

Estimation of NMR Signals in the Time Domain: Methodology, Applications and Software

SIMON G. HULSE

Balliol College
University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity Term, 2023

Supervisors:

Mohammadali Foroozandeh

Timothy Claridge

Fay Probert

ACKNOWLEDGEMENTS

To begin, I would like to thank Dr Mohammadali Foroozandeh, by primary supervisor for the majority of my PhD. It was truly an honour to be able to discuss my work with him; his deep insights into a plethora of topics in magnetic resonance is truly inspiring if a little daunting to mortals like myself. His passion for literature and music also rubbed off on me. While they frequently had a rather violent bent — Cormac McCarthy's *Blood Meridian* and an opera by Alfred Schnittke on Faust come to mind — I cannot say that his recommendations were ever dull.

Prof. Timothy Claridge acted as my co-supervisor for most of my PhD. The regular meetings on NMR that he hosted in the Chemistry Department were very valuable to me, enabling me both to learn about how people were applying NMR in their research, but also as it gave me countless opportunities to present my work and other topics of interest.

Both Dr Foroozandeh and Prof. Claridge left the department during my PhD. Dr Foroozandeh now works in quantum computing at ZURICH INSTRUMENTS, while Prof. Claridge is the director of a new NMR facility at EXSCIENTIA. I wish them both the best with their new endeavours.

I am very grateful to Dr Fay Probert who agreed to be my supervisor for the latter stages of my PhD, providing valuable advice and assistance. I'd like to thank her and the rest of her group for welcoming me during my last year.

Finally, I'd like to thank the ROYAL SOCIETY for generously funding me during my studies.

ABSTRACT

Nuclear magnetic resonance spectroscopy (NMR) is an analytical technique employed in many scientific disciplines which is able to provide insights into the structures and dynamics of chemical species. To maximise the utility of nuclear magnetic resonance spectroscopy (NMR) experiments, appropriate data treatment and analysis is necessary. The conventional route to extracting quantitative information from the raw experimental data, the *free induction decay* (FID), is to convert it to an NMR spectrum, through application of the Fourier transform (FT). Such spectra provide a human-interpretable representation of data, with trained practitioners able to rationalise their appearance by mapping peaks in the spectrum to chemical environments of species in the sample considered. Despite the simplicity of the representation provided, the FT suffers from poor resolution, often leading to peaks with similar frequencies overlapping. Disentangling the quantitative information associated with such peaks is not feasible using typical methods such as integration. As an alternative, parametric estimation techniques aim to provide detailed information about each signal present in the data. These have been shown to perform effectively even in scenarios where significant signal overlap exists.

This thesis focusses on the development of a parametric estimation method for the analysis of free induction decays (FIDs) derived from solution-state NMR experiments. The guiding principle behind the method is that it should require as little user input as possible, while simultaneously providing realistic predictions of the component signals which contribute to the data. Beyond simply providing a breakdown of individual signal components, many useful applications may be realised when estimation techniques are employed. The initial motivation for this thesis was to develop a procedure for the generation of pure shift NMR spectra with desirable properties from 2D J-resolved datasets. Other applications presented here are for the analysis of 2D datasets in which each FID exhibits a variation in its amplitude — including inversion recovery (T_1), CPMG (T_2) and diffusion experiments —, and a means of producing phased, ultra-broadband NMR spectra from an experiment comprising a single frequency-swept excitation pulse.

The methods presented in this thesis are incorporated into a software package written in the **PYTHON** programming language, called *NMR estimation in Python* (NMR-EsPy).

ACRONYMS

1D one-dimensional

2D two-dimensional

3D three-dimensional

2DJ 2D J-resolved

AIC Akaike information criterion

ALPESTRE a linear predictive estimation of signal time reversal

AMARES advanced method for accurate, robust, and efficient spectral fitting

API application programming interface

AR autoregressive

ARMA autoregressive moving average

AWGN additive white gaussian noise

BFGS Broyden–Fletcher–Goldfarb–Shanno

BIRD bilinear rotation decoupling

CHORUS chirped, ordered pulses for ultra-broadband spectroscopy

COSY correlation spectroscopy

CPMG Carl-Purcell-Meiboom-Gill

CPU central processing unit

CRAFT complete reduction to amplitude frequency table

CUPID computer-assisted undiminished-sensitivity protocol for ideal decoupling

DFT discrete Fourier transform

DMSO Dimethyl sulfoxide, $(\text{H}_3\text{C})_2\text{SO}$

DMSO-d₆ Deuterated DMSO

DOSY diffusion-ordered spectroscopy

EYM Eckart-Young-Mirsky

FID free induction decay

FFT fast Fourier transform

FS frequency-swept

FT Fourier transform

GN Gauss-Newton

GUI graphical user interface

HSVD Hankel singular value decomposition

IFT inverse Fourier transform

INEPT insensitive nuclei enhancement by polarization transfer

ITMPM information theoretic matrix pencil method

LED longitudinal eddy current delay

LM Levenberg-Marquardt

LP linear prediction

LPSVD linear prediction singular value decomposition

MDL minimum description length

MEMPM matrix enhancement and matrix pencil method

MLE maximum likelihood estimate

MMEMPM modified matrix enhancement and matrix pencil method

MPM matrix pencil method

MRS *in vivo* magnetic resonance spectroscopy

NLP non-linear programming

NMR nuclear magnetic resonance spectroscopy

NMR-EsPy NMR estimation in Python

NOESY nuclear overhauser effect spectroscopy

pdf probability density function

PFG pulsed field gradient

PGSE pulsed gradient spin echo

PGSTE pulsed gradient stimulated echo

PGSTEBP pulsed gradient stimulated echo with bipolar gradients

PSYCHe pure shift yeilded by chirp excitation

RAM random access memory

RF radio-frequency

SNR signal-to-noise ratio

ST Steihaug-Toint

SVD singular value decomposition

TSE-PSYCHe triple spin echo PSYCHe

TROSY transverse relaxation optimised spectroscopy

VE virtual echo

VARPRO variable projection

WURST wideband, uniform rate, smooth truncation

ZS Zanger-Sterk

NOMENCLATURE

General Mathematics

$\mathcal{A} := \mathcal{B}$	\mathcal{A} is defined to be equal to \mathcal{B}
$\lfloor a \rfloor$	The nearest integer to a , such that $\lfloor a \rfloor \leq a$
$\lceil a \rceil$	The nearest integer to a , such that $\lceil a \rceil \geq a$
$\lfloor a \rfloor$	The nearest integer to a
$a \bmod b$	a modulo b , given by $a \bmod b = a - \left\lfloor \frac{a}{b} \right\rfloor$

Complex Numbers

i	The imaginary unit, $i := \sqrt{-1}$
$\Re(z)$	The real component of z
$\Im(z)$	The imaginary component of z
$ z $	The absolute value of z , given by $ z = \sqrt{\Re(z)^2 + \Im(z)^2}$

Number sets

\mathbb{C}	The set of complex numbers
\mathbb{N}	The set of natural numbers with zero excluded
\mathbb{N}_0	The set of natural numbers with zero included
\mathbb{R}	The set of real numbers
$\mathbb{R}_{>0}$	The set of positive real numbers
\mathbb{Z}	The set of integers

Probability

$x \sim \mathcal{X}$	x behaves according to distribution \mathcal{X}
$A \perp\!\!\!\perp B$	A and B are conditionally independent
$\mathcal{N}(\mu, \sigma^2)$	Normal (Gaussian) distribution with mean μ and variance σ^2 :
	$\mathcal{N}(x \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
$\mathcal{N}_C(\mu, \sigma^2)$	Complex normal distribution with mean μ and variance $\sigma^2/2$
$\mathcal{U}(l, r)$	Uniform distribution with bounds l and r :

$$\mathcal{U}(x | l, r) = \begin{cases} \frac{1}{r-l} & l \leq x \leq r \\ 0 & \text{otherwise} \end{cases}$$

Vectors, Matrices, and Arrays

.T	Transpose
.†	Conjugate transpose
.-1	Inverse
.+	Moore-Penrose pseudo-inverse.
. $\circlearrowright(d)$	Right circular rotation by along axis d of the array by one element. For example, for a 2D matrix $\mathbf{X} \in \mathbb{F}^{M \times N}$, $\mathbf{X}^{\circlearrowright(1)}$ is given by

$$\begin{bmatrix} x_{M,1} & x_{M,2} & \cdots & x_{M,N} \\ x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M-1,1} & x_{M-1,2} & \cdots & x_{M-1,N} \end{bmatrix},$$

while $\mathbf{X}^{\circlearrowright(2)}$ is

$$\begin{bmatrix} x_{1,N} & x_{1,1} & \cdots & x_{1,N-1} \\ x_{2,N} & x_{2,1} & \cdots & x_{2,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M,N} & x_{M,1} & \cdots & x_{M,N-1} \end{bmatrix}.$$

$\cdot^{\leftrightarrow(d)}$ Reversal of elements along axis d of an array. For example, for a 2D matrix $\mathbf{X} \in \mathbb{F}^{M \times N}$, $\mathbf{X}^{\leftrightarrow(1)}$ is given by

$$\begin{bmatrix} x_{M,1} & x_{M,2} & \cdots & x_{M,N} \\ x_{M-1,1} & x_{M-1,2} & \cdots & x_{M-1,N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N} \\ x_{1,1} & x_{1,2} & \cdots & x_{1,N} \end{bmatrix},$$

$\text{diag}(\cdot)$ Produces a diagonal matrix from a vector. Given a vector $\mathbf{x} \in \mathbb{F}^N$, $\text{diag}(\mathbf{x})$ is a matrix $\in \mathbb{F}^{N \times N}$ of the form

$$\begin{bmatrix} x_1 & 0 & \cdots & 0 \\ 0 & x_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_N \end{bmatrix}$$

$\langle \cdot, \cdot \rangle$ Inner product of two arrays with identical shapes. Given \mathbf{X} and \mathbf{Y} , two D -dimensional arrays $\in \mathbb{C}^{N_1 \times \cdots \times N_D}$:

$$\langle \mathbf{X}, \mathbf{Y} \rangle = \sum_{n_1=1}^{N_1} \cdots \sum_{n_D=1}^{N_D} x_{n_1, \dots, n_D}^* y_{n_1, \dots, n_D}$$

$\|\cdot\|$ Norm, equivalent to $\sqrt{\langle \cdot, \cdot \rangle}$.

$\mathbf{x} \otimes \mathbf{y}$ Outer product of $\mathbf{x} \in \mathbb{F}^M$ and $\mathbf{y} \in \mathbb{F}^N$, which generates a matrix $\mathbf{A} \in \mathbb{F}^{M \times N}$, such that

$$a_{m,n} = x_m y_n$$

$\mathbf{x} \times \mathbf{y}$ Cross product of two vectors $\mathbf{x} \in \mathbb{R}^3$ and $\mathbf{y} \in \mathbb{R}^3$, given by

$$\mathbf{x} \times \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \sin(\theta) \mathbf{n},$$

where θ is the angle between \mathbf{x} and \mathbf{y} , in the plane containing them, and \mathbf{n} is the unit vector perpendicular to said plane. N.B. the times symbol is occasionally used to denote scalar multiplication as well.

Calculus

$\nabla f(\mathbf{x})$ Gradient vector of a differentiable, scalar value function $f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}$:

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \dots & \frac{\partial f}{\partial x_N} \end{bmatrix}^T$$

$\nabla^2 f(\mathbf{x})$ Hessian matrix of a twice-differentiable, scalar value function $f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}$:

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \frac{\partial^2 f}{\partial x_N \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix}$$

TABLE OF CONTENTS

1	PRELIMINARIES	1
1.1	Overview of this work	1
1.1.1	Conception and motivation	1
1.1.2	Thesis Overview	2
1.2	An Introduction to NMR	3
1.2.1	A Brief History	3
1.2.2	Nuclear Spin and Magnetism	4
1.2.3	The NMR Spectrometer	10
1.2.4	The structure of the FID	12
1.3	NMR Data Processing and Analysis	15
1.3.1	Conventional NMR Processing	15
1.3.2	Conventional processing for multidimensional datasets	18
1.3.3	Conventional NMR analysis	21
1.3.4	Estimation Techniques for NMR Analysis	21
2	THEORY	27
2.1	Outline of the Problem	27
2.2	Generating an Initial Guess: Matrix Pencil Method	29
2.2.1	1D Matrix Pencil Method	29
2.2.2	2D Matrix Enhancement and Matrix Pencil Method	32
2.2.3	Model Order Selection	36
2.3	Non-linear programming for NMR estimation	37
2.3.1	An overview of NLP	39
2.3.2	Non-linear programming applied to FID estimation	42
2.3.3	Approximating the Hessian	44
2.3.4	Estimation Errors	45
2.3.5	Visualisation of a simple example	46
2.3.6	Phase Variance Minimisation	47

2.4 Profiling the MPM and NLP	50
2.4.1 1D MPM	50
2.4.2 MMEMPM	52
2.4.3 1D Hessian	52
2.4.4 2D Hessian	52
2.5 Frequency Filtration	52
2.5.1 The virtual echo	52
2.5.2 The filtering process	54
2.6 Summary	58
3 1D RESULTS AND APPLICATIONS	59
3.1 Conventional 1D datasets	59
3.1.1 “Twenty signals”	59
3.1.2 Andrographolide	62
3.1.3 Cyclosporin A	66
3.2 Amplitude-attenuated Datasets	67
3.2.1 Relaxation experiments	67
3.2.2 Diffusion experiments	68
3.2.3 Methodology	73
3.2.4 Results	77
3.3 Phased broadband spectra from chirp excitation	81
3.3.1 Chirp excitation	82
3.3.2 Methodology	84
3.3.3 Results	86
3.4 Summary	88
4 PURE SHIFT SPECTRA FROM 2D J-RESOLVED ESTIMATION	89
4.1 Pure Shift NMR	90
4.1.1 The 2D J-resolved Experiment	90
4.1.2 The Zanger-Sterk Method	92
4.1.3 The BIRD Method	93
4.1.4 PSYCHE	93
4.1.5 Pure shift spectra from 2DJ estimation	94

4.2 Methodology	94
4.2.1 The estimation routine	94
4.2.2 The -45° signal	97
4.2.3 Filtration of 2DJ data	97
4.2.4 Multiplet Prediction	98
4.3 Results	99
4.3.1 “Four Multiplets”	99
4.3.2 Strychnine simulated	101
4.3.3 Quinine	102
4.3.4 Camphor	104
4.3.5 Dexamethasone	105
4.3.6 Estradiol	107
4.4 Summary	108
5 SOFTWARE	111
5.1 A description of NMR-EsPy	111
5.1.1 Why PYTHON?	111
5.1.2 Estimator objects	112
5.2 The NMR-EsPy GUI	113
5.3 Summary	117
6 CONCLUSIONS AND FUTURE WORK	119
6.1 Conclusions	119
6.2 Future Work	119
A NMR GLOSSARY	133
B ADDITIONAL THEORY	135
B.1 Mathematical definitions	135
B.1.1 Linear algebra	135
B.1.2 Statistics and probability	136
B.2 Multidimensional virtual echos	136
B.3 Additional algorithms	138

C CODE LISTINGS	143
C.1 Matrix pencil methods	143
C.1.1 MDL	143
C.1.2 MPM	144
C.1.3 MMEMPM	145
C.2 NLP	149
C.2.1 Trust Region Algorithm	149
C.2.2 Computing \mathcal{F}_ϕ , $\nabla \mathcal{F}_\phi$, and $\nabla^2 \mathcal{F}_\phi$	152
C.2.3 The main routine	155
C.3 CUPID	157
C.3.1 Assigning multiplet structures	157
D INFORMATION ON DATASETS AND RESULTS	159
D.1 Simulated datasets	159
D.1.1 SPINACH Simulations	159
D.1.2 2DJ	162
D.1.3 Inversion recovery	164
D.2 Experimental datasets	165
D.2.1 Structures	165
D.2.2 1D datasets	165
D.2.3 Diffusion datasets	166
D.2.4 2DJ datasets	168
D.2.5 PSYCHE datasets	169
D.3 CUPID result metrics	171
E NMR-EsPy WALKTHROUGHS	177
E.1 1D Walkthrough	177
E.2 2DJ Walkthrough	185

LIST OF FIGURES

1.1	The variation of energy of the spin states of ^1H , ^7Li , and ^{17}O with external magnetic field strength.	6
1.2	An illustration of the free evolution of the bulk magnetisation of an ensemble of spin- $1/2$ nuclei according to the Bloch model.	9
1.3	An illustration of the influence of the four parameters associated with a signal in both the time-domain nels and Fourier-domain.	13
1.4	Spectra acquired from amplitude- and phase-modulated 2D signals.	20
2.1	A visualisation of the behaviour of the MDL for three different FIDs comprising the same deterministic component, but with different noise variances.	38
2.2	A visualisation of the trajectory of a 2-parameter optimisation involving a simulated FID comprising a single signal.	46
2.3	Profiles outlining the computation times and peak memory consumption of the MPM and for Hessian construction.	51
2.4	An illustration of the filtering procedure applied to a 1D FID.	55
3.1	The result of estimating a series of 5 simulated signals comprising 20 oscillators, using solely the MPM and also with phase variance-regularised NLP afterwards.	60
3.2	Result of applying the estimation routine to selected regions of a pulse-acquire dataset of andrographolide.	63
3.3	Result of applying the estimation routine to selected regions of a pulse-acquire dataset of cyclosporin A.	66
3.4	Pulse sequences used for the determination of translational diffusion constants.	70
3.5	Three examples of results generated on simulated inversion recovery datasets comprising five ddd multiplet structures.	78
3.6	Result of estimating a Oneshot DOSY dataset of andrographolide.	80
3.7	Result of estimating a diffusion dataset for a mixture of L-threonine, L-valine and D-(+)-glucose.	81
3.8	An illustration of an experiment comprising a single chirp pulse.	83
3.9	Comparison of quadratic phase correction vs frequency-dependent back-propagation in treating simulated single-chirp excitation data.	85

3.10 Comparison of quadratic phase correction vs frequency-dependent back-propagation in treating experimental single-chirp excitation data generated from a sample of Gd-doped H ₂ O in D ₂ O.	87
4.1 Region of a 2DJ spectrum of strychnine.	91
4.2 An illustration of the reasoning behind the name “−45° signal” used to generate pure shift spectra.	96
4.3 An illustration of the filtering procedure for 2DJ data.	98
4.4 The result of applying CUPID to 5 instances of simulated 2DJ datasets with 4 heavily overlapping multiplet structures.	100
4.5 Application of CUPID on a simulated strychnine 2DJ dataset.	101
4.6 Application of CUPID on the non-aromatic regions of a quinine 2DJ dataset. .	103
4.7 Application of CUPID on a camphor dataset.	105
4.8 Application of CUPID on a dexamethasone dataset.	106
4.9 Application of CUPID on a 17β-estradiol dataset.	107
5.1 Inheritance tree for estimation classes in the NMR-EsPy package.	112
5.2 Screenshots of the NMR-EsPy GUI for 1D and 2DJ estimation.	114
D.1 The molecular structures of species giving rise to the experimental NMR datasets considered in this work.	165
D.2 Oneshot DOSY pulse sequence used for the acquisition of andrographolide data.	167
D.3 Pulse sequence used for the acquisition of glucose/threonine/valine diffusion data.	168
D.4 PSYCHE pulse sequence used for the acquisition of estradiol data.	170
D.5 TSE-PSYCHE pulse sequence used for the acquisition of dexamethasone data. .	170

LIST OF TABLES

1.1	A table of regularly encountered nuclei in NMR, along with common nuclei which are not NMR active.	5
2.1	The number of first and second derivatives that are necessary to compute both the gradient vector and Hessian matrix of the fidelity for 1- 2- and 3-dimensional datasets, as well as a general D -dimensional dataset.	44
3.1	Major coupling partners associated with spins in andrographolide whose signals are considered in Figure 3.2.	64
3.2	The various functional forms of \mathcal{A} according to the different amplitude-attenuating NMR experiments considered.	74
D.1	The isotropic chemical shifts, scalar couplings and relaxation times associated with spin systems used in SPINACH simulations.	160
D.2	Experiment parameters for 2DJ simulations run using SPINACH.	163
D.3	Parameters for the “five multiplets” inversion recovery simulation run using SPINACH.	164
D.4	Noteworthy experiment parameters for the pulse-acquire datasets used.	166
D.5	Noteworthy experiment parameters for the diffusion datasets acquired.	166
D.6	Noteworthy experiment parameters for the 2D J-Reolved and PSYCHE experiments run.	169
D.7	Metrics for all the results generated using CUPID.	172

LIST OF ALGORITHMS

2.1	The matrix pencil method, with the optional prediction of model order using the minimum description length.	33
2.2	Nonlinear programming routine employed in NMR-EsPy.	41
2.3	Filtering procedure for 1D data.	57
3.1	Routine for estimating a sequence of 1D FIDs which exhibit variation in amplitudes across increments.	75
B.1	The MMEMPM.	139
B.2	Steihaug-Toint method for determining an update for nonlinear programming.	140
B.3	Filtering procedure for 2D data.	141
B.4	Filtering procedure for 2DJ data.	142

LIST OF CODE LISTINGS

C.1	Required imports for the subsequent PYTHON listings.	143
C.2	PYTHON implementation of the MDL for estimation of the model order of a 1D FID.	143
C.3	PYTHON implementation of the MPM for estimation of a 1D FID.	144
C.4	PYTHON implementation of the MMEMPM for estimation of a 2D hypercomplex FID.	145
C.5	PYTHON implementation of the Steihaug-Toint trust region algorithm.	149
C.6	PYTHON implementation for the generation of the fidelity for NLP applied to 1D estimation, as well as the gradient and (approximated) Hessian	152
C.7	PYTHON implementation for running the NLP routine for FID estimation. . . .	155
C.8	PYTHON implementation for performing multiplet assignment as part of CUPID.157	
D.1	Function for simulating 2DJ experiments using SPINACH	163

The primary theme of this thesis regards extracting quantitative information from nuclear magnetic resonance spectroscopy (NMR) data. This chapter lays the groundwork to contextualise it. After a brief introduction to NMR, including key historical developments and a simple description of its theoretical underpinnings, focus turns to the form that NMR data takes. The means by which the data is conventionally processed to generate NMR spectra, and subsequently analysed to gain chemical insights, is discussed. Parametric estimation estimation, while able to afford richer information about the data than conventional approaches, are not widely employed in NMR, despite many effective methods existing. A review of some of the most prominent ones are given. Finally, the motivation behind this work is given, along with an overview of the subsequent chapters.

1.1 Overview of this work

1.1.1 Conception and motivation

The initial motivation for this work emerged from discussions within the NMR Methodology group in Manchester involving my principal supervisor Dr. Mohammadali Foroozandeh and coworkers — notably Prof. Gareth Morris and Prof. Mathias Nilsson — while Dr. Foroozandeh was a Postdoctoral researcher there. There was an interest in generating pure shift NMR spectra from 2DJ data using an appropriate estimation and reconstruction algorithm. While little progress was made when Dr. Foroozandeh was based in Manchester, he wished to continue with it after moving to Oxford to take up a research fellowship. I took on a project focussed on this, though with a broader scope, when I joined his nascent research group as a PhD student.

1.1.2 Thesis Overview

This thesis is broken into four principal themes, which span the proceeding four chapters:

- **Chapter 2** discusses the theory behind routines which can be applied to determine parameter estimates which describe one-dimensional (1D) and two-dimensional (2D) NMR datasets.
- **Chapter 3** provides illustrations of the performance of the estimation routine on 1D NMR datasets. Furthermore, means in which the established estimation routine can be extended for two applications are explored:
 - Analysis of amplitude-attenuated datasets, such as those derived from diffusion and inversion recovery experiments (Section 3.2).
 - Overcoming quadratic phase behaviour and baseline distortions associated with ultra-broadband excitation by a single frequency-swept pulse (Section 3.3).
- **Chapter 4** outlines a method for generating pure shift spectra and deriving the associated multiplet structures from 2D J-resolved (2DJ) datasets. As described above, this was the initial motivation for embarking on this project.
- **Chapter 5** describes the source code developed as part of this project, called NMR-EsPy.
- Finally, conclusions and considerations for further potential developments are discussed in **Chapter 6**.

Further supporting information can be found in the Appendix, which comprises the following chapters:

- **Appendix A** provides a glossary of terms related to NMR which are not introduced in much detail in the main text.
- **Appendix B** provides additional information on the theory related to this work, including descriptions of mathematical concepts, and outlines of relevant algorithms.
- **Appendix C** provides code listings outlining how the methods described in this work can be implemented in `PYTHON`. These are effectively reduced versions of code found in the NMR estimation in Python (NMR-EsPy) package.
- **Appendix D** outlines how the simulated and experimental datasets considered in this work were generated.
- **Appendix E** is an insert from the documentation of NMR-EsPy, comprising tutorials on getting familiarised with using the package.

1.2 An Introduction to NMR

1.2.1 A Brief History

Since its conception almost 80 years ago, NMR has become a ubiquitous technique in chemistry, biochemistry and numerous other disciplines, thanks to the unique insights into structure and dynamics it can provide. The origins of the subject can be traced back to 1945, when independent work by Felix Bloch on water[1] and Edward Purcell on parrafin[2] gave rise to the first illustrations of nuclear magnetic resonances in condensed phases. The two hadn't met before their respective papers were published with about a month's separation[3]. Both received the Nobel Prize in Physics in 1952 for their pioneering work in the field. A notable mention should also be given to Yevgeny Zavoisky, the father of electron paramagnetic resonance, who probably observed NMR as far back as 1941[4]. Alas, he dismissed his results as irreproducible. In 1949 and 1950, work investigating NMR spectra from compounds containing Cu, ^{31}P , ^{14}N , and ^{19}F nuclei illustrated the concept of the chemical shift[5–7], in which nuclei in different chemical environments exhibit non-identical resonant frequencies. Chemists regarded these results with great interest, as these findings suggested that NMR could give insights into molecular structure.

Russel Varian secured the first patent for a commercial NMR machine, with a 30 MHz spectrometer following soon after. The first spectrometers functioned by slowly sweeping the magnetic field, causing spins to come into resonance at different times, in a process referred to as continuous wave spectroscopy. Richard Ernst and Weston Anderson, working at Varian Inc. at the time, proposed an alternative method: pulsed Fourier transform (FT) spectroscopy[8]. This was not seen as a fruitful endeavour by the company, largely because of the very long time it took to digitise the signal, and subsequently compute its FT[9]. Instead, the first commercial pulsed FT spectrometer was produced by Bruker Corp. in 1969, which revolutionised NMR. The emergence of the Cooley-Tukey's fast Fourier transform (FFT) algorithm[10] led to vast improvements in the speed with which experiments could be conducted, which incentivised the development of the new FT approach.

The idea of 2D NMR spectroscopy was proposed by Jean Jeener in 1971[11, 12], which Ernst and co-workers showcased a few years later in the form of a COSY experiment[13]. The use of multiple dimensions to spread out signals enabled vastly more complex structures to be studied. In 1985, a report of the first protein assigned by NMR (using COSY and NOESY experiments) was presented by Kurt Wütrich and co-workers[14]. Over time, extensive developments in techniques for biomolecular systems have occurred, including the creation of 3D and 4D “triple resonance” experiments[15, 16], as well TROSY experiments[17] for the study of large proteins.

NMR's significance as an analytical tool is evidenced by Nobel Prizes in Chemistry being awarded for work in the field on two separate occasions. First, Ernst received the prize in 1991

“for his contributions to the development of the methodology of high resolution nuclear magnetic resonance spectroscopy”[18]. In 2002, Wüthrich was recognised “for his development of nuclear magnetic resonance spectroscopy for determining the three-dimensional structure of biological macromolecules in solution”[19].

1.2.2 Nuclear Spin and Magnetism

Remark 1. *In most of this work, the convention of expressing vectors with lower-case bold letters and matrices/multidimensional arrays as upper-case bold letters is strictly adhered to. In this section however, with notations frequently encountered in the literature are used instead, with certain vector quantities denoted with upper-case letters.*

NMR relies on an intrinsic property of certain nuclei called *spin* which, along with orbital angular momentum, is one of the two sources of angular momentum in quantum mechanics. The angular momentum associated with a nuclear spin is characterised by the quantum number $I \in \{0, 1/2, 1, 3/2, \dots\}$. Spin- $1/2$ nuclei are the most commonly studied in NMR, as those with $I > 1/2$ often have very short-lived excited states, due to electric quadrupole effects. A rigorous description of NMR requires the application of quantum mechanics, with many excellent texts devoted to the subject[20–22]. However, a basic appreciation can be gained using the Bloch model, a semi-classical description of the simplest of possible system to study: a ensemble of isolated, identical spin- $1/2$ nuclei. Bloch model becomes inadequate for more complex spin systems, featuring multiple spins which interact through mechanisms such as J-couplings. However, it provides valuable insights into the form that a signal detected by an NMR experiment takes.

The nuclear spin angular momentum $\mathbf{I} \in \mathbb{R}^3$ is a vector with (square) magnitude:

$$\mathbf{I}^2 = \mathbf{I} \cdot \mathbf{I} = \hbar I(I + 1), \quad (1.1)$$

where $\hbar = h/2\pi$ is the reduced Planck constant. While it is not possible to specify multiple components of the angular momentum simultaneously due to the uncertainty principle, it is possible to specify one of these, along with \mathbf{I}^2 . Conventionally, this is chosen to be the z-component, for which

$$I_z = \hbar m, \quad (1.2)$$

with $m \in \{-I, -I + 1, \dots, I - 1, I\}$. Equations 1.1 and 1.2 imply that the magnitude of the z-component may only adopt certain discrete values (i.e. it is quantised). A nucleus with non-zero spin has an associated *magnetic moment*, given by:

$$\boldsymbol{\mu} = \gamma \mathbf{I} \implies \mu_z = \gamma I_z = \gamma \hbar m. \quad (1.3)$$

Nucleus	I	γ (rad T $^{-1}$ s $^{-1}$)	Relative Abundance (%)
^1H	1/2	2.6752×10^8	99.9885
^2H	1	4.1066×10^7	0.0115
^6Li	1	3.9371×10^7	7.59
^7Li	3/2	1.0398×10^8	92.41
^{12}C	0	-	98.93
^{13}C	1/2	6.7283×10^7	1.07
^{14}N	1	1.9338×10^7	99.636
^{15}N	1/2	-2.7126×10^7	0.364
^{16}O	0	-	99.756
^{17}O	5/2	-3.6281×10^7	0.038
^{19}F	1/2	2.5162×10^8	100
^{31}P	1/2	1.0839×10^8	100

TABLE 1.1: A table of regularly encountered nuclei in NMR, along with common nuclei which are not NMR active.

$\gamma \in \mathbb{R}$ is a proportionality constant called the *gyromagnetic ratio*, which is dependent on the nucleus of interest. Table 1.1 provides the gyromagnetic ratios for some low-mass nuclei commonly encountered in NMR, along with some which alas are spin-0.

Without the presence of an external magnetic field, the nuclear spin states of different m are degenerate. However, when subjected to such a field, the *Zeeman effect* is observed, in which the relative energies of the different states diverge. The energy of a given magnetic moment relative to its zero-field energy, is given by

$$E = -\boldsymbol{\mu} \cdot \mathbf{B}_0 \quad (1.4)$$

where $\mathbf{B}_0 \in \mathbb{R}^3$ is the magnetic field vector. In NMR it is conventional to define the external field as directed along the laboratory z -axis, such that $B_{0,x} = B_{0,y} = 0$ and $B_{0,z} = B_0$ where B_0 is the magnetic field strength. The energies of the individual spin states are therefore

$$E_m = -\gamma I_z B_0 = -m\hbar\gamma B_0 \quad (1.5)$$

Figure 1.1 illustrates how the relative energies of different spin states vary with B_0 .

NMR samples comprise a vast ensemble of equivalent spin systems, and it is the macroscopic properties of the sample that are observed. At thermal equilibrium, the various spin states will be disproportionately populated in accordance with the Boltzmann distribution, with lower energy states being more heavily populated. For example, an ensemble of non-interacting spin-1/2 nuclei

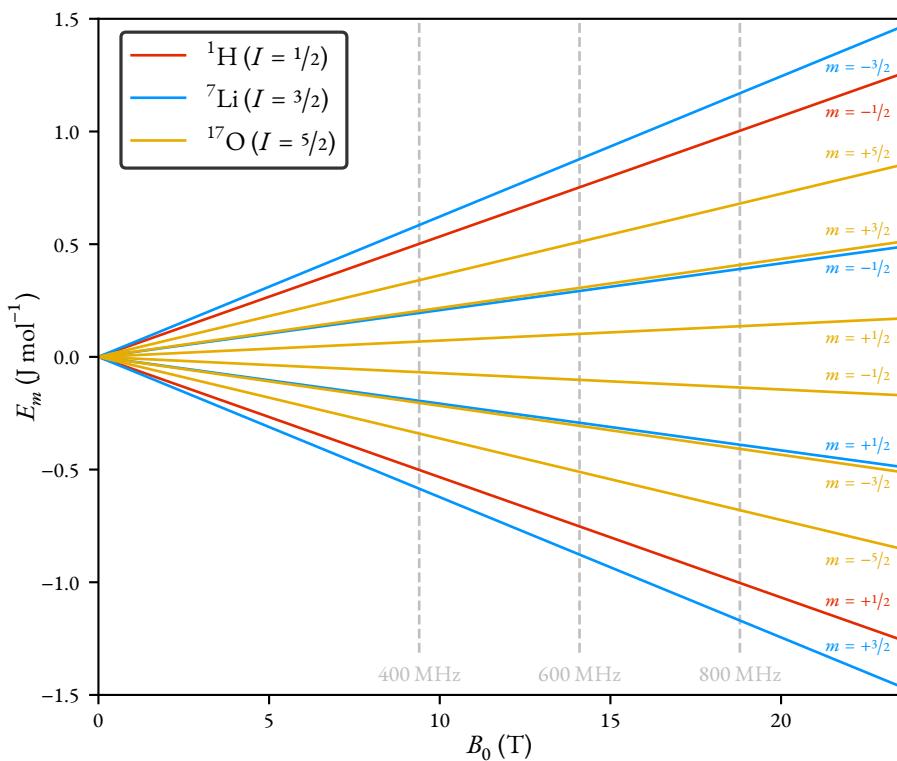


FIGURE I.I: The variation of energy of the spin states of ^1H , ^7Li , and ^{17}O with external magnetic field strength (B_0), up to 23.5 T, which is approximately the strength of a 1 GHz NMR magnet. Three common field strengths for commercial NMR magnets are indicated: 9.40 T (400 MHz), 14.10 T (600 MHz), and 18.79 T (800 MHz).

with $\gamma > 0$ will have a more populated $m = +1/2$ (α) state, relative to the $m = -1/2$ (β) state. Due to the very small relative energy difference however, it should be noted that the discrepancy in state populations is very small. The ensemble acquires a net (bulk) magnetic moment \mathbf{M} , given by the summation of all the individual spin moments:

$$\mathbf{M} = \sum_{s=1}^S \boldsymbol{\mu}_s, \quad (1.6)$$

where $S \gg 1$ is the number of spins in the ensemble. At equilibrium, the x - and y -components of the bulk magnetisation are zero, i.e.

$$\sum_{s=1}^S \mu_{x,s} = \sum_{s=1}^S \mu_{y,s} = 0, \quad (1.7)$$

such that the bulk magnetisation is collinear with the field direction, with a magnitude M_0 , which is approximately given by[22: Section 1.1].

$$M_0 \approx \frac{S\gamma^2\hbar^2B_0I(I+1)}{3k_B T}, \quad (1.8)$$

where k_B is the Boltzmann constant, and T is the sample temperature. For purposes of experiment sensitivity, it is hence desirable to utilise nuclei with high natural abundance (which affects S), and high gyromagnetic ratio. Along with other favourable attributes such as its ubiquity in organic molecules, ${}^1\text{H}$ is therefore by far the most popular nucleus to study using NMR, at least for studying solution-state samples.

The bulk magnetism experiences a torque induced by the magnetic field, with its rate of change with respect to time being

$$\frac{d\mathbf{M}(t)}{dt} = \mathbf{M}(t) \times \gamma \mathbf{B}(t). \quad (1.9)$$

The essence of NMR is to manipulate and subsequently detect the evolution of \mathbf{M} . Manipulation is achieved by applying short bursts of radio-frequency (RF) radiation known as *pulses* which perturb the net magnetic field away from \mathbf{B}_0 . The contribution to the field induced by pulses is commonly denoted \mathbf{B}_1 , such that at any given time

$$\mathbf{B}(t) = \mathbf{B}_0 + \mathbf{B}_1(t). \quad (1.10)$$

Whenever the magnetisation vector is not collinear with the field vector*, it undergoes *precession*

*The cross product of two collinear vectors is 0, so \mathbf{M} remains fixed when it is aligned with \mathbf{B} , as is the case at equilibrium.

about the field vector. Free precession occurs when the magnetisation is aligned away from the z -axis, and $\mathbf{B}_1 = \mathbf{0}$, such that

$$\frac{d\mathbf{M}(t)}{dt} = -\gamma B_0 [M_y \quad -M_x \quad 0]^T, \quad (1.11)$$

i.e. the magnetisation precesses around the z -axis at the *Larmor frequency* $\omega_0 = -\gamma B_0$.[†]

Assuming the RF field is linearly polarised along the x -axis, it can be written to a high degree of accuracy as

$$\mathbf{B}_1(t) = B_1 (\cos(\omega_{RF}t + \phi_{RF})\mathbf{i} + \sin(\omega_{RF}t + \phi_{RF})\mathbf{j}), \quad (1.12)$$

with \mathbf{i} , \mathbf{j} and \mathbf{k} (encountered shortly) being unit vectors along the x -, y -, and z -axes, respectively. B_1 is the strength of the RF field, ω_{RF} is its angular frequency, and ϕ_{RF} is its phase.

A great simplification to the model is realised by considering a frame of reference which, rather than being static, rotates at ω_{RF} , as this makes the RF field appear to be time-independent. This is referred to as the *rotating frame*, and leads to Equation 1.9 being recast as

$$\frac{d\tilde{\mathbf{M}}(t)}{dt} = \tilde{\mathbf{M}}(t) \times \gamma \tilde{\mathbf{B}}(t), \quad (1.13a)$$

$$\tilde{\mathbf{B}} = B_1 \cos(\phi_{RF})\tilde{\mathbf{i}} + B_1 \sin(\phi_{RF})\tilde{\mathbf{j}} + \Delta B_0 \tilde{\mathbf{k}}, \quad (1.13b)$$

$$\Delta B_0 = -\frac{\Omega}{\gamma}, \quad (1.13c)$$

$$\Omega = -\gamma B_0 - \omega_{RF}, \quad (1.13d)$$

$$\tilde{\mathbf{i}} = \cos(\omega_{RF})\mathbf{i}, \quad (1.13e)$$

$$\tilde{\mathbf{j}} = \sin(\omega_{RF})\mathbf{j}, \quad (1.13f)$$

$$\tilde{\mathbf{k}} = \mathbf{k}. \quad (1.13g)$$

$\Omega = \omega_0 - \omega_{RF}$ is the *offset* of the spin magnetisation. When $\Omega = 0 \text{ rad s}^{-1}$, the system is said to be on-resonance, and at times when an RF field is not being applied, the magnetisation vector appears to be static in the rotating frame.

Consider a scenario where a short RF pulse is applied to the system, which is then allowed to undergo free precession. Equation 1.13 implies that $\tilde{\mathbf{M}}$ will rotate indefinitely about the z -axis with a frequency of Ω . However, in reality the system is driven to re-establish its thermal equilibrium state, $\mathbf{M}_{\text{eq}} \equiv \tilde{\mathbf{M}}_{\text{eq}} = [0, 0, M_0]^T$. In the Bloch model, two processes are introduced to account for this[‡] *longitudinal* (spin-lattice) relaxation, which is responsible for the recovery of

[†]While the SI base unit of magnetic field strength is the Tesla (T), when referring to the field strength that a spectrometer operates at, it is common to use MHz instead. This refers to the (negative) Larmor frequency of an isolated ¹H nucleus at the given field strength. For example, a 500 MHz spectrometer operates at a field strength of $5 \times 10^8 \text{ Hz} / 4.2577 \times 10^7 \text{ T}^{-1} \text{ Hz} \approx 11.74 \text{ T}$.

[‡]The introduction of relaxation is phenomenological in the Bloch model; the reversion of the spin system to equilib-

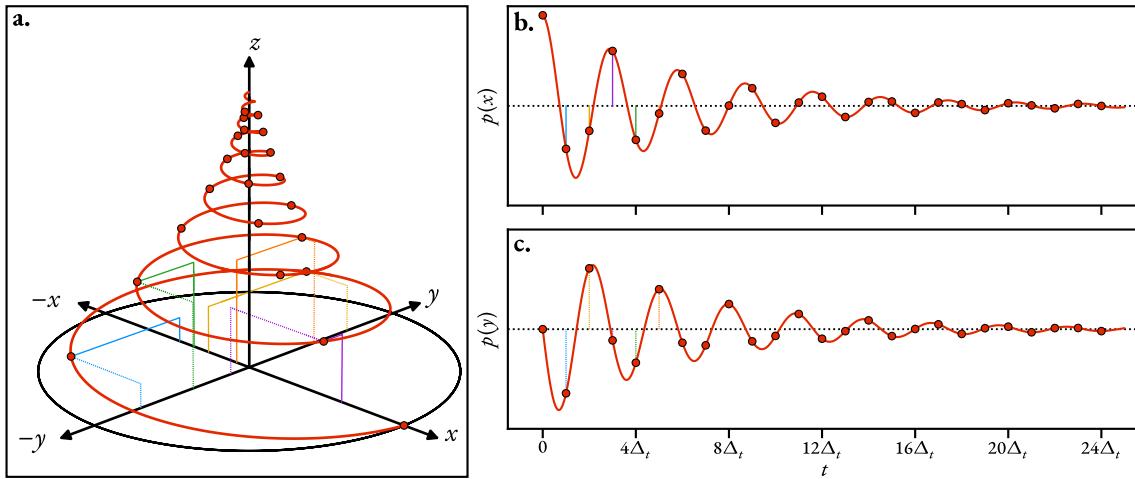


FIGURE 1.2: **a.** An illustration of the free evolution of the bulk magnetisation of an ensemble of spin- $1/2$ nuclei immediately after the application of a 90°_y pulse according to the Bloch model. The projections of the magnetisation vector onto the x - and y -axes are plotted in panels **b.** and **c.**, respectively. Modern NMR spectrometers utilise quadrature detection, such that the x - and y -projections of the time-varying magnetisation is sampled at regular intervals, separated by Δ_t . The resulting FID is given by the complex value $p(x) + ip(y)$.

the z -component of the magnetisation to M_0 , and *transverse* (spin-spin) relaxation, which leads to the x - and y -components reverting to 0. These are assigned rates of R_1 and R_2 , respectively.

Everything has now been established to state the Bloch equations, which describe the evolution of the bulk magnetisation of an ensemble of identical spin- $1/2$ nuclei in the rotating frame:

$$\frac{d\tilde{\mathbf{M}}(t)}{dt} = \begin{bmatrix} -R_2 & -\Omega & -\gamma B_1 \sin(\phi_{RF}) \\ \Omega & -R_2 & \gamma B_1 \cos(\phi_{RF}) \\ \gamma B_1 \sin(\phi_{RF}) & -\gamma B_1 \cos(\phi_{RF}) & -R_1 \end{bmatrix} \tilde{\mathbf{M}}(t) + R_1 M_0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad (1.14)$$

Figure 1.2.a depicts the evolution of an off-resonance ($\Omega \neq 0 \text{ rad s}^{-1}$) magnetisation vector after the application of an RF pulse with $\phi_{RF} = \pi/2$, and an appropriate combination of duration and power to induce a clockwise rotation of 90° about the y -axis. Such a pulse is denoted 90°_y , which indicates the *flip angle* it induces, and its phase. Assuming that negligible evolution due to the offset occurs during the pulse, the magnetisation vector will land on the x -axis, and evolve according to

$$\tilde{M}_x(t) = M_0 \cos(\Omega t) \exp(-R_2 t), \quad (1.15a)$$

$$\tilde{M}_y(t) = M_0 \sin(\Omega t) \exp(-R_2 t), \quad (1.15b)$$

$$\tilde{M}_z(t) = M_0(1 - \exp(-R_1 t)). \quad (1.15c)$$

rium is included purely to ensure the model agrees with observation. More sophisticated theories evoking Liouville-space quantum mechanics can account for relaxation[23, 24].

During acquisition, the transverse components of the bulk magnetisation are detected by the spectrometer probe circuitry, such that the resulting signal, called the *free induction decay* (FID), is given by

$$y(t) = c\tilde{M}_+(t), \quad (1.16a)$$

$$\tilde{M}_+(t) = \tilde{M}_x(t) + i\tilde{M}_y(t) = M_0 \exp(i\Omega t - R_2 t), \quad (1.16b)$$

with $c \in \mathbb{R}_{>0}$ being a proportionality constant.

1.2.3 The NMR Spectrometer

Modern NMR spectrometers are capable of conducting a plethora of experiments to help gain insights into chemical systems of interest. In essence, a spectrometer comprises a high-field magnet, a probe, components which are used to transmit RF pulses to the probe, and components which are used to process the resulting signal from the probe. A brief summary of these is now given.

The Magnet

The static \mathbf{B}_0 field is generated by a magnet, composed of a superconducting solenoid immersed in liquid helium. Common materials used for the solenoid include Nb-Ti alloy and Nb₃Sn. To minimise the extent of helium evaporation, the dewar containing the helium is lined with a thermal radiation shield. The helium dewar is then surrounded by a larger dewar containing liquid nitrogen. A bore passes through the z -direction of the magnet, which is maintained at a user-controlled temperature. Within the bore sits the probe as well as the sample. Magnets with high field strengths are desirable, as both the resolution ($\propto B_0$) and sensitivity ($\propto B_0^{3/2}$) of the data are affected. At the time of writing, commercial spectrometers which operate at and above a ¹H Larmor frequency of 1 GHz (23.5 T) exist, though these are uncommon and are employed primarily for the study of large biomolecules. For most applications, including the study of small molecules, spectrometers with more modest field strengths are typically adequate.

A field-frequency *lock* is used to ensure the stability of the magnetic field. The lock is a small NMR spectrometer, typically tuned to ²H^{\$}, which monitors the sample ²H resonance frequency over time. If the frequency begins to drift, implying that the field strength is changing, the current in the “ Z_0 coil” is adjusted to ensure the ²H frequency is maintained.

To maintain high spatial field homogeneity (a necessity for data with acceptable resolution) a series of coils called *shims* surround the sample. Each coil produces a weak magnetic field with

^{\$}²H-enriched solvents are routinely used to make up NMR samples. In ¹H NMR experiments, this ensures that an extremely intense signal due to the solvent does not dwarf the signals from other spins in the sample. This makes ²H a suitable nucleus to monitor by the lock, as it is ubiquitous, but rarely directly studied.

a specific spatial profile according to a spherical harmonic function, which can cancel out any inhomogeneity inherent to the main magnet.

The Probe

The probe sits inside the bore of the magnet, and features the coil(s) used to pulse the sample with RF radiation. The same coil(s) also receive the response from the sample. The principle source of noise in NMR experiments is thermal noise within the probe circuitry. For this reason, cryogenic probes[25] have become a popular development, in which the transmit/receive coil(s) and other probe electronics are maintained at a very low temperature (typically about 20K).

The Transmitter

The transmitter is responsible for the generation of RF pulses with specified power, timing and phase. A synthesiser acts as an RF source, producing a continuous carrier wave at or very close to the Larmor frequency of the target nucleus. This frequency (ω_{RF}) can be adjusted in order to determine the center of the spectrum. The difference between the carrier frequency and the reference “basic frequency” of the spectrometer is referred to as the *transmitter offset* f_{off} . The output of the synthesiser is gated to ensure pulses are applied at the desired times. An attenuator and amplifier then adjust the power of the pulse, which travels to the probe.

The Receiver

During acquisition, the time-varying current induced in the probe coil(s) by the sample magnetisation is sent to a receiver, comprising a series of components for converting the analogue current to the digital FID which is stored in computer memory. One of the processes that the receiver is responsible for is *quadrature detection*[26: Section 13.6]. Quadrature detection ensures FIDs are frequency discriminated, i.e. that they posses the requisite information to determine whether a given signal in FID has a frequency that is above or below the transmitter frequency in the middle of the spectral window. This is achieved by splitting the signal into two channels. Both signals, which are of very high frequency (MHz) are mixed with a reference signal of frequency ω_{RF} , such that low-frequency signals (kHz) are generated, along with very high frequency signals. One of the reference signals possesses a phase which is shifted by 90° relative to the other one, such that the two channels constitute a quadrature pair. The signals are then sent through a low-pass filter to remove the high frequency component produced through mixing. Finally, an analogue to digital converter translates the signal to the real and imaginary components of a binary dataset which is stored in memory.

1.2.4 The structure of the FID

The result of running an NMR experiment is an FID \mathbf{y} which is sampled at equally-spaced time-points, separated by time Δ_t :

$$\begin{aligned}\mathbf{y} &= \begin{bmatrix} y_0 & y_1 & y_2 & \cdots & y_{N-1} \end{bmatrix}^T \\ &\equiv \begin{bmatrix} y(t=0) & y(t=\Delta_t) & y(t=2\Delta_t) & \cdots & y(t=(N-1)\Delta_t) \end{bmatrix}^T\end{aligned}\quad (1.17)$$

where $y(t)$ is the (continuous) variation of the generated signal with time, and N is the number of points sampled. The inverse of the sampling rate, $1/\Delta_t$, is the *sweep width* f_{sw} which defines how wide the range of samplable frequencies is, in accordance with the Nyquist theorem[27].

FIDs adopt the form of a summation of $M \in \mathbb{N}$ complex exponentials (signals). Each signal will be subjected to damping due to transverse relaxation, which is typically exponential in nature. An FID acquired with N samples $\mathbf{y} \in \mathbb{C}^N$ therefore takes the form[¶]

$$y_n = x_n(\boldsymbol{\theta}) + w_n \quad \forall n \in \{0, 1, \dots, N-1\}, \quad (1.18a)$$

$$x_n(\boldsymbol{\theta}) = \sum_{m=1}^M a_m \exp(i\phi_m) \exp((2\pi i(f_m - f_{\text{off}}) - \eta_m)n\Delta_t). \quad (1.18b)$$

Equation 1.18 indicates that an FID comprises contributions from the (deterministic) evolution of the spin magnetisation \mathbf{x} and experimental noise \mathbf{w} (*vide infra*). Each signal which contributes to \mathbf{x} is defined by four parameters:

- Amplitude $a \in \mathbb{R}_{>0}$,
- Phase $\phi \in (-\pi, \pi]$ (rad),
- Frequency $f \in [f_{\text{off}} - 1/2f_{\text{sw}}, f_{\text{off}} + 1/2f_{\text{sw}}]$ (Hz),
- Damping factor $\eta \in \mathbb{R}_{>0}$ (s^{-1}).

An FID can therefore be parameterised by the vector $\boldsymbol{\theta} \in \mathbb{R}^{4M}$:

$$\boldsymbol{\theta} = \begin{bmatrix} \mathbf{a}^T & \boldsymbol{\phi}^T & \mathbf{f}^T & \boldsymbol{\eta}^T \end{bmatrix}^T, \quad (1.19)$$

where $\mathbf{a} \in \mathbb{R}^M = [a_1 \ a_2 \ \cdots \ a_M]^T$ is a vector of all amplitudes, $\boldsymbol{\phi} \in \mathbb{R}^M$ is a vector of all phases, etc. An alternative, more concise, notation for Equation 2.1b involves the *complex amplitudes*

[¶]This provides an idealised model of the form of an FID, based on the underlying theory of the experiment. In reality, there is the possibility of significant deviations from this model being realised. One potential cause of this is the influence of magnetic field inhomogeneities, which will cause spectral peaks to deviate from an idealised Lorentzian lineshape (Section 1.3.1). In cases where distortions to the data have occurred, techniques such as *reference deconvolution*[28] can be used as a corrective measure.

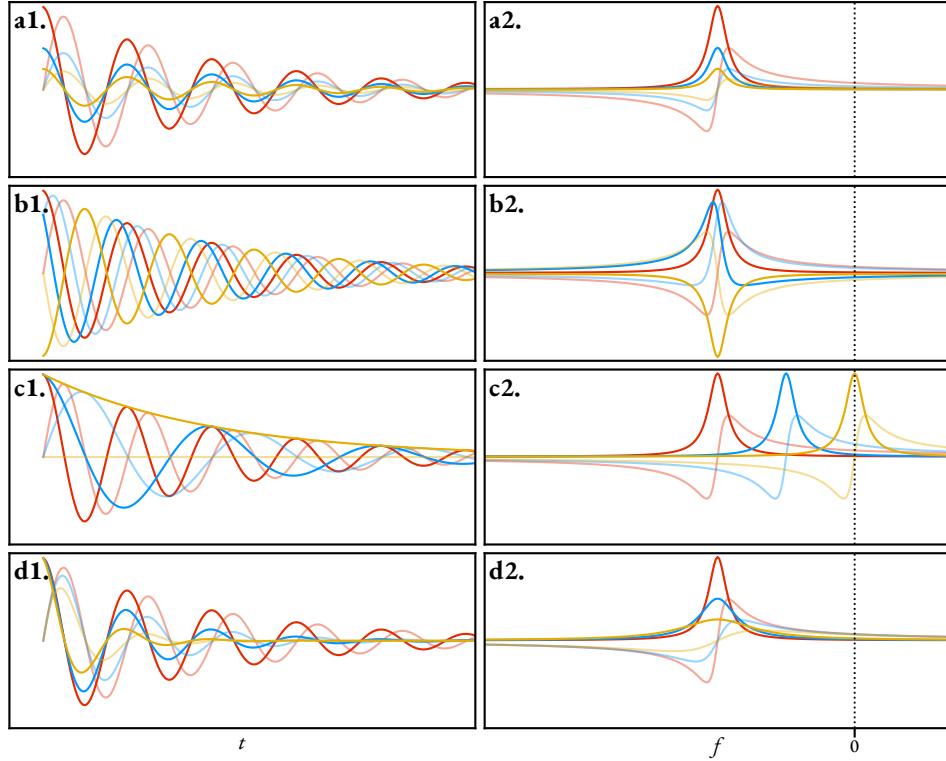


FIGURE 1.3: An illustration of the influence of the four parameters associated with a signal in both the time-domain (panels **a1.** – **d1.**) and Fourier-domain (panels **a2.** – **d2.**). The red signal is generated with the same parameters across all panels: $\alpha = \alpha_{\text{red}}$, $\phi = 0$, $f = f_{\text{red}}$, $\eta = \eta_{\text{red}}$. The blue and yellow signals were produced by altering one parameter out of the four. **a.** $\alpha_{\text{yellow}} = 1/2\alpha_{\text{blue}} = 1/4\alpha_{\text{red}}$. **b.** $\phi_{\text{blue}} = \pi/4$, $\phi_{\text{yellow}} = \pi$. **c.** $f_{\text{blue}} = 1/2f_{\text{red}}$, $f_{\text{yellow}} = 0$. **d.** $\eta_{\text{blue}} = 1/2\eta_{\text{red}}$, $\eta_{\text{yellow}} = 1/4\eta_{\text{red}}$. The real and imaginary components of each signal are plotted, with the imaginary component being paler than its real counterpart.

and *signal poles* associated with the FID:

$$x_n(\theta) = \sum_{m=1}^M \alpha_m z_m^n, \quad (1.20a)$$

$$\alpha_m = \alpha_m \exp(i\phi_m), \quad (1.20b)$$

$$z_m = \exp((2\pi i(f_m - f_{\text{off}}) - \eta_m) \Delta_t). \quad (1.20c)$$

The influence the four parameters have on a signal is depicted in panels a1–d1 in Figure Figure 1.3.

Multidimensional experiments involve incrementing one or more delay within the pulse sequence, in order to obtain an array of 1D FIDs. In a D -dimensional dataset, each contributing signal is parameterised by an amplitude and phase as before, along with D distinct frequencies and damping factors, such that a general parameter vector $\theta \in \mathbb{R}^{2(1+D)M}$ is given by

$$\theta = [\alpha^T \ \phi^T \ \mathbf{f}^{(1)T} \ \dots \ \mathbf{f}^{(D)T} \ \boldsymbol{\eta}^{(1)T} \ \dots \ \boldsymbol{\eta}^{(D)T}]^T, \quad (1.21)$$

where $\mathbf{f}^{(D)}$ and $\boldsymbol{\eta}^{(D)}$ are the frequencies and damping factors in the actively acquired (direct) dimension, and $\mathbf{f}^{(1)} - \mathbf{f}^{(D-1)}$ and $\boldsymbol{\eta}^{(1)} - \boldsymbol{\eta}^{(D-1)}$ are those for the indirect dimension(s). Indirect dimensions can exhibit different forms of evolution, depending on the precise nature of the pulse sequence. Two common functional forms exist[22: Section 4.3.4]. Signals of the form $\cos(2\pi ft)$ and $\sin(2\pi ft)$ modulate the amplitude of the direct dimension signal across increments, while those of the form $\exp(2\pi ift)$ or $\exp(-2\pi ift)$ modulate the phase. Amplitude- or phase-modulated signals are often acquired as pairs when possible, as this ensures that spectra with desirable properties can be generated (*vide infra*). In general, a D -dimensional FID $\mathbf{Y} \in \mathbb{C}^{N^{(1)} \times \dots \times N^{(D)}}$ can be expressed as

$$\mathcal{Y}_{n^{(1)}, \dots, n^{(D)}} = \mathcal{X}_{n^{(1)}, \dots, n^{(D)}}(\boldsymbol{\theta}) + w_{n^{(1)}, \dots, n^{(D)}}, \quad (1.22a)$$

$$\mathcal{X}_{n^{(1)}, \dots, n^{(D)}} = \sum_{m=1}^M a_m \exp(i\phi_m) \prod_{d=1}^D \zeta^{(d)} \left(2\pi \left(f_m^{(d)} - f_{\text{off}}^{(d)} \right) n^{(d)} \Delta_t^{(d)} \right) \exp \left(-\eta_m^{(d)} n^{(d)} \Delta_t^{(d)} \right), \quad (1.22b)$$

$$\zeta^{(d)}(\cdot) \begin{cases} = \exp(i\cdot) & d = D \\ \in \{\cos(\cdot), \sin(\cdot), \exp(i\cdot) \exp(-i\cdot)\} & \text{otherwise} \end{cases}, \quad (1.22c)$$

It is typical to assume that \mathbf{W} is an array of additive white gaussian noise (AWGN), i.e. the noise instances are described by a complex normal distribution with mean 0, and pairs of noise instances are statistically independent, regardless of their time separation:

$$w_{n^{(1)}, \dots, n^{(D)}} \sim \mathcal{N}_C(0, 2\sigma^2) \quad (1.23a)$$

$$\begin{aligned} \implies \Re(w_{n^{(1)}, \dots, n^{(D)}}) &\perp\!\!\!\perp \Im(w_{n^{(1)}, \dots, n^{(D)}}), \\ \Re(w_{n^{(1)}, \dots, n^{(D)}}) &\sim \mathcal{N}(0, \sigma^2), \\ \Im(w_{n^{(1)}, \dots, n^{(D)}}) &\sim \mathcal{N}(0, \sigma^2). \end{aligned} \quad (1.23b)$$

The extent by which a signal is corrupted by noise is given by the signal-to-noise ratio (SNR), the ratio of signal power and noise power:

$$\text{SNR}(\mathbf{Y}) := \frac{1}{2\Re\sigma^2} \sum_{n^{(1)}=0}^{N^{(1)}-1} \dots \sum_{n^{(d)}=0}^{N^{(d)}-1} \left| \mathcal{X}_{n^{(1)}, \dots, n^{(D)}} \right|^2, \quad (1.24)$$

where $\Re := N^{(1)} \times \dots \times N^{(D)}$ is the total number of points the signal comprises. Due to the large dynamic range of the SNR across datasets, it is common to express it using a logarithmic scale

instead, in units of decibels (dB):

$$\text{SNR}_{\text{dB}} := 10 \log_{10} (\text{SNR}). \quad (1.25)$$

1.3 NMR Data Processing and Analysis

TODO

1.3.1 Conventional NMR Processing

The typical means of processing NMR data is to transform the time (measured) domain FID into the frequency domain, producing an NMR spectrum. This is achieved through application of the *Fourier transform* (FT). The FT converts a continuous function over time t , $x(t)$, to one over frequency F , $s(F)$, as follows^{||}:

$$s(F) = \int_0^{\infty} x(t) \exp(-2\pi i t F) dt \quad (1.26)$$

In practice, the discrete signal acquired by the spectrometer is processed using the discrete Fourier transform (DFT):

$$s_n = \sum_{k=0}^{N-1} x_k \exp\left(-\frac{2\pi i k n}{N}\right) \quad \forall n \in \{0, \dots, N-1\}. \quad (1.27)$$

The FT of a continuous exponentially-damped complex sinusoid takes the form of a *Lorentzian*:

$$s(F) = \int_0^{\infty} \alpha \exp(i\phi) \exp((2\pi i f - \eta)t) \exp(-2\pi i t F) dt, \quad (1.28a)$$

$$s(F) = \frac{\alpha \exp(i\phi)}{\eta + 2\pi i(f - F)}. \quad (1.28b)$$

When processing discrete data, the DFT of an FID features a vertical offset, such that the baseline does not sit at 0[29]. This results from only possessing data for positive values of t , rather than having a “full echo”, where negative values of t are accounted for too. Fortunately, this can be corrected easily by halving the initial point of the FID prior to FT. When $\phi = 0$, this function is equivalent to the (unnormalised) probability density function (pdf) of the Cauchy distribution. The FT is a linear function, such that the FT of a summation of signals is equivalent to the summation of the FTs of each signal. A corollary is that an NMR spectrum comprises a series of

^{||}In general, the FT is defined over the range $(-\infty, \infty)$. Since NMR data isn't defined for $t < 0$, the FT has been defined over the range $[0, \infty)$.

Lorentzian “peaks”, located at the frequencies of all the signals in the FID:

$$s(F) = \sum_{m=1}^M \frac{a_m \exp(i\phi_m)}{\eta_m + 2\pi i(f_m - F)}. \quad (1.29)$$

The FT is a very attractive means of processing NMR data, as it presents the data in a format which is human-interpretable, with the basic rules describing how chemical structure is mapped to NMR spectral properties being a fundamental skill that experimentalists in many fields require[30]. Due to innate properties of the NMR experiment, as well as issues arising from analysing discrete signals, the FT of the raw output of an NMR experiment has undesirable characteristics without extra manipulation. Extra processing steps that are frequently applied to NMR data are now outlined, in the order that they are applied to the data. Note that apodisation and zero filling are applied to the prior to FT (i.e. to the FID), while phase correction and baseline correction are applied after FT (i.e. to the resulting NMR spectrum).

Apodisation

Apodisation refers to the process of mutating a signal by multiplying it with a specified function, often called a *window function*[31: Section 3.2.7], in order to enhance either the sensitivity or the resolution of the final spectrum, albeit at the cost of worsening the other feature.

Sensitivity Enhancement As an FID progresses with time, the contributions from the desirable signal (\mathbf{x}) and experimental noise (\mathbf{w}) becomes more weighted towards the noise, as spin relaxation phenomena dampen the signal. As such, multiplying the FID with a function which is initially large and gets progressively smaller can be used to enhance the SNR of the FID. The most common function to achieve this is the negative exponential, in which a given point is multiplied by $\exp(-kn/N-1)$. k is referred to as the line broadening factor, since the increased dampening applied to the signal causes the linewidth of the spectral peaks to increase (see panel d of Figure 1.3).

Resolution Enhancement By contrast, resolution enhancement can be achieved by applying a window function that artificially reduces the rate of oscillator decay. This is achievable through the suppression of points that are early in the FID. Popular examples of window functions for resolution enhancement are the Lorentz-Gauss function, which bestow a sharper, Gaussian shape to the peaks, and the sine-bell, which is commonly used in multidimensional experiments. Since the initial points in the FID are attenuated, these window functions reduce the sensitivity of the resulting spectra.

By being defined to decay to a sufficiently small value (or 0 itself) at the end of the FID, window functions are also able to suppress *truncation artefacts*, which appear when the FID still possesses appreciable signal amplitude at the end of the acquisition period. Truncated FIDs produce spectra

with peaks of a form that is akin to the convolution of the FT of the untruncated FID, with the FT of a box-function, which takes the form of a sinc function ($\frac{\sin(x)}{x}$). The resulting artefacts in spectra are often referred to as *sinc wiggles* for this reason.

Zero-filling

Zero-filling refers to the process of appending zeros to the end of an FID. One reason why this is often done is to ensure that the number of points in the FID is a power of 2, making the FID of an optimal size for processing by divide and conquer methods like the Cooley-Tukey algorithm[10]. Beyond this, it is actually possible to enhance the information content of the spectrum if an FID comprising $2^{m-1} < N \leq 2^m$ points is zero-filled to 2^{m+1} points[32]. By doing this, the real and imaginary components of the spectrum become causally linked; the real component can be derived from the imaginary component and vice versa according to the Kramers-Kronig relations via the Hilbert transform. Zero-filling beyond a factor of 2 can bestow a cosmetic improvement to the spectrum — the additional points are simply interpolations — but it does not incorporate any new information into it.

