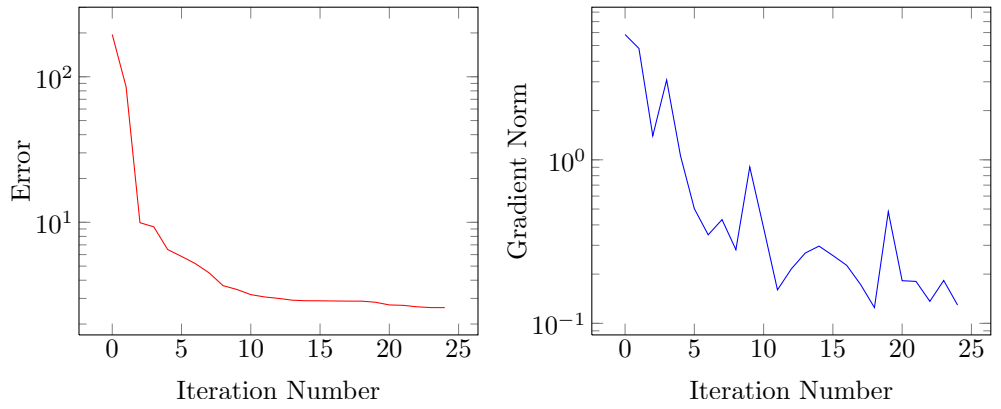
(a) Error Functional vs Iteration Number **(b)** Gradient Norm vs Iteration Number

Figure 4.4 – H_2CNN optimisation error functional and gradient norm vs iteration number.



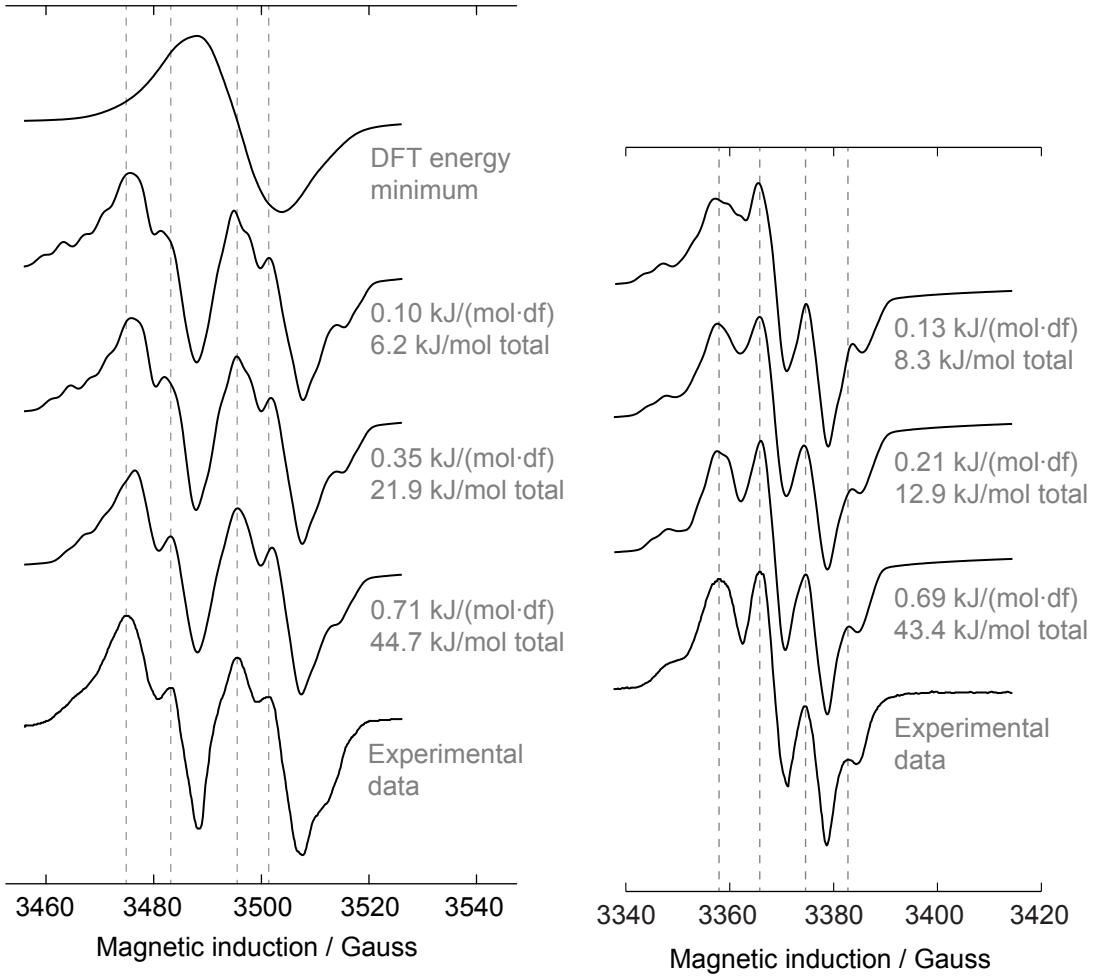
(a) Error Functional vs Iteration Number **(b)** Gradient Norm vs Iteration Number

Figure 4.5 – $HCCCH_2$ optimisation error functional and gradient magnitude vs iteration number.



(a) Error Functional vs Iteration Number **(b)** Gradient Norm vs Iteration Number

Figure 4.6 – Tyrosyl optimisation error functional and gradient magnitude vs iteration number.



(a) E.Coli **(b)** Salmonella

Figure 4.7 – Powder spectra experimental and converged spectra of Salmonella and Escherichia Coli ribonuclease reductase. The abbreviation "df" stands for degree of freedom.

the initial run. Once convergence was obtained using the cumulative integral technique, the geometry was refined using the spectra directly.

4.2.2 Powder EPR Spectra of Embedded Tyrosyl in Proteins

Tyrosyl is the radical formed when the hydrogen is lost from the OH group attached to the ring of the amino acid tyrosine. Tyrosine is one of the most frequent amino acids to form a radical in proteins, appearing in, for example, photosystem II [109, 110]; ribonuclease reductases [111–113]; and, when treated with hydrogen peroxide, soybean leghemoglobin [114], human haemoglobin [115], horse metmyoglobin, [116] and sperm whale metmyoglobin [117]. This list is far from exhaustive.

The ring structure of tyrosyl is able to rotate with respect to the backbone via the C1–C_β bond at very little energy cost with respect to its vacuum state energy minimum and

as such, when it appears in proteins the ring presents itself in many different orientations. Svistunenko [118] points out that this causes particular problems when interpreting EPR spectra of tyrosyl containing proteins. An EPR signal indicates that a radical is present, but quite frequently there are several tyrosines and it is not clear which is responsible for the signal. Svistunenko also points out that even knocking out tyrosines one at a time via mutation may not be sufficient to resolve this problem as this only proves the tyrosine was responsible for the formation of the radical and not that it was the final site.

Svistunenko was able to provide a solution to this dilemma by identifying two main parameters for which knowing the their values is sufficient to predict the observed EPR spectrum [118, 119]. These are the spin density at the C1 carbon and the rotation of the C1–C_β bond. By matching the predicted spectrum of this two parameter model to the observed spectrum, the rotation of the tyrosyl may be inferred and the correct tyrosine may be picked out from a X-ray structure or similar.

Svistunenko's model, while useful, does not generalised to other problems; it is specific to tyrosyl. It does, however, answer the same question as direct structure fitting: take a spectrum and try and find the structure. Presented here is an attempt to perform the same task as Svistunenko's model, that is, obtain a structure for tyrosyl from an experimental spectrum of the radical emended in a protein, in a manner that would generalise.

Two examples of tyrosyl radicals embedded in ribonuclease reductase (RNR) proteins were studied to provide examples of direct structure fitting using powder state EPR spectra. These radicals were chosen as both X-ray structures [120, 121] and EPR based structures [118, 119] are known.

As powder averaging is a linear process, no modifications of the overall fitting procedure were required to be made. In each case, the starting point was the vacuum DFT energy minimum. It is important to note that DFT was not performed on the complete protein—that would be completely impractical—but on the isolated residue. The extra structural information carried in the experimental spectrum is what is expected to force the tyrosyl into the same configuration as it appears when embedded in the protein.

Powder averaging was performed using Lebedev grids [122] which are built into Spinach. Even with such averaging, the cost of the spin dynamics simulation was small enough to be negligible compared to the quantum chemistry step despite the need for powder averaging.

The fitting procedure itself was carried out in two stages. Firstly, the integrated spectra were fitted against each other allowing the offset and spectral window size to vary. This step ensured that all the interesting details of the theoretical spectra would be pulled into the window (in fitting of spectra, rather than integral of spectra, peaks that have disappeared off the low or high frequency ends of the spectra do not contribute to the final error). In

the second stage, fitting was performed using pure spectra rather than integrals of spectra. Generally, this mode of fitting is more sensitive to fine details than the fitting of integrals.

The first example was *Escherichia coli* RNR from the PDB coordinate file 1RIB/Tyr122 [120] (figure 4.7a). The fragment of the structure examined contained 21 atoms for a total of 63 thermally accessible degrees of freedom. Fitting was performed using three regularisation parameters corresponding to 10^4kJ/mol, 10^3kJ/mol, and 10^2kJ/mol full spectral mismatch energies (see section 4.1.1.1). As expected, increasing the amount of regularisation results in less accurately reproduced spectra as the system paying a higher price to move away from the DFT minimum.

The fit achieved at the lowest level of regularisation is nearly perfect, but comes at an energy cost of 44.7kJ/mol (0.71kJ/mol per degree of freedom), while the fit at the highest level of regularisation costs just 6.2kJ/mol (1.2kJ/mol per degree of freedom). Using the equipartition theorem, an energy per degree of freedom can be translated into a characteristic temperature at which that amount of energy becomes available for excitations of the degree of freedom. The lowest level of regularisation corresponded to a characteristic temperature of 170K, and the highest 23.6K. Both of these are thermally accessible at room temperature (assumed to be 297K). We conclude that at room temperature, the perturbation of the structure from the DFT minimum is small enough to seem plausible.

The second example used RNR from *Salmonella Typhimurium*, with atomic coordinates given in the PDB file 1R2F/Tyr105 [121] (figure 4.7b). Once again, an acceptable spectral fit is achieved even at the highest level of regularisation, costing only 0.13kJ/mol per degree of freedom (giving a characteristic temperature of 31 Kelvin—well within room temperature accessibility). The final structure had a root meed squared displacement from the X-ray crystal structure of 0.107Å in the carbons only. Again, a very small and energetically reasonable perturbation from the DFT minimum is all that is required to explain the experimental spectrum.

In summary, for both examples a structure was found that was easily thermally accessible at room temperature and consistent with the observed spectra of the two proteins. No special model or treatment needed to be given, consequently, these fits provide an example of a situation were direct structure fitting could be of practical use of use in a non-toy system. The disadvantage, however, is the very large computational cost involved in the quantum chemistry. Each spectra took about 50 iterations each requiring fourth order finite differencing to be performed for each of 63 coordinates whenever a gradient was required by BFGS. In total, 25,200 quantum chemical calculation were required, each of which took about half an hour on a SGI Altix 4700 supercomputer. Calculations on this scale are far outside of the capacities of desktop hardware; for the moment, direct structure fitting is

a procedure that requires access to high performance computing. However, Moore's law is expected to continue for at least a few more years which may result in direct structure fitting calculations looking increasingly practical, especially given how easily they can be parellelised.

4.3 Potential Refinements of `DiStruct`

This section describes ideas that were experimented with in the `DiStruct` prototype code, but did not appear in the final implementation. The purpose of this section is to outline these ideas so that they might potentially be revisited in future versions of `DiStruct`.

4.3.1 Generalised Coordinate Systems

Direct structure fitting calculations are currently very computationally demanding, so it is worth considering ideas that might reduce the computational effort required by each iteration. One such method that was explored during development, but did not prove necessary for obtaining the numerical examples presented in the paper, was the use of "generalised coordinates". Generalised coordinates are often helpful in reducing the number of degrees of freedom visible to the optimiser and the dimensionality (and hence cost to compute) of the gradient vector. Examples of such problems include:

- The situation where a radical is known to possess a certain symmetry. In this case, the structure may be expressed in fewer than the usual $3n - 6$ real numbers.
- A certain section of the structure of a radical does not affect the EPR spectrum (perhaps it is known through electronic structure calculations that the unpaired electron density in this region is zero) and so should be kept rigid.
- In some cases, there may be a more natural coordinate system in which to express the optimisation problem in, rather than either the Cartesian and Z-matrix coordinate systems provided by Gaussian. Note, however, that the Z-matrix coordinate system is known to already be an excellent choice of coordinate system for many problems (see section 4.1.4).

The concept of generalised coordinates is an idea that originally arose in the study of analytical mechanics (see, for example, the book of *Hand and Finch* [123] or similar) where they are used to remove redundant degrees of freedom from problems in mechanics. While it is sometimes possible to achieve the same effect by feeding constraints to the `MATLAB` optimisation toolbox using `fmincon`, this method is far more limited and still requires the full $3n - 6$ components of the gradient vector to be computed.

"Forward" transforms:

- `Coords`: Input argument, a 3 by n matrix containing the positions of all the atoms in either Cartesian or Z-matrix form.
- `params`: Output argument, a vector of parameters which should be fed to the optimiser.
- `invariants`: Output argument, an opaque object which is passed to the corresponding reverse coordinate transform. This object is used to record any fixed information needed by the reverse transform for reconstructing the original coordinates. See section 4.3.1.2 for an example of its use.

"Reverse" transforms:

- `params`: Input argument, a vector of parameters (possibly varied by the optimiser) which corresponds to the `params` output argument of the previous coordinate transform.
- `invariants`: Opaque object which corresponds to the second output argument of the corresponding "forward" coordinate transform.
- `coords`: Output argument, a 3 by n matrix containing the positions of all the atoms in either Cartesian or z-matrix form.

Figure 4.8 – The signatures of forward and reverse transform functions.

A motivating example for including the capacity to process generalised coordinates in `DiStruct`, one that would not be possible using simple constraints, comes from the papers of *Svistunenko et. al.* [118, 119], which deals with the EPR spectrum of the tyrosyl radical. Svistunenko notes in [118] that the observed experimental spectrum of tyrosyl radicals, taken over a large number of proteins, depends only on two parameters: the spin density at the C1 carbon and the rotation angle of the ring (see figure 4.10). As one particular bond angle has been identified as being particularly important, it would be an advantage to have a way to specify that an optimisation or a scan should be performed using only this parameter *without* modifying the core code of `DiStruct`.

Early builds of the `DiStruct` prototype included a mechanism for specifying a generalised coordinate system, but this capacity proved to be unnecessary for the production examples given in the final paper [2] and in the results section (section 4.2). Nevertheless, the system will be briefly outlined here.

Generalised coordinates in `DiStruct` were specified as a pair of `MATLAB` function handles. The corresponding functions accept(output) a $3 \times n$ matrix of coordinates and output(accept) a flat vector of parameters suitable for feeding to `MATLAB`'s `fminucon` or `fmincon` functions. The "forward" transform is called just before control is passed to the

optimiser and the "reverse" transform is called inside the target function just before the Gaussian input file is constructed.

For simple cases, the above procedure is sufficient, however, it is easy to imagine situations where information is lost in the forward coordinate transform that would be needed to reconstruct the structure of the original molecule (e.g. a coordinate transform fixes bond lengths, reducing the number of degrees of freedom from $3n-6$ to $2n-6$, and loses information about the bond lengths, see section 4.3.1.2). For this reason, "forward" transforms output a second output parameter called `invariants` that contains quantities that need not be varied, but are still important to the reconstruction of the structure. This second parameter is an opaque object, that is, the calling functions outside the coordinate transform pair do not try to read or modify it, they merely pass it on. This choice was made so that any authors of coordinate transforms would be free to use any structure they deem appropriate for storing the information. For example, it is appropriate for the coordinate transform that fixes bond length to return the bond lengths as a vector. Other transform pairs may wish to exchange a matrix or, perhaps, a more complex hierarchical data structure. The resulting pair of functions have signatures given in figure 4.8.

The very simplest example of a coordinate transform, the identity transform, simply rearranges the length $3n$ vector of coordinates given by the optimiser into a 3 by n matrix of coordinates of each nucleus, as expected by the Gaussian input file construction function. More involved examples of uses of generalised coordinates will now follow.

4.3.1.1 Example: Removing Rotational and Translational Degrees of Freedom

When working in Cartesian coordinates, there will always be six degrees of freedom present which do not affect the internal dynamics of a molecule. These arise because of the translational and rotational symmetries of the Hamiltonian [123, p. 172]. It is possible to define a coordinate transformation that throws away these redundant degrees of freedom and, thus, reduces the dimensionality of the overall problem by 6. The procedure for doing so is depicted in figure 4.9.

4.3.1.2 Example: Fixing Bond Lengths

In the Z matrix parametrisation, just over one third of the degrees of freedom represent bond lengths, which tend to be stiff degrees of freedom when compared to bond angles. We can define a coordinate transform to fix bond lengths and only optimise over bond angles. In the forward transform, coordinates representing length are removed from the Z matrix and stored in the `invariants` structure, while the the remaining Z matrix is flattened into

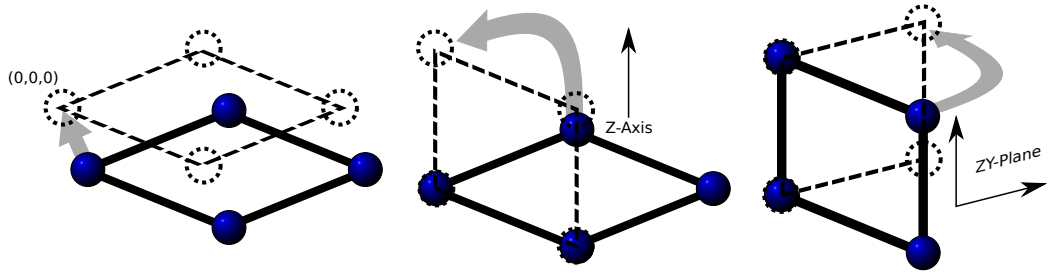


Figure 4.9 – Six degrees of freedom, corresponding to global rotations and translations, can always be removed when dealing with Cartesian coordinates. To perform the reduction, the position of atom 1 is shifted to the origin, atom 2 is rotated onto the z-axis and atom 3 is rotated onto the $x - y$ plane. The zeroed coordinates can then be hidden from the optimiser with no loss of generality.

a row vector. Later, the full Z matrix is reconstructed using the variable parameters and the fixed bond length information stored in `invariants`.

4.3.1.3 Example: Tyrosyl Coordinates

The so-called "Tyrosyl coordinates" provide a good example of a highly problem-specific, generalised coordinate transform. As was discussed in section The ring in the Tyrosyl radical can rotate relatively freely about one of its carbon-carbon bonds (see figure 4.10) and, so, the results of optimising only this coordinate might be interesting. In particular, *Svistunenko* [118] identified this angle to be one of only two parameters (and the only structural one) that determine the EPR spectra of tyrosyl.

Using the generalised coordinate system support, it was possible to instruct `DiStruct` to only optimise this coordinate relatively easily and without modifying the core optimisation code. The setup was achieved by defining a function `fromTyrosylCoordinates` that accepted an angle and used the full `MATLAB` syntax to generate a rotation matrix with which to rotate the relevant half of the radical.

This kind of trick would be much more difficult, if not impossible, when using `fmincon` to specify constraints, but required only 18 lines of `MATLAB` code using generalised coordinates.

4.3.2 SQL Database Layer

During development, communication between the local `MATLAB` process and the remote HPC process was performed through an ad hoc system of SSH tunnels and Windows shares. In addition to being brittle and highly dependent on the particular IT setup under which it ran, under certain conditions the large numbers of HPC processes could interfere with each other, causing race conditions.

A second problem was the sheer number of input and output files being generated; each optimisation cycle invoked something in the order of $3n$ Gaussian processes, with n

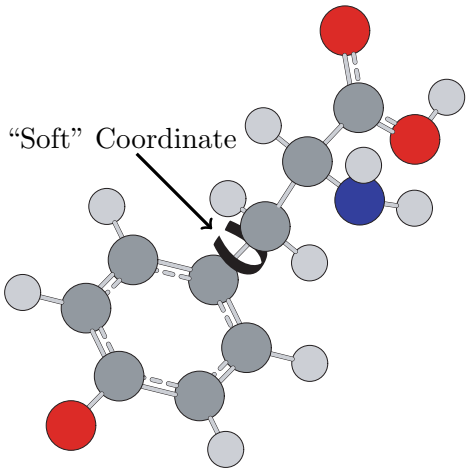


Figure 4.10 – Tyrosyl Radical. Rotation about the indicated bond requires relatively little energy. Svistunenko identified this coordinate [118] as being one of only two parameters that characterised the EPR spectrum of the radical. This observation motivated the development of a systematic method of treating generalised coordinates in `DiStruct`.

being the number of degrees of freedom in the structure (the exact number depended on the coordinate transform used, see section 4.3.1) each with an input and output file, and obtaining convergence required many such cycles. In early versions of the code, managing these hundreds, or even thousands, of files soon became awkward.

In order to improve matters, a SQL (structured query language, a protocol for interacting with databases) database was adopted. Each process could then obtain work by reading the database and assign work or record results by writing to it. All race conditions between processes were then resolved as SQL provides strong guarantees that writes are atomic.[2]

Two additional services provided by the database were the ability to cache the results (see section 4.1.8) and to set timeouts on jobs. To cache the Gaussian results, a table was created containing a hash of every input file encountered, along with the output file it eventually produced. The timeouts were implemented by recording the current time whenever a job was checked out. Other processes looking for jobs checked the recorded time, and if it was too far in the past, the job was eligible to be checked out again.

Thanks to the database, all that was required to retrieve all information about a given optimisation was a single ID, number rather than a collection of output. Using a SQL database proved as simple, or simpler, to set up than a flat file data store and produced better results that were less prone to errors.

[2]In the sense that they are indivisible rather than to do with chemical elements. An “atomic” write either completely succeeds or does not happen without any chance of leaving the database in an invalid, intermediate state.

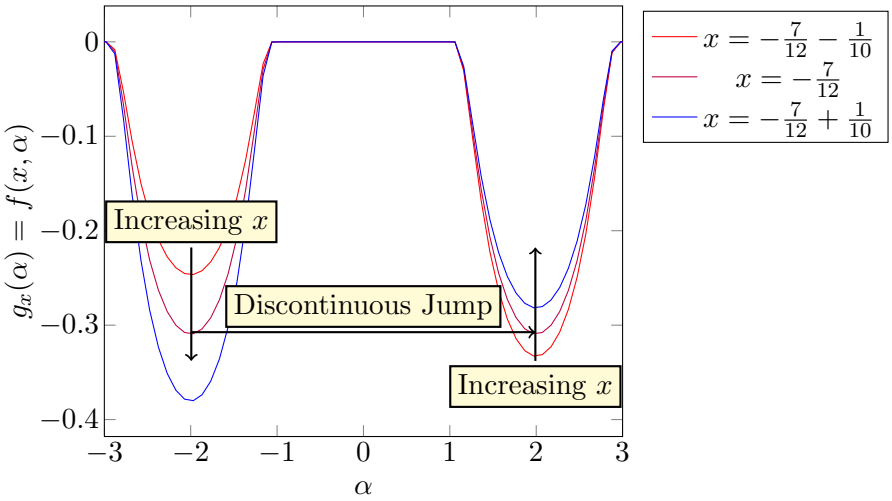


Figure 4.11 – The function $f(x, \alpha)$ plotted with varying α and fixed x for different values of x.

4.3.3 The Micro-optimiser

The idea of performing so called "micro-optimisation" within the main optimisation cycle arose from the observation that recomputing the spin dynamics stage was, in general, cheaper than recomputing the quantum chemistry stage. Therefore, there were three parameters—the relative scaling of the spectra, the spectral offset, and the line broadening—that were easier to vary than the structural parameters.

Mathematically, the use of micro-optimisation techniques transforms the problem of minimising a function, $f(\boldsymbol{\alpha}, \boldsymbol{x})$, with $\boldsymbol{\alpha}$ and $\boldsymbol{x}$ being two vectors of parameters, into the problem of minimising a new function, $g(\boldsymbol{\alpha}) = \min_{\boldsymbol{x}} f(\boldsymbol{\alpha}, \boldsymbol{x})$. The problem of minimising g would appear to be easier than minimising f, as there are fewer model parameters. In practice, micro-optimisation was eventually rejected as a viable part of the final DSF protocol for three reasons that will be described here.

The first reason is mathematical. Gradient based optimisation techniques work by approximating the function in some neighbourhood based on the lowest order terms of a function's Taylor series (for example, the method of steepest descent uses the zeroth and first terms, while Newton's method also uses the second term) and so the function should, ideally, be sufficiently smooth to do this. If we assume that $f(\boldsymbol{x}, \boldsymbol{\alpha})$ is smooth, then we would hope that $g(\alpha)$ would be smooth as well, but this may not the case; g may not even be continuous.

We will now construct an illustrative counterexample to the "$f(\boldsymbol{x}, \boldsymbol{\alpha})$ is smooth $\Rightarrow$ $g(\boldsymbol{\alpha})$ is smooth" conjecture. Let f be a function of two real variables, α and x:

$$f(x, \alpha) = -\operatorname{bump}(\alpha - 2)\exp(-(x+1)^2) - \operatorname{bump}(\alpha + 2)\exp(-(x - \frac{1}{2})^2 + 1), \qquad (4.9)$$

where:

$$\text{bump}(x) = \begin{cases} 0 & \text{when } \ x < -1 \text{ or } x > 1 \\ \exp((1-x^2)^{-1}) & \text{otherwise.} \end{cases} \tag{4.10}$$

f has been constructed with certain properties that will make the following discussion easier. With a fixed x, $g_x(\alpha) = f(x, \alpha)$ has two deep wells with local minimum at $\alpha = \pm 2$ regardless of the value of x, but the depth of these minima do depend on x. The global minima of $g_x(\alpha)$ is at $\alpha = 2$ whenever $x > -\frac{7}{12}$, but is instead at $\alpha = -2$ when $x < -\frac{7}{12}$. At $x = -\frac{7}{12}$, the minimum are the same depth. $\min_\alpha f(x, \alpha)$ is therefore discontinuous around $x = -\frac{7}{12}$ (see figure 4.11).

The second, more practical, problem arises because of the very high degree of reliability that is required of the inner optimisation algorithm (which may be an unreasonable demand to make on an algorithm that is not an "exact arithmetic algorithm" [104]). Once again, let $f(\boldsymbol{x}, \boldsymbol{\alpha})$ be the objective function and let $g(\boldsymbol{\alpha}) = \min_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{\alpha})$. We will assume that g somehow avoids the problems discussed above and does not suffer from discontinuities. Let p be the probability that the micro-optimiser finds the global minimum and does so to sufficient precision. If a central differencing method is used to find the gradient of f with respect to each of its parameters, the probability of the correct gradient being obtained is p^{2n}, as every micro-optimiser run must obtain the global minimal. Even if the micro-optimiser is good, with $p = 0.9$, the incorrect gradient will be obtained more than 50% of the time if there are more than four or more parameters. So, the micro-optimiser must not be merely good; it has to be excellent.

The above argument is highly simplistic, especially as it treats the chance of finding the global minima as being independent of the input parameters. It nevertheless serves to highlight how important it is that the micro-optimiser works and works every time.

The final argument against using a micro-optimiser is that using one increases the complexity of the fitting software, so we would expect a return on the investment of including one, but the actual gain is questionable. For small problems where the structure is only parameterised by a few parameters, the reduction in the dimensionality of the search space can be significant (for example, when using tyrosyl coordinates that were described in section 4.3.1.3, using a micro-optimiser halves the size of the problem). However, it is likely that the most common use case of direct structure fitting would involve larger numbers of degrees of freedom (see the examples below), and the relative gain from using a micro-optimiser stage soon shrinks.

4.4 Conclusion

The results presented here demonstrate that directly fitting structures against their spectra is feasible, at least in the case of molecules that are sufficiently small to be amenable to treatment with quantum chemical software packages. The resulting optimised structures can be expected to be a better reflection of reality than plain DFT minima, due to the additional experimental information from the spectra. This point is illustrated nicely by the case of the tyrosyl radical embedded in a protein; the total energy of the system only weakly depends on the rotation of the ring, but the spectrum depends strongly on it, and, consequently, the ring rotation angle is much more reliably obtained using direct structure fitting than via a DFT minimum alone.

Direct structure fitting techniques are still very computationally demanding, in part because of the forth-order finite differencing scheme required to compute the gradient. The situation would be improved greatly if analytical gradients of the magnetic parameters were to become available in quantum chemical packages. It is hoped that the examples of the JMR paper [2] may provide stimulation for research into providing those analytical gradients.

Chapter 5

Extended Pseudocontact Shift

5.1 Introduction

An estimated one third of all proteins are metalloproteins, of which a significant number are paramagnetic [45], so the question of how to apply NMR is an important one. Unfortunately, standard techniques are often difficult to use when dealing with paramagnetic metalloproteins, due to fast relaxation caused by the dipolar coupling of the unpaired electron to the magnetic nuclei.

Fortunately, special techniques for dealing with paramagnetic metalloproteins have emerged. The first example of a structure refinement in a paramagnetic protein was published by *Banci et al.* [124]. Since then, others have been successful at using pseudocontact shift (PCS) measurements to provide structural constraints used to determine the structures of a wide variety of proteins and other biomolecules, such as Cytochrome C [125, 126], oxidised ferredoxin [127], Calbindin D_{9K} [128], E. coli arginine repressor [129].

Typically, the unpaired electron is approximated as a point dipole sitting at a single point in space (see, for example, the reviews by *Bertini et al* [45, 46]). The aim of the work presented in this chapter is to attempt to model the paramagnetic electron more realistically as a distributed dipole, rather than a point.

5.1.1 Magnetic Susceptibility

When a paramagnetic substance is placed in an external field, a weak additional field is induced within the substance. At the simplest level, this additional field can be thought of as arising because the paramagnetic substance contains unpaired electrons which have associated magnetic dipoles. These magnetic dipoles find it more energetically favourable to align with the external field, and, once aligned, they contribute an extra "induced" magnetic field.

In paramagnetic proteins, the additional induced field is much weaker than the external field, so its ability to align additional dipoles is typically ignored—in contrast with ferromagnetic substances, where the induced field is sufficiently strong that any aligned dipoles will align more dipoles, causing a cascade effect that results in all dipoles becoming aligned.

