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
Dr Mohammadali Foroozandeh,
Prof. Timothy Claridge,
& Dr Fay Probert

In loving memory of my Grandmother, Joyce

Contents

Acknowledgements

To begin, I would like to thank Dr Mohammadali Foroozandeh, my primary supervisor for the majority of my PhD. It was an honour to be able to discuss my work with him; his deep insights into a plethora of topics in magnetic resonance is inspiring if a little daunting to mortals like myself. It was a pleasure to be part of the Foroozandeh group; thanks to fellow members Jonanthan Yong, Jean-Baptiste Verstraete, Ali Sherzad, David Goodwin, Thomas Moss, and Daniel Alimadadian for the numerous discussions and pints over the years. For their generosity in covering my course fees and living costs, I'd like to thank The Royal Society, the funders of Dr Foroozandeh's research stipend.

Prof. Timothy Claridge acted as my co-supervisor alongside Dr Foroozandeh. The regular meetings on NMR that he hosted in the Chemistry Department were very valuable; they informed me on how people were applying NMR in their research, and 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; I wish them both the best with their new endeavours. I am very grateful to Dr Fay Probert for taking over my supervision 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 being so welcoming to me during the last year of my PhD. I'd also like to acknowledge Dr James Montgomery, who was kind enough to support me with the operation of the department's NMR spectrometers when I eventually decided I should acquire some of my own data.

During my 8 years both as an undergraduate and postgraduate student at Oxford, I met many wonderful people that made my time thoroughly enjoyable. To name a few, thanks to the 2015 Jesus College Chemists, Eve, Lauren, Ryan, Joe H, Jenyth, Dominic, Linda, Daniel, Valeria, Joe L, Nick, and James. Most of all, I am immeasurably grateful to Isobel, for her support and inspiration throughout my Masters and PhD; I truly could not have done it without her. Thanks too to Laura, Katherine, and of course Pixie, for hosting me numerous times in the Cotswolds.

Finally, to my family, your enduring love and support cannot be understated. In particular, thanks to my brother Robert, for being a great source of encouragement when I was applying to university undergraduate programmes.

Abstract

Nuclear magnetic resonance (NMR) spectroscopy is an analytical technique employed in many scientific disciplines that is able to provide insights into the structures and dynamics of chemical species. To maximise the utility of NMR, 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). NMR spectra provide a human-interpretable representation of data; trained practitioners are able to rationalise the appearance of a given spectrum by mapping its component peaks to chemical environments in the sample from which the dataset was acquired. However, the FT suffers from poor resolution, with peaks of similar frequencies exhibiting overlap. Disentangling the information associated with such peaks is not feasible using typical methods such as integrating user-defined regions of the spectrum. As an alternative approach, parametric estimation techniques aim to provide a detailed description of each signal which contributes to the FID. These methods have been shown to perform effectively even in scenarios where significant spectral peak overlap exists.

This thesis focusses on the development of a parametric estimation method for the analysis of 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 being able to provide accurate and reliable signal estimates. 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 work was to develop a procedure for the generation of broadband homodecoupled (pure shift) NMR spectra with desirable properties from 2DJ datasets. Furthermore, a means of analysing datasets such as those from inversion recovery (T_1), Carr-Purcell-Meiboom-Gill (CPMG) (T_2), and diffusion experiments, in which each FID exhibits a variation in its amplitude, is presented. The last application described is a means of producing phased, ultra-broadband NMR spectra from an experiment comprising a single frequency-swept (chirp) 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).

Declaration of Authorship

This thesis comprises work solely conducted by me, except for the acquisition of the following experimental NMR data:

- Mohammadali Foroozandeh acquired the single chirp excitation data (Figure 3.10), and the datasets involving quinine (Figure 4.8) and camphor (Figure 4.9).
- Jonathan Yong acquired the datasets involving andrographoilde (Figures 3.2 and 3.6) and dexamethasone (Figure 4.10).
- James Montgomery assisted me in acquiring the datasets involving 17β-estradiol (Figure 4.11).
- The dataset involving cyclosporin A (Figure 3.3) was sourced from Bruker's TopSpin software (version 4.0.8).

This work has not been submitted, either wholly or substantially, for any other qualification at this or any other institution.

Acronyms

1D one-dimensional

2D two-dimensional

2DJ two-dimensional J-resolved

3D three-dimensional

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

CORE component resolution

COSY correlation spectroscopy

CPMG Carr-Purcell-Meiboom-Gill

CPU central processing unit

CRAFT complete reduction to amplitude frequency table

CSR compressed sparse row

CUPID computer-assisted undiminished-sensitivity protocol for ideal decoupling

DECRA direct exponential curve resolution algorithm

DFT discrete Fourier transform

DMSO dimethyl sulfoxide, $(H_3C)_2SO$

DMSO-d_6 deuterated DMSO

DOSY diffusion-ordered spectroscopy

EYM Eckart-Young-Mirsky

FID free induction decay

FDM filter diagonalisation method

FFT fast Fourier transform

FS frequency-swept

FT Fourier transform

GN Gauss-Newton

GNAT general NMR analysis toolbox

GUI graphical user interface

HSQC heteronuclear single quantum coherence spectroscopy

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

MRI magnetic resonance imaging

MRS *in vivo* magnetic resonance spectroscopy

NLP non-linear programming

NMR nuclear magnetic resonance

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

PROJECT periodic refocusing of J-evolution by coherence transfer

PSYCHE pure shift yielded by chirp excitation

PyPI Python Package Index

RAM random access memory

RF radio-frequency

RSS residual sum-of-squares

SCORE speedy component resolution

SI Système International

SNR signal-to-noise ratio

ST Steihaug-Toint

SVD singular value decomposition

TSE-PSYCHE triple spin echo PSYCHE

TROSY transverse relaxation optimised spectroscopy

VARPRO variable projection

VE virtual echo

WURST wideband, uniform rate, smooth truncation

ZS Zangger-Sterk

Nomenclature

General Mathematics

$x := y$	x is by definition equal to y
$x \approx y$	x is approximately equal to y
$x \propto y$	x proportional to y
$\lfloor a \rceil$	The nearest integer to a, such that $\lfloor a \rceil \leq a$
$\lceil a \rceil$	The nearest integer to a, such that $\lceil a \rceil \geq a$
$\lfloor a \rceil$	The nearest integer to a
$a \bmod b$	a modulo b, given by $$a \bmod b = a - \left\lfloor \frac{a}{b} \right\rfloor$$

Sets

$x \in X$	x is a member of the set X
$X \subset Y$	The set X is a subset of the set Y
$\mathbb{C}$	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
$\mathbb{R}_{>1}$	The set of real numbers greater than one
$\mathbb{F}$	A field (i.e. $\mathbb{R}$, $\mathbb{C}$, etc.)
$\{x, \cdots, y\}$	The set of consecutive integers from $x \in \mathbb{Z}$ to $y \in \mathbb{Z}$ (both included)
(x, y)	The range of numbers from $x \in \mathbb{R}$ to $y \in \mathbb{R}$ (both excluded)
$[x, y]$	The range of numbers from $x \in \mathbb{R}$ to $y \in \mathbb{R}$ (both included).
$(x, y]$	The range of numbers from $x \in \mathbb{R}$ (excluded) to $y \in \mathbb{R}$ (included).

Complex Numbers

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

Vectors, Matrices, and Arrays

$\boldsymbol{x}, \boldsymbol{X}, x_n$	Vectors are denoted with bold italic lower-case letters. Matrices, arrays with > 2 dimensions, and generic arrays which may comprise any number of dimensions are denoted with bold italic capital letters. Individual elements of such objects are denoted by lower-case italic letters with subscripts indicating their index. E.g. The element located at the second row and third column of a matrix $\boldsymbol{Y}$ would be denoted $y_{2,3}$. In some circumstances where it is appropriate, indexing begins at 0 rather than 1.
$\boldsymbol{x}^{\mathrm{T}}, \boldsymbol{X}^{\mathrm{T}}$	Vector/matrix Transpose
$\boldsymbol{x}^{\dagger}, \boldsymbol{X}^{\dagger}$	Vector/matrix conjugate transpose
$\boldsymbol{X}^{-1}$	Inverse of a square, non-singular matrix.
$\boldsymbol{X}^{+}$	Moore-Penrose pseudo-inverse of a matrix.
$\boldsymbol{X} \odot \boldsymbol{Y}$	Hadamard (element-wise) product of the arrays $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{F}^{N_1 \times \cdots \times N_D}$, yielding the array $\boldsymbol{A} \in \mathbb{F}^{N_1 \times \cdots \times N_D}$, given by $$a_{n_1,\cdots,n_D} = x_{n_1,\cdots,n_D} y_{n_1,\cdots,n_D}.$$
$\boldsymbol{x} \times \boldsymbol{y}$	Cross product of two vectors $\boldsymbol{x} \in \mathbb{R}^3$ and $\boldsymbol{y} \in \mathbb{R}^3$, given by $$\boldsymbol{x} \times \boldsymbol{y} = \lVert \boldsymbol{x} \rVert \lVert \boldsymbol{y} \rVert \sin(\theta) \boldsymbol{n},$$ where θ is the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$, in the plane containing them, and $\boldsymbol{n}$ is the

unit vector perpendicular to said plane. N.B. the times symbol is occasionally used to denote scalar multiplication as well.