Phase correction

The real and imaginary components of Equation 1.29 are as follows:

$$\Re(s(F)) = \sum_m a_m (\cos(\phi_m) \mathcal{A}_m(F) + \sin(\phi_m) \mathcal{D}_m(F)), \quad (1.30a)$$

$$\Im(s(F)) = \sum_m a_m (\sin(\phi_m) \mathcal{A}_m(F) - \cos(\phi_m) \mathcal{D}_m(F)), \quad (1.30b)$$

where \mathcal{A}_m and \mathcal{D}_m denote *absorption* and *dispersion* Lorentzians, respectively:

$$\mathcal{A}_m(F) = \frac{\eta_m}{\eta_m^2 + 4\pi^2(f_m - F)^2}, \quad (1.31a)$$

$$\mathcal{D}_m(F) = \frac{2\pi(f_m - F)}{\eta_m^2 + 4\pi^2(f_m - F)^2}. \quad (1.31b)$$

As illustrated most clearly in panel c2 of Figure 1.3, a peak with an absorption lineshape is far more desirable than one with a dispersion lineshape for two key reasons:

- (i) Its maximum corresponds to the signal frequency, while a dispersion Lorentzian has a magnitude of 0 at the oscillator frequency.
- (ii) It decays more rapidly** and therefore exhibits better resolution.

** At large frequency offsets, absorption Lorentzians decay at a rate $\propto 1/(f_m - F)^2$, while dispersive Lorentzians decay at a rate $\propto 1/(f_m - F)$.

Generating a spectrum whose real component comprise peaks which all possess absorption Lorentzians is therefore desired, which is possible if all signals have a phase of 0° :

$$\Re(s(F)) = \sum_m a_m \mathcal{A}_m(F), \quad (1.32\text{a})$$

$$\Im(s(F)) = - \sum_m a_m \mathcal{D}_m(F). \quad (1.32\text{b})$$

For the majority of NMR experiments^{††}, the contributing signals possess phases which depend linearly on their frequencies, i.e.

$$\phi_m = \Phi_0 + \Phi_1 f_m, \quad (1.33)$$

where $\Phi_0 \in (-\pi, \pi]$ and $\Phi_1 \in \mathbb{R}$ are zero- and first-order phase terms. Phase correction refers to the process in which Φ_0 and Φ_1 are determined by inspecting the appearance of the spectrum s_ϕ for different values of p_0 and p_1 , according to

$$s_{\phi,n} = s_n \exp \left(-i \left(p_0 + \frac{p_1 n}{N-1} \right) \right). \quad (1.34)$$

When $p_0 = \Phi_0$ and $p_1 = \Phi_1$, the spectrum will be correctly phased.

Baseline correction

The *baseline* of an NMR spectrum is used to describe regions where no discernible peaks reside (i.e. only experimental noise exists). Baseline distortion describes scenarios in which the baseline, rather than exhibiting a flat profile with an average of zero, has some other form. There a number of potential causes of baseline distortion, including “clipping” of the initial points due to excessive receiver gain, and pulse breakthrough. Whatever the cause(s), it is typically a corruption of the initial points in the FID that causes baseline distortion. It is common to apply a baseline correction algorithm to negate any distortion, which involves multiplying the spectrum with a spline or high-order polynomial. One popular approach involves the two principal steps of (a) determining spectral regions which are part of the baseline, and (b) fitting the baseline regions to the chosen function[33, 34].

1.3.2 Conventional processing for multidimensional datasets

As for 1D datasets, it is desirable that multidimensional NMR spectra feature peaks which are frequency discriminated, and comprise pure absorption-mode lineshapes in each dimension. A description of how this can be achieved with both amplitude- and phase-modulated datasets is

^{††}An example of an exception to this rule is when frequency-swept pulses are applied, which generate datasets with quadratic phase behaviour. This is discussed in more detail in Section 3.3.

now given, through the consideration of 2D signals.

Amplitude-modulated signals

It is clear that a cosine-modulated signal, given by Equation 1.22 with $D = 2$ and $\zeta^{(1)} = \cos(\cdot)$ cannot achieve frequency discrimination, on account of the relation

$$\cos(2\pi f^{(1)} t^{(1)}) = \frac{1}{2} (\exp(2\pi i f^{(1)} t^{(1)}) + \exp(-2\pi i f^{(1)} t^{(1)})) \quad (1.35)$$

Performing FT on such a signal in both dimensions leads to a spectrum whose real component comprises two peaks, one at the true resonance frequency $(f^{(1)}, f^{(2)})$, and the other at the mirror-image frequency in the indirect dimension, $(2f_{\text{off}}^{(1)} - f^{(1)}, f^{(2)})$. On top of this, the peaks possess a mixture of absorption and dispersion character, with the resultant peak shape often referred to as *phase twist*[35]. A spectrum of this form is presented in panel a of Figure 1.4. It is possible to generate pure absorption peaks by applying FT in the direct dimension, setting the imaginary component to zero, and finally applying FT in the indirect dimension. The real component of the result (panel b) is referred to as a *double absorption* spectrum.

For frequency discrimination, it is necessary to also possess the analogous sine-modulated signal ($\zeta^{(1)} = \sin(\cdot)$) as this achieves quadrature detection in the indirect dimension. For numerous multidimensional experiments, repeating the pulse sequence, with careful adjustments to the phases of particular pulses (*phase cycling*) enables this[26: Chapter 11]. Applying the same processing as that which achieved the double absorption spectrum for the cosine-modulated case generates a spectrum whose imaginary component features two peaks, but with opposite signs (panel c). It then becomes possible to generate a frequency discriminated spectrum by subtracting the sine spectrum from the cosine spectrum (panel d).

Phase-modulated signals

A “positive” phase-modulated signal with the form $\zeta^{(1)} = \exp(i\cdot)$ (commonly referred to as *hypercomplex*) is frequency discriminated due to its quadrature nature. However, direct FT of such a signal in both dimensions leads to a peak with a phase-twist lineshape, with no means of separating the absorption and dispersion contributions (panel e). Certain pulse sequences, including 2DJ spectroscopy[36, 37] (see Chapter 4) and correlation spectroscopy (COSY)[11–13] produce hypercomplex datasets, and the conventional means of processing is simply to display the spectrum in *magnitude-mode*, where the absolute value of each complex point is plotted. While the phase twist shapes are removed by doing this, the resulting peaks have broad “wings”, due to the influence of the dispersive components. In scenarios where it is possible, it is desirable to acquire the equivalent “negative” signal ($\zeta^{(1)} = \exp(-i\cdot)$), whose FT leads to a peak with the same phase twist form, but centered at $2f_{\text{off}}^{(1)} - f^{(1)}$ (panel f). Inverting this spectrum in $F^{(1)}$ (panel g), and

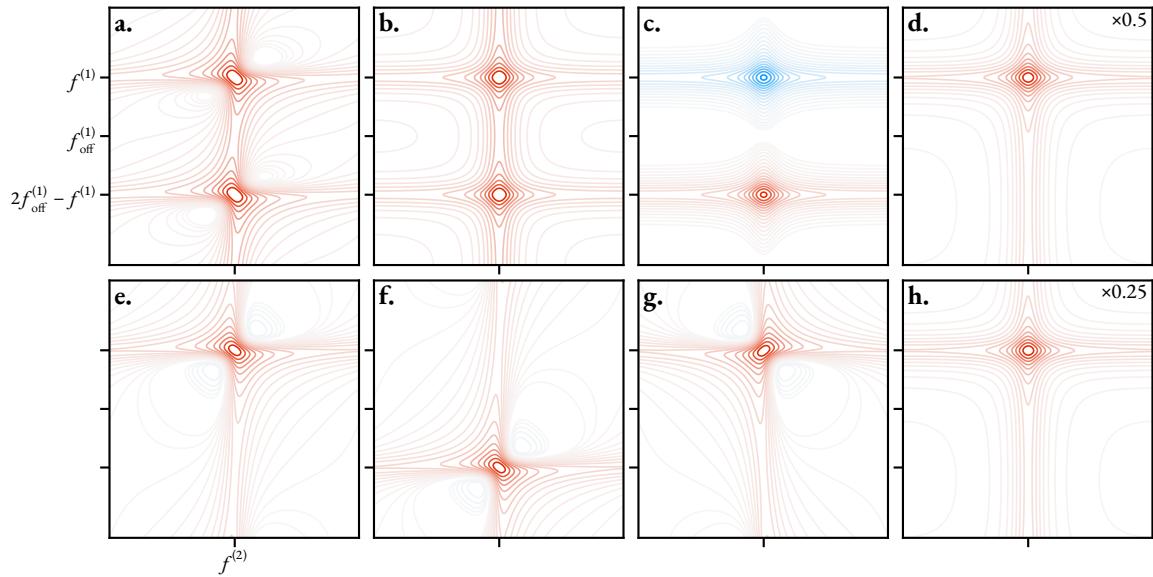


FIGURE I.4: Spectra acquired from amplitude- and phase- modulated 2D signals. Red contour lines denote positive values, while blue contours denote negative values. **a.** The FT of a cosine-modulated FID, featuring peaks both at the true resonance frequency ($f^{(1)}$) and the mirrored frequency ($2f_{\text{off}}^{(1)} - f^{(1)}$), and with phase-twist lineshapes. **b.** Double-absorption spectrum generated by applying FT in the direct dimension, setting the imaginary component to zero, applying FT in the indirect dimension, and retaining the real component. **c.** Spectrum acquired with the same processing method as in **b.** but with a sine-modulated FID, and with the imaginary component retained. **d.** The subtraction of the spectrum in **c.** from that in **b.** leads to a spectrum with frequency discrimination and a pure absorption lineshape. **e.** The FT of a positive phase-modulated FID, exhibiting frequency discrimination, but with a phase twist shape. **f.** The FT of a negative phase-modulated FID. **g.** The spectrum in **f.** inverted along the indirect axis, about $f_{\text{off}}^{(1)}$. **h.** The summation of **e.** and **g.** generates a spectrum with an absorption lineshape.

summing with the positive spectrum nullifies the dispersive contribution, generating a spectrum with an absorption lineshape[38] (panel h).

The concepts described here for 2D signals can be extended for the processing of NMR datasets with any number of dimensions, provided that in each indirect dimension, the requisite pair of amplitude- or phase-modulated signals exist. As such, a set of 2^{D-1} signals need to be acquired.

1.3.3 Conventional NMR analysis

To improve and expand... The amount of detail required from an NMR dataset depends on the user's requirements. In synthetic organic chemistry, NMR is often used simply as a means of verifying that a particular step in a synthetic pathway was successful. A simple inspection of peak locations and splittings due to scalar couplings in the spectrum may be sufficient to verify that the desired molecule was created. In many situations however, a detailed quantitative description of the data is desired. Some examples include:

- Deducing the relative concentrations of a mixture of different molecules, with application in areas such as metabolomics[39]. In these circumstances, it is necessary to deduce the relative amplitudes of the signals in the data.
- Assessing the effects of ligand binding to a protein[40]. Incorporating a ligand into a sample containing a protein can induce changes in the resonance frequencies of the protein nuclei.

Most NMR users are principally interested in the integrals and positions of peaks in spectra^{#‡}. Integrals are usually computed based on user-specified regions of the spectrum, while the most rudimentary form of “peak picking” — beyond the user simply defining where peaks are by inspecting the spectrum — involves assessing where relative maxima exist, subject to tolerances which ensure that maxima that are likely due to the presence of noise do not lead to roughly 1/2 of all points in the spectrum being picked.

1.3.4 Estimation Techniques for NMR Analysis

While integration and peak picking works adequately for high SNR spectra featuring very well-separated peaks, in most practical situations accurate quantification of individual resonances is impossible using this approach, principally due to the inherently large poor resolution associated with the FT. For this reason, considerable interest has been given to the development of tech-

^{#‡}There may be some situations where the linewidth of the peaks are of interest too. An absorption Lorentzian produced from an exponentially-damped complex sinusoid with a damping factor γ will have a linewidth at half its maximum intensity of γ/π . The linewidth of peaks give an insight into the rate of transverse relaxation (T_2) associated with a given spin.

niques which are better-able to estimate NMR datasets^{§§}. Here, a summary of some of the most prominent methods for quantifying NMR data are given. Attention is primarily given to methods which consider time-domain data, as it is the focus of this work.

Linear prediction

Linear prediction (LP)[41, 42] is a procedure which is widely used in NMR data analysis, for the purposes of (a) propagating the FID further in time in order to reduce the presence of truncation artefacts, and/or (b) correcting the commonly corrupted initial points of the FID, as a means of improving the spectral baseline. LP is considerably different to the other methods presented here; while the other approaches aim to determine the parameters defining each contributing signal in the FID, LP is used to provide a holistic picture, describing how a datapoint in the FID is related to datapoints which precede or succeed it. As such, LP does not provide insights into individual signals in the FID.

The concept of LP stems from the idea that a deterministic signal like a 1D FID can be described as an autoregressive (AR) process, meaning that a given sample from the dataset is a linear combination of an appropriate number L of previous samples:

$$y_n = \sum_{l=1}^L c_l y_{n-l} + e_l \quad \forall n \in \{L, L+1, \dots, N-1\}, \quad (1.36)$$

where $L \in \mathbb{N}$ defines the order of the linear estimator, and $c \in \mathbb{R}^L$ is a set of *forward* LP coefficients. $e \in \mathbb{R}^L$ is a set of parameters, often called the innovations, which account of error in the LP model. A datapoint can also be described by a linear combination of a certain number of subsequent points, using the set of *backward* LP coefficients $b \in \mathbb{R}^L$:

$$y_n = \sum_{l=1}^L b_l y_{n+l} + e_l \quad \forall n \in \{0, 1, \dots, N-L-1\}. \quad (1.37)$$

Determining the LP coefficients enables the estimation of FID values beyond the data actually acquired ($n < 0$ and $n > N-1$). It should be noted that Equations 1.36 and 1.37 are only valid for FIDs without any corruption from experimental noise. Noisy datasets are instead an example of an autoregressive moving average (ARMA) process. Despite this, due to the greater simplicity of the AR model, it is far more common to employ this. The most common means of performing LP is by solving the Yule-Walker equations[43, 44], which describe the relationship between the signal autocorrelation coefficients and the LP coefficients[42: Section 3.3], with the Levinson-Durbin

^{§§}The process of estimating the parameters describing a particular NMR dataset is often referred to as *deconvolution*. The term is not used here as it is a misnomer; deconvolution is simply the inverse operation of convolution and has nothing to do with estimating the optimal set of parameters which describe a signal.

algorithm providing an efficient means of solving the equations[45, 46].

SVD-based methods

LPSVD: also called SVD-Prony Include ESPRIT Some of the most prevalent methods for NMR estimation involve *singular value decomposition* (SVD) as a key component (Appendix B.1.1). For each of these methods, a truncated SVD is applied to a Hankel matrix containing the FID. Above a certain SNR threshold, it is found that the singular values in the data matrix associated with signal are larger than those related to noise, and as such forming a low-rank approximation of the data matrix nullifies contributions from noise in the parameter estimate. The degree of truncation is dependent on the suspected model order of the dataset, which must be supplied, or determined by some appropriate metric. All the SVD-based techniques determining the non-linear parameters in Equation 2.1b (frequencies and damping factors, incorporated in the signal poles). Subsequently, the amplitudes and phases can be computed by solving a linear least-squares problem:

$$\boldsymbol{\alpha} = \mathbf{Z}^+ \mathbf{y}, \quad (1.38a)$$

$$\mathbf{Z} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ z_1^1 & z_2^1 & \cdots & z_M^1 \\ \vdots & \vdots & \ddots & \vdots \\ z_1^{N-1} & z_2^{N-1} & \cdots & z_M^{N-1} \end{bmatrix}. \quad (1.38b)$$

Both linear prediction singular value decomposition (LPSVD)[47, 48] and Hankel singular value decomposition (HSVD)[49] have been used extensively in the field of *in vivo* magnetic resonance spectroscopy (MRS) after introduction by Barkhuijsen and co-workers[50–54]. With LPSVD, the signal poles are determined by finding the roots of a polynomial which features backward LP coefficients. HSVD is based on the “state-space” principle, which generated interest in part because it meant that the computationally costly polynomial rooting step in LPSVD was avoided. A further technique in this category, the matrix pencil method (MPM)[55–57] is based on solving a generalised eigenvalue problem. Pines and coworkers introduced the information theoretic matrix pencil method (ITMPM)[58], for use in NMR, which combines the MPM with a method for estimating the model order, such that no *a priori* information is required. Recently, the MPM has been employed for the analysis of NMR relaxometry data[59, 60]. Outlines of the MPM and its 2D equivalent, the modified matrix enhancement and matrix pencil method (MMEMPPM)[61, 62] are provided in detail in Section 2.2.1.

Iterative Methods

For problems involving nonlinear function fitting, the most widespread approach are iterative techniques. The variable projection (VARPRO) method[63] is such an example, which has been applied extensively in the field of MRS, following its introduction by van der Veen and co-workers[64, 65]. VARPRO relies on minimising the following quantity:

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^{4M}} \|\mathbf{y} - \mathbf{x}(\boldsymbol{\theta})\|^2 \equiv \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^{4M}} \|\mathbf{y} - \mathbf{Z}\boldsymbol{\alpha}\|^2 \quad (1.39)$$

The linear parameters, residing in the vector of complex amplitudes, can be recast as $\mathbf{Z}^+ \mathbf{y}$ (see Equation 1.38). By doing this, it becomes possible to recast the optimisation problem to only solve for the nonlinear terms:

$$[\mathbf{f}^T \hat{\boldsymbol{\eta}}^T]^T = \arg \min_{[\mathbf{f}^T \hat{\boldsymbol{\eta}}^T]^T \in \mathbb{R}^{2M}} \|\mathbf{y} - \mathbf{Z}\mathbf{Z}^+ \mathbf{y}\|^2. \quad (1.40)$$

The Levenberg-Marquardt (LM) algorithm[66, 67] is typically employed to perform the optimisation routine, with amplitudes and phases determined with linear least squares afterwards.

For such a routine to perform, a large amount of *a priori* information needs to be provided, in the form of a set of estimated frequencies and damping factors in the data. Further specifications can be applied to the algorithm, reflecting the spectroscopist's knowledge of the signal under inspection. For example, damping factors corresponding to signals that form a particular multiplet can be constrained to be equal. Advanced method for accurate, robust, and efficient spectral fitting (AMARES) is an improvement of VARPRO, which facilitates a wider variety of linear constraints to be imposed on the parameter set[68], and is now established as one of the most prominent means of quantifying MRS signals.

There is not much precedent for using iterative schemes for NMR analysis, despite its widespread adoption in MRS. One reason for this may be linked to the requirement to supply a large amount of prior knowledge; in MRS, it is likely that the practitioner has an intimate understanding of the expected signals which will be present, since all signals will derive from species which make up the human metabolome, whose chemical shifts and multiplet structures will be well-established. Over many experiments, little adjustment to the information provided to the optimiser will need to be given, making the process facile. In NMR experiments, there is a far greater variation in the appearance of the data, such that the process of defining *a priori* information for each dataset of interest may become time-consuming and cumbersome.

Bayesian Methods

TODO complete reduction to amplitude frequency table (CRAFT)[69] 2D[70] perspective[71]

Machine Learning Methods

In recent years, the application of deep learning approaches for estimation have been introduced, with the currently available methods focussing on spectral data[72, 73] **Mention deconvolution networks.**

THEORY

2

Describe what is to come in this chapter... This chapter provides a detailed outline of an estimation routine which has been developed for the estimation of NMR data.

2.1 Outline of the Problem

For the purposes of this work, it is always assumed that an FID to be estimated $\mathbf{Y} \in \mathbb{C}^{N^{(1)} \times \dots \times N^{(D)}}$ is hypercomplex in form, meaning that it obeys Equation 1.22 with $\zeta^{(d)} = \exp(i\cdot)$ $\forall d \in \{1, \dots, D\}$:

$$\mathbf{y}_{n^{(1)}, \dots, n^{(D)}} = \mathbf{x}_{n^{(1)}, \dots, n^{(D)}}(\boldsymbol{\theta}) + \mathbf{w}_{n^{(1)}, \dots, n^{(D)}}, \quad (2.1a)$$

$$\mathbf{x}_{n^{(1)}, \dots, n^{(D)}}(\boldsymbol{\theta}) = \sum_{m=1}^M \alpha_m \exp(i\phi_m) \prod_{d=1}^D \exp\left(\left(2\pi i \left(f_m^{(d)} - f_{\text{off}}^{(d)}\right) - \eta_m\right) n^{(d)} \Delta_t^{(d)}\right), \quad (2.1b)$$

$$\mathbf{w}_{n^{(1)}, \dots, n^{(D)}} \sim \mathcal{N}_{\mathcal{C}}(0, 2\sigma^2), \quad (2.1c)$$

where $\Delta_t^{(d)} = 1/f_{\text{sw}}^{(d)}$. Under this model, it is assumed that an FID consists of a summation of M damped complex sinusoids in the presence in AWGN. It is the goal of parametric estimation to establish the identity of all the quantities which describe the model component \mathbf{X} , which are distilled into the vector $\boldsymbol{\theta} \in \mathbb{R}^{2(D+2)M}$, given by Equation 1.21. Equation 2.1b can be expressed in terms of complex amplitudes and signals poles as follows:

$$\mathbf{x}_{n^{(1)}, \dots, n^{(D)}}(\boldsymbol{\theta}) = \sum_{m=1}^M \alpha_m \prod_{d=1}^D z_m^{(d)} n^{(d)}, \quad (2.2a)$$

$$\alpha_m = \alpha_m \exp(i\phi_m), \quad (2.2b)$$

$$z_m^{(d)} = \exp\left(\left(2\pi i \left(f_m^{(d)} - f_{\text{off}}^{(d)}\right) - \eta_m^{(d)}\right) \Delta_t^{(d)}\right). \quad (2.2c)$$

Due to the assumed AWGN nature of the noise array, the pdf of an individual noise component is

$$p(w_{n^{(1)}, \dots, n^{(D)}}) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{|w_{n^{(1)}, \dots, n^{(D)}}|^2}{2\sigma^2}\right). \quad (2.3)$$

As the elements are independent and identically distributed, the joint pdf describing the entire noise array is given by the product of each element's pdf:

$$\begin{aligned} p(\mathbf{W}) &= \prod_{n^{(1)}=0}^{N^{(1)}-1} \cdots \prod_{n^{(D)}=0}^{N^{(D)}-1} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{|w_{n^{(1)}, \dots, n^{(D)}}|^2}{2\sigma^2}\right) \\ &= \frac{1}{(2\pi\sigma^2)^{\mathfrak{N}}} \exp\left(-\frac{\|\mathbf{W}\|^2}{2\sigma^2}\right) \end{aligned} \quad (2.4)$$

As the noise array is the difference between the data and model, the likelihood function of $\boldsymbol{\theta}$ given \mathbf{Y} , $\mathcal{L}(\boldsymbol{\theta}|\mathbf{Y})$, is given by

$$\mathcal{L}(\boldsymbol{\theta}|\mathbf{Y}) = \frac{1}{(2\pi\sigma^2)^{\mathfrak{N}}} \exp\left(-\frac{\|\mathbf{Y} - \mathbf{X}(\boldsymbol{\theta})\|^2}{2\sigma^2}\right). \quad (2.5)$$

It is common to consider instead the log-likelihood function of $\boldsymbol{\theta}$ given \mathbf{Y} , $\ell(\boldsymbol{\theta}|\mathbf{Y})$. As application of the logarithm is a monotonic transformation, the arguments of the maxima of \mathcal{L} and ℓ are equivalent.

$$\ell(\boldsymbol{\theta}|\mathbf{Y}) = -\mathfrak{N} \ln(2\pi\sigma^2) - \frac{\|\mathbf{Y} - \mathbf{X}(\boldsymbol{\theta})\|^2}{2\sigma^2}. \quad (2.6)$$

Equation 2.6 implies that the optimal set of parameters $\boldsymbol{\theta}^{(*)}$, often referred to as the *maximum likelihood estimate* (MLE), is that which minimises the (square) norm of the difference between the data and model

$$\boldsymbol{\theta}^{(*)} = \arg \max_{\boldsymbol{\theta} \in \mathbb{R}^{2(D+2)M}} \ell(\boldsymbol{\theta}|\mathbf{Y}) \equiv \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^{2(D+2)M}} \|\mathbf{Y} - \mathbf{X}(\boldsymbol{\theta})\|^2. \quad (2.7)$$

The application of *non-linear programming* (NLP) is a well-established approach to solve such a problem[74, 75]. The basic principle behind NLP is to iteratively explore, in a methodical way, how a function varies with its arguments. By using information about the function and optionally its derivatives, such a routine attempts to find a minimum in the function, and terminates once this has been achieved. While derivative-free approaches to NLP do exist, (**cite examples like simplex, simulated annealing, BOBYQA?**) in scenarios where the function under consideration has well-defined, computationally tractable derivatives, the use of these can be valuable to

solving optimisation problems; the problem outlined in Equation 2.7 is such an example.

In order for NLP to perform effectively, a large amount of *a priori* information is typically required, in the form of an initial guess $\boldsymbol{\theta}^{(0)}$. There is precedent for using NLP in the context of NMR data estimation, in which substantial user input defines the initial guess (see the description of VARPRO and AMARES in Section 1.3.4). As an alternative, a routine is presented here in which an initial guess is determined using an SVD-based technique (the MPM), requiring no or minimal user input. Subsequently, the result generated is subjected to NLP under the assumption that the data is phase-corrected, which will be shown to be effective in correcting spurious features in MPM result. In this sense, the NLP routine can be seen as a means of validating the MPM result, and correcting spurious features, if necessary.

2.2 Generating an Initial Guess: Matrix Pencil Method

Brief introductory text. Mention applications w. citations

2.2.1 1D Matrix Pencil Method

Remark 2. *In contexts where 1D datasets are considered specifically, the redundant dimension index⁽¹⁾ will be neglected for conciseness.*

The *matrix pencil method* (MPM), developed by Hua and Sarkar[55–57], provides a route to extracting the signal poles of a 1D dataset, based on the assumption that the number of oscillators M is known. To motivate how the MPM works, first consider a dataset which is devoid of noise, given by Equation 2.2 with $D = 1$:

$$x_n(\boldsymbol{\theta}) = \sum_{m=1}^M \alpha_m z_m^n, \quad (2.8)$$

Consider the Hankel matrix $\mathbf{H}_x \in \mathbb{C}^{(N-L) \times (L+1)}$:

$$\mathbf{H}_x = \begin{bmatrix} x_0 & x_1 & \cdots & x_L \\ x_1 & x_2 & \cdots & x_{L+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N-L-1} & x_{N-L} & \cdots & x_{N-1} \end{bmatrix}. \quad (2.9)$$

This matrix comprises windowed segments of the FID, with each row comprising the segment shifted to the right by one point relative to the row above. $L \in \mathbb{N}$ is the *pencil parameter*, which dictates the size of each window. From \mathbf{H}_x , two matrices are defined, \mathbf{H}_{x1} and \mathbf{H}_{x2} , formed by the

removal of the last or first column of \mathbf{H}_x , respectively:

$$\mathbf{H}_{x1} = \begin{bmatrix} x_0 & x_1 & \cdots & x_{L-1} \\ x_1 & x_2 & \cdots & x_L \\ \vdots & \vdots & \ddots & \vdots \\ x_{N-L-1} & x_{N-L} & \cdots & x_{N-2} \end{bmatrix}, \quad (2.10a)$$

$$\mathbf{H}_{x2} = \begin{bmatrix} x_1 & x_2 & \cdots & x_L \\ x_2 & x_3 & \cdots & x_{L+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N-L} & x_{N-L+1} & \cdots & x_{N-1} \end{bmatrix}. \quad (2.10b)$$

These matrices can be deconstructed into the following forms involving matrices containing the M signal poles and complex amplitudes that the data comprises:

$$\mathbf{H}_{x1} = \mathbf{Z}_L \mathbf{A} \mathbf{Z}_R, \quad (2.11a)$$

$$\mathbf{H}_{x2} = \mathbf{Z}_L \mathbf{A} \mathbf{Z}_D \mathbf{Z}_R, \quad (2.11b)$$

$$\mathbb{C}^{(N^{(1)} - L^{(1)}) \times M} \ni \mathbf{Z}_L = \begin{bmatrix} \mathbf{1} & \mathbf{z} & \mathbf{z}^2 & \cdots & \mathbf{z}^{N-L-1} \end{bmatrix}^T, \quad (2.11c)$$

$$\mathbb{C}^{M \times L} \ni \mathbf{Z}_R = \begin{bmatrix} \mathbf{1} & \mathbf{z} & \mathbf{z}^2 & \cdots & \mathbf{z}^{L-1} \end{bmatrix}, \quad (2.11d)$$

$$\mathbb{C}^{M \times M} \ni \mathbf{Z}_D = \text{diag}(\mathbf{z}), \quad (2.11e)$$

$$\mathbb{C}^{M \times M} \ni \mathbf{A} = \text{diag}(\boldsymbol{\alpha}), \quad (2.11f)$$

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_M \end{bmatrix}^T, \quad (2.11g)$$

$$\mathbf{z} = \begin{bmatrix} z_1 & z_2 & \cdots & z_M \end{bmatrix}^T. \quad (2.11h)$$

The *matrix pencil* $\mathbf{H}_{x2} - \lambda \mathbf{H}_{x1}$, with $\lambda \in \mathbb{C}$, can therefore be expressed as

$$\mathbf{H}_{x2} - \lambda \mathbf{H}_{x1} = \mathbf{Z}_L \mathbf{A} (\mathbf{Z}_D - \lambda \mathbf{I}_M) \mathbf{Z}_R, \quad (2.12)$$

where $\mathbf{I}_M \in \mathbb{C}^{M \times M}$ is the identity matrix. Assuming that the following condition is met:

$$M \leq L \leq N - M, \quad (2.13)$$

the rank of the matrix pencil will be M . Equation 2.13 must be obeyed to ensure that both the number of rows and columns of the matrix pencil are at least M . Now consider the case when the scalar λ is equal to one of the signal poles i.e. $\lambda = z_m \forall m \in \{1, \dots, M\}$. The element $[\mathbf{Z}_D - \lambda \mathbf{I}_M]_{m,m}$

will be set to 0, which will lead to the determinant of the matrix pencil being 0. The eigenvalues of the matrix pencil are the solution of the so-called *generalised eigenvalue problem*, and are defined as[76: Section 7.7]

$$\mathbf{z} = \{z \in \mathbb{C} : \det(\mathbf{H}_{x2} - z\mathbf{H}_{x1}) = 0\} \quad (2.14)$$

One means of finding the signal poles is by finding the eigenvalues of the matrix $\mathbf{H}_{x1}^+ \mathbf{H}_{x2}$. Deriving the corresponding complex amplitudes can then be achieved by solving the set of linear equations

$$\boldsymbol{\alpha} = \mathbf{Z}^+ \mathbf{x}, \quad (2.15a)$$

$$\mathbf{Z} = \begin{bmatrix} \mathbf{1} & \mathbf{z} & \mathbf{z}^2 & \dots & \mathbf{z}^{N-1} \end{bmatrix}^T. \quad (2.15b)$$

Extraction of the amplitudes, phases, frequencies, and damping factors from the signal poles and complex amplitudes can then take place:

$$\boldsymbol{\alpha} = |\boldsymbol{\alpha}|, \quad (2.16a)$$

$$\phi = \arctan\left(\frac{\Im(\boldsymbol{\alpha})}{\Re(\boldsymbol{\alpha})}\right), \quad (2.16b)$$

$$f = \frac{f_{sw}}{2\pi} \Im(\ln \mathbf{z}) + f_{off}, \quad (2.16c)$$

$$\eta = -f_{sw} \Re(\ln \mathbf{z}). \quad (2.16d)$$

Noisy data

The presence of noise in the signal complicates the process of determining the M signal poles, as $\mathbf{H}_y - \mathbf{H}_x$'s equivalent with elements replaced by the noisy data \mathbf{y} — is likely to be full-rank, given by $\min(N-L, L+1)$. To minimise the influence of noise on the estimated signal poles, it is necessary to generate a rank-reduced matrix $\tilde{\mathbf{H}}_y$. By employing the Eckart-Young-Mirsky (EYM) theorem[76: Section 2.2], an appropriate matrix is can be obtained through SVD (see Appendix B.1.1):

$$\tilde{\mathbf{H}}_y = \mathbf{U}_M \boldsymbol{\Sigma}_M \mathbf{V}_M^\dagger, \quad (2.17a)$$

$$\mathbb{C}^{(N^{(1)}-L^{(1)}) \times M} \ni \mathbf{U}_M = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_M \end{bmatrix}, \quad (2.17b)$$

$$\mathbb{C}^{(L^{(1)}+1) \times M} \ni \mathbf{V}_M = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_M \end{bmatrix}, \quad (2.17c)$$

$$\mathbb{C}^{M \times M} \ni \boldsymbol{\Sigma}_M = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_M). \quad (2.17d)$$

σ_m is the m^{th} largest singular value of \mathbf{H}_y ; $\mathbf{u}_m \in \mathbb{C}^{N-L}$ and $\mathbf{v}_m \in \mathbb{C}^{L+1}$ are the corresponding left and right singular vectors, respectively. The EYM proves that $\tilde{\mathbf{H}}_y$ is the closest matrix of rank M

to \mathbf{H}_y in a Frobenius norm sense, i.e.

$$\tilde{\mathbf{H}}_y = \arg \min_{\mathbf{A}: \text{rank}(\mathbf{A})=M} \|\mathbf{A} - \mathbf{H}_y\|. \quad (2.18)$$

With a rank-reduced matrix produced from the noisy matrix, the signal poles can then be derived from the eigenvalues of $\tilde{\mathbf{H}}_{y1}^+ \tilde{\mathbf{H}}_{y2}$, where $\tilde{\mathbf{H}}_{y1}$ and $\tilde{\mathbf{H}}_{y2}$ have the same relation to $\tilde{\mathbf{H}}_y$ as \mathbf{H}_{x1} and \mathbf{H}_{x2} do to \mathbf{H}_x . As a less expensive alternative, the same result can be achieved by computing the eigenvalues of $\mathbf{V}_{M1}^+ \mathbf{V}_{M2}$, with

$$\mathbf{V}_{M1} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \dots \ \mathbf{v}_{M-1}], \quad (2.19a)$$

$$\mathbf{V}_{M2} = [\mathbf{v}_2 \ \mathbf{v}_3 \ \dots \ \mathbf{v}_M]. \quad (2.19b)$$

Algorithm 2.1 provides a pseudo-code description of the MPM, while Listing C.3 outlines a `PYTHON` implementation of it. For optimal results, the pencil parameter should adhere to $\lfloor N/3 \rfloor \leq L \leq \lfloor 2N/3 \rfloor$ [55]. In this work, $\lfloor N/3 \rfloor$ is always used, primarily since the computational complexity of the method is at a maximum when $L = N/2$ ^{*}

2.2.2 2D Matrix Enhancement and Matrix Pencil Method

The MPM was extended for the consideration of 2D data by Hua with the *matrix enhancement and matrix pencil method* (MEMPM)[61]. The method centers around the enhanced matrix $\mathbf{E}_Y \in \mathbb{C}^{(L^{(1)}L^{(2)}) \times (N^{(1)}-L^{(1)}+1)(N^{(2)}-L^{(2)}+1)}$, a block Hankel matrix of the form

$$\mathbf{E}_Y = \begin{bmatrix} \mathbf{H}_{Y,0} & \mathbf{H}_{Y,1} & \cdots & \mathbf{H}_{Y,N^{(1)}-L^{(1)}} \\ \mathbf{H}_{Y,1} & \mathbf{H}_{Y,2} & \cdots & \mathbf{H}_{Y,N^{(1)}-L^{(1)}+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}_{Y,L^{(1)}-1} & \mathbf{H}_{Y,L^{(1)}} & \cdots & \mathbf{H}_{Y,N^{(1)}-1} \end{bmatrix}, \quad (2.20a)$$

$$\mathbf{H}_{Y,n^{(1)}} = \begin{bmatrix} \gamma_{n^{(1)},0} & \gamma_{n^{(1)},1} & \cdots & \gamma_{n^{(1)},N^{(2)}-L^{(2)}} \\ \gamma_{n^{(1)},1} & \gamma_{n^{(1)},2} & \cdots & \gamma_{n^{(1)},N^{(2)}-L^{(2)}+1} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{n^{(1)},L^{(2)}-1} & \gamma_{n^{(1)},L^{(2)}} & \cdots & \gamma_{n^{(1)},N^{(2)}-1} \end{bmatrix}. \quad (2.20b)$$

*With $L = \lfloor N/2 \rfloor$, the matrix \mathbf{H}_Y is at its most “square” i.e. the number of rows and columns are at their most similar. Matrices which are more square will increase the demands on computing the SVD, with a complexity $\mathcal{O}(\min(L+1, N-L+1)^2 \times \max(L+1, N-L+1))$.

ALGORITHM 2.1 The MPM, with the optional prediction of model order using the MDL, if M is set to 0.

```

1: procedure MPM( $\mathbf{y} \in \mathbb{C}^N, f_{\text{sw}} \in \mathbb{R}_{>0}, f_{\text{off}} \in \mathbb{R}, M \in \mathbb{N}_0$ )
2:    $L \leftarrow \lfloor N/3 \rfloor;$ 
3:    $\mathbf{H}_y \leftarrow \begin{bmatrix} y_0 & y_1 & \cdots & y_L \\ y_1 & y_2 & \cdots & y_{L+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N-L-1} & y_{N-L} & \cdots & y_{N-1} \end{bmatrix};$ 
4:    $\mathbf{U}, \sigma, \mathbf{V}^\dagger \leftarrow \text{SVD}(\mathbf{H}_y);$ 
5:   if  $M = 0$  then
6:      $M \leftarrow \text{MDL}(\sigma, L, N);$ 
7:   end if
8:    $\mathbf{V} \leftarrow [\mathbf{V}^\dagger]^\dagger;$ 
9:    $\mathbf{V}_M \leftarrow \mathbf{V}[:, :M];$  ▷ Retain first  $M$  right singular vectors.
10:   $\mathbf{V}_{M1}, \mathbf{V}_{M2} \leftarrow \mathbf{V}_M[:, :M-1], \mathbf{V}_M[:, 1:];$  ▷ Remove last/first column
11:   $\mathbf{z} \leftarrow \text{EIGENVALUES}(\mathbf{V}_{M1}^* \mathbf{V}_{M2});$ 
12:   $\mathbf{Z} \leftarrow [\mathbf{1} \ \mathbf{z} \ \mathbf{z}^2 \ \cdots \ \mathbf{z}^N]^T;$ 
13:   $\alpha \leftarrow \mathbf{Z}^* \mathbf{Y};$ 
14:   $a \leftarrow |\alpha|;$ 
15:   $\phi \leftarrow \arctan\left(\frac{\Im(\alpha)}{\Re(\alpha)}\right);$ 
16:   $f \leftarrow \frac{f_{\text{sw}}}{2\pi} \Im(\ln \mathbf{z}) + f_{\text{off}};$ 
17:   $\eta \leftarrow -f_{\text{sw}} \Re(\ln \mathbf{z});$ 
18:  if  $\eta$  contains negative values then ▷ Purge any oscillators with negative damping
19:    Remove these from  $\eta$ , and remove the corresponding values from  $\alpha, \phi$ , and  $f$ ;
20:  end if
21:   $\theta^{(0)} \leftarrow [\alpha^T \ \phi^T \ f^T \ \eta^T]^T;$ 
22:  return  $\theta^{(0)};$ 
23: end procedure

24: procedure MDL( $\sigma \in \mathbb{R}^{L+1}, L \in \mathbb{N}, N \in \mathbb{N}$ )
25:    $\text{MDL}_{k-1} \leftarrow \infty$  ▷ Ensure that on first iteration ( $k = 0$ ), the conditional does not hold
26:   for  $k = 0, \dots, L$  do
27:      $\text{MDL}_k \leftarrow -\ln\left(\frac{\prod_{r=k}^{L-1} \sigma_{r+1}^{1/(L-k)}}{\sum_{r=k}^{L-1} \sigma_{r+1}}\right)^{(L-k)N};$ 
28:     if  $\text{MDL}_k > \text{MDL}_{k-1}$  then
29:        $M \leftarrow k - 1;$ 
30:       break;
31:     else
32:        $\text{MDL}_{k-1} \leftarrow \text{MDL}_k;$ 
33:     end if
34:   end for
35:   return  $M;$ 
36: end procedure

```

In a similar fashion to Equation 2.11, $\mathbf{H}_{X,n^{(1)}}$, the noiseless equivalent to $\mathbf{H}_{Y,n^{(1)}}$, can be expressed as

$$\mathbf{H}_{X,n^{(1)}} = \mathbf{Z}_L^{(2)} \mathbf{A} \mathbf{Z}_D^{(1)^n^{(1)}} \mathbf{Z}_R^{(2)}. \quad (2.21)$$

This then enables the noiseless enhanced matrix to be expressed as

$$\mathbf{E}_X = \mathbf{E}_L \mathbf{A} \mathbf{E}_R, \quad (2.22a)$$

$$\mathbb{C}^{L^{(1)} L^{(2)} \times M} \ni \mathbf{E}_L = \begin{bmatrix} \mathbf{Z}_L^{(2)} \\ \mathbf{Z}_L^{(2)} \mathbf{Z}_D^{(1)} \\ \vdots \\ \mathbf{Z}_L^{(2)} \mathbf{Z}_D^{(1)L^{(1)}-1} \end{bmatrix}, \quad (2.22b)$$

$$\mathbb{C}^{M \times (N^{(1)} - L^{(1)} + 1)(N^{(2)} - L^{(2)} + 1)} \ni \mathbf{E}_R = \begin{bmatrix} \mathbf{Z}_R^{(2)} & \mathbf{Z}_D^{(1)} \mathbf{Z}_R^{(2)} & \dots & \mathbf{Z}_D^{(1)^{N^{(1)} - L^{(1)}}} \mathbf{Z}_R^{(2)} \end{bmatrix}. \quad (2.22c)$$

As was the case in the 1D MPM, SVD can be utilised to generate a filtered matrix $\tilde{\mathbf{E}}_Y$ with its rank reduced to M , in accordance with the EYM theorem:

$$\tilde{\mathbf{E}}_Y = \mathbf{U}_M \boldsymbol{\Sigma}_M \mathbf{V}_M^\dagger \quad (2.23)$$

If the conditions $N^{(d)} - L^{(d)} + 1 \geq M \forall d \in \{1, 2\}$ are met, $\text{range}(\mathbf{U}_M) = \text{range}(\mathbf{E}_L)$. This implies that there is some nonsingular matrix $\mathbf{T} \in \mathbb{C}^{M \times M}$ such that

$$\mathbf{U}_M = \mathbf{E}_L \mathbf{T}. \quad (2.24)$$

Now consider the following two matrices:

$$\mathbf{U}_{M1} = \mathbf{E}_{L1} \mathbf{T}, \quad (2.25a)$$

$$\mathbf{U}_{M2} = \mathbf{E}_{L1} \mathbf{Z}_D^{(1)} \mathbf{T}, \quad (2.25b)$$

$$\mathbb{C}^{L^{(1)}(L^{(2)}-1) \times M} \ni \mathbf{E}_{L1} = \begin{bmatrix} \mathbf{Z}_L^{(2)} \\ \mathbf{Z}_L^{(2)} \mathbf{Z}_D^{(1)} \\ \vdots \\ \mathbf{Z}_L^{(2)} \mathbf{Z}_D^{(1)L^{(1)}-2} \end{bmatrix}. \quad (2.25c)$$

\mathbf{U}_{M1} and \mathbf{U}_{M2} correspond to \mathbf{U}_M with the last and first $L^{(2)}$ rows removed, respectively. The matrix pencil for \mathbf{U}_{M1} and \mathbf{U}_{M2} can be expressed as

$$\mathbf{U}_{M1} - \lambda \mathbf{U}_{M2} = \mathbf{E}_{L1} \left(\mathbf{Z}_D^{(1)} - \lambda \mathbf{I}_M \right) \mathbf{T}. \quad (2.26)$$

As seen previously, this matrix structure implies that the elements of $\mathbf{z}^{(1)}$ are the solutions to the generalised eigenvalue problem, such that they are the eigenvalues of $\mathbf{U}_{M1}^+ \mathbf{U}_{M2}$.

To extract the signal poles in the other dimension, $\mathbf{z}^{(2)}$, the permutation matrix is defined:

$$\mathbb{R}^{L^{(1)}L^{(2)} \times L^{(1)}L^{(2)}} \ni \mathbf{P} = \begin{bmatrix} \mathbf{e}(1)^T \\ \mathbf{e}(L^{(2)} + 1)^T \\ \vdots \\ \mathbf{e}\left(1 + (L^{(1)} - 1)L^{(2)}\right)^T \\ \mathbf{e}(2)^T \\ \mathbf{e}(2 + L^{(2)})^T \\ \vdots \\ \vdots \\ \mathbf{e}(L^{(2)})^T \\ \mathbf{e}(2L^{(2)})^T \\ \vdots \\ \mathbf{e}(L^{(1)}L^{(2)})^T \end{bmatrix}. \quad (2.27)$$

$\mathbf{e}(i) \in \mathbb{R}^{L^{(1)}L^{(2)}}$ corresponds to a unit vector comprising zeros except for $e_i = 1 \forall i \in \{1, \dots, L^{(1)}L^{(2)}\}$.

Multiplying \mathbf{E}_L by the permutation matrix leads to a matrix in which the roles of the two sets of signal poles are effectively swapped:

$$\mathbf{E}_{LP} := \mathbf{P}\mathbf{E}_L = \begin{bmatrix} \mathbf{Z}_L^{(1)} \\ \mathbf{Z}_L^{(1)}\mathbf{Z}_D^{(2)} \\ \vdots \\ \mathbf{Z}_L^{(1)}\mathbf{Z}_D^{(2)L^{(2)}-1} \end{bmatrix}. \quad (2.28)$$

Note the similarity of Equation 2.28 with Equation 2.22b, which implies that with the same reasoning as given above, $\mathbf{z}^{(2)}$ can be derived by extracting the eigenvalues of $\mathbf{U}_{MP1}^+ \mathbf{U}_{MP2}$, where \mathbf{U}_{MP1} and \mathbf{U}_{MP2} correspond to $\mathbf{P}\mathbf{U}_M$ with the last and first $L^{(1)}$ rows removed, respectively.

In the original account on the MEMPM, the final stage involved employing a pairing algorithm in order to assign the uncorrelated signal poles in $\mathbf{z}^{(1)}$ with $\mathbf{z}^{(2)}$ [61]. The *modified* MEMPM (MMEMPM) was developed in order to overcome two issues with the pairing algorithm: (a) it is

computationally expensive (b) it is prone to return incorrect pairings[62]. As well as the eigenvalues of $\mathbf{U}_{M1}^+ \mathbf{U}_{M2}$, the MMEMPM requires extraction of the eigenvectors too, which are contained in the matrix $\mathbf{W}^{(1)}$. Assuming that there are no repeated poles in $\mathbf{z}^{(1)}$, the correctly paired second set of poles is then generated via

$$\mathbf{z}^{(2)} = \text{diag}(\mathbf{W}^{-1} \mathbf{U}_{MP1}^+ \mathbf{U}_{MP2} \mathbf{W}) \quad (2.29)$$

Talk about case of paired eigenvalues.

See Algorithm B.1 for a pseudo-code outline, and Listing C.4 for a `PYTHON` implementation of the MMEMPM.

2.2.3 Model Order Selection

The MPM and MMEMPM operate under the assumption that the model order M is known, or at least has been predicted. It is possible that an individual inspecting the FID's spectrum could predict M based on the number of peaks visible, however subjective means of predicting model order are typically viewed as disadvantageous as they have bias associated with them. There are various non-subjective criteria which have been established for estimating the model order of a given signal, with probably the two most prominent being the Akaike information criterion (AIC)[77] and minimum description length (MDL)[78, 79]. Both of these consider a family of potential models which describe a given set of observations, parametrised by the vector $\boldsymbol{\theta}$. For the purpose of 1D FID estimation, the family of potential models comprise Equation 2.1b, with variable M . Both the AIC and MDL take the same general form:

$$\mathcal{C}(k) = -c \ln(\mathcal{L}(\boldsymbol{\theta}^{(*)} | \mathbf{y})) + \mathcal{P}(k) \text{ with } \boldsymbol{\theta}^{(*)} \in \mathbb{R}^{4k}, \quad (2.30)$$

$\forall k \in \{0, 1, \dots\}$. $\mathcal{L}(\boldsymbol{\theta}^{(*)} | \mathbf{y})$ is the likelihood function of a given model, with order k , at the MLE $\boldsymbol{\theta}^{(*)}$, $c \in \mathbb{R}_{>0}$ is a scaling constant, and \mathcal{P} is a penalising function, which acts to correct for bias. As the model order increases, the likelihood function at the MLE will increase in size, as a model with more parameters will be able to fit a given dataset more accurately. However, as the model order increases, there will become a point where practically all of the deterministic part of the signal has been incorporated into the model, and increasing the model order further leads to the model also accounting for noise. The penalising term, which is larger for higher k , is required in order to estimate a model order which is parsimonious. Wax and Kailath derived an expression for the

likelihood at the MLE for models comprising a summation of complex sinusoids[80][†]:

$$\mathcal{L}(\boldsymbol{\theta}^{(*)} \in \mathbb{R}^{4k} | \mathbf{y}) = \left(\frac{\prod_{r=k}^{L-1} \sigma_{r+1}^{1/L-k}}{\frac{1}{L-k} \sum_{r=k}^{L-1} \sigma_{r+1}} \right)^{(L-k)N}, \quad (2.31)$$

$\forall k \in \{0, 1, \dots, L-1\}$. $\sigma \in \mathbb{R}^{L^{(1)}}$ is the set of singular values of \mathbf{H}_y , in decreasing order. The forms of the AIC and MDL are given by

$$\text{AIC}(k) = -2 \ln (\mathcal{L}(\hat{\boldsymbol{\theta}} | \mathbf{y})) + 2k(2L - k), \quad (2.32a)$$

$$\text{MDL}(k) = -\ln (\mathcal{L}(\hat{\boldsymbol{\theta}} | \mathbf{y})) + \frac{1}{2}k(2L - k) \ln N. \quad (2.32b)$$

The AIC has been shown to be inconsistent in that it tends to overestimate the model order as the number of samples increases[80]. For this reason, the MDL has found greater favour in signal processing applications. As such, by default the estimation routine employed in this work utilises the MDL:

$$M = \arg \min_{k \in \mathbb{N}_0: k < L} \text{MDL}(k). \quad (2.33)$$

Applying the MDL for model order selection, and subsequently using the MPM for parameter estimation is the basis of the ITMPM[58]. Figure 2.1 illustrates the form of the MDL for three signals with equivalent underlying models, with $M = 7$, and noise instances with different variances. The first 14 singular values of \mathbf{H}_y are plotted in panel b, where it can be seen that beyond the first 7, which account for signal components, the subsequent singular values, decrease at a far slower rate. The noise subspace for FIDs with higher SNRs have singular values which are (a) smaller in magnitude and (b) more consistent, such that distinguishing the noise and signal subspaces is an easier task (cf. the yellow and red lines in panel b). As such, the MDL is more likely to provide a faithful estimate of the true number of components in the FID (panel c) when the SNR is higher.

Mention that multidimensional criteria are available, though not implemented in this work.

2.3 Non-linear programming for NMR estimation

Attention now turns to the application of *non-linear programming* (NLP) in for FID estimation, which in this work accepts an initial guess of parameters from the MPM, $\boldsymbol{\theta}^{(0)}$, and generates the final estimate, $\boldsymbol{\theta}^{(*)}$.

[†]The expression in original paper considers the eigenvalues of the covariance matrix for the signal, rather than the singular values of \mathbf{H}_y . These are equivalent however.

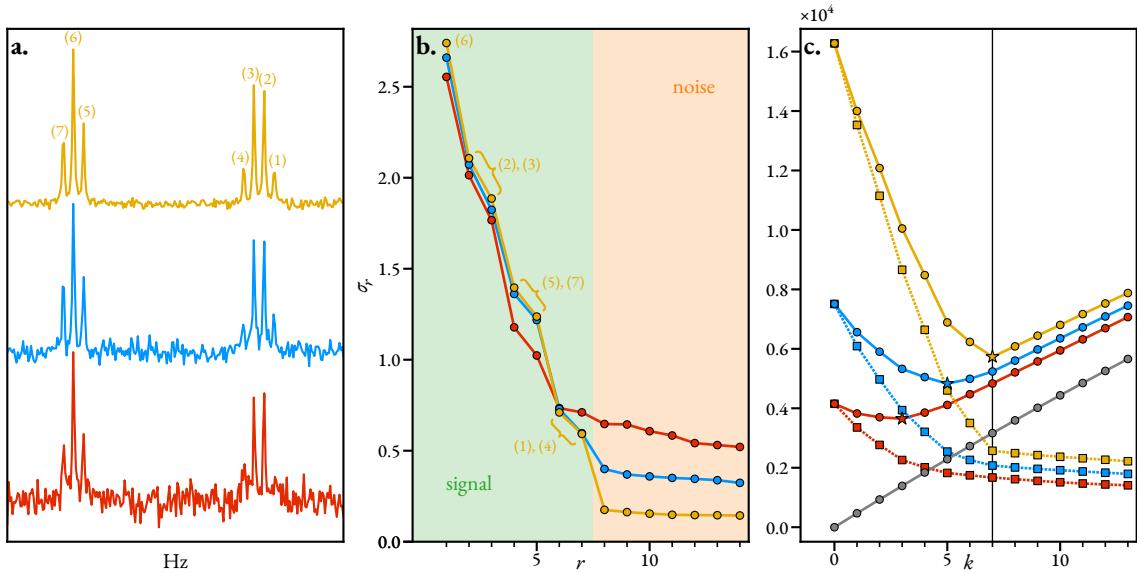


FIGURE 2.1: A visualisation of the behaviour of the MDL for three different FIDs comprising the same deterministic component (x) but with different noise instances of differing variances. The model used to construct the FIDs features 7 signals. The three SNRs used were 7 dB (red), 12 dB (blue), and 20 dB (yellow). The FIDs were generated with $N = 256$. **a.** Spectra of the three FIDs. **b.** The values of the 14 most significant singular values associated with the Hankel matrix \mathbf{H}_y , with the pencil parameter L set to $\lfloor N/3 \rfloor = 85$. **c.** Square points with dotted lines: The negative log-likelihood at the MLE, i.e. the first term of Equation 2.32b. Grey line: the penalty component of the MDL, given by the second term in Equation 2.32b. Circular points with solid lines: the MDL. Stars denote the minimum of the MDL for a given FID. The 20 dB signal is correctly deemed to have a model order of 7, while the other two are underestimated (predicted models orders are 5 and 3 for the 12 dB and 7 dB FIDs, respectively).

2.3.1 An overview of NLP

In an optimisation problem, the goal is to determine the minimum[‡] of a function $\mathcal{F}(\boldsymbol{\theta}) : \mathbb{R}^n \rightarrow \mathbb{R}$, $n \in \mathbb{N}$, often called the *cost function* or *fidelity*. This is typically with the goal of determining the argument $\boldsymbol{\theta}^{(*)}$ at which the optimum is found:

$$\boldsymbol{\theta}^{(*)} = \arg \min_{\boldsymbol{\theta} \in \mathbb{R}^n} \mathcal{F}(\boldsymbol{\theta}). \quad (2.34)$$

The above problem is *unconstrained*, as there are no limitations that the parameter vector is subjected to. Unless $\mathcal{F}(\boldsymbol{\theta})$ has particular properties, such as convexity[§], it is generally only possible to determine a *local minimum*, rather than a *global minimum*. $\boldsymbol{\theta}^{(*)}$ is a local minimiser of $\mathcal{F}(\boldsymbol{\theta})$ if there is a *neighbourhood* $V \ni \boldsymbol{\theta}^{(*)}$ for which

$$\mathcal{F}(\boldsymbol{\theta}^{(*)}) \leq \mathcal{F}(\boldsymbol{\theta}) \quad \forall \boldsymbol{\theta} \in V. \quad (2.35)$$

$V \subset \mathbb{R}^n$ is such that one can move some amount in any direction away from $\boldsymbol{\theta}^{(*)}$ and still be in V .

Key to NLP are the *necessary conditions*, which define whether a given vector $\boldsymbol{\theta}$ is a local minimum of \mathcal{F} . The *first necessary condition* states that if $\mathcal{F}(\boldsymbol{\theta})$ is continuously differentiable, and $\boldsymbol{\theta}^{(*)}$ is a local extremum of $\mathcal{F}(\boldsymbol{\theta})$, then the gradient vector $\mathbf{g}(\boldsymbol{\theta}^{(*)}) := \nabla \mathcal{F}(\boldsymbol{\theta}^{(*)})$ is the zero vector:

$$\mathbf{g}(\boldsymbol{\theta}^{(*)}) = \mathbf{0} \in \mathbb{R}^n \quad (2.36)$$

The *second necessary condition* subsequently states that if $\mathcal{F}(\boldsymbol{\theta})$ and $\mathbf{g}(\boldsymbol{\theta})$ are continuously differentiable, and $\boldsymbol{\theta}^{(*)}$ is a local minimiser of $\mathcal{F}(\boldsymbol{\theta})$, then the Hessian matrix $\mathbf{H}(\boldsymbol{\theta}^{(*)}) := \nabla^2 \mathcal{F}(\boldsymbol{\theta}^{(*)})$ is positive semidefinite, i.e.

$$\mathbf{v}^T \mathbf{H}(\boldsymbol{\theta}^{(*)}) \mathbf{v} \geq 0 \quad \forall \mathbf{v} \in \mathbb{R}^n. \quad (2.37)$$

Furthermore, it is a *unique* local minimiser if the *second-order sufficient condition* is also satisfied, i.e. that the Hessian is positive definite:

$$\mathbf{v}^T \mathbf{H}(\boldsymbol{\theta}^{(*)}) \mathbf{v} > 0 \quad \forall \mathbf{v} \in \mathbb{R}^n. \quad (2.38)$$

A plethora of approaches have been established to determine local minima of scalar functions. One of the better-known strategies is *Newton's method*, in which a quadratic approximation of the fidelity is considered. For a given iteration $k \in \mathbb{N}_0$, the fidelity is approximated using

$$\mathcal{F}_Q(\boldsymbol{\theta}) = \mathcal{F}(\boldsymbol{\theta}^{(k)}) + \mathbf{b}^T \mathbf{g}(\boldsymbol{\theta}^{(k)}) + \frac{1}{2} \mathbf{b}^T \mathbf{H}(\boldsymbol{\theta}^{(k)}) \mathbf{b}, \quad (2.39)$$

[‡]In certain applications, the interest is actually in finding the maximum of a function. However, it is trivial to transform a maximisation problem into a minimisation problem by finding the minimum of negative of the function.

[§]A convex function is one such that a line segment through any two points of the function lies above it.

where $\mathbf{b} = \boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}$. An updated prediction of the parameter vector is derived by finding the minimum of this quadratic approximation:

$$\begin{aligned}\frac{\partial \mathcal{F}(\boldsymbol{\theta})}{\partial \mathbf{b}} &= \mathbf{g}(\boldsymbol{\theta}^{(k)}) + \mathbf{H}(\boldsymbol{\theta}^{(k)}) \mathbf{b} \\ \implies 0 &= \mathbf{g}(\boldsymbol{\theta}^{(k)}) + \mathbf{H}(\boldsymbol{\theta}^{(k)}) (\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}) \\ \therefore \boldsymbol{\theta}^{(k+1)} &= \boldsymbol{\theta}^{(k)} - \mathbf{H}(\boldsymbol{\theta}^{(k)})^{-1} \mathbf{g}(\boldsymbol{\theta}^{(k)}).\end{aligned}\quad (2.40)$$

This process is repeated, until the convergence criterion has been met:

$$\|\mathbf{g}(\boldsymbol{\theta}^{(k)})\| \leq \epsilon. \quad (2.41)$$

The threshold $\epsilon > 0$, can be tuned based on the desired accuracy of the result, and is usually several orders of magnitude less than 1. Equation 2.40 tends not to be used as the update formula in real optimisation problems. One of the major downsides of the Newton update is the possibility that it is not a minimising update if the Hessian is not positive definite. Two primary strategies have emerged which are typically used instead:

- *Line search methods*[75: Chapter 3] determine an appropriate direction $\mathbf{p}^{(k)}$ along which the updated parameter vector is sourced. After this, an appropriate step length $\alpha^{(k)}$ is determined — typically in an efficient, though not optimal manner; abiding by the *Wolfe conditions* is a common approach — leading to $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \alpha^{(k)} \mathbf{p}^{(k)}$.
- *Trust region methods*[75: Chapter 4] define a radius $\Delta^{(k)} > 0$, and determine the minimum of Equation 2.39 subject to the constraint that $\|\mathbf{b}\| \leq \Delta^{(k)}$.

A trust region method is applied in this work, and as such further consideration of it will now be made.

Trust Region Methods

The structure of a typical trust region method is presented in Algorithm 2.2 (ignoring Lines 19 to 22, which is a custom addition, see Section 2.3.6). An initial radius for the trust region $\Delta^{(0)}$ is defined, along with a maximum permitted radius Δ_{\max} , to ensure that excessively adventurous steps do not take place. For each iteration k , a solution to the following sub-problem is sought:

$$\begin{aligned}\mathbf{p}^{(k)} = \arg \min_{\mathbf{p} \in \mathbb{R}^n} \mathcal{F}(\boldsymbol{\theta}^{(k)}) + (\boldsymbol{\theta}^{(k)} + \mathbf{p})^T \mathbf{g}(\boldsymbol{\theta}^{(k)}) + \frac{1}{2} (\boldsymbol{\theta}^{(k)} + \mathbf{p})^T \mathbf{H}(\boldsymbol{\theta}^{(k)}) (\boldsymbol{\theta}^{(k)} + \mathbf{p}) \\ \text{subject to } \|\mathbf{p}\| \leq \Delta^{(k)}.\end{aligned}\quad (2.42)$$

This sub-problem is not usually minimised exactly, but instead an efficient means of determining a sufficiently update is used. Common approaches include computing the Cauchy point,

ALGORITHM 2.2 Nonlinear programming routine employed in NMR-EsPy. This makes use of Algorithms 4.1 & 7.2 in [75], with a extra check inserted to deal with any negative-amplitude oscillators which may be generated as the routine evolves.

```

1: procedure NLP( $\mathbf{Y} \in \mathbb{C}^{N^{(1)} \times \dots \times N^{(D)}}$ ,  $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^{2(D+1)M}$ )
2:    $\Delta^{(0)} \leftarrow 1/10 \|\mathbf{g}(\boldsymbol{\theta}^{(0)} | \mathbf{Y})\|;$ 
3:    $\Delta_{\max} \leftarrow 16\Delta^{(0)};$ 
4:   for  $k = 0, 1, \dots$  do
5:      $\mathbf{p}^{(k)} \leftarrow \text{STEIHAUGTOINT}(\mathbf{Y}, \boldsymbol{\theta}^{(k)}, \Delta^{(k)})$ ; ▷ See Algorithm B.2
6:      $\rho^{(k)} \leftarrow \frac{\mathcal{F}_{\phi}(\boldsymbol{\theta}^{(k)}) - \mathcal{F}_{\phi}(\boldsymbol{\theta}^{(k)} + \mathbf{p}^{(k)})}{\mathcal{F}_{\phi Q}(\boldsymbol{\theta}^{(k)}) - \mathcal{F}_{\phi Q}(\boldsymbol{\theta}^{(k)} + \mathbf{p}^{(k)})};$ 
7:     if  $\rho_k < 1/4$  then
8:        $\Delta^{(k+1)} \leftarrow 1/4\Delta^{(k)};$ 
9:     else if  $\rho_k > 3/4$  and  $\|\mathbf{p}^{(k)}\| = \Delta^{(k)}$  then
10:       $\Delta^{(k+1)} \leftarrow \min(2\Delta^{(k)}, \Delta_{\max});$ 
11:    else
12:       $\Delta^{(k+1)} \leftarrow \Delta^{(k)};$ 
13:    end if
14:    if  $\rho^{(k)} > 3/20$  then
15:       $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + \mathbf{p}^{(k)};$ 
16:    else
17:       $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)};$ 
18:    end if
19:    if  $k \bmod 25 = 0$  and  $\boldsymbol{\theta}^{(k+1)}$  contains negative amplitudes then
20:       $\boldsymbol{\theta}^{(0)} \leftarrow \boldsymbol{\theta}^{(k+1)}$  with negative-amplitude oscillators removed;
21:       $\boldsymbol{\theta}^{(*)}, \epsilon^{(*)} \leftarrow \text{NLP}(\mathbf{Y}, \boldsymbol{\theta}^{(0)})$ ;
22:    end if
23:    if  $\|\mathbf{g}(\boldsymbol{\theta}^{(k+1)})\| < 10^{-8}$  then
24:      break;
25:    end if
26:  end for
27:   $\boldsymbol{\theta}^{(*)} \leftarrow \boldsymbol{\theta}^{(k+1)}$ 
28:   $\epsilon^{(*)} \leftarrow \sqrt{\frac{\mathcal{F}(\boldsymbol{\theta}^{(*)}) \text{diag}([\mathbf{H}(\boldsymbol{\theta}^{(*)})]^{-1})}{(N^{(1)} \dots N^{(D)}) - 1}}$ 
29:  return  $\boldsymbol{\theta}^{(*)}, \epsilon^{(*)}$ ;
30: end procedure

```

the Dogleg method, and a truncated conjugate-gradient approach commonly called the Steihaug-Toint (ST) method[75: Chapter 7]. The latter is employed in this work (see Algorithm B.2 and Listing C.5). In the ST approach, iterates of the conjugate-gradient method[75: Chapter 5] are computed, either until an iterate which is outside the trust region is computed, or negative curvature is discovered.