A more detailed discussion of paramagnetism requires mathematical notation, so some terminology will be defined first. Physically, the external field is the magnetic field that would exist if the paramagnetic material were to be replaced with a vacuum. It will be modelled by a vector field, $\boldsymbol{B}_0$, that will associate a vector with every point in space. It is standard notation in electromagnetism to define a second vector field, $\boldsymbol{H}_0$, as $\boldsymbol{H}_0 = \frac{\boldsymbol{B}_0}{\mu_0}$ where μ_0 is the permeability of free space.

Now, assume that the induced field is incapable of aligning more dipoles—that is, dipoles are only able to "see" $\boldsymbol{B}_0$. We will call the induced field in this case $\boldsymbol{B}_1$, which will be another vector field. We could continue and suppose that dipoles were now only able to "see" the field $\boldsymbol{B}_0 + \boldsymbol{B}_1$, causing more overall alignment and call the second additional field $\boldsymbol{B}_2$, and so on, to build an infinite series of vector fields. The true field would be the limit of this series if it were to converge. In practice, paramagnetism is typically so weak that only the $\boldsymbol{B}_0$ and $\boldsymbol{B}_1$ are significant, while ferromagnetism can be thought of as corresponding to a divergent series. It is standard notation in electromagnetism to define a quantity called magnetisation as $\boldsymbol{M} = \frac{\boldsymbol{B}_1}{\mu_0}$.

At the macroscopic level for an infinite substance:

$$\boldsymbol{M} = \chi_V \boldsymbol{H}_0, \tag{5.1}$$

where χ_V is a dimensionless property of the substance, to be determined by experiment. For a finite-sized substance with a boundary, equation (5.1) becomes an approximation that breaks down towards the edges of the substance. However, in paramagnetic substances, the external field is much stronger than the induced filed; the external field's aligning effect on $\boldsymbol{M}$ will be much stronger than any edge effects, due to further induction by $\boldsymbol{M}$, so $\boldsymbol{M}$ and $\boldsymbol{H}$ can be assumed to be parallel, which would not be the case in ferromagnetic materials.

The quantity χ_V is called the volume susceptibility and is useful for dealing with bulk materials, but when working at the level of molecules, the molar susceptibility, χ_M, is often more useful. These quantities are related by:

$$\chi_M = \frac{m}{\rho}\chi_V, \tag{5.2}$$

where ρ is the density and m is the molar mass. The SI unit for molar susceptibility is seen to be $\mathrm{m}^3\mathrm{mol}^{-1}$.

Understanding susceptibility further requires an examination of the substance at the microscopic level. At this level, materials can no longer be treated as uniform, and we instead think of an external field inducing a dipole in a particle. In this case, the relationship between the molar susceptibility and the strength of a single induced dipole is [45]:

$$\langle \boldsymbol{\mu} \rangle = \frac{\chi_M \boldsymbol{B}_0}{\mu_0 N_A}. \tag{5.3}$$

Consider a free particle possessing a spin that is placed into a magnetic field. We will treat the particle quantum mechanically, while the magnetic field will be treated classically under the assumption that the classical induced field is the same as that induced by the ensemble of classical particles that corresponds to the quantum particle.

The field picks out a direction in space (conventionally taken to be the z-axis); the eigenstates of $\hat{S}_z$ remain energy eigenvalues, while the eigenstates of other operators $\hat{S}_{\neq z}$ are no longer energy eigenstates. Because the z component of spin is quantised, the energy eigenvalues the spin will sit in one of $2S+1$ energy levels, where S is the total spin.

At a finite temperature, the populations of the energy levels will be distributed according to the Boltzmann distribution, with an excess of spins sitting in levels anti-aligned with the external field and a deficit of spins in levels which are closer to being aligned. Each spin possesses a magnetic moment of with z component $\mu_B g_e m_s$, so has total energy $\mu_B g_e m_s B_0$, where μ_B is the Bohr magneton, g_e is the electron g-factor, and m_s is the component of the total spin in the z direction. The average population of any particular energy level at temperature T and external field B_0 is given by the Boltzmann distribution:

$$|\langle S, m_s | \Psi \rangle|^2 = \frac{\exp\left(-g_e \mu_B m_s B / k_B T\right)}{\sum_{m_s=-S}^{S} \exp\left(-g_e \mu_B m_s B / k_B T\right)}. \tag{5.4}$$

The average total magnetic moment of a single paramagnetic ion is simply the weighted sum of the magnetic moments of each energy level:

$$\langle \mu \rangle = \frac{\sum_{m_s=-S}^{S} g_e \mu_B m_s \exp\left(-g_e \mu_B m_s B / k_B T\right)}{\sum_{m_s=-S}^{S} \exp\left(-g_e \mu_B m_s B / k_B T\right)}. \tag{5.5}$$

In NMR experiments, the temperature is typically high with respect to the Zeeman splitting and, so, for most purposes $g\mu_B m_s B \ll k_B T$, and the exponentials may be approximated by the first two terms of their Taylor series:

$$\langle \mu \rangle \approx \frac{\sum_{m_s=-S}^{S} g_e \mu_B m_s \left(1 - g_e \mu_B m_s B / k_B T\right)}{\sum_{m_s=-S}^{S} \left(1 - g_e \mu_B m_s B / k_B T\right)}. \tag{5.6}$$

By a suitable rearrangement of terms, and noting that $\sum_{-S}^{S} 1 = 2S+1$, $\sum_{-S}^{S} m_s = 0$ and $\sum_{-S}^{S} m_s^2 = \frac{S(S+1)(2S+1)}{6}$ the above formula simplifies to:

$$\langle \mu \rangle = -\frac{g_e^2 \mu_B^2 B_0}{3 k_B T} S(S+1). \tag{5.7}$$

The induced magnetic moment per mole is simply $\langle \mu \rangle N_A$. By comparison to equation (5.3), the molar susceptibility is clearly:

$$\chi_M = \frac{\mu_0 g_e^2 \mu_B^2 N_A}{3 k_B T} S(S+1). \tag{5.8}$$

Note that from here on in, all mention of susceptibility will refer to the molar susceptibility. On this understanding, the symbol χ will be used in place of χ_M.

5.1.2 Anisotropy in χ

The unpaired electrons in metalloproteins are not free particles; beyond spin, they have an orbital angular momentum which may also contribute towards the total magnetic moment. The $\frac{\mu_B}{\hbar} g_e \hat{S}_z$ term in the Hamiltonian must be replaced by $\hat{\mu}_z = -\frac{\mu_B}{\hbar}(\hat{L}_z + g_e \hat{S}_z)$ to account for this, and a spin-orbit coupling term, $\lambda \hat{S} \cdot \hat{L}$, needs to be added. In the theory of paramagnetic resonance, the effects of orbital angular momentum are usually implicitly included by promoting g to a rank 2 tensor (see section 1.6.6). The g-tensor is an empirical parameter which is included in the spin Hamiltonian, instead of the Zeeman and spin-orbit terms (see any textbook on the subject of EPR, such as [44]). The effective Hamiltonian is now given by:

$$\hat{H}_{\text{eff}} = \frac{\mu_B}{\hbar} \boldsymbol{B} \cdot g \cdot \hat{\boldsymbol{S}}. \tag{5.9}$$

As spin-orbit coupling was being ignored, the orientation of the molecule containing the magnetic dipole with respect to the external field did not matter. Clearly, with the inclusion of spin-orbit coupling via the g-tensor this is no longer the case; the molecular orientation appears in equation (5.9) via the orientation of the g-tensor so a molecular coordinate system needs to be chosen. The two obvious choices are to attach the coordinate frame to either the external field or to the molecule. We will choose the former and select a frame such that $\boldsymbol{B}_0 = (0, 0, B_0)$.

To repeat the derivation of the magnetic susceptibility (section 5.1.1) for systems with an anisotropic g-tensor, we must begin by finding the energy of the various energy levels. For clarity, we will begin by dealing with a spin $\frac{1}{2}$ system—the generalisation to arbitrary spin will be straightforward.

The effective spin Hamiltonian can be seen to be:

$$\frac{\mu_B}{\hbar}\left(0,0,B_0\right)\begin{pmatrix} g_{xx} & g_{yx} & g_{zx} \\ g_{xy} & g_{yy} & g_{zy} \\ g_{xz} & g_{yz} & g_{zz} \end{pmatrix}\begin{pmatrix} \hat{S}_x \\ \hat{S}_y \\ \hat{S}_z \end{pmatrix} = \frac{\mu_B}{\hbar}(g_{xz}B_0\hat{S}_x + g_{yz}B_0\hat{S}_y + g_{zz}B_0\hat{S}_z). \tag{5.10}$$

The energy of the $\left|\frac{1}{2},\frac{1}{2}\right\rangle$ state is given by:

$$\frac{\mu_B}{\hbar}\langle\frac{1}{2},\frac{1}{2}|g_{xz}B_0\hat{S}_x + g_{yz}B_0\hat{S}_y + g_{zz}B_0\hat{S}_z|\frac{1}{2},\frac{1}{2}\rangle, \tag{5.11}$$

but $\hat{S}_x\left|\frac{1}{2},\frac{1}{2}\right\rangle = \frac{1}{2}\left|\frac{1}{2},-\frac{1}{2}\right\rangle$ and $\hat{S}_y\left|\frac{1}{2},\frac{1}{2}\right\rangle = -i\frac{1}{2}\left|\frac{1}{2},-\frac{1}{2}\right\rangle$, which are states orthogonal to $\left\langle\frac{1}{2},\frac{1}{2}\right|$ and, so, the only relevant part of the Hamiltonian is $\mu_B g_{zz}B_0\hat{S}_z$. For systems of higher spin, we can decompose $\hat{S}_x$ and $\hat{S}_y$ into pure sums of raising and lowering operators and note that raising and lowering operators when applied to an energy eigenstate must produce an orthogonal energy eigenstate or annihilate the state, and, consequently, only the $g_{zz}B_0\hat{S}_z$ term matters, so long as we deal with energy eigenstates.

With this knowledge, we can immediately write down the analog of equation (5.4) for systems with orbital angular momentum:

$$\langle S, m_s|\Psi\rangle = \frac{\exp\left(-g_{zz}\mu_B m_s B_0/k_B T\right)}{\sum_{m_s=-S}^{S}\exp\left(-g_{zz}\mu_B m_s B_0/k_B T\right)}. \tag{5.12}$$

The orientation effects enter into this equation through g_{zz}, which changes as the molecule rotates with respect to $\boldsymbol{B}_0$.

The analog of equation (5.5) is:

$$\langle\mu\rangle = -\frac{\sum_{m_s}^{S}\mu_B g_{zz}m_s\exp\left(-g_{zz}\mu_B m_s B_0/k_B T\right)}{\sum_{m_s}^{S}\exp\left(-g_{zz}\mu_B m_s B_0/k_B T\right)}, \tag{5.13}$$

which may be evaluated in an exactly analogous way to equation (5.5) to obtain:

$$\chi_{zz} = \frac{\mu_0 g_{zz}^2\mu_B^2}{3k_B T}S(S+1). \tag{5.14}$$

The susceptibility has now gained a tensor character. From equation (5.14), it may appear that only the zz component of χ matters and other components are somehow unobservable but, in principle, the other components could be measured by rotating the molecule that the tensor is attached to, causing the tensor components to mix with each other. It is important to note that the susceptibility tensor is generally assumed to be symmetric [45] and its trace is unobservable via PCS measurements (shown in section 5.1.3).

5.1.3 The Point Dipole Model

Molecules in solutions tumble in an essentially random manner. We will make the assumptions that this tumbling is both isotropic and slow compared to the electron relaxation (which typically occurs in microseconds to nanoseconds). The fast electron relaxation will allow us to assume that the instantaneous electron configuration in a rotating protein at a particular instantaneous orientation matches that of a static protein in the same orientation.

In the point dipole model there are two major actors that come into play: the unpaired electron and the nucleus where the dipole shift is measured. Here, the electron will be assumed to sit at the origin. The induced field of the electron (the $\boldsymbol{B}_1$ field) is assumed to have a point dipole character described by the vector $\boldsymbol{\mu}$. It is:

$$\boldsymbol{B}_1 = \frac{\mu_0}{4\pi}\left(\frac{3(\boldsymbol{\mu}\cdot\boldsymbol{r})\boldsymbol{r}}{r^5} - \frac{\boldsymbol{\mu}}{r^3}\right). \tag{5.15}$$

The electron dipole is induced by the external field, $\boldsymbol{\mu} = \frac{\chi\cdot\boldsymbol{B}_0}{\mu_0}$. Substituting this into the above equation:

$$\boldsymbol{B}_1 = \frac{1}{4\pi}\left[\frac{3\boldsymbol{r}^{\mathrm{T}}\cdot\chi\cdot\boldsymbol{B}_0}{r^5}\boldsymbol{r} - \frac{\chi\cdot\boldsymbol{B}_0}{r^3}\right]. \tag{5.16}$$

The interaction energy between a dipole and a magnetic field is $-\boldsymbol{\mu}\cdot\boldsymbol{B}$, so the contribution to the total energy of a nucleus from the dipole field induced by the electron is:

$$E = -\frac{1}{4\pi}\left[\frac{3(\boldsymbol{r}^{\mathrm{T}}\cdot\chi\cdot\boldsymbol{B}_0)(\boldsymbol{\mu}_I^{\mathrm{T}}\cdot\boldsymbol{r})}{r^5} - \frac{\boldsymbol{\mu}_I^{\mathrm{T}}\cdot\chi\cdot\boldsymbol{B}_0}{r^3}\right], \tag{5.17}$$

where $\boldsymbol{\mu}_I$ is the magnetic moment of the nucleus. Absolute energies are not generally observable in spectroscopy; we are interested in the energy differences. Let ΔE be the energy difference between the system with no $\boldsymbol{B}_1$ field and the system with the $\boldsymbol{B}_1$ field turned on and the nuclear spin aligned with the external field (so that $\boldsymbol{\mu}_I = \boldsymbol{e}_z\mu_I$ where $\boldsymbol{e}_z$ is a unit vector in the direction of the z-axis). This additional shift matches the the definition of the PCS:

$$\Delta E = -\frac{1}{4\pi}\left[\frac{3(\boldsymbol{r}^{\mathrm{T}}\cdot\chi\cdot B_0\boldsymbol{e}_z)(\mu_I\boldsymbol{e}_z^{\mathrm{T}}\cdot\boldsymbol{r})}{r^5} - \frac{\mu_I\boldsymbol{e}_z^{\mathrm{T}}\cdot\chi\cdot\boldsymbol{e}_z B_0}{r^3}\right]. \tag{5.18}$$

The constant factors and the $\boldsymbol{e}_z$ unit vectors may be factored out using the identity $(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b})\cdot(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{d}) = \boldsymbol{a}^{\mathrm{T}}\cdot(\boldsymbol{b}\otimes\boldsymbol{c}^{\mathrm{T}})\cdot\boldsymbol{d}$:

$$\Delta E = -\frac{\mu_I B_0}{4\pi}\boldsymbol{e}_z^{\mathrm{T}}\cdot\left[\frac{3\boldsymbol{r}\otimes(\boldsymbol{r}^{\mathrm{T}}\cdot\chi)}{r^5} - \frac{\chi}{r^3}\right]\cdot\boldsymbol{e}_z, \tag{5.19}$$

where $\otimes$ is the matrix-matrix Kronecker product.

Equation (5.19) gives the dipolar shift energy for a static protein. However, in solution, proteins tumble about much more quickly than the timescale of a typical NMR experiment and so the actual shift will be the average over all orientations.

There are several methods for performing this averaging over $SO(3)$. In his book, Bertini gives a direct treatment dealing with susceptibility tensors with axial components only [44] (this method was outlined in section 1.6.8). A more elaborate and general method is to examine the $SO(3)$ averaging of a tensor as a separate sub-problem:

$$T_{\text{average}} = \int_{SO(3)} R(\Omega) T R^{-1}(\Omega) d\mu(\Omega). \tag{5.20}$$

Where $R(\Omega)$ is the rotation matrix representation of orientation $\Omega \in SO(3)$, $d\mu(\Omega)$ is the infinitesimal Haar measure of $SO(3)$ and T is any 3×3 tensor. When the $SO(3)$ group is parameterised by Euler Angles $d\mu(\Omega) = \frac{\sin\beta}{8\pi} d\alpha d\beta d\gamma$. It can be shown (via evaluation of equation (5.20) in explicit coordinates, either by hand or with a computer algebra package) that $T_{\text{average}} = \frac{1}{3}\operatorname{Tr}(T)\mathbb{1}$ where $\mathbb{1}$ is the identity tensor. Thus, ΔE averaged over the rotation group, ΔE_{pcs}, is given by:

$$\begin{aligned} \Delta E_{\text{pcs}} &= -\frac{\mu_I B_0}{4\pi} \boldsymbol{e}_z^{\mathrm{T}} \frac{1}{3}\mathbb{1} \left(\operatorname{Tr}\left[\frac{3\boldsymbol{r} \otimes (\boldsymbol{r}^{\mathrm{T}} \cdot \chi)}{r^5} - \frac{\chi}{r^3} \right] \right) \boldsymbol{e}_z \\ &= -\frac{\mu_I B_0}{12\pi} \operatorname{Tr}\left[\frac{3\boldsymbol{r} \otimes (\boldsymbol{r}^{\mathrm{T}} \cdot \chi)}{r^5} - \frac{\chi}{r^3} \right]. \end{aligned} \tag{5.21}$$

Notice that, for an isotropic χ tensor, ΔE_{pcs} is zero. This leads to the familiar situation in solution state NMR where dipole-dipole couplings are averaged out and can be ignored. However, if χ is not isotropic, then the coupling of the nuclei to the B_1 field causes a change in the energy required to flip a spin, and the NMR spectral lines are shifted by an additional quantity over the existing chemical shift; this shift is the pseudocontact shift (PCS). Experimentally, PCS shows up as an alteration in the chemical shift and is typically reported in parts per million, but equation (5.21) gives an energy difference. Notice that $-\mu_I B_0$ is the "operating energy" of the spectrometer, so both sides of equation (5.21) can be divided by this quantity a chemical shift:

$$\sigma_{\text{pcs}} = \frac{1}{12\pi} \operatorname{Tr}\left[\frac{3\boldsymbol{r} \otimes \boldsymbol{r}^{\mathrm{T}} \cdot \chi}{r^5} - \frac{\chi}{r^3} \right]. \tag{5.22}$$

5.2 Explicit Forms of the Point PCS Model

Depending on the conventions with which χ and the spatial coordinates are written, a large number of explicit equations for the point dipole model are possible, all of which are equivalent (12 of which are listed in the work of *Bertini et al.* [45]). Two additional, explicit forms of equation (5.22) will be derived in this section, which will prove useful in later analysis.

Equation (5.22) when expanded in Cartesian coordinates is:

$$\sigma_{\text{pcs}} = \frac{1}{12\pi r^5} \text{Tr}\left[\begin{pmatrix} (3x^2 - r^2) & 3xy & 3xz \\ 3xy & (3y^2 - r^2) & 3yz \\ 3xz & 3yz & (3z^2 - r^2) \end{pmatrix} \begin{pmatrix} \chi_{xx} & \chi_{xy} & \chi_{xz} \\ \chi_{xy} & \chi_{yy} & \chi_{yz} \\ \chi_{xz} & \chi_{yz} & \chi_{zz} \end{pmatrix}\right]. \quad (5.23)$$

(where the symmetry of the susceptibility tensor has been made explicit). When multiplied out, the above becomes:

$$\sigma_{\text{point}} = \frac{1}{12\pi r^5}[(3x^2 - r^2)\chi_{xx} + 6xy\chi_{xy} + 6xz\chi_{xz} + (3y^2 - r^2)\chi_{yy} + 6yz\chi_{yz} + (3z^2 - r^2)\chi_{zz}]. \quad (5.24)$$

There appear to be nine adjustable parameters: six tensor components and the x, y, and z spatial variables, while in actual fact, there should only be eight. The extra parameter has arisen because the observed PCS is independent of the trace of χ, and yet this has not been made explicit in the formalism of equation (5.23).

The independence of the observed PCS on the trace by of χ may be seen by varying χ_{xx}, χ_{yy}, and χ_{zz} by an arbitrary constant, α. The relevant terms in equation (5.24) then become:

$$(3x^2 - r^2)(\chi_{xx} + \alpha) + (3y^2 - r^2)(\chi_{yy} + \alpha) + (3z^2 - r^2)(\chi_{zz} + \alpha). \quad (5.25)$$

After multiplying out and collecting terms in α, the coefficient of α is seen to be identically equal to zero.

If χ_{xx}, χ_{yy}, and χ_{zz} are thought of as components of a vector in a three-dimensional vector space, then we are only interested in the projection of that vector onto the subspace where $\chi_{xx} + \chi_{yy} + \chi_{zz} = 0$. The projection matrix onto that subspace clearly has an eigenvector of $(1, 1, 1)$, with an eigenvalue of 0 and two degenerate eigenvectors living in the $\chi_{xx} + \chi_{yy} + \chi_{zz} = 0$ subspace with eigenvalues of 1. Two such orthogonal vectors are $(-1, -1, 2)$ and $(1, -1, 0)$ and we will use these vectors to inform a choice of reparametrisation that makes the independence of the PCS on the trace of χ explicit:

$$\begin{aligned} \chi_{z^2} &= \frac{1}{2}(2\chi_{zz} - \chi_{xx} - \chi_{yy}) \\ \chi_{x^2-y^2} &= 3(\chi_{xx} - \chi_{yy}). \end{aligned} \quad (5.26)$$

The reasoning behind the naming of χ_{z^2} and $\chi_{x^2-y^2}$ will become clear later. These parameters are clearly somewhat related to Axiality and Rhombicity in the special case of a tensor that is not rotated, but otherwise they are best thought of as a set of parameters that

are convenient for fitting with, rather than parameters that highlight features of physical interest. After some rearrangements, the reverse transformations are given by:

$$\begin{aligned}\chi_{xx} &= -\frac{1}{3}\chi_{z^2} + \frac{1}{6}\chi_{x^2-y^2} \\ \chi_{yy} &= -\frac{1}{3}\chi_{z^2} - \frac{1}{6}\chi_{x^2-y^2} \\ \chi_{zz} &= \frac{2}{3}\chi_{z^2}.\end{aligned} \tag{5.27}$$

It can easily be verified that $\chi_{xx} + \chi_{yy} + \chi_{zz} = 0$ for all $\chi_{z^2}, \chi_{x^2-y^2} \in \mathbb{R}$. By comparing the definitions of χ_{z^2} and $\chi_{x^2-y^2}$ to those of axially and rhombicity, it is clear that these parameters are proportional to the axiality and rhombicity of χ in the case where the molecular frame and the eigenframes χ coincide.

After substitution and simplification, equation (5.24) becomes:

$$\sigma_{\text{point}} = \frac{1}{12\pi r^5}[(3z^2 - r^2)\chi_{z^2} + (x^2 - y^2)\chi_{x^2-y^2} + 6xy\chi_{xy} + 6xz\chi_{xz} + 6yz\chi_{yz}]. \tag{5.28}$$

The advantage of this explicit form of the point model is that it has derivatives with respect to all eight parameters that are easy to evaluate analytically, which will later prove useful when performing fitting using gradient based optimisation algorithms, such as BFGS. Furthermore, derivatives with respect to the rotation group parameters have been avoided entirely. This is a particular advantage as derivatives with respect to Euler Angles are known to become numerically unstable, while other parametrisations of the rotation group (such as quaternions) require adding at least one extra redundant parameter.

The terms of equation (5.28) may be written in terms of the $l = 2$ spherical harmonics, given by:

$$\begin{aligned}&\frac{xy}{r^2} = i\sqrt{\frac{2\pi}{15}}(Y_2^{-2} + Y_2^2) \quad \frac{xz}{r^2} = \sqrt{\frac{2\pi}{15}}(Y_2^{-1} - Y_2^1) \quad \frac{yz}{r^2} = i\sqrt{\frac{2\pi}{15}}(Y_2^{-1} + Y_2^1) \\ &\frac{x^2 - y^2}{r^2} = 2\sqrt{\frac{2\pi}{15}}(Y_2^{-2} + Y_2^2) \quad \frac{2z^2 - x^2 - y^2}{r^2} = 4\sqrt{\frac{\pi}{5}}Y_2^0.\end{aligned} \tag{5.29}$$

Thus, each of χ_{xy}, χ_{xz}, χ_{yz}, χ_{z^2}, and $\chi_{x^2-y^2}$ may be seen to be proportional to the coefficient of the real spherical harmonic expansion of σ_{point}. After rewriting, equation (5.28) reads:

$$\sigma_{\text{point}} = \frac{Y(\theta, \phi)}{r^3}, \tag{5.30}$$

where

$$Y = \sqrt{\frac{2\pi}{15}}[(2\chi_{x^2-y^2} + i\chi_{xy})Y_{2,-2} + (i\chi_{yz} + \chi_{xz})Y_{2,-1} + 2\sqrt{6}\chi_{z^2}Y_{2,0} + (2\chi_{yz} - \chi_{xz})Y_{2,1} + (i\chi_{xy} + 2\chi_{x^2-y^2})Y_{2,2}]. \tag{5.31}$$

5.3 The Delocalised Point Model

A key assumption of the point model is that the unpaired electron is concentrated at a single point in space. In reality, electrons are always found delocalised to some degree.

Due to the linearity of the electromagnetic field, a reasonable model of the PCS of a delocalised electron would be the sum of the point model at each point in space, weighted by the electron density function at that point. In effect, this would be the convolution integral:

$$\sigma_{\text{deloc}}(\boldsymbol{x}) = \int_{\mathbb{R}^3} \sigma_{\text{point}}(\boldsymbol{x}')\rho(\boldsymbol{x} - \boldsymbol{x}')d\boldsymbol{x}', \tag{5.32}$$

In spherical coordinates, this would be:

$$\sigma_{\text{deloc}}(\boldsymbol{x}) = \int_{\mathbb{R}^3} \frac{Y(\theta,\phi)}{r^3}\rho_{\boldsymbol{x}}(r,\theta,\phi) \cdot r^2 \sin\theta dr d\theta d\phi, \tag{5.33}$$

where the position vector-valued dummy variable, $\boldsymbol{x}'$, has been written in terms of its spherical components r, θ, and ϕ. The expression $\rho(\boldsymbol{x} - \boldsymbol{x}')$ has been written as $\rho_{\boldsymbol{x}}(\boldsymbol{x}')$ in order to avoid explicitly writing out the vector expression, $\boldsymbol{x} - \boldsymbol{x}'$, in terms of spherical coordinates.

Note that equation (5.33) contains an overall factor of $\frac{1}{r^3}$, so the integrand contains a singularity. Due to this singularity, it is necessary to consider carefully if the integral is well defined before attempting to evaluate it. Unfortunately, the integral will turn out to be undefined in the sense of Lebesgue integration, the most common definition of the integral operation used in physics, whenever $\rho_{\boldsymbol{x}}$ is nonzero at the origin. A second piece of misfortune is that attempting to numerically evaluate an integral that fails to exist in the Lebesgue sense will generally fail when standard numerical integration techniques are applied, so the problem with equation (5.33) is more than just a hypothetical one.

A similar situation comes up in the integration of the hyperfine coupling (equation 1.85). The $\frac{1}{r^3}$ also renders the integral as not Lebesgue. Neese notes this in chapter 6 of [130] and states that, although it is more usual to introduce the isotropic Fermi contact term separately, it drops out naturally as a boundary term when the singular nature of the operators is correctly treated.

But firstly, it should be noted that is might have been possible to ignore the problem of the non-existence of expression (5.33). When ρ is nonzero at the origin, we are trying

to compute the expected PCS of a spin that sits within the electron cloud but, due to fast paramagnetic relaxation [46], the PCS shift is unlikely to be observable. In effect, our function has a hole in it where it is impossible to evaluate (a numerical routine could return NaN). In an ideal world, we would never want to evaluate the function in the hole.

However, ignoring this problem adds complexity to the fitting software as, even though a fit system would probably not require evaluating the PCS at any point where $\rho > 0$, the same cannot be said for intermediate states that the BFGS optimiser might pass through. Code would need to be written to ensure BFGS did not stray into regions where the function could not be evaluated. In contract, taking the principled approach of analysing expression (5.33) mathematically upfront actually requires only a very small quantity of relatively straightforward additional code to implement.

Another reason to pursue a mathematical analysis is the possibility of using a Poisson-type partial differential equation solver to obtain values for expression (5.33) through all space, rather than just at specific points which would be very useful for computing the isosurfaces that would be required for fitting protein structures as well as just the susceptibility tensors. The mathematical analysis that follows is a perquisite of this approach. Further speculation about using Poisson equation solves can be found in section 6.4.1.

We begin by reviewing some basic facts about Lebesgue integration. The definition of the Lebesgue integration operation is built up in four stages by considering four families of functions of increasing generality. At each stage, Lebesgue integration is defined in terms of a Lebesgue integration on the previous stage (with the definition on the first stage, the simple functions, being trivial). The four families of functions are as follows:

1. Simple functions. These are functions expressible as the finite sum $\sum_{i=0}^{N} a_i \phi_{E_i}(x)$ where $\phi_{E_i}(x) = 1$ if $x \in E_i$ and zero otherwise and $\{E_i\}$ is a collection of compact subsets of the domain.

2. Functions that are positive, bounded, and finitely supported.

3. Positive integrable functions. At this level, functions are permitted to have singularities.

4. Lebesgue integrable functions.

Definition 5.1. *The support of a function, f, is the set of points in the domain for which $f(x) \neq 0$, where x is a point in the domain.*

As stated, the definition of the integral of the simple functions is trivial. The Lebesgue integral of the positive, bounded, and finite functions is given in terms of a limit of a series

of simple functions. For example, one suitable series would be the rectangle rule:

$$\int f(x)dx = \lim_{N\to\infty} h\sum_{n=0}^{N-1} f(a+nh) \quad \text{where } h = \frac{b-a}{N}. \tag{5.34}$$

For a general positive function to be Lebesgue integrable, it must be both "measurable"[1] and fulfil:

$$\sup_g \int g(x)dx < \infty, \tag{5.35}$$

where g runs over all type 2 Lebesgue integrable functions such that $0 \leq g(x) \leq f(x)$ for all x and g is bounded and finitely supported. The "sup" operation is the supremum.