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

$$\langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \sum_{n_1=1}^{N_1} \cdots \sum_{n_D=1}^{N_D} x^*_{n_1,\cdots,n_D} y_{n_1,\cdots,n_D}$$

$\|\boldsymbol{X}\|$ Norm, given by

$$\|\boldsymbol{X}\| = \sqrt{\langle \boldsymbol{X}, \boldsymbol{X} \rangle}$$

$\boldsymbol{x} \cdot \boldsymbol{y}$ Dot product of $\boldsymbol{x} \in \mathbb{F}^N$ and $\boldsymbol{y} \in \mathbb{F}^N$, which generates a scalar a such that

$$a = \sum_{n=1}^{N} x_n y_n$$

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

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

$\mathrm{diag}(\boldsymbol{x})$ Given a vector $\boldsymbol{x} \in \mathbb{F}^N$, $\mathrm{diag}(\boldsymbol{x})$ is a matrix $\in \mathbb{F}^{N \times N}$ of the form

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

$\boldsymbol{x}[a : b]$ Slicing notation, following the convention used by NumPy*. For $\boldsymbol{x} \in \mathbb{F}^N$, $a \in \{0, \cdots, N-1\}$, and $b \in \{a+1, \cdots, N\}$, the slice $\boldsymbol{x}[a : b]$ produces a vector $\in \mathbb{F}^{b-a}$ comprising the consecutive elements of $\boldsymbol{x}$ from the $(a+1)^{\text{th}}$ to the b^{th}. This can be extended for an array of any dimension, e.g. $\boldsymbol{X}[a_r : b_r, a_c : b_c]$ denotes the slice of a 2D matrix, with the row slice denoted first, followed by the column slice.

$\boldsymbol{X}^{\circlearrowright(d)}$ Right circular rotation along axis d of an array by one element. As an example, for

*https://numpy.org/doc/stable/user/basics.indexing.html

a 2D matrix $\boldsymbol{X} \in \mathbb{F}^{M \times N}$, $\boldsymbol{X}^{\circlearrowleft(1)}$ 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 $\boldsymbol{X}^{\circlearrowleft(2)}$ is

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

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

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

Probability

$x \sim \mathcal{X}$ x behaves according to distribution $\mathcal{X}$

$x \perp\!\!\!\perp y$ x and y are conditionally independent

$\mathcal{N}(\mu, \sigma^2)$ Normal (Gaussian) distribution with mean μ and variance σ^2:

$$\mathcal{N}\left(x \,|\, \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$\mathcal{N}_{\mathbb{C}}(\mu, \sigma^2)$ Complex normal distribution with mean μ and variance $\sigma^2/2$

$\mathcal{U}(l, r)$ Uniform distribution with bounds l and r:

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

Calculus

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

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

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

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

List of Figures

List of Tables

List of Algorithms

List of Code Listings

Chapter 1

Preliminaries

This thesis is principally focussed on extracting quantitative information from nuclear magnetic resonance (NMR) data. In this chapter, concepts are introduced to contextualise the work undertaken. 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 and analysed to gain chemical insights is discussed. Techniques used to estimate the parameters which describe NMR data based on a model of summed complex sinusoids are an alternative means of analysis. While able to afford richer information than conventional approaches, these are not widely employed, despite many effective methods existing; a review of some of the most prominent methods ones is given. Finally, the motivation for carrying out the work in this thesis is discussed, and an overview of the remainder of its contents is provided.

1.1 An Introduction to NMR

1.1.1 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 chemical structure and dynamics that 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 paraffin [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 development of new methods for nuclear magnetic precision measurements and discoveries in connection therewith" [4]. A notable mention should also be given to Yevgeny Zavoisky, the father of the related field of electron paramagnetic resonance, who

probably observed NMR as far back as 1941 [5]. Alas, he dismissed his results as irreproducible. A few years subsequently, works investigating NMR spectra from compounds containing nuclei such as ^{63}Cu, ^{65}Cu, ^{31}P, ^{14}N, and ^{19}F led to an understanding of the *chemical shift*, a phenomenon in which nuclei in different chemical environments exhibit non-identical resonant frequencies [6, 7, 8]. Chemists regarded these findings with great interest, as they suggested that NMR could give insights into molecular structure.

Russell Varian — founder of Varian Associates along with his brother Sigurd — secured the first patent for a commercial NMR machine, with a 30 MHz spectrometer following soon after. The first spectrometers functioned by gradually varying the magnetic field strength, 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 at the time, proposed an alternative method: pulsed Fourier transform (FT) spectroscopy [9]. 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 [10]. Instead, the first commercial pulsed FT spectrometer was produced by Bruker Corp. in 1969, which revolutionised NMR. The innovation emerged shortly after Cooley and Tukey's introduction of the fast Fourier transform (FFT) algorithm [11], which incentivised the development of the pulsed FT approach.

The concept of two-dimensional (2D) NMR spectroscopy was proposed by Jean Jeener in 1971 [12, 13], which Ernst and co-workers showcased a few years later in the form of a correlation spectroscopy (COSY) experiment [14]. The use of multiple dimensions to spread out signals enabled vastly more complex structures to be studied. In 1985, the first protein assignment by NMR — using COSY and nuclear Overhauser effect spectroscopy (NOESY) experiments — was reported by Kurt Wüthrich and co-workers [15]. Over time, extensive developments in techniques for biomolecular systems have occurred, including the creation of 3D and 4D "triple resonance" experiments [16, 17], as well transverse relaxation optimised spectroscopy (TROSY) experiments [18] 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, on top of the 1952 Physics Prize. First, Ernst received the prize in 1991 "for his contributions to the development of the methodology of high resolution nuclear magnetic resonance" [19]. In 2002, Wüthrich was recognised "for his development of nuclear magnetic resonance for determining the three-dimensional structure of biological macromolecules in solution" [20].

1.1.2 The Bloch Model

NMR relies on an intrinsic property of nuclei with an odd number of protons and/or neutrons called *spin* which, along with orbital angular momentum and molecular rotation, is one of the sources of angular momentum in quantum mechanics. The angular momentum associated with a nuclear spin is characterised by the quantum number $I \in \{0, 1/2, 1, 3/2, \cdots\}$. 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 on the subject available to suit newcomers and experts alike; notable examples include the following citations (in approximate order of complexity): [21, 22, 23, 24, 25, 26]. Despite this, a basic appreciation can be gained using the Bloch (vector) model, a semi-classical description of the simplest possible system to study: a ensemble of isolated, identical spin-1/2 nuclei [21: Chapter 1]. The Bloch model becomes inadequate when more complex spin systems are considered, featuring non-identical spins which interact through mechanisms such as scalar couplings (commonly referred to as J-couplings), dipolar couplings etc. However, it provides valuable insights into the basic principles of NMR, including the form that a typical dataset acquired by an experiment takes.

The nuclear spin angular momentum $\boldsymbol{I} \in \mathbb{R}^3$ is a vector* with squared magnitude

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

where $\hbar := h/2\pi$ is the reduced Planck constant (1.055×10^{-34} J s). While it is not possible to specify multiple components of the angular momentum simultaneously in accordance with the uncertainty principle, it is possible to specify one of these along with $\boldsymbol{I}^2$. Conventionally, this is chosen to be the z-component, for which

$$I_z = \hbar m \quad \forall m \in \{-I, -I+1, \cdots, I-1, I\}. \tag{1.2}$$

Equation 1.2 implies that the magnitude of the z-component may only adopt certain discrete values (i.e. it is quantised). A nucleus with spin has an associated *magnetic moment*, given by†:

$$\boldsymbol{\mu} = \gamma \boldsymbol{I} \implies \mu_z = \gamma I_z = \gamma \hbar m. \tag{1.3}$$

$\gamma \in \mathbb{R}$ is a proportionality constant called the *gyromagnetic ratio*, which is dependent on the nu-

*In most of this work, vectors are expressed as lower-case bold letters and matrices/multidimensional arrays are expressed as upper-case bold letters. However, in this section notation frequently encountered in the literature is used, which violates this.

†In the context of solution-state NMR, γ is often replaced with $(1-\sigma)\gamma$, where σ is a unitless quantity that accounts for chemical shift.

Nucleus	I	$\gamma(\mathrm{T^{-1}\,s^{-1}})$	Relative Abundance (%)
^{1}H	1/2	2.6752×10^{8}	99.9885
^{2}H	1	4.1066×10^{7}	0.0115
^{6}Li	1	3.9371×10^{7}	7.59
^{7}Li	3/2	1.0398×10^{8}	92.41
^{12}C	0	-	98.93
^{13}C	1/2	6.7279×10^{7}	1.07
^{14}N	1	1.9329×10^{7}	99.636
^{15}N	1/2	-2.7114×10^{7}	0.364
^{16}O	0	-	99.756
^{17}O	5/2	-3.6276×10^{7}	0.038
^{19}F	1/2	2.5176×10^{8}	100
^{31}P	1/2	1.0833×10^{8}	100

Table 1.1: Statistics related to a number of nuclei which are regularly-encountered in NMR. Also listed are common nuclei which do not possess spin. The gyromagnetic ratios were determined by obtaining the relevant nuclear magnetic dipole moments μ in units of nuclear magneton μ_N [27], and applying the equation $\gamma = \mu\mu_\mathrm{N}/I\hbar$, where $\mu_\mathrm{N} = 5.050\,783\,746\,1 \times 10^{-27}\ \mathrm{J\,T^{-1}}$ [28]).

cleus of interest. Table 1.1 provides the gyromagnetic ratios for some low-mass nuclei commonly encountered in NMR, along with some which do not possess spin and are therefore inert in the context of NMR.