Once a provisional update $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \mathbf{p}^{(k)}$ is determined, a metric is considered which indicates how effectively the quadratic estimate at update agrees with the true value of the fidelity at this point:

$$\rho^{(k)} = \frac{\mathcal{F}(\boldsymbol{\theta}^{(k)}) - \mathcal{F}(\boldsymbol{\theta}^{(k)} + \mathbf{p}^{(k)})}{\mathcal{F}_Q(\boldsymbol{\theta}^{(k)}) - \mathcal{F}_Q(\boldsymbol{\theta}^{(k)} + \mathbf{p}^{(k)})}. \quad (2.43)$$

$\rho^{(k)}$ is the ratio between the actual reduction of the fidelity caused by taking the proposed step, and the predicted reduction based on the quadratic model. If $\rho^{(k)}$ is sufficiently close to 1, the quadratic model being used to generate new iterates is deemed to be acting well enough to warrant accepting the proposed update (Lines 14 and 15 of Algorithm 2.2). Furthermore, if $\rho^{(k)}$ is particularly close to 1, and the proposed update is at the boundary of the trust radius, it is appropriate to enlarge the radius of the trust region for the next iteration in an attempt to increase the rate of convergence (Lines 9 and 10). On the other hand, a small value of $\rho^{(k)}$ implies that the quadratic model reflects the true fidelity poorly, such that the proposed update should be rejected (Lines 16 and 17). As well as this, the trust region's radius should be decreased such that the model is more likely to behave faithfully (Lines 7 and 8). In general, the thresholds which dictate whether to accept an update, and whether to adjust the trust region radius are customisable. The hard-coded numerical values found in Algorithm 2.2 are the values used for the results acquired in this work.

2.3.2 Non-linear programming applied to FID estimation

Remark 3. *Prior to estimating the dataset, it is normalised, such that the signal actually under consideration is $\mathbf{Y}/\|\mathbf{Y}\|$. To make the result reflect the unnormalised dataset, the estimated amplitudes $\mathbf{a}^{(*)}$ are multiplied by $\|\mathbf{Y}\|$.*

Focus now turns to the specific problem of FID estimation using NLP, for which a general D -dimensional dataset will be considered. As established in Section 2.1, the fidelity $\mathcal{F}(\boldsymbol{\theta} | \mathbf{Y}) : \mathbb{C}^{N^{(1)} \times \dots \times N^{(D)}} \times \mathbb{R}^{2(1+D)M} \rightarrow \mathbb{R}$ is given by

$$\mathcal{F}(\boldsymbol{\theta} | \mathbf{Y}) = \|\mathbf{Y} - \mathbf{X}(\boldsymbol{\theta})\|^2. \quad (2.44)$$

The elements of the gradient vector $\mathbf{g}(\boldsymbol{\theta} | \mathbf{Y}) \in \mathbb{R}^{2(1+D)M}$ and the Hessian matrix $\mathbf{H}(\boldsymbol{\theta} | \mathbf{Y}) \in \mathbb{R}^{2(1+D)M \times 2(1+D)M}$ are derived by taking the first and second partial derivatives of the fidelity with

respect to the elements in θ , respectively:

$$g_i = -2\Re \left((\mathbf{Y} - \mathbf{X}), \frac{\partial \mathbf{X}}{\partial \theta_i} \right), \quad (2.45a)$$

$$b_{i,j} = 2\Re \left(\underbrace{\left(\frac{\partial \mathbf{X}}{\partial \theta_i}, \frac{\partial \mathbf{X}}{\partial \theta_j} \right)}_{(1)} - \underbrace{\left((\mathbf{Y} - \mathbf{X}), \frac{\partial^2 \mathbf{X}}{\partial \theta_i \partial \theta_j} \right)}_{(2)} \right), \quad (2.45b)$$

$\forall i, j \in \{1, \dots, 2(1+D)M\}$. The complete set of first and second derivatives of a particular element of the model $x := x_{n^{(1)}, \dots, n^{(D)}}$, given by Equation 2.1b, is as follows $\forall m \in \{1, \dots, M\}$, $\forall d, d' \in \{1, \dots, D\}$:

First derivatives

$$\frac{\partial x}{\partial \theta_m} \equiv \frac{\partial x}{\partial a_m} = \frac{x}{a_m}, \quad (2.46a)$$

$$\frac{\partial x}{\partial \theta_{m+M}} \equiv \frac{\partial x}{\partial \phi_m} = ix, \quad (2.46b)$$

$$\frac{\partial x}{\partial \theta_{m+(d+1)M}} \equiv \frac{\partial x}{\partial f_m^{(d)}} = 2\pi i \Delta_t^{(d)} n^{(d)} x, \quad (2.46c)$$

$$\frac{\partial x}{\partial \theta_{m+(d+D+1)M}} \equiv \frac{\partial x}{\partial \eta_m^{(d)}} = -\Delta_t^{(d)} n^{(d)} x. \quad (2.46d)$$

Second derivatives

$$\frac{\partial^2 x}{\partial \theta_m^2} \equiv \frac{\partial^2 x}{\partial a_m^2} = 0, \quad (2.47a)$$

$$\frac{\partial^2 x}{\partial \theta_m \partial \theta_{m+M}} \equiv \frac{\partial^2 x}{\partial a_m \partial \phi_m} = \frac{ix}{a_m}, \quad (2.47b)$$

$$\frac{\partial^2 x}{\partial \theta_m \partial \theta_{m+(d+1)M}} \equiv \frac{\partial^2 x}{\partial a_m \partial f_m^{(d)}} = \frac{2\pi i \Delta_t^{(d)} n^{(d)} x}{a_m}, \quad (2.47c)$$

$$\frac{\partial^2 x}{\partial \theta_m \partial \theta_{m+(d+D+1)M}} \equiv \frac{\partial^2 x}{\partial a_m \partial \eta_m^{(d)}} = \frac{-\Delta_t^{(d)} n^{(d)} x}{a_m}, \quad (2.47d)$$

$$\frac{\partial^2 x}{\partial \theta_{m+M}^2} \equiv \frac{\partial^2 x}{\partial \phi_m^2} = -x, \quad (2.47e)$$

$$\frac{\partial^2 x}{\partial \theta_{m+M} \partial \theta_{m+(d+1)M}} \equiv \frac{\partial^2 x}{\partial \phi_m \partial f_m^{(d)}} = -2\pi \Delta_t^{(d)} n^{(d)} x, \quad (2.47f)$$

$$\frac{\partial^2 x}{\partial \theta_{m+M} \partial \theta_{m+(d+D+1)M}} \equiv \frac{\partial^2 x}{\partial \phi_m \partial \eta_m^{(d)}} = -i \Delta_t^{(d)} n^{(d)} x, \quad (2.47g)$$

$$\frac{\partial^2 x}{\partial \theta_{m+(d+1)M} \partial \theta_{m+(d'+1)M}} \equiv \frac{\partial^2 x}{\partial f_m^{(d)} \partial f_m^{(d')}} = -4\pi^2 \left(\Delta_t^{(d)} n^{(d)} \right) \left(\Delta_t^{(d')} n^{(d')} \right) x, \quad (2.47h)$$

dimensions	# 1 st derivatives	# 2 nd derivatives
1	$4MN^{(1)}$	$9MN^{(1)}$
2	$6MN^{(1)}N^{(2)}$	$20MN^{(1)}N^{(2)}$
3	$8MN^{(1)}N^{(2)}N^{(3)}$	$35MN^{(1)}N^{(2)}N^{(3)}$
D	$2(1+D)MN^{(1)}\dots N^{(D)}$	$((1+D)(3+2D)-1)MN^{(1)}\dots N^{(D)}$

TABLE 2.I: The number of first and second derivatives that are necessary to compute both the gradient vector and Hessian matrix of the fidelity for 1- 2- and 3-dimensional datasets, as well as a general D -dimensional dataset.

$$\frac{\partial^2 x}{\partial \theta_{m+(d+1)M} \partial \theta_{m+(d'+D+1)M}} \equiv \frac{\partial^2 x}{\partial f_m^{(d)} \partial \eta_m^{(d')}} = -2\pi i \left(\Delta_t^{(d)} n^{(d)} \right) \left(\Delta_t^{(d')} n^{(d')} \right) x, \quad (2.47i)$$

$$\frac{\partial^2 x}{\partial \theta_{m+(d+D+1)M} \partial \theta_{m+(d'+D+1)M}} \equiv \frac{\partial^2 x}{\partial \eta_m^{(d)} \partial \eta_m^{(d')}} = \left(\Delta_t^{(d)} n^{(d)} \right) \left(\Delta_t^{(d')} n^{(d')} \right) x, \quad (2.47j)$$

$$\frac{\partial^2 x}{\partial \theta_i \partial \theta_j} = \frac{\partial^2 x}{\partial \theta_j \partial \theta_i}, \quad (2.47k)$$

$$\frac{\partial^2 x}{\partial \theta_i \partial \theta_j} = 0 \text{ if not specified above.} \quad (2.47l)$$

Equation 2.47l indicates that any second derivative with respect to two parameters which do not belong to the same oscillator will always be 0. This, along with the symmetrical nature of the second derivatives (Equation 2.47k) drastically reduces the required number to explicitly compute, from $4(1+D)^2 M^2$ per data-point to $(1+D)(3+2D)M$. Finally, Equation 2.47a indicates that another M second derivatives do not need to be computed, as they are 0 by definition. See Table 2.1 for the total number of derivatives that need to be computed for signals with different numbers of dimensions.

2.3.3 Approximating the Hessian

Despite many of the model second derivatives being 0, computation of those that are not zero, and subsequently using these the form the Hessian matrix, is often the most computationally expensive part of the optimisation. Numerous optimisation problems exist where this is the case, and as such there is considerable precedent for improving the efficiency of optimisation algorithms by generating approximations of the Hessian which are less expensive. Examples include the Gauss-Newton (GN) method and LM algorithm, which are specifically for residual sum-of-squares problems[75: Chapter 10], as well as quasi-Newton methods such as the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method[75: Chapter 6].

The GN and LM approaches replace the true Hessian matrix at each iteration with the fol-

lowing expression:

$$b_{i,j} = 2\Re \left\langle \frac{\partial \mathbf{X}}{\partial \theta_i}, \frac{\partial \mathbf{X}}{\partial \theta_j} \right\rangle, \quad (2.48)$$

i.e. term ② in Equation 2.45b, which involves the model second derivatives, is neglected. All that needs to be generated is the Jacobian $\mathbf{J} = \partial \mathbf{X} / \partial \theta$. This often brings a very large reduction in the computational cost, as no extra derivatives need to be computed for the Hessian at all, since the Jacobian is already required for generating the gradient vector. In situations where the residuals between the data and model are small, term ① will tend to dominate term ②, and as such these methods often enjoy a convergence rate which is comparable to that of Newton's method when close to local minima. Despite this, by invoking this approximation, the rate of convergence, i.e. the number of iterations required to reach $\theta^{(*)}$, tends to be adversely affected. See Section 2.3.5 for an example of this phenomenon.

2.3.4 Estimation Errors

A measure of the degree of uncertainty in the parameter estimates can be obtained by computing the *standard errors* associated with the NLP routine. Standard errors are related to the *observed Fisher information matrix* at convergence[81: Section 2.7]:

$$\epsilon(\theta^{(*)}) = \sqrt{\text{diag}(\mathbf{I}(\theta^{(*)})^{-1})}, \quad (2.49)$$

where the observed Fisher Information matrix contains the negative partial second derivatives of the log-likelihood with respect to θ :

$$\mathbf{I}(\theta)_{i,j} = -\frac{\partial^2 \ell(\theta | \mathbf{Y})}{\partial \theta_i \partial \theta_j}. \quad (2.50)$$

Recalling the form of the log-likelihood given by Equation 2.6, the elements of $\mathbf{I}(\theta)$ are

$$\mathbf{I}(\theta)_{i,j} = -\frac{1}{\sigma^2} \Re \left(\left\langle \frac{\partial \mathbf{X}}{\partial \theta_i}, \frac{\partial \mathbf{X}}{\partial \theta_j} \right\rangle - \left\langle (\mathbf{Y} - \mathbf{X}), \frac{\partial^2 \mathbf{X}}{\partial \theta_i \partial \theta_j} \right\rangle \right), \quad (2.51)$$

which very closely resembles the Hessian of θ :

$$\mathbf{I}(\theta)_{i,j} = \frac{1}{2\sigma^2} (\mathbf{H}(\theta)_{i,j}). \quad (2.52)$$

The standard errors therefore take the form

$$\epsilon(\theta^{(*)}) = \sqrt{2\sigma^2 \text{diag}(\mathbf{H}(\theta^{(*)})^{-1})}. \quad (2.53)$$

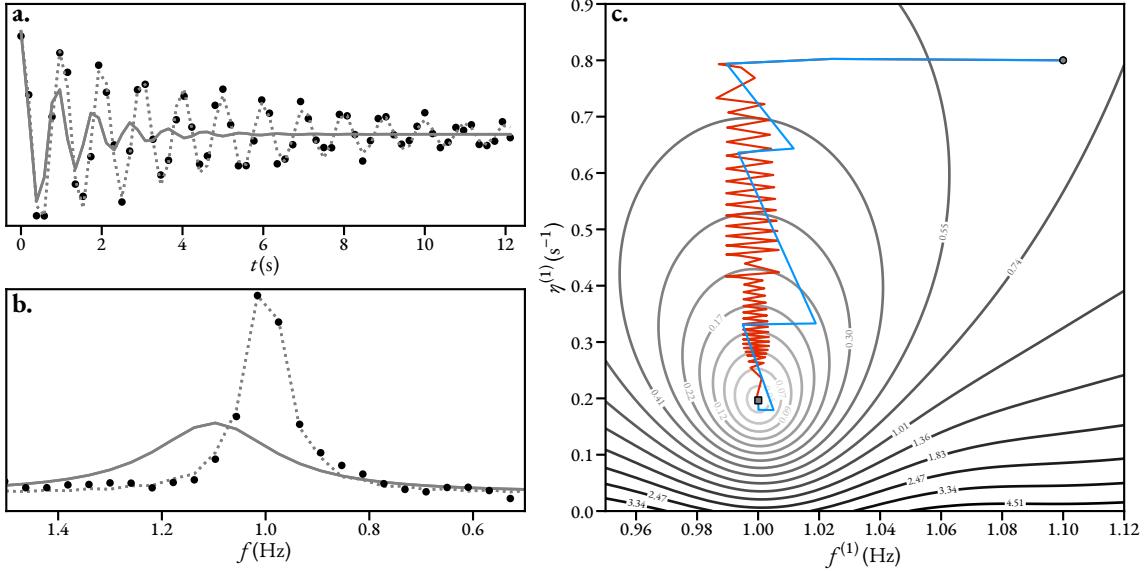


FIGURE 2.2: A visualisation of the trajectory of a 2-parameter optimisation involving a simulated FID comprising a single signal. **a.** & **b.** Representations of the signal in the time domain and Fourier domain, respectively. Black dots: the signal to be estimated \mathbf{y} . Solid grey line: the model generated using the initial guess $\mathbf{x}(\boldsymbol{\theta}^{(0)})$. Dotted grey line: the model generated using the optimised result, $\mathbf{x}(\boldsymbol{\theta}^{(*)})$. **c.** A contour plot of the fidelity. Blue line: the trajectory of the parameter vector with the true Hessian matrix used in computing each update. Red line: the analogous trajectory using the Hessian approximation in place of the true Hessian.

The mean and variance of the noise are 0 and $2\sigma^2$, respectively, leading to:

$$2\sigma^2 = \frac{1}{\mathfrak{N}-1} \|\mathbf{W}\|^2 = \frac{1}{\mathfrak{N}-1} \|\mathbf{Y} - \mathbf{X}(\boldsymbol{\theta}^{(*)})\|^2. \quad (2.54)$$

Finally a useable expression for the standard errors is arrived at:

$$\epsilon(\boldsymbol{\theta}^{(*)}) = \sqrt{\frac{\mathcal{F}(\boldsymbol{\theta}^{(*)}) \text{diag}(\mathbf{H}(\boldsymbol{\theta}^{(*)})^{-1})}{\mathfrak{N}-1}} \quad (2.55)$$

2.3.5 Visualisation of a simple example

Figure 2.2 provides a visual example of the application of NLP to estimate a simulated 1D FID comprising a single signal. The FID was constructed using Equation 1.22 with $M = 1$, $N = 64$, $f_{\text{sw}} = 5.2$ Hz ($\Delta_t \approx 0.192$ s), and $f_{\text{off}} = 0$ Hz. The signal was parameterised by $\boldsymbol{\theta} \in \mathbb{R}^4$ comprising $\alpha = 1$, $\phi = 0$ rad, $f = 1$ Hz, $\eta = 0.2$ s $^{-1}$. AWGN was added to the FID to give it an SNR of approximately 10 dB. As the visualisation of 5D space is beyond the scope of this work, only two parameters, the frequency and damping factor, were optimised from an initial guess $\boldsymbol{\theta}^{(0)}$, with the amplitude and phase being fixed to their true values. The initial guess comprised a frequency of 1.1 Hz, and a damping factor of 0.8 s $^{-1}$, with the solid grey lines in panels a & b

denoting the model generated in the time- and Fourier-domains, respectively. $\theta^{(0)}$ was subjected to NLP twice. In the first instance, the exact Hessian matrix, given by Equation 2.45b was used in order to compute each update step, while in the second the Hessian approximation given by Equation 2.48 was used. The initial radius of the trust region was set to 1/10 of the gradient norm (≈ 0.3), which has a precedent in the literature[82]. The trajectories of the parameter vector are denoted by coloured lines in panel c. In both cases, the NLP routine successfully converged at result $\theta^{(*)}$ in agreement with the true frequency and damping factor used to construct the FID. However, it is clear that using the true Hessian matrix (blue) led to a far better rate of convergence compared with the approximated analogue (red), which exhibited “zig-zagging”[¶]. 14 iterations were required to reach the convergence criterion $\epsilon \leq 10^{-8}$ when the true Hessian was used, while 81 were required for the approximated case. Despite being an anecdotal example, this highlights that use of the true Hessian matrix tends to allow a better rate of convergence. However, for FIDs comprising many signals and far more points, the approximated form often requires a shorter time to converge overall; the cost of computing the second derivatives required for the true Hessian becomes incredibly expensive, as will become clear in Section 2.4.

2.3.6 Phase Variance Minimisation

An NLP procedure tasked with minimising the discrepancy between a model and the observed data $\mathcal{F}(\theta)$ is well-suited to produce an accurate holistic representation of the data, assuming a sufficiently large model order is used. However — as with many inverse problems — FID estimation is an “ill-posed problem”,([details/citation?](#)) such there are conceivably a large subset of parameter vectors which will achieve this. It is desirable to produce an estimate which not only achieves a good fit to the data, but which is also able to explain the process underpinning the observation. It is for this reason that iterative procedures typically require significant quantities of *a priori* information in order to produce meaningful estimation results (see the discussion of VARPRO and AMARES, Section 1.3.4). This is also why they are often able to produce the best results relative to other methods where such information is not incorporated into the method.

While the MPM is often able to generate reasonable parameter estimates, one particular feature has been noticed which is a common feature across results; often, oscillators in the model exhibit *spurious phases*. As has been discussed (see Section 1.3.1), signals in most NMR experiments have phases which depend on their resonance frequencies to first order. This is routinely corrected in conventional NMR spectral processing, such that all signals effectively acquire a phase of approximately 0°. This feature of the dataset can be exploited in order to overcome the aforementioned shortcoming of the MPM, through appropriate adjustment of the NLP routine. As

[¶]This phenomenon is often seen in gradient descent methods, in which each update occurs in the opposite direction to the gradient.

such, assuming that the data has been phase corrected^{||}, regularising the fidelity, by incorporating the variance of oscillator phases, can lead to improved estimation results. Examples of this principle will be provided later (Section 3.1). The updated fidelity becomes

$$\mathcal{F}_\phi(\boldsymbol{\theta} | \mathbf{Y}) = \|\mathbf{Y} - \mathbf{X}(\boldsymbol{\theta})\|^2 + \text{Var}_o(\boldsymbol{\phi}), \quad (2.56)$$

where $\text{Var}_o(\boldsymbol{\phi})$ is the *circular variance* of the oscillator phases (*vide infra*).

Remark 4. *The inclusion of the phase variance into the fidelity is one of the motivating reasons for normalising the data prior to estimation (see Remark 3). $\text{Var}_o(\boldsymbol{\phi})$ is constrained to the interval $[0, 1]$. If the data were not normalised, it is likely that $\|\mathbf{Y} - \mathbf{X}\|^2$ would dominate $\text{Var}_o(\boldsymbol{\phi})$ in Equation 2.56, such that the influence of the phase variance would be negligible.*

Circular Variance

Oscillator phases are an example of a *circular variable*, in that all phases are wrapped within an interval of width 2π . Given an unconstrained (unwrapped) phase $\tilde{\phi} \in \mathbb{R}$, the corresponding wrapped phase $\phi \in (-\pi, \pi]$ is given by

$$\phi = ((\tilde{\phi} + \pi) \bmod 2\pi) - \pi. \quad (2.57)$$

This makes the conventional (linear) definition of variance, given by

$$\text{Var}_l(\boldsymbol{\phi}) = \frac{1}{M} \sum_{m=1}^M (\phi_m - \mu(\boldsymbol{\phi}))^2, \quad (2.58a)$$

$$\mu(\boldsymbol{\phi}) = \frac{1}{M} \sum_m \phi_m, \quad (2.58b)$$

unsuitable for phases. Consider as a simple example a scenario where there are two oscillators with phases $\tilde{\boldsymbol{\phi}} = [\pi + \delta \ \pi - \delta]^T$ for some small δ . The phase variance is expected to be small as the phases are similar. However, with the inclusion of wrapping through application of Equation 2.57, these phases would actually be set to $\boldsymbol{\phi} = [-\pi + \delta \ \pi - \delta]^T$, and the conventional definition of phase variance would be large. It is therefore apparent that a definition of variance which accounts for

^{||}Rather than rely on the data being phase-corrected, one could envisage replacing the phase variance with a term which guides the oscillators to adopt a first-order phase relationship. The reason why the phase variance has been chosen is two-fold: (i) Applying phase-correction to NMR data is straightforward and can be automated, meaning the user would experience minimal burden. (ii) As will be discussed in Section 2.5, it is beneficial to have a spectrum comprising pure absorption-mode Lorentzians in order to produce frequency-filtered “sub-FIDs” from the original data, so the data being estimated will be phase-corrected anyway.

the periodicity of the phases is needed. The *circular variance* is given by[83: Chapter 3]

$$[0, 1] \ni \text{Var}_o(\boldsymbol{\phi}) = 1 - \frac{R}{M}, \quad (2.59a)$$

$$R = \sqrt{c_{\Sigma}^2 + s_{\Sigma}^2}, \quad (2.59b)$$

$$c_{\Sigma} = \sum_m \cos \phi_m, \quad (2.59c)$$

$$s_{\Sigma} = \sum_m \sin \phi_m. \quad (2.59d)$$

R is the length of the resultant vector produced by summing M unit vectors with the angles given by $\boldsymbol{\phi}$. In the case that all the vectors have the same angle, $R = M$, leading to the variance being 0 as expected. At the other extreme, with M vectors uniformly separated about the unit circle (with an angle $2\pi/M-1$ between all pairs of adjacent vectors), the vectors will perfectly cancel, leading to $R = 0$. In this case, the maximum variance of 1 is obtained.

The first and second derivatives of the circular variance are required for the computation of the gradient vector and Hessian matrix, through

$$g_i = -2\Re \left\langle (\mathbf{Y} - \mathbf{X}), \frac{\partial \mathbf{X}}{\partial \theta_i} \right\rangle + \frac{\partial \text{Var}_o(\boldsymbol{\phi})}{\partial \theta_i} \quad (2.60a)$$

$$b_{i,j} = 2\Re \left(\underbrace{\left\langle \frac{\partial \mathbf{X}}{\partial \theta_i}, \frac{\partial \mathbf{X}}{\partial \theta_j} \right\rangle - \left\langle (\mathbf{Y} - \mathbf{X}), \frac{\partial^2 \mathbf{X}}{\partial \theta_i \partial \theta_j} \right\rangle}_{\text{Neglected if approximation used}} \right) + \frac{\partial^2 \text{Var}_o(\boldsymbol{\phi})}{\partial \theta_i \partial \theta_j}. \quad (2.60b)$$

The derivatives of the phase variance are given by:

$$\frac{\partial \text{Var}_o(\boldsymbol{\phi})}{\partial \theta_i} = \begin{cases} \frac{1}{RM} (c_{\Sigma} \sin \phi_{i-M} - s_{\Sigma} \cos \phi_{i-M}) & M \leq i < 2M \\ 0 & \text{otherwise} \end{cases} \quad (2.61a)$$

$$\frac{\partial^2 \text{Var}_o(\boldsymbol{\phi})}{\partial \theta_i \partial \theta_j} = \begin{cases} \frac{1}{RM} \left[\frac{1}{R^2} (c_{\Sigma} \sin \phi_{i-M} - s_{\Sigma} \cos \phi_{i-M})^2 + c_{\Sigma} \cos \phi_{i-M} + s_{\Sigma} \sin \phi_{i-M} - 1 \right] & M \leq i, j < 2M, i = j \\ \frac{1}{RM} \left[\frac{1}{R^2} (c_{\Sigma} \sin \phi_{i-M} - s_{\Sigma} \cos \phi_{i-M}) \times (c_{\Sigma} \sin \phi_{j-M} - s_{\Sigma} \cos \phi_{j-M}) - \cos(\phi_{i-M} - \phi_{j-M}) \right] & M \leq i, j < 2M, i \neq j \\ 0 & \text{otherwise} \end{cases} \quad (2.61b)$$

2.4 Profiling the MPM and NLP

The routine described for FID estimation involves operations which can be computationally demanding, with those demands increasing with the number of points in the FID, as well as the number of oscillators in the model. This is the case both in terms of the amount of work done by the central processing unit (CPU), and the amount of random access memory (RAM) needed to store all the required information as the routine runs. For the MPM, the most demanding aspect is SVD calculations while for numerical optimisation, it is generation of the Hessian matrix at each iteration. Detailed accounts of the computational complexity of the MPM and MMEMPM have been presented[61, 62]. However, it is useful to consider what the actual “wall-clock times” of these routines are on a modern computer. A lot of accounts on the MPM are from decades before this work, and so the time required for the routine to run will have decreased a lot thanks to improvements in processing power. For example, the account by Pines a co-workers from 1997 outlining the ITMPM states that a signal comprising 1024 points would take about 4.5 min to be processed by the MDL and MPM, using a 100 MHz CPU[58]. On the system used for all results generated for this work (see Remark 5) an equivalent computation takes about 100 ms.

Remark 5. All results generated in this work were acquired using a workstation featuring a Intel® Core™ i9-10900X CPU @ 3.7 GHz, and 32 GiB of RAM.

2.4.1 1D MPM

A series of FIDs were constructed with a variable number of time-points $N \in \{512k \mid k \in \{1, 2, \dots, 16\}\}$. For each FID, the MPM was performed 5 times, with a pencil parameter $L = \lfloor N/3 \rfloor$. A PYTHON implementation of the MPM was timed using a line profiler, which outputs the amount of time spent running the code on each line of a given script**. The mean complete time to run the MPM is plotted as a function of N in panel a of Figure 2.3, where it can be seen that for sufficiently large N the MPM is computed in approximately $\mathcal{O}(N^3)$ time. This is due to the most time-consuming aspect of the MPM being the computation of the SVD of \mathbf{H}_y , whose size is approximately $\frac{2N}{3} \times \frac{N}{3}$ ††. For small values of N (i.e. the square points in the plot), a deviation away from a cubic relationship is observed. This is since the computation of the complex amplitudes in accordance with Equation 2.15 has a comparatively significant run time‡‡. The figure shows the fit of a cubic function, of the form $aN^3 + b$, to the circular points, corresponding to values of N in the “cubic regime”. This function has been extrapolated to 16k points, to give an idea of the required run-time over a

**The line profiler used is accessible via: https://github.com/pyutils/line_profiler.

††The time complexity for the SVD of generic a $M \times N$ matrix is $\mathcal{O}(\min(M, N)^2 \cdot \max(M, N))$

‡‡For a 512 point signal, the SVD of \mathbf{H}_y took up roughly 80% of the complete run time, while the computation of the complex amplitudes took up roughly 20%. For a 8192 point signal, these percentages had changed to > 99% and < 1%, respectively.

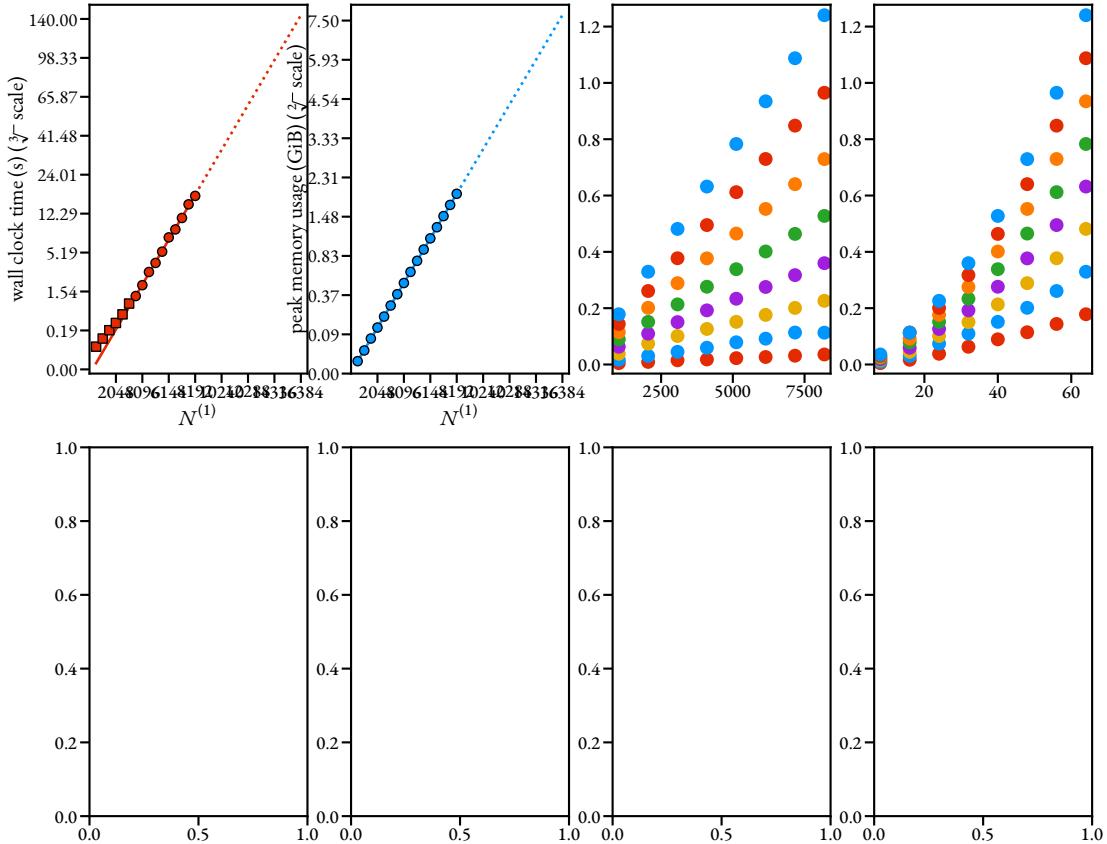


FIGURE 2.3: Profiles outlining the computation times and peak memory consumption of the MPM and for Hessian construction. **a.** Wall clock time to compute the MPM of a signal with 10 signals, as a function of number of points. Plotted alongside the times measured for each N is a cubic fit of the circular points (solid: interpolation, dotted: extrapolation). **b.** Peak memory consumption in performing the MPM, along with a quadratic fit. **TODO: 1D Hessian: variable M and N, MMEMPM, wall clock and memory, 2D Hessian: variable M and product of $N^{(1)}N^{(2)}$.**

wide range of N (of course, the run time will vary depending on the hardware used).

To assess the effect of FID size on the space complexity of the MPM, the routine was run again for each N defined, with a memory profiler analysing the amount of RAM used with time^{§§}. The peak RAM consumption is plotted in panel b of Figure 2.3. A clear quadratic dependence of consumption is realised as function of N , which reflects the fact that the space complexity for the SVD of a $r \times c$ matrix is $\mathcal{O}(rc)$.

2.4.2 MMEMPM

2.4.3 1D Hessian

2.4.4 2D Hessian

2.5 Frequency Filtration

The previous section provides motivation for finding ways to reduce both the number of data-points and the number of signals present in an FID of interest, without compromising the ability of the estimation routine to parameterise it. A means of generating frequency-filtered “sub-FIDs” is presented, which is able to achieve both of these. In essence, this sub-FID approach transforms the problem of FID estimation from a single large-scale estimation problem to a number of smaller-scale problems. As well as realising vast improvements in computational speed, filtering also enables a user to focus solely on spectral regions that are of interest. It is common for certain spectral regions to be so densely populated that it is futile to attempt to extract meaningful quantitative information at the per-signal level, especially with 1D NMR data. By applying filtering, all focus can be devoted to those reasons which can realistically be studied.

The concept of estimating selected regions in NMR can be traced back to Tang and Norris’s LP-ZOOM method[84], which employs the use of *subband decomposition*. A number of methods, including adaptive ones in which the subbands employed for estimation are adjusted according to certain criteria, have also emergedadd citations.

2.5.1 The virtual echo

The key steps of the filtering procedure used in this work consists of (i) taking the FT of the FID (ii) applying a band-pass filter to nullify parts not being considered, and (iii) returning the spectrum back to the time-domain by an inverse Fourier transform (IFT). For a filtered sub-FID to be faithfully modelled by a summation of exponentially damped complex sinusoids, it is necessary

^{§§}The memory profiler used is accessible via: https://github.com/pythonprofilers/memory_profiler.

that the spectral peaks of interest lie effectively entirely within the filter region^{¶¶}. Due to their narrower linewidths relative to dispersion Lorentzians (Footnote **, Page 17), a spectrum solely comprising absorption Lorentzians is therefore desired. The virtual echo (VE) has been employed here, which has found application in the field of compressed sensing NMR[85–87]. The VE is a signal with double the size as the original FID, with the key characteristic that its FT has a real component which is equivalent to its counterpart derived from an unaltered FID zero-filled to doubled its size, and an imaginary component of zeros. The VE concept can be applied to data of any number of dimensions. However, only 1D virtual echoes are employed in the work outlined in this thesis. An account of the 2D virtual echo is provided in Appendix B.2.

Assuming that a 1D FID $\mathbf{y} \in \mathbb{C}^N$ is phased, such that $\phi = \mathbf{0} \in \mathbb{R}^M$, it can be described by

$$y_n = \xi_n(c_n + i s_n) + w_n, \quad (2.62a)$$

$$\xi_n = \sum_m a_m \exp(-\eta_m n \Delta_t), \quad (2.62b)$$

$$c_n / s_n = \sum_m \cos / \sin(2\pi f_m n \Delta_t). \quad (2.62c)$$

The frequency-dependence has been decomposed into its real and imaginary components. With this in mind, a conjugate pair of signals $\psi_{\pm} \in \mathbb{C}^N$ are defined:

$$\psi_{\pm,n} = \xi_n(c_n \pm i s_n) + w_n \equiv \Re(y_n) \pm i \Im(y_n) \quad (2.63)$$

Two vectors $\mathbf{t}_1, \mathbf{t}_2 \in \mathbb{C}^{2N}$ are constructed using the conjugate pair. \mathbf{t}_1 is given by ψ_+ padded with zeros from below:

$$\mathbf{t}_1 = \begin{bmatrix} \psi_+ \\ \mathbf{0} \in \mathbb{C}^N \end{bmatrix}. \quad (2.64)$$

\mathbf{t}_2 is given by ψ_- with its elements in reversed order (\cdot^{\leftrightarrow}), padded with zeros from above, and finally subjected to a right circular shift by one element ($\cdot^{\circlearrowright}$):

$$\mathbf{t}_2 = \begin{bmatrix} \mathbf{0} \in \mathbb{C}^N \\ \psi_-^{\leftrightarrow} \end{bmatrix}^{\circlearrowright}. \quad (2.65)$$

The VE \mathbf{y}_{ve} is then given by $\mathbf{t}_1 + \mathbf{t}_2$, with the first element divided by 2, which is equivalent to

$$\mathbf{y}_{\text{ve}} = \left[\Re(y_0) \ y_1 \ \cdots \ y_{N-1} \ 0 \ y_{N-1}^* \ \cdots \ y_1^* \right]^T. \quad (2.66)$$

^{¶¶}Lorentzian lineshapes tend to, but don't reach zero, as the distance from the maximum tends to ∞ (Equation 1.28). However, as long as a sufficiently wide filter region is defined, the “tails” of the Lorentzian which fall outside the filter window can be assumed to be negligible, especially when in the presence of noise.

As eluded to already, the FT of \mathbf{y}_{ve} produces a spectrum \mathbf{s}_{ve} such that $\Im(\mathbf{s}_{\text{ve}}) = \mathbf{0}$, with $\Re(\mathbf{s}_{\text{ve}})$ featuring absorption Lorentzian peaks.

2.5.2 The filtering process

To filter the spectrum \mathbf{s}_{ve} , it is subjected multiplication with a function which acts as a band-pass filter. An example of a suitable filter is a *super-Gaussian* $\mathbf{g} \in \mathbb{C}^{2N}$ defined by a central index $c \in \{0, \dots, 2N - 1\}$ and a bandwidth $b \in \{0, \dots, 2N - 1\}$:

$$g_n = \exp\left(-2^{p+1}\left(\frac{n-c}{b}\right)^p\right). \quad (2.67)$$

The scalar $p \in \mathbb{R}_{>0}$ dictates the steepness of the filter at the boundaries, with the function becoming more rectangular as it increases. It is set to 40 in this work. Application of the super-Gaussian filter to \mathbf{s}_{ve} would lead to large sections of the filtered spectrum being 0. This has an undesired impact on the MDL, as noise that resides within the filter region will now seem to resemble true signal, as its amplitude is infinitely greater than the zeroed regions. A massive over-estimation of model order is likely to result because of this due to this. In order to obtain better results from model order selection, an array of synthetic AWGN is added to the filtered spectrum. To achieve this, a region in \mathbf{s}_{ve} is specified which contains no discernible signal peaks (referred to as the *noise region*). The variance of this region σ^2 is determined, and used to construct a vector of values sampled from a normal distribution with mean 0 and variance σ^2 , $\mathbf{w}_{\sigma^2} \in \mathbb{R}^{2N}$. The filtered spectrum is then given by

$$\tilde{\mathbf{s}}_{\text{ve}} = \mathbf{s}_{\text{ve}} \odot \mathbf{g} + \mathbf{w}_{\sigma^2} \odot (\mathbf{1} - \mathbf{g}). \quad (2.68)$$

Note that the noise array's magnitude at each point is attenuated based on the value of the super-Gaussian filter, as a means of ensuring the noise variance remains consistent across the frequency space.

After filtering, $\tilde{\mathbf{s}}_{\text{ve}}$ is returned to the time-domain by IFT. The IFT of a real-valued spectrum generates a conjugate-symmetric signal, which is also a VE. This is sliced so as to retain the first half, which is the final filtered sub-FID $\tilde{\mathbf{y}} \in \mathbb{C}^N$. A depiction of the key elements involved in the filtering process is provided by Figure 2.4, while a step-by-step outline is provided by Algorithm 2.3.

The central index and bandwidth of the super-Gaussian filter function are given by the following expressions:

$$c = \frac{1}{2}(l_{\text{idx}} + r_{\text{idx}}), \quad (2.69a)$$

$$b = l_{\text{idx}} - r_{\text{idx}}, \quad (2.69b)$$

where l_{idx} and r_{idx} denote the desired indices where the filter's left and right bounds are located,

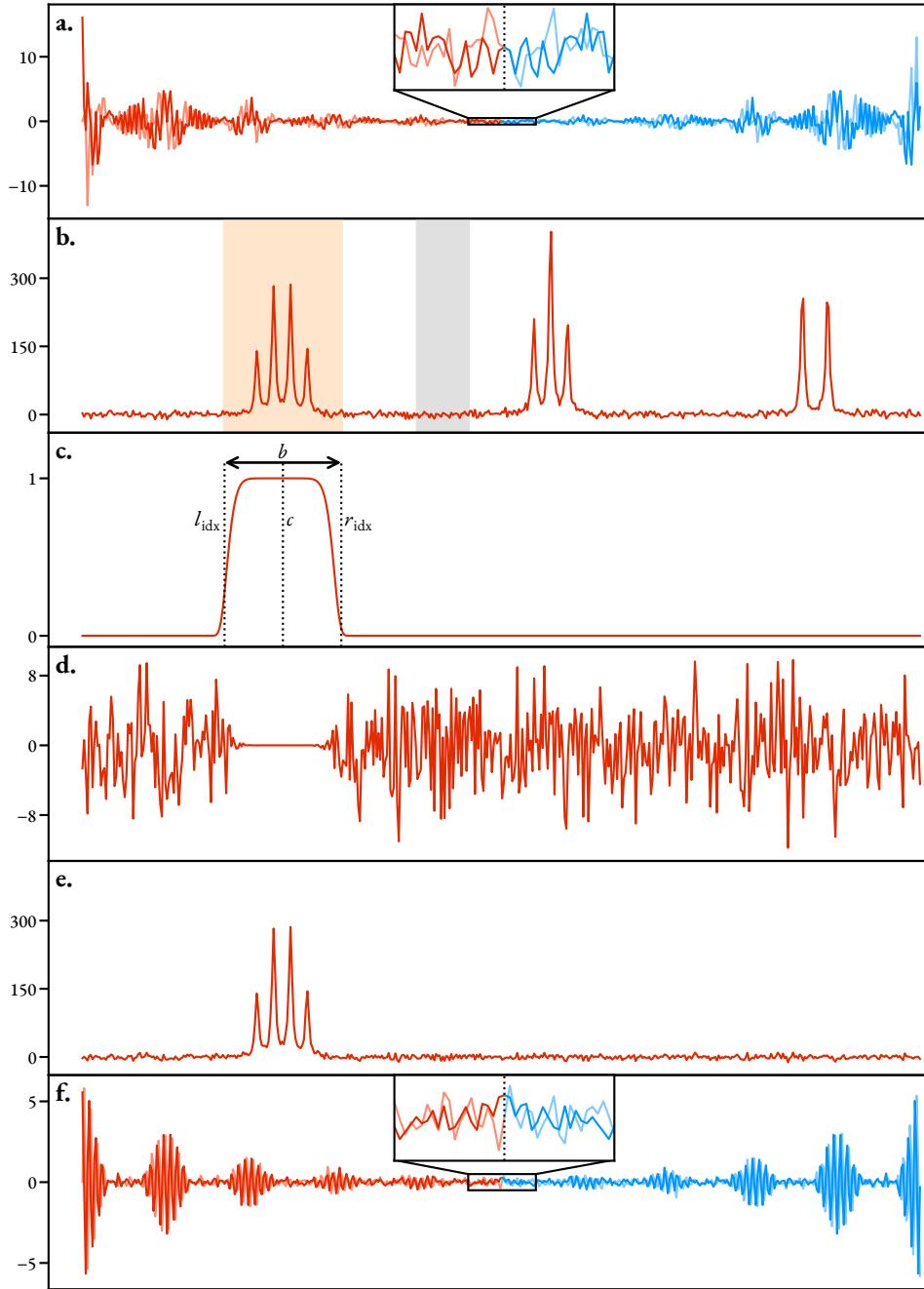


FIGURE 2.4: An illustration of the filtering procedure applied to a 1D FID. **a.** A VE y_{ve} , with the first and last N points coloured red and blue, respectively. The middle of the VE is magnified to highlight its conjugate symmetry. **b.** The FT of the VE, s_{ve} . The region of interest (orange) and noise region (grey) are denoted. **c.** A super-Gaussian function used as a band-pass filter, g . **d.** Synthetic noise vector to be added to the filtered spectrum, $w_{\sigma^2}(1 - g)$. **e.** The filtered spectrum \tilde{s}_{ve} , formed by applying the super-Gaussian filter, and adding the noise vector. **f.** The IFT of the filtered spectrum, \tilde{y}_{ve} , from which the final filtered signal \tilde{y} is obtained by extracting the first N (red) points.

respectively. Array indices can be obtained from the corresponding spectral frequencies f_{Hz} via

$$f_{\text{idx}} = \left\lfloor \frac{(2N - 1)(f_{\text{sw}} + 2(f_{\text{off}} - f_{\text{Hz}}))}{2f_{\text{sw}}} \right\rfloor \quad (2.70)$$

$$\forall f_{\text{Hz}} \in [f_{\text{off}} - \frac{1}{2}f_{\text{sw}}, f_{\text{off}} + \frac{1}{2}f_{\text{sw}}].$$

Alternatively, conversion from ppm to array indices can be achieved by replacing f_{Hz} in Equation 2.70 with $f_{\text{ppm}}f_{\text{sfo}}$, where f_{sfo} is the transmitter frequency (MHz) and f_{ppm} is the frequency expressed as a chemical shift.

Spectrum slicing

Thus far, the method described is able to reduce the number of signals, however the filtered sub-FID still comprises the same number of points. However, it is clear that there are a large number of points outside the region of interest in $\tilde{\mathbf{s}}_{\text{ve}}$ that do not possess any meaningful information. Discarding such points will then lead to a sub-FID with the same information about the signals of interest, but with far fewer points. To achieve this, a slicing ratio is defined, $\chi \in \mathbb{R} : \chi > 1$, which dictates the left and right indices at which the spectrum should be sliced:

$$l_{\text{slice}} = \begin{cases} c - \left\lfloor \frac{b\chi}{2} \right\rfloor & \text{if } \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.71\text{a})$$

$$r_{\text{slice}} = \begin{cases} c + \left\lceil \frac{b\chi}{2} \right\rceil & \text{if } \leq 2N - 1 \\ 2N - 1 & \text{otherwise} \end{cases} \quad (2.71\text{b})$$

The filtered spectrum is then sliced accordingly:

$$\mathbb{R}^{r_{\text{slice}} - l_{\text{slice}}} \ni \tilde{\mathbf{s}}_{\text{ve,slice}} = \tilde{\mathbf{s}}_{\text{ve}}[l_{\text{slice}} : r_{\text{slice}} + 1] \equiv [\tilde{s}_{\text{ve},l_{\text{slice}}}, \tilde{s}_{\text{ve},l_{\text{slice}}+1}, \dots, \tilde{s}_{\text{ve},r_{\text{slice}}}, \tilde{s}_{\text{ve},r_{\text{slice}}+1}]^T. \quad (2.72)$$

Generation of the final sub-FID is then achieved in a similar fashion to before: by performing IFT, and retaining the first half of the signal. It is also necessary to scale the signal by the ratio of the number of points in the sliced spectrum and its unsliced counterpart, in order to ensure that the amplitudes of each signal are unaffected:

$$\tilde{\mathbf{y}} = \frac{r_{\text{slice}} - l_{\text{slice}}}{2N} \text{IFT}(\tilde{\mathbf{s}}_{\text{ve,slice}})[0 : N_{\text{slice}}], \quad (2.73\text{a})$$

$$N_{\text{slice}} = \left\lfloor \frac{r_{\text{slice}} - l_{\text{slice}}}{2} \right\rfloor \quad (2.73\text{b})$$

The associated sweep width and transmitter offset of the FID will have been altered by this process, and in order to derive accurate frequencies and damping factors for the sliced signal, it is necessary to determine these. The corrected values can be computed using

$$f_{\text{sw,slice}} = \frac{r_{\text{slice}} - l_{\text{slice}}}{2N - 1} f_{\text{sw}} \quad (2.74\text{a})$$

$$f_{\text{off,slice}} = f_{\text{off}} + \frac{f_{\text{sw}}}{2} \left(1 - \frac{l_{\text{slice}} + r_{\text{slice}}}{2N - 1} \right) \quad (2.74\text{b})$$

ALGORITHM 2.3 Filtering procedure for 1D data. l_{idx} and r_{idx} are indices of the left and right bounds of the region of interest. $l_{\text{idx,noise}}$ and $r_{\text{idx,noise}}$ are the analogous bounds for the noise region. All of these values should be $\in \{0, \dots, 2N - 1\}$. These would typically be provided in units of Hz or ppm by a user; conversion to indices can be carried out using Equation 2.70. RANDOMSAMPLE indicates taking a random sample from the given distribution.

```

procedure FILTER1D( $\mathbf{y} \in \mathbb{C}^N, l_{\text{idx}}, r_{\text{idx}}, l_{\text{idx,noise}}, r_{\text{idx,noise}}$ )
     $\mathbf{s}_{\text{ve}} \leftarrow \text{VIRTUALECHO1D}(\mathbf{y});$ 
     $\mathbf{s}_{\text{ve}} \leftarrow \text{FT}(\mathbf{s}_{\text{ve}});$ 
     $c_{\text{idx}} \leftarrow (l_{\text{idx}} + r_{\text{idx}})/2;$ 
     $b_{\text{idx}} \leftarrow r_{\text{idx}} - l_{\text{idx}};$ 
     $\mathbf{g} \leftarrow \text{SUPERGAUSSIAN1D}(2N, c_{\text{idx}}, b_{\text{idx}});$ 
     $\mathbf{s}_{\text{noise}} \leftarrow \mathbf{s}_{\text{ve}}[l_{\text{idx,noise}} : r_{\text{idx,noise}} + 1];$ 
     $\sigma^2 \leftarrow \text{Var}(\mathbf{s}_{\text{noise}});$ 
     $\mathbf{w}_{\sigma^2} \leftarrow \mathbf{0} \in \mathbb{R}^{2N};$ 
    for  $n = 0, \dots, 2N - 1$  do
         $w_{\sigma^2,n} \leftarrow \text{RANDOMSAMPLE}(\mathcal{N}(0, \sigma^2));$ 
    end for
     $\tilde{\mathbf{s}}_{\text{ve}} \leftarrow \mathbf{s}_{\text{ve}} \odot \mathbf{g} + \mathbf{w}_{\sigma^2} \odot (\mathbf{1} - \mathbf{g});$ 
     $\tilde{\mathbf{y}}_{\text{ve}} \leftarrow \text{IFT}(\tilde{\mathbf{s}}_{\text{ve}});$ 
     $\tilde{\mathbf{y}} \leftarrow \tilde{\mathbf{y}}_{\text{ve}}[:N];$ 
    return  $\tilde{\mathbf{y}};$ 
end procedure

procedure VIRTUALECHO1D( $\mathbf{y} \in \mathbb{C}^N$ )
    return  $\begin{bmatrix} \Re(y_0) & y_1 & \cdots & y_{N-1} & 0 & y_{N-1}^* & \cdots & y_1^* \end{bmatrix}^T$ 
end procedure

procedure SUPERGAUSSIAN1D( $N \in \mathbb{N}, c_{\text{idx}}, b_{\text{idx}}$ )
     $\mathbf{g} \leftarrow \mathbf{0} \in \mathbb{R}^N;$ 
    for  $n = 0, \dots, N - 1$  do
         $g_n \leftarrow \exp\left(-2^{41}\left(\frac{n - c_{\text{idx}}}{b_{\text{idx}}}\right)^{40}\right);$ 
    end for
    return  $\mathbf{g}$ 
end procedure

```

$\triangleright p$ in Equation 2.67 has been set to 40.

2.6 Summary

The MPM is well established as an effective for parametric estimation of signals in a number of disciplines including NMR. Due to considerable advances in computational processing power since its introduction, estimates of FIDs can be acquired in reasonable times. One notable downside of the technique that has been realised while assessing its effectiveness in parametrising NMR FIDs is its propensity to return oscillators with unexpected phase behaviour, especially in scenarios involving signals with similar frequencies, exhibiting considerable overlap in the Fourier domain. For this reason, using the result of the MPM as an initial guess to feed into a phase-variance regularised NLP routine is proposed as a means of returning improved parameter estimates. The theory underpinning the procedure has been explored in this chapter.

The computational burden of running the procedure is large and often intractable for complete NMR signals, which often comprise thousands of points, and at least hundreds of contributing signals. This has been illustrated through profiling both the CPU time and the peak memory consumption for the 1D and 2D methods. **Say more when completed?** For this reason, a method to break the estimation problem into a series smaller problems, through the construction of frequency-filtered sub-FIDs, is introduced.

Having established an estimation routine, the next chapter focusses on its performance, as well as applications which are possible through parametric estimation.

1D RESULTS AND APPLICATIONS

3

This chapter showcases results generated using the 1D estimation technique outlined in the previous chapter. Furthermore, two means by which the standard 1D estimation procedure can be extended for use in specific applications are presented.

Section 3.1 provides some examples of how the estimation routine performs on conventional 1D FIDs (i.e. datasets akin to those acquired using a pulse-acquire experiment). Section 3.2 illustrates how the basic 1D estimation routine can be extended to enable the consideration of datasets comprising a series of FIDs which feature attenuations in signal amplitudes across increments, including inversion recovery and diffusion experiments. Finally, Section 3.3 describes a protocol to generate ultra-broadband spectra through application of a single 90° frequency-swept pulse, which are devoid of quadratic phase dependencies as well as baseline distortions.

3.1 Conventional 1D datasets

3.1.1 “Twenty signals”

In order to assess the estimation routine proposed in Chapter 2 – specifically the effectiveness of applying NLP using an initial guess generated using the MPM – a series of synthetic FIDs were constructed using Equation 2.1 with $D = 1$. For each FID, a model order of $M = 20$ was used, the number of points sampled was $N = 1024$, the sweep width was $f_{\text{sw}} = 125 \text{ Hz}$, and the transmitter offset was $f_{\text{off}} = 0 \text{ Hz}$. Each oscillator was assigned a phase of 0°, while the amplitudes, frequencies and damping factors were drawn at random from the following distributions: $\alpha_m \sim \mathcal{U}(1, 5)$, $f_m \sim \mathcal{U}(-55 \text{ Hz}, 55 \text{ Hz})$, $\eta_m \sim \mathcal{U}(2 \text{ s}^{-1}, 8 \text{ s}^{-1}) \forall m \in \{0, \dots, 19\}$. An extra constraint was applied to the frequencies, such that no two oscillators were permitted to have frequencies that differed by less than $4f_{\text{sw}}^{(1)}/N^{(1)} \approx 0.49 \text{ Hz}$. Each noiseless FID \mathbf{x} was then corrupted with AWGN, with a

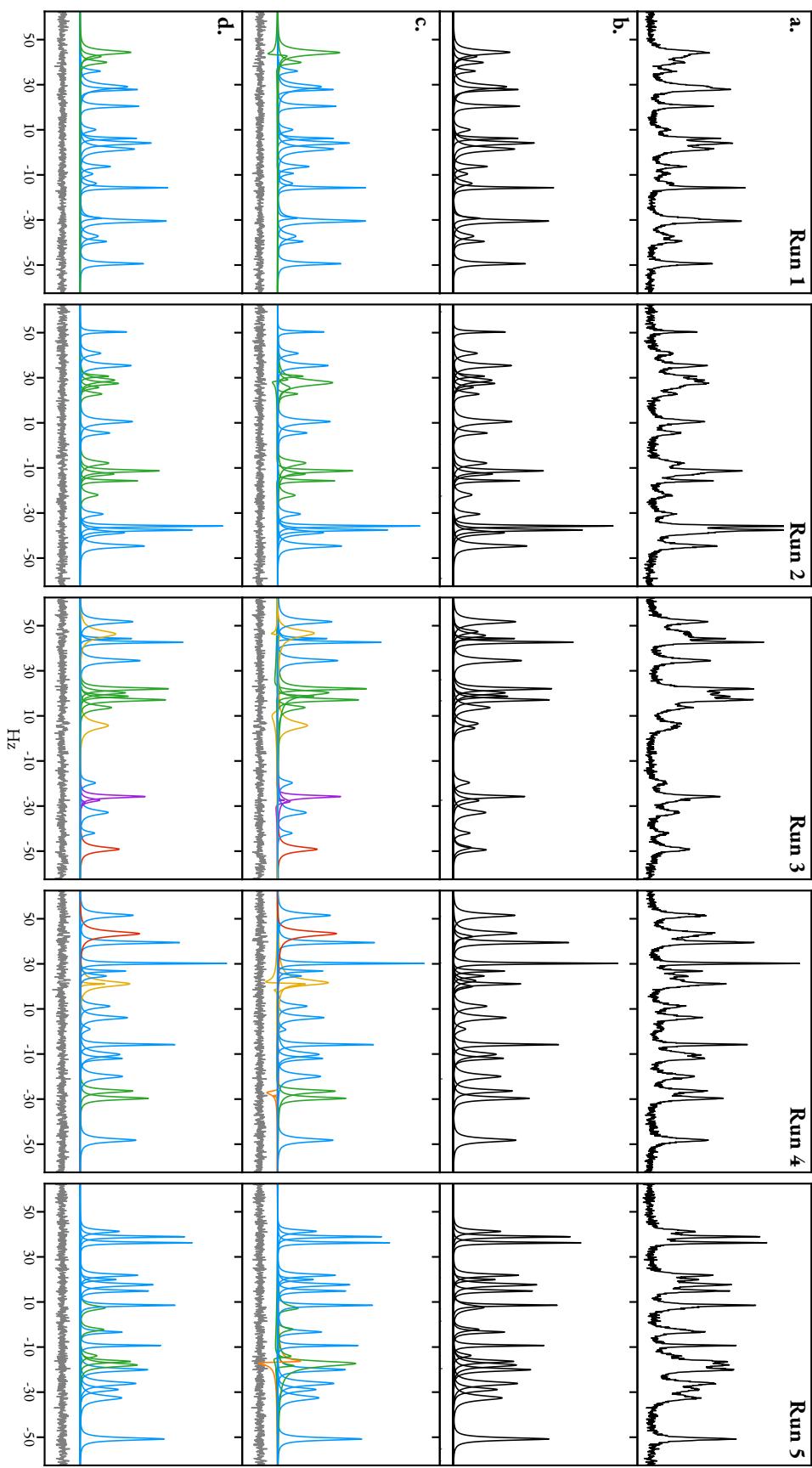


FIGURE 3.1: The result of estimating a series of 5 simulated signals comprising 20 oscillators (see the main text for details on how the datasets were constructed). **a.** Spectra of the datasets generated. **b.** Spectral lines corresponding to the true set of oscillators used to generate each dataset. **c.** Plots of spectral lines for each oscillator generated using the MPM. **d.** An equivalent plot for the result after applying NLP, with the MPM result being the initial guess. Also included in **c.** and **d.** is the residual between the data and the sum of the oscillator peaks (grey line). The colouring of oscillator lines in **c.** and **d.** is described in the main text.

target SNR of 25 dB, with the noise variance for each signal determined using the equation

$$\sigma^2 = \frac{1}{20^{2.5}N} \sum_{n=0}^{N-1} |x_n|^2. \quad (3.1)$$

Reference that this is the lowest SNR that the MPM considered to be effective. The spectra of the simulated FIDs are presented in panel a of Figure 3.1, with the set of true oscillator peaks in panel b.

For each FID, the MPM was performed, assuming that the model order is 30, constituting a considerable over-fit. The MDL tended to produce considerable under-estimates of M when applied to these FIDs, so the hard-coded value was used instead. Simulated signals featuring AWGN typically show a clean division between signal and noise components, with noise components commonly being characterised by small amplitudes and/or very small damping factors. For this reason, prior to subjecting the MPM result to NLP, oscillators which satisfied either $\alpha_m < 0.1$ or $\eta_m < 0.7 \text{ s}^{-1}$ were removed from the parameter set. The individual oscillators which make up the MPM result after purging spurious components are displayed in panel c of Figure 3.1, along with the residual between the data and the summation of all the oscillator peaks.

The MPM invariably generates a model with good agreement with the data, as evidenced by the residual. However, it can be seen that in several spectral regions across the datasets, especially ones that are highly crowded, oscillators possess parameters which deviate significantly from the true parameters, with the most notable feature being individual oscillator phases – these regularly stray far from 0° – and their associated amplitudes. The central motivation behind employing phase variance-regularised NLP is as a means of attempting to overcome this detrimental feature of the MPM. In panel c of Figure 3.1, the blue oscillators are those which agree very closely with a particular oscillator in the true set of parameters. Oscillators with other colours are not clearly mapped to a true oscillator, with the different colourings described shortly. The intention is for the NLP routine to adjust the parameters describing the non-blue oscillators in panel c such that they agree with oscillators found in the true set, while not affecting the blue oscillators. The results of NLP at convergence ($\epsilon = 10^{-8}$) are provided in panel d.

In discussing the outcome of the routine, it will be useful to introduce the concept of a *frequency neighbourhood*, a loose term which describes a small, continuous range of frequencies within the spectral window. As the NLP routine involves taking small steps through parameter space in an attempt to converge, it is unlikely that an oscillator which starts off with a frequency far away from a particular frequency neighbourhood will eventually enter it. As such, in order for the NLP routine to successfully estimate the region, sufficient oscillators need to present within the neighbourhood in the first place. Cases where the MPM generated enough oscillators for a given frequency neighbourhood, albeit with parameters which are noticeably off the true parameters

are in either green or yellow. Green oscillators are those which the NLP routine was able to adjust in order to achieve agreement with the true result. As such, they indicate improvements to the estimation result as opposed to the MPM being used by itself. Conversely, yellow oscillators denote cases where, though sufficient oscillators exist in the frequency neighbourhood in the initial guess, the NLP routine evolves such that at least one of the oscillators is driven by the phase variance constraint to acquire a negative amplitude, leading to it being purged from the parameter set. This typically occurs when an oscillator has an initial phase which is considerably greater than 90° . Yellow oscillators therefore indicate cases where the final result has under-fit the dataset. There are a few instances, denoted by red oscillators, where the MPM assigned too few oscillators to a particular frequency neighbourhood, and as such the NLP would not have been able to yield any improvement. The final two oscillator groupings, denoted by purple and orange, denote cases where the MPM generated more oscillators than are present in a given frequency neighbourhood (i.e. the data was over-fit in this region). Orange oscillators were purged by the NLP routine due to their acquiring negative amplitudes. This enabled a parsimonious fit of the frequency neighbourhood by the oscillators which remained. Finally, the purple oscillators denote the one occasion where an over-fit occurred, and the model order was not successfully reduced by the NLP routine.

Overall, it can be seen that the inclusion of NLP broadly improves the output of the MPM, where crowded regions often get assigned oscillators with spurious complex amplitudes (α and ϕ).

More discussion here...

3.1.2 Andrographolide

Figure 3.2 illustrates the outcome of applying the estimation routine to selected regions of a ^1H dataset of andrographolide (Figure D.1.e) in DMSO-d₆ acquired with a 600 MHz spectrometer. The NLP routine is effective at resolving the spurious phase-behaviour often generated by the MPM (cf. panels b and c). Its ability to estimate parameters from resonances with high dynamic range and high variation of damping factors is also evidenced by the fact that it was able to assign an intense broad singlet, corresponding to water which has entered the sample over time, alongside a nearby quartet corresponding to the methylene of some residual ethanol in the sample.