Definition 5.2. *The supremum of a subset of the extended real numbers (the set of real numbers along with $+\infty$ and $-\infty$) is the least upper bound of that set (definition taken from [132])*[2]

Theorem 5.3. *Let $\{f_n, f\}$ be a set of positive functions where $f_n(x) \leq f(x)$ and $f_n(x) \to f(x)$ for almost every x, then:*

$$\lim_{n\to\infty} \int f_n(x)dx = \int f(x)dx. \tag{5.36}$$

A proof can be found in Stein's book [131]. Theorem 5.3 is important as type 3 Lebesgue integrable functions were defined via the supremum, which seems to suggest that a search through an infinite number of potential g functions would be needed to be sure of finding the maximum value. This theorem instead allows us to chose any series of functions that converge to f.

We will now begin analysing expression (5.33) by considering a series of simplifications of it, building up the expression itself.

5.3.1 The Integral of $\frac{1}{r^n}$

The first of these will be integrals of the form $\frac{1}{r^n}$ over a 3-dimensional space:

[1] The definition of "measurable" is somewhat technical, and also typically irrelevant in physics, because functions are usually assumed to be smooth and are, consequently, measurable. A function, f, with domain E is measurable if, for all possible a, the set $\{x \in E : f(x) < a\}$ is measurable. Roughly speaking, measurable sets are sets which can be assigned a well-defined hyper-volume (*e.g.* length for 1D sets, area for 2D sets *etc.*). (For a discussion on measurable sets and functions can be found in [131]).

[2] Examples: $\sup\{x > 3\} = \infty$, $\sup\{x < 5\} = 5$, $\sup\{x \leq 5\} = 5$. The supremum behaves very much like the maximum, but notice how the maximum of $x < 5$ is not defined, but the supremum is.

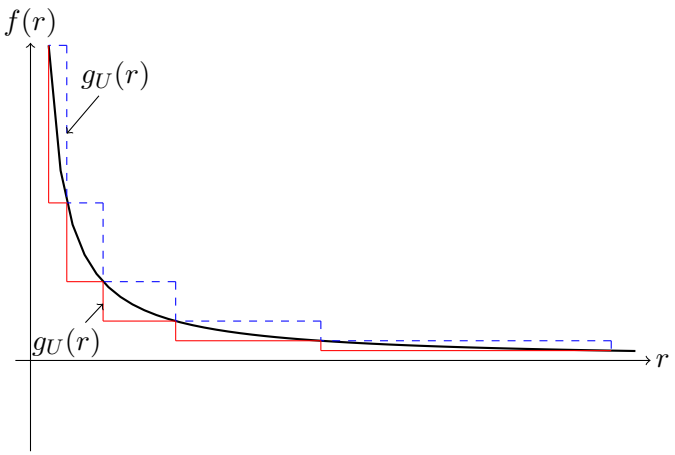


Figure 5.1 – Schematic of the functions f (black), g_U (blue, dashed), and g_L (red, solid) from the proof of theorem 5.4. Notice that $g_U \geq f$, and is thus an upper bound for f, while $g_L \leq f$, and is thus a lower bound.

$$I = \int_{\mathbb{R}^3} \frac{1}{r^n} dr. \tag{5.37}$$

The following argument is an adaptation from a similar argument in Stein and Shakarchi's book [131]. Stein and Shakarchi consider the integral of $\frac{1}{r^{d+1}}$ in d-dimensional space and prove Lebesgue integrals exist outside of all balls containing the origin. Here we only need consider 3-dimensional space, but to do so for arbitrary n and also consider the origin.

Theorem 5.4. *The expression $\frac{1}{r^n}$, with $n \geq 0$, is Lebesgue integrable outside of any ball centred at the origin if and only if $n > 3$ and is Lebesgue integrable over any ball centred at the origin if and only if $n < 3$.*

Proof. We could, of course, proceed by blindly applying rules of elementary calculus, and this would eventually lead to the correct result, but doing so would obscure the reasons behind the existence or non-existence of the integral in expression (5.37). Instead, we examine expression (5.37) at the level of limits.

We begin by splitting $\mathbb{R}^3$ (except for an arbitrary small ball around the origin) into a set of concentric shells, A_k:

$$A_k = \{x \in \mathbb{R}^3 : 2^k \epsilon < r \leq 2^{k+1}\epsilon\}. \tag{5.38}$$

Set $f(x) = \frac{1}{x^n}$. Now let $g_L(x)$ and $g_U(x)$ be two functions such that, if x is in shell A_k, then $g_L(x) = \frac{1}{(2^{k+1}\epsilon)^n}$ and $g_U(x) = \frac{1}{(2^k\epsilon)^n}$. In other words, $g_L(x)$ and $g_U(x)$ resemble a staircases that bound $f(x)$ from above and below, touching $f(x)$ only when $x = 2^k\epsilon$ for some k (see figure 5.1).

Since, $g_U(x) \geq f(x)$, then if the integral of $g(x)$ exists over some bounds, the integral of $f(x)$ must also exist. The conditions for the convergence of $f(x)$ may then be obtained as follows:

$$\int_{\epsilon \leq r \leq 2^N \epsilon} f(x)dx \leq \int_{\epsilon \leq r \leq 2^N \epsilon} g_U(x)dx = \sum_{k=0}^{N} \frac{4}{3}\pi \frac{[(2^{k+1}\epsilon)^3 - (2^k\epsilon)^3]}{(2^k\epsilon)^n} = (2^3 - 1)\frac{4}{3}\pi\epsilon^{3-n}\sum_{k=0}^{N}(2^k)^{3-n}. \tag{5.39}$$

The above implies the integral of f converges as $N \to \infty$ if $n > 3$ and converges as $\epsilon \to 0$ if $n < 3$.

Similar arguments can be used with the lower bound, $g_L(x)$, to obtain conditions for divergence:

$$\int_{\epsilon \leq r \leq 2^N \epsilon} f(x)dx \geq \int_{\epsilon \leq r \leq 2^N \epsilon} g_L(x)dx = \sum_{k=0}^{N} \frac{4}{3}\pi \frac{[(2^{k+1}\epsilon)^3 - (2^k\epsilon)^3]}{(2^{k+1}\epsilon)^n} = \frac{(2^3 - 1)}{2^n}\frac{4}{3}\pi\epsilon^{3-n}\sum_{k=0}^{N}(2^k)^{3-n}. \tag{5.40}$$

Thus, the integral diverges as $\epsilon \to 0$ if $n \geq 3$, and diverges as $N \to \infty$ if $n \leq 3$.

Together these imply that the integral of f converges as $N \to \infty$ *if and only if* $n > 3$ and converges as $\epsilon \to 0$ *if and only if* $n < 3$. □

Corollary 5.5. *For $n = 3$, $\frac{1}{r^n}$ is not Lebesgue integrable if the integral is taken on any unbounded set, or any set that contains the origin.*

As this corollary shows, the integral of $\frac{1}{r^3}$ diverges at both of the limits and, so, $n = 3$ is in some sense the worst case for three dimensions.

5.3.2 The Integral of $\frac{1}{r^3}$ Modulated by a Function that Spherically Averages to Zero

The next integral we will consider is:

$$I = \int_{\mathbb{R}^3} \frac{Y(\theta, \phi)}{r^3} r^2 \sin\theta dr d\theta d\phi, \tag{5.41}$$

where Y is a function defined over a sphere with the special property that it averages to zero on a sphere:

$$\int_{S^2} Y(\theta, \phi) \sin\theta d\theta d\phi = 0. \tag{5.42}$$

Unless $Y = 0$, $\frac{Y}{r^n}$ can and must take on both positive and negative values, so we must move to the fourth and most general definition of Lebesgue integration. Note that we cannot rely on theorem 5.3 directly, as theorem 5.3 only works for positive functions.

The Lebesgue integral of a general function is defined in terms of the integrals of the positive and negative parts separately. Let $f_+(x) = f(x)$ whenever $f(x) > 0$ and zero

otherwise, and $f_-(x) = -f(x)$ whenever $f(x) < 0$ and zero otherwise. f_+ and f_- are both positive, so theorem 5.3 can be applied separately to each part.

$$\int f(x)dx = \int f_+(x)dx - \int f_-(x)dx. \tag{5.43}$$

For a function to be Lebesgue integrable, both the $\int f_+(x)dx$ and $\int f_-(x)dx$ parts must be finite. Unfortunately, this is not the case for $\frac{Y}{r^3}$, except for the trivial case where $Y = 0$.

However, because $\frac{1}{r^3}$ is spherically symmetric and Y averages to zero on a sphere, then $\frac{Y}{r^3}$ averages to zero on every shell about the origin, so:

$$\int_{a\leq r\leq b} \frac{Y(\theta,\phi)}{r^3} r^2 \sin\theta dr d\theta d\phi = 0, \tag{5.44}$$

with $0 < a \leq b < \infty$. It is interesting to note that even though the limit:

$$\lim_{a\to 0, b\to\infty} \int_{a\leq r\leq b} \frac{Y(\theta,\phi)}{r^3} r^2 \sin\theta dr d\theta d\phi = 0, \tag{5.45}$$

exists, the $\frac{Y}{r^3}$ is still not Lebesgue integrable because the limits:

$$\lim_{a\to 0, b\to\infty} \int_{a\leq r\leq b} \left(\frac{Y(\theta,\phi)}{r^3}\right)_+ r^2 \sin\theta dr d\theta d\phi \quad \lim_{a\to 0, b\to\infty} \int_{a\leq r\leq b} \left(\frac{Y(\theta,\phi)}{r^3}\right)_- r^2 \sin\theta dr d\theta d\phi, \tag{5.46}$$

do not exist separately.

In practice, this means that evaluating (5.45) via typical numerical methods, such as using a quadrature scheme, could not be expected to produce a sensible result, because such methods work by subdividing space and approximating the function on each piece (see section 5.4 for a more in depth review of numerical quadrature). Because the positive and negative parts diverge separately, the evaluation will not terminate successfully.

5.3.3 Generalised Functions

If expression (5.45) is not a Lebesgue integral, then what sort of mathematical object is it? It turns out that expression (5.45) is best thought of as a *generalised function*.

The theory of generalised functions, also known as *distribution theory* in the literature, was first formulated by Soboleff and was later systematically developed by Schwartz (the history of the subject is discussed in more depth in *I. M. Gel'fand and G. E. Shilov's* book [133]).

We will begin with some definitions.

Definition 5.6. *A generalised function,* F*, is an object that assigns a number to each function in some space of functions, S, in such a way that:*

$$\mathrm{F}[\alpha f + \beta g] = \alpha \mathrm{F}[f] + \beta \mathrm{F}[g] \quad \alpha \in \mathbb{R} \quad f, g \in S. \tag{5.47}$$

As the application of a generalised function is linear, generalised functions are clearly linear functionals. Generalised functions form a vector space, with the addition and scalar multiplication operations defined as:

$$(\mathrm{f} + \mathrm{g})[\rho] = \mathrm{f}[\rho] + \mathrm{g}[\rho] \quad \text{and} \quad \alpha \mathrm{f}[\rho] = \mathrm{f}[\alpha\rho] \tag{5.48}$$

respectively. f, g are generalised functions, ρ is a function, and α is a scalar.

Definition 5.7. *The function space that a generalised function acts on is called its test function space.*

Typically, the test function space of a generalised function is constrained depending on the application. In this work, it is the electron density functions that take the role of test functions, so we will only be interested in spaces of test functions that can be normalised. Furthermore, we will assume that they decrease as $|x| \to \infty$ faster than any power of $\frac{1}{|x|}$. Test functions that fulfil these restrictions are often referred to as *test functions of rapid descent.*

5.3.4 $\frac{Y}{r^3}$ with a Test Function

The next integral we will analyse takes the form:

$$\frac{Y(\theta,\phi)}{r^3}[\rho] = \lim_{a\to 0, b\to\infty} \int_a^b \int_0^{2\pi} \int_0^{\pi} \frac{Y(\theta,\phi)}{r^3} \rho(r,\theta,\phi) r^2 \sin\theta \, d\theta \, d\phi \, dr, \tag{5.49}$$

with ρ being a test function in the space of bounded test functions of rapid descent. It will eventually represent the distribution of the electron, but here it will simply be a function. The expression inherits the property of not being Lebesgue integrable from $\frac{Y}{r^3}$, but it will turn out to be possible to assign expression (5.49) a value in the sense of equation (5.45), using ideas from the theory of generalised functions.

Theorem 5.8. *The integral:*

$$\int_{|x|>\epsilon} |x|^n \rho(x) d^3x, \tag{5.50}$$

with $\epsilon > 0$ exists in the Lebesgue sense when ρ is a bounded, positive function that is of rapid descent.

Proof. First, note that for any n, if ρ is a rapidly decreasing function, then so is $|x|^n\rho(x)$. Now, as $|x|^n\rho(x)$ is rapidly decreasing, there must be some $a \in \mathbb{R}$ such that $|x|^n\rho(x) < \frac{1}{|x|^m}$ for any m when $|x| > a$. But it was shown in theorem 5.4 that integrals of the form

$\int_{|\boldsymbol{x}|>\epsilon} \frac{1}{|\boldsymbol{x}|^m} d^3x$ exist when $n > 3$. Set $m = 4$, then $\int_{|x|>a} \frac{1}{|\boldsymbol{x}|^4} d^3x > \int_{|x|>a} |x|^n \rho(x) d^3x$. Finally, the $\int_{a\geq|x|>\epsilon} |x|^n \rho(x) d^3x$ part must be finite, as $|x|^n \rho(x)$ is bounded on the region of integration and the region of integration is finite in size. □

Because ρ was selected to vanish faster than any power of $\frac{1}{r}$, we can use theorem 5.8 to take the $b \to \infty$ limit on the right hand side of equation (5.49) without a problem.

Integral Regularisation

An important application of the theory of generalised functions is integral regularisation. The purpose of regularisation is to take a divergent integral and assign it a finite value by removing the divergent part.

Theorem 5.9. *If $f(\boldsymbol{x})$ is a function such that $\int_{|x|>\epsilon} f(\boldsymbol{x})\rho(\boldsymbol{x})d\boldsymbol{x}$ diverges as $\epsilon \to 0$ but for some integer $m > 0$, $\int f(x)|\boldsymbol{x}|^m d\boldsymbol{x}$ does not diverge, then we may construct a regularisation of f as follows:*

$$\mathrm{f}[\rho] = \int f(x) \left[\rho(x) - \left(\rho(0) + \sum_i \frac{\partial \rho(0)}{\partial x_i} x_i + \cdots + \sum_i \cdots \sum_j \frac{\partial^m \rho(0)}{\partial x_i \ldots \partial x_j} \frac{x_i \ldots x_j}{m!} \right) \theta(|x|) \right] dx. \tag{5.51}$$

where θ is a step function, such that $\theta(x) = 1$ for $x \leq 1$ and $\theta(x) = 0$ for otherwise.

Proof. ρ is completely determined by its Taylor series. By subtracting the part in the round brackets, we have removed the low order terms of the Taylor expansion of ρ, leaving only those that contain $|x|$ raised to the power of m or higher. By assumption, $f(x)$ multiplied by such a power is integrable. (Gel'fand and Shilov [133]). □

If f is a regularisation of function $f(x)$ then we may add any generalised function, g, to f without changing the value of f[ρ], so long as $\mathrm{g}(x) = 0$ for all $x \neq 0$.

Regularisation of $\frac{Y}{r^3}$

From theorems 5.4 and 5.8, we know that $\frac{Y}{r^2}\rho$ is Lebesgue integrable, so we can chose $m = 1$ in equation (5.51) to find a regularisation of $\frac{Y}{r^3}$:

$$\frac{Y}{r^3}[\rho] = \int \frac{Y}{r^3} \left[\rho(x) - \left(\rho(0) + \sum_i \frac{\partial \rho(0)}{\partial x_i} x_i \right) \theta(1 - |x|) \right] dx. \tag{5.52}$$

Since ρ is assumed smooth, it has a Taylor expansion. We now prove a remarkable property of the $\frac{Y}{r^3}$ generalised function.

Theorem 5.10. *The value of* $\int_{a\leq r\leq b}\frac{Y}{r^3}\rho d^3x$ *does not depend on the first term of the Taylor expansion of* ρ*. Furthermore, if* Y *can be written as the sum of spherical harmonics with* $l\geq 2$*, then* $\frac{Y}{r^3}[\rho]$ *does not depend on the second term.*

Proof. Firstly, note that, since integration is a linear operation, each term of the Taylor expansion of ρ may be analysed separately.

That the first term of a Taylor series is irrelevant is obvious; the first term is simply a constant, so it drops out of the Lebesgue integral and the limit and we may proceed as we did in section 5.3.2.

The most general choice for a function with only the second terms of its Taylor expansion being nonzero is $\rho(\boldsymbol{r})=(\alpha x+\beta y+\gamma z)$. Thus the integration of this term is:

$$\int_{a\leq r\leq b}\int_0^{2\pi}\int_0^{\pi}\frac{Y(\theta,\phi)}{r^3}(\alpha x+\beta y+\gamma z)r^2\sin\theta d\theta d\phi dr. \tag{5.53}$$

Using standard spherical harmonic identities:

$$x=r\sqrt{\frac{2\pi}{3}}(Y_{1,-1}-Y_{1,1})\quad y=r\sqrt{\frac{2\pi}{3}}(Y_{1,-1}+Y_{1,1})\quad z=r\sqrt{\frac{4\pi}{3}}(Y_{1,0}). \tag{5.54}$$

we can rewrite expression (5.53) in terms of the spherical harmonics:

$$\sqrt{\frac{2\pi}{3}}\int_a^b dr\int_0^{\pi}\int_0^{2\pi}\frac{Y(\theta,\phi)}{r^3}r[(\beta+\alpha)Y_{1,-1}+(\beta-\alpha)Y_{1,1}+2\gamma Y_{1,0}]\cdot r^2\sin\theta d\theta d\phi. \tag{5.55}$$

The integral over r factors out (as we assume that $b>a>0$, the integrands are well-behaved), leaving a pure spherical integration over a pair of spherical harmonics. As spherical harmonics are orthogonal under spherical integration:[3]

$$\int_{S^2}Y_{l,m}Y_{l',m'}d\Omega=\delta_{ll'}\delta_{mm'}, \tag{5.56}$$

expression (5.55) is zero if Y lacks $l=1$ components. □

The above theorem has an immediate corollary.

Corollary 5.11.

$$\int_{a\leq r\leq b}\frac{Y}{r^3}(\alpha x+\beta y+\gamma z+\eta)d^3x=0, \tag{5.57}$$

where α*,* β*,* γ *and* η *are arbitrary constants.*

[3]The $Y_{l,m}=\sqrt{\frac{(2l+1)(l-m)!}{4\pi(l+m)!}}P_{l,m}(\cos\theta)\exp(im\phi)$ normalisation has been used here.

Because of theorem 5.10 and the linearity of the integral operation, we may add any function, ϕ, to ρ in our integral that has, at most, a first derivative and expect that for all $b \geq a > 0$:

$$\int_{a \leq |x| \leq b} \frac{Y(x)}{r^3} \rho(x) d^3x = \int_{a \leq |x| \leq b} \frac{Y}{r^3} (\rho(x) + \phi(x)) d^3x. \tag{5.58}$$

The left hand side of equation (5.58) is not meaningful in the Lebesgue sense when a is set to 0, but a careful choice of ϕ allows the right hand side to exist. In practical terms, attempting to evaluate the left hand side of equation (5.58) numerically becomes more and more difficult as $a \to 0$, as it requires subtracting a increasingly huge positive contribution from a, likewise, increasingly huge negative contribution. However, by choosing:

$$\phi(\boldsymbol{x}) = -\rho(\mathbf{0}) - (\nabla_{\mathbf{0}} \rho) \cdot \boldsymbol{x}, \tag{5.59}$$

the integrand grows no faster than $\frac{1}{r^2}$, as $r \to 0$, so by theorem 5.4, the singularity is integrable. Consequently, the positive and negative contributions remain finite even when $a = 0$, so numerical integration will terminate.

Unfortunately, convergence as $b \to \infty$ has been spoilt by the choice of ϕ in (5.59). This problem would be fixed if ϕ were to be zero beyond a certain finite radius:

$$\int_{|x| < a} \frac{Y}{r^3} (\rho(x) + \phi(x)) d^3x + \int_{|x| \geq a} \frac{Y}{r^3} \rho(x) d^3x, \tag{5.60}$$

with $a > 0$.

Theorem 5.12. *The value of expression (5.60) is independent of a.*

Proof. Let $I(t)$ be the result of substituting a for t into expression (5.60) and let a_1 and a_2 be two positive real numbers $a_1 < a_2$. $I(a_1) - I(a_2)$ is:

$$\int_{a_1 \leq |x| < a_2} \frac{Y}{r^3} (\rho(x) + \phi(x)) d^3x - \int_{a_1 \leq |x| < a_2} \frac{Y}{r^3} \rho(x) d^3x, \tag{5.61}$$

but, by corollary 5.11, the $\int_{a_1 \leq |x| < a_2} \frac{Y}{r^3} \phi(x) d^3x$ part of the first integral is zero and so the entire expression is zero. □

This solution does leave a discontinuity in the integrand which would prove a liability when evaluating via numerical quadrature, as such procedures usually work by approximating the integrand as an n^{th} order polynomial and, thus, work best when the integrand is C^n smooth. Consequently, we will prove a stronger result.

Theorem 5.13. *The value of the following integral is independent of f:*

$$\int_{\mathbb{R}^3} \frac{Y}{r^3}(\rho(x) + f(|x|)\phi(x))d^3x, \tag{5.62}$$

where f is any bounded function such that $f(x) = 1$ when $x \leq a$ and $f(x) = 0$ when $x \geq b$ for some a, b, such that $0 < a < b$.

Proof. The integral may be split into three parts:

$$\int_{|x|\leq a} \frac{Y}{r^3}(\rho(x) + \phi(x))d^3x + \int_{a<|x|<b} \frac{Y}{r^3}(\rho(x) + f(|x|)\phi(x))d^3x + \int_{|x|\geq b} \frac{Y}{r^3}\rho(x)d^3x. \tag{5.63}$$

The first and last terms are self-evidently independent of f so only the middle part needs consideration:

$$\int_{a<|x|<b} \frac{Y}{r^3}(\rho(x) + f(|x|)\phi(x))d^3x = \int_{a<|x|<b} \frac{Y}{r^3}\rho(x)d^3x + \int_{a<|x|<b} \frac{Y}{r^3}f(|x|)\phi(x)d^3x. \tag{5.64}$$

The first term is also self-evidently independent of f. The second part, written out in terms of spherical coordinates, is:

$$\sqrt{\frac{2\pi}{3}} \int_a^b f(r)dr \underbrace{\int_0^\pi \int_0^{2\pi} \frac{Y(\theta,\phi)}{r^3} r[\alpha Y_{1,-1} + \beta Y_{1,1} + \gamma Y_{1,0} + \rho(0)] \cdot r^2 \sin\theta d\theta d\phi}_{=0}, \tag{5.65}$$

with $\alpha = \frac{\partial\rho}{\partial x}\big|_{x=0} + \frac{\partial\rho}{\partial y}\big|_{y=0}$, $\beta = \frac{\partial\rho}{\partial x}\big|_{x=0} - \frac{\partial\rho}{\partial y}\big|_{y=0}$ and $\gamma = 2\frac{\partial\rho}{\partial z}\big|_{z=0}$. f factors out of the expression and the angular part is equal to zero. Expression (5.62) has been reduced to expression (5.60) and we can proceed as with theorem 5.12. □

The choice of a good f used in the numerical code will be discussed in section 5.4.5.

5.4 Review of Numerical Integration

The general setup of a numerical integration problem assumes that a real valued function, f, and some subset of f's domain, S, are given and that an approximation of the integral f over S is required to within some desired precision. The function must be treated as a "black box" by the integration algorithm; it can only be evaluated on any point within its domain—no derivatives or analytical forms can be assumed known.

The desired precision of the integral, namely the difference between the numerical integration functional, Q, and the exact integration functional, I, is usually specified in terms of two constants ϵ_{abs} and ϵ_{rel}, as:

$$|I[f] - Q[f]| < \epsilon_{\text{abs}} + \epsilon_{\text{rel}} I[f], \tag{5.66}$$

where ϵ_{abs} is called the absolute error and ϵ_{rel} is called the relative error.

All numerical integration algorithms approximate I as:

$$I[f] \approx \sum_i \omega_i f(x_i). \tag{5.67}$$

The set $\{\omega_i\}$ are known as the weights. Methods differ only in the way the weights and evaluation points are selected. Nevertheless, there exists a vast number of approaches to numerical integration in the literature, but they can be essentially divided into two main methodologies: the so-called "numerical quadrature" methods and then so-called "Monte-Carlo" methods.

Monte-Carlo methods are characterised by using random variables to select the evaluation points. The standard deviation of the Monte-Carlo estimate of the integral of some function f is given by:

$$\sigma = \sqrt{\frac{\mathrm{Var}(f)}{n}}, \tag{5.68}$$

so long as n is sufficiently large that the central limit theorem applies [134]. Thus, if an estimate of accuracy ϵ is required, the number of function evaluations scales as $n \propto \frac{1}{\epsilon^2}$, giving Monte-Carlo methods the interesting property that the dimensionality of the problem does not affect the scaling of the required function evaluations against the required error. Quadrature methods, which are deterministic, scale exponentially with the number of dimensions involved, so Monte-Carlo methods win out as the dimensionality of the problem increases.

As the integral we are interested in evaluating here is merely three-dimensional, Monte-Carlo methods were judged to be unlikely to be appropriate, and, so, only methods based on numerical quadrature where considered.

5.4.1 Choosing Weights

A common strategy for selecting integration rules is to project the integrand onto some finite basis set of functions where the basis functions can be integrated exactly. The basis set of functions is usually chosen to be polynomials, though other basis sets can be used, depending on the problem (for example, decomposing a function in terms of its Fourier series may be more appropriate for oscillatory integrands).

In this review, only polynomial approximations will be considered, but methods involving other approximations are discussed in Engels' book [135] and are roughly analogous.

Definition 5.14. *The degree of an integration rule is the order of the highest order multivariate polynomial that rule integrates exactly.*

To derive integration rules of a particular degree in one dimension, we will apply the exact integration functional, I, and an n-point quadrature functional, Q_n, to the same smooth function, f, written in terms of its Taylor series and compare the coefficients of the terms involving the derivatives.

The exact integration is:

$$\int_{-1}^{1} f(x)dx = I[f] = 2f(0) + \frac{2}{2! \cdot 3} f''(0) + \cdots + \frac{1-(-1)^{n+1}}{n! \cdot (n+1)} f^{(n)}(0) + \ldots, \tag{5.69}$$

and the application of a generic quadrature rule, $Q[f] = \sum_i \omega_i f(x_i)$, gives:

$$Q[f] = (\sum_i \omega_i) f(0) + (\sum_i \omega_i x_i) f'(0) + \frac{1}{2}(\sum_i \omega_i x_i^2) f''(0) + \cdots + \frac{1}{n!}(\sum_i \omega_i x^n) f^{(n)}(0). \tag{5.70}$$

If the evaluation points are fixed, then comparing the first n terms of this series gives n linear equations with n unknowns that can be solved to get the weights. These weights will exactly integrate functions nonzero derivatives $(n-1)^{\text{th}}$ order only of only, in other words, $(n-1)^{\text{th}}$ order polynomials.

If the evaluation points are also allowed to vary, the number of unknowns increases to $2n$, the first $2n$ terms of sequences (5.69) and (5.70) may be equated to find $2n$ nonlinear equations. These can be solved to find weights and evaluation points that will integrate a $(2n-1)^{\text{th}}$ order polynomial exactly. This procedure is the Gaussian quadrature method and, in general, it builds the rules of the highest degree possible for a given number of evaluation points [135].

5.4.2 Error Terms

In practical calculations, particularly those relying on adaptive algorithms, having an estimate of the error is vital. Unfortunately, there is no way to find a formal upper bound of the error on the integration error without knowing more information about the form of the integrand. To see this, consider a quadrature scheme with integration points at $\{x_i\}$. We can vary any integrand by adding an arbitrary function with roots at $\{x_i\}$ (such as the polynomial $\prod_i (x - x_i)$) without changing the result of applying quadrature.

This apparently gloomy theoretical result does not translate into a prohibition against error estimates that are sufficiently reliable "to get work done", but one must remember that, for every integration scheme based on quadrature, pathological objective functions exist. Lyness gives a particularly lucid discussion of the failure modes inherent in automatic quadrature routines in [104], that include cases more plausible than the clearly contrived example given above. Lyness also makes the point that, if a quadrature is suspected to have

failed on a pathological case, increasing the tolerance is unlikely to help. Instead, a better response would be to select different subdivision of the target interval.

For completeness, it should be noted that there is a case where a formal upper bound on the numerical error may be obtained, which is if a bound on the derivative of the target function is known [104, 135]. However, in practice, the most common scheme approach to error analysis used is to check that the results of using a higher degree integration rule match the results of the target rule to within a certain tolerance [104, 135].

5.4.3 Locally Adaptive vs Globally Adaptive Algorithms

Algorithms that are used in practice are typically *adaptive algorithms*. Adaptive algorithms divide the integration interval into pieces, apply a quadrature rule to each individual piece, and then sum the contributions to the integral from each piece.

Subdivision strategies are classified as either locally adaptive or globally adaptive. Local algorithms subdivide each interval, $[a_i, b_i]$, until:

$$E(a_i, b_i) \leq \frac{b_i - a_i}{b - a}\epsilon, \tag{5.71}$$

is satisfied, where $E(a_i, b_i)$ indicates an error estimate of some quadrature procedure on the interval $[a_i, b_i]$ and ϵ is the requested precision. If criterion (5.71) is not met, $[a_i, b_i]$ will be split into left and right parts and the analysis repeated for the left part until the criterion is met. Intuitively, local adaptive algorithms can be thought of as working from left to right over the interval, only moving on if the error to left of b_i has been made sufficiently small. In contrast, a globally adaptive algorithm maintains a list of all subdivisions and selects the interval from the entire set that is currently contributing the most to the total error. Globally adaptive algorithms are usually preferable and tend to require fewer integration steps to guarantee a sufficiently small error (see the paper by *Malcolm et. al.* [136] for a comparison).

5.4.4 The Cuhre Algorithm

The algorithm chosen for use in this work was the Cuhre algorithm from the multidimensional numerical integration library Cuba. Cuba is collection of multidimensional integration algorithms written in C, described two papers by *Hahn* [137, 138]. The Cuhre algorithm itself was first published in by *Berntsen et al.* [139], based on an earlier adaptive algorithm by *Dooren and Riddler* [140].