Without the presence of an external magnetic field, the different nuclear spin states are degenerate. However, once a magnetic field is applied, the *Zeeman effect* is observed, whereby the relative energies of the different states diverge. The energy of a given magnetic moment relative to its zero-field energy is given by

$$E = -\boldsymbol{\mu} \cdot \boldsymbol{B}_0, \tag{1.4}$$

where $\boldsymbol{B}_0 \in \mathbb{R}^3$ 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 (Figure 1.1)

$$E_m = -\gamma I_z B_0 = -m\hbar\gamma B_0. \tag{1.5}$$

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 — albeit to a meager extent due to the small relative energies involved — 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 with $\gamma > 0$, such as ^{1}H, will have a more populated $m = +1/2$ (α) state, relative to the $m = -1/2$ (β) state. Due to the

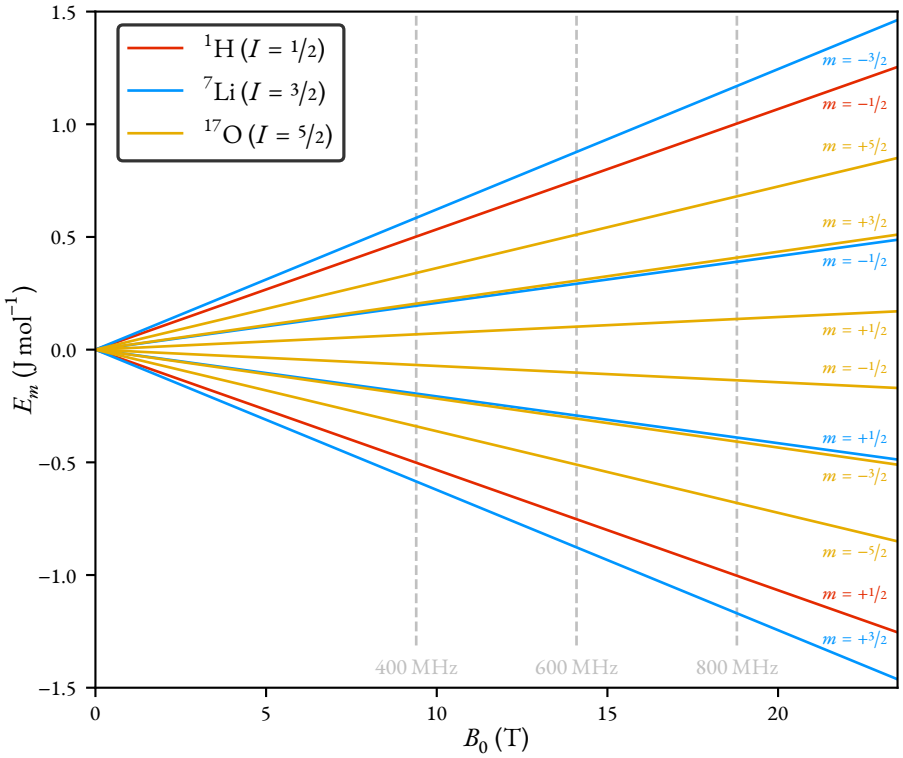


Figure 1.1: The variation of energy of the spin states of ^{1}H, ^{7}Li, and ^{17}O with external magnetic field strength (B_0) up to 23.5 T, which is approximately the strength of a 1 GHz NMR magnet. Three common field strengths for commercial NMR magnets are indicated: 9.40 T (400 MHz), 14.10 T (600 MHz), and 18.79 T (800 MHz). Note the relative ordering of the spin states for ^{1}H and ^{7}Li (with positive γ) versus ^{17}O (with negative γ).

population imbalance, the ensemble acquires a net (bulk) magnetic moment $\boldsymbol{M}$, given by the sum of all the individual spin moments:

$$\boldsymbol{M} = \sum_{s=1}^{S} \boldsymbol{\mu}_s, \tag{1.6}$$

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:

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

such that the bulk magnetisation is collinear with the field direction, with a magnitude M_0. By invoking the *high temperature approximation*[‡], M_0 may be expressed as [23: Section 1.1]:

$$M_0 \approx \frac{S\gamma^2\hbar^2 B_0 I(I+1)}{3k_{\mathrm{B}}T}, \tag{1.8}$$

where k_{B} is the Boltzmann constant ($1.381 \times 10^{-23}\,\mathrm{J\,K^{-1}}$), and T is the sample temperature. To maximise experiment sensitivity, it is desirable to utilise nuclei with high natural abundance (affecting S for a given sample concentration), and high gyromagnetic ratio. Along with other favourable attributes such as its ubiquity in organic molecules, ^{1}H is therefore by far the most popular nucleus to study using NMR, at least in solution-state contexts.

The bulk magnetism experiences a torque induced by the magnetic field, with its evolution described by

$$\frac{\mathrm{d}\boldsymbol{M}(t)}{\mathrm{d}t} = \boldsymbol{M}(t) \times \gamma \boldsymbol{B}(t). \tag{1.9}$$

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

$$\boldsymbol{B}(t) = \boldsymbol{B}_0 + \boldsymbol{B}_1(t). \tag{1.10}$$

Whenever the magnetisation vector is not collinear with the field vector[§], it undergoes *precession* about the field vector. Free precession occurs when the magnetisation is aligned away from the

[‡]Equation 1.8 is arrived at under the assumption that $m\hbar\gamma B_0/k_{\mathrm{B}}T \ll 1$. Given typical values of γ ($\approx 10^8\,\mathrm{T^{-1}\,s^{-1}}$) and B_0 ($\approx 10\,\mathrm{T}$), the implicit requirement for the temperature to be sufficiently large is virtually always adhered to in solution-state NMR.

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

z-axis and $\boldsymbol{B}_1 = \mathbf{0}$, such that

$$\frac{\mathrm{d}\boldsymbol{M}(t)}{\mathrm{d}t} = -\gamma B_0 \begin{bmatrix} M_y \\ -M_x \\ 0 \end{bmatrix} \tag{1.11}$$

Under free precession, the magnetisation rotates about the z-axis at the *Larmor frequency* $\omega_0 = -\gamma B_0$.[¶]

NMR experiments typically employ RF pulses which are linearly polarised and modulated cosinusoidally. A pulse which is polarised along the laboratory x-axis takes the form

$$\boldsymbol{B}_1(t) = 2B_1 \cos(\omega_{\mathrm{RF}} t + \phi_{\mathrm{RF}})\boldsymbol{i}, \tag{1.12}$$

where B_1 is the strength of the RF field (the presence of the factor of 2 will become clear shortly), ω_{RF} is its angular frequency, and ϕ_{RF} is its phase. $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$ (encountered shortly) denote the unit vectors along the laboratory x-, y-, and z-axes, respectively. Such a pulse can instead be thought of as the superposition of two circular fields, rotating with opposite senses:

$$\begin{aligned} \boldsymbol{B}_1(t) =& B_1(\cos(\omega_{\mathrm{RF}} t + \phi_{\mathrm{RF}})\boldsymbol{i} + \sin(\omega_{\mathrm{RF}} t + \phi_{\mathrm{RF}})\boldsymbol{j}) \\ &+ B_1(\cos(\omega_{\mathrm{RF}} t + \phi_{\mathrm{RF}})\boldsymbol{i} - \sin(\omega_{\mathrm{RF}} t + \phi_{\mathrm{RF}})\boldsymbol{j}). \end{aligned} \tag{1.13}$$

Assuming that $|\omega_{\mathrm{RF}}|$ and $|\gamma B_0|$ are close in value, the circular component which rotates with the same sense as the spin magnetisation is said to be *on resonance*. The component which rotates with the opposite sense has a negligible effect on the magnetisation in most circumstances. An RF pulse may therefore be expressed to a high degree of accuracy as

$$\boldsymbol{B}_1(t) = B_1 \left(\cos(\omega_{\mathrm{RF}} t + \phi_{\mathrm{RF}})\boldsymbol{i} + \sin(\omega_{\mathrm{RF}} t + \phi_{\mathrm{RF}})\boldsymbol{j}\right), \tag{1.14}$$

A great simplification to the model of magnetisation evolution 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

$$\frac{\mathrm{d}\tilde{\boldsymbol{M}}(t)}{\mathrm{d}t} = \tilde{\boldsymbol{M}}(t) \times \gamma \tilde{\boldsymbol{B}}(t), \tag{1.15a}$$

$$\tilde{\boldsymbol{B}}(t) = B_1 \cos(\phi_{\mathrm{RF}})\tilde{\boldsymbol{i}} + B_1 \sin(\phi_{\mathrm{RF}})\tilde{\boldsymbol{j}} + \Delta B_0 \tilde{\boldsymbol{k}}, \tag{1.15b}$$

[¶]While the Système International (SI) 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 Larmor frequency of a reference ^{1}H nucleus at the given field strength. For example, a 500 MHz spectrometer operates at a field strength of $5{\times}10^8\ \mathrm{Hz} / 4.2577{\times}10^7\ \mathrm{T^{-1}\ Hz} \approx 11.74\ \mathrm{T}$.

$$\Delta B_0 = -\frac{\Omega}{\gamma}, \tag{1.15c}$$

$$\Omega = -\gamma B_0 - \omega_{\mathrm{RF}}, \tag{1.15d}$$

$$\tilde{\boldsymbol{i}} = \cos(\omega_{\mathrm{RF}} t)\boldsymbol{i} + \sin(\omega_{\mathrm{RF}} t)\boldsymbol{j}, \tag{1.15e}$$

$$\tilde{\boldsymbol{j}} = \cos(\omega_{\mathrm{RF}} t)\boldsymbol{j} - \sin(\omega_{\mathrm{RF}} t)\boldsymbol{i}, \tag{1.15f}$$

$$\tilde{\boldsymbol{k}} = \boldsymbol{k} \tag{1.15g}$$

The tilde has been used to distinguish quantities in the rotating frame from their laboratory frame counterparts. $\Omega := \omega_0 - \omega_{\mathrm{RF}}$ is the *offset* of the spin magnetisation. When $\Omega = 0\,\mathrm{rad\,s^{-1}}$, the RF field is perfectly on resonance, and at times when it 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 1.15 implies that $\tilde{\boldsymbol{M}}$ 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, $\boldsymbol{M}_{\mathrm{eq}} \equiv \tilde{\boldsymbol{M}}_{\mathrm{eq}} = [0, 0, M_0]^{\mathrm{T}}$. The process by which this occurs is called *relaxation*. Spin relaxation is driven by the stochastic motion of the molecules in the sample, which give rise to variations in the magnetic fields present in the sample. Crucially:

1. The fields induced by molecular motion vary with time, and average to zero.
2. In different locations within the sample, the variation in these fields with time is uncorrelated. As a result, these fields are often referred to as *local* fields.