The routine generally does well at generating oscillators which describe the apparent multiplet structures associated with each spin. Table 3.1 provides an overview of the most significant couplings associated with the spins giving rise to the multiplet structures considered. One of the most challenging aspects of estimating NMR signals is the fact that data frequently contains individual resonances with incredibly similar frequencies due to the effect of scalar couplings. Molecules with fused ring systems such as andrographolide are prime examples of spin systems which generate such datasets, as they tend to have very dense coupling networks leading to complex multi-

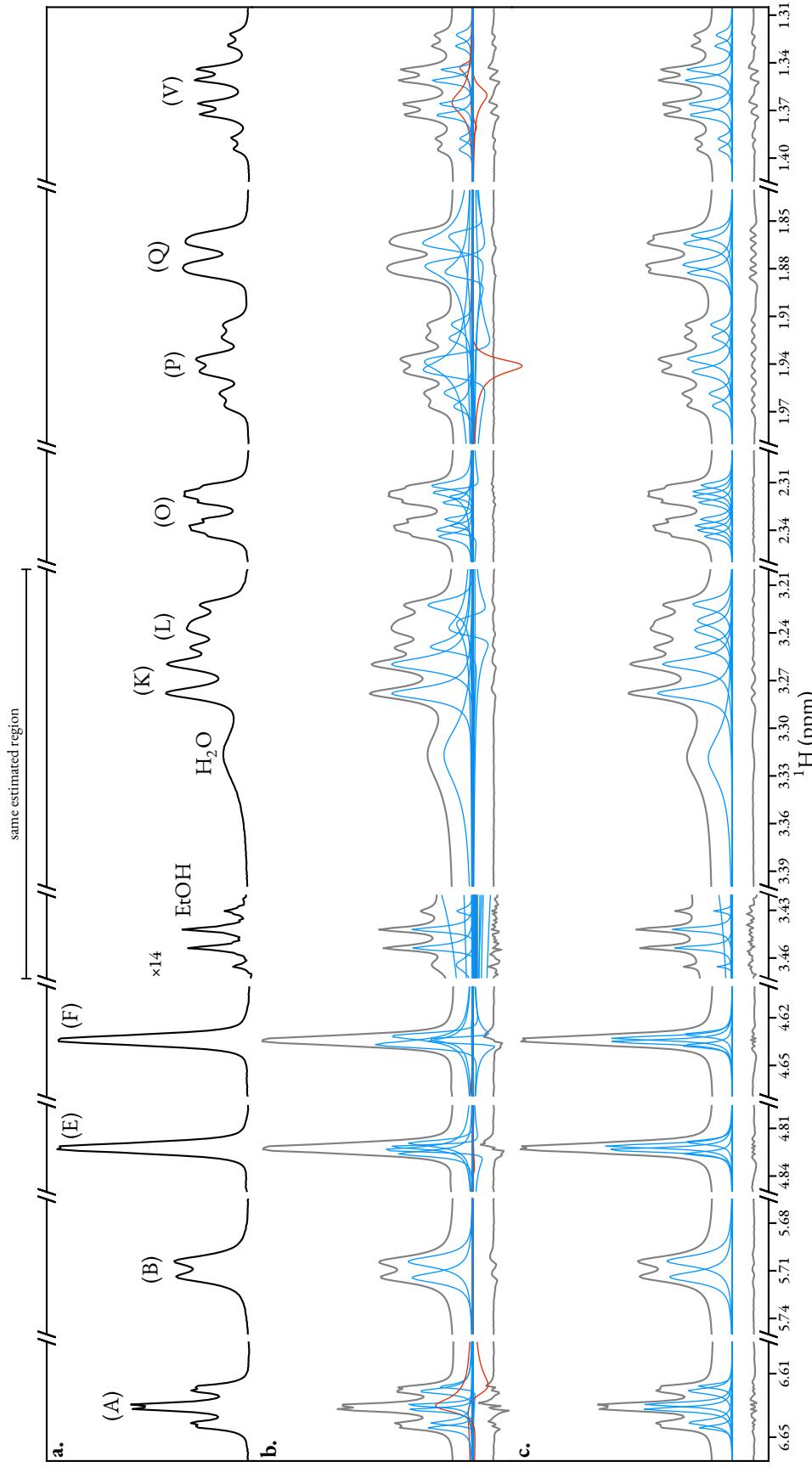


FIGURE 3.2: Result of applying the estimation routine to selected regions of a pulse-acquire dataset of andrographolide in DMSO-d_6 . **a.** Spectral data corresponding to the regions considered. **b.** The result of applying the MPM to the regions, with the model order predicted with the MDL. Blue/red lines: peaks of individual oscillators, grey line above: the model (sum of all oscillators), grey line below: the residual between the data and the model. **c.** The result after convergence of the NLP routine, again with the model above and residual below. Red peaks in panel b correspond to oscillators which acquire negative amplitudes during the NLP routine, and are subsequently purged. Note that one of the reasons estimated has been split in two in the figure to save space, with one half, featuring a signal from ethanol, being magnified.

Spin	Coupling partners	Multiplet structure
Andrographolide		
(A)	(D) ^{long} , (M) ^{vic} , (N) ^{vic}	ddd (<i>dt</i>)
(B)	(D) ^{ex}	d
(E)	(F) ^{vinyl} , ...	d...
(F)	(E) ^{vinyl} , ...	d...
(K)	(J) ^{gem} , (H) ^{ex}	d
(L)	(C) ^{ex} , (T) ¹⁸⁰ , (U) ⁶⁰	dd
(O)	(P) ^{gem} , (R) ⁶⁰ , (V) ⁶⁰	ddd
(P)	(O) ^{gem} , (R) ⁶⁰ , (V) ¹⁸⁰	ddd (<i>dt</i>)
(Q)	(M) ^{vic} , (N) ^v , ...	dd...
(V)	(O) ⁶⁰ , (P) ¹⁸⁰ , (R) ^g , (W) ¹⁸⁰	dddd (<i>dq</i>)
Cyclosporin A		
(A)	diastereotopic pair on β C	dd
(B)	—”—	dd
(C)	—”— & amide proton	ddd (<i>dt</i>)
(D)	proton on β C & amide proton	dd
(E)	methyl protons on β C & amide proton	dq
(F)	—”—	dq (<i>quintet</i>)

TABLE 3.1: Major coupling partners associated with spins in andrographolide and cyclosporin A whose signals are considered in Figure 3.2, respectively, along with the apparent multiplet structures that arise. For andrographolide, coupling partners are labelled as follows: ^{vinyl} geminal coupling between two vinylic protons, ^{ex} geminal coupling between two protons, in which one is bonded to an oxygen, leading to exchange decoupling, ^{gem} geminal coupling, ^{long} long-range coupling, ^{vic} vicinal coupling, ⁶⁰ geminal coupling, with a fixed dihedral angle of 60°, ¹⁸⁰ geminal coupling, with a fixed dihedral angle of 180°. All cyclosporin couplings of significance are geminal couplings. In cases where the observed multiplet structure is different to the true structure, the observed structure is in brackets.

plet structures. Also, fused systems often exhibit appreciable long-range couplings (between spins separated by four or more bonds) alongside more prevalent two-bond (geminal) and three-bond (vicinal) couplings.

Take the multiplet structure from spin (Q) as an example. This has separate vicinal couplings to the diastereotopic protons (M) and (N), which are likely the greatest magnitude couplings associated with (Q). If these were the only couplings, a doublet of doublets (dd) structure would be expected, which is what has been generated by the estimation routine (see panel c in the region of 1.85 ppm to 1.88 ppm). However, a comparison of the data and the model indicates that there is a clear discrepancy between the two, evidenced by systematic “wiggles” in the residual. This feature hints at an under-fit of the multiplet structure. The MPM generated oscillators with phases deviating far from 0° in order to achieve a good agreement with the data in a residual sense, though of course such a set of oscillators is unrealistic for a well-phased FID. Long-range couplings with magnitudes that are large enough to influence the appearance of (Q)’s multiplet structure are likely to be present, which leads to a signal in which all contributing resonances are too poorly resolved to realistically gleam any further meaningful information, at least at the field strength used.

As a second illustration, the signal corresponding to spin (V) is also under-fit, this time because the presence of a number of couplings of similar magnitudes leads to resonances coalescing at roughly the same frequency. For (V), a multiplet structure featuring 16 resonances forming in a “dddd” structure is expected, however 3 of the couplings are of similar magnitudes, such that individual resonances coalesce to form what is apparently a quartet of doublets (dq). The estimation routine was able to resolve this dq structure, however the large wobbles in the residual again signal that under-fitting has occurred, and each oscillator is in fact being used to fit two or more resonances present in the FID. Again, at the field strength used to acquire the FID, it is unlikely that an accurate resolution of all 16 oscillators by estimation is feasible.

The fit to spin (O)’s multiplet structure has a very flat residual, indicating that under-fitting is unlikely. With three major couplings of different magnitude, a “ddd” multiplet structure in which all 8 signals are discernible exists. The NLP routine has performed effectively in taking the initial guess from the MPM, featuring the correct number of oscillators albeit though with spurious phases, and generating a well-phased set of oscillators defining the ddd structure.

Spin (L) also has 3 major couplings, though only a dd structure is apparent, which is what has been assigned by the estimation routine. The small residual also implies that an under-fit has not occurred, even though a ddd structure would be anticipated at first glance. However, one of the coupling partners is a labile hydroxyl proton, which rapidly undergoes exchange with other protons on the sample*. Chemical exchange at a sufficiently high rate leads to a “decoupling” of the spins, such that the two resonances associated with a doublet coalesce into a broad singlet[31:

*Protons which are plausible exchange partners include those associated with residual water and ethanol in the sample, and hydroxyl protons from other andrographolide molecules.

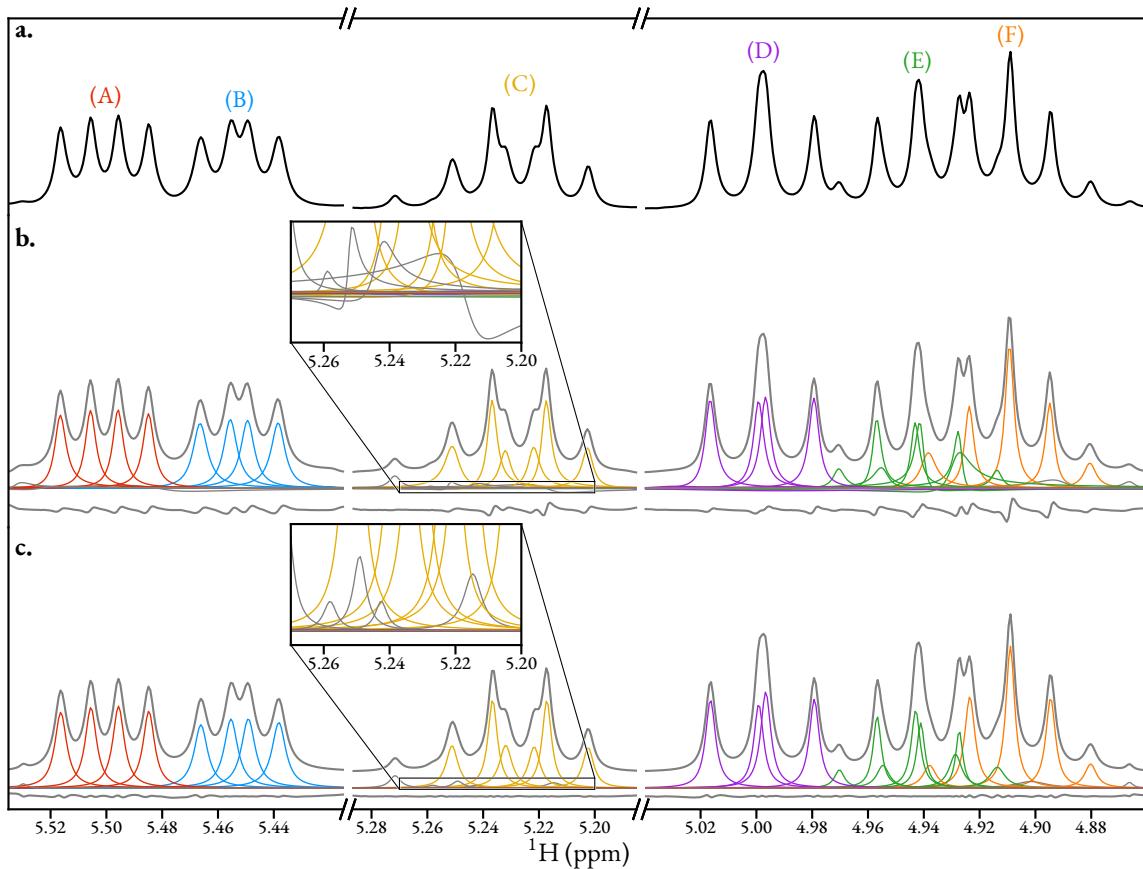


FIGURE 3.3: Result of applying the estimation routine to selected regions of a pulse-acquire dataset of cyclosporin A in benzene-d₆. The figure is of a similar layout to Figure 3.2. Coloured peaks correspond to oscillators which have been assigned to multiplet structures of known spins. Grey oscillators correspond to those with an unknown association to a particular spin.

Section 2.6.1.5].

Discussion of more challenging regions? Probably beyond the scope of the routine without further user input/maybe just too difficult full stop?

3.1.3 Cyclosporin A

The estimation routine was applied to regions of a ^1H pulse-acquire experiment of cyclosporin A **Structure in appendix** — a cyclic peptide comprising 11 amino acids — in benzene-d₆. The regions considered all contain signals caused by protons bound to C $^\alpha$ atoms in the peptide backbone[88]. Particular challenges in estimating this dataset are (a) the presence of heavily overlapping signals, and (b) the presence of low intensity “nuisance” peaks, likely from impurities in the sample. Nevertheless, the routine has performed effectively in generating a parameter estimate which agrees with the multiplet structures present.

3.2 Amplitude-attenuated Datasets

There are a number of 2D NMR experiments in which the variation of a parameter in the pulse sequence leads to the generation of FIDs of the same form except for an attenuation in their amplitudes across increments. These include experiments for the determination of translational diffusion rates and relaxation properties such as longitudinal and transverse decoherence rates. Here, an extension to the 1D estimation technique is described, facilitating the determination of these properties.

3.2.1 Relaxation experiments

Any spin system which has been perturbed from its equilibrium position will eventually return to equilibrium by relaxation. As introduced in Chapter 1, the simplest model of relaxation centers around two quantities: the longitudinal relaxation rate and the transverse relaxation rate, quantified by the times T_1 and T_2 respectively. Though a rigorous picture of relaxation in NMR requires the use quantum mechanics [22: Chapter 5, 23, 24], T_1 and T_2 are still valuable concepts which provide insight into the chemical system being studied[†]. Numerous factors affect these quantities, including the rate at which the spin tumbles in space, and its electronic environment. Well-known experiments exist to quantify both of these quantities.

PS figure

Measuring T_1 : Inversion recovery

The inversion recovery experiment involves the simple pulse sequence $180^\circ \rightarrow \tau \rightarrow 90^\circ \rightarrow t^{(1)}$. The initial 180° pulse inverts the magnetisation, so that it is along $-z$. During τ , the spin system undergoes longitudinal relaxation, with the spin state populations at equilibrium gradually being restored. The 90° pulse rotates the magnetisation into the transverse plane, enabling detection. The resultant phase and magnitude of the signal is directly related to the amount of time that longitudinal relaxation is allowed to occur. With $\tau = 0$ s, a signal with maximal amplitude, but a phase of 180° will result[‡]. At the other extreme of very long τ , the spin system will have reverted back to equilibrium, such that a signal which also has maximal amplitude, but a phase of 0° will be realised. By sequentially adjusting τ in a 2D experiment, a series of spectra will be obtained in

[†]Loosely, the T_1 of a spin can be thought of as the inverse of the self-relaxation rate of the spin's \hat{I}_z operator, while T_2 is the corresponding quantity for its \hat{I}_+ operator, equivalent to $\hat{I}_x + i\hat{I}_y$.

[‡]In a T_1 experiment, it is customary to perform phase correction across the series of spectra such that the initial spectrum has absorption-mode peaks with negative integrals (i.e. a phase of 180°). It should be noted that a spectrum with 180° phase is not the expected direct output from the spectrometer, since the relative phase of the FID and the receiver cannot be easily calibrated. **There is probably a better way of explaining this...**

which the intensity of each peak will vary according to

$$\alpha(\tau) = \alpha_\infty \left(1 - 2 \exp\left(-\frac{\tau}{T_1}\right) \right), \quad (3.2)$$

where α_∞ is the intensity of the peak acquired when the spin system has returned to equilibrium. Note that $\alpha(0) = -\alpha_\infty$, as the spin system has been completely inverted, and no time has been allowed for longitudinal relaxation to take place. An example of a series of spectra acquired using an inversion recovery experiment is given by panel d in Figure 3.5.

Measuring T_2 : CPMG

In an analogous fashion to the inversion recovery experiment, by creating transverse magnetisation and leaving it to evolve for different amounts of time, it is possible to determine T_2 . This might lead one to believe that the pulse sequence $90^\circ \rightarrow \tau \rightarrow t^{(1)}$ would be effective for T_2 determination. However the presence of J-modulation would generate undesirable spectra with phase-modulated peaks. As well as this the presence of field inhomogeneities will cause relaxation at a faster rate than anticipated. The effect of field inhomogeneities is incorporated into the “observed” transverse relaxation time, T_2^* . The effects of J-modulation and field inhomogeneity can be nullified if rapid refocussing is applied, by subjecting the spin system to a train of spin echoes. The classic route to T_2 measurement is the Carl-Purcell-Meiboom-Gill (CPMG) experiment[89, 90], comprising $90^\circ_x \rightarrow [\tau \rightarrow 180^\circ_y \rightarrow \tau]_n \rightarrow t^{(1)}$, where the spin echo duration τ is short and fixed, and the number of cycles n can be varied to alter the total evolution time. T_2 attenuates the intensity of the resulting signal according to

$$\alpha(n) = \alpha_0 \exp\left(-\frac{2\tau n}{T_2}\right), \quad (3.3)$$

where α_0 is the intensity where no spin echo cycles were employed, such that the pulse sequence reduces to a standard pulse-acquire experiment.

3.2.2 Diffusion experiments

NMR is well established as a means of determining the rates of diffusion of chemical species[91, 92]. The first showcase for determining translational diffusion coefficients came from Stejskal and Tanner, in which they described the pulsed gradient spin echo (PGSE) pulse sequence[93] (Figure 3.4.a). The PGSE sequence consists of a conventional spin-echo ($90^\circ \xrightarrow{\tau} 180^\circ \xrightarrow{\tau}$ acquire), with pulsed field gradients (PFGs) applied after each of the RF pulses. As a simple overview of how the pulse sequence works, consider a single spin on resonance with the transmitter (i.e. its rotating frame frequency is zero) in a sample tube at position z along the axis collinear with the

main field. After the 90° pulse, the magnetisation will be $-M_y$. During the first PFG, the spin's resonance frequency will become $\omega_{\text{PFG}} = -\gamma g z$, where g is the strength of the PFG. Assuming the gradient is applied for a time δ , the spin will precess by an angle of $\alpha = -\gamma g z \delta$. After the 180° pulse, the spin's magnetisation is as follows:

$$-M_y \xrightarrow{\text{PFG}} -M_y \cos(\alpha) + M_x \sin(\alpha) \xrightarrow{180^\circ} -M_y \cos(\alpha) - M_x \sin(\alpha).$$

Supposing that the spin has moved to a new position $z + \Delta_z$ between the end of the first gradient and the beginning of the second, application of the second gradient causes precession by the angle $\beta = -\gamma g(z + \Delta_z)\delta$:

$$\begin{aligned} & \xrightarrow{\text{PFG}} -M_y \cos(\alpha) \cos(\beta) + M_x \cos(\alpha) \sin(\beta) - M_x \sin(\alpha) \cos(\beta) - M_y \sin(\alpha) \sin(\beta) \\ &= -M_y \cos(\gamma g \delta \Delta_z) - M_x \sin(\gamma g \delta \Delta_z), \end{aligned}$$

In the scenario that the spin has not translated in the z -direction between PFGs ($\Delta_z = 0$), the net effect of the pulse sequence is nothing (except for a loss of signal amplitude through T_2 relaxation). However, if translation does occur, the signal phase is adjusted, as a function of the extent of translation Δ_z . The gradients have effectively been employed to encode the change in position of the spin after a known amount of time. Extending this idea to a system of many identical spins, which will translate by different extents between the PFGs, individual spin contributions to the bulk magnetisation will become dephased, leading to an attenuation of the amplitude of the resulting FID. For species which diffuse at a faster rate, this effect is more severe, such that a more rapid attenuation is anticipated.

Through consideration of the Bloch-Torrey equations[94], which extend the classic Bloch equations to account for the effects of diffusion on magnetisation, the following equation, known as the *Stejskal-Tanner equation*, may be derived:

$$\alpha(g) = \alpha_0 \exp\left(-\gamma^2 \delta^2 g^2 D \left(\Delta - \frac{\delta}{3}\right)\right), \quad (3.4)$$

where $\alpha_0 = \lim_{g \rightarrow 0} \alpha$, γ is the gyromagnetic ratio of the target nucleus (rad MHz T^{-1}), g is the gradient strength ($T m^{-1}$)^{\$}, δ is the duration of each PFG (s), Δ is the delay between the PFGs, often known as the diffusion time (s), and D is the translational diffusion constant of the species giving rise to the signal ($m^2 s^{-1}$). While Equation 3.4 is widely stated in the literature, it is only strictly applicable when the PGSE sequence is used, and PFGs with rectangular amplitude profiles are applied[¶].

^{\$}Gradient strengths are often expressed in units of $G cm^{-1}$, which is equivalent to $10^{-2} T m^{-1}$.

[¶]Rectangular PFGs (i.e. those in which there is an infinitesimal time to rise to full strength, and to fall back to zero) are in fact impossible to achieve as they would require gradient coils with zero inductance.

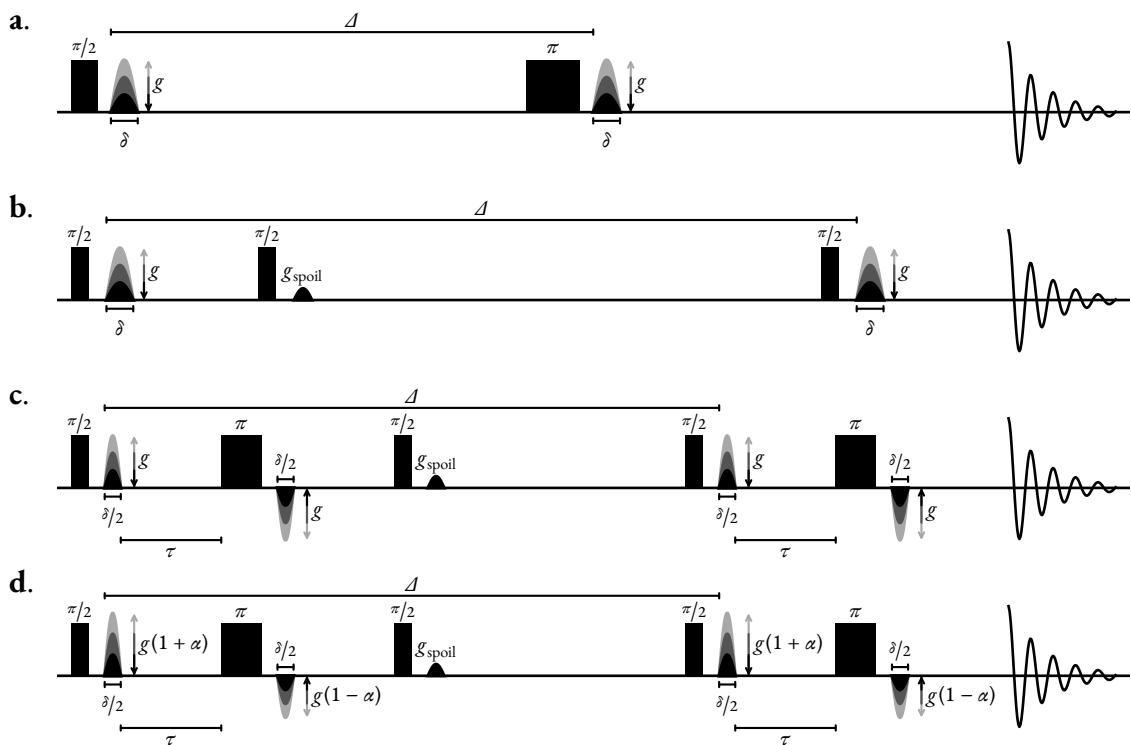


FIGURE 3.4: Pulse sequences used for the determination of translational diffusion constants. **a.** PGSE, **b.** PGSTE, **c.** PGSTEBP, **d.** One-shot DOSY. RF pulses are denoted by solid rectangles. Diffusion-encoding gradients are denoted by sine-bell shapes with varying shades, indicating that the intensity is incremented to create a 2D dataset. Spoiler gradients are denoted by solid black sine-bell shapes.

Tanner introduced a variant of the original PGSE experiment called pulsed gradient stimulated echo (PGSTE)[95] (Figure 3.4.b). Instead of the diffusion period including a 180° pulse, PGSTE features two 90° pulses, with the first being applied shortly after the initial PFG, and the second being applied just before the second PFG. The key difference between this and the PGSE experiment is that decoherence during the diffusion time is dictated by longitudinal relaxation (T_1) rather than transverse relaxation (T_2). PGSTE is therefore favoured in scenarios where $T_1 \ll T_2$, as improved sensitivity is attainable.

Both PGSE and PGSTE employ *monopolar* PFGs for diffusion encoding, in the sense that both diffusion-encoding PFGs are polarised in a single direction. Experiments also exist which employ *bipolar* gradient elements[96, 97], which consist of a PFG, followed by a 180° pulse, and then a second PFG with the opposite polarity to the first. A well-known example is the pulsed gradient stimulated echo with bipolar gradients (PGSTEBP) experiment (Figure 3.4.c). Bipolar gradients are useful in circumstances where it is important to purge the effects of static gradients in the sample, caused by field inhomogeneities. Morris and coworkers have developed the *one-shot* experiment[98] (Figure 3.4.d), which requires a single transient per gradient strength (i.e. there is no requirement for a phase-cycling scheme). This is achieved through the use of bipolar gradients which comprise asymmetrical PFGs with relative powers $1 + \alpha : 1 - \alpha$ for some $0 > \alpha > 1$ (a common value is 0.2).

The most common means of performing a diffusion experiment is to run an appropriate pulse sequence, and vary g across increments. It is virtually always the case that the amplitudes of each signal in an FID abide by the following general form of the Stejskal-Tanner equation:

$$\alpha(g) = \alpha_0 \exp(-cg^2D) \quad (3.5)$$

for some constant c (T s^{-2}). The functional form of c depends on the type of experiment used, as well as the amplitude profile of the PFGs. A consideration of the Bloch-Torrey equations for a given experiment is necessary, with an extensive overview provided by Sinnaeve for most diffusion NMR experiments[99]. In general, c is as follows:

$$c = \gamma^2 \delta^2 \sigma^2 \Delta'. \quad (3.6)$$

σ is the *shape factor* of the PFGs, which is related to the shape factor. Δ' is the effective time that diffusion is allowed to occur. Examples of the value of Δ' include:

$$\text{Monopolar gradients (PGSE, PGSTE)} \quad \Delta + 2(\kappa - \lambda)\delta, \quad (3.7a)$$

$$\text{Bipolar gradients (PGSTEBP)} \quad \Delta + \frac{(2\kappa - 2\lambda - 1)\delta}{4} - \frac{\tau}{2}, \quad (3.7b)$$

One-shot

$$\mathcal{A} + \frac{(\kappa - \lambda)(\alpha^2 + 1)\delta}{2} + \frac{(\delta + 2\tau)(\alpha^2 - 1)}{4}. \quad (3.7c)$$

τ is the delay between the initial PFG and the 180° pulse in experiments with bipolar gradients. The factors σ , λ , and κ are related to the shape function $s(\epsilon) : \epsilon \in [0, 1]$ of the PFG, which describes the variation in the intensity of the gradient as a function of its progression. For a rectangular gradient, $s(\epsilon) = 1 \forall \epsilon$, whereas for a sine-bell gradient, $s(\epsilon) = \sin(\pi\epsilon)$. The cumulative distribution of the shape function is given by:

$$S(\epsilon) = \int_0^\epsilon s(\epsilon') d\epsilon' \quad \forall \epsilon \in [0, 1]. \quad (3.8)$$

The corresponding definition of S for the case of a gradient made of N_g discrete steps with shape $s \in \mathbb{R}^{N_g}$ is

$$S_n = \frac{1}{n} \sum_{i=1}^n s_i \quad \forall n \in \{1, \dots, N_g\}, \quad (3.9)$$

The three factors are given by

$$\sigma = S(1), \quad (3.10a)$$

$$\lambda = \frac{1}{\sigma} \int_0^1 S(\epsilon) d\epsilon, \quad (3.10b)$$

$$\kappa = \frac{1}{\sigma^2} \int_0^1 S^2(\epsilon) d\epsilon, \quad (3.10c)$$

with their discrete counterparts being

$$\sigma = S_{N_g} \quad (3.11a)$$

$$\lambda = \frac{1}{\sigma N_g} \sum_{n=1}^{N_g} S_n = \frac{1}{\sigma N_g} \sum_{n=1}^{N_g} \left(\frac{1}{n} \sum_{i=1}^n s_i \right) \quad (3.11b)$$

$$\kappa = \frac{1}{\sigma^2 N_g} \sum_{n=1}^{N_g} S_n^2 = \frac{1}{\sigma^2 N_g} \sum_{n=1}^{N_g} \left(\frac{1}{n} \sum_{i=1}^n s_i \right)^2 \quad (3.11c)$$

For PFGs with a symmetrical shape, $\lambda = 1/2$. κ is typically equal to or close to $1/3$. It can now be seen that the original Stejskal-Tanner equation (Equation 3.4) comes from plugging Equation 3.7a into Equation 3.6, with parameters for rectangular PFGs: $\sigma = 1$, $\lambda = 1/2$, and $\kappa = \frac{1}{3}$. In many situations, \mathcal{A} dominates in the expression of \mathcal{A}' , and so ensuring the correct form of c could be seen as excessive. However, especially when \mathcal{A} is not orders of magnitude greater than δ , the exact form of \mathcal{A}' used in Equation 3.6 will be extremely important for accurate measurements of D .

Talk about means of determining T_1 , D etc: peak pick and fit amplitudes across increments, DOSY, DECRA/SCORE etc

3.2.3 Methodology

Give algorithm for fit of each oscillator. Include how initial guess is generated for invrec and diffusion, and initial trust radius.

The datasets arising from the experiments described above can be considered in a general fashion. Suppose the experiment of interest is run with $K \in \mathbb{N}$ increments, such that there is a vector $\mathbf{p} \in \mathbb{R}^K$ which gives the pulse sequence variable for each increment. The complete dataset that results from the experiment is expected to take the form $\mathbf{Y} \in \mathbb{C}^{K \times N}$ in which the value of the experimental parameter attenuates the amplitudes of the contributing resonances:

$$y_{k,n} = \sum_{m=1}^M a_{k,m} \exp(i\phi_m) \exp((2\pi i(f_m - f_{\text{off}}) - \eta_m)n\Delta_t). \quad (3.12)$$

$\forall k \in \{1, \dots, K\}$ and $\forall n \in \{0, \dots, N-1\}$. $\mathbf{A} \in \mathbb{R}^{K \times M}$ is a matrix of the oscillator amplitudes across the increments:

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,M} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{K,1} & a_{K,2} & \cdots & a_{K,M} \end{bmatrix} \quad (3.13)$$

A complete parameter vector for the dataset is given by $\theta \in \mathbb{R}^{(K+3)M}$:

$$\theta = [\mathbf{a}_1 \ \cdots \ \mathbf{a}_K \ \boldsymbol{\phi}^T \ \mathbf{f}^T \ \boldsymbol{\eta}^T]^T, \quad (3.14)$$

where \mathbf{a}_k denotes the relevant row in \mathbf{A} . The amplitudes are a function of the experiment parameter with the following form:

$$a_{k,m} = a_{0,m} \mathcal{A}(\psi_m | p_k), \quad (3.15)$$

where $\mathbf{a}_0 \in \mathbb{R}^M$ is a vector of the “maximal” amplitudes for the oscillators, and $\boldsymbol{\psi} \in \mathbb{R}^M$ is a vector of the parameter of interest (diffusion coefficient, relaxation time etc.) for each oscillator. $a_{0,m}$ can be thought of as the largest possible amplitude that could be obtained for oscillator m with the given experiment:

- In an inversion recovery experiment, the largest amplitude is achieved as $\tau \rightarrow \infty$, as the spin system will have returned back to equilibrium prior to the 90° pulse.
- With CPMG experiments, the greatest amplitude will occur when $n = 0$, as no time is designated for transverse relaxation to take place.
- For diffusion experiments, the largest amplitude is achieved when $g = 0$, since no diffusion-induced dephasing of spins will have occurred.

Experiment	p	ψ	$\mathcal{A}(\psi p)$	$\frac{\partial \mathcal{A}(\psi p)}{\partial \psi}$	$\frac{\partial^2 \mathcal{A}(\psi p)}{\partial \psi^2}$
Inversion Recovery	τ	T_1	$(1 - 2 \exp(-\frac{\tau}{T_1}))$	$-\frac{2\tau}{T_1^2} \exp(-\frac{\tau}{T_1})$	$\frac{2\tau}{T_1^3} \exp(-\frac{\tau}{T_1})(2 - \frac{\tau}{T_1})$
CPMG	n	T_2	$\exp(-\frac{2\tau n}{T_2})$	$\frac{2\tau n}{T_2^2} \exp(-\frac{2\tau n}{T_2})$	$\frac{2\tau n}{T_2^3} \exp(-\frac{2\tau n}{T_2})(\frac{2\tau n}{T_2} - 2)$
Diffusion	g	D	$\exp(-cg^2 D)$	$-cg^2 \exp(-cg^2 D)$	$c^2 g^4 \exp(-cg^2 D)$

TABLE 3.2: The various functional forms of \mathcal{A} according to the different amplitude-attenuating NMR experiments considered, along with its first and second derivatives, which are required to extract estimates of ψ using NLP.

The function \mathcal{A} describes how the amplitudes of resonances are attenuated by the experimental variable, and has a form which is intimately linked to the type of experiment. For the experiments described in Sections 3.2.1 and 3.2.2, these forms are provided in Table 3.2.

Estimating amplitude-attenuated datasets

The close relationship between signals across increments means that completely estimating each signal in turn is typically not necessary. Instead, after the first increment is estimated from scratch, yielding $\boldsymbol{\theta}_{k=1} \in \mathbb{R}^{4M}$, the phases, frequencies and damping factors are fixed, and subsequent increments are determined by taking the parameter estimate of the previous iteration, and subjecting it to a NLP routine in which only the amplitudes are allowed to be varied. Thus, determining parameter estimates for each iteration $k \in \{2, \dots, K\}$ is reduced to the problem^{||}

$$\boldsymbol{\alpha}_k = \arg \min_{\boldsymbol{\alpha} \in \mathbb{R}^M} \mathcal{F}(\boldsymbol{\alpha} | \boldsymbol{\phi}_{k=1}, \mathbf{f}_{k=1}, \boldsymbol{\eta}_{k=1}, \mathbf{y}_k). \quad (3.16)$$

An NLP routine can solve this very efficiently, typically in few iterations, on account of the linear dependence of the model with respect to the oscillator amplitudes. The linear dependence also means that second derivatives of the model are all zero (see Equation 2.47a), such that only first derivatives need to be computed to derive an exact Hessian matrix of the fidelity.

Due to the linear dependence on amplitudes, an alternative means of deriving amplitudes for each increment is to determine the following:

$$\boldsymbol{\alpha}_k = \mathbf{Z}^+ \mathbf{y}_k, \quad (3.17a)$$

^{||} \mathcal{F} in Equation 3.16 reads as “the fidelity with respect to the amplitudes $\boldsymbol{\alpha}$, given phases $\boldsymbol{\phi}$, frequencies $\mathbf{f}^{(1)}$, damping factors $\boldsymbol{\eta}^{(1)}$, and FID $\mathbf{Y}[k, :]$ ”. The expression has exactly the same mathematical form as Equation 2.44, though it emphasises that the phases, frequencies and damping factors are no longer variables to be optimised, but fixed parameters.

ALGORITHM 3.1 Routine for estimating a sequence of 1D FIDs which exhibit variation in amplitudes across increments. NLPAMP denotes a routine which is akin to NLP (Algorithm 2.2), except only amplitudes are allowed to be altered, whilst phases, frequencies and damping factors are fixed.

```

1: procedure ESTIMATEAMPATTENUATED(  $\mathbf{Y} \in \mathbb{C}^{K \times N}$ ,  $\mathbf{r}_{\text{interest}} \in \mathbb{R}^2$ ,  $\mathbf{r}_{\text{noise}} \in \mathbb{R}^2$ ,  $M \in \mathbb{N}_0$  )
2:    $\theta_1, \epsilon_1 \leftarrow \text{ESTIMATE1D}(\mathbf{y}_1, \mathbf{r}_{\text{interest}}, \mathbf{r}_{\text{noise}}, M)$ ;                                 $\triangleright$  Estimate first increment
3:    $M \leftarrow \text{len}(\theta_1)/4$ ;
4:    $\theta, \epsilon \leftarrow \mathbf{0} \in \mathbb{R}^{(K+3)M}, \mathbf{0} \in \mathbb{R}^{(K+3)M}$ ;                                 $\triangleright$  Initialise complete parameter vector.
5:    $\theta[:M], \epsilon[:M] \leftarrow \theta_1[:M], \epsilon_1[:M]$ ;                                 $\triangleright$  Amplitudes for first increment.
6:    $\theta[KM:], \epsilon[KM:] \leftarrow \theta_1[M:], \epsilon_1[M:]$ ;       $\triangleright$  Phases, frequencies and damping factors, which are held
   constant across increments.
7:   for  $k = 2, \dots, K$  do
8:      $\tilde{\mathbf{y}} \leftarrow \text{FILTER1D}(\mathbf{y}_k, \mathbf{r}_{\text{interest}}, \mathbf{r}_{\text{noise}})$ ;                                 $\triangleright$  Algorithm 2.3.
9:      $\theta_k, \epsilon_k \leftarrow \text{NLPAMP}(\tilde{\mathbf{y}}, \theta_k)$ ;
10:     $\theta[kM:(k+1)M], \epsilon[kM:(k+1)M] \leftarrow \theta_k[:M], \epsilon_k[:M]$ ;       $\triangleright$  Extract amplitudes.
11:   end for
12:   return  $\theta, \epsilon$ ;
13: end procedure

```

$$\mathbf{Z} = \begin{bmatrix} \exp(i\phi_1) & \cdots & \exp(i\phi_M) \\ \exp(i\phi_1)z_1 & \cdots & \exp(i\phi_M)z_M \\ \vdots & \ddots & \vdots \\ \exp(i\phi_1)z_1^{N-1} & \cdots & \exp(i\phi_M)z_M^{N-1} \end{bmatrix}, \quad (3.17b)$$

$$z_m = \exp((2\pi i(f_m - f_{\text{off}}) - \eta_m)\Delta_t). \quad (3.17c)$$

Determining the parameter of interest

Having generated a complete parameter estimate for the FID, focus subsequently moves to determining the parameters of interest ψ . For each oscillator, the maximal amplitude and parameter of interest are determined by solving the following problem $\forall m \in \{1, \dots, M\}$:

$$\begin{bmatrix} a_{0,m}^{(*)} \\ \psi_{0,m}^{(*)} \end{bmatrix} = \arg \min_{[a_0 \ \psi]^T \in \mathbb{R}^2} \| \mathbf{a}_m - a_0 \mathcal{A}(\psi | \mathbf{p}) \|^2, \quad (3.18)$$

where \mathbf{a}_m corresponds to the m^{th} column of \mathcal{A} . This is another example of a residual sum-of-squares problem, and can also be solved using an NLP routine. The gradient vector and Hessian matrix of the fidelity take very similar functional forms to those for FID estimation (see Equation 2.45a and Equation 2.45b), and are as follows $\forall i, j \in \{0, 1\}$:

$$g_i = -2 \left\langle \mathbf{a}_m - a_0 \mathcal{A}(\psi | \mathbf{p}), \frac{\partial a_0 \mathcal{A}(\psi | \mathbf{p})}{\partial \vartheta_i} \right\rangle \quad (3.19a)$$

$$h_{i,j} = 2 \left(\left\langle \frac{\partial a_0 \mathcal{A}(\psi | \mathbf{p})}{\partial \vartheta_i}, \frac{\partial a_0 \mathcal{A}(\psi | \mathbf{p})}{\partial \vartheta_j} \right\rangle - \left\langle \mathbf{a}_m - a_0 \mathcal{A}(\psi | \mathbf{p}), \frac{\partial^2 a_0 \mathcal{A}(\psi | \mathbf{p})}{\partial \vartheta_i \partial \vartheta_j} \right\rangle \right), \quad (3.19b)$$

$$\mathbb{R}^2 \ni \boldsymbol{\vartheta} = [\alpha_0 \quad \psi]^T, \quad (3.19c)$$

with explicit expressions for the requisite first and second derivatives being

$$\frac{\partial \alpha_0 \mathcal{A}(\psi | \mathbf{p})}{\partial \alpha_0} = \mathcal{A}(\psi | \mathbf{p}), \quad (3.20a)$$

$$\frac{\partial \alpha_0 \mathcal{A}(\psi | \mathbf{p})}{\partial \psi} = \alpha_0 \frac{\partial \mathcal{A}(\psi | \mathbf{p})}{\partial \psi}, \quad (3.20b)$$

$$\frac{\partial^2 \alpha_0 \mathcal{A}(\psi | \mathbf{p})}{\partial \alpha_0^2} = 0, \quad (3.20c)$$

$$\frac{\partial^2 \alpha_0 \mathcal{A}(\psi | \mathbf{p})}{\partial \psi^2} = \alpha_0 \frac{\partial^2 \mathcal{A}(\psi | \mathbf{p})}{\partial \psi^2}, \quad (3.20d)$$

$$\frac{\partial^2 \alpha_0 \mathcal{A}(\psi | \mathbf{p})}{\partial \alpha_0 \partial \psi} = \frac{\partial^2 \alpha_0 \mathcal{A}(\psi | \mathbf{p})}{\partial \psi \partial \alpha_0} = \frac{\partial \mathcal{A}(\psi | \mathbf{p})}{\partial \psi}. \quad (3.20e)$$

The functional forms of the first and second derivatives for the different experiments of interest of \mathcal{A} are given in Table 3.2.

Displaying results

Visualising the results from the routine described above can be done in a similar fashion to diffusion-ordered spectroscopy (DOSY) analysis. For each oscillator in the estimation result, a 2D array is generated, corresponding to the outer product of the FT of the oscillator (\mathbf{s}_m) and a distribution describing the predicted value of ψ (\mathbf{d}_m).

$$\mathbb{R}^{R \times N} \ni \mathbf{S} = \sum_{m=1}^M \mathbf{d}_m \otimes \mathbf{s}_m, \quad (3.21a)$$

$$\mathbf{s}_m = \Re(\text{FT}(\mathbf{x}_m)), \quad (3.21b)$$

$$x_{m,n} = \alpha_{0,m} \exp(i\phi_m) \exp((2\pi i(f_m - f_{\text{off}}) - \eta_m)n\Delta_t), \quad (3.21c)$$

where $R \in \mathbb{N}$ is the number of samples to generate the distribution from. The exact form that the distribution should take is not set in stone, though it should indicate two key pieces of information: (i) its maximum should coincide with the predicted value of ψ_m and (ii) its width should indicate the level of uncertainty associated with the prediction. In this work, the distribution used is a Gaussian distribution with mean ψ_m and standard deviation $c\epsilon_m$, where ϵ_m is the estimation error associated with oscillator m 's parameter of interest,** and $c \in \mathbb{R}_{>0}$ is an arbitrary linewidth

**The errors can be extracted from the Hessian matrix once the NLP routine has reached convergence. See Section 2.3.4 for more information.

factor which can be chosen to ensure clear visibility of the peaks.^{††}

$$d_{m,r} = -\frac{1}{\sqrt{2\pi(c\epsilon_m)^2}} \exp\left(-\frac{(p_r - \psi_m)^2}{2(c\epsilon_m)^2}\right) \quad \forall r \in \{1, \dots, R\}, \quad (3.22a)$$

$$p_r = p_{\min} + \frac{r(p_{\max} - p_{\min})}{R - 1}. \quad (3.22b)$$

p_{\min} and p_{\max} specify the range of values over which to generate the distribution.

3.2.4 Results

“Five multiplets”

Figure 3.5 shows the result of the described method in determining the T_1 values of resonances in a simulated inversion recovery datasets featuring 5 overlapping ddd multiplet structures. The spin systems used to generate the datasets were constructed in a similar way to that described for the “Four Multiplets” example in Section 4.3. In this case however, 5 estimated spins were present instead of 4. Constraints were also placed on the shifts and couplings to ensure that no two oscillators would have frequencies with a difference less than $f_{\text{sw}}^{(1)}/N^{(1)}$. For each spin, a T_1 value was sampled from $\mathcal{U}(1 \text{ s}, 5 \text{ s})$, and a T_2 value was sampled from $\mathcal{U}(0.2 \text{ s}, 0.6 \text{ s})$. The inversion recovery experiment was simulated using SPINACH, with the relaxation phenomena described by the “extended T_1/T_2 approximation” (see Appendix REF). Each dataset was provided AWGN such that the target SNR of the datasets as a whole was 40 dB. The estimation routine outlined by Algorithm 3.1 was applied to the generate a parameter estimate of the region which contained signals from the estimated spins.

Despite heavy overlap between peaks, the routine was successful at assigning each signal in the dataset with a T_1 value that closely agreed with the true value (see panel b.). As is to be expected, in scenarios where little multiplet overlap existed, T_1 predictions tended to be more accurate, with smaller associated errors (see for example the purple and orange multiplets in Run 1. Nevertheless, adequate estimates could still be obtained in cases of severe overlap, especially when the predicted T_1 s for all oscillators associated with a given multiplet are averaged. (see the red, blue and yellow multiplets in Run 1). Particular oscillators for which the estimate of T_1 is particularly far from the true value tend to be associated with large errors, with examples denoted with an asterisk.

^{††}In cases where the estimation error is small, the distribution can be so sharp relative to the resolution used for plotting, that $\max(d_m)$ can be 0 if the peak of the distribution does not coincide with one of the sampling values. In these situations, errors arise due to attempted division by 0 (see Equation 3.21a). Broadening the distribution (i.e. setting $c > 1$) resolves issues with plotting discrete samples.

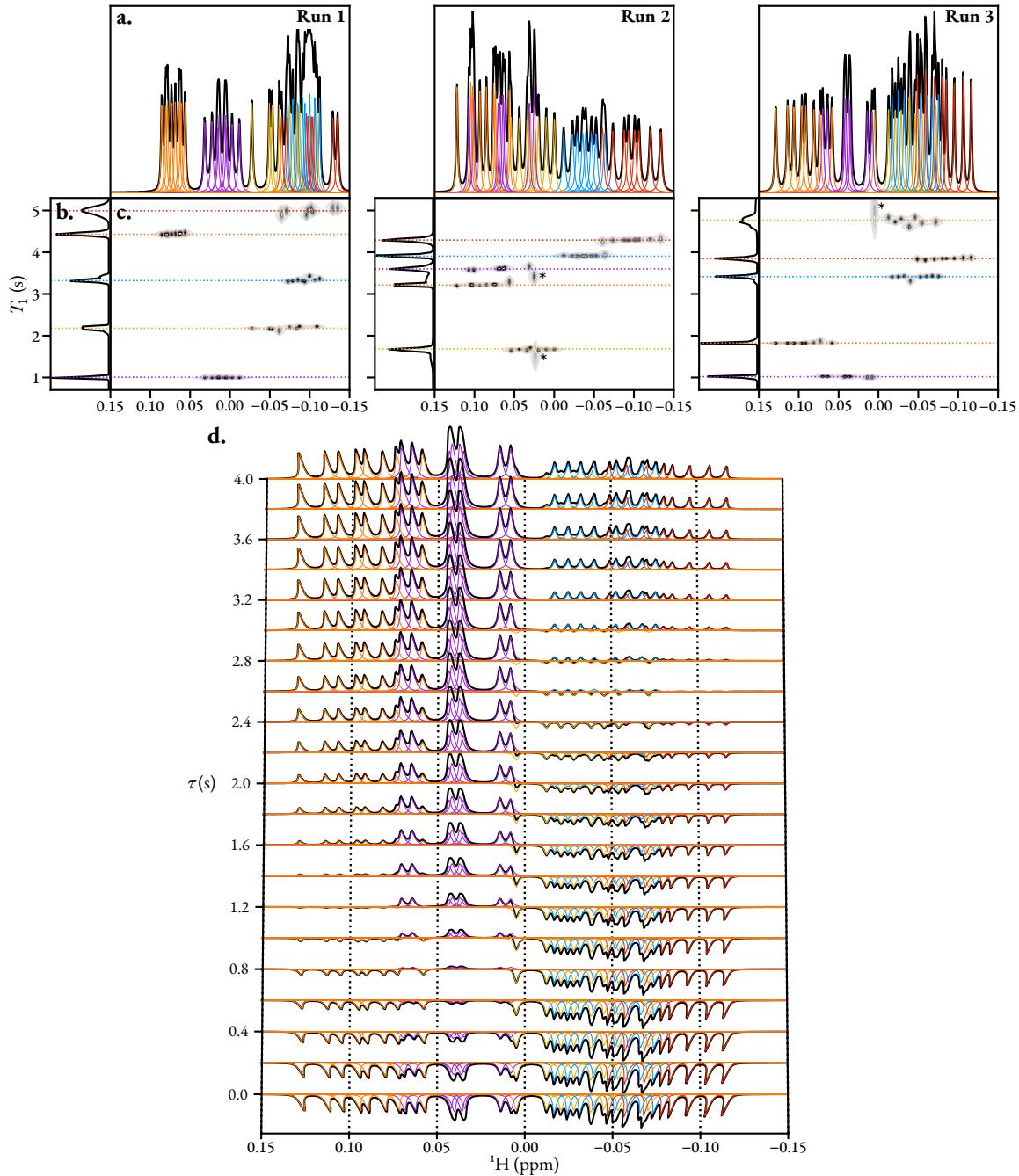


FIGURE 3.5: Three examples of results generated on simulated inversion recovery datasets comprising five ddd multiplet structures. **a.** Plot of the result generated for the first increment ($\tau = 0$ s), with each plot multiplied by -1 . Black: spectrum of the data. Coloured lines: spectra of individual oscillators generated by the estimation routine. Oscillators with the same colour are components of the same multiplet. **b.** Distribution of T_1 values, generated using Equation 3.22, with $p_{\min} = 0.7$ s, $p_{\max} = 5.3$ s, $c = 40$, $R = 128$. **c.** DOSY-style contour plot of the result, generated using Equation 3.21. Dashed horizontal lines denote the true T_1 values for each spin. **c.** Estimation result for each increment for Run 3, illustrating the evolution of the amplitudes of each oscillator with τ .

Andrographolide Diffusion

TODO: Re-run figure generation, with sigma set to 1. Compare with dynamic center

Figure 3.6 shows the result of applying the estimation technique on a oneshot DOSY dataset of andrographolide in unfresh DMSO-d₆ at 298 K. Exposure of the sample to water is evidenced by the broad peak around 3.3 ppm, estimated to have a diffusion constant of $4.57 \times 10^{-10} \text{ m}^2 \text{ s}^{-1}$. On top of this, the acidic hydroxyl protons (B, C, H) of andrographolide show significant line-broadening, and their estimated diffusion coefficients are considerably different compared with those of the non-hydroxyl protons, due chemical exchange with water in the sample[100]. The diffusion profile generated suggests a diffusion constant of andrographolide of $2.54 \times 10^{-10} \text{ m}^2 \text{ s}^{-1}$. The predicted diffusion constant for each estimated oscillator shows decent consistency, especially with oscillators of greater intensity. Lower intensity oscillators - especially those which significantly overlap with other oscillators - tended to be associated with less consistent diffusion constants and larger errors with examples of this phenomenon apparent in panel d of the figure. A few oscillators also show a significant deviation at around 2.5 ppm. This is likely due to the presence of signals that make up a 1:2:3:2:1 quintet due to the presence of partially protonated DMSO. As the data is insufficiently resolved to enable the separation of andrographolide and Dimethyl sulfoxide, (H₃C)₂SO (DMSO) signals in the estimation result, oscillators exist which will have an amplitude profile influenced by both species, leading to an aggregated diffusion constant. As $D_{\text{DMSO}} > D_{\text{andrographolide}}$, the affected oscillators show larger apparent diffusion constants.

Glucose/valine/threonine diffusion

TODO: get result from dynamic center and compare Another example is provided by Figure 3.7, where the estimation routine was applied to a diffusion dataset derived from a sample comprising the molecules L-valine ($M_r = 117.148 \text{ g mol}^{-1}$), L-threonine ($M_r = 119.120 \text{ g mol}^{-1}$), and D-(+)-glucose ($M_r = 180.156 \text{ g mol}^{-1}$) dissolved in D₂O at 298 K. Not included in the figure is the result for the water signal, from which a diffusion constant of $1.88 \times 10^{-9} \text{ m}^2 \text{ s}^{-1}$ was determined. The routine was able to achieve separation of the three species in the sample. Predicted diffusion coefficients for valine and threonine were $6.20 \times 10^{-10} \text{ m}^2 \text{ s}^{-1}$ and $6.39 \times 10^{-10} \text{ m}^2 \text{ s}^{-1}$. For glucose, the situation is complicated by the presence of two major forms, α-D-glucopyranose and β-D-glucopyranose^{†‡}, due to anomeration[101: Chapter 3]. There is some evidence of separation of these anomers, principally due to the downfield doublets at 5.15 ppm (α) and 4.56 ppm (β), which have estimated diffusion coefficients of $5.47 \times 10^{-10} \text{ m}^2 \text{ s}^{-1}$ and $5.34 \times 10^{-10} \text{ m}^2 \text{ s}^{-1}$, respectively. Beyond these however, the other estimated oscillators corresponding to glucose have associated errors which are too large for clear resolution of the two anomers.

^{†‡}The equilibrium mixture in water comprises 38% of the α isomer and 62% of the β isomer. A tiny amount of the open-chain form will also be present, though in a negligible quantity. This is evidenced by the spectrum in Figure 3.7.a, where the relative integrals of the doublets at 5.15 ppm and 4.56 ppm agree with this ratio.

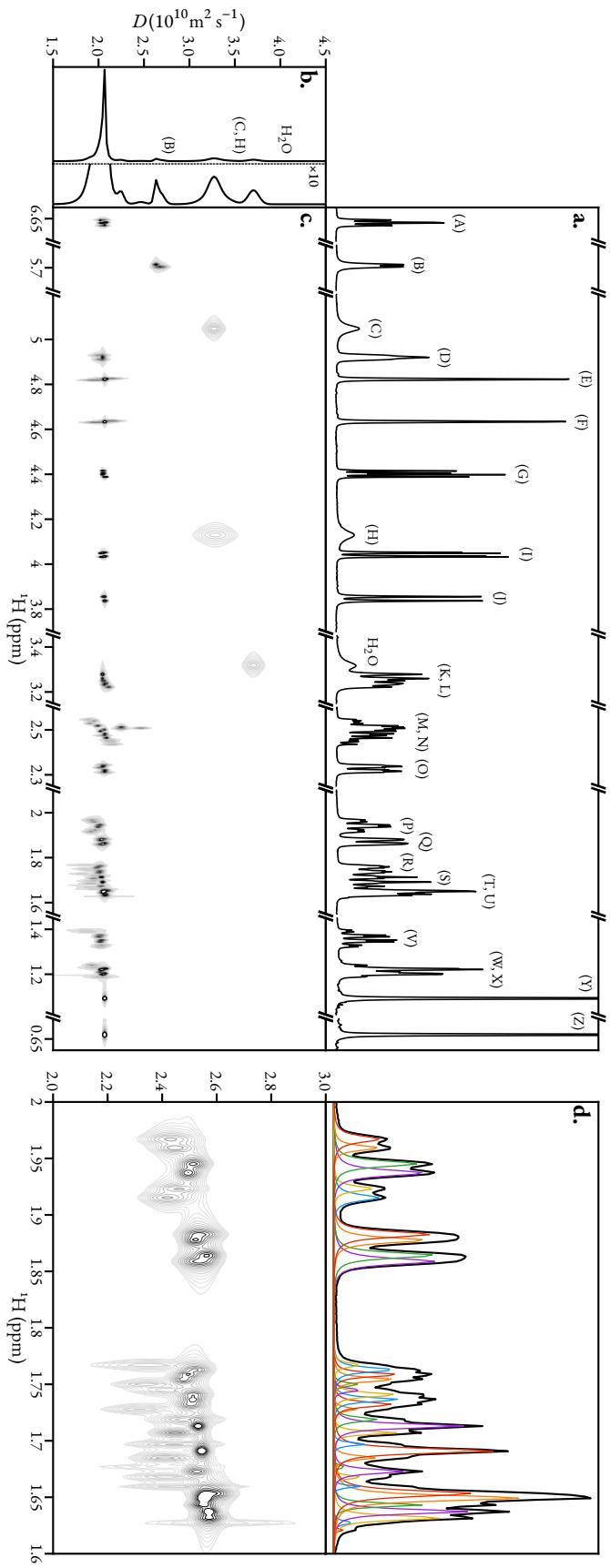


FIGURE 3.6: Result of estimating a Oneshot DOSY dataset of andrographolide in unfresh Deuterated DMSO (DMSO-d₆). **a.** 1D spectrum. **b.** Diffusion profile obtained by summing the contour plot in c. along the x-axis. **c.** Contour plot mapping estimated oscillators to diffusion constants, with $p_{\min} = 2 \times 10^{-10} \text{ m}^2 \text{ s}^{-1}$, $p_{\max} = 5 \times 10^{-10} \text{ m}^2 \text{ s}^{-1}$, $c = 2.5$, $R = 128$. **d.** Magnified view of the 2 ppm to 1.6 ppm spectral range, with estimated oscillator peaks plotted.

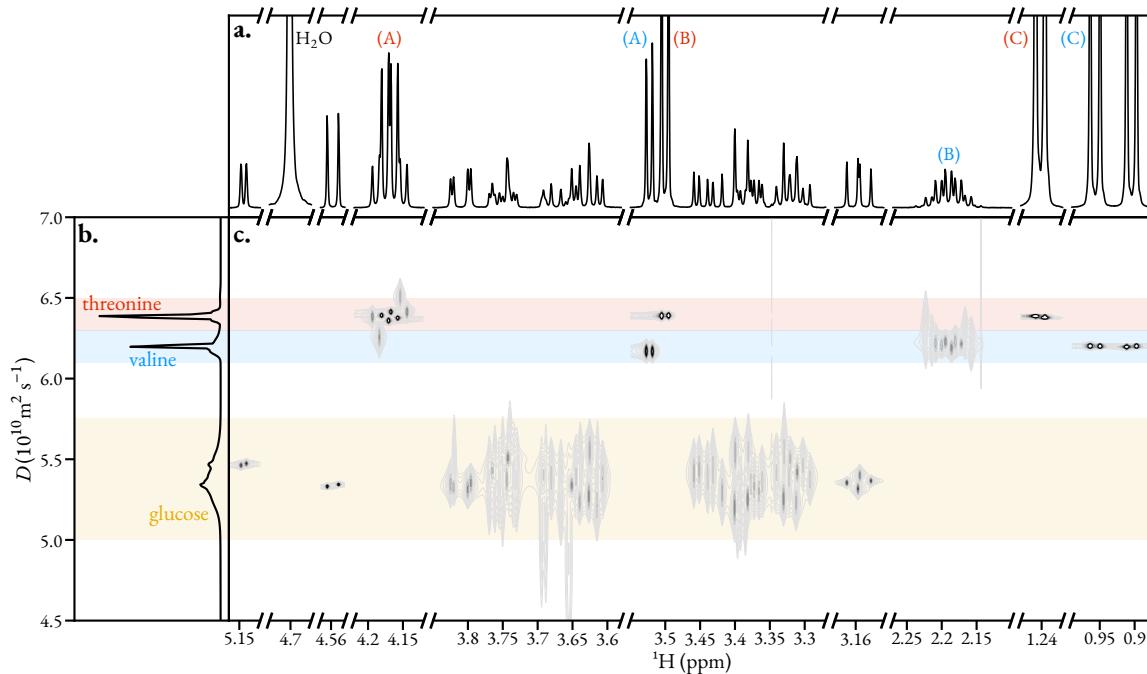


FIGURE 3.7: Result of estimating a diffusion dataset for a mixture of L-threonine, L-valine and D-(+)-glucose in D_2O . **a.** 1D spectrum, taken from the first FID of the diffusion dataset. **b.** Diffusion coefficient distribution. **c.** DOSY-style plot of chemical shifts vs diffusion constant, generated using Equation 3.22, with $p_{\min} = 4.5 \times 10^{-10} \text{ m}^2 \text{ s}^{-1}$, $p_{\max} = 7 \times 10^{-10} \text{ m}^2 \text{ s}^{-1}$, $R = 256$, and $c = 1.5$.

As is to be expected, signals which are of greater intensity, and which are more clearly resolved enable the determination of diffusion coefficients with lower errors. A clear example of this behaviour can be recognised when considering the valine result (see the area shaded blue in panel c). The oscillators which lead to diffusion constants with the lowest errors correspond to the high intensity doublet of doublets around 0.95 ppm, resulting from six equivalent protons from two methyl groups. Far greater uncertainty is observed for the predictions associated with proton (B), which has a doublet of septets structure, featuring many low-intensity signals.

Application of multivariate methods on the dataset was unsuccessful at extracting diffusion information for the separate components. Applying **DECRA!** (**DECRA!**) and **SCORE!** (**SCORE!**) with 2 components led to the separation of the water signal from the rest of the dataset with the two components being associated diffusion coefficients of $6.31 \times 10^{-10} \text{ m}^2 \text{ s}^{-1}$ and $1.88 \times 10^{-9} \text{ m}^2 \text{ s}^{-1}$. Using more than two components produced results with spurious components **Ask about this**.

3.3 Phased broadband spectra from chirp excitation

The are numerous nuclei of value in NMR with very wide chemical shift ranges, including ^{13}C , ^{19}F (of particular interest in the pharmaceutical industry) ^{31}P , ^{195}Pt . Attaining spectra covering the entire chemical shift range of such spins for use in quantitative applications is challenging due

to off-resonance effects, which severely alter the amplitudes and phases of resonances with frequencies far from the transmitter frequency[22: Section 3.4.1]. One popular means of achieving broadband excitation, in which a consistent amplitude- and phase-profile across a spectral window of tens or even hundreds of kHz is achieved, is to use frequency-swept (FS) pulses, during which the frequency of RF irradiation varies with time[102]. One of the most common classes of FS pulses are those where the variation of frequency with time is linear, with such pulses commonly referred to as *chirp* pulses. The application of a single 90° chirp pulse to achieve broadband excitation, while simple, yields spectra with undesirable phase behaviour, on account of resonances with different frequencies being excited at different moments in time. There are well-established methods for overcoming this using pulse sequences featuring an initial excitation, followed by one or more refocussing FS pulses[103–107].

With knowledge of the form of the chirp pulse, the expected phase of a particular resonance is determinable, and in this section, it will be shown that well-phased spectra can be obtained from excitation with a single FS pulse when appropriate post-processing of the FID is employed. The main advantage of being able to derive spectra with desirable features from a single chirp excitation experiment is the fact that ultra-broadband spectra can be generated using with a far shorter pulse sequence than state of the art methods such as chirped, ordered pulses for ultra-broadband spectroscopy (CHORUS)[106, 107], where both a 90° chirp pulse, and two 180° chirp pulses are applied. Spectra with greater intensity, as which could include broad resonances from slowly tumbling species could therefore be realised. Here, a description of the technique is presented, followed by an illustration of its performance on a simulated and an experimental dataset. This work is fairly nascent, and while the initial results presented here show promise, there is yet to be consideration on samples of greater interest.

3.3.1 Chirp excitation

Here, focus is limited to chirp pulses which sweep from low to high frequencies. Such a pulse is parameterised by its duration τ_p (s), excitation bandwidth ΔF (Hz), and RF “amplitude” ω_{RF} (Hz). The frequencies that the pulse sweeps through are in the range $[f_{\text{off}} - 1/2\Delta F, f_{\text{off}} + 1/2\Delta F]$, and the rate at which the frequency of the chirp is increased (the sweep rate) is given by $\Delta F/\tau_p$. Figure 3.8 provides an illustration of a single chirp excitation experiment. After application of the chirp pulse, there is typically a short pre-scan delay τ_{del} , usually on the order of a few μs , prior to the start of acquisition. While the pre-scan delay can be determined if an intimate knowledge of the spectrometer hardware is known, this varies from instrument to instrument, and is not trivial to ascertain. It is this delay which induces a first-order phase shift in NMR experiments. The

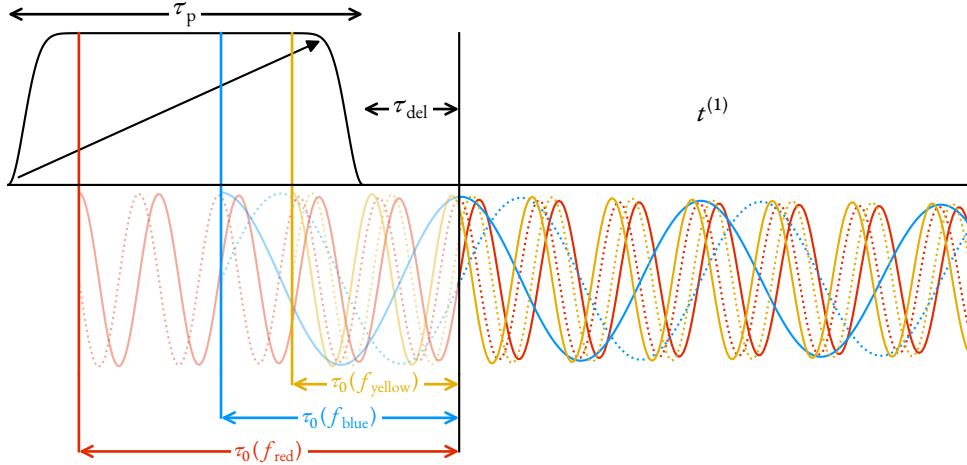


FIGURE 3.8: An illustration of an experiment comprising a single chirp pulse sweeping low to high frequencies of duration τ_p , followed by a pre-scan delay period or τ_{del} , prior to acquisition. The fate of three resonances with different frequencies is denoted, with $f_{\text{red}} < f_{\text{blue}} < f_{\text{yellow}}$. Each resonance is excited at different points in time, with lower frequency resonances being excited earlier, such that each resonance is allowed to evolve for different amounts of time prior to acquisition (τ_0). The resulting FID possesses quadratic phase behaviour. Coloured oscillations denote the evolution of each resonance, with solid and dashed lines representing real and imaginary components, respectively. It is assumed that the 90° chirp rotates each resonance to be initially in phase with the receiver.

various pulse parameters are inter-related as follows[107, 108]:

$$\omega_{\text{RF}} = \sqrt{\frac{\Delta F Q}{2\pi\tau_p}}, \quad (3.23)$$

where $Q \in \mathbb{R}_{>0}$ is the *adiabaticity factor*. For a pulse with flip angle $\beta < 180^\circ$, Q is related to β via

$$Q = \frac{2}{\pi} \ln \left(\frac{2}{\cos(\beta) + 1} \right), \quad (3.24)$$

such that an appropriate pulse to achieve a flip angle of 90° requires selecting a combination of ω_{RF} , ΔF , and τ_p which satisfies $Q \approx 0.441$ ^{§§}. For a pulse with sufficiently low ω_{RF} — which requires a sufficiently large pulse duration for a given excitation bandwidth — it is reasonable to assume that the chirp pulse induces an instantaneous 90° rotation at the point of resonance, as illustrated in Figure 3.8. As such, resonances with different Larmor frequencies evolve for different amounts of time prior to the start of acquisition, according to **Double check this with Ali: his draft has + for last term, rather than -. I have -, as this makes τ_0 larger for frequencies less than the**

^{§§}The combination used in examples in this work are $\omega_{\text{RF}} \approx 16.8 \text{ kHz}$, $\Delta F = 400 \text{ kHz}$, $\tau_p = 100 \mu\text{s}$

transmitter, as I would expect for a low-to-high sweep.

$$\tau_0(f) = \tau_{\text{del}} + \frac{\tau_p}{2} - \frac{(f - f_{\text{off}})\tau_p}{2\Delta F}. \quad (3.25)$$

$\tau_{\text{del}} + \tau_p/2$ is the amount of time between excitation and detection for the on-resonance case, in which excitation occurs exactly halfway through the pulse. Resonances with an frequency smaller than the transmitter are excited earlier and hence have a larger τ_0 , while the converse is true for resonances with greater frequencies. The resulting overall phase as a function of frequency can be approximated as[107] **Double check sign for quadratic term**

$$\phi(f) = \phi_0 + 2\pi \left(\tau_{\text{del}} + \frac{\tau_p}{2} \right) (f - f_{\text{off}}) - 2\pi \left(\frac{\tau_p}{2\Delta F} \right) (f - f_{\text{off}})^2. \quad (3.26)$$

One might assume that it is possible to generate phased spectra by simply applying phase correction to the spectrum, via

$$s_\phi(f) = s(f) \exp(-i\phi(f)). \quad (3.27)$$

While the quadratic phase behaviour of peaks is corrected by doing this, another issue with the dataset is not addressed. For any resonance, the signal that is detected can be thought of as the difference between two signals: (a) the “complete” signal, which starts at the time of excitation, and (b) a “truncated” signal which is identical to the complete signal before acquisition, and which comprises zeros once acquisition has begun. The linear nature of the FT dictates that the resulting delayed-acquisition spectrum comprises the difference between the FTs of the complete signal and the truncated signal. The FT of a severely truncated FID is well approximated as a broad sinc function with its maximum at the resonance frequency. The appearance of the sinc “wiggle” depends on the gap between excitation and acquisition. Resonances of lower frequencies, will exhibit deeper, narrower artefacts since the signal is more significantly truncated. The result of applying quadratic phase correction is therefore a spectrum of well-phased peaks, but with major baseline distortions, particularly to the low-frequency end. Panel b in Figures 3.9 and 3.10 both provide an example of this phenomenon.