An extremely important advantage offered by the the Cuhre algorithm over other similar algorithms is that it is able to handle finite differencing well (uses of finite differencing in `pcsfit` are discussed in section 5.5.5). Cuhre is capable of evaluating a vector of integrands

at once using the same subdivision scheme for each integral in the integral set, so the finite differencing error is independent of the numerical integration error. To see this, note that the central finite difference of two integrals approximated via quadrature is:

$$\frac{\sum_i \alpha_i f(x+h)(x_i) - \sum_j \beta_j f(x-h)(y_j)}{2h}, \tag{5.72}$$

where $\{\alpha_i\}$ and $\{\beta_j\}$ are two distinct weight sets, with corresponding evaluation point sets $\{x_i\}$ and $\{y_j\}$. $f : \mathbb{R}^n \to (\mathbb{R}^3 \to \mathbb{R})$ is the "model building function." It takes a set of model parameters, such as the tensor and the position of the metal and returns a function of domain $\mathbb{R}^3$ and range $\mathbb{R}$ that describes the PCS over all space. This terminology, while a little non-standard, maps well onto the implementation in `pcsfit` (section 5.6).

There is no reason to believe that the numerical errors in $\sum_i \alpha_i f(x+h)(x_i)$ and $\sum \beta_j f(x-h)(y_j)$ are correlated, so we would need to demand that the precision to which the integral was evaluated was less than $I[f(x+h)] - I[f(x-h)]$.

However, if $(\{\alpha_i\}, \{x_i\})$ and $(\{\beta_j\}, \{y_i\})$ are set equal, then the summation can be moved outside of the finite differencing operation:

$$\frac{\sum \alpha_i f(x+h)(x_i) - \sum_i \alpha_i f(x-h)(x_i)}{2h} = \sum_i \alpha_i \frac{[f(x+h) - f(x-h)]}{2h}(x_i), \tag{5.73}$$

and so h may be shrunk to obtain the required finite differencing precision without worrying about the numerical integration precision. In practice, this feature of Cuhre greatly reduced the required precision goal, and, correspondingly, the runtime of the fitting program.

5.4.5 The Integral Smoothing Function

In section 5.3.4 we established that the integral to be evaluated is:

$$\int_{\mathbb{R}^3} \frac{Y}{r^3}(\rho(x) + f(|x|)\phi(x))d^3x, \tag{5.74}$$

with a free choice of f under certain constraints given in that section. In this section, we ask what the best f to pick would be to facilitate efficient numerical integration.

The most obvious choice of f is a sharp step function that cuts off the effects of ϕ beyond some radius. Unfortunately, doing so introduces a discontinuity in the integrand and, as the Cuhre algorithm approximates the integrand as high order polynomials that poorly approximate a discontinuous cut-off, convergence is slow. Testing revealed that using a sharp cut-off function resulted in an excessive number of function evaluations along the boundary of the discontinuity.

Ideally, we need to choose an f such that the integrand is C^n smooth where n is a suitably high integer, to be found by trial and error, such that additional function evaluations at the

cut-off radius are sufficiently reduced. This requirement, in turn, imposes the condition that f should be C^n smooth.

We begin by generating a polynomial, $P_n(x)$, that is 1 when $x = 0$ and 0 when $x = 1$ and has vanishing m^{th} derivatives at 0 and 1 for all $m = [1 \dots n^{\text{th}}]$. $P_n(x)$ is generated as follows:

$$P_n(x) = 1 - \frac{Q_n(x)}{Q_n(1)} \quad \text{where } Q_n(x) = \int_0^x y^n (1-y)^n dy, \tag{5.75}$$

f can now be written as a rescaling of P_n

$$f(x) = \begin{cases} 1 & \text{when } x < a \\ P_n((x-a)/(b-a)) & \text{when } a \leq x < b \\ 0 & \text{when } b \leq x, \end{cases} \tag{5.76}$$

where a and b are two real numbers. In the `pcsfit` program, a was set to the width parameter of the Gaussian function (defined later in section 5.5.3), s, while b was set to $2s$. `pcsfit` uses an 11^{th} order polynomial, which proved more than sufficient.

5.5 Analytical Derivatives

To perform a fitting procedure with a gradient-based method such as BFGS (reviewed in appendix F.1), a method of calculating the gradients is required. Although obtaining gradients via finite differencing is always possible, accurate analytical gradients are preferable. In this section, full analytical gradients of the delocalised PCS model are found. We will take a "top down" approach, beginning with the full error functional and working inwards.

5.5.1 Derivatives of Error Functionals

When performing fitting, the gradient of the error functional, not the model, is required. The error functional in terms of some model σ and the experimental value σ_{exp} is:

$$\mathcal{E} = \sum_i (\sigma(x_i) - \sigma_{\text{exp}}(x_i))^2, \tag{5.77}$$

where the set $\{x_i\}$ runs over the locations of the positions of the nuclei showing resonances. From now on, the (x_i) arguments will be suppressed. The derivative of the error functional with respect to some parameter α is:

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial \alpha} &= \frac{\partial}{\partial \alpha}\sum_i (\sigma - \sigma_{\text{exp}})^2 \\ &= \frac{\partial}{\partial \alpha}\sum_i (\sigma^2 + \sigma_{\text{exp}}^2 - 2\sigma\sigma_{\text{exp}}) \\ &= 2\sum_i \frac{\partial \sigma}{\partial \alpha}(\sigma - \sigma_{\text{exp}}).\end{aligned} \tag{5.78}$$

5.5.2 Derivatives of a Convolution

Let f and g be two smooth functions that depend on the spatial parameters, $\boldsymbol{x}$, and also additional vectors of parameters that are not dependent on space. f depends on the vector $\boldsymbol{\alpha}$ and g on the vector $\boldsymbol{\beta}$. Their convolution over the spatial coordinates is given by the expression:

$$(f \star g)(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \int_{\mathbb{R}^n} f(\boldsymbol{x}', \boldsymbol{\alpha}) g(\boldsymbol{x} - \boldsymbol{x}', \boldsymbol{\beta}) d^n x'. \tag{5.79}$$

We are interested in the derivatives of the convolution with respect to the parameters $\boldsymbol{x}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Because f and g are well behaved, the derivative may be moved inside the integral:

$$\begin{aligned}\frac{\partial}{\partial x_i}(f \star g)(\boldsymbol{x}) &= \int_{\mathbb{R}^n} d^n \boldsymbol{x}' \frac{\partial}{\partial x_i}[f(\boldsymbol{x}') g(\boldsymbol{x} - \boldsymbol{x}')] \\ &= \int_{\mathbb{R}^n} d^n \boldsymbol{x}' f(\boldsymbol{x}') \frac{\partial}{\partial x_i} g(\boldsymbol{x} - \boldsymbol{x}') \\ &= \int_{\mathbb{R}^n} d^n \boldsymbol{x}' f(\boldsymbol{x}') \frac{\partial g}{\partial x_i}(\boldsymbol{x} - \boldsymbol{x}') = (f \star \frac{\partial g}{\partial x_i})(\boldsymbol{x}).\end{aligned} \tag{5.80}$$

Through the symmetry of the convolution operation, $(f \star \frac{\partial g}{\partial x_i})(\boldsymbol{x}) = (\frac{\partial f}{\partial x_i} \star g)(\boldsymbol{x})$. The derivatives with respect to the non-spatial parameters are:

$$\begin{aligned}\frac{\partial}{\partial \alpha_i}(f \star g)(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &= \int_{\mathbb{R}^n} d^n \boldsymbol{x}' \frac{\partial}{\partial \alpha_i}[f(\boldsymbol{x}', \boldsymbol{\alpha}) g(\boldsymbol{x} - \boldsymbol{x}', \boldsymbol{\beta})] \\ &= \int_{\mathbb{R}^n} d^n \boldsymbol{x}' [\frac{\partial}{\partial \alpha_i} f(\boldsymbol{x}', \boldsymbol{\alpha})] g(\boldsymbol{x} - \boldsymbol{x}', \boldsymbol{\beta}) \\ &= \int_{\mathbb{R}^n} d^n \boldsymbol{x}' \frac{\partial f}{\partial \alpha_i}(\boldsymbol{x}', \boldsymbol{\alpha}) g(\boldsymbol{x} - \boldsymbol{x}', \boldsymbol{\beta}) = (\frac{\partial f}{\partial \alpha_i} \star g)(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}).\end{aligned} \tag{5.81}$$

5.5.3 Derivatives of the Gaussian

The normalised Gaussian function used to represent the electron density is:

$$\rho(\boldsymbol{x}) = (s^2\pi)^{-\frac{3}{2}} \exp(-\frac{x^2 + y^2 + z^2}{s^2}). \tag{5.82}$$

Derivative	Expression
$\frac{\partial\sigma_{\text{point}}}{\partial\chi_{z^2}}$	$\frac{2z^2-y^2-z^2}{12\pi r^5}$
$\frac{\partial\sigma_{\text{point}}}{\partial\chi_{x^2-y^2}}$	$\frac{x^2-y^2}{12\pi r^5}$
$\frac{\partial\sigma_{\text{point}}}{\partial\chi_{xy}}$	$\frac{xy}{2\pi r^5}$
$\frac{\partial\sigma_{\text{point}}}{\partial\chi_{xz}}$	$\frac{xz}{2\pi r^5}$
$\frac{\partial\sigma_{\text{point}}}{\partial\chi_{yz}}$	$\frac{yz}{2\pi r^5}$
$\frac{\partial\sigma_{\text{point}}}{\partial x}$	$-\frac{5xA}{12\pi r^7}+\frac{-2x\chi_1+2x\chi_2+6y\chi_{xy}+6z\chi_{xz}}{12\pi r^5}$
$\frac{\partial\sigma_{\text{point}}}{\partial y}$	$-\frac{5yA}{12\pi r^7}+\frac{-2y\chi_1-2y\chi_2+6x\chi_{xy}+6z\chi_{yz}}{12\pi r^5}$
$\frac{\partial\sigma_{\text{point}}}{\partial z}$	$-\frac{5zA}{12\pi r^7}+\frac{4x\chi_1+6x\chi_{xz}+6y\chi_{yz}}{12\pi r^5}$

Table 5.1 – Derivatives of equation (5.28) with respect to the model parameters. The symbol A is defined as $(2z^2-x^2-y^2)\chi_{z^2}+(x^2-y^2)\chi_{x^2-y^2}+6xy\chi_{xy}+6xz\chi_{xz}+6yz\chi_{yz}$ to save space.

The derivative with respect to s, which shall be called the "width parameter", is required for feeding the fitting routine, while the derivatives with respect to x, y, and z are required for integral regularisation. The derivative with respect to s is:

$$\frac{\partial\rho}{\partial s}(\boldsymbol{x})=\pi^{-\frac{3}{2}}\left[2s^{-6}(x^2+y^2+z^2)-3s^{-4}\right]\exp(-\frac{x^2+y^2+z^2}{s^2}), \tag{5.83}$$

and with respect to x is:

$$\frac{\partial\rho}{\partial x}(\boldsymbol{x})=-2\pi^{-\frac{3}{2}}\frac{x}{\sqrt{s}}\exp(-\frac{x^2+y^2+z^2}{s^2}). \tag{5.84}$$

x, y, and z appear symmetrically in equation (5.82), so derivatives with respect to y and z can be found analogously.

5.5.4 Derivatives of the Point Model

In section 5.2 is was shown that:

$$\sigma_{\text{point}}=\frac{1}{12\pi r^5}[(2z^2-y^2-x^2)\chi_{z^2}+(x^2-y^2)\chi_{x^2-y^2}+6xy\chi_{xy}+6xz\chi_{xz}+6yz\chi_{yz}]. \tag{5.85}$$

The derivatives with respect to the model parameters are given in table 5.1:

5.5.5 Finite Differencing

Calculation of the delocalised model with respect the Gaussian width parameter is currently done via eight-point central finite differencing, rather than with an analytic formula. Although such a formula exists, it was found that using finite differencing was more numerically stable than using the analytical gradient in expression (5.83) integrated over space.

Gradients of the spatial parameters are also computed using six-point central finite differencing. Unlike the case of the gradient with respect to the Gaussian width given above,

the analytic forms of these gradients are difficult to compute. To see this, notice that the point model gradients with respect to x, y, and z in table 5.1 contain a total power of r^{-4} and can no longer be written in the form $\frac{Y}{r^n}$. Consequently, when put in a convolution integral, the analysis performed in section 5.3.4 breaks down and the numerical integration routine fails to converge.

It is important to stress that none of the above in any way implies that an expression for the analytical derivative of the extended model with respect to the spacial parameters in some way fails to exist, simply that to arrive at such an expression would require the application of the full theory of generalised functions. Firstly, a regularisation, R, of the non-existent integral:

$$\int_{\mathbb{R}^3} \rho(\boldsymbol{x}') \frac{\partial}{\partial x_i} \sigma_{\text{point}}(\boldsymbol{x} - \boldsymbol{x}') d\boldsymbol{x}', \tag{5.86}$$

would need to be constructed using equation (5.51) and, then, a generalised function concentrated at the origin, g, would need to be chosen, using the condition that the derivatives matched:

$$(R + g)[\rho] = \frac{\partial}{\partial x_i} \int_{\mathbb{R}^3} \frac{Y}{r^3} (\rho(\boldsymbol{x}) + f(|\boldsymbol{x}|)\phi(\boldsymbol{x})) d^3\boldsymbol{x}. \tag{5.87}$$

It was decided that using a finite differencing provided an approach that would lead to a working implementation more quickly without incurring an unacceptable computational overhead.

5.5.6 Rescaling

Quasi-Newton methods, such as BFGS work best when the problem is well scaled. Generally, the input parameters should all be of the same order of magnitude, but because the input parameters of the point model have different physical units, the problem is not generally well scaled.

Instead of minimising the error functional, a rescaled error functional is defined by assuming that the starting parameters of the optimisation have orders of magnitude that are "typical". Let $\boldsymbol{x}_0$ be the vector of starting parameters and f be the error functional, then the rescaled functional is defined as $\tilde{f}(\boldsymbol{y}) = f(\boldsymbol{y} \odot \boldsymbol{x}_0)$ where "$\odot$" is the element wise, or Hadamard, product of vectors. With this rescaling, all the components of the new starting parameter vector, $\boldsymbol{y}$, are one. Nothing needs to be done to the value of f, but the gradient must be corrected:

$$\frac{\partial \tilde{f}}{\partial y_i} = (\boldsymbol{x}_0)_i \frac{\partial f}{\partial x_i}. \tag{5.88}$$

Using this type of rescaling requires that the starting parameters be all nonzero otherwise, a division by zero will occur during rescaling.

5.6 Implementation Details

The ideas that have been presented in this chapter have been used to write a simple command line fitting program called `pcsfit` in C++, capable of fitting PCS data to both the point and extended models quickly. The BFGS implementation in the GNU scientific library [141] was used for fitting. Multithreading has been used to speed calculations on computers with multiple cores. `pcsfit` uses a small number of Linux specific system calls, so it is currently not portable across operating systems. However, because the majority of the code is portable, porting `pcsfit` to other operating systems would not be a significant undertaking, should the need arise.

Documentation of how to use `pcsfit` can be found in appendix G.

5.6.1 Input File Formats

`pcsfit` requires two types of input file: the model file and the data set file. The model file specifies the type of model to fit against and the starting parameters to use, while the data set file contains the x, y, and z coordinates of each relevant nucleus, along with a PCS value.

The first line of a model file specifies the type of model to use when fitting. Currently, the only valid values are "`point`" and "`gauss`." Each line after the model specification contains a single floating point number recorded in C convention that corresponds to the starting value of a model parameter. For the point model, eight parameters are expected in the following order: x, y, z, χ_{z^2}, $\chi_{x^2-y^2}$, χ_{xy}, χ_{xz}, and χ_{yz}. The parameter order for "gauss" model is identical, except an additional parameter, the Gaussian width, is expected after the first eight parameters.

The data set file is simply a collection of lines, each containing four C-style floating point numbers. The first of these corresponds to the PCS, while the subsequent three are associated x, y, and z coordinates of the spin.

When writing model and data-set files, if all positional variables are specified in Ångströms and shifts are given in *p.p.m.*, then inputted susceptibility tensor parameters should be given in $m^3 \times 10^{-32}$. If other units are used, then the units of the susceptibility tensor should be adjusted accordingly.

5.6.2 `pcsfit` Self Tests

An unfortunate reality of heavily numerical code is that there is a large scope for subtle bugs. Unlike programs in other domains, numerical bugs may silently affect the output and often lack highly visible symptoms such as program crashes or visual artefacts. Inevitably, some method of verification of numerical code is needed for the results to be credible. Two

verification strategies were used when developing `pcsfit`: 1) a series of self tests based on random data were written and; 2) the results from `pcsfit` were compared against those of the existing PCS fitting program, Numbat [142]. This section will describe the self tests, while comparison with Numbat will appear in section 5.7.

The self test functions are all found in `tests.cpp` and can be invoked by running `pcsfit` with the `selftest` command. Four main test functions were written, each will now be described in turn.

`check_derivative` Computes the analytical and numerical derivatives of a random model at a random point on the unit cube. The results are printed to the standard output. Generally, close agreement between the analytical and numerical gradients is expected.

`check_error_derivative` Computes the analytical and numerical derivatives of a random point model and a random data set that is generated from a different set of parameters (if the same parameters were used for the data set and the point model, the error derivative would always be zero). Once again, a close agreement between the analytical and numerical gradients is expected. Obviously, if `check_derivative` fails, `check_error_derivative` must also fail.

`check_minimum` As mentioned above, the gradient of the error functional, evaluated for a set of parameters that were also used to generate a data set, must be zero. `check_minimum` does this, printing out the calculated gradient which is expected to be zero in all parameters. Failure may indicate a noisy minimum.

`check_convergence` Is a complete test of the fitting system. A random data set is generated from a random model, the random model parameters are then perturbed by a random change and then used as the starting parameters of a fitting procedure. The original parameters should then be recovered.

No explicit pass/fail criteria for these tests has been built into `pcsfit`; it is up to the operator to judge if the degree of agreement between two results that are expected to match is sufficient. Nevertheless, these self tests did prove helpful for quickly detecting bugs during development.

5.6.3 The Overall Layout of `pcsfit`

`main.cpp` Responsible for parsing the command line and setting up global state.

`maths.cpp` Contains a number of small mathematical utilities. Of these, the two most important are a lightweight three-dimensional vector template, `Vec3<T>`, and the `magic_polynomial` function that is used in interpolation (see section 5.4.5).

data.cpp Contains classes used for in-memory representation of data sets and associated functions for loading them from disk.

model.cpp Contains the PCS models and is responsible for parsing the parameter files. This module also contains a number of other utility functions.

fit.cpp Responsible for coordinating the code from the GNU scientific library, the `thread.hpp` header, and the scientific code in `models` in order to perform the actual fitting.

tests.cpp Contains a sequence of tests that are run when `pcsfit` is run using the `selftest` command.

5.6.4 Models

The file `model.cpp` and its header `model.hpp` contain the implementations of both the point and extended models and facilities for computing their gradients via analytic and numerical methods.

`model.hpp` begins with type definitions for the models:

```
1  typedef void (*ModelF) (Vec3d evalAt, const double* model, double*
       value,double* gradient,const ModelOptions* modelOptions);
```

Values of type `ModelF` are function pointers to a function that computes a model. The model should be evaluated at `evalAt`, `params` is a pointer to an array of model parameters, such at the location of the metal and the susceptibility tensor, `value` is set to the value of the PCS as computed by the model, and `gradient` is an array of doubles that will be set to the value of the model gradient. If the gradient is not required, `gradient` should be set to `NULL`.

Model functions of type `ModelF` are packaged along with additional metadata in the `Model` structure:

```
1  struct Model {
2    ModelF        modelf;
3    unsigned long size;
4    const char*   name;
5  };
```

5.6.5 Multithreading

Evaluation of the error functional requires the evaluation of the PCS for each spin in the protein, but each PCS is independent, so the problem of evaluation of the error functional

```
1 #include <boost/thread.hpp>
2 #include <boost/thread/barrier.hpp>
3 #include <boost/function.hpp>
4 #include <vector>
5
6 template<typename T>
7 class Multithreader {
8 public:
9     Multithreader();
10   ~Multithreader();
11
12
13     void map(const std::vector<boost::function<T ()> >& funcs,std::
          vector<T>& mapTo);
14 };
```

Listing 5.1 – The public interface for the Multithreader template. All concurrency is accomplished through this interface.

(with the exception of the trivial reduction step of summing the errors) is embarrassingly parallel.

A simple template was written, which may be found in `threads.hpp`, to abstract the details of setting up parallel calculations and thread management. The public interface of this template is shown in listing 5.1.

The public interface of `Multithreader<T>` consists of three functions: a constructor, a destructor, and `map`. The constructor queries the operating system about the number of CPU cores available and creates one thread for each core. The destructor blocks while those threads terminate.

A set of jobs to be executed can be passed to the template by the `map` function. The input variable, `funcs`, is a variable holding a `std::vector` of nullary functions returning a `T` and the output variable is a `std::vector` of `T`. `map` performs the equivalent of:

```
1   mapTo.resize(funcs.size());
2
3   for(std::vector<boost::function<T()> >::iterator i = funcs.begin();
4       i != multithreader.end();++i) {
5         mapTo[i] = funcs[i]();
6   }
```

but the loop is run in parallel.

As the functions stored in `funcs` are nullary, there does not seem to be an easy way to supply work to be done except via global variables. However, because `map` accepts a `std::vector` of `boost::function<T()>`, rather than a plain C-style function pointer, it is possible to specify parallel jobs as partially applied n-ary functions. Typically, partial

application will be performed via `boost::bind`, which is designed to work well with `boost::function`. In this way, a job may be specified without constraining the number or type of the job's arguments.

A typical use of the `Multithreader<T>` template is given below:

```
1  //A type for containing a PCS and a gradient, defined at global scope.
2  typedef pair<double,vector<double> > fdf_t;
3
4  //A function for computing a single PCS and gradient.
5  fdf_t worker(const ErrorContext* context,const double* params,unsigned
       long jobN);
6
7  //In eval_error()...
8  std::vector<boost::function<fdf_t()> > work;
9  std::vector<fdf_t()> results;
10
11 for(unsigned long i = 0;i<nuclei.size();i++) {
12   work.push_back(boost::bind(&worker,context,params,i));
13 }
14
15 //Map the work to the result.
16 pool->map(work,results);
```

The function `worker` is the function that computes a PCS and its gradient, which should be executed in parallel. Partial application is performed on line 12 and the work is dispatched on line 16.

5.6.6 Notes on Integral Evaluation Precision

At first it would seem that increasing the target precision for the Cuhre algorithm would automatically lead to better final results (on the understanding that doing so may increase run time), however, there are possible reasons why increasing the precision past a certain threshold may actually degrade the results. While these issues are fixable in principle, the additional precision was not needed to optimise the model parameters to a precision that exceeded experimental uncertainties, so correcting them was not needed.

The first issue arises when evaluating:

$$\rho(\boldsymbol{x}-\boldsymbol{x}') - \rho(\boldsymbol{x}) - \left.\frac{\partial\rho}{\partial x}\right|_{\boldsymbol{x}} x - \left.\frac{\partial\rho}{\partial y}\right|_{\boldsymbol{x}} y - \left.\frac{\partial\rho}{\partial z}\right|_{\boldsymbol{x}} z. \tag{5.89}$$

When $\boldsymbol{x}'$ is close to $\mathbf{0}$, evaluating this expression involves subtracting floating point numbers that are close together ($\rho(\boldsymbol{x}-\boldsymbol{x}')$ and $\rho(\boldsymbol{x})$), possibly resulting in loss of numerical precision. When the integral is evaluated at higher precision, the likelihood of $\boldsymbol{x}'$ being close to 0 increases.

The second issue arises because of the way gradients with respect to the position variables are evaluated using finite differencing. When evaluating the integrand in regions where $|\boldsymbol{x}'|$

is much greater than the finite difference step, h, there is no problem. However, in regions where $|\boldsymbol{x}'|$ is comparable to h, the quadratic and higher terms of the integrand begin to dominate, and the calculated gradient becomes unreliable.

Through testing it was found that setting the relative tolerance of 10^{-4} was sufficient to perform fitting without the issues described above from becoming a problem. As the tolerance is increased, the fitting process runs for more steps and comes closer to the exact minimum before terminating.

5.7 Point Model Verification Against Numbat

Numbat is a fully featured PCS fitting program, capable of determining the susceptibility tensors for proteins from measured PCS lineshifts and assignments that has been published by *Schmitz et. al.* [143]. The program is unique among PCS fitting programs in that it comes with a GUI, and user-friendliness has been strongly emphasised throughout the design.

Numbat was chosen to verify the fitting results of `pcsfit`'s point model against, as the meaning of the Numbat's tensor representation convention (the so called "Universal Tensor Representation") is described in detail in the paper and Numbat's emphasis on user friendliness reduced the chance of operator errors.

The PCS shifts for the θ subunit, from the $Dy^{3+}/\epsilon 168/\theta$ complex found in table S1 of the supplementary information of the Numbat paper [142], was chosen as a data against which to verify `pcsfit`. Numbat used the PDB coordinate set 2AXD by *Keniry et al.* [144]. A Haskell script was written to extract the relevant hydrogen coordinates and combine them with the PCS data to produce an input file suitable for `pcsfit`.

The susceptibility tensor produced by running `pcsfit` on this input file, using the point model, was:

$$\underbrace{\begin{pmatrix} 13.9337 & 14.173 & -9.6766 \\ 14.173 & -10.9123 & -11.3271 \\ -9.6766 & -11.3271 & -3.02133 \end{pmatrix}}_{\texttt{pcsfit tensor}} \quad \underbrace{\begin{pmatrix} 14.4178 & 14.1369 & -9.46331 \\ 14.1369 & -10.9123 & -11.3271 \\ -9.46331 & -11.3271 & -3.06795 \end{pmatrix}}_{\text{Numbat Tensor}}, \tag{5.90}$$

where the analogous tensor found by Numbat has also been shown for comparison. The units are $m^3 \times 10^{-32}$. The two tensors are sufficiently close together, and the `pcsfit` tensor certainly falls within the estimated error quoted by Numbat. The differences are likely due to differences in the fitting algorithm (BFGS vs. Powell Minimisation), termination criteria, and the parametrisation used to represent the tensor during fitting.

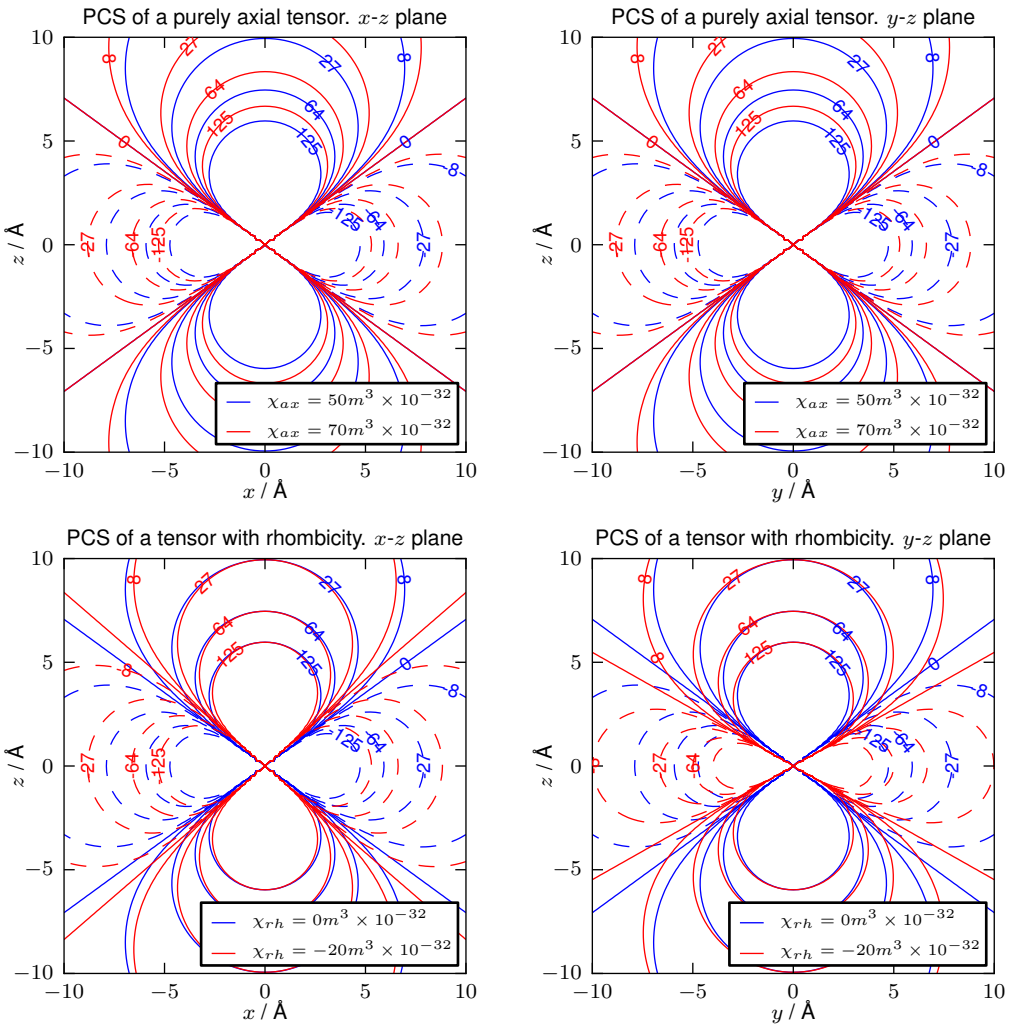


Figure 5.2 – Effects of varying axiality and rhombicity on the point model. Cross sections of the 3D distribution are given in the XZ and YZ planes of the tensor eigenframe. Unless otherwise specified, parameters are $\chi_{\mathrm{ax}} = 75m^3 \times 10^{-32}$, $\chi_{\mathrm{rh}} = 0m^3 \times 10^{-32}$. Units of PCS are in *ppm*.

5.8 Discussion of the Functional Form of Extended Model

The functional form of the extended model is closely related to that of the point model it derives from. There are some details that do not change from the point model: both models predict a nearly identical PCS far from the metal centre and many of the features of the extended model function are inherited from the point model.