The presence of time-varying local fields in the sample lead to a number of mechanisms that contribute to relaxation. These mechanisms are often classified based on whether they are associated with an energy transfer:

- *Adiabatic interactions* do not require a transfer of energy between a spin system of interest and its surroundings[‖]. Molecular motion results in the net field in the z-direction experienced by a given spin to vary with time. The associated Larmor frequency of the spin is perturbed in accordance with this field variation. On a wider scale, spins in different locations, experiencing differing local z-field fluctuations, will gradually lose synchronisation with each other (i.e. become *dephased*) as their instantaneous Larmor frequencies are non-equivalent. The effect of this is for the transverse component of the bulk magnetisation to decay. Hence, adiabatic interactions contribute to *transverse relaxation*, also referred to as *spin-spin relaxation*.

[‖]The surroundings of a particular spin system of interest is often referred to as the *lattice*, which contains all degrees of freedom in the sample other than those of the spin system itself, including rotational degrees of freedom of all molecules in the system, and the spin energy levels of all other molecules.

- *Nonadiabatic interactions* involve the transfer of energy between a spin and its surroundings. Molecular motion which leads to varying transverse ($xy-$) magnetisation with a component that fluctuates at the spin's Larmor frequency is able to induce such an energy transfer. This mechanism drives the relative populations of the spin's energy levels towards their equilibrium configuration, restoring the z-component of the bulk magnetisation. As such, nonadiabatic interactions contribute towards *longitudinal relaxation*, also referred to as *spin-lattice relaxation*. Furthermore, due to the finite lifetimes of the spin states, there is an uncertainty in their associated energies in accordance with the Heisenberg uncertainty principle. Because of this, nonadiabatic interactions also contribute to the dephasing of spins, and hence transverse relaxation.

In the Bloch model, the influence of longitudinal and transverse relaxation are accounted for by the incorporation of two processes with respective rate constants R_1 and R_2 (both with units of s^{-1}). The processes are often described in terms of relaxation times instead: $T_i := 1/R_i, i \in \{1, 2\}$. In the vast majority of situations, the rate of transverse relaxation is greater than that of longitudinal relaxation, i.e. $T_2 \leq T_1$ [22: Section 11.9]. The rate of longitudinal relaxation places a strict limit on the rate of transverse relaxation: $T_2 \leq 2T_1$; if this were violated, during the evolution of the bulk magnetisation vector, its magnitude would surpass its magnitude at equilibrium, which is a physical impossibility [29].

The introduction of relaxation is phenomenological in the Bloch model; the reversion of the spin system to equilibrium is included purely to ensure the model agrees with observation. More sophisticated theories invoking Liouville-space quantum mechanics can account for relaxation however [23: Chapter 5, 26: Chapter 6, 30, 31].

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:

$$\frac{\mathrm{d}\tilde{\boldsymbol{M}}(t)}{\mathrm{d}t} = \begin{bmatrix} -R_2 & -\Omega & -\gamma B_1 \sin(\phi_{\mathrm{RF}}) \\ \Omega & -R_2 & \gamma B_1 \cos(\phi_{\mathrm{RF}}) \\ \gamma B_1 \sin(\phi_{\mathrm{RF}}) & -\gamma B_1 \cos(\phi_{\mathrm{RF}}) & -R_1 \end{bmatrix} \tilde{\boldsymbol{M}}(t) + R_1 M_0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{1.16}$$

Figure 1.2.a depicts the evolution of a bulk magnetisation vector after the application of an RF pulse with $\phi_{\mathrm{RF}} = \pi/2$, and an appropriate combination of duration and power to induce a clockwise rotation of 90° about the y-axis; such a pulse is denoted $90°_y$. 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

$$\tilde{M}_x(t) = M_0 \cos(\Omega t) \exp(-R_2 t), \tag{1.17a}$$

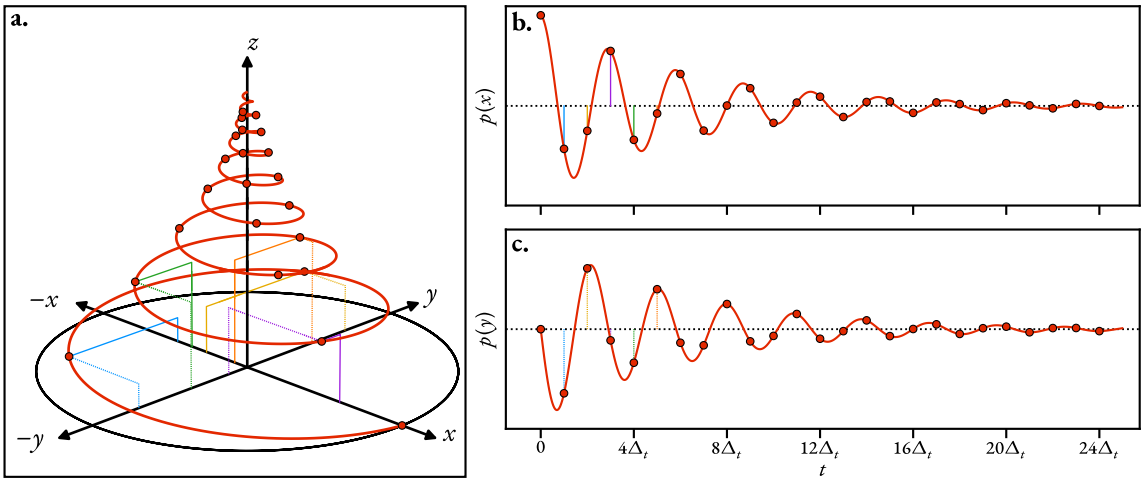


Figure 1.2: a. An illustration of the free evolution of the bulk magnetisation of an ensemble of spin-1/2 nuclei immediately after the application of a 90°$_y$ pulse according to the Bloch model. The rate of transverse relaxation was set to be double that of longitudinal relaxation ($T_2 = T_1/2$). The 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 are sampled at regular time intervals, separated by Δ_t. The resulting FID is given by the complex value $p(x)+\mathrm{i}p(y)$.

$$\tilde{M}_y(t) = M_0 \sin(\Omega t) \exp(-R_2 t), \tag{1.17b}$$

$$\tilde{M}_z(t) = M_0(1 - \exp(-R_1 t)), \tag{1.17c}$$

with $t = 0$ s denoting the time that the magnetisation lands on the x-axis. During acquisition, the transverse components of the bulk magnetisation are detected by the spectrometer probe circuitry (Figures 1.2.b and 1.2.c), such that the resulting signal, called the *free induction decay* (FID), is given by

$$y(t) = c\tilde{M}_+(t), \tag{1.18a}$$

$$\tilde{M}_+(t) = \tilde{M}_x(t) + \mathrm{i}\tilde{M}_y(t) = M_0 \exp(\mathrm{i}\Omega t - R_2 t), \tag{1.18b}$$

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

1.1.3 The NMR Spectrometer

Modern NMR spectrometers are capable of conducting a plethora of experiments which can aide chemists. 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 $\boldsymbol{B}_0$ field is generated by a magnet which is composed of a superconducting solenoid immersed in liquid helium; common materials used for the solenoid include Nb-Ti alloy and Nb_3Sn. 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, which is finally encased in a vacuum chamber. A bore passes through the z-direction of the magnet, which is maintained at a user-specified 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 signal-to-noise ratio (SNR) ($\propto B_0^{3/2}$) of the data are affected. At the time of writing, commercial spectrometers which operate at and above a ^{1}H Larmor frequency of 1 GHz (23.5 T) exist, though these are uncommon and are employed primarily for the study of large biomolecules and solid-state NMR. For most applications, including the study of small molecules, spectrometers with more modest field strengths on the order of 100 MHz are typically adequate. Due to their cheap operating costs and small size, "benchtop" NMR spectrometers, which comprise permanent magnets and typically operate on the order of 10 MHz, have also become popular in educational and high-throughput settings [32].

To ensure a 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 a specific spatial profile in accordance with a spherical harmonic function; a collection of shims can cancel out small inhomogeneities inherent to the main magnet. A field-frequency lock is used to ensure the stability of the field. The lock is effectively a small NMR spectrometer, tuned to a specified isotope (typically ^{2}H**), which monitors the resonance frequency of the isotope over time. If the frequency begins to drift, the current in the Z_0 coil is appropriately adjusted, which induces a constant change in field strength throughout the sample volume.