3.3.2 Methodology

Both the quadratic phase behaviour and delay-induced baseline distortions can be resolved if an estimate of the FID’s parameters is obtained. This enables the construction of an FID featuring oscillators which are back-propagated, such that they begin not at the point of acquisition, but at the point of excitation. The appropriate start time for an oscillator with frequency $f^{(1)}$ is therefore

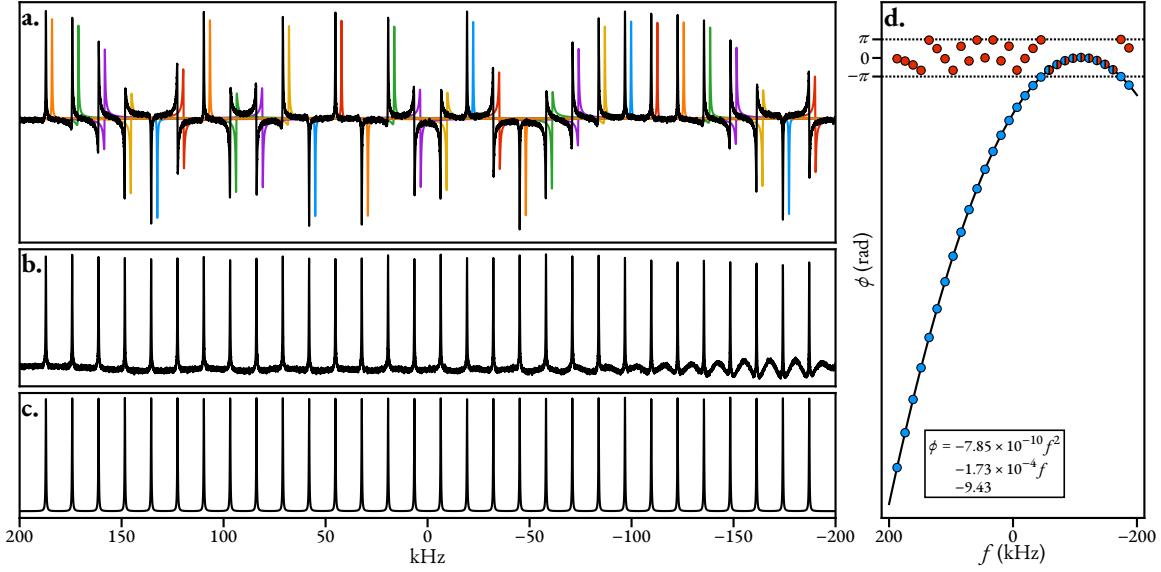


FIGURE 3.9: Comparison of quadratic phase correction vs frequency-dependent back-propagation in treating simulated single-chirp excitation data. **a.** Simulated spectrum for a spin system comprising 30 spins with uniformly-separated resonance frequencies. The data was generated with $N = 2^{14}$, $f_{\text{sw}} = 400$ kHz, $f_{\text{off}} = 0$ Hz, $\tau_p = 100$ μ s, $\tau_{\text{del}} = 0$ s, $\Delta F = 400$ kHz. Coloured lines depict individual signals generated using the MPM. **b.** Spectrum generated using quadratic phase correction, with Equation 3.27. **c.** Spectrum generated from estimation using the MPM, and back-propagation. **d.** Estimated phases of each oscillator as a function of frequency. Red points: phases wrapped within the range $(-\pi, \pi]$. Blue points: the same phases, adjusted by addition of a suitable multiple of 2π to each red point in order to display the quadratic dependence of the phases. Black curve: quadratic fit of the blue points.

given by $-\tau_0$, with τ_0 defined in Equation 3.25. The resulting corrected FID $\mathbf{y} \in \mathbb{C}^N$ is defined as

$$\mathbf{y}_n = \sum_{m=1}^M a_m \exp(i\phi_m) \exp((2\pi i(f_m - f_{\text{off}}) - \eta_m)(n\Delta_t - \tau_0(f_m))). \quad (3.28)$$

This concept has similarities to the use of LP in order to back-propagate an FID to correct for corrupted initial points. A holistic approach such as LP — in which the behaviour of the FID as a whole over time is determined — is not of use in this application, as each signal component must be treated differently according to its frequency.

In this work so far, it has been assumed that all oscillators that make up the data are of the same phase. Of course this isn't the case for signals generated from single-chirp excitation. As such, it is inappropriate to incorporate the variance of oscillator phases in fidelity for NLP. For the examples presented in this work, NLP was not applied; the direct output of the MPM was used as the estimate of the FIDs parameters.

3.3.3 Results

Figures 3.9 and 3.10 present comparisons between the application of quadratic phase correction and the proposed back-propagation procedure. In Figure 3.9, a simulated dataset is considered, comprising 30 evenly-spaced signals, and generated using Equation 3.28 with $-\tau_0(f_m)$ replaced with $+\tau_0(f_m)$. Other relevant parameters used are stated in the caption. For the purposes of clarity, a very large damping factor (1000 s^{-1}) was assigned to each oscillator, as this augments the baseline distortions in the spectrum. AWGN was added to the FID, with a target SNR of 25 dB.

The spectrum after quadratic phase correction, in panel b, exhibits the typical baseline distortions as discussed, with more intense, narrower baseline distortions associated with lower-frequency resonances. The MPM was used to estimate the FID parameters, with a truncated signal comprising only the first 2048 points considered. Performing the MPM on a signal with 2^{14} points would take (a) a long time, and (b) require a very large amount of RAM (see panels a and b of Figure 2.3). Consideration of the first 2048 points is justifiable here, since all resonance frequencies are spaced reasonably far apart, meaning each signal in the FID becomes resolvable from the others early on into evolution. The spectrum generated via back-propagation is presented in panel c, where a well-phased spectrum without baseline distortion could be generated. The variation of the estimated oscillator phases against their frequencies is plotted in panel d, with the quadratic dependence clearly illustrated. Fitting the blue points in panel d to a quadratic function yielded a second-order coefficient of $-7.85 \times 10^{-10}\text{ rad s}^2$, in agreement with the expected value of $-2\pi(\tau_p/2\Delta F)$.

Figure 3.10 features an experimental dataset, acquired from a sample of 1% Gd-doped H_2O in D_2O . The dataset was acquired using a 2D experiment, in which the transmitter offset was adjusted for each increment. As such, FIDs with a single resonance of differing frequency from H_2O are produced across the increments, which when summed lead to the spectrum in panel a of Figure 3.10.

As τ_{del} is not a readily known parameter, only second-order phase correction was applied to the spectrum in panel a (i.e. the first-order correction in Equation 3.26 was not carried out). This left a spectrum with a first-order phase distortion, which was manually corrected. An analogous approach was taken to acquire the spectrum in panel c through the back-propagation approach; τ_0 in Equation 3.28 was replaced with

$$\tau'_0 = -\frac{(f - f_{\text{off}})\tau_p}{2\Delta F}. \quad (3.29)$$

First-order phase correction was applied to the resulting spectrum to yield the final result. As with the simulated case, severe baseline distortions are apparent when quadratic phase correction is applied (panel b) with the most severe dips exhibited by low-frequency resonances. Again, estimation-based back-propagation yields a far cleaner spectral baseline.

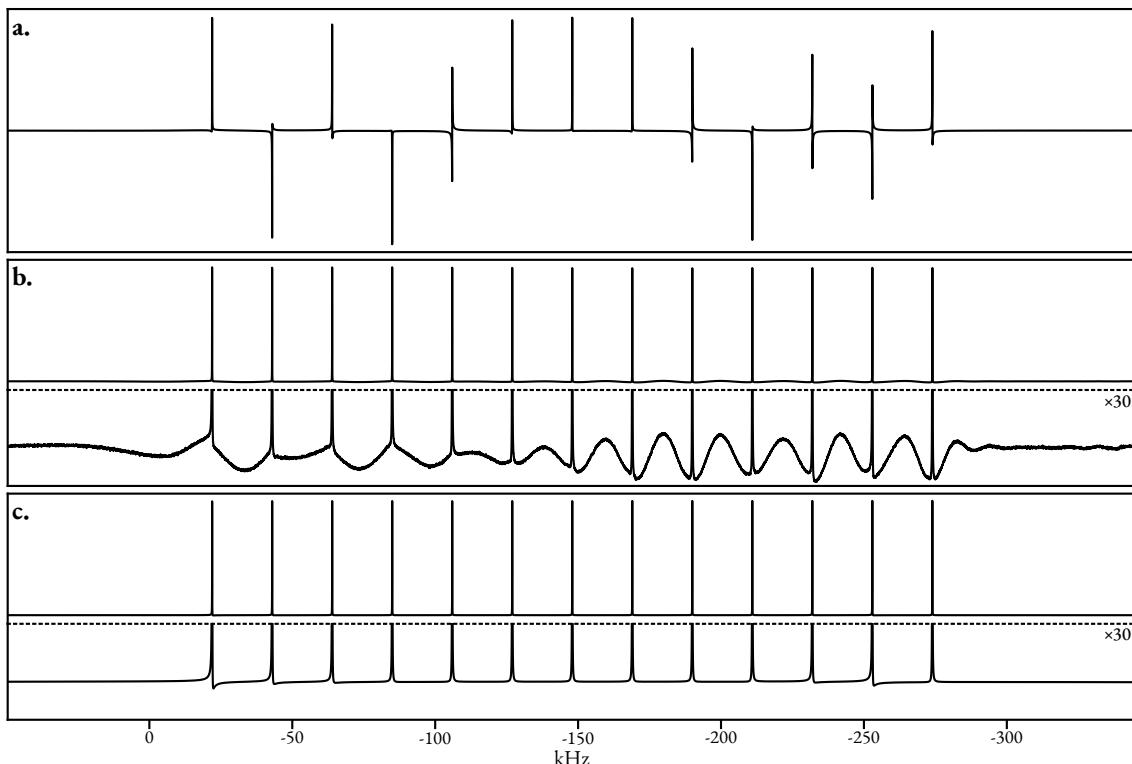


FIGURE 3.10: Comparison of quadratic phase correction vs frequency-dependent back-propagation in treating experimental single-chirp excitation data generated from a sample of 1% Gd-doped H₂O in D₂O. **a.** Spectrum generated directly from the acquired FID. **b.** Spectrum generated using quadratic phase correction. **c.** Spectrum generated from estimation using the MPM, followed by frequency-dependent back-propagation.

3.4 Summary

TODO

PURE SHIFT SPECTRA FROM 2D J-RESOLVED ESTIMATION

4

Two key features of the NMR experiment for which improvements are constantly being sought are sensitivity and resolving power. There are numerous means of enhancing sensitivity, such as the use of magnets with higher field strengths[109] ($\text{sensitivity} \propto B_0^{3/2}$) and cryogenic probes[25], as well as simply increasing the number of scans ($\text{sensitivity} \propto NS^{1/2}$). **There is a limitation to this: can't find the paper on it though.** However, few means of achieving better resolution exist beyond increased field strengths ($\text{resolution} \propto B_0$). Significant interest has therefore been given to the development of techniques which generate broadband homodecoupled (*pure shift*) spectra, in which the effects of homonuclear scalar couplings are absent from the data. While often valuable for structural assignment purposes, the influence of scalar couplings can lead to spectra which are too crowded for meaningful insights to be gleamed. While it is commonplace to decouple heteronuclear couplings at the point of FID acquisition[110–112], homonuclear decoupling is far more challenging. At the time of writing, there are a number of well-established pure shift experiments, which involve running a 2D pulse sequence, and concatenating the initial sections of each FID, in a process referred to as “chunking”[113–115]. The key drawback of all of these techniques is that the resultant pure shift signal is considerably less sensitive relative to a standard pulse-acquire experiment, since only a fraction of the available spin magnetisation contributes that which is detectable by the probe.

In this chapter, a method for deriving pure shift spectra indirectly via the estimation of 2DJ datasets is presented, named *computer-assisted undiminished-sensitivity protocol for ideal decoupling* (CUPID). It is illustrated that by extracting the parameters which describe a 2DJ dataset, a pure shift spectrum can be produced without the signal loss associated with all experimental pure shift methods, and with desirable absorption-mode lineshapes.

4.1 Pure Shift NMR

In this section, a survey of some of the most prominent procedures for producing pure shift spectra are presented.

4.1.1 The 2D J-resolved Experiment

The 2DJ experiment[36, 37] provided the first means of achieving pure shift spectra. It has a simple pulse sequence:

$$90^\circ \xrightarrow{t^{(1)}/2} 180^\circ \xrightarrow{t^{(1)}/2} t^{(2)}.$$

After excitation of magnetisation onto the transverse plane, the indirect dimension evolution consists of a spin echo, with acquisition following immediately afterwards. Fourier transformation in both dimensions leads to a spectrum in which only scalar couplings contribute in $F^{(1)}$, as the chemical shifts are refocussed by the spin echo, while both scalar couplings and chemical shifts contribute in $F^{(2)}$. An FID generated by the 2DJ experiment is hypercomplex, taking the form of Equation 1.22 with $D = 2$ and $\zeta^{(1)} = \exp(i\cdot)$, i.e.

$$\gamma_{n^{(1)}, n^{(2)}} = \sum_{m=1}^M a_m \exp(i\phi_m) \exp\left(\left(2\pi i f_m^{(1)} - \eta_m^{(1)}\right) n^{(1)} \Delta_t^{(1)}\right) \times \exp\left(\left(2\pi i \left(f_m^{(2)} - f_{\text{off}}\right) - \eta_m^{(2)}\right) n^{(2)} \Delta_t^{(2)}\right) + w_{n^{(1)}, n^{(2)}}. \quad (4.1)$$

The transmitter offset term has been neglected in the indirect dimension, since chemical shift evolution does not occur. For each signal in the FID, the indirect- and direct-dimension frequencies are intimately linked. Consider a 2DJ dataset generated by a spin system with S distinct spins. The signals giving rise to a particular spin $s \in \{1, \dots, S\}$ form a grouping $G_s \subset \{1, \dots, M\}$. All of the signals in G_s have (angular) frequencies given by

$$2\pi f_m^{(1)} = \Delta\omega_m, \quad (4.2a)$$

$$2\pi f_m^{(2)} = \omega_{0,s} + \Delta\omega_m, \quad (4.2b)$$

$\forall m \in G_s$, where $\omega_{0,s}$ is the Larmor frequency of the spin, and $\Delta\omega_m$ is the displacement of the signal from $\omega_{0,s}$, as a result of J-couplings*. Due to the relationship between the direct- and indirect-dimension frequencies, all signals which are part of the same multiplet lie along a line which bisects (i.e. makes a 45° angle with) both the $F^{(1)}$ and $F^{(2)}$ axes, as depicted in Figure 4.1.a.

One limitation of the 2DJ experiment is the fact that spectra with pure absorption lineshapes

* $\Delta\omega_m$ will be a linear combination of all the scalar couplings associated with the spin giving rise to the signal, with all the coefficients being $\pm 1/2$.

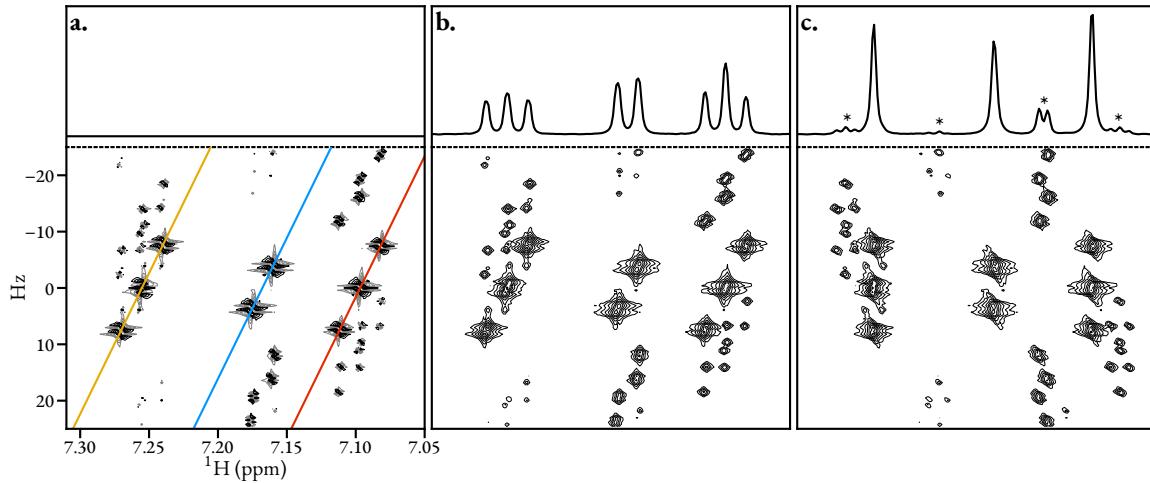


FIGURE 4.1: Region of a simulated 2DJ spectrum of strychnine. Each panel depicts the spectrum following different processing procedures. Below: contour plots of the spectrum. Above: the summation of the spectrum along the indirect (y) axis. **a.** Spectrum produced by applying sine-bell apodisation followed by FT in both dimensions. Coloured lines denote 45° cross-sections along which the present multiplet structures lie. **b.** Magnitude-mode spectrum. **c.** Spectrum generated after application of a 45° shear on the magnitude-mode spectrum. Peaks marked with asterisks panel c arise from the presence of strong coupling artefacts.

cannot be produced. This is since, due to the absence of a mixing period, it is not possible to produce a complementary pair of phase- or amplitude-modulated FIDs, which are required to nullify dispersive contributions (see Section 1.3.2). The FT of a 2DJ FID produces a spectrum with phase-twist peaks (Figure 4.1.a). As with other experiments which produce hypercomplex signals, such as COSY, the data is conventionally displayed in “magnitude-mode” (Figure 4.1.b) in which the absolute value of each point in the spectrum is plotted. A pure shift spectrum is generated from the 2DJ spectrum by performing a 45° shear — often referred to as a tilt — on the spectrum array, leading to the separation of chemical shifts and scalar couplings onto orthogonal axes (Figure 4.1.c). Each slice through the direct dimension of the 2DJ spectrum is subjected to a right circular rotation such that

$$s_{n^{(1)}, n^{(2)}}^{\text{tilt}} = s_{n^{(1)}, n^{(2)'}} \quad (4.3a)$$

$$n^{(2)'} = \left(n^{(2)} + \left\lfloor \frac{f_{\text{sw}}^{(1)} N^{(2)}}{f_{\text{sw}}^{(2)} N^{(1)}} \left(\frac{N^{(1)}}{2} - n^{(1)} \right) \right\rfloor \right) \bmod N^{(2)}. \quad (4.3b)$$

This achieves the mapping $s(F^{(1)}, F^{(2)}) \rightarrow s(F^{(1)}, F^{(2)} - F^{(1)})$, which leads to a spectrum in which all peaks arising from a given spin reside at the same direct-dimension frequency. The effectiveness of the shear is maximised when both $f_{\text{sw}}^{(2)}/f_{\text{sw}}^{(1)}$ and $N^{(2)}/N^{(1)}$ are powers of 2 **check this**. Summing the sheared spectrum along $F^{(1)}$ leads to the pure shift spectrum. If the spectrum wasn't in magnitude-mode, shearing and summing would lead to the absorptive and dispersive compo-

nents of the spectrum cancelling each other out, such that a vector of noise would be obtained. With a magnitude-mode spectrum, the process leads to undesirable pure shift spectra with broad “wings” on account of the presence of dispersive character, and non-linearities. These effects can be suppressed by appropriate processing to make the FID envelope symmetric in both dimensions, such as with sine-bell apodisation or pseudo-echo reshaping[116], though this results in a significant reduction in sensitivity being incurred, along with distortions in relative peak amplitudes.

Another feature which limits the effectiveness of the 2DJ experiment to produce pure shift spectra are *strong coupling artefacts*[†], which arise due to mixing effects induced by the 180° pulse in the 2DJ sequence[117, 118]. Examples of these are seen in the spectra of Figure 4.1, on account of the three spins giving rise to the signals seen being strongly coupled. These artefacts always have direct-dimension frequencies which match those of the conventional signals in the spectrum — a feature which will be exploited in the CUPID procedure — however they do not lie along the same 45° cross sections. As a result, the final spectrum produced by shearing and summing will feature extra low intensity signals that do not agree with the chemical shift of a particular spin (see peaks marked with asterisks in Figure 4.1.c).

4.1.2 The Zanger-Sterk Method

Zanger and Sterk introduced a pulse sequence element which achieves *slice-selective excitation*, by applying a low RF power 180° pulse[‡] in the presence of a PFG along the z -axis[120]. Such an element excites a given spin only in a narrow range of heights in the sample, as the PFG induces a shift in resonance frequency according to $\Delta\omega(z) = \gamma g z$, where g is the magnitude of the PFG. By placing a hard 180° pulse adjacent to the selective pulse, the “active” spin in a given slice is rotated by 360° (i.e. no net rotation), while all other (“passive”) spins are only rotated by 180°. Placing such a element in the middle of the $t^{(1)}$ evolution therefore achieves refocussing of the J-couplings associated with the active spin[121]. In order to achieve effective decoupling of any given pair of spins, it is necessary that the bandwidth of the selective π -pulse is smaller than the difference in their Larmor frequencies. However, with more selective pulses, a smaller proportion of the available spin magnetisation will contribute to the final FID, and hence sensitivity will be diminished[§]. Therefore a trade-off exists between effective decoupling of all spins, and achieving the greatest sensitivity possible. In the case of strong coupling, the Zanger-Sterk (ZS) method tends to perform poorly relative to other options for this reason. The ZS element has been utilised in order to generate 2DJ datasets comprising phase-modulated pairs, enabling the generation of

[†]As stressed in [117], these are not strictly artefacts, but rather genuine signals, which are expected to be present in the 2DJ dataset. Despite this, the term is widespread in the literature.

[‡]Conventionally, a R-SNOB pulse is used[119].

[§]The reduction in sensitivity is $\propto \Delta F / \gamma g l_z$, where ΔF is the selective pulse bandwidth, and l_z is the length of the sample lying within the receiver coil (≈ 1.5 cm).

pure absorption-mode spectra[122]. Pure shift spectra with far more desirable lineshapes can be achieved relative to using a typical magnitude-mode spectrum 2DJ, though with a significant loss of sensitivity.

4.1.3 The BIRD Method

The bilinear rotation decoupling (BIRD) pulse sequence element[123, 124] also takes advantage of the idea of selectively inverting passive spins, while leaving active spins unaffected. However the active spins are those which are directly bound to a low natural abundance heteronucleus, with the two most common heteronuclei used being ^{13}C (1.1% abundance) and ^{15}N (0.37% abundance). The passive spins are those bound to far more abundant nucleus (i.e. ^{12}C or ^{14}N). The reduction in sensitivity of the experiment relative to a full-sensitivity experiment is therefore known and constant across samples. In scenarios where strong coupling exists, BIRD can achieve improved sensitivity over ZS, since with the latter a very weak selective pulse would be required to ensure it is of a sufficiently small bandwidth. The BIRD method is particularly attractive in scenarios where the sensitivity penalty due to the involvement of a low-abundance nucleus has already been paid, for example in sequences where an insensitive nuclei enhancement by polarization transfer (INEPT) element is present[125]. One of BIRD's primary drawbacks is the fact that geminal protons (i.e. protons bound to the same heteroatom) cannot be decoupled from each other, since such protons are always in the same subset of either active or inactive nuclei. Doublets rather than singlets will arise in such cases.

4.1.4 PSYCHE

The most recent major development in pure shift spectroscopy is the pure shift yeilded by chirp excitation (PSYCHE) experiment[126, 127]. **Description... Element, How it works (very simple), Effectiveness**

With PSYCHE, the proportion of active and passive spins is dependent on the flip angle of the chirp pulses; for a PSYCHE element featuring chirp pulses with flip angles β , the proportions of are $\cos^2 \beta$ and $\sin^2 \beta$, respectively.

The PSYCHE element has also been employed in conjunction with the 2DJ experiment in order to produce spectra which already feature orthogonal separation of the chemical shifts and couplings along the two frequency axes[128, 129]. Being a 3D experiment, the PSYCHE-2DJ requires long experiment times (typically tens of hours) in order to produce a spectrum with well-resolved multiplet structures in the indirect dimension.

4.1.5 Pure shift spectra from 2DJ estimation

Mandelstahm?

Beyond specialised pulse sequences, procedures based on the estimation of 2DJ datasets have also been developed to achieve broadband homodecoupling. Nuzillard introduced a linear predictive estimation of signal time reversal (ALPESTRE)[130, 131], in which the parameters of each indirect-dimension FID are estimated using LPSVD, such that a set of parameters $\boldsymbol{\Theta} \in \mathbb{R}^{N^{(2)} \times 4M}$ is generated.

$$\boldsymbol{\theta}_{n^{(2)}} = \begin{bmatrix} \boldsymbol{\alpha}_{n^{(2)}}^T & \boldsymbol{\phi}_{n^{(2)}}^T & \boldsymbol{f}_{n^{(2)}}^T & \boldsymbol{\eta}_{n^{(2)}}^T \end{bmatrix}^T. \quad (4.4)$$

The parameters generated are used to propagate each FID backward into $-t^{(1)}$, producing a “full-echo”:

$$\gamma_{n^{(1)}, n^{(2)}}^{\text{full}} = \sum_{m=1}^M \alpha_{n^{(2)}, m} \exp(i\phi_{n^{(2)}, m}) \exp\left(\left(2\pi i f_{n^{(2)}, m} n^{(1)} - \eta_{n^{(2)}, m} |n^{(1)}|\right) \Delta_t^{(1)}\right), \quad (4.5)$$

$$\forall n^{(1)} \in \{-N^{(1)} + 1, \dots, 0, \dots, N^{(1)} - 1\}, \forall n^{(2)} \{0, \dots, N^{(2)} - 1\}.$$

FT of Equation 4.5 generates a spectrum whose real component comprises absorption-mode Lorentzian character in both dimensions. This opens up the means of producing pure-shift spectra from the 2DJ experiment with sharp lineshapes and without signal loss. A similar approach proposed by Mutzenhardt et al. instead constructs full echoes via LP of each direct-dimension FID, and generates a full echo by propagating into $-t^{(2)}$ [132].

4.2 Methodology

CUPID aims to generate pure shift spectra by utilising the result of parametric estimation of 2DJ data, assumed to take the functional form of Equation 4.1. In this section, a description of the method is given.

4.2.1 The estimation routine

The primary steps involved in estimating a 2DJ dataset are

- (i) Generation of a frequency-filtered sub-FID (Section 4.2.3).
- (ii) Prediction of the model order, either by applying the MDL on the first FID in the direct dimension (Section 2.2.3) or by manually specifying a value.
- (iii) Generation of an initial guess using the MMEMPM (Section 2.2.2).
- (iv) Subjection of the initial guess to NLP (Section 2.3).

Instead of estimating successive 1D FIDs, as proposed by Nuzillard and Mutzenhardt et al., 2DJ sub-FIDs are estimated holistically. By doing this, a number of benefits are realised. Firstly, multiplet structures which heavily overlap in a conventional 1D dataset become separated in the 2DJ dataset (assuming that the Larmor frequencies of the relevant spins are sufficiently different). Accurate resolution of the signal components in more crowded spectral regions is far more likely to be successful with a full 2D estimation as a result. On top of this, there is an extra resolution advantage relative to the estimation of *direct* dimension FIDs. Due to the presence of a spin echo during $t^{(1)}$, signal damping effects caused by field inhomogeneities are nullified, such that damping is dictated solely by transverse relaxation (T_2). During $t^{(2)}$ however, the influence of field inhomogeneities are not corrected, such that damping occurs at a faster rate, characterised by T_2^* . As such, multiplet structures in the indirect dimension exhibit better resolution (assuming $f_{sw}^{(1)}/N^{(1)}$ and $f_{sw}^{(2)}/N^{(2)}$ are comparable). A further benefit comes with having access to the frequencies of each oscillator in *both* dimensions, since this allows one to group together those which belong to the same multiplet (see Section 4.2.4). Similar information can be obtained by extracting slices of a tilted magnitude-mode 2DJ spectrum at appropriate values of $F^{(2)}$, though the line-shapes of peaks suffer from the undesirable characteristics described above. The ZS-2DJ[122] and PSYCHE-2DJ[128, 129] experiments are also able to generate individual multiplet structures, though with long three-dimensional (3D) pulse sequences, and with reduced sensitivity relative to a conventional 2DJ experiment.

As was mentioned in Section 2.2.3, application of the MDL to a 2D FID is not desirable, since a full SVD would need to be computed on the Hankel matrix \mathbf{E}_Y . Assuming that the spectral region being considered is not too crowded, applying the MDL on the first direct-dimension FID can return reasonable estimates of M at a far smaller computational cost. For particularly crowded regions, resorting to a manual specification of model order by inspecting the 2DJ spectrum is the best solution currently available. An interesting benefit is realised when the 1D MDL is applied. As described above, the presence of strong coupling artefacts introduces nuisance peaks into spectra produced by shearing and summation. Exactly the same effect would be realised using estimation, assuming that the relevant signals are quantified. However, since strong coupling artefacts have direct-dimension frequencies which are identical to those of first-order signals in the dataset, it is virtually impossible to resolve these using the 1D MDL approach. Therefore the MDL is often found to generate a model order which agrees with the number of *first-order* signals, rather than the *true* number of signals in the FID. As the MMEMPM generates a parameter estimate based on the first M significant components of the dataset, the more intense first-order signals are quantified, whereas the weaker strong coupling artefacts are not included.

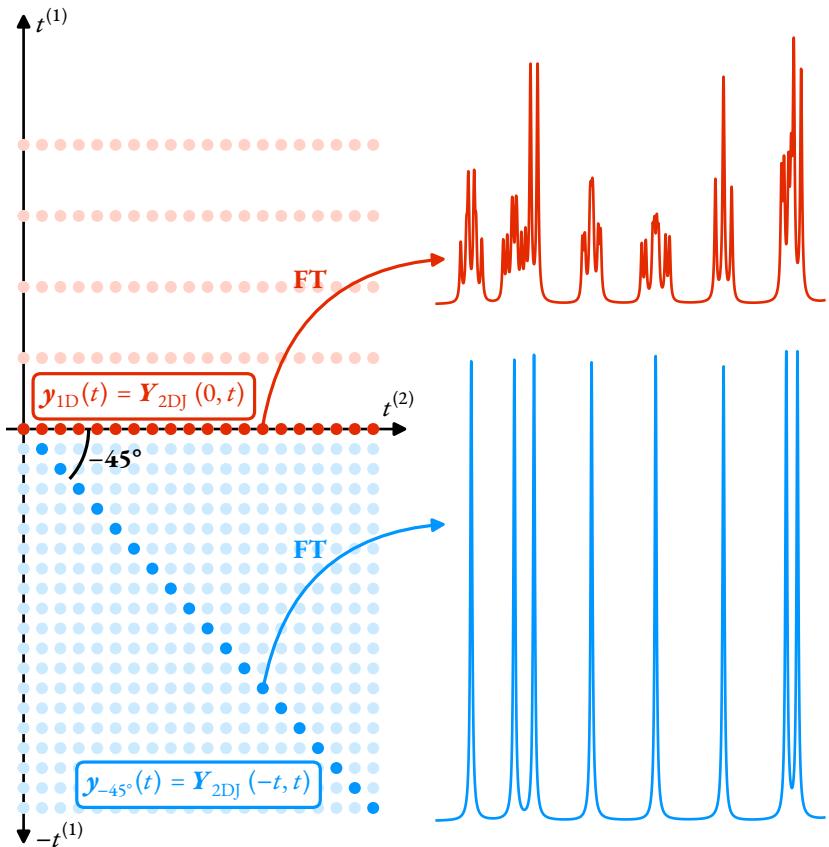


FIGURE 4.2: An illustration of the reasoning behind the name “ -45° signal”, which is used to generate pure shift spectra. The pale red dots denote a typical 2DJ FID, where the amount and rate of sampling in the direct dimension is greater than in the indirect dimension (i.e. $N^{(1)} \ll N^{(2)}$ and $f_{\text{sw}}^{(1)} \ll f_{\text{sw}}^{(2)}$). The bright red dots correspond to the first direct-dimension signal $y(0, t^{(2)})$, which has the same form as an FID from a pulse-acquire experiment. A hypothetical signal generated by propagating the FID into $-t^{(1)}$, with the same rate of sampling in both dimensions, is denoted with pale blue dots. Taking the diagonal of this signal, such that it forms a -45° angle to the $t^{(2)}$ axis, yields an FID y_{-45° which is homodecoupled. Note that there is a slight discrepancy between Equation 4.6 and this description, in that the indirect-dimension damping factors $\gamma^{(1)}$ are neglected in the former case.

4.2.2 The -45° signal

The 2DJ estimation routine generates a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{6M}$. With knowledge of the frequencies and damping factors in both dimensions, it is possible to generate an FID which will produce a pure shift spectrum directly, rather than constructing a full-echo 2DJ signal, and subsequently shearing and summing it. The desired signal is named the -45° signal $\mathbf{y}_{-45^\circ} \in \mathbb{C}^{N^{(2)}}$:

$$\mathbf{y}_{-45^\circ,n^{(2)}} = \sum_{m=1}^M a_m \exp(i\phi_m) \exp\left(\left(2\pi i \left(f_m^{(2)} - f_m^{(1)} - f_{\text{off}}\right) - \eta_m^{(2)}\right) n^{(2)} \Delta_t^{(2)}\right), \quad (4.6)$$

with the reasoning behind the name provided by Figure 4.2. The -45° signal takes the form of a 1D FID expected from a pulse-acquire experiment, except that the frequency of each oscillator, which would usually be $f_m^{(2)}$, is replaced with $f_m^{(2)} - f_m^{(1)}$, such that oscillators belonging to a given multiplet all provide a contribution with the frequency $\omega_{0,s}$. Assuming that the parameters associated with the 2DJ FID are accurately determined, a pure shift spectrum with sharp absorption-mode lineshapes and no loss of signal can be generated by constructing the -45° signal.

4.2.3 Filtration of 2DJ data

Unlike the direct-dimension, which can often comprise sparsely distributed peaks in the Fourier domain, the indirect dimension of 2DJ datasets tends to be densely populated since all multiplet structures are centered at 0 Hz, and rarely span beyond ± 50 Hz. As such, generation of frequency-filtered sub-FIDs is limited to consideration of the direct dimension. The filtering procedure applied to 2DJ data is an extension of that for 1D data described in Section 2.5, and is depicted in Figure 4.3:

- (i) The signal $\mathbf{Y}_{\text{ve}} \in \mathbb{C}^{N^{(1)} \times 2N^{(2)}}$ is constructed, such that a virtual echo is formed from each direct-dimension signal. Each row of the signal $\mathbf{y}_{\text{ve},n^{(1)}} \forall n^{(1)} \in \{0, \dots, N^{(1)} - 1\}$ given by

$$\mathbf{y}_{\text{ve},n^{(1)}} = \begin{bmatrix} \Re(\mathbf{y}_{n^{(1)},0}) & \mathbf{y}_{n^{(1)},1} & \cdots & \mathbf{y}_{n^{(1)},N^{(2)}-1} & 0 & \mathbf{y}_{n^{(1)},N^{(2)}-1}^* & \cdots & \mathbf{y}_{n^{(1)},1}^* \end{bmatrix}. \quad (4.7)$$

- (ii) \mathbf{Y}_{ve} is subjected to FT along the direct dimension to produce the spectrum \mathbf{S}_{ve} (panel a of Figure 4.3). This has an imaginary component of zero.
- (iii) A super-Gaussian $\mathbf{G} \in \mathbb{R}^{N^{(1)} \times 2N^{(2)}}$ is constructed (panel b):

$$\mathbf{G} = \mathbf{1} \otimes \mathbf{g}^{(2)}, \quad (4.8)$$

where $\mathbf{1} \in \mathbb{R}^{N^{(1)}}$ is a vector of ones, and $\mathbf{g}^{(2)} \in \mathbb{R}^{2N^{(2)}}$ is a super-Gaussian vector given by Equation 2.67.

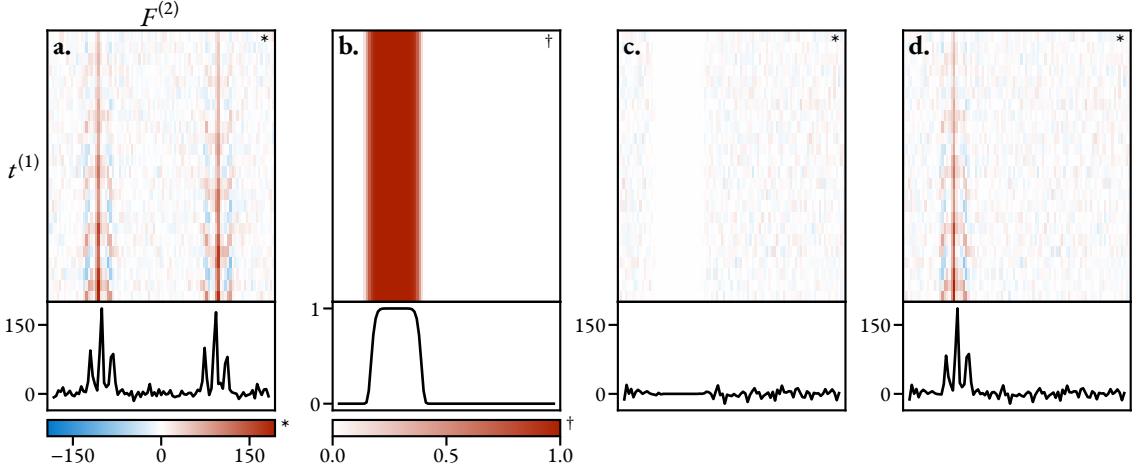


FIGURE 4.3: An illustration of the filtering procedure for 2DJ data. For each panel is a heat-map of the full 2D signal, as well as a plot underneath of the first slice of the signal in the direct dimension. **a.** The spectrum \mathbf{S}_{ve} , **b.** Super-Gaussian filter \mathbf{G} , **c.** Additive noise, attenuated by the super-Gaussian, $\mathbf{W}_{\sigma^2} \odot (\mathbf{1} - \mathbf{G})$, **d.** Filtered spectrum $\tilde{\mathbf{S}}_{\text{ve}}$. Panels **a.–d.** are analogous to panels **b.– e.** in Figure 2.4 for the 1D case.

- (iv) A matrix of additive noise is generated by extracting the variance σ^2 of a direct-dimension strip of \mathbf{S}_{ve} which is devoid of peaks, and generating an array $\mathbf{W}_{\sigma^2} \in \mathbb{R}^{N^{(1)} \times N^{(2)}}$ with values independently sampled from a normal distribution with mean 0 and variance σ^2 .
- (v) The spectrum is filtered (panel d):

$$\tilde{\mathbf{S}}_{\text{ve}} = \mathbf{S}_{\text{ve}} \odot \mathbf{G} + \mathbf{W}_{\sigma^2} \odot (\mathbf{1} - \mathbf{G}). \quad (4.9)$$

- (vi) $\tilde{\mathbf{S}}_{\text{ve}}$ is subjected to IFT and is sliced in half in the direct dimension, yeilding the final filtered signal $\tilde{\mathbf{Y}}$.

4.2.4 Multiplet Prediction

CUPID's ability to group oscillators present in a parameter set into multiplet structures relies knowledge of both thw indirect- and direct-dimension frequencies of each oscillator. As has already been established, for oscillators which are associated with the same multiplet grouping G_s , the quantities $f_{m_1}^{(2)} - f_{m_1}^{(1)}$ and $f_{m_2}^{(2)} - f_{m_2}^{(1)}$ should be equal ($\omega_{0,s}$) for any pairing $m_1, m_2 \in G_s$. An assessment of whether two oscillators belong to the same multiplet can therefore be made using the following criterion:

$$\left| \left(f_{m_1}^{(2)} - f_{m_1}^{(1)} \right) - \left(f_{m_2}^{(2)} - f_{m_2}^{(1)} \right) \right| < \epsilon. \quad (4.10)$$

$\epsilon \in \mathbb{R}_{>0}$ is a suitable threshold to account for error in the estimation result. A lower bound on the threshold is the separation between adjacent points in the better resolved dimension of the spectrum, i.e. $\epsilon = \min \left(f_{\text{sw}}^{(1)} / N^{(1)}, f_{\text{sw}}^{(2)} / N^{(2)} \right)$. However, limitations in resolution due to signal damping

and field inhomogeneities can mean that ϵ has to be increased beyond this for reasonable multiplet assignments to be achieved. Listing Listing C.8 provides a `PYTHON` routine that can be used for multiplet prediction.

There are certain circumstances where it is possible to safely assume that a particular oscillator in the estimation result is not associated with a first-order signal in the dataset:

- (i) The oscillator is not grouped with any other oscillator as part of the multiplet assignment.
- (ii) The magnitude of the indirect dimension frequency of the oscillator is appreciably greater than 0 Hz.

No first-order signal will abide by both of these; if the signal is the only member of a multiplet grouping, it must be a singlet, and as such it will have an indirect-dimension frequency of 0 Hz. Oscillators which do agree with the two points above can be assumed to be related to either strong coupling artefacts or noise, and can be discarded.

4.3 Results

A number of examples of the application of CUPID are now presented. Each result was obtained using the same procedure. For every region of interest in the dataset, a frequency-filtered FID was produced. An initial estimate of model order was determined either using the MDL, or by hard-coding a value determined through manual inspection of the 2DJ spectrum[¶]. An estimation result was then generated using the MMEMPM followed by phase-variance regularised NLP. Finally, a suitable frequency threshold ϵ was determined, and any oscillators in the parameter estimate which abided by the two criteria in Section 4.2.4 were removed.

For details relating to generation of the 2DJ datasets, see Appendices D.1 and D.2.4. Useful metrics, such as run times and the number of oscillators present at different stages of the routine are provided in Table D.7.

4.3.1 “Four Multiplets”

A series of five simulated 2DJ datasets were generated using `SPINACH` such that within a known region of the spectrum (-30 Hz to 30 Hz) four ddd multiplet structures with significant overlap, abiding by the weak coupling approximation, were present. AWGN was added to each FID, with a target SNR of 30 dB. The resulting frequency-filtered FIDs were expected to comprise 32 (4×2^3) signals. As can be seen in Figure 4.4.b, the first direct-dimension FIDs of these datasets are too

[¶]The MDL was used in circumstances where it provided reasonable predictions of model order, as less user input needs to be given to the routine when this is so. However, in certain situations when the MDL performed inadequately — most commonly when the region being considered was very crowded — a manual specification needed to be provided.

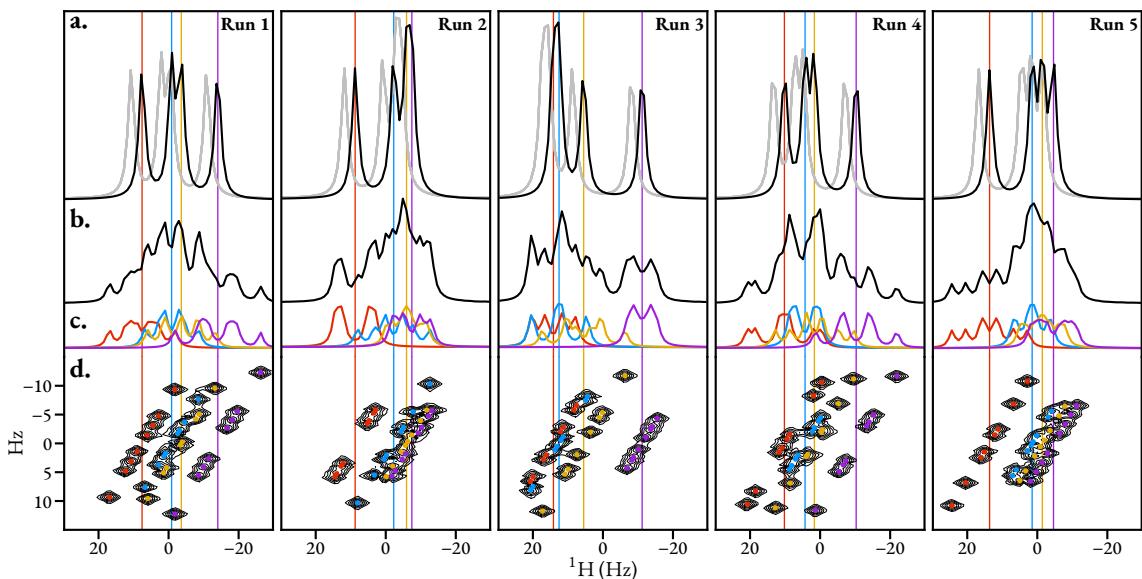


FIGURE 4.4: The result of applying CUPID to 5 instances of simulated 2DJ datasets with 4 heavily overlapping multiplet structures. **a.** Black: pure shift spectrum generated by CUPID (via the -45° signal). Grey: 1D spectrum simulated with Spinach, using the same spin system as was used to produce the 2DJ dataset, but with all scalar couplings set to 0 Hz. This has been offset slightly for clarity. **b.** 1D spectrum of the dataset, produced using the first direct-dimension FID in the 2DJ dataset. **c.** Multiplet structures predicted, using a threshold $\epsilon = f_{\text{sw}}^{(2)}/N^{(2)} \approx 0.98$ Hz. **d.** Contour plot of the magnitude-mode 2DJ spectrum. Coloured points denote the frequencies of oscillators in the estimation result. Coloured vertical lines denote the predicted central frequencies of each multiplet structure.

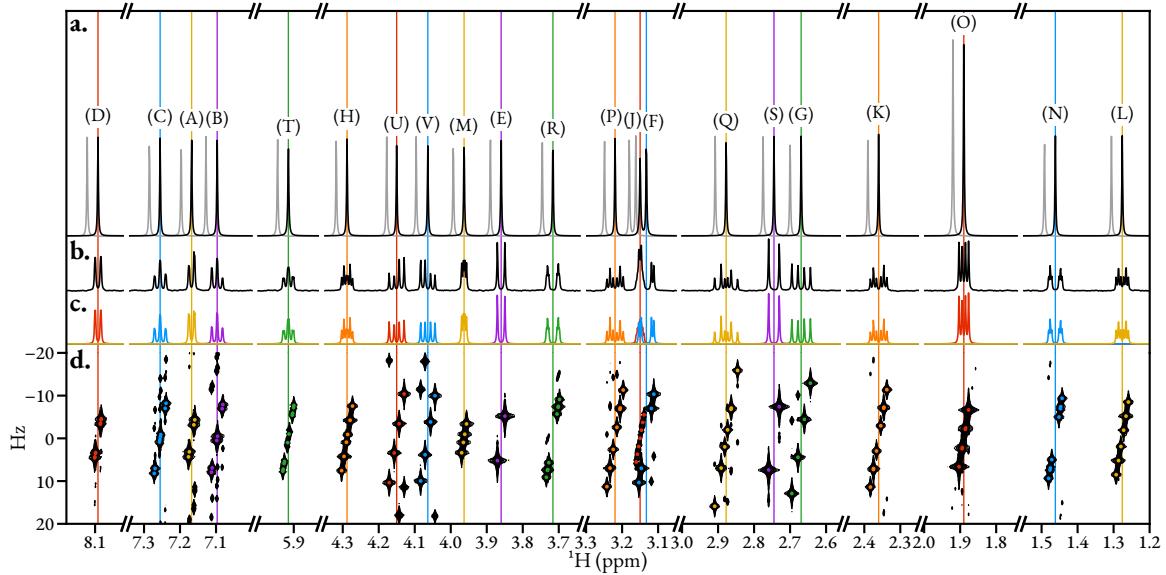


FIGURE 4.5: Application of CUPID on a simulated strychnine 2DJ dataset. **a.** Black: the spectrum generated from FT of the -45° signal. Grey: the spectrum of a simulated dataset with the same chemical shifts, with all scalar couplings set to 0 Hz. **b.** Conventional 1D spectrum. **c.** Multiplet structures assigned ($\epsilon = f_{sw}^{(1)}/N^{(1)} \approx 0.39$ Hz). **d.** Magnitude-mode 2DJ spectrum, with the locations of assigned oscillators given as coloured points.

crowded for reasonable estimates of model order to be made using the MDL, so a value was manually provided. For each run, a random integer from the range [33, 40] was selected as the initial number of oscillators. Hence, the initial guess from the MMEMPM would comprise a slightly excessive number of oscillators. Figure 4.4 illustrates the result achieved for each of the runs. For each FID generated, the method was effective at producing an estimation result with 32 oscillators, as desired, despite the excessive number that were present in the initial guess. Most of the excessive oscillators were purged during the NLP procedure. For 2 of the 5 datasets, the result after NLP comprised 33 oscillators, with a single oscillator being associated with noise. These were automatically detected and removed using the first-order criteria. With simulated examples, it is possible to confirm that the pure shift spectrum generated using CUPID agrees with the desired result; the “true” pure shift spectrum can be obtained by simulating a pulse-acquire experiment, using a spin system with same chemical shifts, but all scalar couplings set to 0 Hz. As seen in Figure 4.4.a. the spectra produced using CUPID agree well with these. The ddd multiplet structures were all assigned successfully, and are plotted individually in panel c of the figure.

4.3.2 Strychnine simulated

As a second example of applying CUPID on simulated data, the chemical shifts and isotropic scalar couplings associated with strychnine were used to construct a 2DJ dataset. AWGN was included with a target SNR of 20 dB. The CUPID procedure was applied to filtered sub-FIDs such that

the signals arising from all spins were considered, with the result presented in Figure 4.5. There are numerous regions in the dataset where strong coupling artefacts reside, and as such this dataset provides a good gauge on the effectiveness of CUPID when these are present.

The MDL was applied to the first direct-dimension FID in order to predict model order. In most circumstances, the model order used resulted in estimation results in which the first-order signals were well quantified, while those corresponding to strong coupling artefacts were neglected. In some circumstances, certain strong couplings signals were quantified, though the relevant oscillators were purged on every occasion based on the first-order criteria. As such, no such artefacts appear in the final pure shift spectrum. The absence of strong coupling artefacts in the estimation result also leads to generated multiplet structures which do not exhibit the typical “roofing” phenomenon associated with strongly coupled spins (*cf.* panels b and c of Figure 4.5). The clearest examples of this are associated with the pair (U) & (V), as well as trio (A), (B) & (C). As with the “four multiplets” example, good agreement is achieved between the pure shift spectrum generated via the -45° signal, and a spectrum generated by running a 1D simulation with the same spin system, expect that scalar couplings are set to 0 Hz.

4.3.3 Quinine

Figure 4.6 illustrates the result of applying CUPID on a dataset generated from a sample comprising quinine (Figure D.1.a) in CD_3OD , with all signals arising from non-aromatic protons considered. The method successfully generated a pure shift spectrum with distinct peaks for each ^1H environment. An example of strong coupling artefacts being neglected can be seen around the 4.9 ppm region, featuring signals from spins (C) & (D). As well as this, the multiplet grouping procedure is able to separate the signals corresponding to spin (D) (purple) and residual water in the sample (grey). The presence of water is a hindrance since it heavily overlaps with (D)’s multiplet structure. To obtain a clean singlet for spin (D) in the pure shift spectrum, the oscillator corresponding to the water signal was simply neglected from the parameter set used to generate the -45° signal. This concept of neglecting nuisance signals through post-processing has similarities with SVD-based approaches for solvent suppression[133]. Solvent suppression approaches of this manner tend to operate by assuming that the most significant component(s) in the data are derived from the solvent, and these are subtracted from the dataset to remove their influence. The removal of the water signal is slightly different here, in that it was removed manually by inspecting the CUPID result. A knowledgeable user would be able to locate the water signal, determine that it is unrelated to, and neglect it. In scenarios where little is known about the sample, or the user does not have a high level of expertise, manually neglecting signals in this manner may not be achievable.

This example provides a few examples where a noticeable under-fitting of multiplet structures

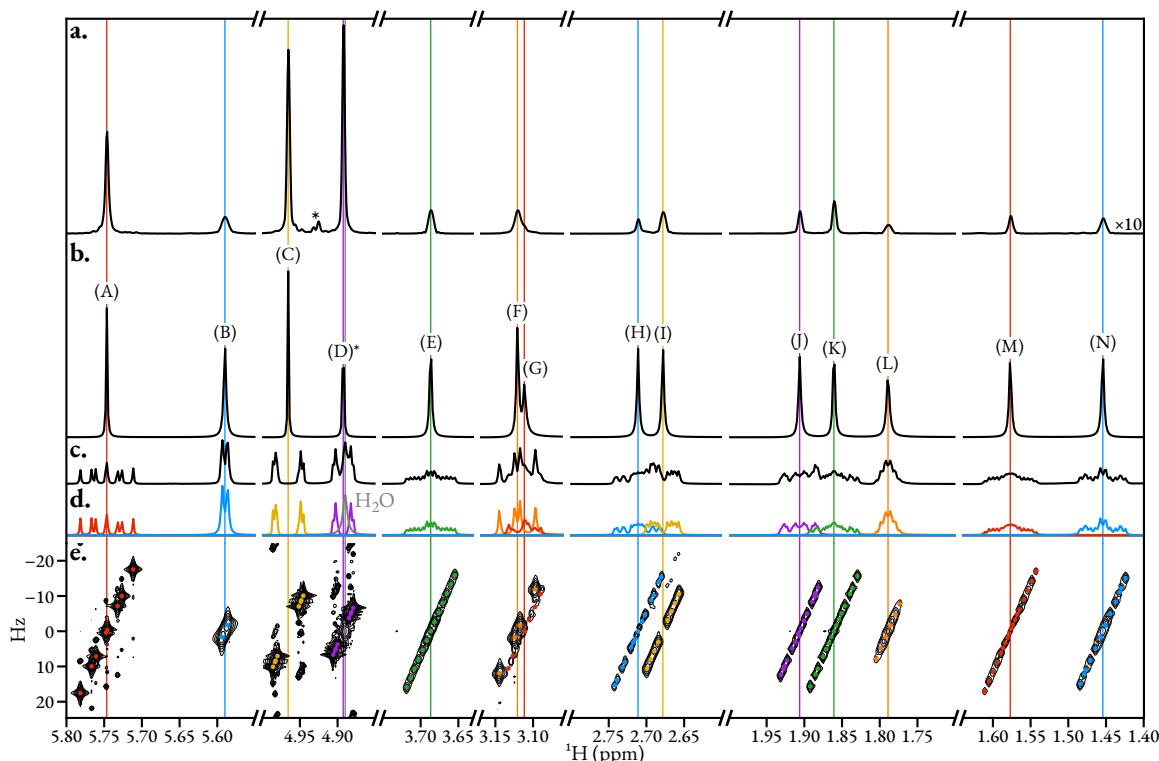


FIGURE 4.6: Application of CUPID on the non-aromatic regions of a quinine 2DJ dataset. **a.** Spectrum produced using the 45° shear and summation methodology. The peaks denoted by and asterisk arise from strong coupling artefacts. **b.** The spectrum generated from FT of the -45° signal, with the signal arising from H_2O (grey, close to 4.9 ppm neglected). **c.** Spectrum of the first direct-dimension signal in the 2DJ FID. **d.** Multiplet structures assigned ($\epsilon = f_{\text{sw}}^{(2)}/N^{(2)} \approx 0.92$ Hz). **e.** Contour plot of the absolute value mode 2DJ spectrum, with the locations of assigned oscillators given as coloured points.

has occurred. The most notable case comes from the spin (G) multiplet, where close proximity with spin (F)'s multiplet has likely compounded the task of accurately estimating the associated signals. With fewer oscillators than the true number of signals at its disposal, the NLP routine will compensate by giving said oscillators large amplitudes and damping factors, so that they can reasonably fit multiple similar-frequency signals. This phenomenon culminates in the pure shift peak possessing an augmented linewidth. This behaviour is also exhibited to a lesser extent by the multiplet for spin (B), which comprises two pairs of very close signals in a dd structure. A single oscillator is fit to each pair of signals, culminating in a broadened pure shift peak.

For comparison, panel a of Figure 4.6 presents a pure-shift spectrum produced via application of a 45° shear, followed by summation along the indirect dimension. Due to the application of sine-bell apodisation, the relative amplitudes of the pure-shift peaks are drastically different. In particular, the alkenyl signals from spins (A), (C) & (D), are less perturbed by the apodisation relative to the aliphatic signals due to longer T_2 times^{||}. As well as this, signals arising from strong coupling between (C) and (D) are visible (these are denoted with an asterisk).

4.3.4 Camphor

The application of CUPID to the non-methyl signals of a 2DJ dataset of camphor (Figure D.1.c) in DMSO-d₆ is presented in Figure 4.7. As with the quinine example, a spectrum generated through the shear and summation procedure is presented for comparison. In most regions of the dataset, the estimation technique successfully parametrised the first-order signals, while neglecting strong coupling artefacts. However, it was not possible to solely estimate the first-order signals associated with the pair of spins (F) and (G). The extent of strong coupling between the nuclei is such that some of the strong coupling artefacts have comparable amplitudes to the first-order signals. Reliance on the MMEMPM —which determines the most significant components in the data— therefore makes it challenging to solely estimate the first-order signals in this case. The strong coupling signals which were estimated by the routine are denoted in grey in the 1.36 ppm to 1.24 ppm region of the spectrum. Their contributions to the spectrum of the -45° signal are also plotted in grey. As with the shear and summation spectrum, three extra peaks reside between the pure shift peaks associated with (F) and (G), which ideally would not exist. In much the same way that the parameters associated with water in the quinine example were neglected in constructing the -45° signal, those associated with strong coupling artefacts can be too. In neglecting said parameters, the black spectrum in panel b results. Achieving this requires manual intervention, such that the user needs the expertise to distinguish between first-order signals and strong coupling artefacts.

^{||}Spins with longer T_2 s produce signals which decay less rapidly. Sine-bell apodisation diminishes the amplitudes of the initial points in the FID. Therefore, if the signal decays less rapidly, the relative extent by which the *power* of the signal is diminished is less compared with a signal derived from a spin with a small T_2 .

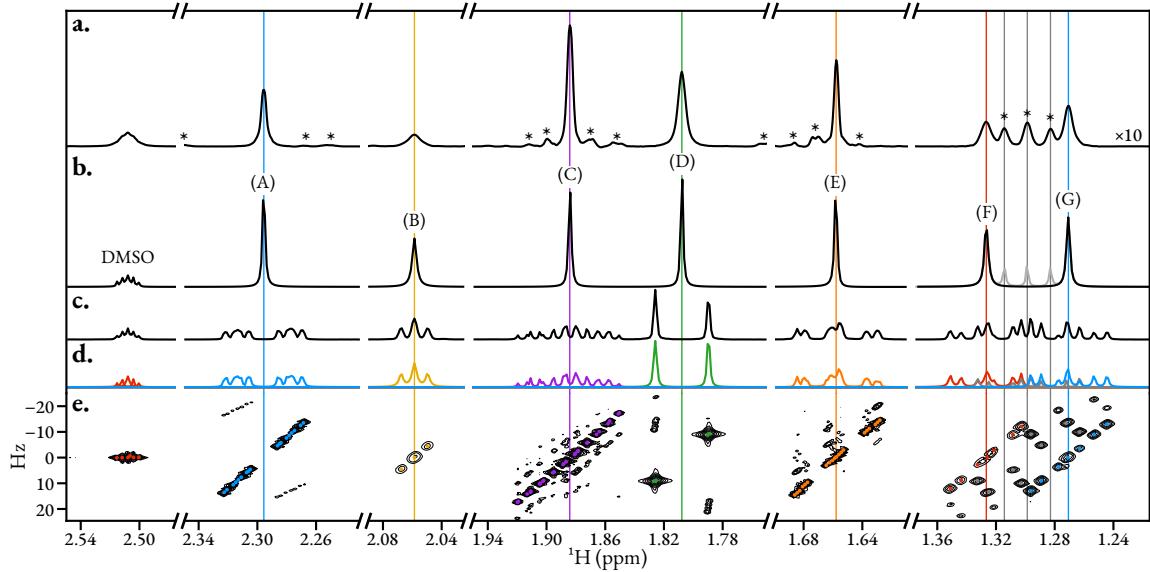


FIGURE 4.7: Application of CUPID on camphor 2DJ dataset. **a.** Spectrum produced using the 45° shear and summation methodology. The peaks denoted by and asterisk arise from strong coupling artefacts. **b.** Black: the spectrum generated from FT of the -45° signal. Oscillators associated with strong coupling artefacts between spins (F) and (G) were neglected. Grey: spectrum generated without neglecting oscillators associated with strong coupling artefacts. **c.** Spectrum of the first direct-dimension FID. **d.** Multiplet structures assigned ($\epsilon = 2f_{sw}^{(2)}/N^{(2)} \approx 1.23$ Hz). **e.** Magnitude-mode 2DJ spectrum, with the locations of assigned oscillators given as coloured points.

4.3.5 Dexamethasone

Figure 4.8 shows the result of applying CUPID on a dataset acquired from a sample dexamethasone in DMSO-d₆. A pure shift spectrum was also acquired using the triple spin echo PSYCHE (TSE-PSYCHE) experiment[127, 128] for comparison (see Figure D.5). Overall, excellent agreement is achieved between the PSYCHE spectrum, and that generated using the -45° signal. The estimation routine performed amicably in cases where heteronuclear coupling to ^{19}F were present. For spins (H) and (N), two distinct multiplet structures were assigned, which are coloured blue and yellow (H) and yellow and purple (N) in the figure. The perceptible though very small heteronuclear coupling between ^{19}F and (D) could not be resolved by the estimation routine however.

In this example, there are a few cases where oscillators which result from parametrising strong coupling artefacts exist. These persist because either they have an indirect frequency ≈ 0 Hz, or because at least two parameterised artefacts are grouped together as part of the multiplet assignment. As with the camphor example, a knowledgeable user could identify and neglect the culprit oscillators if desired, though they are not purged in producing the -45° signal in this case.