Unfortunately, the fitting parameters used in `pcsfit` do not lend themselves to being intuitively interpreted. They were selected avoid having to deal with explicit rotations and the associated complication of possibly divergent derivatives of Euler angles rather than for being directly related to physically significant properties of the tensor. In particular, their

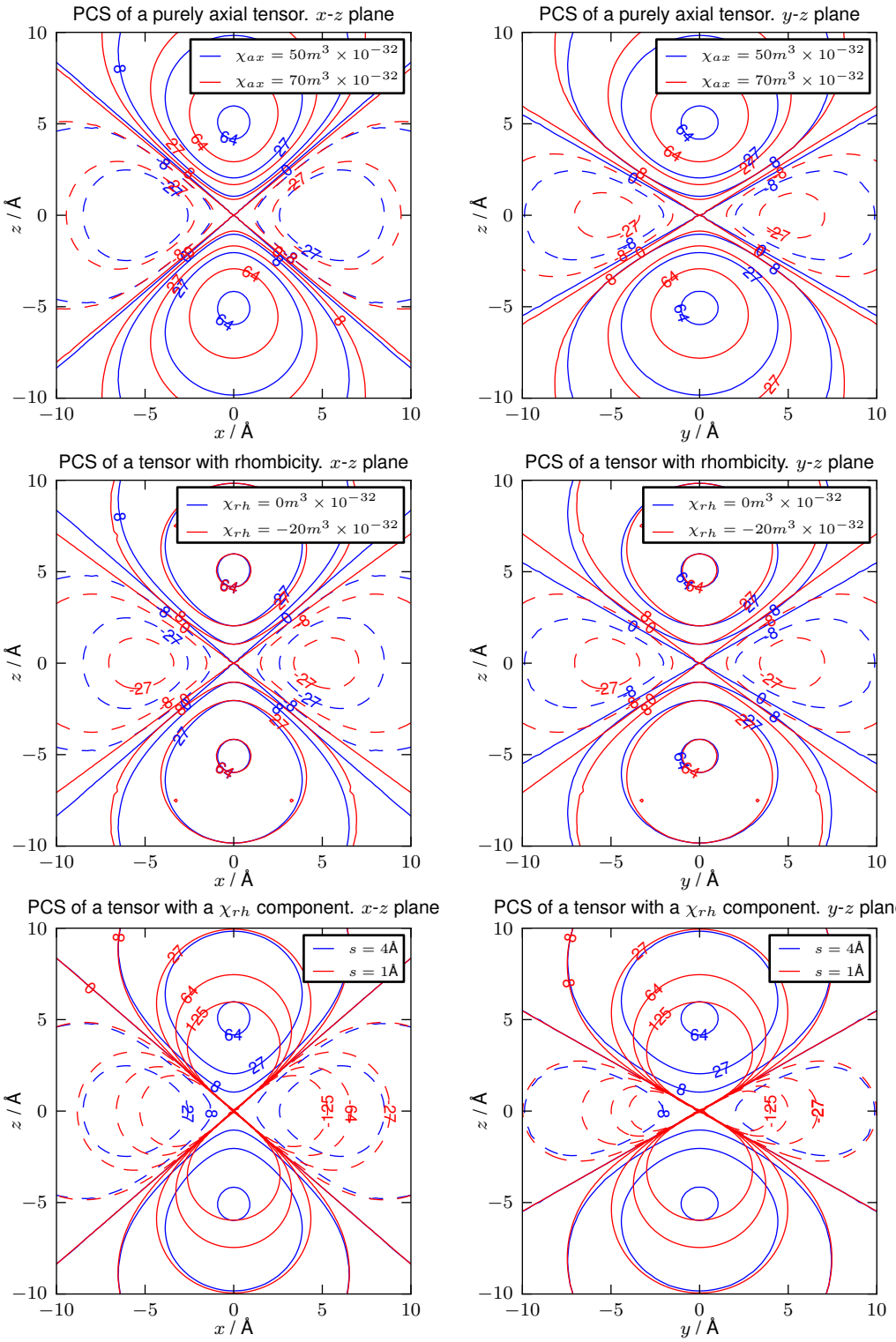


Figure 5.3 – Effects of varying axiality, rhombicity, and the width parameter on the extended model. Cross sections of the 3D distribution are given in the XZ and YZ planes of the tensor eigenframe. Unless otherwise specified, parameters are $\chi_{\mathrm{ax}} = 75m^3 \times 10^{-32}$, $\chi_{\mathrm{rh}} = 6.6m^3 \times 10^{-32}$, $s = 4$Å. Units of PCS are in *ppm*.

values are heavily dependent on the coordinate system. Consequently, they are best thought of as an intermediate representation to be converted to a more appropriate tensor parameter convention before interpretation.

Perhaps the most intuitive parametrisation for gaining insight into the behaviour of both the point and extended models is the axiality and rhombicity parametrisation [45]:

$$\sigma_{\text{point}} = \frac{1}{12\pi r^3}\left[\frac{1}{2}(3\cos^2\theta - 1)\chi_{\text{ax}} + \frac{3}{2}\chi_{\text{rh}}\sin^2\theta\cos 2\phi\right] \tag{5.91}$$

where the interaction has been written in the eigenframe the tensor with the axes ordered as specified in section 3.2.3. In the case where χ_{rh} is zero the function changes sign along a cone 54.7 degrees from the z axis due to the $(3\cos\theta - 1)$ factor. When χ_{rh} is not zero, the function is decreased close to the x axis and increased close to the y due to the $\cos 2\phi$. This behaviour illustrated in figure 5.2.

Moving to the delocalised model results in the smearing out of dipole like function. In the point model, as the sampling point is moved towards the metal position, the absolute value of the PCS increases without limit (the only exception being an approach along a zero cone). The sign depends on the angle of approach meaning the value at the metal was indeterminate. In the extended model this is no longer the case. At large distances, approaching the metal still results in an increase in the absolute value of the PCS but it will eventually reach a maximum before finally dropping back to zero at the centre. This behaviour is illustrated in figure 5.3.

The effects of adjusting the axiality and rhombicity of the susceptibility tensor are analogous to the point model case. In both point and extended cases the three Euler angles (or other rotation parametrisation) serve to orient the function in space.

5.9 Results and Discussion

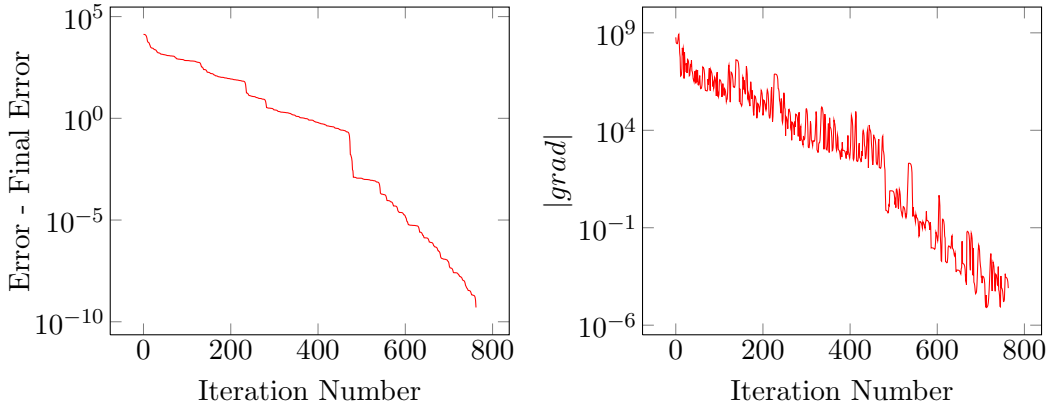
Presented here is a comparison of the obtained susceptibility tensor and other model parameters fit to a published set of chemical shifts and assignments for *E. coli* DNA Polymerase III. The distribution of the unpaired electron is assumed to be Gaussian:

$$\rho(\boldsymbol{x}) = (s^2\pi)^{-\frac{3}{2}}\exp(-\frac{x^2+y^2+z^2}{s^2}). \tag{5.92}$$

with s being the width parameter, with units of length.

5.9.1 Results

E. coli DNA Polymerase III was chosen as a test case, as it was suspected to be a good candidate for observing a delocalised unpaired electron (Gottfried Otting, private communication. See the acknowledgements section at the end of this chapter).

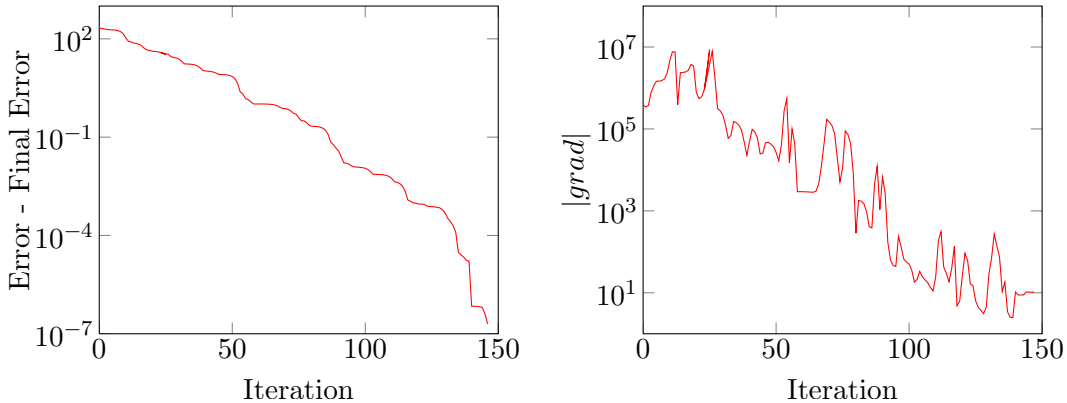


(a) Error Functional **(b)** Gradient Magnitude

Parameter	x	y	z	χ_{z^2}	$\chi_{x^2-y^2}$	χ_{xy}	χ_{xz}	χ_{yz}
Value	-2.02	30.3	-4.62	28.1	31.2	31.2	-7.94	0.84
Parameter	x	y	z	χ_{ax}	χ_{rh}	α	β	γ
Value	-2.02	30.3	-4.62	64.9	-24.8	238	16.0	156

(c) Table of Optimum Point Model Parameters

Figure 5.4 – Results for point model fitting for $Dy^{3+}/\epsilon 168/\theta$, using the hydrogen PCS data set from [145]. The final total error was 722.884 ppm^2. x, y, and z are given in Ångströms, while the tensor parameters are given in $m^3 \times 10^{-32}$



Parameter	x	y	z	χ_{z^2}	$\chi_{x^2-y^2}$	χ_{xy}	χ_{xz}	χ_{yz}	s
Value	-2.71	29.4	-5.09	35.4	32.5	12.9	-5.28	8.79	4.09
Parameter	x	y	z	χ_{ax}	χ_{rh}	α	β	γ	s
Value	-2.71	29.4	-5.09	77.13	-29.98	96.06	11.36	117.2	4.09

Figure 5.5 – Results for delocalised model fitting for $Dy^{3+}/\epsilon 168/\theta$ using the hydrogen PCS data-set from [145]. The final error was 449.247ppm^2. x, y, and z are given in Ångströms, while the tensor parameters are given in $m^3 \times 10^{-32}$, and the width parameter of the Gaussian function, s, is also given in Ångströms.

The complete protein contains three main subunits α, ϵ, and θ, as well as seven ancillary subunits. The structure of the ϵ subunit has been obtained by Hamdan *et al.* [146] via X-ray crystallography while the structure of the θ subunit has been obtained by NMR spectroscopy [144]. It is believed that the α subunit is responsible for DNA replication while the ϵ subunit acts as a proofreader. The function of the θ subunit is currently unknown.

The α and ϵ subunits may form complexes without θ, as may the ϵ and θ without α, but the α subunit does not bind to the θ subunit, suggesting an α–ϵ–θ configuration. In this work, only data sets from the $\epsilon 168/\theta$ complex were used.

In the wild, the protein contains a pair of Mn(II) ions in its active site. When preparing the protein for PCS studies the Mn(II) ions are substituted for a single lanthanide. The effects of the PCS on the chemical shift of each line are isolated as follows:

- Substitute a diamagnetic lanthanide such as Lanthanum for the Mn(II) ions. Record the chemical shits.
- Substitute a paramagnetic lanthanide for the Mn(II) ions. Again, record the chemical shits.
- The difference between each pair of shits is assumed to be entirely down to PCS.

In reality, this process it not necessarily trivial as full assignments of lines to nuclei are required for both substitutions but such assignments are available in the literature.

Two sets of published sets of PCS data and assignments for the $\epsilon 168/\theta$ complex were examined with `pcsfit`. Of these, the first, published by *John et al.* [147], contained chemical shift assignments and values for the methyl groups of $\epsilon 168/\theta$. Unfortunately, in this case, no significant improvement over the point model was observed.

The second set of assignments, published by *John et al.* [145], was more promising due to the novel pulse sequence used. Generally signal enhancements gained through moving to a more sensitive nucleus type when observing chemical shifts near a paramagnetic centre are cancelled out by the enhanced relaxation. The novel pulse sequence avoided this issue by transferring magnetisation from a sensitive nucleus to a less sensitive one allowing PCS observations within as little as 6Å of the paramagnetic centre to be made (compare to the first dataset, which quotes the cut-off as being around 15Å [147]).

Spatial coordinates for the second data set were taken from the PDB coordinate set 1J53. These coordinates were from an X-ray structure (Hamdan *et al.* [146]) so lacked hydrogens. The proprietary program, ChemBio3D, was used to add hydrogens and a Haskell script was written to parse the resulting PDB file and, for each nitrogen, mark the hydrogen closest to the linking nitrogen. It was assumed that these hydrogens would be the hydrogens bonding

to the relevant nitrogens. The resulting PDB file was then used to prepare an input file for `pcsfit` to work on.

Another limitation of this coordinate was that it was that the active site contained the Mn(II) ions so any distortion due to the substitution of Lanthanides would be missing. Unfortunately, no structure of Lanthanide-substitute $\epsilon 168/\theta$ appears to be available. The initial guess for the location of the Lanthanide was set to the mean positions of the two Mn(II).

The results of fitting using the point model are shown in figure 5.4. Both the raw tensor `pcsfit` parameters and axiality and rhombicity are given. As the point model may be evaluated to essentially machine precision, the optimiser is able to run for many iterations and obtain a minimum at close to machine precision. The total squared error obtained was $723 ppm^2$.

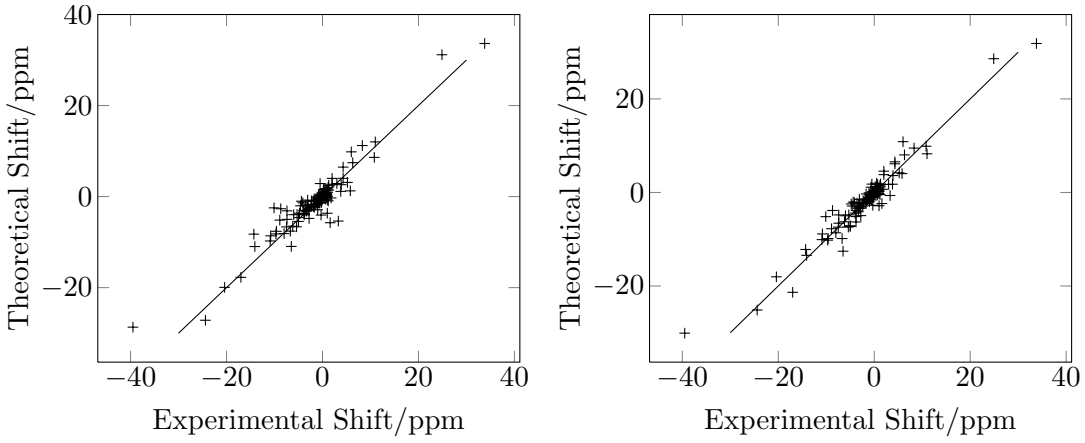
Using the point model-optimised parameters as a starting point, the extended model was fit. The initial value of the width parameter, s, was selected to be the minimum of a one parameter scan of s, with the other parameters held constant. Fitting was then performed in a manner analogous to the point model case.

Because the evaluation of integrals via numerical quadrature does not result in a machine precision answer, the optimiser terminates earlier than the point case. The precision of the optimisation is indirectly controlled by ϵ_{rel} (defined in section 5.4), which controls the precision of the integral. Requesting more precision requires more function evaluations per integration and allows the optimiser to continue for more iterations so there computation cost for increasing the precision. In addition, it is expected that due to the way `pcsfit` is implemented, problems (detailed in section 5.6.6) are likely to occur at very high precision thus it was important not to request unnecessary precision.

Values of 10^{-1}, 10^{-2}, 10^{-3}, and 10^{-4} for ϵ_{rel} were tried in turn, resulting in termination after 49, 73, 119, and 147 iterations respectively. Using $\epsilon_{\text{rel}} = 10^{-4}$ resulted in the numerical error in the model parameters being much less than the reported experimental error of the tensor parameters in [145] so higher precision was not required.

The results are shown of the $\epsilon_{\text{rel}} = 10^{-4}$ are shown in figure 5.5. The squared error drops from $723 ppm^2$ to $449 ppm^2$; an improvement of 37%, despite the fact that only one additional adjustable parameter has been added to the fitting model. The conclusion is that this particular data set is better explained as arising from a delocalised unpaired electron than an unpaired electron concentrated at a point.

Figure 5.6 shows a pair of scatter plots for each of the predicted set of PCS of the point and extended models using the optimised parameters given in figures 5.4 and 5.5 respectively. Each plotted point represents an observed spin. If modelling was perfect then for each



(a) Point Model vs Experimental PCS **(b)** Delocalised Model vs Experimental PCS

Figure 5.6 – Scatter plots of experimental vs theoretical PCS for fitted point and delocalised models, using the optimum fitting parameters given in figures 5.4 and 5.5, respectively. If the model was perfect, all points would lie on a line through the origin with gradient 1. As a guide to the eye, a line of gradient 1 passing through the origin has been drawn. The Pearson product-moment-correlation-coefficients are 0.946 and 0.966 for the point and extended models respectively.

observed nuclear spin and the experimental value of the PCS would match the theoretical value and all the points on these scatter plots would lie exactly on a straight line through the origin of gradient one. Imperfections in the predictions of the two models, any experimental error in the shifts, and any the errors in the positions of the nuclei will tend to move the plot points off the central line. In short, the more scattered the points, the worse the fit.

There are more points close to the middle at, $(0, 0)$, as most of the nuclei are far away from the paramagnetic centre and so exhibit only small PCS. Points near the top right and bottom left represents nuclei that exhibit more extreme values of PCS and thus must be close to the paramagnetic centre. It is just about discernible that the points cluster closer to the trend line when the extended model is used. This can be formalised by computing the Pearson product-moment-correlation-coefficient for each set of results. The computed values are 0.946 and 0.966 for the point and extended models respectively. Still, these diagrams do not give much intuition about the possible effects of the improved fit on the any computed protein structure, which is what is interesting. This will be the topic of the next section.

5.9.2 Potential Differences in Predicted Structures

The ultimate objective of this work is to improve the accuracy of protein structure determination via the measurements of PCS. The standard procedure for structure determination via PCS is to fit a susceptibility tensor, then to use that tensor to obtain structural constraints that can be used with a constraint solver such as Xplor-NIL [81, 148] to obtain a structure.

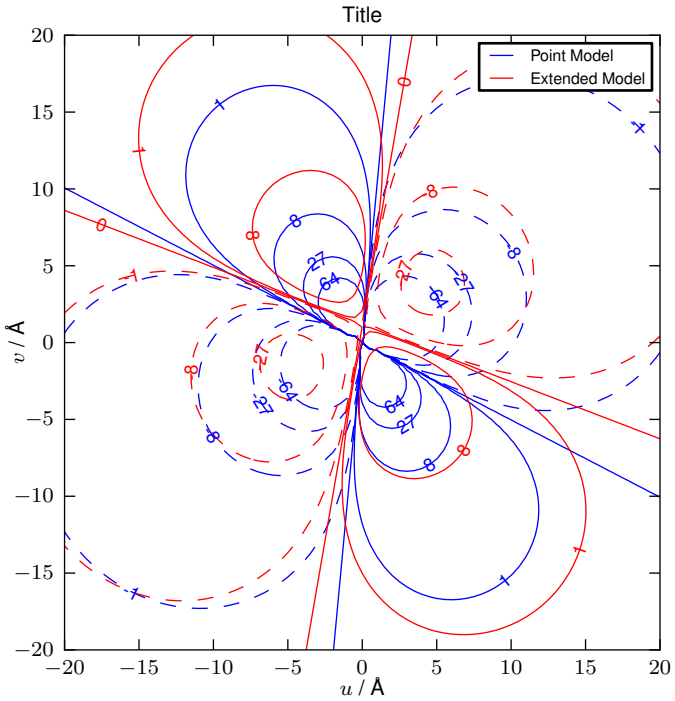


Figure 5.7 – Superimposed contour plots of the predicted PCS for the point and delocalised models, using fitted parameters given in figures 5.4 and 5.5, on a plane passing through the predicted metal centres given by both models. The point model predictions are shown in red and the delocalised model predictions in blue. Contour plots are particularly appropriate to structure determination via PCS distance constraints, as if a spin with a particular PCS is observed, then its position in space can be constrained to fall on a contour, so the differences between red and blue contours of the same level suggest differences in the predicted positions of spins of that particular PCS.

From this new structure, an updated susceptibility tensor may be obtained. This procedure is then iterated until convergence is obtained.

Unfortunately, due to time constraints, it was not possible to attempt iterated structure refinement using the delocalised model of PCS, so the question of what effect the delocalised model would have on the resultant structure is left open. An attempt, however, will be made to speculate on the question of how much of an effect the use of a delocalised model might have on the final calculated structure.

In general, structural constraints can be expressed in terms of isosurfaces. The nuclear Overhauser effect provides distance constraints and so the corresponding isosurfaces are spheres. PCS constraints are of a more complicated nature but if it is known that a particular spin has a given PCS, then that spin must lie on an isosurface whose shape is determined by the functional form of the PCS function (disused in section 5.8). Consequently, contour plots are very relevant when comparing potential models of PCS in terms of where they

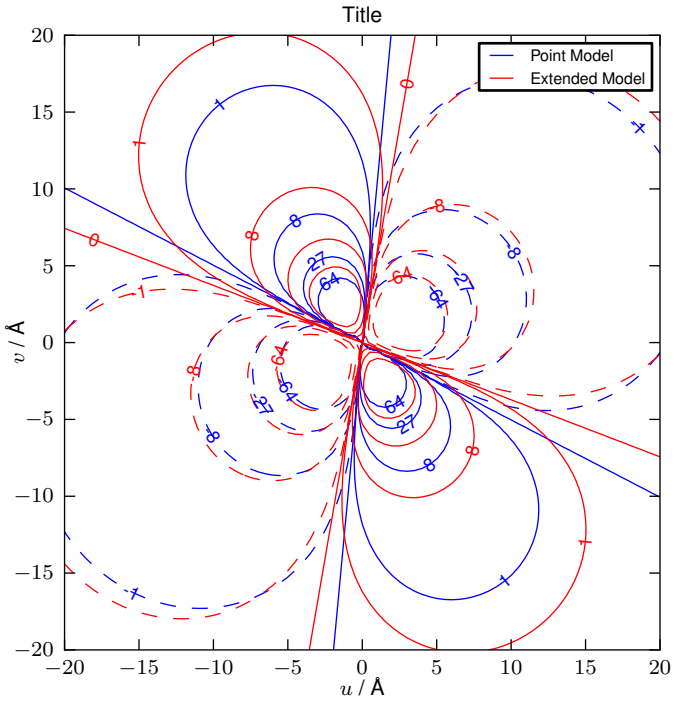


Figure 5.8 – Plot identical to figure 5.7, where a more conservative value of 2.31Å (the covalent radii of dysprosium) has been used for s and the metal positions for both the point and delocalised models have been set equal. Significant differences in line positions are still visible still suggesting a structure different enough to be significant might still be obtained even in this more conservative scenario.

might place spins. Each contour at a particular level represents the locust of points that spin could sit at and if the contours at a particular value of the PCS for two models are different, this will likely translate into differences in the predicted structure once structural refinement is carried out.

Figure 5.7 shows two such superimposed contour plots of calculated PCS for both the point and delocalised model respectively using the optimised parameters from figures 5.4 and 5.5 respectively. The plotting plane represents a 2D plane passing through the protein that is not aligned with the coordinate axes (the coordinate axes are an accident of how the protein was recorded when PDB file 1J53 was saved, so there is no loss in ignoring them). The comparison has been made on a 2D surface like this because comparing the isosurfaces of two functions in 3D is difficult as they tend to obscure each other.

The choice of which plane to plot was somewhat arbitrary. Three points are required to define a plane and two obvious choices were the positions of the paramagnetic centres in each model but there was no obvious choice of what to use as a third point, so the position of the closest observed spin to the paramagnetic centre was selected, though any random

point would have done just as well.

Conceptually, the deviation between the two models can be split into two sources: that coming directly from a non-zero width parameter and that coming from differences in the parameters shared with the point model. As varying the width parameter only changes the PCS function to any significant degree close to the paramagnetic centre, the direct effects of having a non-zero width parameter are only observable by spins close to that centre. Referring to the bottom two plots in figure 5.3, we see that increasing s has the effect of pulling the isosurfaces closer to the paramagnetic centre and so larger values axiality and rhombicity are required to match the same observed PCS for close spins. The increased axiality and rhombicity (as well as differences paramagnetic centre position and tensor orientation) are responsible for pushing out the isosurfaces at longer range.

In figure 5.3 the positions of equivalent contours for each parameter set disagree by significant degrees, suggesting that the point and extended models would predict very different structures after the first iteration has been carried out. The effects after the first iteration are harder to speculate about as the structure and the tensor will have been updated, but it seems likely that some degree of difference would continue to be observed in the protein structure.

5.9.3 The Width Parameter

An open question is why the value of the fitted width parameter for the *John et al.* [145] data-set seems unusually large (4.10Å), compared to the Van der Waals and covalent radii of dysprosium (2.31Å and 1.80Å respectively [149]). Two possible explanations will be offered:

1. The dependence of the error function on the width parameter might have been shallow. This situation would be the opposite of the situation in direct structure fitting, where the error function depends very strongly on the structural parameters (see section 4.1.3); here, the final error might depend only weekly on s and, so, the converged value of s could not be relied on. This explanation is easy to test, and appears to be not the case.

2. The protein structure from PDB coordinate set 1J53 was likely to be distorted compared to the actual structure of the protein in the experiment of John *et al.* The large width parameter could have been an artifact of this fact. This is especially likely as 1J53 is the structure of the protein with a pair of Mn(II) ions while the experiment was performed on the same protein with the Mn(II) ions substituted for Lanthanides.

The first explanation was investigated and ruled out by performing fitting against all parameters of the delocalised model except for the width parameter, s, which was held

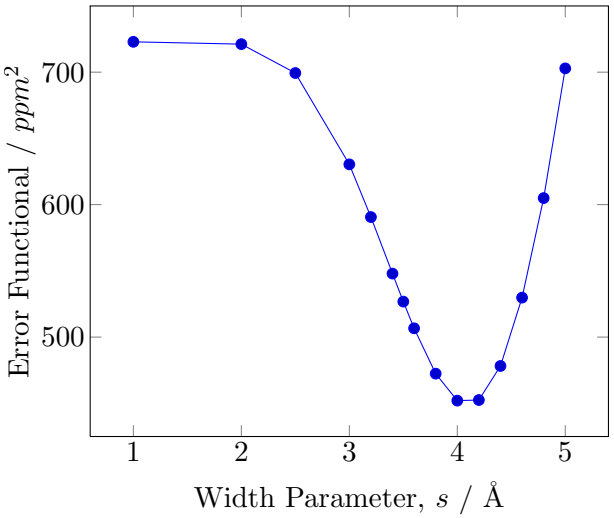


Figure 5.9 – Optimum value of the error function for fixed values of s, when all other values are allowed to vary. The global optimum sits within a deep valley, which suggests that the fitting procedure is sensitive to s.

fixed for a number of values of s. As expected, smaller values of s produce a total error similar to the point model, while values around 4.0 produce a total error similar to the fully unconstrained model. The results are shown in figure 5.9. The optimum s lies within a sharp valley rather than a flat basin and values of s close to 1.78Å produce a total error similar to the point model.

Testing the second explanation would require integration with a constraints solver and would be a major undertaking and is outside of the scope of this work.

A valid question to ask at this point whether the potential structural differences implied by figure 5.7 are exaggerated and are an artifact of the large s. To attempt to gauge what effects a smaller s might produce, figure 5.7 has been plotted again (see figure 5.8) with s set to the covalent radius of dysprosium. Differences are still apparent in the location of corresponding contour lines, suggesting that even with a more reasonable s, significant differences in the predicted structure might be found.

5.9.4 Conclusions

A reasonably efficient and mathematically rigorous method of calculating the PCS of unpaired electrons that are delocalised to a significant extend has been presented. This method has been implemented as a program capable of optimising the traditional PCS fitting parameters along with the width parameter of a Gaussian distribution of electron density using BFGS. Other possible distributions of electron density could be added quite trivially if needed; the underlying mathematics does not depend on any special properties of the Gaussian function.

Fitting usually takes about an hour or so on standard desktop hardware[4] when threading is enabled.

Several data sets have been tested, but only one data set so far has shown a significant improvement in the quality of the fit. Evidence has been presented suggesting that, when differences in the fitted susceptibility tensor parameters between the point and extended models are present, they are likely to lead to differences in the final predicted structure of metalloproteins.

Several open questions still remain, most notably, what the differences in the final structure would be after performing a structural refinement step, using a molecular dynamics stage and constraints obtained from a delocalised model of PCS.

The second question is whether the anomalously large value of the width parameter for PDB coordinate set 1J53 is cause for concern. Two explanations were suggested for the large value (section 5.9.3), one of which was eliminated, one of which remains open.

The next obvious question is that of finding other data sets that benefit significantly from fitting using the delocalised model, as, due to time constraints, only a small number of data sets were investigated here. It is interesting that 1J53, which is an X-ray structure, showed such a large degree of improvement while other structures tested, which had been optimised using a point model of the PCS, showed little or no improvement. It may be that, because the structure has been optimised using a point PCS model, it becomes a self-fulfilling prophecy that the point model explains the structure the best, so in hindsight it is not a surprise that no improvement was observed when the delocalised model was used. If so, then when the structure is permitted to vary along with the PCS model parameters, the expected improvements may begin to appear. This is, however, speculation and would require `pcsfit` to be integrated with a molecular dynamics package to answer.