The Probe

The probe sits inside the bore of the magnet and has a number of responsibilities including holding the sample, regulating the sample temperature, and housing the coils used to pulse the sample with RF radiation as well as coils which are used to generate field gradients[22: Section 4.7]. The RF coils also receive the response from the sample during detection. The principle source of data corruption in NMR experiments is thermal noise within the probe circuitry. For this reason, cryogenic probes have become a popular development, in which the coils and other probe electronics are maintained at a very low temperature (typically about 20 K) [33].

**^{2}H-enriched solvents are routinely used to make up NMR samples. In ^{1}H 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 ^{2}H a suitable nucleus to monitor by the lock, as it present in high concentrations, but rarely directly studied.

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. Attenuators/amplifiers then adjust the power of the pulse, which travels to the probe.

The Receiver

During detection, the time-varying current induced in the probe coil by the sample magnetisation travels to a receiver, which comprises a series of components designed to convert 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* [34: Section 13.6], which ensures FIDs are frequency discriminated, i.e. that they possess the requisite information to determine whether a given component in the FID has a frequency that is above or below the transmitter frequency. This is achieved by splitting the signal from the probe into two channels. In each channel, the signal, which is of a very high frequency (MHz), is mixed with a reference signal of frequency ω_{RF}. The mixing process results in a low-frequency (kHz) signal being generated, along with a very high frequency signal. The reference signal in one channel possesses a phase which is shifted by 90° relative to the other, such that the combined signal constitutes a quadrature pair. Both 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 constitutes the FID.

1.1.4 The Structure of the FID

The result of running a one-dimensional (1D) NMR experiment is an FID $\boldsymbol{y} \in \mathbb{C}^N$ which is sampled at equally spaced points in time, with consecutive samples separated by time Δ_t:

$$\begin{aligned} \boldsymbol{y} &= \begin{bmatrix} y_0 & y_1 & y_2 & \cdots & y_{N-1} \end{bmatrix}^{\mathrm{T}} \\ &\equiv \begin{bmatrix} y(t=0) & y(t=\Delta_t) & y(t=2\Delta_t) & \cdots & y(t=(N-1)\Delta_t) \end{bmatrix}^{\mathrm{T}}, \end{aligned} \tag{1.19}$$

where $y(t)$ is the (continuous) variation of the generated signal as a function of time, and N (often a power of 2) is the number of points sampled. The inverse of the sampling rate, $1/\Delta_t$ is the *spectral width* f_{sw} which defines how wide the range of samplable frequencies is, in accordance with the

Nyquist theorem [35].

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 therefore takes the form[††]

$$y_n = x_n(\boldsymbol{\theta}) + w_n \qquad \forall n \in \{0, 1, \cdots, N-1\}, \tag{1.20a}$$

$$x_n(\boldsymbol{\theta}) = \sum_{m=1}^{M} a_m \exp(\mathrm{i}\phi_m) \exp\left((2\pi\mathrm{i}(f_m - f_{\mathrm{off}}) - \eta_m) n\Delta_t\right). \tag{1.20b}$$

Equation 1.20 indicates that an FID comprises contributions from the (deterministic) evolution of the spin magnetisation $\boldsymbol{x}$ and experimental noise $\boldsymbol{w}$ (*vide infra*). Each signal which contributes to $\boldsymbol{x}$ is defined by four parameters:

- Amplitude $a \in \mathbb{R}_{>0}$,
- Phase $\phi \in (-\pi, \pi]$ (rad),
- Frequency $f \in [f_{\mathrm{off}} - {}^1\!/_2 f_{\mathrm{sw}}, f_{\mathrm{off}} + {}^1\!/_2 f_{\mathrm{sw}}]$ (Hz),
- Damping factor $\eta \in \mathbb{R}_{>0}$ (s^{-1}).

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

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{a}^{\mathrm{T}} & \boldsymbol{\phi}^{\mathrm{T}} & \boldsymbol{f}^{\mathrm{T}} & \boldsymbol{\eta}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{1.21}$$

where $\boldsymbol{a} \in \mathbb{R}^M = [a_1\, a_2 \cdots a_M]^{\mathrm{T}}$ is a vector of all amplitudes, $\boldsymbol{\phi} \in \mathbb{R}^M$ is a vector of all phases, etc. A more concise alternative to Equation 1.20b involves the *complex amplitudes* and *signal poles* associated with the FID:

$$x_n(\boldsymbol{\theta}) = \sum_{m=1}^{M} \alpha_m z_m^n, \tag{1.22a}$$

$$\alpha_m = a_m \exp(\mathrm{i}\phi_m), \tag{1.22b}$$

$$z_m = \exp\left((2\pi\mathrm{i}\,(f_m - f_{\mathrm{off}}) - \eta_m)\,\Delta_t\right). \tag{1.22c}$$

The respective influences of the four parameters on a signal in the time-domain are depicted in Figures 1.3.a1 to 1.3.d1.

Multidimensional experiments involve incrementing one or more delays within the pulse sequence,

[††]This provides an idealised model 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 having Lorentzian lineshapes (Section 1.2.1). In cases where distortions to the data have occurred, techniques such as *reference deconvolution* [36] can be used as a corrective measure in a bid to make the data agree more closely with this model.

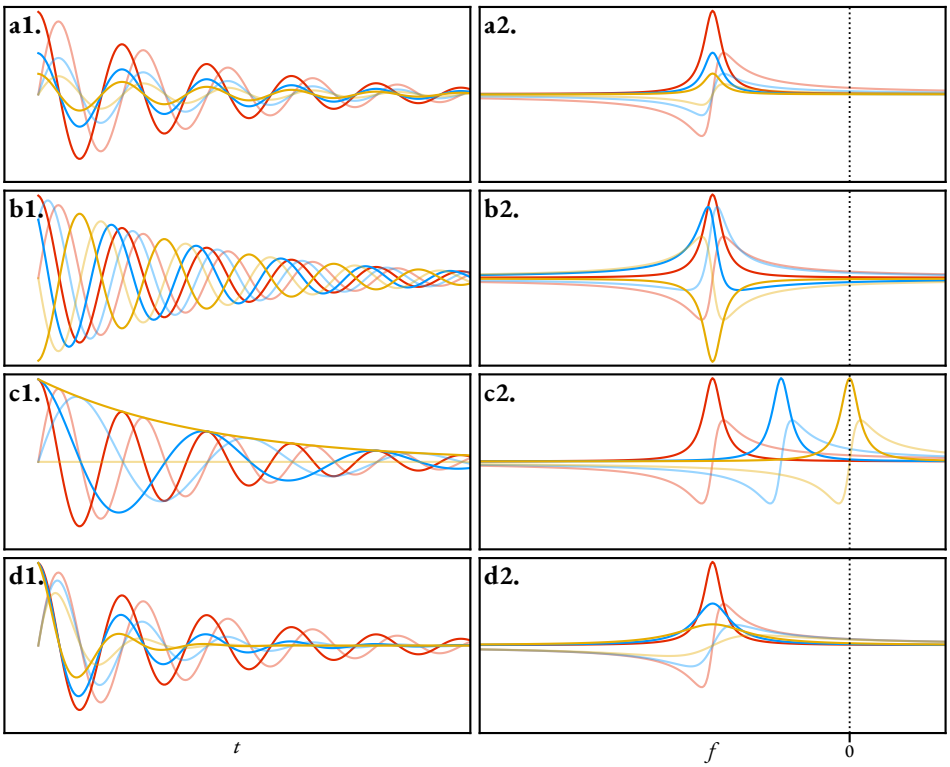


Figure 1.3: An illustration of the influence of the four parameters associated with a signal in both the time-domain (a1 to d1) and Fourier-domain (a2 to d2). The red signal is generated with the same parameters across all panels: $a = a_{\text{red}}$, $\phi = 0\,\text{rad}$, $f = f_{\text{red}}$, $\eta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

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 $\boldsymbol{\theta} \in \mathbb{R}^{2(1+D)M}$ is given by

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{a}^{\mathrm{T}} & \boldsymbol{\phi}^{\mathrm{T}} & \boldsymbol{f}^{(1)\mathrm{T}} & \cdots & \boldsymbol{f}^{(D)\mathrm{T}} & \boldsymbol{\eta}^{(1)\mathrm{T}} & \cdots & \boldsymbol{\eta}^{(D)\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{1.23}$$

where $\boldsymbol{f}^{(D)}$ and $\boldsymbol{\eta}^{(D)}$ are the frequencies and damping factors in the actively acquired (direct) dimension, and $\{\boldsymbol{f}^{(1)}, \cdots, \boldsymbol{f}^{(D-1)}\}$ and $\{\boldsymbol{\eta}^{(1)}, \cdots, \boldsymbol{\eta}^{(D-1)}\}$ are those for the indirect dimension(s). Indirect dimensions can exhibit different forms of evolution, depending on the precise nature of the pulse sequence, of which two are very common [23: Section 4.3.4]. FIDs whose constituent signals evolve according to $\cos(2\pi f t)$ or $\sin(2\pi f t)$ modulate the amplitude of the direct dimension across increments, while those whose signals evolve according to $\exp(2\pi \mathrm{i} f t)$ and $\exp(-2\pi \mathrm{i} f t)$ modulate the phase instead. For experiments which produce amplitude-modulated FIDs, both the cosine and sine forms should be acquired if possible, as this ensures that spectra with desirable properties can be generated. The same is true for the positive and negative forms when phase-modulated FIDs are acquired (*vide infra*). In general, a D-dimensional FID $\boldsymbol{Y} \in \mathbb{C}^{N^{(1)} \times \cdots \times N^{(D)}}$ can be expressed as

$$y_{n^{(1)},\cdots,n^{(D)}} = x_{n^{(1)},\cdots,n^{(D)}}(\boldsymbol{\theta}) + w_{n^{(1)},\cdots,n^{(D)}}, \tag{1.24a}$$

$$x_{n^{(1)},\cdots,n^{(D)}} = \sum_{m=1}^{M} a_m \exp(\mathrm{i}\phi_m) \prod_{d=1}^{D} \zeta^{(d)}\left(2\pi\left(f_m^{(d)} - f_{\mathrm{off}}^{(d)}\right) n^{(d)} \Delta_t^{(d)}\right) \exp\left(-\eta_m^{(d)} n^{(d)} \Delta_t^{(d)}\right), \tag{1.24b}$$

$$\zeta^{(d)}(\cdot) \begin{cases} = \exp(\mathrm{i}\cdot) & d = D \\ \in \{\cos(\cdot), \sin(\cdot), \exp(\mathrm{i}\cdot)\exp(-\mathrm{i}\cdot)\} & \text{otherwise} \end{cases}, \tag{1.24c}$$