Ask Ali: is PSYCHE quantitative? Quite major discrepancies in peak integrals

4 Pure shift spectra from 2D J-resolved estimation

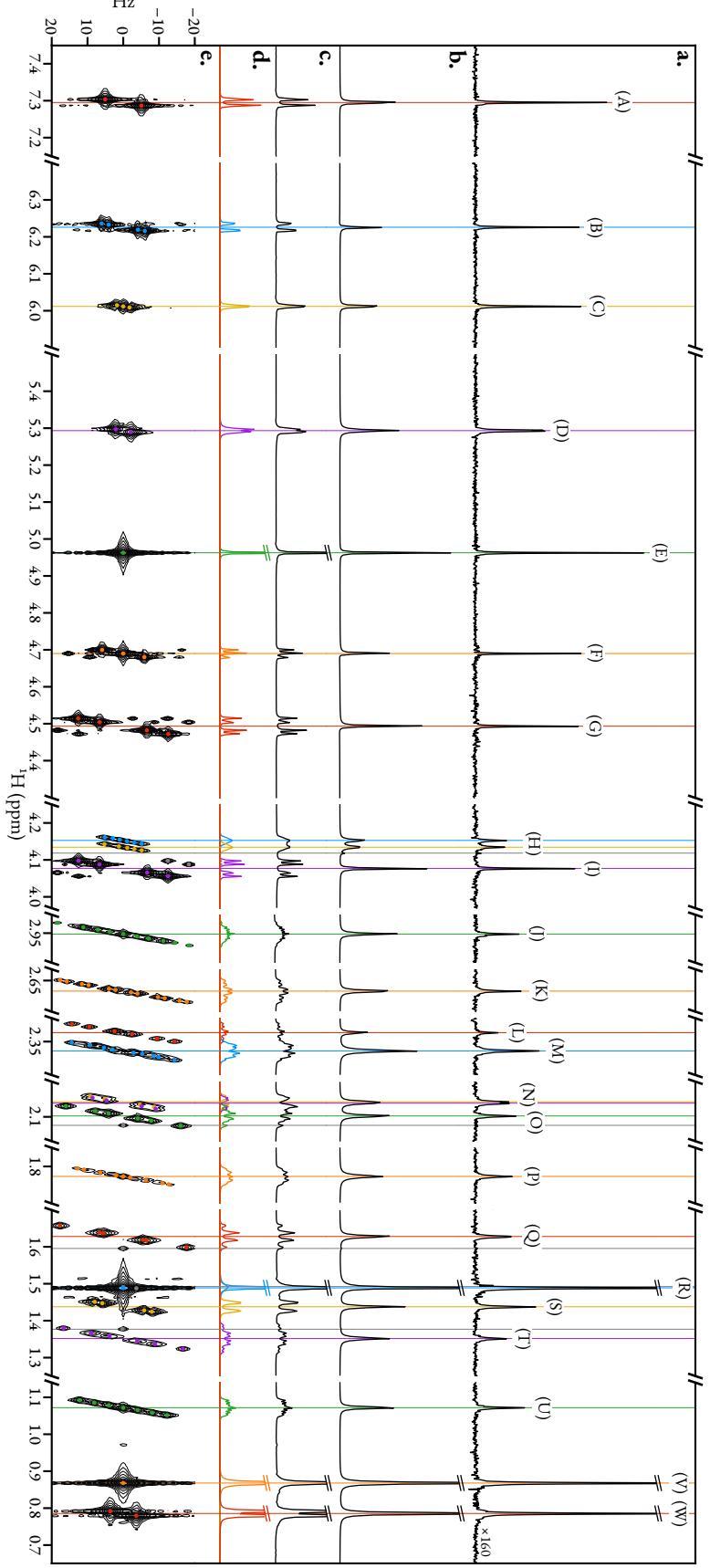


FIGURE 4.8: Application of CUPID on a 2DJ dataset of dexamethasone in DMSO-d₆. **a.** TSE-PSYCHE spectrum of the sample. **b.** The spectrum generated from FT of the -45° signal. **c.** Conventional 1D spectrum. **d.** Multiplet structures assigned ($\epsilon = f_{sw}^{(2)}/N^{(2)} \approx 0.92$ Hz). **e.** Magnitude-mode 2DJ spectrum, with the locations of assigned oscillators given as coloured points.

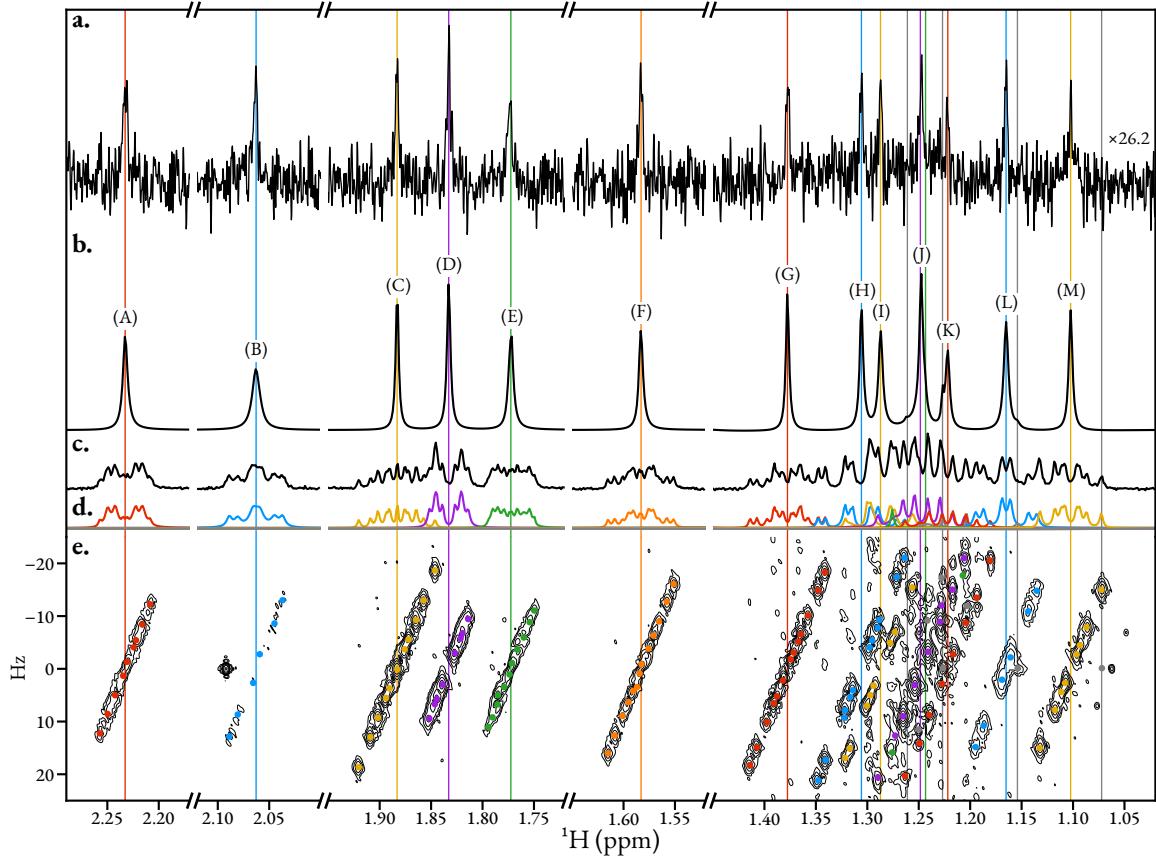


FIGURE 4.9: Application of CUPID on 2DJ dataset of 17β -estradiol in DMSO-d_6 . **a.** PSYCHE spectrum of the sample (see Figure D.4 for details on the pulse sequence). The spectrum has been scaled such that its maximum is of the same magnitude as the spectrum in panel b. 2DJ dataset. **b.** Pure shift spectrum generated using CUPID. **c.** Spectrum of the first direct-dimension FID in the **d.** Multiplet structures assigned ($\epsilon = 2$ Hz). **e.** Magnitude-mode 2DJ spectrum, with coloured points denoting the frequencies of oscillators assigned using estimation. N.B. The 2DJ spectrum was produced using a more concentrated sample of estradiol, since the original 2DJ dataset was too insensitive to produce a useable spectrum after sine-bell apodisation.

4.3.6 Estradiol

A final showcase of CUPID is provided by Figure 4.9, where a low concentration (2 mM) sample of 17β -estradiol (Figure D.1.d) in DMSO-d_6 is considered. This presents the most challenging example of using CUPID due to (a) the low SNR and (b) the presence of incredibly complex regions for a small molecule spectrum, featuring many overlapping multiplet structures. Because of the complexity of the dataset, estradiol was used as a showcase of the PSYCHE experiment in the original work describing it[126]. With the sample and concentration and experimental setup used, the PSYCHE experiment produced a spectrum with very poor SNR, such that it is barely sensitive enough for pure shift peaks to be distinguish from noise (panel a). However due to the innately higher sensitivity associated with the 2DJ experiment (assuming similar experiment

parameters) it was still possible to generate a decent parameter estimate of the 2DJ dataset. Due to the severe signal overlap in the first direct-dimension FID, initial values for the model order had to be manually specified. The time elapsed to run the estimation routine was rather long, at 10.25 min for all regions considered (see Table D.7 for a more detailed description of timings). This is largely attributable to the high model orders, particularly in the most downfield region (this region alone took over 5.5 min to estimate).

4.4 Summary

In this chapter CUPID, a procedure for the parametrisation of 2DJ datasets using a holistic 2D estimation routine, is outlined. With access to the parameters associated with a 2DJ dataset, pure shift spectra can be generated with myriad beneficial features relative to alternative methods.

The original method for pure shift spectrum generation consisted of shearing a magnitude-mode 2DJ spectrum by 45° , and computing the projection onto the $F^{(2)}$ axis. To overcome the grotesque lineshapes which arise due to the presence of dispersion character, and non-linearities in the magnitude-mode spectrum, severe data treatment such as the used of sine-bell apodisation are applied. The resulting pure shift spectra suffer from reduced intensities because of this. On top of this, the intensities of peaks are attenuated by different extents, such that relative peak integrals become meaningless. The presence of strong coupling in the spin system also introduces unwanted artefacts into the spectra.

Experimental procedures based on “chunking” the initial sections of the FIDs in a 2D experiment — including ZS, BIRD and PSYCHE — have largely superseded the shear and summation approach. One key disadvantage of all of these is that only a fraction of the available spin magnetisation contributes to the final pure shift spectrum, leading to poorer sensitivity.

Through a number of examples, it has been shown that by employing parametric estimation, a simple 2DJ experiment can be harnessed to generate pure shift spectra with sharp, absorption Lorentzian peaks which retain the same signal intensity as the 2DJ experiment. It has been shown to perform admirably even when state of the art techniques like PSYCHE produce spectra with such low SNRs that they are useless. Frequently, the specific procedure employed here is able to automatically discard oscillators present in the model which either correspond to strong coupling artefacts or noise, leading to simplified spectra in which appear to adhere to the weak coupling approximation. There are cases where strong coupling signals do end up in the estimation result. It has been shown that these can be manually neglected from the parameter set so they don’t have an unwanted influence on final pure shift spectrum, though this requires manual intervention from a knowledgeable user.

Simultaneously, CUPID can assign multiplet structures, by grouping oscillators which lie along a specific 45° cross section in frequency space. Achieving this experimentally — effectively

involving a 2DJ experiment in conjunction with a pure shift element — requires running an extremely long (hours or even days) 3D pulse sequence. The usefulness of the multiplet structures generated by CUPID is dependent on how accurately the estimation routine was able to parameterise the dataset. In numerous occasions in the examples presented here, the estimation routine was unable to resolve certain, similar frequency signals. A complete understanding of the coupling network associated with a given spin in not attainable when this is so. However, such multiplet structures are still able to provide large amounts of information about the spin system.

CUPID is limited by the complexity of the dataset of interest. The reasons for this are two-fold. First, with datasets comprising progressively more peaks in a given spectral region, the difficulty in estimating the dataset becomes harder. Second, with an increased model order required to estimate the dataset, the time required for computation increases rather drastically. This feature is most clearly observed when comparing the times required in estimating the different regions considered in the estradiol example. As a rule of thumb, it's anticipated that CUPID will perform admirably on datasets derived from small molecules, though I am sceptical about its prospects as an effective tool to generate pure shift spectra from large molecules such as proteins.

The estimation routine and applications that have been presented in the previous three chapters are accessible via the *NMR estimation in Python* (NMR-EsPy) package. NMR-EsPy aims to provide a feature-rich yet simple interface in order perform estimation on datasets of interest. In this chapter, a short overview of the package, as well as its accompanying graphical user interface (GUI), is given.

5.1 A description of NMR-EsPy

5.1.1 Why PYTHON?

There a number of reasons why PYTHON was the chosen programming language for NMR-EsPy:

- It has a large user-base, particularly within the scientific community.
- The SCIPY ecosystem[134], including the packages NUMPY[135] and MATPLOTLIB[136] is a powerful tool which enables high-performance scientific computation and visualisation in PYTHON, despite the language's reputation for slow speed.
- Being a scripting language with user-friendly syntax makes PYTHON ideal for exploring datasets in a step-by-step fashion. This is useful in the context of NMR-EsPy, as a user will want to (i) inspect and pre-process the data of interest, (ii) determine the regions they wish to estimate and set up the routine, (iii) output and inspect the result. This can be achieved easily by “hacking” and re-running PYTHON scripts or by using “notebook” environments, such as JUPYTER.
- It is free and open-source, as opposed to well-known scientific computing platforms such as MATLAB and MATHEMATICA.
- PYTHON supports sophisticated object-oriented programming features, including multiple

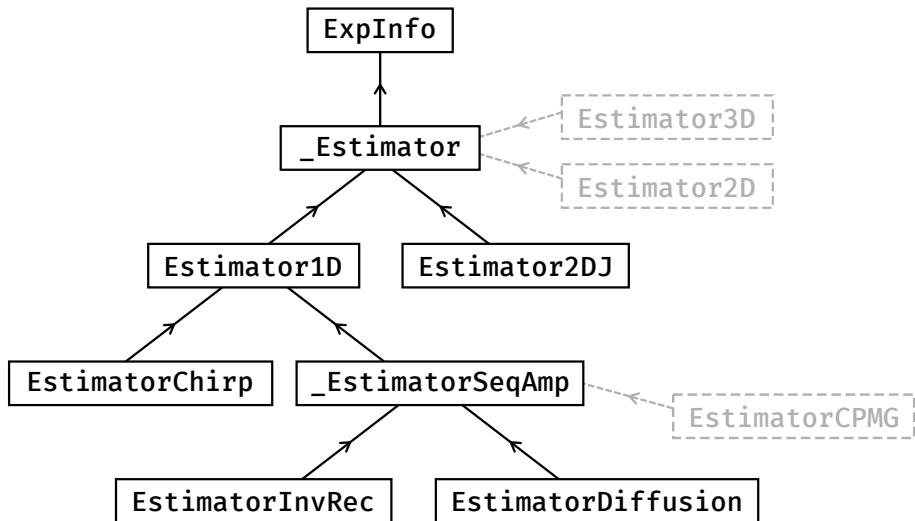


FIGURE 5.1: Inheritance tree for estimation classes in the NMR-EsPy package. Arrows are directed from child classes to the parent class they directly inherit from. Classes in grey are objects that could be added to the application programming interface (API) in the future.

levels of inheritance. This is exploited in NMR-EsPy in order to create numerous objects designed with specific NMR data types in mind.

Probably the largest disadvantage in using PYTHON is its slow performance on account of it being an interpreted, dynamically typed language which relies on garbage collection for memory management. While NUMPY provides interfaces to run fast computations with pre-compiled C-code, a significant performance benefit would likely be realised if a low-level compiled language like C, C++, or RUST were used instead (this is discussed more in Section 6.2). While this may be so, the development time in writing programs with these lower-level languages is typically a lot greater than with a language with a higher level of abstraction like PYTHON.

5.1.2 Estimator objects

The fundamental user-facing objects (or classes) that NMR-EsPy provides are *estimators*. Instances of these estimator objects contain relevant *attributes* (the FID, experiment parameters including the sweep width, transmitter offset etc), as well as *methods* which perform useful routines like estimation and result figure generation. Thanks to PYTHON’s support for multiple levels of inheritance, it is possible to build numerous objects with certain shared attributes and methods, but which also possess bespoke features that are solely of relevance to the type of NMR data being considered. As an example, only the `Estimator2DJ` object possesses a method called `cupid_spectrum`, which returns the FT of the -45° signal. Figure 5.1 provides an *inheritance diagram* for the different estimator classes in NMR-EsPy. Basic overviews of each class are as follows:

- **ExpInfo:** Stores parameters used in a particular NMR experiment. `ExpInfo` also has methods

for the generation of the timepoints/chemical shifts sampled based on these parameters, and for producing synthetic FIDs.

- **_Estimator:** Also possesses the NMR data as an attribute on top of the experiment information. This class is designed to contain the functionality which can be generalised across all NMR data types supported by NMR-EsPy. It does not possess all the features necessary to useful as a standalone object, and as such the user is not permitted to use it directly (such an object is often referred to as an *abstract class*). For child classes of **_Estimator**, the main feature that requires methods to be defined is the means by which the data is imported (experimental data) or generated (simulated data).
- **Estimator1D:** Conventional 1D datasets are analysed using this class (Section 3.1).
- **_EstimatorSeqAmp:** An abstract class which enables the estimation of sequential 1D datasets (Section 3.2). Analysis of the dataset varies depending on the type of experiment considered, such as the function used for fitting amplitudes (Table 3.2), and the means by which that data should be imported/generated. As such, this is not designed for direct use; one of the classes which inherit it should be used instead. **_EstimatorSeqAmp** stores a number of **Estimator1D** objects, with each object containing one of the FIDs in the sequential dataset. Estimation simply comprises looping through these estimators, giving the parameter estimate of the previous estimator to next as its initial guess.
- **EstimatorInvRec:** For the estimation of inversion recovery (T_1) datasets.
- **EstimatorDiffusion:** For the estimation of diffusion datasets.
- **EstimatorChirp:** For the estimation of FIDs acquired using single-chirp excitation (Section 3.3).
- **Estimator2DJ:** For the estimation of 2DJ data. This class also provides features to generate pure shift spectra and assign multiplet structures using CUPID (Chapter 4).

Other estimator objects which are yet to be implemented, but are potential future additions to the package are also depicted in the inheritance diagram. **EstimatorCPMG** would be for the consideration of CPMG datasets, which enable the determination of T_2 times. **Estimator2D** and **Estimator3D** would be for the estimation of datasets comprising sets of 2D or 3D amplitude- or phase-modulated FIDs, respectively. A discussion of the feasibility of implementing these is given in Section 6.2.

5.2 The NMR-EsPy GUI

Along with the NMR-EsPy API, an accompanying GUI based on the TKINTER toolkit ships with the package. This can be accessed either via the command line, or within BRUKER's Top-

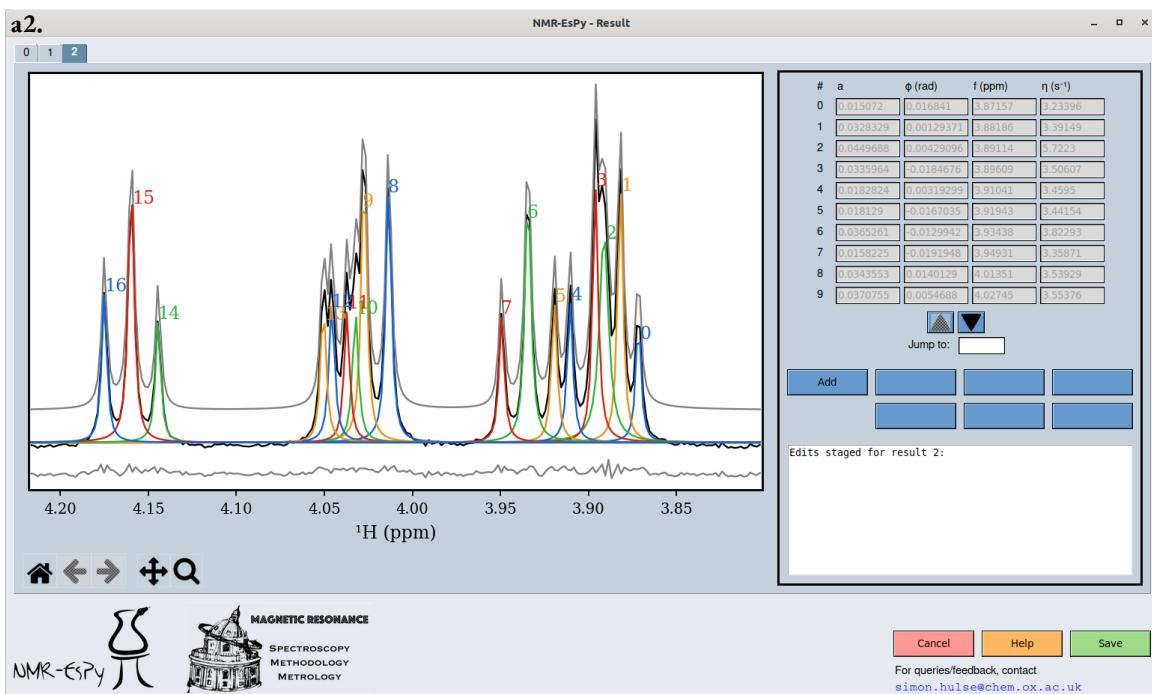
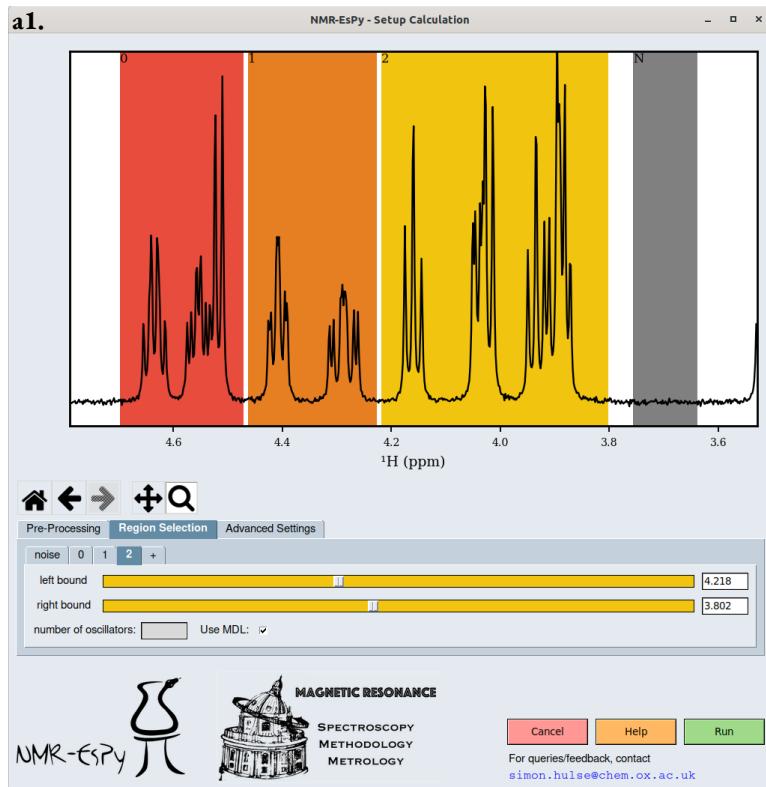
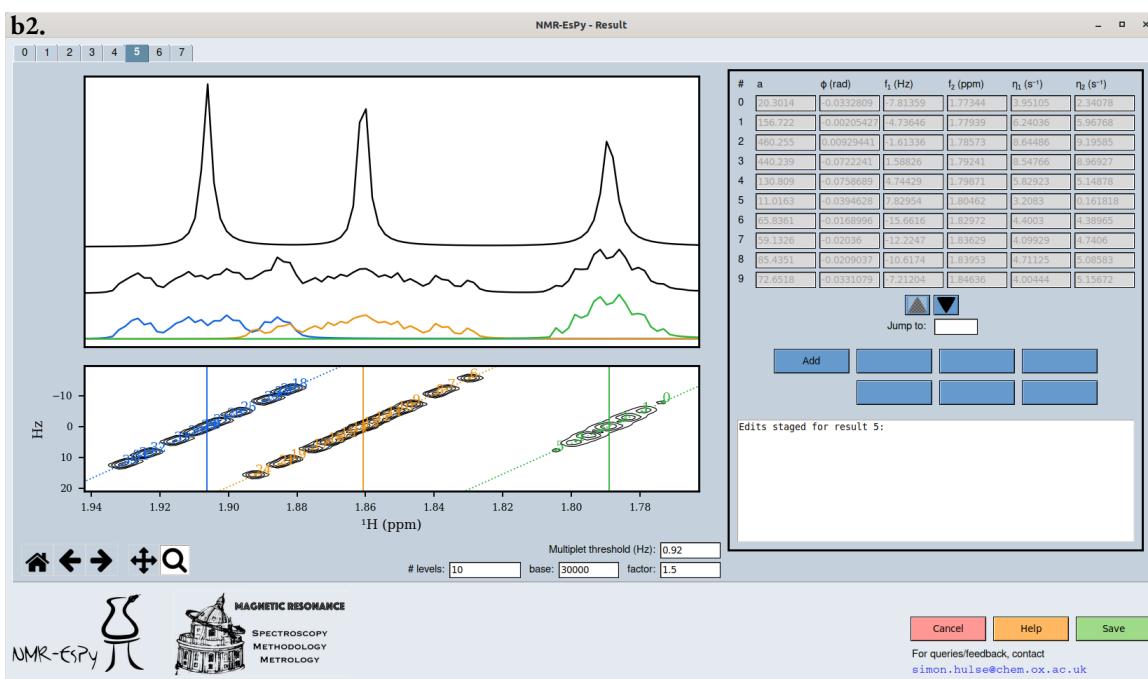
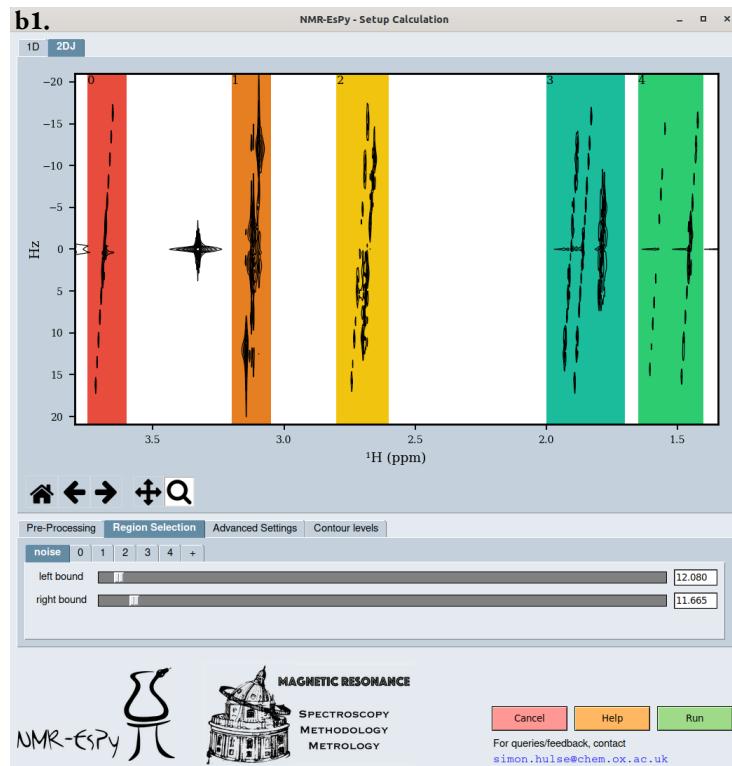


FIGURE 5.2: Screenshots of windows that form part of the NMR-EsPy GUI for 1D estimation (a.) and 2DJ estimation (b.). For both data types, the windows used to setup the estimation routine (1.) and inspect the result (2.) are shown. (Continues on the next page)

Continuation of **FIGURE 5.2.**

SPIN software to provide a more seamless workflow in analysing NMR data acquired on BRUKER spectrometers. At the time of writing, the GUI only supports conventional 1D and 2DJ datasets. Screenshots of the GUIs for both data types are provided in Figure 5.2. The GUI comprises two primary windows; the first enables the estimation routine to be set up (panels a1 & b1), while the second is for inspecting the result (panels a2 & b2). For the set-up window, the following actions are carried out by the user:

- The *Pre-Processing* tab enables phase correction, application of exponential line-broadening*, and baseline correction. If the imported data has already been processed by some other software such as TOPSPIN[†], this step can be skipped.
- Regions of interest and a region to determine the noise variance are specified with the *Region Selection* tab. Panels a1 and b1 in Figure 5.2 provide illustrations of the appearance of the GUI after a number of regions — each denoted by a coloured rectangle — have been defined by the user.
- *Additional Setting* allows features related to the estimation routine to be customised, including whether to approximate the Hessian matrix or compute its exact form; whether to predict the model order using the MDL or to manually specify a value; and setting a threshold for the maximum number of iterations allowed.

The result window effectively features a figure depicting the outcome of the estimation routine, with a table of all the estimated parameters. For any features in the result which the user deems to be erroneous, there is scope for some basic edits to be made to be. Oscillators can be *added* to the result if a particular signal has been missed by the routine; they can be *removed* if they are deemed spurious; multiple oscillators can be *merged* if a particular signal is deemed to be over-fit; and a single oscillator can be *split* if more than one signal is deemed to be under-fit. If any edits are requested by the user, the updated parameter set is subjected to NLP in order to reduce the bias in the result, and ensure good agreement between the model and data is maintained. The ability to edit the result can lead to better outcomes, since the user is able to guide the optimiser using their expertise. The final stage in using the GUI involves specifying the formats to output the result to. Both figures and parameter tables summarising the result can be exported to various formats. As well as this, the estimator object which conducted everything under the hood can be saved to a byte stream for future inspection.

*Exponential line-broadening is the only type of apodisation which is supported in NMR-EsPy; use of any other window function would render the model used to fit the data incompatible. Line-broadening should only be applied in situations where the FID is truncated, such that sinc wiggles are visible in the spectrum, as these will have an unwanted influence on filtered sub-FIDs generated.

[†]For 1D datasets, it is possible to import raw FID data (`fid`) or processed spectral data (`1r`). If pre-processed data is imported, NMR-EsPy performs IFT and truncates the conjugate-symmetric signal generated in half to recover the FID. For 2DJ data, it is necessary to import 2DJ data as a raw FID (`ser`).

5.3 Summary

In this chapter, an overview of the software associated with this project, a PYTHON package call NMR-EsPy, has been provided. There are more resources available for additional information on NMR-EsPy. A rigorous description of the usage of NMR-EsPy, including details on installation, and a reference for the API are given in the documentation for NMR-EsPy, found at <https://foroozandehgroup.github.io/NMR-EsPy/>. One chapter of the documentation, which provides walkthroughs to help users gain familiarity with the API, is appended to this thesis, and can found in Chapter E of the appendix. NMR-EsPy is open source (under the MIT license), with the source code hosted on the Foroozandeh group's GITHUB page at <https://github.com/foroozandehgroup/NMR-EsPy>.

CONCLUSIONS AND FUTURE WORK

6

6.1 Conclusions

6.2 Future Work

Any future work related to this project would be centered around extending and improving the NMR-EsPy package. Improvements in the speed of the estimation routine would be realised if the most intensive parts of it were written in a low-level compiled language like C++ or RUST. This may not be the case with the MPM and MMEMPM. The majority of time running these routines involves SVD, with NUMPY’s implementation calling well-optimised routines from the LAPACK and ARPACK software package. However, in its current state, the NLP routine could well be sped up considerably if current PYTHON implementation were ported to one of the aforementioned languages.

Thinking further afield from the specific estimation routine considered in this work, the NMR-EsPy package provides a large number of useful features which would be applicable to the estimation using any conceived method. This includes functionality to import data, pre-process data, and inspect the result of an estimation routine through figures*. Incorporating other estimation routines into the software would be straightforward.

Another potential future venture would be to extend the routine for the consideration of other 2D datasets, such as those that comprise amplitude- or phase-modulated pairings, as well as 3D datasets, including many of the “triple-resonance” experiments popular in the biological NMR community[22: Section 7.4]. As shown in Chapter 4, the method is capable of performing well on 2D datasets. In order to achieve 3D estimation, it would be necessary to make use of a 3D equivalent of the MMEMPM. Such a technique exists, and is used principally for applications in

*All figures in this thesis were generated using NMR-EsPy routines, with the odd adjustment for aesthetic purposes.

which the direction (i.e. azimuth and elevation angles) as well as the frequency of waves arriving at an antenna are sought[137]. However, there is a key issue with multidimensional NMR signals which would likely hamper the proposed method in this thesis from being effective. 2DJ datasets are rather anomalous in that they have very densely populated indirect dimensions. For this reason, it was necessary to be concerned about filtering the data in the indirect dimension. However, for most 2D datasets, it would be desirable to filter the data in both dimensions. While the filtering procedure presented works well on direct-dimension FIDs, which have usually decayed to noise at the end of acquisition, indirect dimension FIDs are often truncated. The FT of such signals without any treatment would produce spectra with considerable truncation artefacts. Filtering such a spectrum would incorporate undesirable artefacts into the final filtered FID, rendering it worthless. Compensating for the truncation could be achieved, by applying exponential apodisation prior to FT. However, this would then lead to spectra with broader lineshapes, such that wider spectral regions need to be selected to ensure practically all of the peaks of interest lie within it. Dealing with truncated signals could possibly be handled in one of two ways, beyond running the experiment to acquire the data with a huge number of increments. One possibility could be to propagate the FID further in time using LP. Another option might be to apply conventional apodisation, such as a sine-bell function to the FID, producing a FID without truncation artefact, and with acceptable resolution. Applying a non-exponential weighting to the dataset would however render the data incompatible with the estimation routine in its current form, such that a modified routine with a suitable model would need to be implemented. For 3D estimation, the computational burden required would also likely be too high for the routine to be practicable.

Once 2D profiling has been done, perhaps this could be used to guestimate typical 3D running times?

BIBLIOGRAPHY

- [1] F. Bloch, W. W. Hansen, and M. Packard. “Nuclear Induction”. In: *Phys. Rev.* 69 (3-4 Feb. 1946), pp. 127–127.
- [2] E. M. Purcell, H. C. Torrey, and R. V. Pound. “Resonance absorption by nuclear magnetic moments in a solid”. In: *Physical review* 69.1-2 (1946), p. 37.
- [3] E. D. Becker. “A brief history of Nuclear Magnetic Resonance”. In: *Analytical Chemistry* 65.6 (1993), 295A–302A.
- [4] G. R. Eaton, S. S. Eaton, and K. M. Salikhov. *Foundations of modern EPR*. Singapore ; London: World Scientific, 1998.
- [5] W. D. Knight. “Nuclear Magnetic Resonance Shift in Metals”. In: *Phys. Rev.* 76 (8 Oct. 1949), pp. 1259–1260.
- [6] W. G. Proctor and F. C. Yu. “The Dependence of a Nuclear Magnetic Resonance Frequency upon Chemical Compound”. In: *Phys. Rev.* 77 (5 Mar. 1950), pp. 717–717.
- [7] W. C. Dickinson. “Dependence of the ^{19}F Nuclear Resonance Position on Chemical Compound”. In: *Phys. Rev.* 77 (5 Mar. 1950), pp. 736–737.
- [8] R. R. Ernst and W. A. Anderson. “Application of Fourier Transform Spectroscopy to Magnetic Resonance”. In: *Review of Scientific Instruments* 37.1 (1966), pp. 93–102.
- [9] R. Freeman and G. A. Morris. “The Varian story”. In: *Journal of Magnetic Resonance* 250 (2015), pp. 80–84.
- [10] J. W. Cooley and J. W. Tukey. “An Algorithm for the Machine Calculation of Complex Fourier Series”. In: *Mathematics of Computation* 19.90 (1965), pp. 297–301.
- [11] J. Jeener. “Ampere international summer school”. In: *Basko Polje, Yugoslavia* 197.1 (1971).
- [12] J. Jeener and A. G. ““Pulse pair technique in high resolution NMR” a reprint of the historical 1971 lecture notes on two-dimensional spectroscopy”. In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 94-95 (2016), pp. 75–80.
- [13] W. P. Aue, E. Bartholdi, and R. R. Ernst. “Two-dimensional spectroscopy. Application to nuclear magnetic resonance”. In: *The Journal of Chemical Physics* 64.5 (1976), pp. 2229–2246.

- [14] M. P. Williamson, T. F. Havel, and K. Wüthrich. "Solution conformation of proteinase inhibitor IIA from bull seminal plasma by ^1H nuclear magnetic resonance and distance geometry". In: *Journal of Molecular Biology* 182.2 (1985), pp. 295–315.
- [15] D. Marion, P. C. Driscoll, L. E. Kay, P. T. Wingfield, A. Bax, A. M. Gronenborn, and G. M. Clore. "Overcoming the overlap problem in the assignment of proton NMR spectra of larger proteins by use of three-dimensional heteronuclear proton-nitrogen-15 Hartmann-Hahn-multiple quantum coherence and nuclear Overhauser-multiple quantum coherence spectroscopy: application to interleukin 1. β ". In: *Biochemistry* 28.15 (1989), pp. 6150–6156.
- [16] L. Kay, G. Clore, A. Bax, and A. Gronenborn. "Four-dimensional heteronuclear triple-resonance NMR spectroscopy of interleukin-1 beta in solution". In: *Science* 249.4967 (1990), pp. 411–414.
- [17] K. Pervushin, R. Riek, G. Wider, and K. Wüthrich. "Attenuated T2 relaxation by mutual cancellation of dipole-dipole coupling and chemical shift anisotropy indicates an avenue to NMR structures of very large biological macromolecules in solution". In: *Proceedings of the National Academy of Sciences* 94.23 (1997), pp. 12366–12371.
- [18] R. R. Ernst. "Nuclear Magnetic Resonance Fourier Transform Spectroscopy (Nobel Lecture)". In: *Angewandte Chemie International Edition in English* 31.7 (1992), pp. 805–823.
- [19] K. Wüthrich. "NMR Studies of Structure and Function of Biological Macromolecules (Nobel Lecture)". In: *Angewandte Chemie International Edition* 42.29 (2003), pp. 3340–3363.
- [20] P. Abragam and A. Abragam. *The Principles of Nuclear Magnetism*. Comparative Pathobiology - Studies in the Postmodern Theory of Education. Clarendon Press, 1961.
- [21] M. Goldman. *Quantum description of high-resolution NMR in liquids*. eng. International series of monographs on chemistry ; 15. Oxford: Clarendon Press, 1988.
- [22] J. Cavanagh. *Protein NMR spectroscopy : principles and practice*. eng. 2nd ed. Burlington, Mass. ; London: Academic Press, 2007.
- [23] M. Goldman. "Formal Theory of Spin-Lattice Relaxation". In: *Journal of Magnetic Resonance* 149.2 (2001), pp. 160–187.
- [24] I. Kuprov, N. Wagner-Rundell, and P. Hore. "Bloch-Redfield-Wangsness theory engine implementation using symbolic processing software". In: *Journal of Magnetic Resonance* 184.2 (2007), pp. 196–206.

- [25] H. Kovacs, R. Kuemmerle, D. Moskau, and B. Perrone. “Cryogenic NMR Probes”. In: *Encyclopedia of Biophysics*. Ed. by G. Roberts and A. Watts. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, pp. 1–8.
- [26] J. Keeler. *Understanding NMR spectroscopy*. eng. 2nd ed. Chichester: Wiley, 2010.
- [27] C. E. Shannon. “Communication in the Presence of Noise”. In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21.
- [28] G. A. Morris, H. Barjat, and T. J. Home. “Reference deconvolution methods”. In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 31.2 (1997), pp. 197–257.
- [29] C. Tang. “An Analysis of Baseline Distortion and Offset in NMR Spectra”. In: *Journal of Magnetic Resonance, Series A* 109.2 (1994), pp. 232–240.
- [30] P. J. Hore. *Nuclear magnetic resonance*. eng. Second edition. Oxford chemistry primers. Oxford, 2015.
- [31] T. D. Claridge. *High-Resolution NMR Techniques in Organic Chemistry: Third Edition*. eng. 2016.
- [32] E. Bartholdi and R. Ernst. “Fourier spectroscopy and the causality principle”. In: *Journal of Magnetic Resonance (1969)* 11.1 (1973), pp. 9–19.
- [33] W. Dietrich, C. H. Rüdel, and M. Neumann. “Fast and precise automatic baseline correction of one- and two-dimensional nmr spectra”. In: *Journal of Magnetic Resonance (1969)* 91.1 (1991), pp. 1–11.
- [34] J. Carlos Cobas, M. A. Bernstein, M. Martín-Pastor, and P. G. Tahoces. “A new general-purpose fully automatic baseline-correction procedure for 1D and 2D NMR data”. In: *Journal of Magnetic Resonance* 183.1 (2006), pp. 145–151.
- [35] J. Keeler and D. Neuhaus. “Comparison and evaluation of methods for two-dimensional NMR spectra with absorption-mode lineshapes”. In: *Journal of Magnetic Resonance (1969)* 63.3 (1985), pp. 454–472.
- [36] W. Aue, J. Karhan, and R. Ernst. “Homonuclear broad band decoupling and two-dimensional J-resolved NMR spectroscopy”. In: *The Journal of Chemical Physics* 64.10 (1976), pp. 4226–4227.
- [37] G. A. Morris. “Two-Dimensional J -Resolved Spectroscopy”. In: *eMagRes*. John Wiley & Sons, Ltd, 2009.
- [38] A. L. Davis, J. Keeler, E. D. Laue, and D. Moskau. “Experiments for recording pure-absorption heteronuclear correlation spectra using pulsed field gradients”. In: *Journal of Magnetic Resonance (1969)* 98.1 (1992), pp. 207–216.

- [39] A.-H. Emwas, R. Roy, R. T. McKay, L. Tenori, E. Saccenti, G. A. N. Gowda, D. Raftery, F. Alahmari, L. Jaremko, M. Jaremko, and D. S. Wishart. “nmr spectroscopy for metabolomics research”. In: *Metabolites* 9.7 (2019).
- [40] M. P. Williamson. “Using chemical shift perturbation to characterise ligand binding”. In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 73 (2013), pp. 1–16.
- [41] D. S. Stephenson. “Linear prediction and maximum entropy methods in NMR spectroscopy”. In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 20.6 (1988), pp. 515–626.
- [42] P. Koehl. “Linear prediction spectral analysis of NMR data”. In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 34.3 (1999), pp. 257–299.
- [43] G. U. YULE. “On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer’s Sunspot Numbers”. eng. In: *Philosophical transactions of the Royal Society of London. Series A, Containing papers of a mathematical or physical character* 226 (1927), pp. 267–298.
- [44] G. T. Walker. “On periodicity in series of related terms”. eng. In: *Proceedings of the Royal Society of London. Series A, Containing papers of a mathematical and physical character* 131.818 (1931), pp. 518–532.
- [45] N. Levinson. “The Wiener (Root Mean Square) Error Criterion in Filter Design and Prediction”. In: *Journal of Mathematics and Physics* 25.1-4 (1946), pp. 261–278.
- [46] J. Durbin. “The Fitting of Time-Series Models”. eng. In: *Revue de l’Institut international de statistique* 28.3 (1960), p. 233.
- [47] R. Kumaresan and D. Tufts. “Estimating the parameters of exponentially damped sinusoids and pole-zero modeling in noise”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 30.6 (1982), pp. 833–840.
- [48] R. Kumaresan. “On the zeros of the linear prediction-error filter for deterministic signals”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 31.1 (1983), pp. 217–220.
- [49] S. Y. Kung, K. S. Arun, and D. V. B. Rao. “State-space and singular-value decomposition-based approximation methods for the harmonic retrieval problem”. In: *J. Opt. Soc. Am.* 73.12 (Dec. 1983), pp. 1799–1811.
- [50] H. Barkhuijsen, R. de Beer, W. Bovée, and D. van Ormondt. “Retrieval of frequencies, amplitudes, damping factors, and phases from time-domain signals using a linear least-squares procedure”. In: *Journal of Magnetic Resonance (1969)* 61.3 (1985), pp. 465–481.

- [51] H. Barkhuijsen, R. De Beer, W. M. M. J. Bovee, J. H. N. Creyghton, and D. Van Ormondt. “Application of linear prediction and singular value decomposition (LPSVD) to determine NMR frequencies and intensities from the FID”. In: *Magnetic Resonance in Medicine* 2.1 (1985), pp. 86–89.
- [52] H. Barkhuijsen, R. de Beer, and D. van Ormondt. “Improved algorithm for noniterative time-domain model fitting to exponentially damped magnetic resonance signals”. In: *Journal of Magnetic Resonance* (1969) 73.3 (1987), pp. 553–557.
- [53] R. De Beer, D. van Ormondt, W. Pijnappel, and J. van der Veen. “Quantitative Analysis of Magnetic Resonance Signals in the Time Domain”. In: *Israel Journal of Chemistry* 28.4 (1988), pp. 249–261.
- [54] W. Pijnappel, A. van den Boogaart, R. de Beer, and D. van Ormondt. “SVD-based quantification of magnetic resonance signals”. In: *Journal of Magnetic Resonance* (1969) 97.1 (1992), pp. 122–134.
- [55] Y. Hua and T. Sarkar. “Matrix pencil method for estimating parameters of exponentially damped/undamped sinusoids in noise”. In: *IEEE Trans. on Acoust. Speech Signal Process.* 38.5 (1990), pp. 814–824.
- [56] Y. Hua and T. Sarkar. “Matrix pencil and system poles”. In: *Signal Processing* 21.2 (1990), pp. 195–198.
- [57] Y. Hua and T. Sarkar. “On SVD for estimating generalized eigenvalues of singular matrix pencil in noise”. In: *IEEE Trans. Signal Process.* 39.4 (1991), pp. 892–900.
- [58] Y.-Y. Lin, P. Hodgkinson, M. Ernst, and A. Pines. “A Novel Detection–Estimation Scheme for Noisy NMR Signals: Applications to Delayed Acquisition Data”. In: *J. Magn. Reson.* 128.1 (1997), pp. 30–41.
- [59] S. Fricke, J. Seymour, M. Battistel, D. Freedberg, C. Eads, and M. Augustine. “Data processing in NMR relaxometry using the matrix pencil”. In: *Journal of Magnetic Resonance* 313 (2020), p. 106704.
- [60] D. Wörtge, M. Parziale, J. Claussen, B. Mohebbi, S. Stapf, B. Blümich, and M. Augustine. “Quantitative stray-field T1 relaxometry with the matrix pencil method”. In: *Journal of Magnetic Resonance* 351 (2023), p. 107435.
- [61] Y. Hua. “Estimating two-dimensional frequencies by matrix enhancement and matrix pencil”. eng. In: *IEEE transactions on signal processing* 40.9 (1992), pp. 2267–2280.
- [62] F.-J. Chen, C. Fung, C.-W. Kok, and S. Kwong. “Estimation of Two-Dimensional Frequencies Using Modified Matrix Pencil Method”. eng. In: *IEEE transactions on signal processing* 55.2 (2007), pp. 718–724.

- [63] G. H. Golub and V. Pereyra. “The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate”. In: *SIAM Journal on Numerical Analysis* 10.2 (1973), pp. 413–432.
- [64] J. W. C. van der Veen, R. de Beer, P. R. Luyten, and D. van Ormondt. “Accurate quantification of in vivo ^{31}P NMR signals using the variable projection method and prior knowledge”. In: *Magnetic Resonance in Medicine* 6.1 (1988), pp. 92–98.
- [65] C. Decanniere, P. Vanhecke, F. Vanstapel, H. Chen, S. Vanhuffel, C. Vandervoort, B. Vantomeren, and D. Vanormondt. “Evaluation of signal-processing methods for the quantification of strongly overlapping peaks in ^{31}P NMR spectra”. eng. In: *Journal of magnetic resonance. Series B* 105.1 (1994), pp. 31–37.
- [66] K. Levenberg. “A method for the solution of certain non-linear problems in least squares”. In: *Quarterly of Applied Mathematics* 2.2 (1944), pp. 164–168.
- [67] D. W. Marquardt. “An Algorithm for Least-Squares Estimation of Nonlinear Parameters”. In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.
- [68] L. Vanhamme, A. van den Boogaart, and S. Van Huffel. “Improved Method for Accurate and Efficient Quantification of MRS Data with Use of Prior Knowledge”. In: *Journal of Magnetic Resonance* 129.1 (1997), pp. 35–43.
- [69] K. Krishnamurthy. “CRAFT (complete reduction to amplitude frequency table) – robust and time-efficient Bayesian approach for quantitative mixture analysis by NMR”. In: *Magnetic Resonance in Chemistry* 51.12 (2013), pp. 821–829.
- [70] K. Krishnamurthy, A. M. Sefler, and D. J. Russell. “Application of CRAFT in two-dimensional NMR data processing”. In: *Magnetic Resonance in Chemistry* 55.3 (2017), pp. 224–232.
- [71] K. Krishnamurthy. “Complete Reduction to Amplitude Frequency Table (CRAFT)—A perspective”. In: *Magnetic Resonance in Chemistry* 59.8 (2021), pp. 757–791.
- [72] D.-W. Li, A. L. Hansen, C. Yuan, L. Bruschweiler-Li, and R. Brüschweiler. “DEEP picker is a deep neural network for accurate deconvolution of complex two-dimensional NMR spectra”. eng. In: *Nature communications* 12.1 (2021), pp. 5229–5229.
- [73] N. Schmid, S. Bruderer, F. Paruzzo, G. Fischetti, G. Toscano, D. Graf, M. Fey, A. Henrici, V. Ziebart, B. Heitmann, H. Grabner, J. Wegner, R. Sigel, and D. Wilhelm. “Deconvolution of 1D NMR spectra: A deep learning-based approach”. In: *Journal of Magnetic Resonance* 347 (2023), p. 107357.
- [74] R. Fletcher. *Practical methods of optimization*. eng. 2nd ed. Chichester: Wiley, 1987.

- [75] J. Nocedal and S. J. Wright. *Numerical optimization*. 2nd ed. Springer series in operations research. New York: Springer, 2006.
- [76] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013.
- [77] H. Akaike. “A new look at the statistical model identification”. In: *IEEE Trans. Automat. Control* 19.6 (1974), pp. 716–723.
- [78] G. Schwarz. “Estimating the Dimension of a Model”. In: *Ann. Stat.* 6.2 (1978), pp. 461–464.
- [79] J. Rissanen. “Modeling by shortest data description”. In: *Automatica* 14.5 (1978), pp. 465–471.
- [80] M. Wax and T. Kailath. “Detection of signals by information theoretic criteria”. In: *IEEE Trans. Acoust. Speech Signal Process.* 33.2 (1985), pp. 387–392.
- [81] Y. Pawitan. *In All Likelihood: Statistical Modelling and Inference Using Likelihood*. Oxford science publications. OUP Oxford, 2001.
- [82] N. I. Gould, D. Orban, A. Sartenaer, and P. L. Toint. “Sensitivity of trust-region algorithms to their parameters”. eng. In: *4OR* 3.3 (2005), pp. 227–241.
- [83] N. I. Fisher. *Statistical Analysis of Circular Data*. Cambridge University Press, 1993.
- [84] J. Tang and J. Norris. “LP-ZOOM, a linear prediction method for local spectral analysis of NMR signals”. In: *Journal of Magnetic Resonance (1969)* 79.1 (1988), pp. 190–196.
- [85] M. Mayzel, K. Kazimierczuk, and V. Y. Orekhov. “The causality principle in the reconstruction of sparse NMR spectra”. In: *Chem. Commun.* 50.64 (2014), pp. 8947–8950.
- [86] D. Gołowicz, P. Kasprzak, and K. Kazimierczuk. “Enhancing Compression Level for More Efficient Compressed Sensing and Other Lessons from NMR Spectroscopy”. In: *Sensors (Basel)* 20.5 (2020), p. 1325.
- [87] J. Luo, Q. Zeng, K. Wu, and Y. Lin. “Fast reconstruction of non-uniform sampling multidimensional NMR spectroscopy via a deep neural network”. In: *Journal of Magnetic Resonance* 317 (2020), p. 106772.
- [88] A. Verma, S. Bhattacharya, and B. Baishya. “Perfecting band selective homo-decoupling for decoupling two signals coupled within the same band”. In: *RSC Adv.* 8 (36 2018), pp. 19990–19999.
- [89] H. Y. Carr and E. M. Purcell. “Effects of Diffusion on Free Precession in Nuclear Magnetic Resonance Experiments”. In: *Phys. Rev.* 94 (3 May 1954), pp. 630–638.

- [90] S. Meiboom and D. Gill. “Modified Spin-Echo Method for Measuring Nuclear Relaxation Times”. In: *Review of Scientific Instruments* 29.8 (Aug. 1958), pp. 688–691.
- [91] C. Johnson. “Diffusion ordered nuclear magnetic resonance spectroscopy: principles and applications”. In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 34.3 (1999), pp. 203–256.
- [92] G. A. Morris. “Diffusion-Ordered Spectroscopy”. In: *eMagRes*. John Wiley & Sons, Ltd, 2009.
- [93] E. O. Stejskal and J. E. Tanner. “Spin Diffusion Measurements: Spin Echoes in the Presence of a Time-Dependent Field Gradient”. In: *The Journal of Chemical Physics* 42.1 (1965), pp. 288–292.
- [94] H. C. Torrey. “Bloch Equations with Diffusion Terms”. In: *Phys. Rev.* 104 (3 Nov. 1956), pp. 563–565.
- [95] J. E. Tanner. “Use of the Stimulated Echo in NMR Diffusion Studies”. In: *The Journal of Chemical Physics* 52.5 (1970), pp. 2523–2526.
- [96] R. Cotts, M. Hoch, T. Sun, and J. Markert. “Pulsed field gradient stimulated echo methods for improved NMR diffusion measurements in heterogeneous systems”. In: *Journal of Magnetic Resonance* (1969) 83.2 (1989), pp. 252–266.
- [97] D. Wu, A. Chen, and C. Johnson. “An Improved Diffusion-Ordered Spectroscopy Experiment Incorporating Bipolar-Gradient Pulses”. In: *Journal of Magnetic Resonance, Series A* 115.2 (1995), pp. 260–264.
- [98] M. D. Pelta, G. A. Morris, M. J. Stchedroff, and S. J. Hammond. “A one-shot sequence for high-resolution diffusion-ordered spectroscopy”. In: *Magnetic Resonance in Chemistry* 40.13 (2002), S147–S152.
- [99] D. Sinnaeve. “The Stejskal–Tanner equation generalized for any gradient shape—an overview of most pulse sequences measuring free diffusion”. In: *Concepts in Magnetic Resonance Part A* 40A.2 (2012), pp. 39–65.
- [100] A. Chen, C. S. Johnson, M. Lin, and M. J. Shapiro. “Chemical Exchange in Diffusion NMR Experiments”. eng. In: *Journal of the American Chemical Society* 120.35 (1998), pp. 9094–9095.
- [101] B. G. Davis and A. J. Fairbanks. *Carbohydrate chemistry*. eng. Oxford chemistry primers ; 99. Oxford: Oxford University Press, 2002.
- [102] M. Foroozandeh. “Spin dynamics during chirped pulses: applications to homonuclear decoupling and broadband excitation”. In: *Journal of Magnetic Resonance* 318 (2020), p. 106768.

- [103] J.-M. Bohlen, M. Rey, and G. Bodenhausen. “Refocusing with chirped pulses for broadband excitation without phase dispersion”. In: *Journal of Magnetic Resonance (1969)* 84.1 (1989), pp. 191–197.
- [104] J. Bohlen and G. Bodenhausen. “Experimental Aspects of Chirp NMR Spectroscopy”. In: *Journal of Magnetic Resonance, Series A* 102.3 (1993), pp. 293–301.
- [105] K. E. Cano, M. A. Smith, and A. Shaka. “Adjustable, Broadband, Selective Excitation with Uniform Phase”. In: *Journal of Magnetic Resonance* 155.1 (2002), pp. 131–139.
- [106] J. E. Power, M. Foroozandeh, R. W. Adams, M. Nilsson, S. R. Coombes, A. R. Phillips, and G. A. Morris. “Increasing the quantitative bandwidth of NMR measurements”. In: *Chem. Commun.* 52 (14 2016), pp. 2916–2919.
- [107] M. Foroozandeh, M. Nilsson, and G. A. Morris. “Improved ultra-broadband chirp excitation”. In: *Journal of Magnetic Resonance* 302 (2019), pp. 28–33.
- [108] E. Kupce and R. Freeman. “Adiabatic Pulses for Wideband Inversion and Broadband Decoupling”. In: *Journal of Magnetic Resonance, Series A* 115.2 (1995), pp. 273–276.
- [109] H. Maeda and Y. Yanagisawa. “Future prospects for NMR magnets: A perspective”. In: *Journal of Magnetic Resonance* 306 (2019), pp. 80–85.
- [110] A. Shaka, J. Keeler, T. Frenkel, and R. Freeman. “An improved sequence for broadband decoupling: WALTZ-16”. In: *Journal of Magnetic Resonance (1969)* 52.2 (1983), pp. 335–338.
- [111] A. Shaka, J. Keeler, and R. Freeman. “Evaluation of a new broadband decoupling sequence: WALTZ-16”. In: *Journal of Magnetic Resonance (1969)* 53.2 (1983), pp. 313–340.
- [112] A. Shaka, P. Barker, and R. Freeman. “Computer-optimized decoupling scheme for wideband applications and low-level operation”. In: *Journal of Magnetic Resonance (1969)* 64.3 (1985), pp. 547–552.
- [113] N. H. Meyer and K. Zangger. “Simplifying Proton NMR Spectra by Instant Homonuclear Broadband Decoupling”. In: *Angewandte Chemie International Edition* 52.28 (2013), pp. 7143–7146.
- [114] R. W. Adams. “Pure Shift NMR Spectroscopy”. In: *eMagRes*. John Wiley & Sons, Ltd, 2014, pp. 295–310.
- [115] K. Zangger. “Pure shift NMR”. In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 86-87 (2015), pp. 1–20.

- [116] A. Bax, R. Freeman, and G. A. Morris. “A simple method for suppressing dispersion-mode contributions in NMR spectra: The “pseudo echo””. In: *Journal of Magnetic Resonance* (1969) 43.2 (1981), pp. 333–338.
- [117] M. J. Thrippleton, R. A. Edden, and J. Keeler. “Suppression of strong coupling artefacts in J-spectra”. In: *Journal of Magnetic Resonance* 174.1 (2005), pp. 97–109.
- [118] G. Wider, R. Baumann, K. Nagayama, R. R. Ernst, and K. Wüthrich. “Strong spin-spin coupling in the two-dimensional J-resolved 360-MHz ^1H NMR spectra of the common amino acids”. In: *Journal of Magnetic Resonance* (1969) 42.1 (1981), pp. 73–87.
- [119] E. Kupce, J. Boyd, and I. Campbell. “Short Selective Pulses for Biochemical Applications”. In: *Journal of Magnetic Resonance, Series B* 106.3 (1995), pp. 300–303.
- [120] K. Zanger and H. Sterk. “Homonuclear Broadband-Decoupled NMR Spectra”. In: *Journal of Magnetic Resonance* 124.2 (1997), pp. 486–489.
- [121] J. A. Aguilar, S. Faulkner, M. Nilsson, and G. A. Morris. “Pure Shift ^1H NMR: A Resolution of the Resolution Problem?” In: *Angewandte Chemie International Edition* 49.23 (2010), pp. 3901–3903.
- [122] A. J. Pell and J. Keeler. “Two-dimensional J-spectra with absorption-mode lineshapes”. In: *Journal of Magnetic Resonance* 189.2 (2007), pp. 293–299.
- [123] J. Garbow, D. Weitekamp, and A. Pines. “Bilinear rotation decoupling of homonuclear scalar interactions”. In: *Chemical Physics Letters* 93.5 (1982), pp. 504–509.
- [124] A. Bax. “Broadband homonuclear decoupling in heteronuclear shift correlation NMR spectroscopy”. In: *Journal of Magnetic Resonance* (1969) 53.3 (1983), pp. 517–520.
- [125] L. Paudel, R. W. Adams, P. Király, J. A. Aguilar, M. Foroozandeh, M. J. Cliff, M. Nilsson, P. Sándor, J. P. Walther, and G. A. Morris. “Simultaneously Enhancing Spectral Resolution and Sensitivity in Heteronuclear Correlation NMR Spectroscopy”. In: *Angewandte Chemie International Edition* 52.44 (2013), pp. 11616–11619.
- [126] M. Foroozandeh, R. W. Adams, N. J. Meharry, D. Jeannerat, M. Nilsson, and G. A. Morris. “Ultrahigh-Resolution NMR Spectroscopy”. In: *Angewandte Chemie International Edition* 53.27 (2014), pp. 6990–6992.
- [127] M. Foroozandeh, G. A. Morris, and M. Nilsson. “PSYCHE Pure Shift NMR Spectroscopy”. In: *Chemistry – A European Journal* 24.53 (2018), pp. 13988–14000.
- [128] M. Foroozandeh, R. Adams, P. Kiraly, M. Nilsson, and G. Morris. “Measuring couplings in crowded NMR spectra: Pure shift NMR with multiplet analysis”. eng. In: *Chemical communications (Cambridge, England)* 51.84 (2015), pp. 15410–15413.

- [129] P. Kiraly, M. Foroozandeh, M. Nilsson, and G. A. Morris. “Anatomising proton NMR spectra with pure shift 2D J-spectroscopy: A cautionary tale”. In: *Chemical Physics Letters* 683 (2017). Ahmed Zewail (1946-2016) Commemoration Issue of Chemical Physics Letters, pp. 398–403.
- [130] J.-M. Nuzillard. “Time-Reversal of NMR Signals by Linear Prediction. Application to Phase-Sensitive HomonuclearJ-Resolved Spectroscopy”. In: *Journal of Magnetic Resonance, Series A* 118.1 (1996), pp. 132–135.
- [131] A. Martinez, F. Bourdreux, E. Riguet, and J.-M. Nuzillard. “High-resolution and high-sensitivity 2D homonuclear J-resolved NMR spectroscopy”. In: *Magnetic Resonance in Chemistry* 50.1 (2012), pp. 28–32.
- [132] P. Mutzenhardt, F. Guenneau, and D. Canet. “A Procedure for Obtaining Pure Absorption 2D J-Spectra: Application to Quantitative Fully J-Decoupled Homonuclear NMR Spectra”. In: *Journal of Magnetic Resonance* 141.2 (1999), pp. 312–321.
- [133] G. Zhu, D. Smith, and Y. Hua. “Post-Acquisition Solvent Suppression by Singular-Value Decomposition”. In: *Journal of Magnetic Resonance* 124.1 (1997), pp. 286–289.
- [134] P. Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nat. Methods* 17 (2020), pp. 261–272.
- [135] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, and E. Wieser. “Array programming with NumPy.” In: *Nature* 585.7825 (2020), p. 357.
- [136] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.
- [137] N. Yilmazer, R. Fernandez-Recio, and T. K. Sarkar. “Matrix pencil method for simultaneously estimating azimuth and elevation angles of arrival along with the frequency of the incoming signals”. In: *Digital Signal Processing* 16.6 (2006), pp. 796–816.
- [138] G. Strang. *Linear Algebra and Its Applications*. 5 ed. Florence: Cengage Learning, Inc, 2018.
- [139] H. Hogben, M. Krzstyniak, G. Charnock, P. Hore, and I. Kuprov. “Spinach – A software library for simulation of spin dynamics in large spin systems”. eng. In: *Journal of magnetic resonance* (1997) 208.2 (2011), pp. 179–194.
- [140] *MATLAB engine for PYTHON*. <https://uk.mathworks.com/help/matlab/matlab-engine-for-python.html>. Accessed 24/7/2023.
- [141] S. Berger and S. Braun. *200 and more NMR experiments: a practical course*. eng. [3rd rev. and expanded ed.] Weinheim: Wiley-VCH, 2004.

- [142] *SPINACH relaxation theory parameters.* https://www.spindynamics.org/wiki/index.php?title=Relaxation_theory_parameters. Accessed 24/7/2023.
- [143] L. A. O'Dell. "The WURST kind of pulses in solid-state NMR". In: *Solid State Nuclear Magnetic Resonance* 55-56 (2013), pp. 28–41.

NMR GLOSSARY

A

This glossary comprises many terms related to NMR which are mentioned, but not necessarily described in much detail in the main text.

TODO

Chemical Shift The term chemical shift typically refers to the resonance frequency of a nucleus relative to some standard frequency:

$$\delta = \frac{\omega - \omega_{\text{ref}}}{\omega_{\text{ref}}} \quad (\text{A.1})$$

The numerator and denominator are usually given in units of Hz and MHz, respectively. As such, the unit of the chemical shift is 10^{-6} , i.e. “parts per million” (ppm).

ADDITIONAL THEORY

B

B.1 Mathematical definitions

Descriptions of linear algebra and statics concepts that are referred to in this thesis are provided here. More detail can be found in numerous relevant texts[81, 138].

B.1.1 Linear algebra

TODO: null space, rank, range

Singular value decomposition

SVD is a generalisation of eigendecomposition for a matrix of any shape. Given a matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$, the SVD is a factorisation given by

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\dagger. \quad (\text{B.1})$$

The matrices that make up the decomposition are as follows

$\Sigma \in \mathbb{C}^{m \times n}$ is a rectangular diagonal matrix with diagonal elements comprising the *singular values* in descending order of magnitude. The singular values are the square roots of the non-zero eigenvalues of both $\mathbf{A}^\dagger \mathbf{A}$ and $\mathbf{A} \mathbf{A}^\dagger$.

$\mathbf{U} \in \mathbb{C}^{m \times m}$ is a unitary matrix whose columns comprise the *left singular vectors*. The left singular vectors are the eigenvectors of the matrix $\mathbf{A} \mathbf{A}^\dagger$.

$\mathbf{V} \in \mathbb{C}^{n \times n}$ is a unitary matrix whose columns comprise the *right singular vectors*. The right singular vectors are the eigenvectors of the matrix $\mathbf{A}^\dagger \mathbf{A}$.

One fundamental property of the SVD is that the number of non-zero singular values is equivalent to the rank of the matrix. As the EYM theorem highlights, the SVD is valuable in constructing

low-rank approximations of matrices, with applications in various fields such as signal processing, (see Section 2.2.1) and data compression.