Of particular interest would be naturally paramagnetic proteins. Most of the differences in the predicted structure between the point and extended models are likely to be concentrated around the paramagnetic centre (which is usually the active site) but if this site has been distorted by the substitution of a large Lanthanide then the information gleaned is of less biological interest than that from a protein where no substitution is required.

Although many open questions remain, it has been demonstrated that delocalised models are worthy of consideration and are likely to become even more important in the future as techniques for observing the PCS of fast relaxing spins close to the metal improve.

[4]Four cores of a single Intel Q6700 CPU were used to compute the results presented in this chapter.

5.10 Acknowledgements

I would like to acknowledge the help of Gottfried Otting here for recommending data sets to investigate.

Chapter 6

Overall Conclusions and Outlook

In this final chapter, the work presented in each of the four research chapters will be summarised and the conclusions drawn restated. In each case, directions for further research will be suggested. These ideas should be considered speculative and are, in general, not fully formed; they are presented in the hope that they will inspire a more thorough research.

6.1 Accuracy of the State Space Restriction Approximation

The conditions required for the spin order pruning method for selecting states to remove from the state space were identified in chapter 2 and were found to be quite forgiving, so long as some kind of relaxation is present in the system under study. The analysis was performed by tracking the flow of coherence through the spin order subspaces and assuming a worst case scenario where the coherence flowing upward is maximised, while that flowing downward is minimised. This picture can be shown to be equivalent to the problem of tracking an incompressible fluid through a system of pipes, and is thus very intuitively appealing.

This method does not give a direct estimation of the differences between FID or spectra (such as, perhaps, bounding $||s - s_{\text{ssr}}||_2$ or $||f - f_{\text{ssr}}||_2$ with s and f being the spectra and FID respectively) but, as pointed out in the later chapter on direct structure fitting, such measures of closeness are only tenuously related to how closeness between spectra might be judged intuitively; the norm of the difference of a pair of spectra that a human might judge to be very similar might, and often will be, quite large.

A worst case approximation for the dynamics of the density matrix norm between spin orders was presented as linear system of equations (system 2.39), repeated here:

$$\frac{\partial|\rho_k|}{\partial t} = -h|\rho_{k+1}| + h|\rho_{k-1}| + rf(k)|\rho_k| - \delta_{1,k}r|\rho_{\text{eq}}| \quad f(k) = k \text{ or } \sqrt{k}. \tag{6.1}$$

The linear system (6.1) for a spin system of n spins contains n equations, so would be amenable to being solved numerically, especially as the matrix representing the system is tridiagonal.

A continuous approximation of the linear system suitable to "pen and paper" analysis via analytic methods was also found. In the time-independent case, this approximation was:

$$\rho(k) = C\exp(-\beta k^{1+\alpha}) + \exp\left(\beta(1-k^{1+\alpha})\right)\frac{r_0}{2h}\Theta(k-1), \tag{6.2}$$

and the time-dependent case:

$$\rho(k,t) = \exp(\beta(1-k^{1+\alpha}))\frac{r_0}{2h}\chi_{[1,2ht+1]}(k) + F(k-2ht)\exp(\beta[(k-2ht)^{1+\alpha} - k^{1+\alpha}]), \tag{6.3}$$

with F is usually taken to be a delta function.

However, although progress has been made on performing error estimates in the specific case where spin order pruning is used, the general question, namely, given a system and a restricted basis set, decide if the simulation be expected to be accurate, is still open. Even resolving this question for the other important special case of this general problem (interaction topography) remains open.

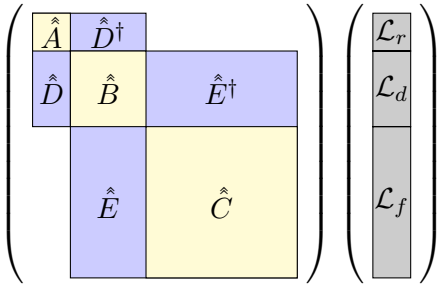
A possible strategy for doing this will be briefly outlined here. Referring to figure 2.2 we see that a particular state cannot be directly connected to other states in a completely arbitrary manner. If a particular state of a n spin system is in the k^{th} spin order, it can connect to at most:

$$3^{k-1}\binom{n}{k-1} + 3^{k}\binom{n}{k} + 3^{k+1}\binom{n}{k+1} - 1, \tag{6.4}$$

other states, which is not exponential in n.

Any coherence that would be lost to the wider system must pass through one of these directly connected states and, as there are at most polynomially many of such directly connected states, an estimation of the coherence that is lost can be found by estimating the amount of coherence that would have flowed into the directly connected space, if that space were being simulated.

This situation, in the linear algebra picture, can be summarised graphically as follows:



. (6.5)

Here $\mathcal{L}_r$, is the restricted space, $\mathcal{L}_d$ is the directly connected space, and $\mathcal{L}_r \oplus \mathcal{L}_d \oplus \mathcal{L}_f$ is the full space. $\mathcal{L}_r$ and $\mathcal{L}_d$ are both polynomial in dimension in the number of spins. $\hat{\hat{A}}$, $\hat{\hat{B}}$, and $\hat{\hat{C}}$ mix their respective subspaces, $\hat{\hat{D}}$ moves coherence from the restricted space to the

directly connected space, and $\hat{\hat{E}}$ moves coherence from the directly connected space to the rest of the system. $\hat{\hat{A}}$, $\hat{\hat{B}}$, and $\hat{\hat{D}}$ are of polynomial size in n, so can conceivably be worked with in computer memory, whereas $\hat{\hat{C}}$ and $\hat{\hat{E}}$ are exponential and so cannot.

The dynamics of the full system, split into subspaces are:

$$\begin{aligned}\frac{\partial\hat{\rho}_r}{\partial t} &= -\frac{i}{\hbar}(\hat{\hat{A}}\hat{\rho}_r + \hat{\hat{D}}^\dagger\hat{\rho}_d)\\ \frac{\partial\hat{\rho}_d}{\partial t} &= -\frac{i}{\hbar}(\hat{\hat{B}}\hat{\rho}_d + \hat{\hat{D}}\hat{\rho}_r + \hat{\hat{E}}^\dagger\hat{\rho}_f)\\ \frac{\partial\hat{\rho}_f}{\partial t} &= -\frac{i}{\hbar}(\hat{\hat{C}}\hat{\rho}_f + \hat{\hat{E}}\hat{\rho}_d),\end{aligned} \tag{6.6}$$

Similar tricks to those that were used to obtain equation (2.27) may be used to obtain the norm density dynamics:

$$\begin{aligned}\frac{\partial|\hat{\rho}_r|^2}{\partial t} &= -\frac{2}{\hbar}\operatorname{Im}\langle\,\hat{\rho}_d\,|\,\hat{\hat{D}}\,|\,\hat{\rho}_r\,\rangle\\ \frac{\partial|\hat{\rho}_d|^2}{\partial t} &= -\frac{2}{\hbar}\operatorname{Im}\langle\,\hat{\rho}_r\,|\,\hat{\hat{D}}^\dagger\,|\,\hat{\rho}_d\,\rangle - \frac{2}{\hbar}\operatorname{Im}\langle\,\hat{\rho}_f\,|\,\hat{\hat{E}}\,|\,\hat{\rho}_d\,\rangle\\ \frac{\partial|\hat{\rho}_f|^2}{\partial t} &= -\frac{2}{\hbar}\operatorname{Im}\langle\,\hat{\rho}_f\,|\,\hat{\hat{E}}^\dagger\,|\,\hat{\rho}_d\,\rangle\,.\end{aligned} \tag{6.7}$$

Unfortunately, unlike the spin order case, there are no convenient patterns or symmetries apparent that could be put to use and knowledge of the evolution history of $\hat{\rho}_r$, $\hat{\rho}_d$, and $\hat{\rho}_f$ appears to be required so a different approach is needed.

The linear system (6.6) can be simplified to

$$\begin{aligned}\frac{\partial\hat{\rho}_r}{\partial t} &= -\frac{i}{\hbar}\hat{\hat{A}}\hat{\rho}_r\\ \frac{\partial\hat{\rho}_d}{\partial t} &= -\frac{i}{\hbar}\hat{\hat{D}}\hat{\rho}_r.\end{aligned} \tag{6.8}$$

where the assumptions principles have been applied:

- The dynamics restricted system is a good approximation of the dynamics of the full system.
- The dynamics of coherence that leaves the restricted system is unimportant, and so need not be simulated.

The first principle seems particularly suspect, as the question of whether the restricted system is a good approximation of the full system hasn't been decided yet, however, for a sufficiently short timescale, this assertion is true (see section 1.4.2).

During a spin dynamics simulation, a set of $\{\hat{\rho}_{r,i}\}$, each representing the state of the system at each time-step, is computed as a matter of course. We can then apply $-i\hat{\hat{D}}$ to each $\hat{\rho}_r$ to obtain the rate of change of $\hat{\rho}_d$ at each time-step. The expression:

$$|-i\hat{\hat{D}}\hat{\rho}_{r,i}\Delta t|^2. \tag{6.9}$$

gives the squared norm that has left the part of the system we are explicitly simulating in time-step i. So long as the sum of these squared norms remains small, $\hat{\rho}_r(t)$ can be trusted.

From the point of view of a user of a simulation package, such as Spinach, it can be imagined that this extra analysis could be enabled via an option, on the understanding that the simulation might become slower than it otherwise would be. If more than a threshold amount of coherence were to be lost, a warning would be issued.

This error estimate has a number of advantages and disadvantages over that presented in 2. The biggest problem is that it is an *a posteriori*, rather than an *a priori* estimate—the result can only be given after a restricted state space simulation has been performed, whereas the chapter 2 estimate can be performed before the simulation. Performing this estimate of a simulation is also likely to be expensive compared to the simulation itself, although not exponentially so. It requires one matrix-vector multiplication per time-step, just like the simulation, but the $\hat{\hat{D}}$ matrix is likely to be bigger than the $\hat{\hat{A}}$ matrix. Finally, this method is not elegant, in that no easily analysable patterns are visible. However, this method does have the advantage of not suffering from the single bucket assumption (section 2.7) and dealing with arbitrary basis sets.

The detailed analysis of the suitability the above method, or the development of other methods to deal with interaction topology-based pruning are left as open questions.

6.2 Spin XML Graphical User Interface

At the conclusion of chapter 3 a formal definition of the spin XML file format and graphical user interface had both been presented. Unless an unanticipated need arises, the definition of the spin XML format should, ideally, remain unchanged from now on, to facilitate interoperability between programs that wish to use the format.

This said, if there ever arises a need to modify spin XML in a manner that breaks backward compatibility, it would be advisable to take that opportunity to move the reference frames to the beginning of the files. This is because, as was noted in section 3.7, interpretation of the components of spins and interactions cannot begin until the reference frames have been read.

There is much scope, however, for further development of the graphical user interface itself.

- **Distribution** The graphical user interface should, in theory, be cross-platform. Certainly, all operating system specific system calls are hidden behind libraries that are cross-platform in nature. Unfortunately, extensive real-world testing has not been performed, particularly on Windows.

- **Reference Frames** Information on reference frames is currently loaded into memory and is preserved when a spin system is saved again, but there is no graphical way to edit them. This needs to be addressed.

- **Spinach Integration** The API for Spinach [22] is now stable enough that a Spinach equivalent to the EasySpin dialogue detailed in section 3.5.9 should now be added.

6.3 Direct Structure Fitting

In chapter 4, the direct fitting of magnetic resonance spectra against structures was shown to be a potentially viable process, and was demonstrated to work using the small molecules cyanomethyl and propargyl and the more complicated tyrosyl radical. More significantly, fitting was performed with tyrosyl radicals embedded in two variants of ribonuclease reductase (RNR) proteins; a clear example of the technique working in a in a biologically significant context (as the orientation of the tyrosyl ring would be difficult to determine by DFT methods alone).

The major difficulty with the direct structure fitting technique remains the enormous amount of computation resources taken up by the quantum chemistry step. While it can be expected that this problem will become less significant over time through Moore's Law alone, any steps that could potentially reduce the computational load are welcome. Probably, the most obvious way that situation could be improved would be by analytical gradients of the magnetic parameters becoming available.

Another potential speed-up that has not been investigated yet would be the use of magnetic resonance parameters prediction. Work has recently been published by *Honlhoff et al.* [150] describing the prediction of the magnetic parameters based on an *purely empirical* formula fit to a very large data set of chemical shifts of protein backbones in the RefDB chemical shift database [151]. The formula used was:

$$\delta_a^{\text{pred}} = \delta_a^{rc} + \sum_{b,c} \alpha_{bc} d_{bc}^{\beta_b c}, \tag{6.10}$$

where δ_a^{pred} was the predicted chemical shift, d_{bc} was the inter-atomic distance between atoms b and c, and α_{ab} and δ_a^{rc} are adjustable parameters, which were fit to a very large database of shifts. Only protein backbones were dealt with by Honlhoff.

While it is not clear whether these predicted parameters would be accurate enough for direct structure fitting, this from of prediction could potentially reduce, or even remove entirely the need for a quantum chemistry step and so seems worth investigated. Even if prediction were only useful for early, coarse, fitting, with quantum chemistry being used

later when the system came closer to convergence, the savings would still be worth the extra complexity.

As was noted in section 4.1.1, the λ parameter from equation (4.3) was chosen in an *ad hoc* manner, but the addition of an energy term to equation (4.3) formally counts as an example of regularisation of an ill-posed problem (details can be found in appendix E). There exists a great deal of formal analysis on the problem of regularisation and the selection of a regularisation parameter, which might be applicable.

Unfortunately, as the forward problem is composition of the quantum chemistry and spin dynamics steps, it is very much non-linear. Much of the theory of ill-posed problems makes the simplifying assumption that the forward problem is linear (such as Tikhonov Regularisation [100], Landweber Iteration [152], L-curves [153, 154], and the Discrepancy Principle of Morozov [100]). Treatments of non-linear forward problems are possible, for example, Kirsch [100] gives a treatment of the inverse scattering problem in his book. Nevertheless, a formal treatment of the structure to spectrum problem most likely represents a formidable mathematical challenge.

A simpler and more practical approach may be to find a linear approximation for the forward problem. The shielding parameters are already evaluated at suitable points for their gradients to be computed as a side effect of computing the total gradient, and gradients of spin dynamics simulations, with respect to the shielding parameters, have become available [102]) so finding a linear approximation to the forward problem seems feasible. The resulting linear approximation could then be used to estimate an optimal λ. It would be expected that the linear approximation of the forward problem would be good when the starting structure is close to the final structure.

6.4 Extended Pseudocontact Shift

Chapter 5 ended with a presentation of a method of fitting susceptibility tensors to models of PCS that included delocalised electrons. The specific case of an electron in a Gaussian distribution was assumed, but the techniques does not use any special properties of such distributions, so could easily generalise to other normalisable distributions of electron density.

The two most important insights were: 1) the use of the mathematical theory of generalised functions to give formal meaning to the convolution integral (which allows the integral to be written in a form that does not diverge when solved via numerical quadrature) and 2) the use of the Cuhre algorithm's ability to process a vector of integrals at once, allowing numerical derivatives to be taken without demanding excessive precision from the quadrature scheme.

The ultimate goal of this work would be to demonstrate that structure fitting using an extended model of PCS resulted in better protein structures than the point model. Progress was made towards this goal, in that a data set was identified where use of the delocalised model resulted in a significantly different susceptibility tensor than when the point model was used. The ultimate effects on the optimised structure are, as yet, unknown as this would require integration of `pcsfit` with a molecular dynamics package. However, it was shown in figure 5.7 that significant differences in the predicted distance constants between the point model and delocalised model exist, so the optimal structures produced by at least the first iteration of structure fitting would likely differ.

It may be significant that PDB coordinate set 1J53, which was found to produce differences between the point and delocalised models was an X-ray structure, while other structures that were tested were found to produce no significant difference were optimised assuming a point model and, so, in hindsight is is perhaps unsurprising that the point model was found to fit them best. It is possible that, when a molecular dynamics phase is added to the optimisation, rendering the structure mutable, the delocalised model will begin to show improvements in data sets such as the one detailed in [147]. Another source of concern was any distortion that may have been caused by the substitution of the Mn(II) ions with a Lanthanide.

In summary, the future direction that this should be taken in are clear: 1) `pcsfit` needs to be integrated with a suitable molecular dynamics package or other distance constraint solver such as Xplor-NIL [81, 148] to allow a full structure optimisation mimicking those already performed via the point model. 2) A demonstration needs to be found that the linked package produces a different structure to the point model for at least one protein (and, preferably, several). The PDB coordinate set 1J53 used in this work would appear to be a good starting point. 3) Once differences are demonstrated, the question of which structure, point-optimised or delocalised-optimised, is the better reflection of reality still remains open.

The tasks detailed in (1), (2), and especially (3) are beyond of the scope of this thesis and are left for future work.

6.4.1 Isosurfaces and the Poisson Equation

The value of the PCS outside of the support of $\rho(\boldsymbol{x})$ is given by the integral:

$$\sigma_{\text{deloc}}(\boldsymbol{x}) = \int_{\mathbb{R}^3} \frac{Y(\theta,\phi)}{r^3} \rho_{\boldsymbol{x}}(r,\theta,\phi) r^2 \sin\theta dr d\theta d\phi, \tag{6.11}$$

which is Lebesgue, thanks to the $\rho(\boldsymbol{x})$ being zero at the singularity. Consider $\nabla^2 \sigma_{\text{deloc}}(\boldsymbol{x})$. As the integral is Lebesgue for special cases of $\boldsymbol{x}$, for these cases the derivative symbol may

be moved within the integral as usual:

$$\nabla^2\sigma_{\text{deloc}}(\boldsymbol{x}) = \int_{\mathbb{R}^3} \frac{Y(\theta,\phi)}{r^3}\nabla^2\rho_{\boldsymbol{x}}(r,\theta,\phi)r^2\sin\theta dr d\theta d\phi, \quad \text{for } \boldsymbol{x} \text{ not in the support of } \rho. \tag{6.12}$$

By integration by parts, it can be shown that $\int_{-\infty}^{\infty} f'(x)g(x)dx = -\int_{-\infty}^{\infty} f(x)g'(x)dx$ when $g(x) \to 0$ as $|x| \to \infty$. By applying this transformation twice for each of the Cartesian directions, we obtain:

$$\nabla^2\sigma_{\text{deloc}}(\boldsymbol{x}) = \int_{\mathbb{R}^3} \rho_{\boldsymbol{x}}(r,\theta,\phi)\nabla^2\left(\frac{Y(\theta,\phi)}{r^3}\right)r^2\sin\theta dr d\theta d\phi, \quad \text{for } \boldsymbol{x} \text{ not in the support of } \rho. \tag{6.13}$$

where ∇ now acts on the dummy variables.

As $Y(\theta,\phi)$ is a spherical harmonic, $\nabla^2\frac{Y(\theta,\phi)}{r^3} = 0$. The integral outside of support of $\rho(\boldsymbol{x})$ must inherit this property through linearity.

$$\nabla^2\sigma_{\text{deloc}}(\boldsymbol{x}) = 0, \quad \text{for } \boldsymbol{x} \text{ not in the support of } \rho. \tag{6.14}$$

This observation suggests that the integral could instead be evaluated by numerical methods for solving Poisson's equation, which are well developed and understood, so long as the value of the $\nabla^2\sigma_{\text{deloc}}(\boldsymbol{x})$ within the support of $\rho(\boldsymbol{x})$ could be found in order to provide a source term. Because the naive integral is not defined in this region, the ∇^2 cannot be brought inside the integral symbol. As was stressed in section 5.5.5, such a function does exist in the mathematical sense, as the function it is supposed to be a derivative of is smooth. The analysis presenting in chapter 5 represents an important step to finding analytic second derivatives that would be required to evaluate the Poisson equations source term.

The advantage of an approach based on partial differential equations is that the value of the integral could be found over all space at once, rather than at specific points, which is important for finding isosurface constraints quickly. Doing so by direct evaluation on a 3D grid would likely be much more computationally expensive so use of Poisson's equation may be the evaluation method of choice for protein structure optimisation.

Appendix A

Definitions Relating to Lie Algebras

This appendix contains a number of definitions relating to Lie Groups and Lie Algebras.

A.1 Groups and their Representations

Definition A.1. *A group is a set, $\mathcal{G}$, together with a binary operation, $\diamond$, that satisfies the following axioms:*

- ***Closure**: $g_1 \diamond g_2 \in \mathcal{G}$ where $g_1, g_2 \in \mathcal{G}$.*
- ***Associativity**: $g_1 \diamond (g_2 \diamond g_3) = (g_1 \diamond g_2) \diamond g_3$ where $g_1, g_2, g_3 \in \mathcal{G}$.*
- ***Identity**: There is an element $I \in \mathcal{G}$ such that for every $g \in \mathcal{G}$ $I \diamond g = g \diamond I = g$.*
- ***Inverse**: For every element $g \in \mathcal{G}$ there is an element $g^{-1} \in \mathcal{G}$ such that $g^{-1} \diamond g = I$.*

The set may be of any cardinality; finite, countably infinite, and uncountably infinite groups are all possible.

Group theory is an extremely important part of both modern mathematics and physics, and many resources on group theory are available, such as Pinter's book, [155], Carter's book [156] or Elliott and Dawber's book [30], which reviews the applications of group theory in physics.

Definition A.2. *Let $\mathcal{G}$ be a group with product $\diamond$, $\mathcal{S}$ be an algebraic structure[1] with a product $\star$, and let ϕ be a map from $\mathcal{G}$ to $\mathcal{S}$. ϕ is a homomorphism if*

$$\phi(g_1) \star \phi(g_2) = \phi(g_1 \diamond g_2) \tag{A.1}$$

where $g_1, g_2 \in \mathcal{G}$.

[1] An algebraic structure is a set along with any number of binary operations defined for all members of the set.

Homomorphisms of groups onto other sets is a key idea. One particular class of group homomorphisms, the group action, is particularly important in physics. A homomorphism does not necessarily preserve the full richness of the structure of the group in question. An extreme example of a homomorphism that erases information would be to map every element of a group to the real number 1 and take the binary operation, $\star$, to be multiplication. Equation A.1 is satisfied, but information has been clearly lost.

Definition A.3. *The action of a group, $\mathcal{G}$, on a set, S, is a homomorphism from the group onto the bijective of maps S to itself.*

That is, for each element of the group, there is a permutation of the members of the set so that those permutations compose in the same way as the group product. For example, the action of the 52 element permutation group on a deck of cards would map each group element to a unique way to shuffle those cards. The action of $SO(3)$ on $\mathbb{R}^3$ maps each group element to a rotation matrix. A matrix is, in effect, a compact way to write a shuffling of all the vectors in a vector space that satisfies certain properties, such as linearity.

The group action of a group element, $g \in \mathcal{G}$, is often written as $g(x)$, where $x \in S$ or even simply $g \cdot x$. The concept of a group action, while seeming abstract in isolation, is key to linking abstract groups to mathematical models of interest in the physical sciences. For example, the action of a member of the rotation group on a vector from $\mathbb{R}^3$ is usually defined as the rotation of that vector.

Definition A.4. *A representation of a group onto a vector space is a homomorphism from the group to either the invertible matrices, or space of automorphisms that act on that vector space.*

In practice, there is little difference between these definitions, but generally casting representation theory in terms of matrix representations emphasises that a basis has been selected for the vector space, while use of automorphisms emphasises that a basis-free approach is being taken.

A subrepresentation of a representation is a subspace of a vector space that is left invariant under that action of a group. This leads to the following definition:

Definition A.5. *If a representation has no non-trivial subrepresentations, it is an* irreducible representation *[14].*

Irreducible representations are of importance in spin dynamics because they correspond to invariant subspaces, and so they can be used to perform basis pruning (section 1.4.1).

From now on, the group product will be implied by the juxtaposition of two group elements, rather than being written out explicitly.

A.2 Lie Groups

Definition A.6. *A Lie group is an group that is also a smooth manifold.*

In other words, elements of the group may be parameterised by a finite number of real parameters and the group product may be written as:

$$\mathcal{G}(\boldsymbol{\alpha})\mathcal{G}(\boldsymbol{\beta}) = \mathcal{G}(f(\boldsymbol{\alpha}, \boldsymbol{\beta})), \tag{A.2}$$

where the function f is smooth.

A.3 Lie Algebras and Lie Algebra Representations

Definition A.7. *A representation of a Lie algebra, $\mathfrak{g}$, over a vector space, V, is a map, r, that carries each element of $\mathfrak{g}$ to an invertible linear operator acting on V such that:*

$$[X, Y](v) = X(Y(v)) - Y(X(v)), \tag{A.3}$$

where $X = r(x)$ and $Y = r(y)$ and $x, y \in \mathfrak{g}$.

Every Lie algebra has exactly one adjoint representation (up to a basis transformation applied to the actual algebra) that may be written down immediately.

Definition A.8. *View the basis elements of a Lie algebra, X, Y, Z... as belonging to a vector space, V, and then let "little"* $\mathrm{ad}(X)$ *(the* adjoint representation of X*) be the matrix that acts on V such that:*

$$\mathrm{ad}(X)Y = [X, Y] \tag{A.4}$$

Note that the left hand side of this equation involves matrix-vector multiplication, while the right hand side is simply the Lie Bracket.

The "big" Ad operation is defined by:

$$\mathrm{Ad}(a)X = aXa^{-1}, \tag{A.5}$$

where a is an invertible matrix, and X and Y are general matrices. There is an important identity (which will not be proved here) linking little ad and big Ad:

$$\exp(\mathrm{ad}(Y))X = \mathrm{Ad}(\exp(Y))X, \tag{A.6}$$

If the Liouville-von Neumann equation is written in these terms:

$$\frac{d\rho}{dt} = -i\,\mathrm{ad}(H)\rho, \tag{A.7}$$

then it can be seen that identity (A.6) links the Hilbert space representation of the dynamics with the Liouville space equivalent. The Liouville-von Neumann equation has the known solution of $\rho(t) = \mathrm{Ad}(\exp(-iHt))\rho(0)$, so the Liouville space solution may be written down immediately: $\exp(-i\,\mathrm{ad}(H)t)\rho(0)$.

Appendix B

$SO(3)$ and its Representations

The "vector valued" representations of $SO(3)$ will be discussed in this appendix. These representations are those that are directly expressible in terms of matrices and vectors, but they are not the only representations. Spinor valued representations of $SO(3)$ may also be found; these were discussed in section 1.2.2. Most of the discussion here is adapted from [34].

Consider the set of functions $\mathcal{F} = \{f : \mathbb{R}^3 \mapsto \mathbb{R}\}$. $\mathcal{F}$ is an infinite-dimensional Hilbert space over which there is a space of linear transforms. The *left regular* and *right regular* representations of $SO(3)$ are given by:

$$L(r)f(\boldsymbol{x}) = f(R^{-1}\boldsymbol{x}) \quad R(r)f(\boldsymbol{x}) = f(R\boldsymbol{x}), \tag{B.1}$$

respectively. r represents an abstract rotation, R is the 3×3 rotation matrix that represents r, and L and R are linear operators acting on the Hilbert space of functions. In a quantum mechanical context they would be written $\hat{L}$ and $\hat{R}$. L and R are the infinite-dimensional regular representations of the rotation group.

Finite-dimensional representations of $SO(3)$ may be obtained by restricting the fs under consideration to sets of functions that transform among themselves under the action of $SO(3)$.

Definition B.1. *Let $\mathcal{H}$ be a space of functions and let a group, $\mathcal{G}$, induce an action on $\mathcal{H}$. Let $\mathcal{F}$ be a subset of functions from $\mathcal{H}$. Then the functions in $\mathcal{F}$ "transform among themselves" under the action of $\mathcal{G}$ when:*

$$\forall g \in \mathcal{G}, f \in \mathcal{F} \quad L(g)f \in \mathcal{F} \tag{B.2}$$

In literature, $\mathcal{F}$ will sometimes be called an "invariant subspace" of $\mathcal{H}$. If $\mathcal{F}$ is finite dimensional then a finite basis set may be chosen, and the action of $\mathcal{G}$ on $\mathcal{H}$ can be described by a matrix.

When dealing with $SO(3)$, the correct choice for the basis of $\mathcal{F}$ is the spherical harmonics. Any $f \in \mathcal{F}$ may be written:

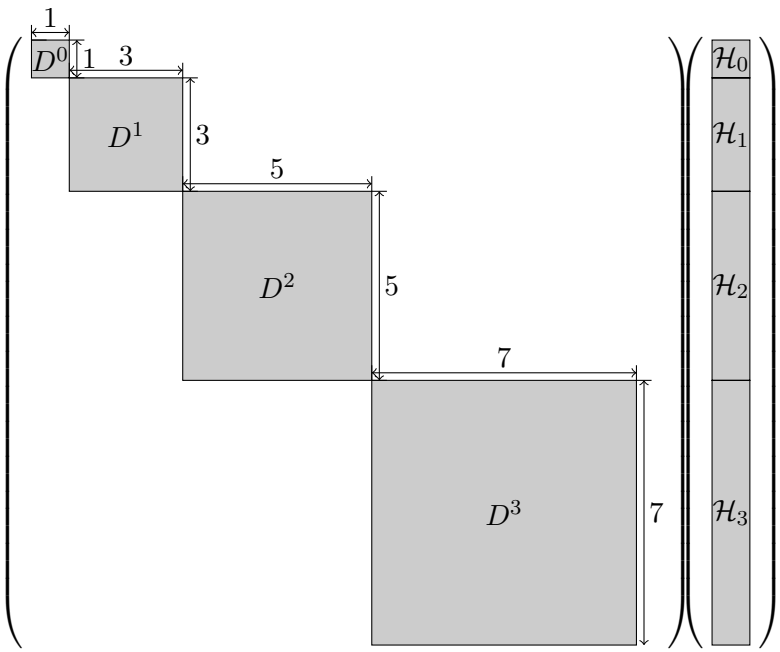


Figure B.1 – The transformation a function, $f : \mathbb{R}^3 \mapsto \mathbb{R}$, by a member, R, of $SO(3)$, is the regular representation of $SO(3)$. By writing f in terms of radial components and spherical harmonics, the regular representation of R becomes block-diagonalised (the radial components of f are not pictured). The blocks are irreducible representations of $SO(3)$ and are known as the Wigner D-matrices. Because $SU(2)$ is a double cover of $SO(3)$, not all of the irreducible representations are found in the regular representation.

$$f(\boldsymbol{x}) = \sum_{l,m} c_{l,m} R(|\boldsymbol{x}|) Y_{l,m}(\boldsymbol{x}) \tag{B.3}$$

where R is the radial function. Now, $L(g)Y_{l,m} \in \mathcal{H}_l$ where $\mathcal{H}_l = \text{span}\{Y_{l,m} : -l \leq m \leq l\}$, so $\mathcal{H}_l$ is an invariant subspace. In other words, in the basis of spherical harmonics, the regular representation is block diagonal. The regular representation written in this basis set is illustrated in figure B.1.