It is typical to assume that the data is corrupted by 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:

$$w_{n^{(1)},\cdots,n^{(D)}} \sim \mathcal{NC}\left(0, 2\sigma^2\right) \tag{1.25a}$$

$$\implies \Re\left(w_{n^{(1)},\cdots,n^{(D)}}\right) \perp\!\!\!\perp \Im\left(w_{n^{(1)},\cdots,n^{(D)}}\right),$$

$$\Re\left(w_{n^{(1)},\cdots,n^{(D)}}\right) \sim \mathcal{N}\left(0, \sigma^2\right), \tag{1.25b}$$

$$\Im\left(w_{n^{(1)},\cdots,n^{(D)}}\right) \sim \mathcal{N}\left(0, \sigma^2\right).$$

The extent by which an FID is corrupted by noise is given by its SNR, the ratio of signal power

and noise power:

$$\mathrm{SNR}\left(\boldsymbol{Y}\right) \coloneqq \frac{1}{2N_{\mathrm{tot}}\sigma^2} \sum_{n^{(1)}=0}^{N^{(1)}-1} \cdots \sum_{n^{(d)}=0}^{N^{(D)}-1} \left| x_{n^{(1)},\cdots,n^{(D)}} \right|^2, \tag{1.26}$$

where $N_{\mathrm{tot}} \coloneqq N^{(1)} \times \cdots \times N^{(D)}$ is the total number of points the FID comprises. Due to the large dynamic range observed for the SNR across datasets, it is common to express it using a logarithmic scale instead, in units of decibels (dB):

$$\mathrm{SNR}_{\mathrm{dB}} \coloneqq 10 \log_{10}\left(\mathrm{SNR}\right). \tag{1.27}$$

1.2 NMR Data Processing and Analysis

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

1.2.1 Processing NMR Data

The typical means of processing NMR data is to transform the time (measured) domain FID into the frequency domain, yielding an *NMR spectrum*. This is achieved through application of the FT. The FT converts a continuous function over time t, to one over frequency F as follows[‡‡]:

$$s(F) = \int_0^\infty x(t) \exp(-2\pi \mathrm{i} t F) \mathrm{d}t \tag{1.28}$$

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

$$s_n = \sum_{k=0}^{N-1} x_k \exp\left(-\frac{2\pi \mathrm{i} k n}{N}\right) \quad \forall n \in \{0, \cdots, N-1\}. \tag{1.29}$$

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

$$s(F) = \int_0^\infty a \exp(\mathrm{i}\phi) \exp((2\pi \mathrm{i} f - \eta) t) \exp(-2\pi \mathrm{i} t F) \mathrm{d}t, \tag{1.30a}$$

$$s(F) = \frac{a \exp(\mathrm{i}\phi)}{\eta + 2\pi \mathrm{i}(f - F)}. \tag{1.30b}$$

When processing discrete data, the DFT of an FID features a vertical offset, such that the baseline does not sit at 0. 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

[‡‡]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 $[0, \infty)$ instead.

easily by halving the initial point of the FID prior to DFT [37]. The FT is a linear function, such that the FT of a summation of signals is equivalent to the summation of all the individual FTs of the signals. A corollary is that an NMR spectrum comprises a series of Lorentzian "peaks", located at the frequencies of the signals in the FID:

$$s(F) = \sum_{m=1}^{M} \frac{a_m \exp(\mathrm{i}\phi_m)}{\eta_m + 2\pi\mathrm{i}(f_m - F)}. \tag{1.31}$$

The FT is a very attractive means of processing NMR data, as it presents the data in a format which is human-interpretable; the basic rules describing how chemical structure is mapped to NMR spectra is a fundamental skill that experimentalists in many fields require [38]. Due to innate properties of the NMR experiment, as well as issues arising from analysing discrete data, the FT of the raw FID has undesirable characteristics without further manipulation. Additional processing steps that are frequently applied to NMR data are now outlined, in the usual order that they are employed. Apodisation and zero filling are applied prior to FT (i.e. to the FID), while phase correction and baseline correction are applied to the resultant spectrum.

Apodisation

Apodisation refers to the process of mutating a signal by multiplying it with a specified function, often called a *window function* [39: Section 3.2.7], as a means of improving the sensitivity or the resolution of the final spectrum.

Sensitivity Enhancement As an FID progresses with time, the contributions from the desirable signal ($\boldsymbol{x}$) and experimental noise ($\boldsymbol{w}$) becomes more weighted towards the noise, as spin relaxation phenomena attenuate the signal. Multiplying the FID with a function which smoothly decreases in value with time can be used to enhance the SNR of the FID, as later points which are heavily noise-weighted are suppressed. While not the optimal choice[§§], the most common function used 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 damping applied to the FID causes the linewidth of the spectral peaks to increase (see Figure 1.3.d).

Resolution Enhancement By contrast, resolution enhancement can be achieved by applying a window function that artificially reduces the rate of signal decay. It is possible to do this by attenuating the initial datapoints in an FID. The Lorentz-Gauss function bestows a Gaussian shape, featuring a narrower base relative to an equivalent Lorentzian, to spectral peaks. Another popular window function for resolution enhancement is the sine-bell, which is commonly used in mul-

[§§] The theoretically optimal window function is the Dolph-Chebycheff window, which reduces the magnitude of truncation ripples for a given degree of sensitivity enhancement[23: Section 3.3.2.2]. This is rarely used in NMR however due to its complex mathematical form.

tidimensional experiments. Since the initial points in the FID are reduced in magnitude, these window functions decrease the sensitivity of the resulting spectra.

As they are defined to decay to a small value or zero, window functions are also able to suppress artefacts which manifest in the spectra of FIDs that possess appreciable signal amplitude at the end of the acquisition period; such FIDs are often referred to as *truncated*. The spectrum of a truncated FID is akin to the convolution of spectrum of its untruncated analogue, with the FT of a box-function defining the point at which acquisition ended; under these circumstances, spectral peaks take the form of sinc functions[¶¶]. The resulting artefacts are commonly called *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 efficient processing by divide and conquer DFT methods like the Cooley-Tukey algorithm [11]. Beyond this, it is actually possible to enhance the information content of the real component of the spectrum if an FID comprising $2^{x-1} < N \leq 2^x$ points is zero-filled to 2^{x+1} points. 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 via the Hilbert Transform, in accordance with the Kramers-Kronig relations [40]:

$$\Re(s(F)) = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{\Im(s(F'))}{F-F'}dF', \tag{1.32a}$$

$$\Im(s(F)) = -\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{\Re(s(F'))}{F-F'}dF'. \tag{1.32b}$$

Zero-filling beyond a factor of 2 can bestow a cosmetic improvement to spectra, but it does not incorporate any new information into them; any additional points are simply interpolations.

Phase correction

The real and imaginary components of Equation 1.31 are as follows:

$$\Re(s(F)) = \sum_m a_m(\cos(\phi_m)\mathcal{A}_m(F) + \sin(\phi_m)\mathcal{D}_m(F)), \tag{1.33a}$$

$$\Im(s(F)) = \sum_m a_m(\sin(\phi_m)\mathcal{A}_m(F) - \cos(\phi_m)\mathcal{D}_m(F)), \tag{1.33b}$$

[¶¶] The sinc function is defined as $\mathrm{sinc}(x) = {}^{\sin(x)}/_{x}$.

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

$$\mathcal{A}_m(F) = \frac{\eta_m}{\eta_m^2 + 4\pi^2(f_m - F)^2}, \tag{1.34a}$$

$$\mathcal{D}_m(F) = \frac{2\pi(f_m - F)}{\eta_m^2 + 4\pi^2(f_m - F)^2}. \tag{1.34b}$$

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

1. Its maximum corresponds to the signal frequency, while a dispersion Lorentzian has a magnitude of 0 at this frequency.
2. It decays more rapidly and therefore exhibits better resolution; at large frequency offsets (i.e. when $|F - f_m|$ is notably greater than 0 Hz) 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 comprises peaks which all possess absorption Lorentzians is therefore desired, which is possible if all signals have a phase of 0°:

$$\Re(s(F)) = \sum_m a_m \mathcal{A}_m(F), \tag{1.35a}$$

$$\Im(s(F)) = -\sum_m a_m \mathcal{D}_m(F). \tag{1.35b}$$

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

$$\phi_m = \Phi_0 + \Phi_1 f_m, \tag{1.36}$$

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 $\boldsymbol{s}_\phi$ for different values of p_0 and p_1, according to

$$s_{\phi,n} = s_n \exp\left(-\mathrm{i}\left(p_0 + \frac{p_1 n}{N-1}\right)\right). \tag{1.37}$$

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

***The consideration of dispersion lineshapes is preferred in certain applications. As an example, the lock conventionally monitors the solvent deuterium signal through consideration of its dispersion-mode spectrum. This is since, if the magnetic field drifts, it is trivial to ascertain the direction of the drift based on whether the zero of the dispersion spectrum becomes positive or negative[41].