Special matrices

A *Hankel matrix* is a matrix in which each ascending diagonal from left to right possesses identical elements. While Hankel matrices are often defined to be square, in this work such a restriction is not applied. Given a matrix $\mathbf{X} \in \mathbb{F}^{M \times N}$, the matrix is Hankel if

$$x_{m,n} = x_{m+1,n-1} \quad (\text{B.2})$$

$\forall m \in \{1, \dots, M - 1\} \forall n \in \{2, \dots, N\}$. Similarly, a *Toeplitz matrix* is a matrix in which every descending diagonal from left to right possesses identical elements, i.e.

$$x_{m,n} = x_{m+1,n+1} \quad (\text{B.3})$$

$\forall m \in \{1, \dots, M - 1\} \forall n \in \{1, \dots, N - 1\}$.

B.1.2 Statistics and probability

TODO: Likelihood function, MLE

Probability density function

The pdf $p(x) : \mathbb{R} \rightarrow \mathbb{R}$ is a function over a continuous sample space which provides relative likelihoods between potential values of x . The probability that a random sample obeying a known distribution lies within the range $[x_a, x_b]$ is given by the integral

$$P(x_a \leq x \leq x_b) = \int_{x_a}^{x_b} p(x) dx. \quad (\text{B.4})$$

The integral of $p(x)$ over the entire sample space is defined to be unity. Also, the pdf of a specific value is always 0, since the width of the region of integration is 0.

Likelihood function

B.2 Multidimensional virtual echos

The VE concept (Section 2.5.1) can be generalised to any number of dimensions, assuming that a pair of amplitude-modulated signals exist for each indirect-dimension. Thus a set of 2^{D-1} signals is required for a D -dimensional FID. For the 2D case, this corresponds to the pair of signals

$\{\mathbf{Y}^{\cos}, \mathbf{Y}^{\sin}\}$, given by Equation 1.22 with $D = 2$ and $\zeta = \{\cos(\cdot), \sin(\cdot)\}$, taking the forms (with noise neglected)

$$y_{n^{(1)}, n^{(2)}}^{\cos} = \xi_{n^{(1)}, n^{(2)}} c_{n^{(1)}, n^{(2)}}^{(1)} \left(c_{n^{(1)}, n^{(2)}}^{(2)} + i s_{n^{(1)}, n^{(2)}}^{(2)} \right), \quad (\text{B.5a})$$

$$y_{n^{(1)}, n^{(2)}}^{\sin} = \xi_{n^{(1)}, n^{(2)}} s_{n^{(1)}, n^{(2)}}^{(1)} \left(c_{n^{(1)}, n^{(2)}}^{(2)} + i s_{n^{(1)}, n^{(2)}}^{(2)} \right), \quad (\text{B.5b})$$

$$\xi_{n^{(1)}, n^{(2)}} = \sum_m a_m \exp \left(-\eta_m^{(1)} n^{(1)} \Delta_t^{(1)} - \eta_m^{(2)} n^{(2)} \Delta_t^{(2)} \right), \quad (\text{B.5c})$$

$$(c/s)_{n^{(1)}, n^{(2)}}^{(1/2)} = \sum_m \cos / \sin \left(2\pi f^{(1/2)} n^{(1/2)} \Delta^{(1/2)} \right). \quad (\text{B.5d})$$

Four matrices $\psi_{\pm\pm}$ are then constructed of the form

$$\begin{aligned} \psi_{\pm\pm, n^{(1)}, n^{(2)}} &= \xi_{n^{(1)}, n^{(2)}} \left(c_{n^{(1)}, n^{(2)}}^{(1)} \pm^{(1)} i s_{n^{(1)}, n^{(2)}}^{(1)} \right) \left(c_{n^{(1)}, n^{(2)}}^{(2)} \pm^{(2)} i s_{n^{(1)}, n^{(2)}}^{(2)} \right) \\ &\equiv \Re \left(y_{n^{(1)}, n^{(2)}}^{\cos} \right) \pm^{(1)} \pm^{(2)} - \Im \left(y_{n^{(1)}, n^{(2)}}^{\sin} \right) + i \left(\pm^{(1)} \Re \left(y_{n^{(1)}, n^{(2)}}^{\sin} \right) \pm^{(2)} \Im \left(y_{n^{(1)}, n^{(2)}}^{\cos} \right) \right), \end{aligned} \quad (\text{B.6})$$

from which the matrices $\mathbf{T}_{1 \rightarrow 4} \in \mathbb{C}^{2N^{(1)} \times 2N^{(2)}}$ are generated:

$$\mathbf{T}_1 = \begin{bmatrix} \mathbf{Y}_{++} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad (\text{B.7a})$$

$$\mathbf{T}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{Y}_{-+}^{\leftrightarrow(1)} & \mathbf{0} \end{bmatrix}^{\circlearrowleft(1)}, \quad (\text{B.7b})$$

$$\mathbf{T}_3 = \begin{bmatrix} \mathbf{0} & \mathbf{Y}_{+-}^{\leftrightarrow(2)} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}^{\circlearrowleft(2)}, \quad (\text{B.7c})$$

$$\mathbf{T}_4 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}_{--}^{\leftrightarrow(1,2)} \end{bmatrix}^{\circlearrowleft(1,2)}. \quad (\text{B.7d})$$

The virtual echo is then given by $\mathbf{Y}_{\text{ve}} = \sum_{i=1}^4 \mathbf{T}_i$, with the first row and column divided by two. For a full outline of the 2D filtering procedure, see Algorithm B.3.

It is possible to construct a virtual echo using an appropriate set of phase-modulated signals too, which for the 2D case would be $\{\mathbf{Y}^{\text{pos}}, \mathbf{Y}^{\text{neg}}\}$, given by Equation 1.22 with $D = 2$ and $\zeta = \{\exp(i\cdot), \exp(-i\cdot)\}$. These can be used to generate an amplitude modulated pair via

$$\mathbf{Y}^{\cos} = \frac{\mathbf{Y}^{\text{pos}} + \mathbf{Y}^{\text{neg}}}{2}, \quad (\text{B.8a})$$

$$\mathbf{Y}^{\sin} = \frac{\mathbf{Y}^{\text{pos}} - \mathbf{Y}^{\text{neg}}}{2i}. \quad (\text{B.8b})$$

B.3 Additional algorithms

ALGORITHM B.1 The MMEMPM.

1: **procedure** MMEMPM($\mathbf{Y} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}, M \in \mathbb{N}$)
2: $L^{(1)}, L^{(2)} \leftarrow \lfloor N^{(1)}/2 \rfloor, \lfloor N^{(2)}/2 \rfloor;$
3: **for** $n^{(1)} \leftarrow \{0, \dots, N^{(1)} - 1\}$ **do**
4: $\mathbf{H}_{Y,n^{(1)}} \leftarrow \begin{bmatrix} \mathbf{Y}[n^{(1)}, 0] & \mathbf{Y}[n^{(1)}, 1] & \dots & \mathbf{Y}[n^{(1)}, N^{(2)} - L^{(2)}] \\ \mathbf{Y}[n^{(1)}, 1] & \mathbf{Y}[n^{(1)}, 2] & \dots & \mathbf{Y}[n^{(1)}, N^{(2)} - L^{(2)} + 1] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Y}[n^{(1)}, L^{(2)} - 1] & \mathbf{Y}[n^{(1)}, L^{(2)}] & \dots & \mathbf{Y}[n^{(1)}, N^{(2)} - 1] \end{bmatrix}$
5: **end for;**
6: $\mathbf{E}_Y \leftarrow \begin{bmatrix} \mathbf{H}_{Y,0} & \mathbf{H}_{Y,1} & \dots & \mathbf{H}_{Y,N^{(1)} - L^{(1)}} \\ \mathbf{H}_{Y,1} & \mathbf{H}_{Y,2} & \dots & \mathbf{H}_{Y,N^{(1)} - L^{(1)} + 1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{H}_{Y,L^{(1)} - 1} & \mathbf{H}_{Y,L^{(1)}} & \dots & \mathbf{H}_{Y,N^{(1)} - 1} \end{bmatrix};$
7: $\mathbf{U}_M, \Sigma_M, \mathbf{V}_M^\dagger \leftarrow \text{TRUNCATEDSVD}(\mathbf{E}_Y, M);$
8: $\mathbf{P} \leftarrow \mathbf{0} \in \mathbb{C}^{L^{(1)}L^{(2)} \times L^{(1)}L^{(2)}};$
9: $r \leftarrow 0$
10: **for** $i = 0, \dots, L^{(2)} - 1$ **do**
11: **for** $j = 0, \dots, L^{(1)} - 1$ **do**
12: $c \leftarrow i + jL^{(2)};$
13: $\mathbf{P}[r, c] \leftarrow 1;$
14: $r = r + 1;$
15: **end for**
16: **end for**
17: $\mathbf{U}_{M1}, \mathbf{U}_{M2} \leftarrow \mathbf{U}_M[:, L^{(1)}(L^{(2)} - 1)], \mathbf{U}_M[L^{(2)} :];$ ▷ Last/First $L^{(2)}$ rows deleted
18: $\mathbf{z}^{(1)}, \mathbf{W}^{(1)} \leftarrow \text{EIGENDECOMPOSITION}(\mathbf{U}_{M1}^+ \mathbf{U}_{M2});$
19: $\mathbf{f}^{(1)}, \boldsymbol{\eta}^{(1)} \leftarrow (f_{\text{sw}}^{(1)}/2\pi) \Im(\ln \mathbf{z}^{(1)}) + f_{\text{off}}^{(1)}, -f_{\text{sw}}^{(1)} \Re(\ln \mathbf{z}^{(1)});$
20: $\mathbf{U}_{MP} \leftarrow \mathbf{P} \mathbf{U}_M;$
21: $\mathbf{U}_{MP1}, \mathbf{U}_{MP2} \leftarrow \mathbf{U}_{MP}[:, (L^{(1)} - 1)L^{(2)}], \mathbf{U}_{MP}[L^{(1)} :];$ ▷ Last/First $L^{(1)}$ rows deleted
22: $\mathbf{z}^{(2)} \leftarrow \text{diag}([\mathbf{W}^{(1)}]^{-1} \mathbf{U}_{MP1}^+ \mathbf{U}_{MP2} \mathbf{W}^{(1)});$
23: $\mathbf{f}^{(2)}, \boldsymbol{\eta}^{(2)} \leftarrow (f_{\text{sw}}^{(2)}/2\pi) \Im(\ln \mathbf{z}^{(2)}) + f_{\text{off}}^{(2)}, -f_{\text{sw}}^{(2)} \Re(\ln \mathbf{z}^{(2)});$
24: $\mathbf{Z}_L^{(2)} = [\mathbf{1} \quad \mathbf{z}^{(2)} \quad \mathbf{z}^{(2)2} \quad \dots \quad \mathbf{z}^{(2)L^{(2)}-1}]^T;$
25: $\mathbf{Z}_R^{(2)} \leftarrow [\mathbf{1} \quad \mathbf{z}^{(2)} \quad \mathbf{z}^{(2)2} \quad \dots \quad \mathbf{z}^{(2)N^{(2)}-L^{(2)}}];$
26: $\mathbf{Z}_D^{(1)} \leftarrow \text{diag}(\mathbf{z}^{(1)});$
27: $\mathbf{E}_L \leftarrow \begin{bmatrix} \mathbf{Z}_L^{(2)} \\ \mathbf{Z}_L^{(2)} \mathbf{Z}_D^{(1)} \\ \vdots \\ \mathbf{Z}_L^{(2)} [\mathbf{Z}_D^{(1)}]^{L^{(1)}-1} \end{bmatrix};$
28: $\mathbf{E}_R \leftarrow [\mathbf{Z}_R^{(2)} \quad \mathbf{Z}_D^{(1)} \mathbf{Z}_R^{(2)} \quad \dots \quad [\mathbf{Z}_D^{(1)}]^{N^{(1)}-L^{(1)}} \mathbf{Z}_R^{(2)}];$
29: $\boldsymbol{\alpha} \leftarrow \text{diag}(\mathbf{E}_L^+ \mathbf{E}_Y \mathbf{E}_R^+);$
30: $\boldsymbol{\alpha}, \boldsymbol{\phi} \leftarrow |\boldsymbol{\alpha}|, \arctan\left(\frac{\Im(\boldsymbol{\alpha})}{\Re(\boldsymbol{\alpha})}\right);$
31: $\boldsymbol{\theta}^{(0)} \leftarrow [\boldsymbol{\alpha}^T \quad \boldsymbol{\phi}^T \quad [\mathbf{f}^{(1)}]^T \quad [\mathbf{f}^{(2)}]^T \quad [\boldsymbol{\eta}^{(1)}]^T \quad [\boldsymbol{\eta}^{(2)}]^T]^T;$
32: **return** $\boldsymbol{\theta}^{(0)}$
33: **end procedure**

ALGORITHM B.2 Steihaug-Toint method for determining an update for nonlinear programming. This is equivalent to Algorithm 7.2 in [75].

```

1: procedure STEIHAUGTOINT( $\mathbf{Y} \in \mathbb{C}^{N^{(1)} \times \dots \times N^{(D)}}$ ,  $\boldsymbol{\theta}^{(k)} \in \mathbb{R}^{2(1+D)M}$ ,  $\Delta^{(k)} \in \mathbb{R}_{>0}$ )
2:    $\mathbf{g} \leftarrow \nabla \mathcal{F}_\phi(\boldsymbol{\theta}^{(k)} | \mathbf{Y})$ ; ▷ Grad vector: Equation 2.45a
3:    $\mathbf{H} \leftarrow \nabla^2 \mathcal{F}_\phi(\boldsymbol{\theta}^{(k)} | \mathbf{Y})$ ; ▷ Hessian matrix, either exact: Equation 2.45b or approximate: Equation 2.48
4:    $\epsilon^{(k)} \leftarrow \min(1/2, \sqrt{\|\mathbf{g}\|}) \|\mathbf{g}\|$ ;
5:    $\mathbf{z}^{(0)} \leftarrow \mathbf{0} \in \mathbb{R}^{6M}$ ;
6:    $\mathbf{r}^{(0)} \leftarrow \mathbf{g}$ ;
7:    $\mathbf{d}^{(0)} \leftarrow -\mathbf{r}^{(0)}$ ;
8:   if  $\|\mathbf{r}^{(0)}\| < \epsilon^{(k)}$  then
9:     return  $\mathbf{z}^{(0)}$ ;
10:  end if
11:  for  $j = \{0, 1, \dots\}$  do
12:    if  $\mathbf{d}^{(j)^\top} \mathbf{H} \mathbf{d}^{(j)} \leq 0$  then
13:      Find  $\tau$  such that  $\mathbf{p}^{(k)} = \mathbf{z}^{(j)} + \tau \mathbf{d}^{(j)}$  minimises  $\mathcal{F}_{\phi Q}(\boldsymbol{\theta}^{(k)} + \mathbf{p}^{(k)})$ , subject to  $\|\mathbf{p}^{(k)}\| = \Delta^{(k)}$ ;
14:      return  $\mathbf{p}^{(k)}$ ;
15:    end if
16:     $\alpha^{(j)} \leftarrow \frac{\mathbf{r}^{(j)^\top} \mathbf{r}^{(j)}}{\mathbf{d}^{(j)^\top} \mathbf{H} \mathbf{d}^{(j)}}$ ;
17:     $\mathbf{z}^{(j+1)} \leftarrow \mathbf{z}^{(j)} + \alpha^{(j)} \mathbf{d}^{(j)}$ ;
18:    if  $\|\mathbf{z}^{(j+1)}\| < \epsilon^{(k)}$  then
19:      Find  $\tau \in \mathbb{R}_{>0}$  such that  $\mathbf{p}^{(k)} = \mathbf{z}^{(j)} + \tau \mathbf{d}^{(j)}$  satisfies  $\|\mathbf{p}^{(k)}\| = \Delta^{(k)}$ ;
20:      return  $\mathbf{p}^{(k)}$ ;
21:    end if
22:     $\mathbf{r}^{(j+1)} \leftarrow \mathbf{r}^{(j)} + \alpha^{(j)} \mathbf{H} \mathbf{d}^{(j)}$ ;
23:    if  $\|\mathbf{r}^{(j+1)}\| < \epsilon^{(k)}$  then
24:      return  $\mathbf{z}^{(j+1)}$ ;
25:    end if
26:     $\beta^{(j+1)} \leftarrow \frac{\mathbf{r}^{(j+1)^\top} \mathbf{r}^{(j+1)}}{\mathbf{r}^{(j)^\top} \mathbf{r}^{(j)}}$ ;
27:     $\mathbf{d}^{(j+1)} \leftarrow -\mathbf{r}^{(j+1)} + \beta^{(j+1)} \mathbf{d}^{(j)}$ ;
28:  end for
29: end procedure

```

ALGORITHM B.3 Filtering procedure for 2D data.

```

1: procedure FILTER2D( $\mathbf{Y}_{\text{cos}} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}, \mathbf{Y}_{\text{sin}} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}, \mathbf{R}_{\text{interest}} \in \mathbb{N}_0^{2 \times 2}, \mathbf{R}_{\text{noise}} \in \mathbb{N}_0^{2 \times 2}$ )
2:    $\mathbf{Y}_{\text{ve}} \leftarrow \text{VIRTUALECHO2D}(\mathbf{Y}_{\text{cos}}, \mathbf{Y}_{\text{sin}});$ 
3:    $\mathbf{S}_{\text{ve}} \leftarrow \text{FT}(\mathbf{Y}_{\text{ve}});$ 
4:    $l_{\text{idx}}^{(1)}, r_{\text{idx}}^{(1)}, l_{\text{idx}}^{(2)}, r_{\text{idx}}^{(2)} \leftarrow \mathbf{R}_{\text{interest}}[0, 0], \mathbf{R}_{\text{interest}}[0, 1], \mathbf{R}_{\text{interest}}[1, 0], \mathbf{R}_{\text{interest}}[1, 1];$ 
5:   for  $d = 1, 2$  do
6:      $c_{\text{idx}}^{(d)} \leftarrow (l_{\text{idx}}^{(d)} + r_{\text{idx}}^{(d)})/2;$ 
7:      $b_{\text{idx}}^{(d)} \leftarrow r_{\text{idx}}^{(d)} - l_{\text{idx}}^{(d)};$ 
8:      $\mathbf{g}^{(d)} \leftarrow \text{SUPERGAUSSIAN1D}(2N^{(d)}, c_{\text{idx}}^{(d)}, b_{\text{idx}}^{(d)});$ 
9:      $\mathbf{G} \leftarrow \mathbf{g}^{(1)} \otimes \mathbf{g}^{(2)};$ 
10:    end for
11:     $l_{\text{idx,noise}}^{(1)}, r_{\text{idx,noise}}^{(1)}, l_{\text{idx,noise}}^{(2)}, r_{\text{idx,noise}}^{(2)} \leftarrow \mathbf{R}_{\text{noise}}[0, 0], \mathbf{R}_{\text{noise}}[0, 1], \mathbf{R}_{\text{noise}}[1, 0], \mathbf{R}_{\text{noise}}[1, 1];$ 
12:     $\mathbf{S}_{\text{noise}} \leftarrow \mathbf{S}_{\text{ve}} \left[ l_{\text{idx,noise}}^{(1)} : r_{\text{idx,noise}}^{(1)} + 1, l_{\text{idx,noise}}^{(2)} : r_{\text{idx,noise}}^{(2)} + 1 \right]$ 
13:     $\sigma^2 \leftarrow \text{Var}(\mathbf{S}_{\text{noise}});$ 
14:     $\mathbf{W}_{\sigma^2} \leftarrow \mathbf{0} \in \mathbb{R}^{2N^{(1)} \times 2N^{(2)}};$ 
15:    for  $n^{(1)} = 0, \dots, 2N^{(1)} - 1$  do
16:      for  $n^{(2)} = 0, \dots, 2N^{(2)} - 1$  do
17:         $\mathbf{W}_{\sigma^2}[n^{(1)}, n^{(2)}] \leftarrow \text{RANDOMSAMPLE}(\mathcal{N}(0, \sigma^2));$ 
18:      end for
19:    end for
20:     $\tilde{\mathbf{S}}_{\text{ve}} \leftarrow \mathbf{S}_{\text{ve}} \odot \mathbf{G} + \mathbf{W}_{\sigma^2} \odot (\mathbf{1} - \mathbf{G});$ 
21:     $\tilde{\mathbf{Y}}_{\text{ve}} \leftarrow \text{IFT}(\tilde{\mathbf{S}}_{\text{ve}});$ 
22:     $\tilde{\mathbf{Y}} \leftarrow \tilde{\mathbf{Y}}_{\text{ve}} \left[ :N^{(1)}, :N^{(2)} \right];$ 
23:    return  $\tilde{\mathbf{Y}};$ 
24: end procedure

25: procedure VIRTUALECHO2D( $\mathbf{Y}_{\text{cos}} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}, \mathbf{Y}_{\text{sin}} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}$ )
26:    $\mathbf{Y}_{++} \leftarrow \Re(\mathbf{Y}_{\text{cos}}) - \Im(\mathbf{Y}_{\text{sin}}) + i(\Im(\mathbf{Y}_{\text{cos}}) + \Re(\mathbf{Y}_{\text{sin}}));$ 
27:    $\mathbf{Y}_{+-} \leftarrow \Re(\mathbf{Y}_{\text{cos}}) + \Im(\mathbf{Y}_{\text{sin}}) + i(\Re(\mathbf{Y}_{\text{sin}}) - \Im(\mathbf{Y}_{\text{cos}}));$ 
28:    $\mathbf{Y}_{-+} \leftarrow \Re(\mathbf{Y}_{\text{cos}}) + \Im(\mathbf{Y}_{\text{sin}}) + i(\Im(\mathbf{Y}_{\text{cos}}) - \Re(\mathbf{Y}_{\text{sin}}));$ 
29:    $\mathbf{Y}_{--} \leftarrow \Re(\mathbf{Y}_{\text{cos}}) - \Im(\mathbf{Y}_{\text{sin}}) - i(\Im(\mathbf{Y}_{\text{cos}}) + \Re(\mathbf{Y}_{\text{sin}}));$ 
30:    $\mathbf{Z} \leftarrow \mathbf{0} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}$ 
31:    $\mathbf{T}_1 \leftarrow \begin{bmatrix} \mathbf{Y}_{++} & \mathbf{Z} \\ \mathbf{Z} & \mathbf{Z} \end{bmatrix};$ 
32:    $\mathbf{T}_2 \leftarrow \begin{bmatrix} \mathbf{Z} & \mathbf{Y}_{+-}^{(2)} \\ \mathbf{Z} & \mathbf{Z} \end{bmatrix}^{\odot(2)};$ 
33:    $\mathbf{T}_3 \leftarrow \begin{bmatrix} \mathbf{Z} & \mathbf{Z} \\ \mathbf{Y}_{-+}^{(1)} & \mathbf{Z} \end{bmatrix}^{\odot(1)}$ 
34:    $\mathbf{T}_4 \leftarrow \begin{bmatrix} \mathbf{Z} & \mathbf{Z} \\ \mathbf{Z} & \mathbf{Y}_{--}^{(1,2)} \end{bmatrix}^{\odot(1,2)};$ 
35:    $\mathbf{Y}_{\text{ve}} \leftarrow \mathbf{T}_1 + \mathbf{T}_2 + \mathbf{T}_3 + \mathbf{T}_4;$ 
36:   for  $n^{(1)} = 0, \dots, 2N^{(1)} - 1$  do
37:      $\mathbf{Y}_{\text{ve}}[n^{(1)}, 0] \leftarrow \mathbf{Y}_{\text{ve}}[n^{(1)}, 0]/2;$ 
38:   end for
39:   for  $n^{(2)} = 0, \dots, 2N^{(2)} - 1$  do
40:      $\mathbf{Y}_{\text{ve}}[0, n^{(2)}] \leftarrow \mathbf{Y}_{\text{ve}}[0, n^{(2)}]/2;$ 
41:   end for
42:   return  $\mathbf{Y}_{\text{ve}};$ 
43: end procedure

```

ALGORITHM B.4 Filtering procedure for 2DJ data.

```

1: procedure FILTER2DJ(  $\mathbf{Y} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}$ ,  $\mathbf{r}_{\text{interest}} \in \mathbb{N}_0^2$ ,  $\mathbf{r}_{\text{noise}} \in \mathbb{N}_0^2$  )
2:    $\mathbf{Y}_{\text{ve}} \leftarrow \mathbf{0} \in \mathbb{C}^{N^{(1)} \times 2N^{(2)}}$ ;
3:   for  $n^{(1)} = 0, \dots, N^{(1)} - 1$  do
4:      $\mathbf{Y}_{\text{ve}} \left[ n^{(1)}, : \right] \leftarrow \text{VIRTUALECHO1D} \left( \mathbf{Y} \left[ n^{(1)}, : \right] \right);$ 
5:   end for
6:    $\mathbf{S}_{\text{ve}} \leftarrow \text{FT}^{(2)} \left( \mathbf{Y}_{\text{ve}} \right);$ 
7:    $l_{\text{idx}}^{(2)}, r_{\text{idx}}^{(2)} \leftarrow \mathbf{r}_{\text{interest}}[0], \mathbf{r}_{\text{interest}}[1];$ 
8:    $l_{\text{idx,noise}}^{(2)}, r_{\text{idx,noise}}^{(2)} \leftarrow \mathbf{r}_{\text{noise}}[0], \mathbf{r}_{\text{noise}}[1];$ 
9:    $c_{\text{idx}}^{(2)} \leftarrow (l_{\text{idx}}^{(2)} + r_{\text{idx}}^{(2)})/2;$ 
10:   $b_{\text{idx}}^{(2)} \leftarrow r_{\text{idx}}^{(2)} - l_{\text{idx}}^{(2)};$ 
11:   $\mathbf{g}^{(1)} \leftarrow \mathbf{1} \in \mathbb{R}^{N^{(1)}}$ ;
12:   $\mathbf{g}^{(2)} \leftarrow \text{SUPERGAUSSIAN1D} \left( N^{(2)}, c_{\text{idx}}^{(2)}, b_{\text{idx}}^{(2)} \right);$ 
13:   $\mathbf{G} \leftarrow \mathbf{g}^{(1)} \otimes \mathbf{g}^{(2)}$ ;
14:   $\mathbf{S}_{\text{noise}} \leftarrow \mathbf{S}_{\text{ve}} \left[ :, l_{\text{idx,noise}}^{(2)} : r_{\text{idx,noise}}^{(2)} + 1 \right];$ 
15:   $\sigma^2 \leftarrow \text{Var}(\mathbf{S}_{\text{noise}});$ 
16:   $\mathbf{W}_{\sigma^2} \leftarrow \mathbf{0} \in \mathbb{R}^{N^{(1)} \times 2N^{(2)}}$ ;
17:  for  $n^{(1)} = 0, \dots, N^{(1)} - 1$  do
18:    for  $n^{(2)} = 0, \dots, 2N^{(2)} - 1$  do
19:       $\mathbf{W}_{\sigma^2} \left[ n^{(1)}, n^{(2)} \right] \leftarrow \text{RANDOMSAMPLE} \left( \mathcal{N}(0, \sigma^2) \right);$ 
20:    end for
21:  end for
22:   $\tilde{\mathbf{S}}_{\text{ve}} \leftarrow \mathbf{S}_{\text{ve}} \odot \mathbf{G} + \mathbf{W}_{\sigma^2} \odot (\mathbf{1} - \mathbf{G});$ 
23:   $\tilde{\mathbf{Y}}_{\text{ve}} \leftarrow \text{IFT}^{(2)} \left( \tilde{\mathbf{S}}_{\text{ve}} \right);$ 
24:   $\tilde{\mathbf{Y}} \leftarrow \tilde{\mathbf{Y}}_{\text{ve}} \left[ :, : N^{(2)} \right];$ 
25:  return  $\tilde{\mathbf{Y}}$ ;
26: end procedure

```

CODE LISTINGS

C

NEED TO ADD TEXT TO DESCRIBE THINGS The following is all the required imports:

CODE LISTING C.1: Required imports for the subsequent listings in this chapter. The `product` function is employed in the MMEMPM routine (Listing C.4). The imports from the `typing` module are used to annotate what the expected types are for each argument, and what the return type is. The `numpy` and `scipy` modules are ubiquitous in the listings, providing access to efficient routines for numerical computations.

```
1 from itertools import product
2 from typing import Any, Iterable, Optional, Tuple, Union
3 import numpy as np
4 import scipy as sp
```

C.1 Matrix pencil methods

C.1.1 MDL

CODE LISTING C.2: The MDL for estimation of the model order of a 1D FID. The first relative minimum in `mdl_vec` is determined to be the estimate of M , rather than the global minimum (Line 22), since the presence of very small singular values when k is large can lead to errors involving floating-point arithmetic.

```
1 def mdl(sigma: np.ndarray, N: int, L: int) -> int:
2     """Compute the Minimum Description Length
3
4     Parameters
5     -----
6     sigma
7         Vector of singular values associated with the data matrix.
8
```

```

9     N
10    Number of points the signal comprises.
11
12    L
13    Pencil parameter.
14 """
15
16    mdl_vec = np.zeros(L)
17    for k in range(L):
18        mdl_vec[k] = (
19            -N * np.sum(np.log(sigma[k:]))) +
20            N * (L - k) * np.log(np.sum(sigma[k:])) / (L - k)) +
21            k * np.log(N) * (2 * L - k) / 2
22
23    M = sp.signal.argelextrema(mdl_vec, np.less)[0][0]
24
25    return M

```

C.1.2 MPM

CODE LISTING C.3: The MPM for estimation of a 1D FID, with the option of estimating the model order using the MDL.

```

1 def mpm(Y: np.ndarray, sw: float, offset: float, M: int = 0) -> np.array:
2     """Matrix Pencil Method for estimating a 1D FID.
3
4     Parameters
5     -----
6     Y
7         FID.
8
9     sw
10        Sweep width (Hz).
11
12    offset
13        Transmitter offset (Hz).
14
15    M
16        Number of oscillators. If 0, this is estimated using the MDL.
17 """
18
19    N = Y #  $N^{(1)}$ 
20    L = N // 3 #  $L^{(1)}$ : pencil parameter
21    norm = np.linalg.norm(Y) #  $\|Y\|$ 
22    Y /= norm #  $Y/\|Y\|$ 
23    col = Y[:N - L] # First column of  $H_Y$ 
24    row = Y[N - L - 1 : ] # Last row of  $H_Y$ 

```

```

24     HY = sp.linalg.hankel(col, row) #  $\mathbf{H}_Y$ 
25     _, S, Vh = np.linalg.svd(HY) #  $\sigma$  and  $\mathbf{V}^\dagger$ 
26     V = Vh.T #  $\mathbf{V}$ 
27
28     if M == 0:
29         M = mdl(sigma, N, L)
30
31     VM = V[:, :M] #  $\mathbf{V}_M$ 
32     VM1 = Vm[:-1, :] #  $\mathbf{V}_{M1}$ 
33     VM2 = Vm[1:, :] #  $\mathbf{V}_{M2}$ 
34     VM1inv = np.linalg.pinv(VM1) #  $\mathbf{V}_{M1}^+$ 
35     VM1invVM2 = VM1inv @ VM2 #  $\mathbf{V}_{M1}^+ \mathbf{V}_{M2}$ 
36     z, _ = np.linalg.eig(VM1invVM2) #  $\mathbf{z}^{(1)}$ : signal poles
37     Z = np.power.outer(z, np.arange(N)).T #  $\mathbf{Z}^{(1)}$ 
38     alpha = np.linalg.pinv(Z) @ Y #  $\alpha$ : complex amplitudes
39
40     # Extract amplitude, phase, frequency and damping factor
41     amp = np.abs(alpha) * norm #  $\alpha$ 
42     phase = np.arctan2(np.imag(alpha), np.real(alpha)) #  $\phi$ 
43     freq = (sw / (2 * np.pi)) * np.imag(np.log(z)) + offset #  $f^{(1)}$ 
44     damp = -sw * np.real(np.log(z)) #  $\gamma^{(1)}$ 
45     theta = np.vstack((amp, phase, freq, damp)).T #  $\theta$ , as a  $M \times 4$  array
46
47     # Remove negative damping factors
48     neg_damp_idx = np.nonzero(damp < 0.0)[0]
49     theta = np.delete(theta, neg_damp_idx, axis=0)
50
51     theta = theta[np.argsort(theta[:, 2])] # order by  $f^{(1)}$ 
52     return theta

```

C.1.3 MMEMPM

CODE LISTING C.4: The MMEMPM for estimation of a 2D hypercomplex FID. Due to the very large size of the Hankel matrix \mathbf{E}_Y , a truncated SVD routine is employed, which determines only the first M components of the decomposition. This is only available to arrays stored in sparse form in SciPy (Lines 65–66).

```

1 def mmempm(
2     Y: np.ndarray, sw1: float, sw2: float,
3     offset1: float, offset2: float, M: int = 0,
4 ) -> np.ndarray:
5     """Modified Matrix Enhancement Matrix Pencil Method for estimating a
6     2D FID.
7

```

```

8     Parameters
9     -----
10    Y
11        FID.
12
13    sw1
14        Sweep width in first (indirect) dimension (Hz).
15
16    sw2
17        Sweep width in second (direct) dimension (Hz).
18
19    offset1
20        Transmitter offset in first dimension (Hz).
21
22    offset2
23        Transmitter offset in second dimension (Hz).
24
25    M
26        Number of oscillators. If 0, this is estimated by applying the MDL
27        to the first direct dimension FID (i.e. Y[0])
28    """
29    N1, N2 = Y.shape # N(1),N(2)
30    L1, L2 = N1 // 2, N2 // 2 # L(1),L(2): pencil parameters
31    norm = np.linalg.norm(Y) # ||Y||
32    Y /= norm # Y/||Y||
33
34    if M == 0:
35        # Extract first direct dimension signal, and estimate model order
36        # with the MDL
37        L_mdl = N2 // 3
38        Y_mdl = Y[0]
39        col_mdl = Y_mdl[: N2 - L_mdl]
40        row_mdl = Y_mdl[N2 - L_mdl - 1 :]
41        HY_mdl = sp.linalg.hankel(col_mdl, row_mdl)
42        _, sigma_mdl, _ = np.linalg.svd(HY_mdl)
43        M = mdl(sigma_mdl, N2, L_mdl)
44
45    # === Construct block Hankel EY ===
46    row_size = L2
47    col_size = N2 - L2 + 1
48    EY = np.zeros(
49        (L1 * L2, (N1 - L1 + 1) * (N2 - L2 + 1)),
50        dtype="complex",
51    )
52    for n1 in range(N1):

```

```

53      # Construct  $H_{Y,n^{(1)}}$ , and assign to appropriate positions in  $E_Y$ 
54      col = Y[n1, :L2]
55      row = Y[n1, L2 - 1:]
56      HYn1 = sp.linalg.hankel(col, row)
57      for n in range(n1 + 1):
58          r, c = n, n1 - n
59          if r < L1 and c < N1 - L1 + 1:
60              EY[
61                  r * row_size : (r + 1) * row_size,
62                  c * col_size : (c + 1) * col_size
63              ] = HYn1
64
65      EY = sp.sparse.csr_matrix(EY) # Make  $E_Y$  sparse
66      UM, *_ = sp.sparse.linalg.svds(EY, k=M) #  $U_M$ 
67
68      # === Construct permutation matrix  $P$  ===
69      P = np.zeros((L1 * L2, L1 * L2))
70      r = 0
71      for l2 in range(L2):
72          for l1 in range(L1):
73              c = l1 * L2 + l2
74              P[r, c] = 1
75              r += 1
76
77      UM1 = UM[: L1 * (L2 - 1)] # Last  $L^{(2)}$  rows deleted:  $U_{M1}$ 
78      UM2 = UM[L2:] # First  $L^{(2)}$  rows deleted:  $U_{M2}$ 
79      z1, W1 = np.linalg.eig(np.linalg.pinv(UM1) @ UM2) #  $z^{(1)}$ ,  $W^{(1)}$ 
80
81      UMP = P @ UM #  $U_{MP}$ 
82      UMP1 = UMP[:, (L1 - 1) * L2:] # Last  $L^{(1)}$  rows deleted:  $U_{MP1}$ 
83      UMP2 = UMP[L1:] # First  $L^{(1)}$  rows deleted:  $U_{MP2}$ 
84      Z2 = np.linalg.inv(W1) @ np.linalg.pinv(UMP1) @ UMP2 @ W1 #  $Z^{(2)}$ 
85      z2 = np.diag(Z2).copy() #  $z^{(2)}$ : copy needed as slice is readonly
86
87      # === Check for and deal with similar frequencies ===
88      freq1 = (0.5 * sw1 / np.pi) * np.imag(np.log(z1)) + offset1 #  $f^{(1)}$ 
89      threshold = sw1 / N1 #  $f_{sw}^{(1)}/N^{(1)}$ 
90      groupings = {}
91      # Iterate through values in  $f^{(1)}$  and group any with
92      # similar frequencies together
93      for idx, f1 in enumerate(freq1):
94          assigned = False
95          for group_f1, indices in groupings.items():
96              if np.abs(f1 - group_f1) < threshold:
97                  indices.append(idx)

```

```

98             n = len(indices)
99             indices = sorted(indices)
100            # Get new mean freq of the group
101            new_group_f1 = (n * group_f1 + f1) / (n + 1)
102            groupings[new_group_f1] = groupings.pop(group_f1)
103            assigned = True
104            break
105        if not assigned:
106            groupings[f1] = [idx]
107
108    for indices in groupings.values():
109        n = len(indices)
110        if n != 1:
111            A_slice = tuple(zip(*product(indices, repeat=2)))
112            A = Z2[A_slice].reshape(n, n)
113            new_group_z2, _ = np.linalg.eig(A)
114            z2[indices] = new_group_z2
115
116    # === Construct  $E_L$  and  $E_R$  ===
117    ZL2 = np.power.outer(z2, np.arange(L2)).T #  $Z_L^{(2)}$ 
118    ZR2 = np.power.outer(z2, np.arange(N2 - L2 + 1)) #  $Z_R^{(2)}$ 
119    Z1D = np.diag(z1) #  $Z_D^{(1)}$ 
120
121    EL = np.zeros((L1 * L2, M), dtype="complex")
122    Z2LZ1D = ZL2
123    for i in range(L1):
124        EL[i * row_size : (i + 1) * row_size] = Z2LZ1D
125        Z2LZ1D = Z2LZ1D @ Z1D
126    ER = np.zeros((M, (N1 - L1 + 1) * (N2 - L2 + 1)), dtype="complex")
127    Z1DZ2R = ZR2
128    for i in range(N1 - L1 + 1):
129        ER[:, i * col_size : (i + 1) * col_size] = Z1DZ2R
130        Z1DZ2R = Z1D @ Z1DZ2R
131    #  $\alpha = \text{diag}(E_L^+ E_Y E_R^+)$ 
132    alpha = np.diag(np.linalg.pinv(EL) @ EY @ np.linalg.pinv(ER))
133
134    amp = np.abs(alpha) * norm #  $\alpha$ 
135    phase = np.arctan2(np.imag(alpha), np.real(alpha)) #  $\phi$ 
136    freq2 = (0.5 * sw2 / np.pi) * np.imag(np.log(z2)) + offset2 #  $f^{(2)}$ 
137    damp1 = -sw1 * np.real(np.log(z1)) #  $\eta^{(1)}$ 
138    damp2 = -sw2 * np.real(np.log(z2)) #  $\eta^{(2)}$ 
139    #  $\theta$ , as a  $M \times 6$  array
140    theta = np.vstack((amp, phase, freq1, freq2, damp1, damp2)).T
141

```

```

142     # Remove negative damping factors
143     neg_damp_idx1 = list(np.nonzero(damp1 < 0. )[0])
144     neg_damp_idx2 = list(np.nonzero(damp2 < 0. )[0])
145     neg_damp_idx = list(set(neg_damp_idx1 + neg_damp_idx2))
146     theta = np.delete(theta, neg_damp_idx, axis=0)
147
148     theta = theta[np.argsort(theta[:, 3])] # order by  $f^{(2)}$ 
149     return theta

```

C.2 NLP

C.2.1 Trust Region Algorithm

CODE LISTING C.5: Steihaug-Toint trust region algorithm. Included is a check for oscillators with negative amplitudes, which causes the routine to terminate, in order for said oscillators to be purged (Lines 107–113).

```

1 def trust_steihaug_toint(
2     theta0: np.ndarray,
3     function_factory: FunctionFactory,
4     args: Iterable[Any] = (),
5 ) -> Tuple[np.ndarray, np.ndarray, bool]:
6     """Trust Region algorithm with Steihaug-Toint subroutine.
7
8     Parameters
9     -----
10    theta0
11        Initial guess, with shape (M, 4).
12
13    function_factory
14        Object for computing the objective, gradient and Hessian.
15
16    args
17        Extra arguments required for computing the objective and its
18        derivatives.
19
20    Returns
21    -----
22    theta
23        Parameter vector at termination.
24
25    errors
26        Errors associated with parameter vector.

```

```

27
28     negative_amps
29         Flag indicating whether or not termination occurred because
30             negative amplitudes were detected.
31     """
32
33     theta = theta0
34     M = theta.shape[0] // 4
35     factory = function_factory(theta, *args)
36
37     # === Define relevant parameters ===
38     # These have been hard-coded, though in NMR-EsPy they are all
39     # configurable.
40     eta = 0.15,
41     initial_trust_radius = 0.1 * factory.gradient_norm
42     max_trust_radius = 16 * initial_trust_radius
43     epsilon: float = 1.e-8,
44     max_iterations: int = 200,
45     check_neg_amps_every: int = 25,
46
47     k = 0
48     while True:
49         # === Steihaug-Toint ===
50         epsi = min(0.5, np.sqrt(factory.gradient_norm)) *
51             factory.gradient_norm
52         z = np.zeros_like(theta)
53         r = factory.gradient
54         d = -r
55         while True:
56             Bd = factory.hessian @ d
57             dBd = d.T @ Bd
58             if dBd <= 0:
59                 ta, tb = get_boundaries(z, d, trust_radius)
60                 pa = z + ta * d
61                 pb = z + tb * d
62                 p = min(factory.model(pa), factory.model(pb))
63                 hits_boundary = True
64                 break
65             r_sq = r.T @ r
66             alpha = r_sq / dBd
67             z_next = z + alpha * d
68             if sp.linalg.norm(z_next) >= trust_radius:
69                 _, tb = get_boundaries(z, d, trust_radius)
70                 p = z + tb * d
71                 hits_boundary = True
72             break

```

```

71         r_next = r + alpha * Bd
72         r_next_sq = r_next.T @ r_next
73         if np.sqrt(r_next_sq) < epsi:
74             hits_boundary = False
75             p = z_next
76             break
77         beta_next = r_next_sq / r_sq
78         d_next = -r_next + beta_next * d
79         z = z_next
80         r = r_next
81         d = d_next
82
83     # === Assess effectiveness of update ===
84     predicted_value = factory.model(p)
85     theta_proposed = theta + p
86     factory_proposed = function_factory(theta_proposed, *args)
87     actual_reduction = factory.objective - factory_proposed.objective
88     predicted_reduction = factory.objective - predicted_value
89     if predicted_reduction <= 0:
90         # No improvement could be found: terminate
91         negative_amps = False
92         break
93     rho = actual_reduction / predicted_reduction
94     if rho < 0.25:
95         # Quadratic model performing poorly: reduce TR
96         trust_radius *= 0.25
97     elif rho > 0.75 and hits_boundary:
98         # Quadratic model performing well: increase TR
99         trust_radius = min(2 * trust_radius, max_trust_radius)
100    if rho > eta:
101        # Accept update: new iteration
102        theta = theta_proposed
103        factory = factory_proposed
104        k += 1
105
106    # === Check for termination criteria ===
107    if (k % check_neg_amps_every == 0):
108        neg_amps = np.where(theta[amp_slice] <= 0)[0]
109        print(neg_amps)
110        if neg_amps.size > 0:
111            # Negative amps found: this run in order to purge
112            negative_amps = True
113            break
114
115    if factory.gradient_norm < epsilon:

```

```

116         # Convergence
117         negative_amps = False
118         break
119
120     if k == max_iterations:
121         # Maximum allowed iterations reached
122         negative_amps = False
123         break
124
125     # Routine terminated: compute errors and return parameter array
126     errors = np.sqrt(
127         factory.objective *
128         np.abs(np.diag(np.linalg.inv(factory.hessian))))
129     )
130
131     return theta, errors, negative_amps
132
133 def get_boundaries(
134     z: np.ndarray, d: np.ndarray, trust_radius: float
135 ) -> Tuple[float, float]:
136     """Determine the intersections of the search direction and the trust
137     region."""
138     a = d.T @ d
139     b = 2 * z.T @ d
140     c = (z.T @ z) - (trust_radius ** 2)
141     aux = b + np.copysign(
142         np.sqrt(b * b - 4 * a * c),
143         b,
144     )
145
146     return sorted([-aux / (2 * a), -(2 * c) / aux])

```

C.2.2 Computing \mathcal{F}_ϕ , $\nabla \mathcal{F}_\phi$, and $\nabla^2 \mathcal{F}_\phi$

CODE LISTING C.6: Code for the generation of the fidelity for NLP applied to 1D estimation, as well as the gradient and (approximated) Hessian. The FunctionFactory object accepts a parameter set (theta) and function (fun), which computes the objective, gradient and Hessian. The first time a quantity is requested from the factory, the function is run, and the objective and its derivatives are cached (memoised), such that the next time a quantity is requested, the cached result is used, rather than the expensive function being re-computed. FunctionFactoryGaussNewton1D (Lines 41–43) inherits from the base class, for specific use in 1D estimation, with an approximated Hessian.

```

1 class FunctionFactory:
2     """Object which computes and memoises the objective, gradient and
3     hessian for a given set of parameters."""

```

```
4     def __init__(self, theta: np.ndarray, fun: callable, *args) -> None:
5         self.theta = theta
6         self._new_theta = True
7         self.fun = fun
8         self.obj = None
9         self.grad = None
10        self.hess = None
11        self.args = args
12
13    def _compute_if_needed(self):
14        """Compute the obj, grad and Hess if they haven't been yet"""
15        if self.obj is None:
16            self.obj, self.grad, self.hess = self.fun(self.theta,
17                *self.args)
18
19    def model(self, p) -> float:
20        return self.objective + self.gradient @ p + 0.5 * (p.T @
21            self.hessian @ p)
22
23    @property
24    def objective(self) -> float:
25        self._compute_if_needed()
26        return self.obj
27
28    @property
29    def gradient(self) -> np.ndarray:
30        self._compute_if_needed()
31        return self.grad
32
33    @property
34    def gradient_norm(self) -> float:
35        return sp.linalg.norm(self.gradient)
36
37    @property
38    def hessian(self) -> np.ndarray:
39        self._compute_if_needed()
40        return self.hess
41
42    class FunctionFactoryGaussNewton1D(FunctionFactory):
43        def __init__(self, theta: np.ndarray, *args) -> None:
44            super().__init__(theta, obj_grad_gauss_newton_hess_1d, *args)
45
46    def obj_grad_gauss_newton_hess_1d(
```

```

47     theta: np.ndarray, *args: Tuple[int, int, np.ndarray],
48 ) -> Tuple[float, np.ndarray, np.ndarray]:
49     Y, tp = args # Unpack args: FID, timepoints
50     N = Y.shape[0]
51     M = theta.shape[0] // 4
52     X_per_osc = np.exp(
53         np.outer(tp, (2j * np.pi * theta[2 * M : 3 * M] - theta[3 * M :]))
54     ) * (theta[:M] * np.exp(1j * theta[M : 2 * M]))
55
56     # Jacobian: all first partial derivatives,  $\partial X / \partial \theta$ 
57     jac = np.zeros((N, 4 * M), dtype="complex")
58     jac[:, :M] = X_per_osc / theta[:M] #  $\partial X / \partial \alpha$ 
59     jac[:, M : 2 * M] = 1j * X_per_osc #  $\partial X / \partial \phi$ 
60     jac[:, 2 * M : 3 * M] = \
61         np.einsum("ij,i->ij", X_per_osc, 2j * np.pi * tp) #  $\partial X / \partial f^{(1)}$ 
62     jac[:, 3 * M :] = np.einsum("ij,i->ij", X_per_osc, -tp) #  $\partial X / \partial \eta^{(1)}$ 
63
64     X = np.einsum("ij->i", X_per_osc)
65     Y_minus_X = Y - X
66
67     obj = (Y_minus_X.conj().T @ Y_minus_X).real #  $\mathcal{F}(\theta)$ 
68     grad = -2 * (Y_minus_X.conj().T @ jac).real #  $\nabla \mathcal{F}(\theta)$ 
69     hess = 2 * (jac.conj().T @ jac).real #  $\nabla^2 \mathcal{F}(\theta)$ 
70
71     # === Determine phase variance and derivatives ===
72     phi = theta[M : 2 * M]
73     cos_phi = np.cos(phi)
74     cos_sum = np.sum(cos_phi)
75     sin_phi = np.sin(phi)
76     sin_sum = np.sum(sin_phi)
77     R = np.sqrt(cos_sum ** 2 + sin_sum ** 2)
78     pv_obj = 1 - (R / M)
79     pv_grad = (sin_phi * cos_sum - cos_phi * sin_sum) / (M * R)
80     x = (sin_phi * cos_sum) - (cos_phi * sin_sum)
81     term_1 = np.outer(x, x) / (R ** 2)
82     phi_array = np.zeros((M, M))
83     phi_array[:] = phi
84     term_2 = -np.cos(phi_array.T - phi_array)
85     pv_hess = term_1 + term_2
86     pv_hess[np.diag_indices(M)] += cos_phi * cos_sum + sin_phi * sin_sum
87     pv_hess /= M * R
88     obj += pv_obj #  $\mathcal{F}_\phi(\theta)$ 
89     grad[M : 2 * M] += pv_grad #  $\nabla \mathcal{F}_\phi(\theta)$ 
90     hess[M : 2 * M, M : 2 * M] += pv_hess #  $\nabla^2 \mathcal{F}_\phi(\theta)$ 
91

```

```
92     return obj, grad, hess
```

C.2.3 The main routine

CODE LISTING C.7: Code for running the NLP routine for FID estimation. The routine consists of running the ST algorithm (Listing C.5) until it returns a parameter array without negative amplitudes. If negative amplitudes are present, the corresponding oscillators are removed, and the ST algorithm is re-run.

```

1 def nlp(
2     Y: np.ndarray,
3     sw: float,
4     offset: float,
5     theta0: np.ndarray,
6 ) -> Tuple[np.ndarray, np.ndarray]:
7     """Nonlinear programming routine for 1D FID estimation.
8
9     Parameters
10    -----
11    Y
12        FID.
13
14    sw
15        Sweep width (Hz).
16
17    offset
18        Transmitter offset (Hz).
19
20    theta0
21        Initial guess, of shape (M, 4)
22
23    Returns
24    -----
25    theta
26        Parameter estimate.
27
28    errors
29        Errors associated with theta.
30
31    """
32    norm = np.linalg.norm(Y)
33    Y /= norm
34    N = Y.shape[0]
35    M = theta0.shape[0]
36    # Flatten parameter array: Fortran (column-wise) ordering

```

```

36     theta0_vec = theta0.flatten(order="F")
37     theta0_vec[:M] /= norm # Normalise amplitudes
38     # Remove transmitter offset from frequencies
39     theta0_vec[2 * M : 3 * M] -= offset
40
41     # Extra arguments needed to compute the objective, grad, and Hessian:
42     # FID and timepoints sampled
43     opt_args = [Y, np.linspace(0, float(N - 1) / sw, N)]
44
45     while True:
46         theta_vec, errors_vec, negative_amps = trust_ncg(
47             theta0=theta0_vec,
48             function_factory=FunctionFactoryGaussNewton1D,
49             args=opt_args,
50         )
51
52         if negative_amps:
53             # Negative amps exist: remove these
54             negative_idx = list(np.where(theta_vec[:M] <= 0.0)[0])
55             slice_ = []
56             for idx in range(negative_idx):
57                 slice_.extend([i * M + idx for i in range(4)])
58             theta_vec = np.delete(theta_vec, slice_)
59             M -= len(negative_idx) # New model order
60
61         else:
62             # No negative amps: routine complete
63             break
64
65         # Reshape parameter array back to (M, 4)
66         theta = theta_vec.reshape((M, 4), order="F")
67         errors = errors_vec.reshape((M, 4), order="F")
68         errors_vec /= N - 1
69         theta[:, 2] += offset # Re-add transmitter offset to frequencies
70         theta[:, 0] *= norm # Re-scale amplitudes
71         errors[:, 0] *= norm
72         theta[:, 1] = (theta[:, 1] + np.pi) % (2 * np.pi) - np.pi # Wrap
73             # phases
74
75     return theta, errors

```

C.3 CUPID

C.3.1 Assigning multiplet structures

CODE LISTING C.8: Code for performing multiplet assignment as part of CUPID.

```

1 def predict_multiplets(
2     params: np.ndarray,
3     thold: float,
4 ) -> Dict[float, Iterable[int]]:
5     """
6         Parameters
7         -----
8         params
9             Estimated parameter array with shape (M, 6) such that each row
10            provides the parameters of a particular oscillator, in the order
11            [a, ϕ, f1, f2, η1, η2].
12
13         thold
14             Frequency threshold  $\varepsilon > 0$ .
15
16         Returns
17         -----
18         A dictionary with the multiplet central frequencies as keys and
19         oscillator indices as values
20         """
21         multiplets = {}
22         for m, osc in enumerate(params):
23             assigned = False
24             f1, f2 = osc[2], osc[3] # Extract  $f^{(1)}$  and  $f^{(2)}$ 
25             fc = osc[3] - osc[2] # Central frequency for oscillator:  $f_m^{(2)} - f_m^{(1)}$ 
26             # Check whether central frequency agrees with any
27             # already-established multiplet grouping
28             for fmp in multiplets:
29                 if fmp - thold < fc < fmp + thold:
30                     # Update grouping:
31                     # Add m to list of oscillators
32                     # Update central frequency: mean of all oscillators
33                     ms = multiplets.pop(fmp)
34                     ms.append(m)
35                     mp_size = len(ms)
36                     new_fmp = ((mp_size - 1) * fmp + fc) / mp_size
37                     multiplets[new_fmp] = ms
38                     assigned = True

```

```
39          break
40      # No match: create new multiplet group
41      if not assigned:
42          multiplets[fc] = [m]
43
44  return multiplets
```

INFORMATION ON DATASETS AND RESULTS

D

D.1 Simulated datasets

D.1.1 SPINACH Simulations

Many of the simulated datasets presented in this work were generated using the SPINACH MATLAB package[139]. In each case, the dataset was generated via a call to the `new_spinach` method associated with the relevant `Estimator` object in NMR-EsPy. `new_spinach` works by instantiating a MATLAB engine from PYTHON[140], which then runs a SPINACH simulation of the relevant experiment with the specifications provided. The FID generated is then stored within in a new `Estimator` object along with other necessary information about the simulation.

Spin Systems

Table D.1 provides a specification of the chemical shifts and scalar couplings that made up the spin systems used to construct simulated data with SPINACH. Below is a description of how the spin systems were generated.

“Five Multiplets” The spin systems used to generate the “five multiplets” inversion recovery datasets featured 8 spins, comprising a group of 5 “estimated spins” and a group of 3 “coupling spins”:

- The estimated spins were each assigned chemical shifts randomly sampled from $\mathcal{U}(-0.1 \text{ ppm}, 0.1 \text{ ppm})$.
- The coupling spins were assigned chemical shifts $\gg 0.15 \text{ ppm}$. Each coupling spin was assigned a non-zero J-coupling to each of the estimated spins, with the couplings randomly sampled from $\mathcal{U}(-20 \text{ Hz}, 20 \text{ Hz})$.

As a result, a region of the dataset, centred at 0 ppm, would feature 5 ddd multiplets, which abide by the weak coupling approximation. A further constraint was applied, to ensure that the $40 (5 \times$

2^3) signals generated by the 5 estimated spins would have sufficiently separated frequencies, such that they could feasibly be resolved by estimation. Each of the spins were assigned longitudinal and transverse relaxation times, using the distributions $T_1 \sim \mathcal{U}(1\text{ s}, 5\text{ s})$, and $T_2 \sim \mathcal{U}(0.2\text{ s}, 0.6\text{ s})$.

“Four Multiplets” The “four multiplets” 2DJ datasets were generated using spin systems with a similar form to the “five multiplets” spin systems. Each spin system comprised 7 spins, with 4 estimated and 3 coupling spins. The estimated spins were assigned chemical shifts randomly sampled from $\mathcal{U}(-0.03\text{ ppm}, 0.03\text{ ppm})$, while the coupling spins were given shifts which were $\gg 0.03\text{ ppm}$. J-couplings between the estimated and coupling spins were sampled from $\mathcal{U}(-10\text{ Hz}, 10\text{ Hz})$.

Strychnine The strychnine spin system was derived from the SPINACH function <SPINACHROOT> /etc/strychnine.m, which returns a spin system specification using chemical shifts and scalar couplings from [141: Appendix 5].

TABLE D.I: The isotropic chemical shifts (δ), corresponding rotating frame frequencies (ω_0), and scalar couplings (J) associated with spin systems used in SPINACH simulations. For the “Five multiplets” spin systems, the associated T_1 times are provided too.

Spin	δ (ppm)	ω_0 (Hz)	J (Hz)	T_1 (s)
Five Multiplets, Run 1				
A	-6.80×10^{-2}	-34.01	F: 10.965, G: 12.657, H: 17.070	2.178
B	7.09×10^{-2}	35.44	F: 3.610, G: 2.543, H: 8.448	4.430
C	-9.23×10^{-2}	-46.13	F: 10.630, G: 6.282, H: 3.012	3.319
D	1.00×10^{-2}	5.01	F: 8.101, G: 4.589, H: 9.068	1.007
E	-9.95×10^{-2}	-49.77	F: 3.014, G: 16.537, H: 15.587	4.992
Five Multiplets, Run 2				
A	6.61×10^{-2}	33.07	F: 19.488, G: 18.279, H: 3.147	3.600
B	-3.75×10^{-2}	-18.75	F: 11.924, G: 8.400, H: 5.515	3.905
C	-9.68×10^{-2}	-48.39	F: 13.672, G: 6.543, H: 16.275	4.291
D	2.70×10^{-2}	13.48	F: 12.007, G: 5.141, H: 9.981	1.687
E	8.91×10^{-2}	44.53	F: 8.715, G: 14.309, H: 9.805	3.214
Five Multiplets, Run 3				
A	-8.23×10^{-2}	-41.13	F: 4.984, G: 18.119, H: 10.642	3.846
B	3.90×10^{-2}	19.50	F: 13.518, G: 14.381, H: 3.074	1.018
C	-3.33×10^{-2}	-16.66	F: 8.758, G: 16.689, H: 12.956	4.766

Continues on next page...

Spin	δ (ppm)	ω_0 (Hz)	J (Hz)	T_1 (s)
D	-4.60×10^{-2}	-23.02	F: 17.648, G: 7.514, H: 3.918	3.414
E	9.35×10^{-2}	46.76	F: 16.396, G: 7.455, H: 11.352	1.827
Four Multiplets, Run 1				
A	-2.78×10^{-2}	-13.93	E: -9.627, F: -8.202, G: 6.742	-
B	-7.11×10^{-3}	-3.56	E: -4.491, F: 5.333, G: 9.303	-
C	-1.63×10^{-3}	-0.81	E: 3.953, F: 5.422, G: 5.914	-
D	1.53×10^{-2}	7.66	E: -7.902, F: -4.556, G: 6.217	-
Four Multiplets, Run 2				
A	-1.48×10^{-2}	-7.38	E: -7.492, F: 0.917, G: 2.933	-
B	-1.18×10^{-2}	-5.88	E: -4.304, F: -1.815, G: 5.420	-
C	-4.32×10^{-3}	-2.16	E: 4.832, F: 7.573, G: 8.268	-
D	1.76×10^{-2}	8.80	E: -9.244, F: -1.816, G: -0.478	-
Four Multiplets, Run 3				
A	-2.23×10^{-2}	-11.17	E: -5.347, F: -1.851, G: 1.407	-
B	1.13×10^{-2}	5.66	E: 6.425, F: 7.291, G: 9.806	-
C	2.53×10^{-2}	12.64	E: -8.640, F: 0.613, G: 6.998	-
D	2.84×10^{-2}	14.21	E: -8.613, F: 0.782, G: 3.830	-
Four Multiplets, Run 4				
A	-2.03×10^{-2}	-10.16	E: -8.646, F: 6.719, G: 7.921	-
B	3.53×10^{-3}	1.77	E: -8.857, F: 4.314, G: 9.197	-
C	8.61×10^{-3}	4.30	E: -0.620, F: 1.767, G: 6.567	-
D	2.06×10^{-2}	10.30	E: -9.060, F: 2.355, G: 9.810	-
Four Multiplets, Run 5				
A	-9.16×10^{-3}	-4.58	E: -8.281, F: 1.621, G: 3.229	-
B	-2.79×10^{-3}	-1.40	E: 1.655, F: 4.219, G: 6.998	-
C	3.00×10^{-3}	1.50	E: -4.280, F: 1.045, G: 5.896	-
D	2.74×10^{-2}	13.72	E: -9.316, F: -8.322, G: -3.938	-
Strychnine				
A	7.167	3583.5	B: 7.490, C: 1.080, D: 0.230	-

Continues on next page...

Spin	δ (ppm)	ω_0 (Hz)	J (Hz)	T_1 (s)
B	7.098	3549.0	A : 7.490, C : 7.440, D : 0.980	–
C	7.255	3627.5	A : 1.080, B : 7.440, D : 7.900	–
D	8.092	4046.0	A : 0.230, B : 0.980, C : 7.900	–
E	3.860	1930.0	I : 10.410	–
F	3.132	1566.0	G : -17.340, H : 3.340	–
G	2.670	1335.0	F : -17.340, H : 8.470	–
H	4.288	2144.0	F : 3.340, G : 8.470, I : 3.300	–
I	1.276	638.0	E : 10.410, H : 3.300, J : 3.290	–
J	3.150	1575.0	I : 3.290, K : 4.110, L : 1.960, R : 1.610, T : 0.470	–
K	2.360	1180.0	J : 4.110, L : -14.350, M : 4.330	–
L	1.462	731.0	J : 1.960, K : -14.350, M : 2.420	–
M	3.963	1981.5	K : 4.330, L : 2.420	–
N	1.890	945.0	O : -13.900, P : 5.500, Q : 7.200	–
O	1.890	945.0	N : -13.900, P : 3.200, Q : 10.700	–
P	3.219	1609.5	N : 5.500, O : 3.200, Q : -13.900	–
Q	2.878	1439.0	N : 7.200, O : 10.700, P : -13.900	–
R	3.716	1858.0	J : 1.610, S : -14.800, T : 1.790	–
S	2.745	1372.5	R : -14.800	–
T	5.915	2957.5	J : 0.470, R : 1.790, U : 7.000, V : 6.100	–
U	4.148	2074.0	T : 7.000, V : -13.800	–
V	4.066	2033.0	T : 6.100, U : -13.800	–

D.1.2 2DJ

The simulated datasets for the “Four multiplets” and Strychnine results in Sections 4.3.1 and 4.3.2 were generated using NMR-EsPy’s `Estimator2DJ.new_spinach` method, with parameters used in the simulation provided by Table D.2.

A 2DJ pulse sequence simulation does not ship with SPINACH. Therefore, an in-house one was written, which is given by Listing D.1.

Parameter	Four Multiplets	Strychnine
$f_{\text{bf}}^{(1)}$ (MHz)	500	499.9
$f_{\text{off}}^{(2)}$ (Hz)	0	2500
$f_{\text{off}}^{(2)}$ (ppm)	0	5.001
$f_{\text{sw}}^{(1)}$ (Hz)	40	50
$f_{\text{sw}}^{(2)}$ (Hz)	1000	5000
$f_{\text{sw}}^{(2)}$ (ppm)	2	10
$N^{(1)}$	128	128
$N^{(2)}$	1024	8192

TABLE D.2: Experiment parameters for 2DJ simulations run using SPINACH.**CODE LISTING D.1:** Function for simulating 2DJ experiments using SPINACH.

```

1 function fid = jres_seq(spin_system, parameters, H, R, K)
2     % Compose Liouvillian
3     L = H + 1i * R + 1i * K; clear('H', 'R', 'K');
4     % Coherent evolution timestep
5     d1 = 1 / parameters.sweep(1);
6     d2 = 1 / parameters.sweep(2);
7     % Number of points
8     pts1 = parameters.npoints(1);
9     pts2 = parameters.npoints(2);
10    % Targeted nucleus
11    nuc = parameters.spins{1}
12    % Initial state
13    rho = state(spin_system, 'Lz', nuc);
14    % Detection state
15    coil = state(spin_system, 'L+', nuc);
16    % Get the pulse operators
17    Lp = operator(spin_system, 'L+', nuc);
18    Lx = (Lp + Lp') / 2;
19    % First pulse: 90 about x
20    rho = step(spin_system, Lx, rho, pi / 2);
21    % First half of t1 evolution
22    rho = evolution(spin_system, L, [], rho, d1 / 2, pts1 - 1,
23    ↵ 'trajectory');
24    % Select "-1" coherence
25    rho = coherence(spin_system, rho, {{nuc, -1}});
26    % Second pulse: 180 about x
27    rho = step(spin_system, Lx, rho, pi);
28    % Select "+1" coherence
29    rho = coherence(spin_system, rho, {{nuc, +1}});
30    % Second half of t1 evolution

```

```

30     rho = evolution(spin_system, L, [], rho, d1 / 2, pts1 - 1, 'refocus');
31     % Run the F2 evolution
32     fid = evolution(spin_system, L, coil, rho, d2, pts2 - 1,
33     ↵ 'observable');
33 end

```

D.1.3 Inversion recovery

The simulated datasets for the “Five multiplets” results in Section 3.2.4 was generated using NMR-EsPy’s `EsimatorInvRec.new_spinach` method, with parameters used in the simulation provided by Table D.3. To model relaxation, SPINACH’s T_1/T_2 approximation was used[142]. Under this model, all longitudinal states associated with a given spin relax at a rate $R_1 := 1/T_1$, while transverse states relax at a rate $R_2 := 1/T_2$. For multi-spin order states, the relaxation rate is given by the sum of the appropriate rates. The dataset was produced using SPINACH’s built-in inversion recovery pulse sequence simulation, found at <SPINACHROOT>/experiments/inv_rec.m. 21 evenly-spaced increments of τ were used; the first increment was 0 s, the last was 4 s, and consecutive increments differed by 0.4 s.