Appendix C

Spin XML Schema

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org
    /2001/XMLSchema">
3   <xs:complexType name="vector">
4     <xs:attribute name="x" type="xs:double" use="required" />
5     <xs:attribute name="y" type="xs:double" use="required" />
6     <xs:attribute name="z" type="xs:double" use="required" />
7     <xs:attribute name="reference_frame" type="xs:integer" use="required
        " />
8   </xs:complexType>
9
10  <xs:complexType name="matrix">
11    <xs:attribute name="xx" type="xs:double" use="required" />
12    <xs:attribute name="xy" type="xs:double" use="required" />
13    <xs:attribute name="xz" type="xs:double" use="required" />
14    <xs:attribute name="yx" type="xs:double" use="required" />
15    <xs:attribute name="yy" type="xs:double" use="required" />
16    <xs:attribute name="yz" type="xs:double" use="required" />
17    <xs:attribute name="zx" type="xs:double" use="required" />
18    <xs:attribute name="zy" type="xs:double" use="required" />
19    <xs:attribute name="zz" type="xs:double" use="required" />
20    <xs:attribute name="reference_frame" type="xs:integer" use="required
        " />
21  </xs:complexType>
22
23  <xs:complexType name="orientation">
24    <xs:choice minOccurs="1" maxOccurs="1">
25      <xs:element name="euler_angles">
26        <xs:complexType>
27          <xs:attribute name="alpha" type="xs:double" use="required" />
28          <xs:attribute name="beta" type="xs:double" use="required" />
29          <xs:attribute name="gamma" type="xs:double" use="required" />
30          <xs:attribute name="reference_frame" type="xs:integer" use="
              required" />
31        </xs:complexType>
32      </xs:element>
33      <xs:element name="angle_axis">
34        <xs:complexType>
```

```
35          <xs:element name="angle" type="xs:double" minOccurs="1"
               maxOccurs="1" />
36          <xs:element name="axis" type="vector" minOccurs="1" maxOccurs=
               "1" />
37        </xs:complexType>
38      </xs:element>
39      <xs:element name="quaternion">
40        <xs:complexType>
41          <xs:attribute name="re" type="xs:double" use="required" />
42          <xs:attribute name="i" type="xs:double" use="required" />
43          <xs:attribute name="j" type="xs:double" use="required" />
44          <xs:attribute name="k" type="xs:double" use="required" />
45          <xs:attribute name="reference_frame" type="xs:integer" use="
               required" />
46        </xs:complexType>
47      </xs:element>
48      <xs:element name="dcm" type="matrix" />
49    </xs:choice>
50  </xs:complexType>
51
52  <xs:complexType  name="reference_frame">
53    <xs:attribute name="number" type="xs:integer" use="required" />
54    <xs:attribute name="label" type="xs:string" use="optional" />
55    <xs:element name="origin" type="vector" minOccurs="1" maxOccurs="1"
         />
56    <xs:element name="orientation" type="orientation" minOccurs="1"
         maxOccurs="1"/>
57    <xs:element name="reference_frame" minOccurs="0" maxOccurs="
         unbounded" />
58  </xs:complexType>
59
60  <xs:complexType name="interaction">
61    <xs:choice minOccurs="1" maxOccurs="1">
62      <xs:element name="scalar" type="xs:dobule" />
63      <xs:element name="tensor" type="matrix" />
64      <xs:sequence>
65        <xs:choice minOccurs="1" maxOccurs="1">
66          <xs:element name="eigenvalues">
67            <xs:complexType>
68              <xs:attribute name="XX" type="xs:double" use="required" />
69              <xs:attribute name="YY" type="xs:double" use="required" />
70              <xs:attribute name="ZZ" type="xs:double" use="required" />
71            </xs:complexType>
72          </xs:element>
73          <xs:element name="axiality_rhombicity">
74            <xs:complexType>
75              <xs:attribute name="iso" type="xs:double" use="required"
                   />
76              <xs:attribute name="ax" type="xs:double" use="required" />
77              <xs:attribute name="rh" type="xs:double" use="required" />
78            </xs:complexType>
79          </xs:element>
80          <xs:element name="span_skew">
81            <xs:complexType>
```

```
82              <xs:attribute name="iso" type="xs:double" use="required"
                    />
83              <xs:attribute name="span" type="xs:double" use="required"
                    />
84              <xs:attribute name="skew" type="xs:double" use="required"
                    />
85            </xs:complexType>
86          </xs:element>
87        </xs:choice>
88        <xs:element minOccurs="1" maxOccurs="1" name="orientation" type=
              "orientation" />
89      </xs:sequence>
90    </xs:choice>
91    <xs:attribute name="kind" use="required">
92      <xs:simpleType>
93        <xs:restriction base="xs:string">
94          <xs:enumeration value="hfc" />
95          <xs:enumeration value="shielding" />
96          <xs:enumeration value="quadrupolar" />
97          <xs:enumeration value="scalar" />
98          <xs:enumeration value="dipolar" />
99          <xs:enumeration value="g-tenser" />
100         <xs:enumeration value="zfs" />
101         <xs:enumeration value="exchange" />
102         <xs:enumeration value="custom" />
103       </xs:restriction>
104     </xs:simpleType>
105   </xs:attribute>
106   <xs:attribute name="units" type="xs:string" use="required" />
107   <xs:attribute name="spin_1" type="xs:integer" use="required" />
108   <xs:attribute name="spin_2" type="xs:integer" use="optional" />
109   <xs:attribute name="label" type="xs:string" use="optional" />
110 </xs:complexType>
111
112 <xs:element name="spin_system">
113   <xs:complexType>
114     <xs:sequence minOccurs="1" maxOccurs="1">
115       <xs:element minOccurs="0" maxOccurs="unbounded" name="spin">
116         <xs:complexType>
117           <xs:element minOccurs="0" maxOccurs="1" name="coordinates"
                  type="vector" />
118           <xs:attribute name="number" type="xs:integer" use="required"
                  />
119           <xs:attribute name="isotope" type="xs:string" use="required"
                  />
120           <xs:attribute name="label" type="xs:string" use="optional"
                  />
121         </xs:complexType>
122       </xs:element>
123       <xs:element minOccurs="0" maxOccurs="unbounded" name="
              interaction" type="interaction" />
124       <xs:element minOccurs="0" maxOccurs="unbounded" name="
              reference_frame" type="reference_Frame"/>
125       </xs:element>
```

```
126        </xs:sequence>
127      </xs:complexType>
128    </xs:element>
129 </xs:schema>
```

Appendix D

Graphical User Interface Build Process

As with many large bodies of C/C++ code, setting up a build on a particular operating system/compiler is far from trivial. This section contains notes on how the build process works on supported platforms and hints on setting up a build on new platforms.

Whenever possible, so-called "out of source" builds should be used. In an "out of source" build, the source directory is not changed, and all intermediate and output files from the build are kept in a separate directory. "Out of source" builds are often desirable when working with version control systems, because temporary files often clutter up status reports and may end up accidentally being committed. Version control systems can be set to ignore certain files, but an out of source build will eliminate the problem entirely. Out of source builds are also the recommended practice when working with `cmake`.

Define	**Windows**	**Windows Debug**	**Linux**	**Linux Debug**	**Comment**
`_FILE_OFFSET_BITS`	✗	✗	✔	✔	
`_LARGE_FILES`	✗	✗	✔	✔	
`__WXGTK__`	✗	✗	✔	✔	Used by wxWidgets
`EIGEN_D...ALLY`	✔	✔	✔	✔	See text
`WIN32`	✔	✔	✗	✗	
`_WINDOWS`	✔	✔	✗	✗	
`_DEBUG`	✗	✔	✗	✗	
`NDEBUG`	✔	✗	✔	✗	Standard C++ define
`_UNICODE`	✔	✔	✗	✗	For windows Unicode
`UNICODE`	✔	✔	✗	✗	For windows Unicode

Table D.1 – C preprocessor defines that should be set for debug and release builds on Windows and Linux. `EIGEN_D...ALLY` is short for `EIGEN_DONT_ALIGN_STATICALLY`

D.1 Building on the Linux Operating System

Building on Linux is a four-step process: 1) generate the wxFormBuilder files and copy to the build directory; 2) copy the contents of the `/data` to the build directory; 3) generate the make files; 4) run `make`. The script `setup-build.sh`, found at the root of the source directory, will perform steps (1) and (2). `cmake` handles the generation of the make files. The compilers that are known to work on Linux are g++-4.3 to g++-4.5. Other recent versions of g++ will almost certainly work. The commands to generate the complete build are:

```
$../src/setup-build.sh
$cmake ../src/
$make
```

Generally, after most changes that involve editing the source files only, just the `make` command needs to be rerun. If new source files have been created, `cmake` must be rerun, and, if wxFormBuilder has been used to edit the GUI template, `setup-build.sh` must be rerun.

On systems tested, `cmake` is able to find wxWidgets and Boost. Other libraries are not found automatically by `cmake` and should be installed in standard locations (see table D.2).

Component	**Location**	**Comment**
Eigen	`/usr/local/include/eigen3/`	Reasonable standard location for placing 3rd party include files.
sigc++	`/usr/include/sigc++-2.0/`	

Table D.2 – Header file locations for libraries not automatically found by `cmake`.

There are two builds available, debug and release. They can be selected by uncommenting the appropriate one of the two lines:

```
set(CMAKE_BUILD_TYPE Debug)
set(CMAKE_BUILD_TYPE Release)
```

The release build enables optimisations, while the debug build enables debugging symbols. These symbols allow debuggers, such as GDB to work with human-readable names.

D.2 Building on this Windows Operating System

MSVC++ was chosen as the compiler for Windows because the current version of GCC cannot be used here, due to a bug encountered when compiling `g03.cpp`. To avoid bugs

due to subtle incompatibilities between compiler versions, all the supporting libraries not included in the Windows API were compiled with the same version of MSVC++. At the time of writing, the only version of MSVC++ which is simultaneously supported by all the supporting libraries is the 2008 version. MSVC++ provides build automation in the form of project files and solution files (a solution is simply a collection of related projects).

The following comments are hints on compiling the supporting libraries. More complete instructions are usually published with the libraries.

- **Boost** Run "Visual Studio 2008 command prompt" to obtain a terminal with the compiler and supported tools available on the system path. From here, navigate to the folder containing Boost. Run `bootstrap` followed by `./bjam`.
- **sigc++** A project file is provided with this library that was found to work.
- **Eigen** Eigen is a header only library and so no building is required.
- **OpenGL** OpenGL and the OpenGL utility toolkit are part of the windows API and so no building is required.
- **wxWidgets** A project file is provided for an older version of MSVC++ which must be converted to the 2008 format. MSVC++ will offer to do this automatically when the older project file is opened. wxWidgets may then be built with the new project file.

Windows lacks a centralised registry of installed libraries or a standard way to locate installed libraries or headers. Many libraries fail to even define a way to locate them on the file system. Therefore, no attempt has been made to ensure the build process on Windows is able to find the libraries automatically. Instead, all of the libraries should be placed on a path relative to the MSVC++ solution file. Consequently, the Windows build environment as a whole is a folder with a known directory structure with all build resources, including the solution file, being at predictable relative locations.

The directory structure of the build environment is as follows:

- `\auto` Files build by wxFormBuilder, such as `SpinachGUI.cpp`, should be copied here.
- `\boost_1_16_1` Contains the unpacked boost source tree.
- `\distributables` Contains builds of the GUI, packed in Windows .zip files.
- `\eigen-3.0.0` The eigen source tree.

- \libsigc++-2.2.9 The sigc++ signals library.
- \project Contains the MSVC "solution" file and files generated by the build. The solution file is spinachgui.sln and references the two project files gui.vcproj and spinxml.vcproj.
- \src The GUI source tree checked out from the repository.
- \wxWidgets-2.8.12 the wxWidgets source tree.

Libraries may be copied verbatim into their respective folders in the build environment. All paths are relative, so there is no special locations that the build environment expects to be.

Appendix E

Regularisation and Ill-Posed Problems

The concept of well-posed and ill-posed problems was first introduced by Hadamard [157]. Hadamard asserted that, for a mathematical system to be an acceptable candidate model for a physical system, it was required that it has to fulfil certain requirements.

Definition E.1. *A well-posed problem is the problem of finding x for a given y and K in the equation $Kx = y$, where $x \in X$, $y \in Y$, X and Y are normed spaces, and K is a mapping $K : X \to Y$ that satisfies the following three conditions (taken from [100]).*

- ***Existence**: For every $y \in Y$, there is an $x \in X$ such that $Kx = y$.*
- ***Uniqueness**: For every $y \in Y$, there is, at most, one $x \in X$ with $Kx = y$.*
- ***Stability**: The solution x depends continuously on y,* i.e.*, for every sequence $x_n \subset X$ with $Kx_n \to Kx$ as $n \to \infty$, it follows that $x_n \to x$ as $n \to \infty$.*

Problems of this type that fail to be well-posed are ill-posed.

One does not need to look far to find problems that are physically interesting but ill-posed. Simply finding the derivative of a function, namely solving the following for x:

$$y(t) = \int_0^t x(s)ds, \tag{E.1}$$

turns out to be ill-posed (see [100] for a full working of this example) as small perturbations in y may result in unbounded errors in x. A symptom of this ill-posedness is that attempting to take the derivatives of a function subject to a white noise perturbation numerically will tend to result in a large error in the derivative that tends to infinity as the step size is decreased to zero.

Acceptable solutions to many ill-posed problems may be obtained in practice if additional information about the solution is available, or assumed, using the technique of regularisation.

In many classical examples, it is assumed that the derivatives of the solution are bounded, or that the norm of the solution is minimised [100], although other assumptions have been used or suggested such as Shannon Entropy and even Kolmogorov Complexity [101].

Appendix F

Overview of Optimisation Techniques

Two common function minimisation techniques are introduced in this appendix: the gradient-based BFGS method and the gradient-free Nelder-Mead simplex algorithm. They are independently relevant to chapters 4 and 5.

F.1 BFGS

The BFGS method, which is described in [158–161], is an optimisation algorithm from the quasi-Newton family of function minimisers. A key property shared by all quasi-Newton methods is that the objective function is approximated as a quadratic function around the current search point [98]. BFGS has the advantage of only requiring the gradient to be found at each function evaluation, instead of a full Hessian. The algorithm updates an estimate of the Hessian each iteration, based on the history of the observed gradients, conferring many of the benefits of Hessian-based methods to the BFGS method.

For pure quadratic functions of the form:

$$F = \boldsymbol{x}^{\mathrm{T}} \cdot A \cdot \boldsymbol{x} - \boldsymbol{b}^{T} \cdot \boldsymbol{x} + c \tag{F.1}$$

with A, $\boldsymbol{b}$, and c being an arbitrary matrix, vector, and scalar respectively, and if equation (F.1) is positive definite, then the BFGS method can be shown to obtain the true minimum in no more than n steps, where n is the dimension of the problem [159].

F.2 Nelder-Mead

The Nelder-Mead simplex algorithm [162] (convergence properties of the algorithm are discussed in [163]) is a popular optimisation algorithm that does not require gradients. The algorithm works by maintaining a list of $N+1$ points in an N-dimensional search space that

form a simplex. Transformations (reflections, expansions and contractions) are performed on the simplex in order to approach the minimum. After the first iteration, at most, only two additional function evaluations are required per iteration.

The Nelder-Mead algorithm generally converges more slowly than gradient-based methods such as BFGS, therefore it is only advisable to use it when gradients are not available. It was largely used during development of the software described in chapters 4 and 5.

Appendix G

`pcsfit` Usage

G.1 Calling `pcsfit` From the Command Line

The user interface of `pcsfit` follows a fairly standard pattern for a Linux command line application. It is invoked using the following syntax:

`./pcsfit` ⟨command⟩ ⟨arguments⟩

The valid values for ⟨command⟩ include `fit`, `scan`, `errorscan`, `eval`, `evalplane`, and `selftest`.

If `selftest` is specified, `pcsfit` performs a number of sanity checks on random data. These are specified in section 5.6.2. `scan` performs a one- or two-parameter scan by varying parameter(s) of the model, while `errorscan` performs a similar function but, instead, compute the value of the error function of a model against experimental data. `eval` evaluates a fixed model at a fixed point in space over a grid of points, `evalplane` evaluates a fixed model on a 2D plane, specified as a set of three points (this mode was used to obtain the data for figure 5.7).

A non-exhaustive list of the most important command line options follows:

`-i,--input-file` Specifies the file containing the atomic coordinates and PCS shifts for a molecule.

`-p,--params-file` Specifies the file specifying the starting parameters and model type to use in the fitting.

`--threads` If this option is set, computation of PCS shifts will be shared over available CPU cores.

`--param-to-scan` When using the `scan` or `errorscan` commands, this option specifies which parameter the scan should be performed over. The parameter is specified as a number corresponding to the line in the parameter file at which the required parameter appears.

--param-to-scan2 This option works in the same way as `param-to-scan`. If set, then the scan will be performed over two parameters, rather than just one.

--min,--max The minimum and maximum values of the parameter over which to scan.

--min2,--max2 The minimum and maximum values of the second parameter to scan over when performing 2D scans.

--steps The number of steps over which each parameter should be scanned.

--absError,--relError The values of absolute and relative errors. These parameters are passed to the Cuhre numerical integrator.

--seed An integer that will be used to initialise the random number generator, should random numbers be used. This option is only relevant when using the `selftest` command.

G.2 Examples

```
$ pcsfit fit -i example_molecule.dat -p starting.model
```

Run `pcsfit` on input file `example_molecule.dat`, starting with the model specified in `starting.model`. This example is the most common way to run the program.