†††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.

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 refers to 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 adding a spline or high-order polynomial to the spectrum. One class of approaches which are used widely involve 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 before subtracting the fit from the spectrum [42, 43].

1.2.2 Processing Multidimensional Data

As for 1D datasets, it is desirable that multidimensional NMR spectra feature peaks which are frequency discriminated, and comprise pure absorption lineshapes in each dimension. A description of how this can be achieved for both amplitude- and phase-modulated datasets is now given, through the consideration of 2D FIDs which comprise a single signal.

Amplitude-Modulated Datasets

It is clear that a cosine-modulated signal, given by Equation 1.24 with $D = 2$ and $\zeta^{(1)} = \cos(\cdot)$ cannot achieve frequency discrimination in the indirect dimension, on account of the relation

$$\cos\left(2\pi f^{(1)} t^{(1)}\right) = \frac{\exp\left(2\pi \mathrm{i} f^{(1)} t^{(1)}\right) + \exp\left(-2\pi \mathrm{i} f^{(1)} t^{(1)}\right)}{2} \tag{1.38}$$

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 $\left(f^{(1)}, f^{(2)}\right)$, and the other at the mirror-image frequency in the indirect dimension, $(2f^{(1)}_{\text{off}} - f^{(1)}, f^{(2)})$. 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* [44]. A spectrum of this form is presented in Figure 1.4.a. 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 (Figure 1.4.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 the necessary quadrature detection in the indirect dimension. For numerous multidimensional experiments, repeating the pulse sequence with methodical adjust-

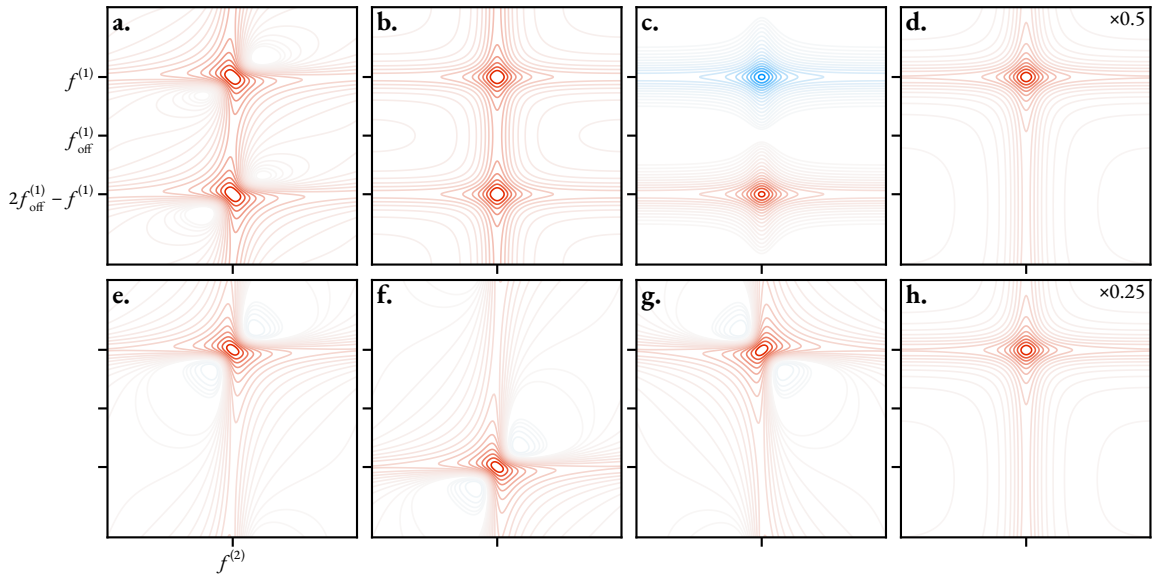


Figure 1.4: Spectra acquired from amplitude- and phase- modulated 2D signals. Red contour lines denote positive values, while blue contours denote negative values. **a.** The FT of a cosine-modulated FID, featuring peaks both at the true resonance frequency ($f^{(1)}$) and the mirrored frequency ($2f^{(1)}_{\text{off}} - f^{(1)}$), 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 $f^{(1)}_{\text{off}}$. **h.** The summation of **e.** and **g.** generates a spectrum with an absorption lineshape.

ments to the phases of particular pulses in a process referred to as *phase cycling* enables this [34: 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 (Figure 1.4.c) on account of sine being an odd function. It then becomes possible to generate a frequency discriminated spectrum by subtracting the sine spectrum from the cosine spectrum (Figure 1.4.d).

Phase-modulated signals

A positive (*hypercomplex*) phase-modulated signal ($\zeta^{(1)} = \exp(\mathrm{i}\cdot)$) is frequency discriminated due to its quadrature nature in the indirect dimension. However, a 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 (Figure 1.4.e). Certain pulse sequences, including two-dimensional J-resolved (2DJ) spectroscopy [45, 46] (see Chapter 4) and COSY [12, 13, 14] 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, and they also suffer from significant non-linearities. In scenarios where it is possible, it is desirable to acquire the equivalent negative signal ($\zeta^{(1)} = \exp(-\mathrm{i}\cdot)$), whose FT leads to a peak with the same phase twist form, but centered at $2f^{(1)}_{\text{off}} - f^{(1)}$ (Figure 1.4.f). Inverting this spectrum about $f^{(1)}_{\text{off}}$ (Figure 1.4.g), and summing with the positive spectrum nullifies the dispersive contribution, generating a spectrum with an absorption lineshape [47] (Figure 1.4.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 to produce D-dimensional spectra with absorption-mode peaks in all dimensions.

1.2.3 Analysing NMR Data

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. However, in many situations a detailed quantitative description of the data is desired. Some examples include:

- Deducing the relative concentrations of molecules in a mixture, with application in areas such as reaction monitoring [48] and metabolomics [49]. In these circumstances, it is necessary to deduce the relative amplitudes of the signals in the data.
- Determining properties such as translational diffusion coefficients and relaxation rates, which can be achieved using a suite of 2D experiments in which successive 1D FIDs exhibit attenuations in their signal amplitudes (these are discussed in Section 3.2). Typical approaches to analysing these datasets require identification of the frequencies of spectral peaks of interest, from which amplitudes are extracted and fit to a function which models the expected decay profile of the data.

Most NMR users are principally interested in the integrals and positions of peaks in spectra. Integrals are usually computed by applying a discrete integration method such as Simpson's rule on a collection of consecutive points in the spectrum which lie within user-specified regions. 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 in the spectrum, subject to tolerances which aim to prevent blips due to noise being categorised as peaks.

While integration and peak picking works adequately for high SNR spectra featuring very well-separated peaks, in most practical situations accurate quantification of individual signals is impossible using this approach, principally due to the inherently poor resolution associated with the FT. For this reason, considerable interest has been given to the development of techniques which are better-able to quantify NMR datasets, by attempting to estimate the optimal set of parameters $\boldsymbol{\theta}$ which describe them[‡‡‡]. 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; for reasons that will be explained shortly, time-domain estimation is the focus of this work.

Linear Prediction

Linear prediction (LP) [50, 51, 52, 53] is a procedure which is widely used in NMR data analysis for the purposes of (a) propagating an FID further in time in order to reduce the presence of truncation artefacts without the reliance on severe apodisation, and/or (b) correcting the commonly corrupted initial points of the FID, as a means of improving the spectral baseline. LP is notably 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 those which precede or succeed it.

LP stems from the concept that an 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. For a 1D FID, it is assumed that

$$y_n = \sum_{l=1}^{L} c_l y_{n-l} + e_l \quad \forall n \in \{L, L+1, \cdots, N-1\}, \tag{1.39}$$

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

$$y_n = \sum_{l=1}^{L} b_l y_{n+l} + e_l \quad \forall n \in \{0, 1, \cdots, N-L-1\}. \tag{1.40}$$

Determining the LP coefficients enables the estimation of FID values beyond the data actually acquired ($n > N-1$), as well as to replace datapoints which are anticipated to be corrupted. It

[‡‡‡]The process of estimating the parameters describing an 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 parameters which describe a dataset. Equivalently, *convolution* is not the process of generating an NMR dataset by summing a number of complex sinusoids, though it is often described as so in the literature.

should be noted that Equations 1.39 and 1.40 are only technically 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, with effective results attainable as long as the data is not extrapolated by too great an extent. The most common means of performing LP is by solving the Yule-Walker equations [54, 55], which describe the relationship between the signal autocorrelation coefficients and the LP coefficients [51: Section 3.3], with the Levinson-Durbin algorithm providing an efficient means of solving the equations [56, 57].