Parameter	Five Multiplets
$f_{bf}^{(1)}$ (MHz)	500
$f_{off}^{(1)}$ (Hz)	2500
$f_{off}^{(1)}$ (ppm)	5
$f_{sw}^{(1)}$ (Hz)	5000
$f_{sw}^{(1)}$ (ppm)	10
$N^{(1)}$	16384

TABLE D.3: Parameters for the “five multiplets” inversion recovery simulation run using SPINACH.

D.2 Experimental datasets

D.2.1 Structures

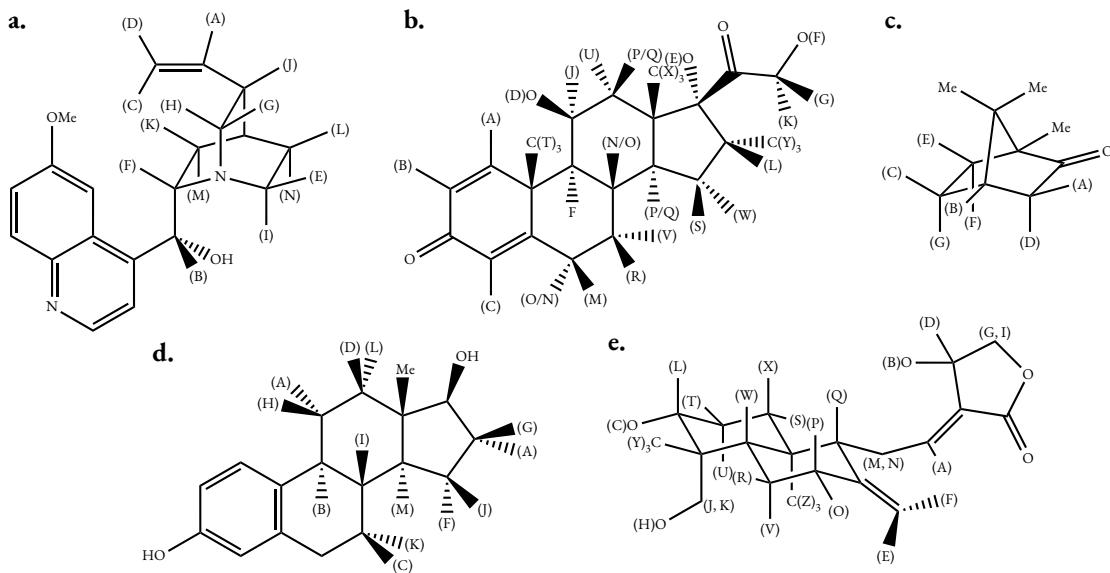


FIGURE D.1: The molecular structures of species giving rise to the experimental NMR datasets considered in this work. **a.** Quinine, **b.** Dexamethasone, **c.** Camphor, **d.** 17β -estradiol, **e.** Andrographolide. Proton environments giving rise to signals which are considered in this work are denoted with bracketed alphabetical characters. Non-bracketed alphabetical characters denote chemical symbols. Me denotes the methyl group, equivalent to CH_3 . **TODO: check assignments (esp. estradiol). If more structures need adding, edit the ChemDraw file on Chive called simon_stuff/thesis_structures.cdxml. Use EB-Garamond for atom labels. One a structure is made, scale to 75% of original size, and set font to 7pt. Then in inkscape, rescale this by multiplying by 0.8. Dexamethasone: replace (J) with (H), (K) with (I), etc...**

D.2.2 1D datasets

Both the andrographolide and cyclosporin A pulse-acquire experiments were acquired BRUKER's zg30 pulse sequence, which involves applying a pulse with a target flip angle of 30° , followed by acquisition. The cyclosporin A pulse-acquire dataset was taken from BRUKER's TOPSPIN software (v 4.0.8), located at <TOPSPINROOT>/topspin4.0.8/examdata/exam1d_1H/1.

	Andrographolide	Cyclosporin A
f_{bf} (MHz)	600.18	500.13
f_{off} (Hz)	2400.7	2249.2
f_{off} (ppm)	4	4.4972
f_{sw} (Hz)	4795.4	5494.5
f_{sw} (ppm)	7.9899	10.986
N	16384	65384
NS	1	16
DS	0	2
PLW1 (W)	24	?
P1 (μs)	12	10.8
D1 (s)	1	1

TABLE D.4: Noteworthy experiment parameters for the pulse-acquire datasets used. NS: Number of scans, DS: Number of dummy scans, PLW1: Hard pulse power (W), P1: Duration of $\pi/2$ pulse, D1: Duration of relaxation delay. N.B. The duration of the pulse used was $1/3$ that of P1.

D.2.3 Diffusion datasets

The andrographolide diffusion dataset (Figure 3.6) was acquired using the one-shot DOSY pulse sequence[98] (version 1.0c, published on 27/3/2012), accessible via the webpage <https://www.nmr.chemistry.manchester.ac.uk/?q=node/264>. The pulse sequence is displayed in Figure D.2.

	Andrographolide	Glucose/valine/threonine
f_{bf} (MHz)	600.18	499.98
f_{off} (Hz)	3000.9	2499.9
f_{off} (ppm)	5	5
f_{sw} (Hz)	7211.5	10000
f_{sw} (ppm)	12.016	20.001
N	16384	65536
NS	4	32
DS	2	4
PLW1 (W)	24	18.204
P1 (μs)	12	10
D1 (s)	1.5	6

TABLE D.5: Noteworthy experiment parameters for the diffusion datasets acquired. NS: Number of scans, DS: Number of dummy scans, PLW1: Hard pulse power (W), P1: Duration of $\pi/2$ pulse, D1: Duration of relaxation delay.

The glucose/valine/threonine diffusion dataset (Figure 3.7) was acquired using Bruker's ledbpqp2s pulse sequence (version 1.9, published on 19/2/2011). This is a stimulated echo pulse sequence, featuring bipolar gradients and a longitudinal eddy current delay (LED) component[97]. The pulse sequence is displayed in Figure D.3.

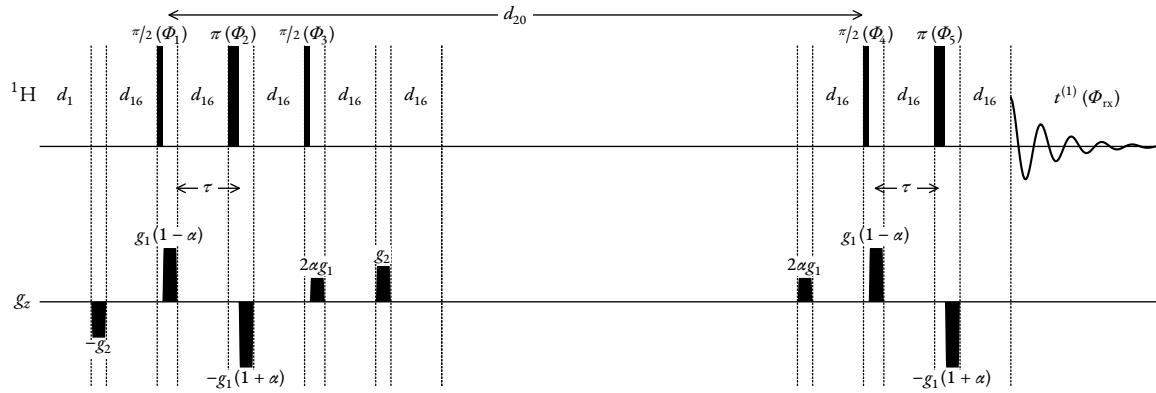


FIGURE D.2: Oneshot DOSY pulse sequence used for the acquisition of andrographolide data (Figure 3.6). All delays are included, though they are not to scale. Delays: d_1 (relaxation delay): 1.5 s, d_{16} (gradient recovery delay): 200 μ s, d_{20} (diffusion time, equivalent to Δ): 0.1 s. Hard pulses had a power of 24 W, with the duration of the $\pi/2$ pulse being 12 μ s. Diffusion encoding gradients had a smoothed square profile (SMSQ10.100), a duration of 1 ms, and had strengths related to the values g_1 and $\alpha = 0.2$. g_1 was varied across increments, with the values used being (G cm^{-1}): 6.270, 12.470, 16.483, 19.695, 22.451, 24.905, 27.137, 29.200, 31.126, 32.939, 34.658, 36.296, 37.862, 39.367, 40.816, 42.215, 43.570, 44.883, 46.159, 47.401. Spoiler gradients had the same SMSQ10.100 profile, had a duration of 600 μ s, and a strength which was 75% of g_1 . The phase cycling scheme used was: $\Phi_1 : 2 \times (0^\circ, 180^\circ)$; $\Phi_2 : 4 \times 0^\circ$; $\Phi_3 : 4 \times 0^\circ$; $\Phi_4 : 2 \times 0^\circ, 2 \times 180^\circ$; $\Phi_5 : 4 \times 0^\circ$; $\Phi_{\text{rx}} : 0^\circ, 180^\circ, 180^\circ, 0^\circ$.

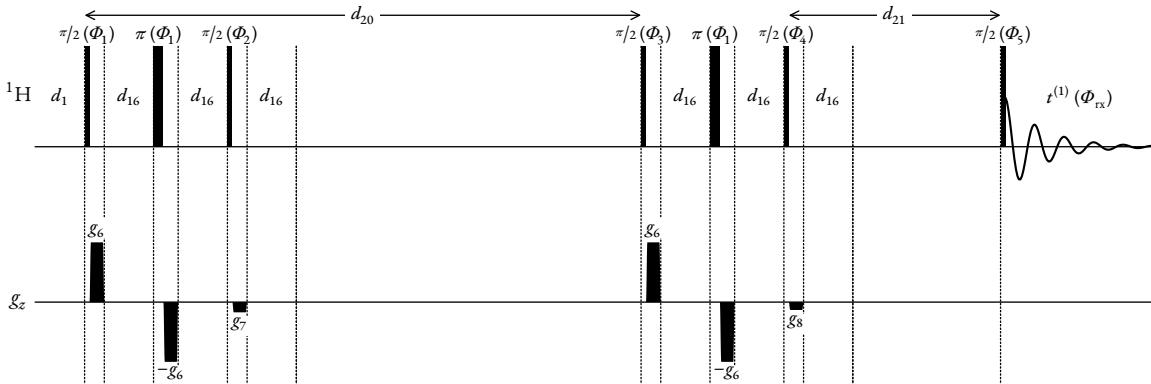


FIGURE D.3: Pulse sequence used for the acquisition of glucose/threonine/valine diffusion data (Figure 3.7). All delays are included, though they are not to scale. Delays: d_1 (relaxation delay): 6 s, d_{16} (gradient recovery delay): 200 μ s, d_{20} (diffusion time, equivalent to Δ): 0.1 s, d_{21} (eddy-current delay): 5 ms. Hard pulses had a power of 18.204 W, with the duration of the $\pi/2$ pulse being 10 μ s. All gradients had a smoothed square profile (SMSQ10.100). Gradients for diffusion encoding had a duration of 1 ms, with strength g_6 varied across increments, with the values used being (G cm^{-1}): 4.500, 12.728, 17.428, 21.107, 24.233, 27.000, 29.508, 31.820, 33.974, 36.000. Spoiler gradients had a duration of 600 μ s. The gradients had the relative strengths $g_7 = -0.1713g_6$ and $g_8 = -0.1317g_6$. The phase cycling scheme used was: $\Phi_1 : 32 \times 0^\circ$; $\Phi_2 : 8 \times (2 \times 0^\circ, 2 \times 180^\circ)$; $\Phi_3 : 2 \times (4 \times 0^\circ, 4 \times 180^\circ, 4 \times 90^\circ, 4 \times 270^\circ)$; $\Phi_4 : 2 \times (2 \times (0^\circ, 180^\circ), 2 \times (180^\circ, 0^\circ), 2 \times (90^\circ, 270^\circ), 2 \times (270^\circ, 90^\circ))$; $\Phi_5 : 2 \times (4 \times 0^\circ, 4 \times 180^\circ, 4 \times 90^\circ, 4 \times 270^\circ)$; $\Phi_{\text{rx}} : 2 \times (0^\circ, 180^\circ, 180^\circ, 0^\circ, 180^\circ, 270^\circ, 90^\circ, 90^\circ, 270^\circ, 270^\circ, 90^\circ)$.

D.2.4 2DJ datasets

The 2D J-Resolved datasets presented were acquired using Bruker's jresqf pulse sequence (version 1.7, released 31/1/2012). This pulse sequence comprises $\pi/2(\Phi_1) \rightarrow t^{(1)}/2 \rightarrow \pi(\Phi_2) \rightarrow t^{(1)}/2 \rightarrow t^{(2)}(\Phi_{\text{rx}})$, with the EXORCYCLE phase-cycling scheme[26: Section 11.6]:

$$\begin{aligned}\Phi_1 &: 0^\circ \quad 0^\circ \quad 0^\circ \quad 0^\circ \\ \Phi_2 &: 0^\circ \quad 90^\circ \quad 180^\circ \quad 270^\circ \\ \Phi_{\text{rx}} &: 0^\circ \quad 180^\circ \quad 0^\circ \quad 180^\circ\end{aligned}$$

Key experiment parameters are provided in Table D.6.

	Quinine	Dexamethasone	Camphor	Estradiol
f_{bf} (MHz)	500.13	600.18	500.13	500.3
$f_{\text{off}}^{(2)}$ (Hz)	2500	2815.4	1000	2501.5
$f_{\text{sw}}^{(1)}$ (Hz)	50	50	50	100
$f_{\text{sw}}^{(2)}$ (Hz)	7500	7211.5	5000	5000
$f_{\text{sw}}^{(2)}$ (ppm)	14.996	12.016	9.9974	9.994
$N^{(1)}$	128	64	128	128
$N^{(2)}$	16384	8192	16384	16384
NS	4	2	4	4
DS	4	8	4	2
PLW1 (W)	20.893	24	20.893	31.537
P1 (μs)	10	12	10	15
D1 (s)	2	1.5	2	1

TABLE D.6: Noteworthy experiment parameters for the 2D J-Reolved and PSYCHE experiments run. NS: Number of scans, DS: Number of dummy scans, PLW1: Hard pulse power (W), P1: Duration of $\pi/2$ pulse, D1: Duration of relaxation delay.

D.2.5 PSYCHE datasets

The pulse sequence used for the acquisition of the estradiol PSYCHE spectrum (Figure 4.9) is presented in Figure D.4, where pulses, gradients, and delays are described in detail. Equivalent parameters were used for the basic setup as the estradiol 2DJ experiment, given in Table D.6.

The pure shift spectrum of dexamethasone (Figure 4.8.a) was acquired using a TSE-PSYCHE experiment (Figure D.5). The pulse sequence file UoM_1d_if_tsepsyche_ts4x can be obtained via the link <https://research.manchester.ac.uk/en/datasets/manchester-pure-shift-nmr-workshop-bruker-software>.

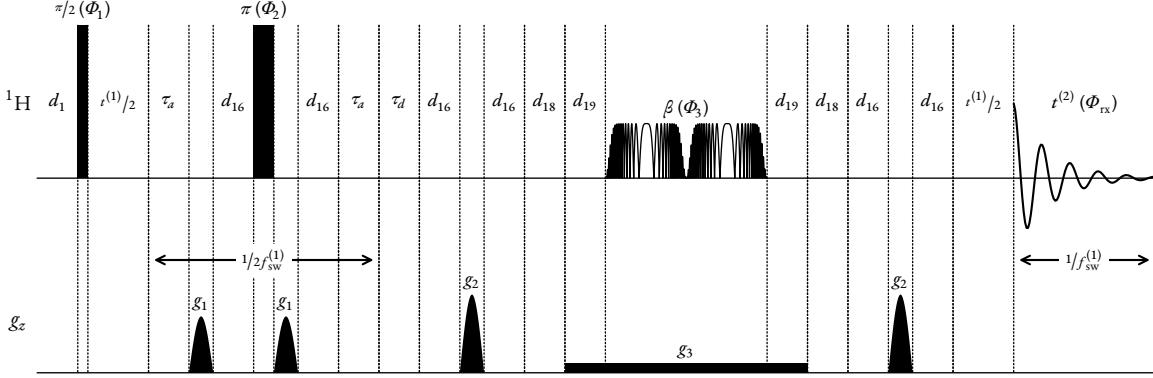


FIGURE D.4: PSYCHE pulse sequence used for the acquisition of estradiol data. All delays are included, though they are not to scale. Delays: d_1 (relaxation delay): 1 s, d_{16} : 200 μs , d_{18} : 200 μs , d_{19} : 1 ms, τ_a : 1.3 ms, τ_d : 18.9 ms. The PSYCHE element featured two saltire chirp pulses with a wideband, uniform rate, smooth truncation (WURST)[143] amplitude envelope, with a target flip angle $\beta = 20^\circ$. Each saltire pulse had a bandwidth of 10 kHz, a duration of 25 ms, and a power of 280 μW . Hard pulses had a power of 31.537 W, with the duration of the $\pi/2$ pulse being 15 μs . G_1 and G_2 were gradients for coherence order selection. Each comprised a 100-point sine shape profile, and lasted 1 ms. G_3 was a rectangular weak gradient applied during the PSYCHE element, with a duration of 52 ms. The gradeint strengths as a percentage of the maximum permissible z-gradient were, respectively 31%, 47%, 1.6%. The phase cycling scheme used was: $\Phi_1 : 2 \times (0^\circ, 180^\circ)$; $\Phi_2 : 4 \times 0^\circ$; $\Phi_3 : 2 \times 0^\circ, 2 \times 90^\circ$; $\Phi_{rx} : 0^\circ, 180^\circ, 180^\circ, 0^\circ$.

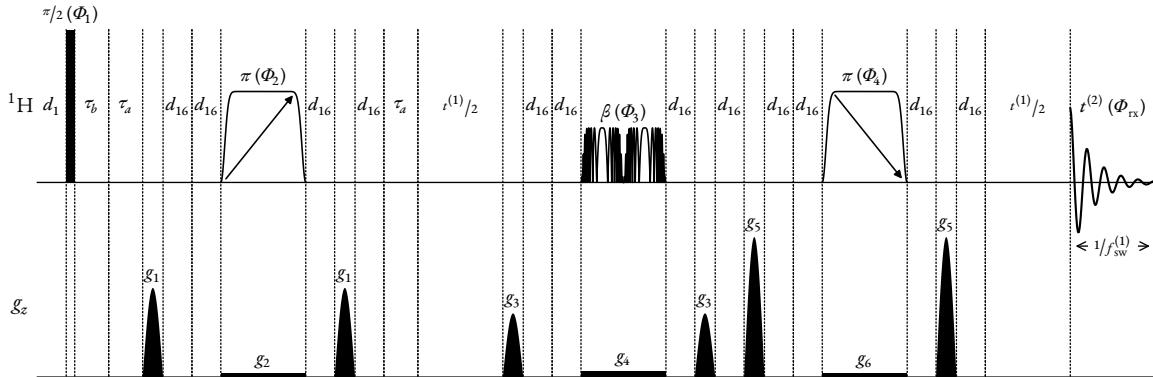


FIGURE D.5: TSE-PSYCHE pulse sequence used for the acquisition of dexamethasone data (Figure 4.8.a). Delays: d_1 (relaxation delay): 2 s, d_{16} : (gradient recovery delay): 200 μs , τ_a : 5 ms ($= 1/4f_{sw}^{(1)}$). The hard $\pi/2$ pulse had a duration of 12 μs , and a power of 24 W. The two π pulses were unidirectional frequency-swept (chirped) pulses, with the first pulse sweeping from low to high frequencies, and the second pulse sweeping from high to low. These each had a WURST amplitude envelope, lasted a duration of 40 ms, and had a power of 11.05 mW. The PSYCHE element had a target flip angle $\beta = 15^\circ$, and featured two saltire chirp pulses. Both saltire pulses had a WURST amplitude envelope, a duration of 15 ms, and a power of 1.28 mW. g_1 , g_3 and g_5 were gradients for coherence order selection. Each comprised a 100-point sine profile, and lasted 1 ms. g_2 , g_4 and g_6 were weak rectangular gradients which were applied at the same time as the chirped pulses. The magnitudes of gradients g_1 to g_6 as a percentage of the maximum permitted gradient were, respectively: 49%, 2%, 35%, 3%, 77%, 2%. The phase cycling scheme used was: $\Phi_1 : 8 \times 0^\circ$; $\Phi_2 : 2 \times (2 \times 0^\circ, 2 \times 180^\circ)$; $\Phi_3 : 2 \times (0^\circ, 90^\circ), 2 \times (180^\circ, 270^\circ)$; $\Phi_4 : 8 \times 0^\circ$; $\Phi_{rx} : 4 \times (0^\circ, 180^\circ)$.

D.3 CUPID result metrics

Needs updating: replace sucrose with strychnine, re-order according to the order in the thesis.

TABLE D.7: Metrics for all the results generated using CUPID (Section 4.3). The *Initial \mathcal{M}* specifies the number of oscillators given to the MMEMP. Values with a $*$ indicate that they were determined by applying the MDL on the first direct-dimension slice of the data. Values with a † indicate that they were manually provided. The \mathcal{M} *after MMEMP* column indicates how many oscillators were present in the initial guess $\boldsymbol{\theta}^{(0)}$. This can differ from *Initial \mathcal{M}* , as any oscillators possessing negative damping factors in the MMEMP result were purged. The \mathcal{M} *after NLP* column indicates how many oscillators were present in $\boldsymbol{\theta}^{(*)}$, the result of non-linear programming. When this value is smaller than \mathcal{M} *after MMEMP*, negative-amplitude oscillators were found and purged during the optimisation. *Purged oscillators* indicates how many oscillators were removed from the final estimation result, based on the first-order signal criteria outlined in Section 4.2.4.

Region (ppm)	Initial \mathcal{M}	MMEMP time (s)	\mathcal{M} after MMEMP	NLP time (s)	NLP iterations	\mathcal{M} after NLP	Purged oscillators
Quinine							
5.8 – 5.55	18*	11.2	18	61.4	124	10	None
5 – 4.85	17 [†]	3.5	17	32.1	47	17	None
3.75 – 3.63	15*	2.4	15	34.7	69	13	None
3.17 – 3.06	15*	1.9	15	67.9	150	10	None
2.8 – 2.6	25 [†]	7.5	25	112.9	125	22	None
2 – 1.7	40 [†]	19.8	40	233.3	150	37	1
1.64 – 1.52	20*	2.8	20	38.4	55	18	None
1.52 – 1.4	14*	2.6	13	26.6	58	12	None
Estradio							
2.29 – 2.17	20 [†]	11.5	17	31.7	150	11	2
2.12 – 2	15 [†]	11.4	15	27.9	150	9	3
1.95 – 1.72	40 [†]	42.7	40	93.2	125	34	4
1.65 – 1.52	20 [†]	12.2	19	47.9	150	15	2

Continues on next page...

Region (ppm)	Initial M	MMEMP time (s)	M after MMEMP	NLP time (s)	NLP iterations	M after NLP	Purged oscillators
0.06 – -0.06	40 [†]	4.2	40	45.6	61	32	None
Four Multiplets (Run 5)							
0.06 – -0.06	37 [†]	3.5	37	74.8	100	33	1
Camphor							
2.55 – 2.475	9*	2.0	8	40.6	150	6	1
2.35 – 2.23	18*	5.7	18	68.3	100	18	None
2.09 – 2.025	8*	1.3	18	18.3	96	3	None
1.95 – 1.75	35*	19.9	32	260.2	225	30	None
1.7 – 1.61	21*	3.9	21	72.3	84	21	1
1.375 – 1.215	29*	10.6	29	117.2	123	22	None
Dexamethasone							
7.45 – 7.15	2 [†]	0.8	2	0.5	13	2	None
6.4 – 5.9	15*	4.4	7	14.2	66	7	None
5.5 – 4.8	3 [†]	4.0	3	1.1	17	3	None
4.8 – 4.3	11*	3.2	11	12.7	58	9	2
4.25 – 3.97	20 [†]	1.1	19	35.2	133	17	0
3 – 2.87	13*	0.3	11	48.8	219	10	None
2.68 – 2.43	23*	1.2	18	68.2	250	14	None

Continues on next page...

Region (ppm)	Initial \mathcal{M}	MMEMP time (s)	\mathcal{M} after MMEMP	NLP time (s)	NLP iterations	\mathcal{M} after NLP	Purged oscillators
2.413 – 2.26	18*	0.395	17	61.7	197	15	None
2.195 – 2.034	17*	0.4	17	61.7	200	17	None
1.85 – 1.7	12*	0.4	10	15.0	79	9	None
1.7 – 1.25	30*	3.2	30	127.2	250	27	8
1.14 – 1	12*	0.4	10	36.9	183	10	3
1 – 0.65	3†	2.0	3	0.9	17	3	None

NMR-EsPy WALKTHROUGHS

E

The remainder of this thesis is an insert from the documentation of version 2.0 of NMR-EsPy. The section of the documentation included provides walkthroughs describing how to use the package's API for the consideration of 1D and 2DJ NMR datasets **Sequential data too?**. These walkthroughs provide a short description of the key features associated with the package, and is an ideal first place to get up-and-running with using it.

**CHAPTER
TWO**

WALKTHROUGHS

In this chapter, walkthroughs are provided to help you get up-and-running with the main features of the NMR-EsPy API. For a rigorous description of the API, you should consult the [Reference](#) afterwards.

2.1 Using Estimator1D

The `nmrespy.Estimator1D` class is provided for the consideration of 1D NMR data.

2.1.1 Generating an instance

There are a few ways to create a new instance of the estimator depending on the source of the data

Bruker data

It is possible to load both raw FID data and processed spectral data from Bruker using `new_bruker()`. All that is needed is the path to the dataset:

1. If you wish to import FID data, set the path as "`<path_to_data>/<expno>/`". There should be an `fid` file and an `acqus` file directly under this directory. The data in the `fid` file will be imported, and the artefact from digital filtering is removed by a first-order phase shift.

Note: If you import FID data, there is a high chance that you will need to phase the data, and apply baseline correction before proceeding to run estimation. Look at `phase_data()` and `baseline_correction()`, respectively.

```
>>> import nmrespy as ne
>>> estimator = ne.Estimator1D.new_bruker("/home/simon/nmr_data/andrographolide/1")
>>> estimator.phase_data(p0=2.653, p1=-5.686, pivot=13596)
>>> estimator.baseline_correction()
```

2. To import processed data, set the path as "`<path_to_data>/<expno>/pdata/<procno>`". There should be a `1r` file and a `procs` file directly under this directory. The data in `1r` will be Inverse Fourier Transformed, and the resulting time-domain signal is sliced so that only the first half is retained.

Note: It can be more convenient to provide processed data, even though the data will be converted to the time-domain for estimation, as you can then rely on TopSpin's automated processing scripts

NMR-EsPy, Release 2.0

to phase and baseline correct. However, **you should not apply any window function to the data other than exponential line broadening.**

```
>>> import nmresp as ne
>>> estimator = ne.Estimator1D.new_bruker("home/simon/nmr_data/andrographolide/1/pdata/1")
>>> # Note there is no need for extra data-processing steps
```

Simulated data from a set of oscillator parameters

You can create an estimator with synthetic data constructed from known parameters using `new_from_parameters()`. The parameters must be provided as a 2D NumPy array with `params.shape[1] == 4`. Each row should contain an oscillator's amplitude, phase (rad), frequency (Hz), and damping factor (s^{-1}).

```
>>> import nmresp as ne
>>> import numpy as np
>>> # Using frequencies of 2,3-Dibromopropanoic acid @ 500MHz
>>> params = np.array([
>>>     [1., 0., 1864.4, 7.],
>>>     [1., 0., 1855.8, 7.],
>>>     [1., 0., 1844.2, 7.],
>>>     [1., 0., 1835.6, 7.],
>>>     [1., 0., 1981.4, 7.],
>>>     [1., 0., 1961.2, 7.],
>>>     [1., 0., 1958.8, 7.],
>>>     [1., 0., 1938.6, 7.],
>>>     [1., 0., 2265.6, 7.],
>>>     [1., 0., 2257.0, 7.],
>>>     [1., 0., 2243.0, 7.],
>>>     [1., 0., 2234.4, 7.],
>>> ])
>>> sfo = 500.
>>> estimator = ne.Estimator1D.new_from_parameters(
>>>     params=params,
>>>     pts=2048,
>>>     sw=1.2 * sfo, # 1ppm
>>>     offset=4.1 * sfo, # 4.1ppm
>>>     sfo=sfo,
>>>     snr=40.,
>>> )
```

Note: For the rest of this tutorial, we will be using the estimator created in the above code snippet.

Simulated data from Spinach

Assuming you have installed the *relevant requirements*, you can create an instance with data simulated using Spinach with `new_spinach()`. The spin system is defined by a specification of isotropic chemical shifts, and scalar couplings:

- For the chemical shift, a list of floats is required.
- For J-couplings, a list with 3-element tuples of the form (`spin1, spin2, coupling`) is required.
N.B. the spin indices start at “1” rather than “0”.

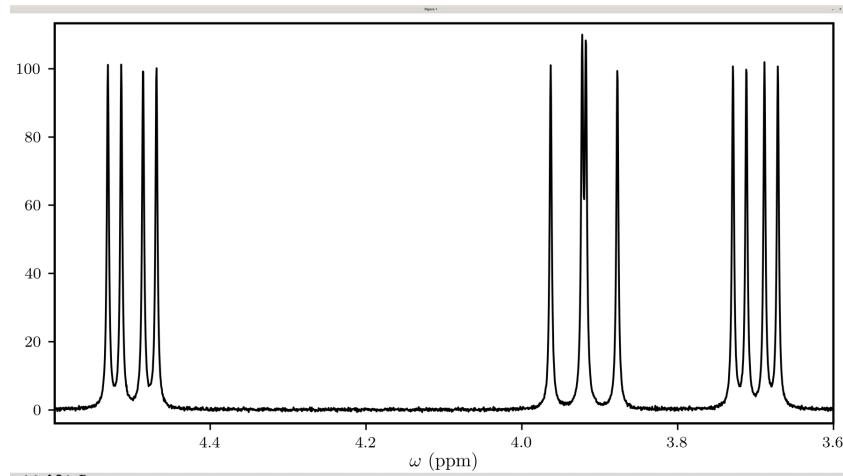
It can take some time to run this function if it involves (a) starting up MATLAB and (b) running a simulation of the experiment.

```
>>> import nmrespy as ne
>>> # 2,3-Dibromopropionic acid
>>> shifts = [3.7, 3.92, 4.5]
>>> couplings = [(1, 2, -10.1), (1, 3, 4.3), (2, 3, 11.3)]
>>> sfo = 500.
>>> offset = 4.1 * sfo # Hz
>>> sw = 1.2 * sfo
>>> estimator = ne.Estimator1D.new_spinach(
>>>     shifts=shifts,
>>>     couplings=couplings,
>>>     pts=2048,
>>>     sw=sw,
>>>     offset=offset,
>>>     sfo=sfo,
>>> )
```

2.1.2 Viewing and accessing the dataset

You can inspect the data associated with the estimator with `view_data()`, which loads an interactive matplotlib figure:

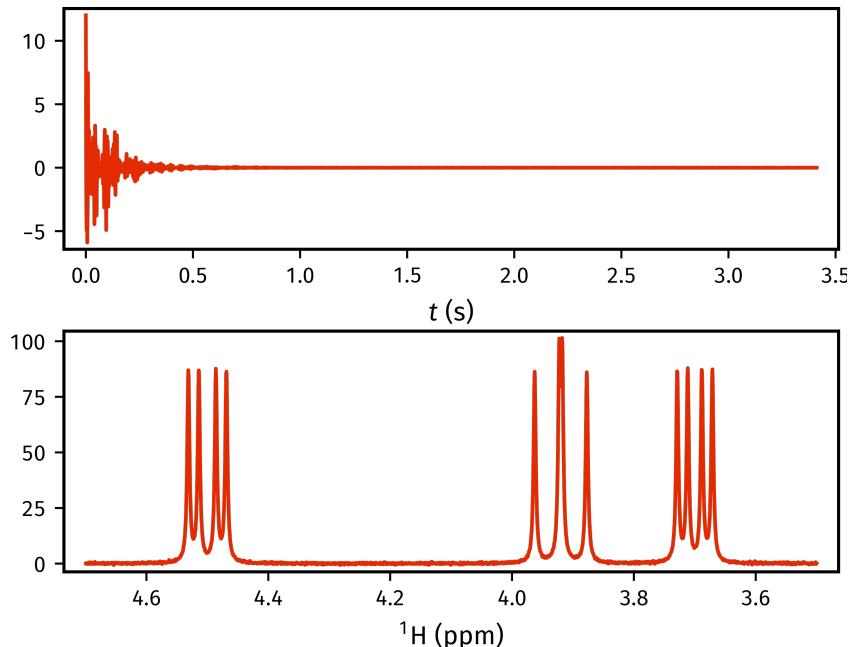
```
>>> estimator.view_data(freq_unit="ppm")
```



You can access the time-domain data with the `data()` property, and the associated time-points can be retrieved using `get_timepoints()`. The spectral data is accessed with `spectrum()`, and the corresponding chemical shifts with `get_shifts()`.

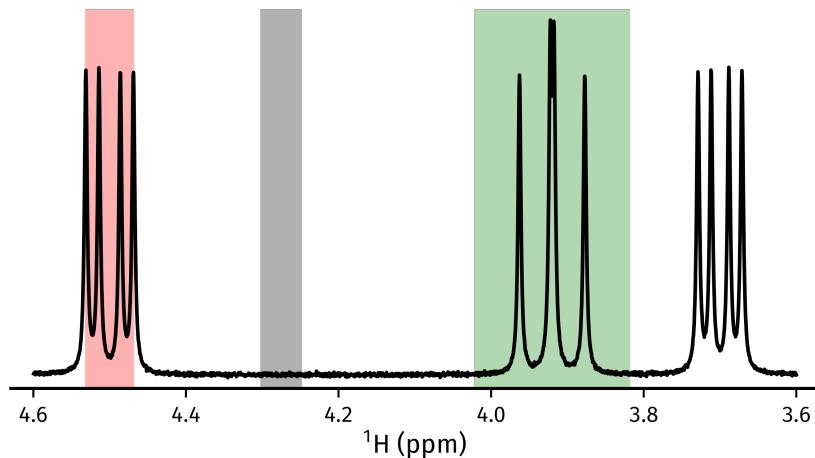
NMR-EsPy, Release 2.0

```
>>> fid = estimator.data
>>> tp = estimator.get_timepoints()[0]
>>> spectrum = estimator.spectrum
>>> shifts = estimator.get_shifts(unit="ppm")[0]
>>> fig, axs = plt.subplots(nrows=2)
>>> axs[0].plot(tp, fid.real)
>>> axs[0].set_xlabel("$t$ (s)")
>>> axs[1].plot(shifts, spectrum.real)
>>> # Flip x-axis limits (ensure plotting from high to low shifts)
>>> axs[1].set_xlim(reversed(axs[1].get_xlim())))
>>> axs[1].set_xlabel("$^1\text{H}$ (ppm)")
>>> plt.show()
```



2.1.3 Estimating the dataset

The generation of parameter estimates for the dataset is facilitated using the `estimate()` method. In most scenarios, your dataset will possess too many oscillators for it to be feasible computationally to estimate the entire signal at once. For this reason, NMR-EsPy generates frequency-filtered “sub-FIDs” to break the problem down into more manageable chunks. To create suitable sub-FIDs, it is important to select regions where the bounds are placed at points that comprise the baseline. As well as this, a region that comprises just the baseline must be indicated. In the figure below, the red region would be inappropriate as it slices through signal. The green region is acceptable, as the bounds are located on the baseline. Finally, the grey region is a suitable noise region as it contains only baseline.



For our dataset, we will estimate three regions, comprising each multiplet structure in the spectrum. A region should be given as a tuple of 2 floats, specifying the left and right boundaries of the region of interest (the order of these doesn't matter). By default, these are assumed to be given in Hz, unless `region_unit` is set to "ppm".

```
>>> regions = [(4.6, 4.4), (4.02, 3.82), (3.8, 3.6)]
>>> noise_region = (4.3, 4.25)
>>> for region in regions:
>>>     estimator.estimate(
>>>         region=region, noise_region=noise_region, region_unit="ppm",
>>>     )
```

2.1.4 Inspecting estimation results

Note: Result Indices

Each time the `estimate()` method is called, the result is appended to a list of all generated results. For many methods that use estimation results, an argument called `indices` exists. This lets you specify the results you are interested in. By default (`indices = None`) all results will be used. A subset of the results can be considered by including a list of integers. For example `indices = [0, 2]` would mean only the 1st and 3rd results acquired with the estimator are considered.

A NumPy array of the generated results can be acquired using `get_params()`. The corresponding errors associated with each parameters are obtained with `get_errors()`.

```
>>> # All params, frequencies in Hz:
>>> estimator.get_params()
[[ 1.0018e+00  1.5921e-03  1.8356e+03  7.0187e+00]
 [ 1.0003e+00  2.4881e-03  1.8442e+03  6.9968e+00]
 [ 1.0024e+00  1.5817e-03  1.8558e+03  7.0281e+00]
 [ 1.0008e+00  9.1591e-04  1.8644e+03  7.0007e+00]
 [ 1.0022e+00  7.1936e-04  1.9386e+03  7.0109e+00]
 [ 9.9470e-01 -7.4609e-04  1.9588e+03  6.9866e+00]
 [ 1.0080e+00 -1.0112e-03  1.9612e+03  7.0448e+00]
 [ 1.0009e+00 -7.1398e-04  1.9814e+03  7.0131e+00]
 [ 1.0003e+00  1.1306e-03  2.2344e+03  7.0095e+00]
 [ 1.0011e+00  6.0150e-04  2.2430e+03  7.0011e+00]]
```

(continues on next page)

NMR-EsPy, Release 2.0

(continued from previous page)

```
[ 9.9902e-01  2.8231e-04  2.2570e+03  6.9856e+00]
[ 1.0004e+00 -1.8229e-03  2.2656e+03  7.0057e+00]]
>>> # All errors, frequencies in Hz
>>> estimator.get_errors()
[[0.0013 0.0013 0.0019 0.0121]
 [0.0014 0.0014 0.002  0.0124]
 [0.0014 0.0014 0.002  0.0125]
 [0.0013 0.0013 0.0019 0.012 ]
 [0.0012 0.0012 0.0018 0.0114]
 [0.0036 0.0036 0.0034 0.0212]
 [0.0036 0.0036 0.0034 0.0213]
 [0.0012 0.0012 0.0018 0.0114]
 [0.0013 0.0013 0.0019 0.0116]
 [0.0013 0.0013 0.0019 0.0118]
 [0.0013 0.0013 0.0019 0.0118]
 [0.0013 0.0013 0.0018 0.0116]]
>>> # Params for first region, frequencies in ppm
>>> estimator.get_params(indices=[0], funit="ppm")
[[ 1.0003e+00  1.1306e-03  4.4688e+00  7.0095e+00]
 [ 1.0011e+00  6.0150e-04  4.4860e+00  7.0011e+00]
 [ 9.9902e-01  2.8231e-04  4.5140e+00  6.9856e+00]
 [ 1.0004e+00 -1.8229e-03  4.5312e+00  7.0057e+00]]
>>> # Params for second and third regions, split up
>>> estimator.get_params(indices=[1, 2], merge=False, funit="ppm")
[array([[ 1.0022e+00,  7.1936e-04,  3.8772e+00,  7.0109e+00],
       [ 9.9470e-01, -7.4609e-04,  3.9176e+00,  6.9866e+00],
       [ 1.0080e+00, -1.0112e-03,  3.9224e+00,  7.0448e+00],
       [ 1.0009e+00, -7.1398e-04,  3.9628e+00,  7.0131e+00]], array([[1.0018e+00,  1.5921e-03,
       3.6712e+00,  7.0187e+00],
       [ 1.0003e+00,  2.4881e-03,  3.6884e+00,  6.9968e+00],
       [ 1.0024e+00,  1.5817e-03,  3.7116e+00,  7.0281e+00],
       [ 1.0008e+00,  9.1591e-04,  3.7288e+00,  7.0007e+00]])]
```

Writing result tables

Tables of parameters can be saved to .txt and .pdf formats, using `write_result()`. For PDF generation, you will need a working LaTeX installation. See the [installation instructions](#).

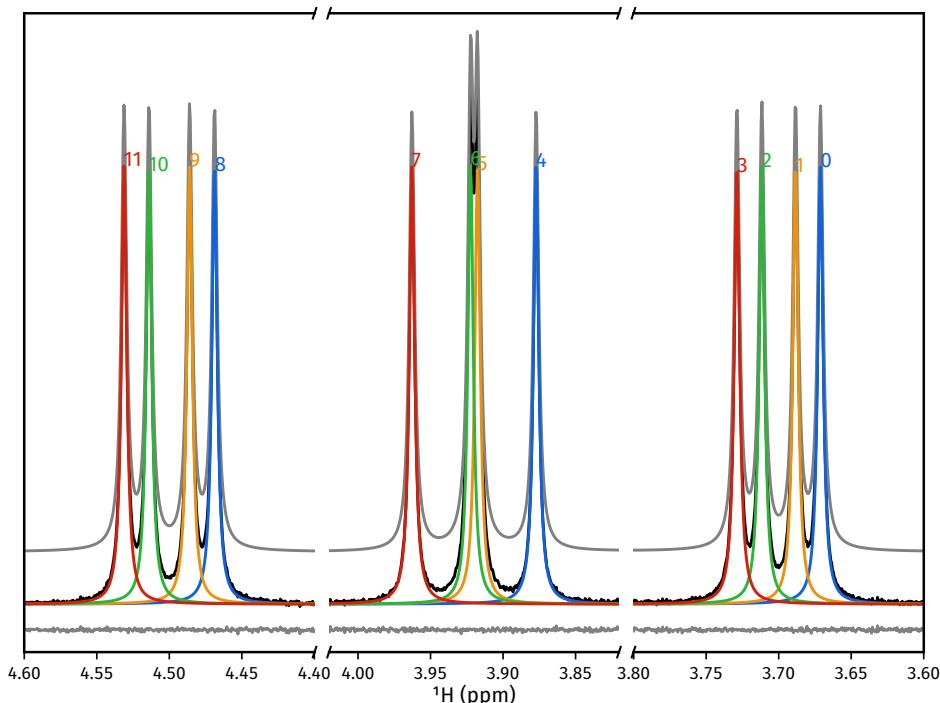
```
>>> for fmt in ("txt", "pdf"):
>>>     estimator.write_result(
>>>         path="tutorial_1d",
>>>         fmt=fmt,
>>>         description="Simulated 2,3-Dibromopropanoic acid signal.",
>>>     )
Saved file tutorial_1d.txt.
Saved file tutorial_1d.tex.
Saved file tutorial_1d.pdf.
You can view and customise the corresponding TeX file at tutorial_1d.tex.
```

Creating result plots

Figures giving an overview of the estimation result can be generated using `plot_result()`.

```
>>> for (txt, indices) in zip(("complete", "index_1"), (None, [1])):
>>>     fig, ax = estimator.plot_result(
>>>         indices=indices,
>>>         figure_size=(4.5, 3.),
>>>         region_unit="ppm",
>>>         axes_left=0.03,
>>>         axes_right=0.97,
>>>         axes_top=0.98,
>>>         axes_bottom=0.09,
>>>     )
>>>     fig.savefig(f"tutorial_1d_{txt}_fig.pdf")
```

Below is the figure `tutorial_1d_complete_fig.pdf`:



Saving the estimator

The `estimator` object itself can be saved and reloaded for future use with the `to_pickle()` and `from_pickle()` methods, respectively:

```
>>> estimator.to_pickle("tutorial_1d")
Saved file tutorial_1d.pkl.
>>> # Load the estimator and save to the `estimator_cp` variable
>>> estimator_cp = ne.Estimator1D.from_pickle("tutorial_1d")
```

NMR-EsPy, Release 2.0**Saving a logfile**

A logfile listing all the methods called on the estimator can be saved using `save_log()`:

```
>>> estimator.save_log("tutorial_1d")
Saved file tutorial_1d.log.
```

2.2 Using Estimator2DJ

The `nmrespy.Estimator2DJ` class enables the estimation of J-Resolved (2DJ) spectroscopy datasets. This facilitates use of **CUPID** (Computer-assisted Ultrahigh-resolution Protocol for Ideal Decoupling) which can be used to generate homodecoupled spectra and to predicting multiplet structures.

Many methods in this class have analogues in `Estimator1D`. You are advised to read through the *1D walkthrough* before continuing, as I will be providing minimal descriptions of things covered in that.

2.2.1 Generating an instance

Bruker data

Use `new_bruker()`. Unlike `Estimator1D`, you must import time-domain 2DJ data. The path should be set as "`<path_to_data>/<exnpo>`". There should be a `ser` file, an `acqus` file, and an `acqu2s` file directly under this directory. Again, phasing in the direct-dimension will be needed. If deemed necessary, baseline correction is possible too.

```
>>> import nmrespy as ne
>>> estimator = ne.Estimator2DJ.new_bruker("/home/simon/nmr_data/quinine/dexamethasone/2")
>>> estimator.phase_data(p0=0.041, p1=-6.383, pivot=1923)
>>> estimator.baseline_correction()
<Estimator2DJ object at 0x7f4b7e1f5cd0>
```

Experiment Information

Parameter	F1	F2
Nucleus	¹ H	¹ H
Transmitter Frequency (MHz)	N/A	600.18
Sweep Width (Hz)	50	7211.5
Sweep Width (ppm)	N/A	12.016
Transmitter Offset (Hz)	0	2815.4
Transmitter Offset (ppm)	N/A	4.691

No estimation performed yet.

Simulated data from Spinach

Use `new_spinach()`

```
>>> # Sucrose shifts and couplings
>>> shifts = [
>>>     6.005,
>>>     3.510,
>>>     3.934,
>>>     3.423,
>>>     4.554,
>>>     3.891,
>>>     4.287,
>>>     3.332,
>>>     1.908,
>>>     1.555,
>>>     0.644,
>>>     4.042,
>>>     4.517,
>>>     3.889,
>>>     4.635,
>>>     4.160,
>>>     4.021,
>>>     4.408,
>>>     0.311,
>>>     1.334,
>>>     0.893,
>>>     0.150,
>>> ]
>>> couplings = [
>>>     (1, 2, 2.285),
>>>     (2, 3, 4.657),
>>>     (2, 8, 4.828),
>>>     (3, 4, 4.326),
>>>     (4, 5, 4.851),
>>>     (5, 6, 5.440),
>>>     (5, 7, 2.288),
>>>     (6, 7, -6.210),
>>>     (7, 11, 7.256),
>>>     (12, 13, -4.005),
>>>     (12, 19, 1.460),
>>>     (14, 15, 4.253),
>>>     (15, 16, 4.448),
>>>     (15, 21, 3.221),
>>>     (16, 18, 4.733),
>>>     (17, 18, -4.182),
>>>     (18, 22, 1.350),
>>> ]
>>> estimator = ne.Estimator2DJ.new_spinach(
>>>     shifts=shifts,
>>>     couplings=couplings,
>>>     pts=(64, 4096),
>>>     sw=(30., 2200.),
>>>     offset=1000.,
>>>     field=300.,
>>>     field_unit="MHz",
>>>     snr=20.,
>>> )
```

NMR-EsPy, Release 2.0

Note: We will be using the estimator generated above for the rest of this tutorial. If you do not have access to MATLAB/Spinach, you can construct the estimator by using an FID I made earlier:

```
>>> import nmrspy as ne
>>> from pathlib import Path
>>> import pickle
>>> fid_path = Path(ne.__file__).expanduser().parents[1] \
...     / "samples/jres_sucrose_sythetic/sucrose_jres_fid.pkl"
>>> with open(fid_path, "rb") as fh:
...     fid = pickle.load(fh)
...
>>> expinfo = ne.ExpInfo(
...     dim=2,
...     sw=(30., 2200.),
...     offset=(0., 1000.),
...     sfo=(None, 300.),
...     nucleus=(None, "1H"),
...     default_pts=(64, 4096),
... )
>>> estimator = ne.Estimator2DJ(fid, expinfo)
```

2.2.2 Estimating the dataset

The procedure for estimating 2DJ data is very similar to that of 1D data. You need to specify regions in the direct dimension that are of interest for generating filtered sub-FIDs. No filtering is done in the indirect dimension. In our example, it turns out that for a couple of the regions selected, the number of oscillators automatically generated is slightly smaller than the “true” number, and so we force the optimiser to use the true number (see the lines involving `initial_guesses`).

```
>>> regions = (
...     (6.08, 5.91),
...     (4.72, 4.46),
...     (4.46, 4.22),
...     (4.22, 4.1),
...     (4.09, 3.98),
...     (3.98, 3.83),
...     (3.58, 3.28),
...     (2.08, 1.16),
...     (1.05, 0.0),
... )
>>> n_regions = len(regions)
>>> initial_guesses = n_regions * [None]
>>> initial_guesses[1:3] = [16, 16]
>>> # kwargs common to estimation of each region
>>> common_kwargs = {
...     "noise_region": (5.5, 5.33),
...     "region_unit": "ppm",
...     "max_iterations": 200,
...     "phase_variance": True,
... }
>>> for init_guess, region in zip(initial_guesses, regions):
...     kwargs = {**{"region": region, "initial_guess": init_guess}, **common_kwargs}
...     estimator.estimate(**kwargs)
```

(continues on next page)

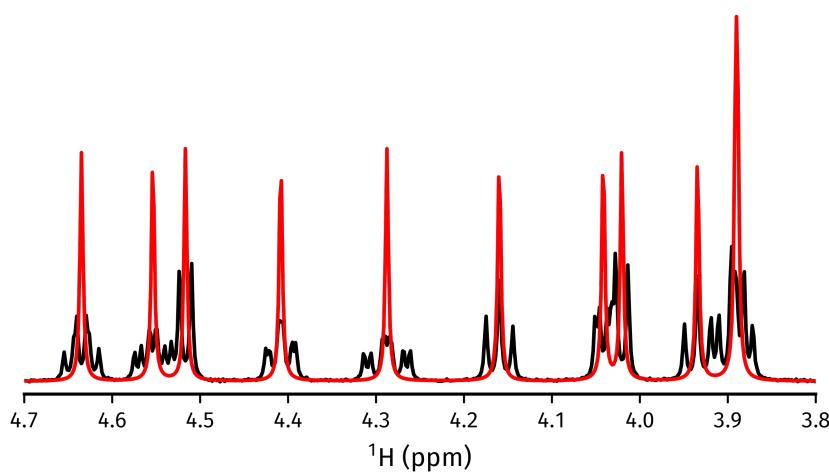
(continued from previous page)

```
>>> # It is a good idea to pickle the estimator after estimation
>>> estimator.to_pickle("sucrose")
```

2.2.3 Acquiring a homodecoupled spectrum

The `cupid_spectrum()` method produces a homodecoupled spectrum using the estimation parameters. In the code snippet below, a figure is made comparing the homodecoupled spectrum with the spectrum of the first direct-dimension slice in the 2DJ data, which is a normal 1D spectrum.

```
>>> # Normal 1D spectrum
>>> init_spectrum = estimator.spectrum_zero_t1.real
>>> # Homodecoupled spectrum produced using CUPID
>>> cupid_spectrum = estimator.cupid_spectrum().real
>>> # Get direct-dimension shifts
>>> shifts = estimator.get_shifts(unit="ppm", meshgrid=False)[-1]
>>> import matplotlib.pyplot as plt
>>> fig, ax = plt.subplots(figsize=(4.5, 2.5))
>>> ax.plot(shifts, init_spectrum, color="k")
>>> ax.plot(shifts, cupid_spectrum, color="r")
>>> # The most interesting region of the spectrum
>>> ax.set_xlim(4.7, 3.8)
>>> # =====
>>> # These lines are just for plot aesthetics
>>> for x in ("top", "left", "right"):
>>>     ax.spines[x].set_visible(False)
>>> ax.set_xticks([4.7 - 0.1 * i for i in range(10)])
>>> ax.set_yticks([])
>>> ax.set_position([0.03, 0.175, 0.94, 0.83])
>>> ax.set_xlabel(f"{estimator.latex_nuclei[1]} (ppm)")
>>> # =====
>>> fig.savefig("cupid_spectrum.png")
```



2.2.4 Multiplet prediction

Oscillators belonging to the same multiplet can be predicted based on the fact that in a 2DJ signal any pair should satisfy the following:

$$\left| \left(f_i^{(2)} - f_i^{(1)} \right) - \left(f_j^{(2)} - f_j^{(1)} \right) \right| < \epsilon$$

where ϵ is an error threshold. $f^{(1)}$ and $f^{(2)}$ are the indirect- and direct-dimension frequencies, respectively. The `predict_multiplets()` generates groups of oscillator indices satisfying the above criterion. A key parameter for this is `thold`, which sets the error threshold ϵ . By default, this is set to be $\max(f_{\text{sw}}^{(1)}/N^{(1)}, f_{\text{sw}}^{(2)}/N^{(2)})$, i.e whichever is larger out of the indirect- and direct-dimension spectral resolutions. However, especially when considering real data, this threshold can be a little optimistic. For good multiplet groupings, you may need to manually provide a slightly larger threshold.

In the example below, multiplet groups are determined for regions with indices 1-5 (covering the region plotted above).

```
>>> indices = [1, 2, 3, 4, 5]
>>> multiplets = estimator.predict_multiplets(indices=indices)
>>> for (freq, idx) in multiplets.items():
...     print(f"freq / estimator.sfo[1]:.4f}ppm: {idx}")
...
3.8890ppm: [1, 4]
3.8910ppm: [0, 2, 3, 5]
3.9344ppm: [6, 7, 8]
4.0205ppm: [9, 10]
4.0416ppm: [11, 12, 13, 14]
4.1598ppm: [15, 16, 17]
4.2876ppm: [18, 19, 20, 21, 22, 23, 24, 25]
4.4083ppm: [26, 27, 28, 29, 30, 31, 32, 33]
4.5167ppm: [34, 35]
4.5537ppm: [36, 37, 38, 39, 40, 41, 42, 43]
4.6349ppm: [44, 45, 46, 47, 48, 49]
```

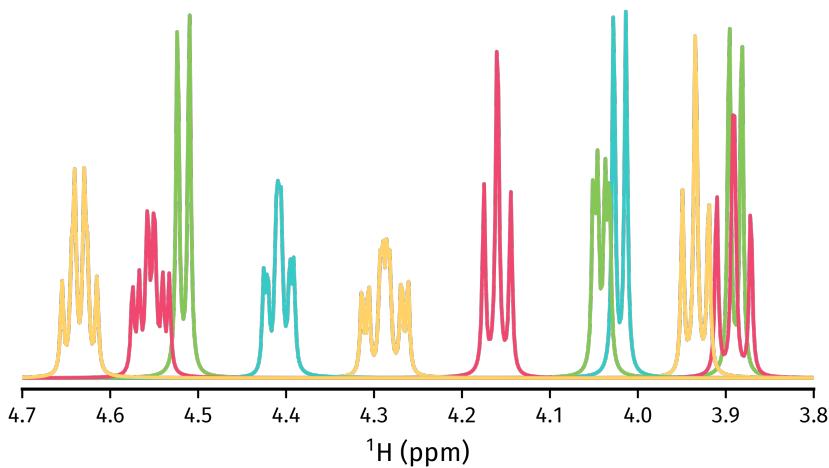
To generate FIDs corresponding to each multiplet structure, use the `construct_multiplet_fids()` method. In the following code snippet, each generated FID undergoes FT, with all the spectra being plotted.

```
>>> # Direct-dimension shifts
>>> shifts_f2 = estimator.get_shifts(unit="ppm", meshgrid=False)[-1]
>>> fids = estimator.construct_multiplet_fids(indices=indices)
>>> # Create an iterator which cycles through values infinitely
>>> from itertools import cycle
>>> colors = cycle(["#84c757", "#ef476f", "#ffd166", "#36c9c6"])
>>> fig, ax = plt.subplots(figsize=(4.5, 2.5))
>>> for fid in fids:
...     # Halve first point prior to FT to prevent vertical baseline shift
...     fid[0] *= 0.5
...     # FT and retrieve real component
...     spectrum = ne.sig.ft(fid).real
...     ax.plot(shifts_f2, spectrum, color=next(colors))
...
>>> ax.set_xlim(4.7, 3.8)
>>> # =====
>>> # These lines are just for plot aesthetics
>>> for x in ("top", "left", "right"):
```

(continues on next page)

(continued from previous page)

```
...     ax.spines[x].set_visible(False)
...
>>> ax.set_xticks([4.7 - 0.1 * i for i in range(10)])
>>> ax.set_yticks([])
>>> ax.set_position([0.03, 0.175, 0.94, 0.83])
>>> ax.set_xlabel(f"{estimator.latex_nuclei[1]} (ppm)")
>>> # =====
>>> fig.savefig("multiplets.png")
```



2.2.5 Generating tilted spectra

The well-known 45° “tilt” that is applied to 2DJ spectra for orthogonal separation of chemical shifts and scalar couplings effectively maps the frequencies in the direct dimension $f^{(2)}$ to $f^{(2)} - f^{(1)}$. Armed with an estimation result, a signal with these adjusted frequencies can easily be constructed. As well as this, generating a pair of phase- or amplitude-modulated FIDs enables the construction of absorption-mode spectra. Use `nmrespypy.Estimator2DJ.sheared_signal()`, with `indirect_modulation` set to either "amp" or "phase" to generate the desired spectra. Then, you can use either `nmrespypy.sig.proc_phase_modulated()` or `nmrespypy.sig.proc_amp_modulated()` as appropriate to construct the spectrum:

```
>>> # Generate P- and N- type FIDs with "sheared" frequencies
>>> sheared_fid = estimator.sheared_signal(indirect_modulation="phase")
>>> # sheared_fid[0] -> P-type, sheared_fid[1] -> N-type
>>> sheared_fid.shape
(2, 64, 4096)
>>> # Generates 2rr, 2ri, 2ir, 2ii spectra
>>> sheared_spectrum = ne.sig.proc_phase_modulated(sheared_fid)
>>> sheared_spectrum.shape
(4, 64, 4096)
>>> spectrum_2rr = sheared_spectrum[0]
>>> # Note the `meshgrid` kwarg is True here to make 2D shift arrays
>>> shifts_f1, shifts_f2 = estimator.get_shifts(unit="ppm")
>>> fig, ax = plt.subplots(figsize=(4.5, 2.5))
>>> # Contour levels
>>> base, factor, nlevels = 25, 1.3, 10
>>> levels = [base * factor ** i for i in range(nlevels)]
>>> ax.contour(
```

(continues on next page)

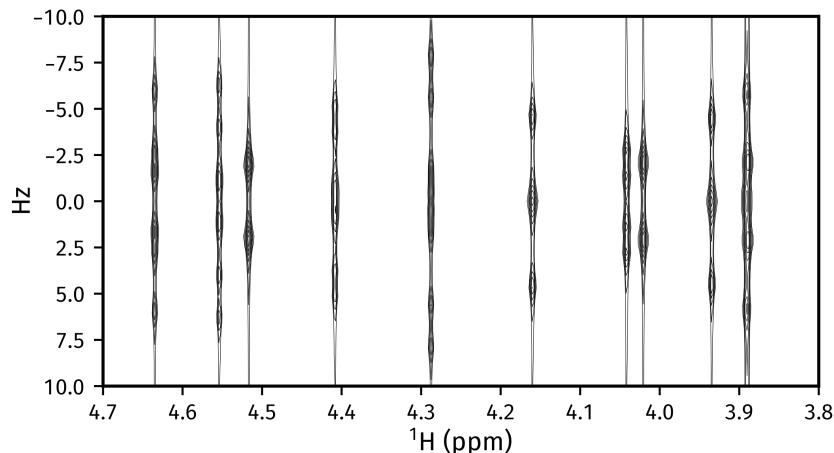
NMR-EsPy, Release 2.0

(continued from previous page)

```

...
    shifts_f2,
...
    shifts_f1,
...
    spectrum_2rr,
    colors="k",
...
    levels=levels,
    linewidths=0.3,
...
)
>>> ax.set_xlim(4.7, 3.8)
>>> ax.set_ylim(10., -10.)
>>> # =====
>>> # These lines are just for plot aesthetics
>>> ax.set_xticks([4.7 - 0.1 * i for i in range(10)])
>>> ax.set_position([0.12, 0.175, 0.85, 0.79])
>>> ax.set_xlabel(f"{estimator.latex_nuclei[1]} (ppm)", labelpad=1)
>>> ax.set_ylabel("Hz", labelpad=1)
>>> # =====
>>> fig.savefig("sheared_spectrum.png")

```



2.2.6 Plotting result figures

The `plot_result()` method enables the generation of a figure giving an overview of the estimation result. The figure comprises the following, from top to bottom:

- The homodecoupled spectrum generated using `cupid_spectrum()`.
- The 1D spectrum corresponding to the 2DJ dataset.
- The multiplet structures predicted. Note that to get decent multiplet assignments, you may need to increase the value of the `multiplet_thold` argument manually.
- A contour plot of the 2DJ spectrum, with points indicating the positions of estimated peaks.

```

>>> fig, axs = estimator.plot_result(
...     indices=[1, 2, 3, 4, 5],
...
...     region_unit="ppm",
...
...     marker_size=5.,
...
...     figsize=(4.5, 2.5),
...
...     # Number of points to construct homodecoupled signal

```

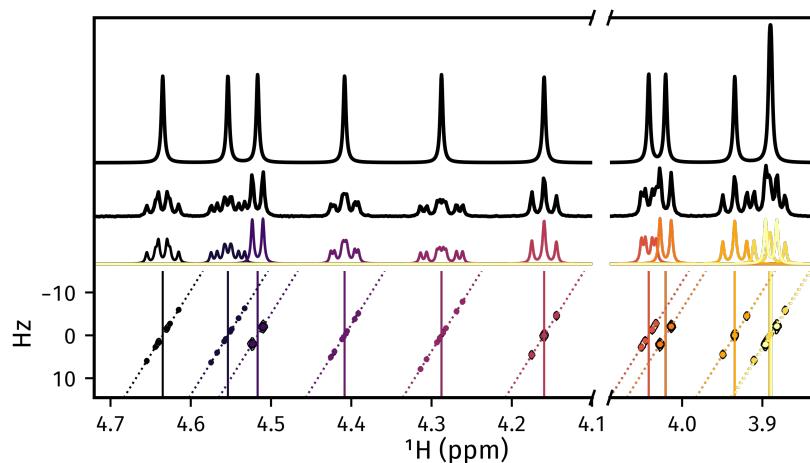
(continues on next page)

(continued from previous page)

```

...
# and multiplet structures from
...
high_resolution_pts=16384,
...
# There is a lot of scope for editing the colours of
# multiplets. See the reference!
...
# Here I specify a recognised name of a colourmap in
# matplotlib.
...
multiplet_colors="inferno",
...
# Arguments of the position of the plot in the figure
...
axes_left=0.1,
...
axes_bottom=0.15,
...
)
>>> fig.savefig("plot_result_2dj.png")

```



2.2.7 Writing data to Bruker format

Note: I am aware that it is currently not possible to analyse the pdata that is generated by NMR-EsPy (you'll get an error along the lines of "*This application requires frequency domain data*" if you try to peak pick, integrate etc). As a workaround for now, you can run a processing script on the FID to generate processed data that is readable by TopSpin. All I do to get the spectrum from the FID is halve the initial point and FT (there is no apodisation).

Multiplets

To write individual multiplet structures to separate Bruker experiments, you can use `write_multiplets_to_bruker()`. It is a good idea to set a prefix for the experiment numbers, especially if the directory you are saving to already has data directories, so you can easily remember which of the directories correspond to multiplet structures. In the example below, as 22 multiplets were resolved, and `expno_prefix` was set to 99, the directories 9901/, 9902/, ..., 9922/ are created.

```

>>> estimator.write_multiplets_to_bruker(
...
    path=".",
...
    expno_prefix=99,
...
    pts=16384,

```

(continues on next page)

NMR-EsPy, Release 2.0

(continued from previous page)

```
...     force_overwrite=True,
...
Saved multiplets to folders ./[9901-9922]/
```

CUPID data

To save the homodecoupled signal generated by our CUPID method, use `write_cupid_to_bruker()`:

```
>>> estimator.write_cupid_to_bruker(
...     path=".",
...     expno=1111,
...     pts=16384,
...
Saved CUPID signal to ./1111/
```

2.2.8 Miscellaneous

For writing result tables to text and PDF files, saving estimators to binary files for later use, and saving log files, look at the relevant sections in the [Using Estimator1D](#) walkthrough. The process is identical.