```
$ pcsfit errorscan -i example_molecule.dat -p
starting.model --steps 40 --param-to-scan 9 --min 0.3 --max
4.0
```

Compute 40 values of the error function for the model contained in `starting.model` against the data-set contained in `example_molecule.dat`. The value of model parameter 9 (the width parameter) is adjusted from 0.3 to 4.0 Ångströmsover the course of the scan.

Appendix H

Raw PCS Data

Table H.1 – Spatial coordinates, calculated and experimental values of the PCS of the dataset used in section 5.9. The fitting procedure was illustrated in figure 5.5. Experimental data was taken from *John et al.* [145].

x/Å	y/Å	z/Å	**Exp./*ppm***	**Point/*ppm***	**Deloc./*ppm***
7.63	45.	-14.3	0.1	-0.00493	-0.201
2.49	22.9	-2.56	-39.5	-28.70	-30.1
2.42	19.4	-1.06	-14.1	-11.00	-13.4
1.5	21.	2.22	-10.8	-8.62	-8.87
3.19	23.	5.02	-4.8	-3.65	-2.45
4.45	22.7	8.6	-1.7	-0.984	-0.319
2.72	24.4	11.5	-0.7	0.522	1.13
10.3	26.3	8.87	-0.3	-0.384	0.15
6.91	26.	7.06	-0.1	-0.789	0.248
8.48	24.5	3.89	-1.	-2.90	-1.98
10.9	22.	5.32	-1.4	-1.82	-1.53
10.2	18.6	3.79	-1.7	-2.35	-2.36
7.6	19.9	1.37	-2.8	-4.80	-4.96
7.7	20.8	-2.32	-5.4	-6.63	-7.4
6.22	23.3	-4.81	-6.5	-10.90	-12.6
2.86	34.1	-12.3	11.	12.00	8.27
1.36	36.9	-14.3	3.	2.87	1.72
3.1	40.	-15.7	0.7	0.758	0.243
1.77	43.5	-15.2	-0.3	-0.828	-1.1
3.41	46.5	-16.8	-0.3	-0.439	-0.592
2.22	50.1	-16.4	-0.5	-0.654	-0.747
-0.837	48.8	-14.5	-0.8	-1.31	-1.43
-1.84	46.5	-17.4	-0.9	-1.04	-1.2
-1.77	42.7	-17.7	-1.2	-0.818	-1.06
0.584	41.9	-20.6	-0.2	0.21	0.0719
-0.651	38.4	-21.1	0.4	0.959	0.943
2.89	37.	-21.	0.7	1.76	1.69
2.72	34.7	-18.	2.	4.00	3.81
4.61	31.7	-16.4	4.3	6.48	6.13

Continued on next page

Table H.1 – *Continued from previous page*

x/**Å**	y/**Å**	z/**Å**	**Exp./*ppm***	**Point/*ppm***	**Deloc./*ppm***
2.81	28.5	-15.5	6.	9.83	10.9
3.21	22.	-12.7	1.3	1.44	1.21
6.97	16.6	-9.98	-2.4	-2.01	-2.89
6.79	12.9	-2.51	-2.9	-2.70	-3.4
3.06	12.7	-1.81	-3.3	-3.18	-4.11
0.636	15.6	-1.5	-5.	-5.46	-7.28
-0.731	15.9	2.04	-6.	-3.97	-4.99
-5.07	18.8	3.69	-3.6	-1.85	-2.27
-3.23	20.3	0.711	-6.7	-7.59	-9.86
-5.17	18.1	-1.7	-8.9	-5.16	-7.75
-8.36	19.4	-0.104	-4.1	-1.36	-2.19
-7.35	22.9	-1.19	-3.1	-0.78	-1.49
-6.05	22.5	-4.75	-14.3	-8.25	-12.2
-7.24	19.	-5.75	-7.4	-3.13	-4.81
-4.	18.2	-7.63	-5.	-4.62	-7.12
-3.52	14.5	-7.67	-8.7	-2.67	-3.86
-1.	14.1	-12.4	-1.9	-1.06	-1.42
2.5	15.2	-11.4	-2.3	-1.69	-2.45
4.37	12.1	-12.4	-1.7	-1.04	-1.46
5.2	13.2	-15.9	-1.	-0.314	-0.495
5.92	16.9	-15.	-0.4	-0.151	-0.417
11.2	19.	-11.6	-0.9	-0.648	-1.15
11.9	22.4	-10.1	-1.1	-0.528	-1.14
15.5	22.	-11.2	-0.3	-0.122	-0.463
14.4	21.8	-14.8	-0.3	0.354	0.0682
12.4	25.1	-14.7	0.2	1.20	0.798
14.5	27.1	-12.3	0.3	0.908	0.466
16.6	28.9	-14.9	0.5	0.916	0.626
13.6	30.2	-17.	0.2	1.56	1.24
11.7	30.9	-13.7	1.3	2.48	1.81
14.6	33.2	-12.6	1.	1.44	0.988
14.8	34.8	-16.	0.8	1.26	0.943
11.1	35.6	-15.8	1.2	2.03	1.53
14.5	38.8	-12.1	1.1	0.95	0.641
14.4	42.5	-11.2	0.8	0.493	0.309
10.6	42.7	-10.9	0.7	0.422	0.173
8.42	43.5	-7.96	0.7	-0.209	-0.311
6.83	40.3	-6.89	1.	0.0103	-0.221
3.31	40.7	-5.49	1.	-3.66	-2.84
2.26	37.7	-3.36	1.6	-5.74	-2.37
-0.609	37.2	-0.893	3.3	-5.38	-0.655
1.25	36.2	2.3	6.3	7.45	8.03
4.6	36.6	0.506	5.2	3.12	4.2
6.67	34.9	3.22	3.9	2.61	3.61
4.9	31.6	2.43	4.3	4.02	6.55

Continued on next page

Table H.1 – *Continued from previous page*

x/**Å**	y/**Å**	z/**Å**	**Exp./*ppm***	**Point/*ppm***	**Deloc./*ppm***
5.76	31.4	-1.27	5.8	1.26	4.03
9.13	33.2	-0.97	3.8	1.14	1.76
10.	30.7	1.73	1.8	-0.298	0.677
11.	29.4	-3.54	0.6	-0.557	-0.557
14.3	30.	-1.66	0.7	-0.265	-0.167
14.	26.5	-0.184	-0.2	-1.22	-1.04
13.7	24.9	-3.63	-0.6	-1.44	-1.61
16.4	27.2	-5.04	-0.1	-0.29	-0.474
18.8	26.2	-2.2	-0.1	-0.439	-0.49
18.3	22.5	-2.84	-0.6	-0.757	-0.873
19.8	23.	-6.28	-0.3	-0.363	-0.529
23.2	23.7	-4.74	-0.1	-0.233	-0.325
24.	26.4	-7.25	0.1	-0.0171	-0.12
21.5	31.	-5.9	0.1	0.153	0.0494
14.9	36.	-3.98	0.9	0.651	0.55
14.6	39.4	-2.22	0.9	0.467	0.492
18.2	40.2	-3.31	0.7	0.32	0.301
17.2	40.2	-7.	0.9	0.455	0.342
13.5	40.9	-6.74	0.9	0.546	0.394
11.5	43.4	-4.75	0.9	0.128	0.129
8.89	41.6	-2.67	1.	0.114	0.295
5.49	43.	-1.82	1.2	-0.931	-0.445
3.11	41.2	0.5	1.	-0.724	0.215
-0.375	42.	-0.794	-0.3	-4.06	-2.74
0.058	43.1	4.51	0.7	0.12	0.548
-1.03	45.8	2.1	-0.3	-1.30	-0.929
-4.66	44.7	2.4	-1.1	-1.60	-1.38
-4.46	44.5	6.19	-0.4	-0.00881	0.121
-3.09	48.1	6.3	-0.2	-0.316	-0.191
-5.91	49.4	4.07	-0.7	-0.806	-0.762
-8.64	47.4	5.77	-0.7	-0.479	-0.51
-9.9	44.8	11.5	0.2	0.443	0.427
-11.5	43.2	8.42	0.2	0.356	0.243
-8.24	36.6	2.46	-0.5	2.86	1.78
-8.1	36.7	-1.34	-9.7	-8.21	-10.2
-11.8	35.7	-1.47	-10.1	-2.48	-5.18
-12.8	38.4	0.931	-4.4	-1.02	-2.05
-10.9	40.9	-1.25	-5.3	-3.92	-4.79
-12.6	39.6	-4.36	-7.4	-5.00	-6.58
-16.	40.4	-2.86	-4.6	-2.03	-2.9
-14.8	43.8	-1.66	-3.2	-1.91	-2.44
-13.6	44.7	-5.16	-3.2	-2.78	-3.33
-16.2	42.8	-7.1	-3.	-2.31	-2.97
-13.7	40.4	-8.75	-3.8	-4.00	-5.16
-14.9	37.1	-10.1	-4.	-3.12	-4.43

Continued on next page

Table H.1 – *Continued from previous page*

x/**Å**	y/**Å**	z/**Å**	**Exp./*ppm***	**Point/*ppm***	**Deloc./*ppm***
-13.8	34.1	-8.08	-3.8	-3.77	-6.31
-16.7	31.7	-9.04	-2.3	-0.946	-1.77
-14.3	29.4	-10.8	-1.1	-0.0875	-0.0752
-12.2	29.1	-7.65	-0.8	0.278	-0.821
-13.9	26.2	-5.84	0.5	1.67	1.96
-10.4	24.7	-5.82	2.	2.71	4.55
-7.39	26.9	-5.69	8.3	11.20	9.49
-4.86	26.9	-8.52	24.9	31.20	28.6
-2.23	30.5	-12.1	33.8	33.60	31.9
-5.89	31.	-11.2	10.8	8.61	9.88
-2.26	34.7	-9.4	-20.4	-19.90	-18.
-4.69	36.1	-12.	-9.6	-7.62	-9.85
-7.34	36.8	-9.43	-17.	-17.70	-21.4
-4.8	38.5	-7.13	-24.4	-27.20	-25.1
-3.56	40.6	-10.	-10.9	-9.75	-10.1
-7.08	42.	-10.6	-7.5	-6.66	-7.42
-7.54	42.5	-6.9	-8.	-8.12	-8.63
-4.19	44.2	-6.46	-6.2	-6.51	-6.27
-4.72	46.5	-9.45	-3.8	-3.71	-3.83
-8.06	47.6	-7.98	-3.4	-3.16	-3.38
-6.6	47.9	-4.49	-2.8	-3.04	-3.09
-3.88	50.2	-5.86	-1.9	-2.19	-2.17
-6.14	51.8	-8.46	-1.6	-1.76	-1.84

Bibliography

[1] A. Karabanov, I. Kuprov, G. T. P. Charnock, A. Van Der Drift, L. J. Edwards, and W. Köckenberger, "On the accuracy of the state space restriction approximation for spin dynamics simulations," *Journal of Chemical Physics*, vol. 135, no. 8, 2011.

[2] G. Charnock, M. Krzystyniak, and I. Kuprov, "Molecular structure refinement by direct fitting of atomic coordinates to experimental ESR spectra," *Journal of Magnetic Resonance*, vol. 216, no. 0, pp. 62–68, 2012.

[3] S. R. White, "Density matrix formulation for quantum renormalization groups," *Phys. Rev. Lett.*, vol. 69, pp. 2863–2866, 1992.

[4] U. Schollwöck, "The density-matrix renormalization group," *Reviews of Modern Physics*, vol. 77, pp. 259–315, 2005.

[5] K. A. Hallberg, "New trends in density matrix renormalization," *Advances in Physics*, vol. 55, no. 5-6, pp. 477–526, 2006.

[6] I. Kuprov, N. Wagner-Rundell, and P. J. Hore, "Polynomially scaling spin dynamics simulation algorithm based on adaptive state-space restriction," *Journal of Magnetic Resonance*, vol. 189, no. 2, pp. 241–250, 2007.

[7] M. H. Levitt, *Spin Dynamics*. Wiley, 2001.

[8] R. R. Ernst, G. Bodenhausen, and A. Wokaun, *Principles of Nuclear Magnetic Resonance in One and Two Dimensions*. Oxford Science Publications, 1987.

[9] P. J. Hore, J. A. Jones, and S. Wimperis, *NMR: The Toolkit*. Oxford University Press, 2000.

[10] P. Dirac, *The Principles of Quantum Mechanics, Fourth Edition*. Oxford Science Publications, 1958.

[11] M. E. Peskin and D. V. Schroeder, *An Introduction to Quantum Field Theory*. The Advanced Book Program, 1995.

[12] F. S. Levin, *An Introduction to Quantum Theory*. Cambridge University Press, 2002.

[13] K. Jänich, *Topology*. Springer-Verlag, 1958.

[14] J. H. W. Fulton, *Representation Theory: A First Course*. Springer, 1991.

[15] M. Staley, "Understanding quaternions and the dirac belt trick," *European Journal of Physics*, vol. 31, no. 3, pp. 467–478, 2010.

[16] Y. Aharonov and L. Susskind, "Observability of the sign change of spinors under 2π rotations," *Physical Review*, vol. 158, pp. 1237–1238, 1967.

[17] H. Rauch, A. Zeilinger, G. Badurek, A. Wilfing, W. Bauspiess, and U. Bonse, "Verification of coherent spinor rotation of fermions," *Physics Letters A*, vol. 54, no. 6, pp. 425–427, 1975.

[18] S. A. Werner, R. Colella, A. W. Overhauser, and C. F. Eagen, "Observation of the phase shift of a neutron due to precession in a magnetic field," *Physical Review Letters*, vol. 35, pp. 1053–1055, Oct 1975.

[19] A. Abragam, *Principles of Nuclear Magnetism*. Oxford Science Publications, 1961.

[20] K. Blum, *Density Matrix Theory and Applications, Second Edition*. Plenum Press, 1981.

[21] P. Hodgkinson and L. Emsley, "Numerical simulation of solid-state NMR experiments," *Progress in Nuclear Magnetic Resonance Spectroscopy*, vol. 36, pp. 201–239, 2000.

[22] H. J. Hogben, M. Krzystyniak, G. T. P. Charnock, P. J. Hore, and I. Kuprov, "Spinach—a software library for simulation of spin dynamics in large spin systems," *Journal of Magnetic Resonance*, vol. 208, no. 2, pp. 179–194, 2011.

[23] I. Kuprov, "Diagonalization-free implementation of spin relaxation theory for large spin systems," *Journal of Magnetic Resonance*, vol. 209, no. 1, pp. 31–38, 2011.

[24] R. K. Wangsness and F. Bloch, "The dynamical theory of nuclear induction," *Physical Review*, vol. 89, pp. 728–739, 1953.

[25] F. Bloch, "Nuclear induction," *Physical Review*, vol. 70, pp. 460–474, 1946.

[26] A. G. Redfield, "On the theory of relaxation processes," *IBM Journal of Research and Development*, vol. 1, pp. 19–31, 1957.

[27] M. Goldman, "Formal theory of spin-lattice relaxation," *Journal of Magnetic Resonance*, vol. 149, pp. 160–197, Oct 2000.

[28] I. Kuprov, "Polynomially scaling spin dynamics II: Further state-space compression using Krylov subspace techniques and zero track elimination," *Journal of Magnetic Resonance*, vol. 195, no. 1, pp. 45–51, 2008.

[29] H. J. Hogben, P. J. Hore, and I. Kuprov, "Strategies for state space restriction in densely coupled spin systems with applications to spin chemistry," *Journal of Chemical Physics*, vol. 132, no. 17, p. 174101, 2010.

[30] J. P. Elliott and P. G. Dawber, *Symmetry in Physics Volume 1: Principles and Simple Applications*. Macmillan, 1979.

[31] M. Krzystyniak, L. J. Edwards, and I. Kuprov, "Destination state screening of active spaces in spin dynamics simulations," *Journal of Magnetic Resonance*, vol. 210, no. 2, pp. 228–232, 2011.

[32] J. J. Sakurai, *Modern Quantum Mechanics*. Addison-Wesley Publishing Company, 1994.

[33] C. F. Loan, "The ubiquitous Kronecker product," *Journal of Computational and Applied Mathematics*, vol. 123, pp. 85–100, 2000.

[34] W. Rossmann, *Lie Groups, an Introduction Through Linear Groups*. Oxford University Press, 2002.

[35] D. J. H. Garling, *Clifford Algebras: An Introduction*. Cambridge Univerty Press, 2011.

[36] B. H. Bransden and C. J. Joachain, *Quantum Mechanics, Second Edition*. Pearson Prentice Hall, 1989.

[37] T. Helgaker, M. Jaszuński, and K. Rudd, "*Ab Initio* methods for the calculation of NMR shielding and indirect spin-spin coupling constants," *Chemical Reviews*, vol. 99, pp. 293–352, 1999.

[38] J. A. Weil and J. R. Bolton, *Electron Paramagnetic Resonance, Second Edition*. Wiley Interscience, 2007.

[39] Atherton, *Principles of Electron Spin Resonance*. Ellis Horwood PTR Prentice Hall, 1993.

[40] C. P. Slichter, *Principles of Magnetic Resonance, Third Enlarged and Updated Edition.* Springer, 1990.

[41] M. H. L. Pryce, "A modified perturbation procedure for a problem in paramagnetism," *Proceedings of the Physical Society. Section A*, vol. 63, no. 1, pp. 25–29, 1950.

[42] Carrington and McLachlan, *Introduction to Magnetic Resonance.* Harper and Row, 1969.

[43] E. Fermi, "Über die magnetischen Momente der Atomkerne," *Zeitschrift für Physik*, vol. 60, pp. 320–333, 1930.

[44] I. Bertini, C. Luchinat, and G. Parigi, *Solution NMR of Paramagnetic Molecules.* Elsevier, 2001.

[45] I. Bertini, C. Luchinat, and G. Parigi, "Magnetic susceptibility in paramagnetic NMR," *Progress in Nuclear Magnetic Resonance Spectroscopy*, vol. 40, no. 3, pp. 249–273, 2002.

[46] I. Bertini, C. Luchinat, G. Parigi, and R. Pierattelli, "NMR spectroscopy of paramagnetic metalloproteins," *ChemBioChem*, vol. 6, pp. 1536–1549, 2005.

[47] M. John and G. Otting, "Strategies for measurement of pseudocontact shifts in NMR spectroscopy," *ChemPhysChem*, vol. 8, p. 2309, 2007.

[48] R. D. Guildes, S. Sarma, R. J. DiGate, D. Banville, V. J. Basus, I. D. Duntz, and L. Waskell, "Pseudocontact shifts used in the restraint solution structures of electron transfer complexes," *Nature Structural Biology*, vol. 3, pp. 333–339, 1996.

[49] G. M. Palma, K. Suominen, and A. K. Ekert, "Quantum computers and dissipation," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 452, no. 1946, pp. 567–584, 1996.

[50] W. G. Unruh, "Maintaining coherence in quantum computers," *Physical Review A*, vol. 51, pp. 992–997, 1995.

[51] M. Carravetta and M. H. Levitt, "Theory of long-lived nuclear spin states in solution nuclear magnetic resonance. I. singlet states in low magnetic field," *Journal of Chemical Physics*, vol. 122, no. 21, 2005.

[52] G. Pileio and M. H. Levitt, "Theory of long-lived nuclear spin states in solution nuclear magnetic resonance. II. singlet spin locking," *Journal of Chemical Physics*, vol. 130, no. 21, 2009.

[53] A. D. Becke, "A new mixing of Hartree-Fock and local density-functional theories," *Journal of Chemical Physics*, vol. 98, pp. 1372–1377, 1993.

[54] K. Ruud, T. Helgaker, K. L. Bak, P. Jorgensen, and H. J. A. Jensen, "Hartree–Fock limit magnetizabilities from London orbitals," *Journal of Chemical Physics*, vol. 99, pp. 3847–3859, 1993.

[55] V. Barone, *Recent Advances in Density Functional Methods Part I.* World Scientific Publishing Co., 1996.

[56] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, O. Farkas, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford, J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, C. Gonzalez, and J. A. Pople, "Gaussian 03, Revision C.02." Gaussian, Inc., Wallingford, CT, 2004.

[57] J. A. Jones and P. J. Hore, "Spin-selective reactions of radical pairs act as quantum measurements," *Chemical Physics Letters*, vol. 488, no. 1-3, pp. 90–93, 2010.

[58] H. Mouritsen and P. J. Hore, "The magnetic retina: Light-dependent and trigeminal magnetoreception in migratory birds," *Current opinion in neurobiology*, vol. 22, no. 2, pp. 343–352, 2012.

[59] S. Stoll and A. Schweiger, "EasySpin, a comprehensive software package for spectral simulation and analysis in EPR," *Journal of Magnetic Resonance*, vol. 178, no. 1, pp. 42–55, 2006.

[60] M. Veshtort and R. G. Griffin, "SPINEVOLUTION: A powerful tool for the simulation of solid and liquid state NMR experiments," *Journal of Magnetic Resonance*, vol. 178, no. 2, pp. 248–282, 2006.

[61] M. Bak, R. Schultz, T. Vosegaard, and N. C. Nielsen, "Specification and visualization of anisotropic interaction tensors in polypeptides and numerical simulations in biological solid-state NMR," *Journal of Magnetic Resonance*, vol. 154, no. 1, pp. 28–45, 2002.

[62] M. Bak, J. T. Rasmussen, and N. C. Nielsen, "SIMPSON: A general simulation program for solid-state NMR spectroscopy," *Journal of Magnetic Resonance*, vol. 147, no. 2, pp. 296–330, 2000.

[63] J. Herzfeld and A. E. Berger, "Sideband intensities in NMR spectra of samples spinning at the magic angle," *The Journal of Chemical Physics*, vol. 73, no. 12, pp. 6021–6030, 1980.

[64] Y. Siderer and Z. Luz, "Analytical expressions for magnetic resonance lineshapes of powder samples," *Journal of Magnetic Resonance (1969)*, vol. 37, no. 3, pp. 449–463, 1980.

[65] D. W. Alderman, M. S. Solum, and D. M. Grant, "Methods for analyzing spectroscopic line shapes. NMR solid powder patterns," *The Journal of Chemical Physics*, vol. 84, no. 7, pp. 3717–3725, 1985.

[66] D. A. Varshalovich, A. N. Moskalev, and V. K. Khersonskii, *Quantum Theory of Angular Momentum*. World Scientific, 1988.

[67] J. Vince, *Geometric Algebra for Computer Graphics*. Springer-Verlag London, 2008.

[68] D. H. Eberly, *3D Game Engine Design*. Morgan Kaufmann Publishers, 2001.

[69] K. Shoemake, "Animating rotation with quaternion curves," *SIGGRAPH Comput. Graph.*, vol. 19, no. 3, pp. 245–254, 1985.

[70] L. Dorst, D. Fontijne, and S. Mann, *Geometric Algebra*. Morgan Kaufmann, 2007.

[71] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: principles, techniques, and tools*. Addison-Wesley Longman Publishing, 1986.

[72] D. Grune and C. Jacobs, *Parsing Techniques—Second Edition*. Springer US, 2008.

[73] E. Roduner, P. W. Percival, D. G. Fleming, J. Hochmann, and H. Fischer, "Muonium-substituted transient radicals observed by muon spin rotation," *Chemical Physics Letters*, vol. 57, no. 1, pp. 37–40, 1978.

[74] B. S. Institute, *The C Standard Incorporating Technical Corrigendum 1, ISO/IEC 9899:1999*. John Wiley & Sons, Ltd, 2003.

[75] N. J. Stone, "Table of nuclear magnetic dipole and electric quadrupole moments," *Atomic Data and Nuclear Data Tables*, vol. 90, no. 1, pp. 75–176, 2005.

[76] J. Vince, *Mathematics for Computer Graphics, Second Edition*. Springer, 2006.

[77] B. S. *et al.*, *The C++ Standard Incorporating Technical Corrigendum 1, BS ISO/IEC 14882:2003*. John Wiley & Sons, Ltd, 2003.

[78] D. Abrahams and A. Gurtovoy, *C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond (C++ in Depth)*. Addison Wesley, 2004.

[79] G. M. Clore and A. M. Gronenborn, "Determining the structures of large proteins and protein complexes by NMR," *Trends in Biotechnology*, vol. 16, no. 1, pp. 22–34, 1998.

[80] M. Helgstrand, P. Kraulis, P. Allard, and T. Härd, "Ansig for Windows: An interactive computer program for semiautomatic assignment of protein NMR spectra," *Journal of Biomolecular NMR*, vol. 18, no. 4, pp. 329–336, 2000.

[81] C. D. Schwieters, J. J. Kuszewski, and G. Marius Clore, "Using Xplor-NIH for NMR molecular structure determination," *Progress in Nuclear Magnetic Resonance Spectroscopy*, vol. 48, no. 1, pp. 47–62, 2006.

[82] Y. Shen, O. Lange, F. Delaglio, P. Rossi, J. M. Aramini, G. Liu, A. Eletsky, Y. Wu, K. K. Singarapu, A. Lemak, A. Ignatchenko, C. H. Arrowsmith, T. Szyperski, G. T. Montelione, D. Baker, and A. Bax, "Consistent blind protein structure generation from NMR chemical shift data," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, no. 12, pp. 4685–4690, 2008.

[83] W. F. Vranken, W. Boucher, T. J. Stevens, R. H. Fogh, A. Pajon, M. Llinas, E. L. Ulrich, J. L. Markley, J. Ionides, and E. D. Laue, "The CCPN data model for NMR spectroscopy: Development of a software pipeline," *Proteins: Structure, Function and Genetics*, vol. 59, no. 4, pp. 687–696, 2005.

[84] E. Salager, R. S. Stein, C. J. Pickard, B. Elena, and L. Emsley, "Powder NMR crystallography of thymol," *Journal of Chemical Physics*, vol. 11, pp. 2610–2621, 2009.

[85] R. K. Harris, "NMR crystallography: The use of chemical shifts," *Solid State Sciences*, vol. 6, no. 10, pp. 1025–1037, 2004.

[86] R. A. Atkinson and V. Saudek, "Direct fitting of structure and chemical shift to NMR spectra," *Journal of the Chemical Society—Faraday Transactions*, vol. 93, no. 18, pp. 3319–3323, 1997.

[87] D. H. Brouwer, "A structure refinement strategy for NMR crystallography: An improved crystal structure of silica-ZSM-12 zeolite from 29Si chemical shift tensors," *Journal of Magnetic Resonance*, vol. 194, no. 1, pp. 136–146, 2008.

[88] P. P. Borbat, H. S. Mchaourab, and J. H. Freed, "Protein structure determination using long-distance constraints from double-quantum coherence ESR: Study of T4 lysozyme," *Journal of the American Chemical Society*, vol. 124, no. 19, pp. 5304–5314, 2002.

[89] L. Hermosilla, C. Sieiro, P. Calle, M. Zerbetto, and A. Polimeno, "Modeling of cw-EPR spectra of propagating radicals in methacrylic polymerization at different temperatures," *Journal of Physical Chemistry B*, vol. 112, no. 36, pp. 11202–11208, 2008.

[90] A. Soleilhavoup, M. R. Hampson, S. J. Clark, J. S. O. Evans, and P. Hodgkinson, "Using 17O solid-state NMR and first principles calculation to characterise structure and dynamics in inorganic framework materials," *Magnetic Resonance in Chemistry*, vol. 45, no. SUPPL., pp. S144–S155, 2007.

[91] S. J. Clark, M. D. Segall, C. J. Pickard, P. J. Hasnip, M. I. J. Probert, K. Refson, and M. C. Payne, "First principles methods using CASTEP," *Zeitschrift für Kristallographie*, vol. 220, no. 5-6, pp. 567–570, 2005.

[92] F. Neese, "Configuration interaction calculation of electronic g-tensors in transition metal complexes," *International Journal of Quantum Chemistry*, vol. 83, no. 3-4 SPEC. ISS., pp. 104–114, 2001.

[93] F. Neese, "Quantum chemical calculations of spectroscopic properties of metalloproteins and model compounds: EPR and Mössbauer properties," *Current Opinion in Chemical Biology*, vol. 7, no. 1, pp. 125–135, 2003.

[94] F. Neese, "Efficient and accurate approximations to the molecular spin-orbit coupling operator and their use in molecular g-tensor calculations," *Journal of Chemical Physics*, vol. 122, no. 3, 2005.

[95] R. Improta and V. Barone, "Interplay of electronic, environmental, and vibrational effects in determining the hyperfine coupling constants of organic free radicals," *Chemical Reviews*, vol. 104, no. 3, pp. 1231–1253, 2004.

[96] N. Rega, M. Cossi, and V. Barone, "Development and validation of reliable quantum mechanical approaches for the study of free radicals in solution," *Journal of Chemical Physics*, vol. 105, no. 24, pp. 11060–11067, 1996.

[97] V. Barone and P. Cimino, "Validation of the B3LYP/N07D and PBE0/N07D computational models for the calculation of electronic g-tensors," *Journal of Chemical Theory and Computation*, vol. 5, no. 1, pp. 192–199, 2009.

[98] R. Fletcher, *Practical Methods of Optimization*. John Wiley and Sons, 2000.

[99] H. Husein Mor, H. Weihe, and J. Bendix, "Fitting of EPR spectra: The importance of a flexible bandwidth," *Journal of Magnetic Resonance*, vol. 207, no. 2, pp. 283–286, 2010.

[100] A. Kirsch, *An Introduction to the Mathematical Theory of Inverse Problems*. Springer, 1996.

[101] Z. Chen and S. Haykin, "On different facets of regularization theory," *Neural computation*, vol. 14, no. 12, pp. 2791–2846, 2002.

[102] I. Kuprov and C. T. Rodgers, "Derivatives of spin dynamics simulations," *Journal of Chemical Physics*, vol. 131, no. 23, p. 234108, 2009.

[103] W. Cheney and D. Kincaid, *Numerical Mathematics and Computing, Fifth Edition*. Thomson Brooks/Cole, 2004.

[104] J. N. Lyness and J. J. Kaganove, "Comments on the nature of automatic quadrature routines.," *ACM Transactions on Mathematical Software*, vol. 2, no. 1, pp. 65–81, 1976.

[105] J. . Dumez and C. J. Pickard, "Calculation of NMR chemical shifts in organic solids: Accounting for motional effects," *Journal of Chemical Physics*, vol. 130, no. 10, 2009.

[106] K. Ruud, P. O. Åstrand, and P. R. Taylor, "Zero-point vibrational effects on proton shieldings: Functional-group contributions from *ab initio* calculations," *Journal of the American Chemical Society*, vol. 123, no. 20, pp. 4826–4833, 2001.

[107] P. Pulay and G. Fogarasi, "Geometry optimization in redundant internal coordinates," *The Journal of Chemical Physics*, vol. 96, no. 4, pp. 2856–2860, 1992.

[108] H. B. Schlegel, "Analytical second derivatives of two electron integrals over s and p cartesian gaussians," *The Journal of Chemical Physics*, vol. 90, no. 10, pp. 5630–5634, 1989.

[109] C. W. Hoganson and G. T. Babcock, "Protein-tyrosyl radical interactions in photosystem II studied by electron spin resonance and electron nuclear double resonance spectroscopy: Comparison with ribonucleotide reductase and in vitro tyrosine," *Biochemistry*, vol. 31, no. 47, pp. 11874–11880, 1992.

[110] C. Tommos, X. . Tang, K. Warncke, C. W. Hoganson, S. Styring, J. McCracken, B. A. Diner, and G. T. Babcock, "Spin-density distribution, conformation, and hydrogen bonding of the redox-active tyrosine YZ in photosystem II from multiple electron magnetic-resonance spectroscopies: Implications for photosynthetic oxygen evolution," *Journal of the American Chemical Society*, vol. 117, no. 41, pp. 10325–10335, 1995.

[111] P. Allard, A. L. Barra, K. K. Andersson, P. P. Schmidt, M. Atta, and A. Gräslund, "Characterization of a new tyrosyl free radical in salmonella typhimurium ribonucleotide reductase with EPR at 9.45 and 245 GHz," *Journal of the American Chemical Society*, vol. 118, no. 4, pp. 895–896, 1996.

[112] P. Reichard and A. Ehrenberg, "Ribonucleotide reductase. A radical enzyme," *Science*, vol. 221, no. 4610, pp. 514–519, 1983.

[113] M. Sahlin, A. Graslund, A. Ehrenberg, and B. M. Sjoberg, "Structure of the tyrosyl radical in bacteriophage T4-induced ribonucleotide reductase," *Journal of Biological Chemistry*, vol. 257, no. 1, pp. 366–369, 1982.

[114] M. J. Davies and A. Puppo, "Direct detection of a globin-derived radical in leghaemoglobin treated with peroxides," *Biochemical Journal*, vol. 281, no. 1, pp. 197–201, 1992.

[115] D. A. Svistunenko, J. Dunne, M. Fryer, P. Nicholls, B. J. Reeder, M. T. Wilson, M. G. Bigotti, F. Cutruzzolà, and C. E. Cooper, "Comparative study of tyrosine radicals in hemoglobin and myoglobins treated with hydrogen peroxide," *Biophysical journal*, vol. 83, no. 5, pp. 2845–2855, 2002.

[116] M. R. Gunther, B. E. Sturgeon, and R. P. Mason, "A long-lived tyrosyl radical from the reaction between horse metmyoglobin and hydrogen peroxide," *Free Radical Biology and Medicine*, vol. 28, no. 5, pp. 709–719, 2000.

[117] M. R. Gunther, R. A. Tschirret-Guth, H. E. Witkowska, Y. C. Fann, D. P. Barr, P. R. Ortiz De Montellano, and R. P. Mason, "Site-specific spin trapping of tyrosine radicals in the oxidation of metmyoglobin by hydrogen peroxide," *Biochemical Journal*, vol. 330, no. 3, pp. 1293–1299, 1998.

[118] D. A. Svistunenko and C. E. Cooper, "A new method of identifying the site of tyrosyl radicals in proteins," *Biophysical Journal*, vol. 87, no. 1, pp. 582–595, 2004.

[119] D. A. Svistunenko and G. A. Jones, "Tyrosyl radicals in proteins: A comparison of empirical and density functional calculated EPR parameters," *Physical Chemistry Chemical Physics*, vol. 11, no. 31, pp. 6600–6613, 2009.

[120] P. Nordlund and H. Eklund, "Structure and function of the Escherichia coli ribonucleotide reductase protein R2," *Journal of Molecular Biology*, vol. 232, no. 1, pp. 123–164, 1993.

[121] M. Eriksson, A. Jordan, and H. Eklund, "Structure of Salmonella typhimurium nrdF ribonucleotide reductase in its oxidized and reduced forms," *Biochemistry*, vol. 37, no. 38, pp. 13359–13369, 1998.

[122] M. Edén and M. H. Levitt, "Computation of orientational averages in solid-state nmr by gaussian spherical quadrature," *Journal of Magnetic Resonance*, vol. 132, no. 2, pp. 220–239, 1998.

[123] L. N. Hand and J. D. Finch, *Analytical Mechanics*. Cambridge University Press, 1998.

[124] L. Banci, I. Bertini, L. D. Eltis, I. C. Felli, D. H. W. Kastrau, C. Luchinat, M. Piccioli, R. Pierattelli, and M. Smith, "The three-dimensional structure in solution of the paramagnetic high-potential iron-sulfur protein I from ectothiorhodospira halophila through nuclear magnetic resonance," *European Journal of Biochemistry*, vol. 225, no. 2, pp. 715–725, 1994.

[125] S. Aono, I. Bertini, J. Cowan, C. Luchinat, A. Rosato, and M. Viezzoli, "1H NMR studies of the Fe7S8 ferredoxin from Bacillus schlegelii: A further attempt to understand Fe3S4 clusters," *Journal of Biological Inorganic Chemistry*, vol. 1, no. 6, pp. 523–528, 1996.

[126] L. Banci, I. Bertini, H. B. Gray, C. Luchinat, T. Reddig, A. Rosato, and P. Turano, "Solution structure of oxidized horse heart cytochrome c," *Biochemistry*, vol. 36, no. 32, pp. 9867–9877, 1997.

[127] I. Bertini, A. Donaire, C. Luchinat, and A. Rosato, "Paramagnetic relaxation as a tool for solution structure determination: Clostridium pasteurianum ferredoxin as an example," *Proteins: Structure, Function and Genetics*, vol. 29, no. 3, pp. 348–358, 1997.

[128] I. Bertini, A. Donaire, B. Jiménez, C. Luchinat, G. Parigi, M. Piccioli, and L. Poggi, "Paramagnetism-based versus classical constraints: An analysis of the solution structure of Ca Ln calbindin D9k," *Journal of Biomolecular NMR*, vol. 21, no. 2, pp. 85–98, 2001.

[129] G. Pintacuda, A. Moshref, A. Leonchiks, A. Sharipo, and G. Otting, "Site-specific labelling with a metal chelator for protein-structure refinement," *Journal of Biomolecular NMR*, vol. 29, pp. 351–361, 2004.

[130] Multiple Authors, *Multifrequency Electron Paramagnetic Resonance*. Wiley-VCH, 2011.

[131] E. M. Stein and R. Shakarchi, *Real Analysis, Measure Theory, Integration, and Hilbert Spaces*. Princeton University Press, 2005.

[132] S. K. Berberian, *Fundamentals of Real Analysis*. Springer, 1999.

[133] I. M. Gel'fand and G. E. Shilov, *Generalized Functions. Volume I: Properties and Operations*. Academic Press, 1964.

[134] F. James, "Monte-Carlo theory and practice," *Reports on Progress in Physics*, vol. 43, no. 9, p. 1145, 1980.

[135] H. Engels, *Numerical Quadrature and Cubature*. Academic Press, 1980.

[136] M. A. Malcolm and R. B. Simpson, "Local versus global strategies for adaptive quadrature," *Transactions on Mathematical Software*, vol. 1, pp. 129–146, June 1975.

[137] T. Hahn, "Cuba—a library for multidimensional numerical integration," *Computer Physics Communications*, vol. 168, no. 2, pp. 78–95, 2005.

[138] T. Hahn, "Cuba—a library for multidimensional numerical integration," *Computer Physics Communications*, vol. 176, no. 11-12, pp. 712–713, 2007.

[139] J. Berntsen, T. O. Espelid, and A. Genz, "An adaptive algorithm for the approximate calculation of multiple integrals," *ACM Transactions on Mathematical Software*, vol. 17, pp. 437–451, Dec. 1991.

[140] P. van Dooren and L. de Ridder, "An adaptive algorithm for numerical integration over an n-dimensional cube," *Journal of Computational and Applied Mathematics*, vol. 2, no. 3, pp. 207–217, 1976.

[141] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, and F. Rossi, *GNU Scientific Library Reference Manual*. Network Theory Ltd., third ed., 2009.

[142] C. Schmitz, M. J. Stanton-Cook, X. C. Su, G. Otting, and T. Huber, "Numbat: An interactive software tool for fitting $\delta\chi$-tensors to molecular coordinates using pseudocontact shifts," *Journal of Biomolecular NMR*, vol. 41, no. 3, pp. 179–189, 2008.

[143] C. Schmitz, M. John, A. Y. Park, N. E. Dixon, G. Otting, G. Pintacuda, and T. Huber, "efficient χ-tensor determination and NH assignment of paramagnetic proteins," *Journal of Biomolecular NMR*, vol. 35, no. 2, pp. 79–87, 2006.

[144] M. A. Keniry, H. A. Berthon, J. Y. Yang, C. S. Miles, and N. E. Dixon, "NMR solution structure of the θ subunit of DNA polymerase III from Escherichia coli," *Protein Science*, vol. 9, no. 4, pp. 721–733, 2000.

[145] M. John, A. Park, N. Dixon, and G. Otting, "NMR detection of protein 15N spins near paramagnetic lanthanide ions," *Journal of the American Chemical Society*, vol. 129, no. 3, pp. 462–463, 2007.

[146] S. Hamdan, P. D. Carr, S. E. Brown, D. L. Ollis, and N. E. Dixon, "Structural basis for proofreading during replication of the Escherichia coli chromosome," *Structure*, vol. 10, no. 4, pp. 535–546, 2002.

[147] M. John, C. Schmitz, Y. P. Ah, N. E. Dixon, T. Huber, and G. Otting, "Sequence-specific and stereospecific assignment of methyl groups using paramagnetic lanthanides," *Journal of the American Chemical Society*, vol. 129, no. 44, pp. 13749–13757, 2007.

[148] C. D. Schwieters, J. J. Kuszewski, and G. Marius Clore, "Using xplor-nih for nmr molecular structure determination," *Progress in Nuclear Magnetic Resonance Spectroscopy*, vol. 48, no. 1, pp. 47–62, 2006.

[149] Multiple Authors, *Handbook of Chemistry and Physics*. CRC Press, Taylor and Francis Group, 2011.

[150] K. J. Honlhoff, P. Robustelli, A. Cavalli, X. Salvatella, and M. Vendruscolo, "Fast and accurate predictions of protein NMR chemical shifts from interatomic distances," *Journal of the American Chemical Society*, vol. 131, pp. 13894–13895, 2009.

[151] H. Zhang, S. Neal, and D. S. Wishart, "RefDB: A database of uniformly referenced protein chemical shifts," *Journal of Biomolecular NMR*, vol. 25, no. 3, pp. 173–195, 2003.

[152] L. Landweber, "An iteration formula for Fredholm integral equations of the first kind," *American Journal of Mathematics*, vol. 73, no. 3, pp. 615–624, 1951.

[153] H. W. Engl and W. Grever, "Using the l-curve for determining optimal regularization parameters," *Numerische Mathematik*, vol. 69, pp. 25–31, 1994.

[154] C. Hansen, "Analysis of discrete ill-posed problems by means of the l-curve," *SIAM Review*, vol. 34, pp. 561–580, 1992.

[155] C. C. Pinter, *A Book of Abstract Algebra.* Dover, 1990.

[156] N. Carter, *Visual Group Theory.* Mathematical Association of America, 2009.

[157] J. Hadamard, *Lectures on the Cauchy Problem in Linear Partial Differential Equations.* Yale University Press, New Haven, 1923.

[158] C. G. Broyden, "The convergence of a class of double-rank minimization algorithms 1. general considerations," *Journal of the Institute of Mathematical Applications*, vol. 6, pp. 76–90, 1969.

[159] R. Fletcher, "A new approach to variable metric algorithms," *The Computer Journal*, vol. 13, pp. 317–322, 1970.

[160] D. Goldfarb, "A family of variable-metric methods derived by variational means," *Journal of the Institute of Mathematical Applications*, vol. 24, pp. 23–26, 1970.

[161] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Mathematics of Computation*, vol. 24, pp. 647–656, 1970.

[162] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[163] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder-Mead simplex method in low dimensions," *Siam Journal of Optimization*, vol. 9, pp. 112–147, 1998.