SVD-based methods

Some of the most prevalent methods for NMR estimation involve *singular value decomposition* (SVD) as a key component. For each of these methods, SVD is applied to a Hankel or Toeplitz matrix containing the FID. Above a certain SNR threshold, it is found that the singular values in the data matrix corresponding to 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 required is dependent on the suspected *model order* (i.e. the number of contributing signals, M) of the dataset, which must be supplied, or determined by some appropriate metric. All the SVD-based techniques determine the non-linear parameters (frequencies and damping factors, incorporated in the signal poles) associated with the dataset. Subsequently, the amplitudes and phases can be computed by solving a linear least-squares problem involving a Vandermonde matrix of the signal poles:

$$\boldsymbol{\alpha} = \boldsymbol{Z}^{+}\boldsymbol{y}, \tag{1.41a}$$

$$\boldsymbol{Z} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ z_1^1 & z_2^1 & \cdots & z_M^1 \\ \vdots & \vdots & \ddots & \vdots \\ z_1^{N-1} & z_2^{N-1} & \cdots & z_M^{N-1} \end{bmatrix}. \tag{1.41b}$$

Both linear prediction singular value decomposition (LPSVD) [58, 59] and Hankel singular value decomposition (HSVD) [60] have be used extensively in the field of *in vivo* magnetic resonance spectroscopy (MRS) after introduction by Barkhuijsen and co-workers [61, 62, 63, 64, 65]. With LPSVD, the signal poles are determined by finding the roots of a polynomial which features the backward LP coefficients associated with the signal, in a similar fashion to the classic Prony method [66]. 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) [67, 68, 69] is based on solving a gen-

eralised eigenvalue problem. Pines and co-workers introduced the information theoretic matrix pencil method (ITMPM) [70], for use in NMR, which combines the MPM with a method for estimating the model order, such that no *a priori* information is required from the user. Recently, the MPM has been employed for the analysis of NMR relaxometry data [71, 72]. Outlines of the MPM and its 2D equivalent, the modified matrix enhancement and matrix pencil method (MMEMPM) [73, 74] are provided in detail in Section 2.2.1.

Iterative Methods

For problems involving nonlinear function fitting, the most widespread approaches are iterative techniques. The variable projection (VARPRO) method [75] is such an example, which has been applied extensively in the field of MRS, following its introduction by van der Veen and co-workers [76, 77]. VARPRO relies on determining the following quantity, an example of a residual sum-of-squares (RSS) problem:

$$\boldsymbol{\theta}^{(*)} = \underset{\boldsymbol{\theta}\in\mathbb{R}^{4M}}{\arg\min} \left\|\boldsymbol{y} - \boldsymbol{x}(\boldsymbol{\theta})\right\|^2 \equiv \underset{\boldsymbol{\theta}\in\mathbb{R}^{4M}}{\arg\min} \left\|\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{\alpha}\right\|^2, \tag{1.42}$$

where $\boldsymbol{\theta}^{(*)}$ is the optimal set of parameters which describe the data, often referred to as the *maximum likelihood estimate* (MLE). The linear parameters, residing in the vector of complex amplitudes, can be expressed as $\boldsymbol{Z}^{+}\boldsymbol{y}$ (see Equation 1.41). By doing this, it becomes possible to recast the optimisation problem to only solve for the nonlinear terms:

$$\boldsymbol{\rho}^{(*)} = \underset{\boldsymbol{\rho}\in\mathbb{R}^{2M}}{\arg\min} \left\|\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{Z}^{+}\boldsymbol{y}\right\|^2, \tag{1.43a}$$

$$\boldsymbol{\rho} = \begin{bmatrix}\boldsymbol{f}^{\mathrm{T}} & \boldsymbol{\eta}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}. \tag{1.43b}$$

The Levenberg-Marquardt (LM) algorithm [78, 79] is typically employed to perform the optimisation routine, with amplitudes and phases determined with linear least squares afterwards.

For such a routine to be effective, 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; iterative methods like the LM algorithm only tend to perform successfully if they start with a set of parameters sufficiently close to the MLE in the parameter space. Further specifications can be supplied to VARPRO, 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 — a feature which is common in the NMR of small molecules. Advanced method for accurate, robust, and efficient spectral fitting (AMARES) is an improvement of VARPRO, as it facilitates the imposition of a wider range of linear constraints on the parameter set [80]; it is now established as one of the most prominent means of quantifying MRS signals.

There is little precedent for the use of iterative parametric estimation methods like VARPRO and AMARES in NMR analysis, despite its widespread adoption in MRS. One key reason for this is due 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 in an acquired dataset, since all signals will derive from species which make up the human metabolome, whose chemical shifts and multiplet structures will be well-established. As ^{31}P is the most common nucleus studied in MRS, the number of potential species contributing to datasets is reduced even further. 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 is likely to become time-consuming and cumbersome.

Bayesian Methods

Complete reduction to amplitude frequency table (CRAFT) [81, 82] is a routine developed by Krishnamurthy, inspired by previous work by Bretthorst [83, 84, 85, 86, 87], for the parametric estimation of FIDs based on the principle of Bayesian inference. In brief, CRAFT consists of defining prior distributions for the signal parameters, and gradually increasing the number of signal components in the model until a maximisation of the posterior probably is achieved; through this approach, both model order selection and parameter estimation are achieved in tandem. In order to reduce the considerable computational burden associated with the method, the complete FID is downsampled into a series of "sub-FIDs", with each sub-FID being analysed separately. CRAFT has been extended for the consideration of 2D FIDs, through a simple extension of the 1D estimation procedure [88]; after the direct-dimension of the dataset is processed using the conventional FT approach, each FID in the indirect-dimension is estimated separately, yielding a $4M$ parameter set for each indirect increment. Such an approach is not able to extract all $6M$ parameters associated with the 2D dataset however.

Machine Learning Methods

A vast surge in the prevalence of machine learning methods across virtually all scientific disciplines has taken place in recent years. Impressive showcases in the field of NMR estimation have been presented, which has so far been limited to data in the Fourier domain [89, 90]. These methods employ deep (i.e. many-layered) neural networks featuring numerous convolution layers, which have been shown to be highly successful in image- and signal-processing applications [91]. The neural network is "trained", by subjecting it to a vast corpus of NMR data; each dataset is labelled with the correct parameter estimate. By assessing how well the network performs at estimating a given dataset, the network's parameters are adjusted. After exposure to many datasets,

it is hoped that the network becomes effective at performing in a general manner, being able to estimate datasets which it has not been exposed to previously. These heavily intensive methods have become tractable in recent years thanks to advancements in the processing power of computers. One particularly significant milestone has been the generation of programmable graphical processing units, which can carry out the large-scale numerical computations involved in training neural networks highly efficiently.

1.3 Overview of this Work

1.3.1 Conception and Motivation

The focus of this work is the development of a routine which performs parametric estimation on NMR datasets. Motivation initially came from discussions within the NMR Methodology Group in Manchester involving Dr Mohammadali Foroozandeh and co-workers (notably Prof. Gareth Morris and Prof. Mathias Nilsson) while Dr Foroozandeh was a Postdoctoral researcher there. The group were interested in generating pure shift NMR spectra from 2DJ datasets via appropriate post-processing of the data. While little progress was made when Dr Foroozandeh was based in Manchester, he wished to continue with the project after moving to Oxford to take up a research fellowship; I took the reins of the project when I joined his nascent research group as a PhD student.

To ensure its applicability to 2DJ datasets, the following properties were sought while devising what a suitable estimation routine would entail:

Support for 1D and 2D data The method should be able to analyse both 1D FIDs and also hypercomplex 2D FIDs (the form which 2DJ datasets take). 2D data should be analysed holistically, rather than as successive 1D increments, as is the case in methods like CRAFT [88] and other previous approaches which will be discussed later. It is often the case that signals which heavily overlap in a 1D NMR dataset are well-separated in a dataset with more dimensions; exploiting this concept means that holistic estimation approaches are able to resolve and quantify individual signals more effectively.

Time-domain based As discussed in Section 1.2.2, due to the hypercomplex nature of 2DJ FIDs, generating spectra with desirable absorption lineshapes is not possible. Typically, resorting to displaying the spectra in magnitude-mode is deemed optimal, as this overcomes the phase-twist peak lineshapes. However, such spectra suffer from severe non-linearities and dispersion-mode contributions, both of which make the task of estimating 2DJ data in the Fourier domain challenging. For this reason, estimating the dataset by considering its FID rather than its spectrum is preferred.

Accessibility To achieve wide-spread adoption, especially by non-expert NMR users, the method should require minimal user intervention to perform effectively. As such, a method requiring the specification of as little prior knowledge about the data as possible is desired. On top of this, the method should be available as software that users can gain familiarity with easily.

1.3.2 Thesis Overview

The remainder of this thesis comprises the following chapters:

- **Chapter 2** discusses the theory behind routines which can be applied to determine parameter estimates related to 1D and 2D FIDs.
- **Chapter 3** provides illustrations of the performance of the estimation routine on 1D NMR datasets. Furthermore, means in which parametric estimation routine can be harnessed for two applications are explored:
  - The analysis of amplitude-attenuated datasets, such as those derived from diffusion-, T_1- and T_2-measuring 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 devised method, given the acronym CUPID, for generating pure shift spectra via 2DJ estimation.
- **Chapter 5** describes the source code developed as part of this project, called NMR estimation in Python (NMR-EsPy).
- Finally, conclusions and considerations for further potential developments are made in **Chapter 6**.

Additional supporting information can be found in the Appendix:

- **Appendix A** provides additional information on the theory related to this work, including descriptions of mathematical concepts, more details on the estimation routine, and outlines of relevant algorithms. Descriptions of some mathematical concepts which are mentioned but not described in detail in the main text are provided in Appendix A.1. These concepts are underlined the first time they are encountered in the main text.
- **Appendix B** provides code listings outlining how the methods described in this work can be implemented in the Python programming language. These are effectively minimalist variants of code found in the NMR-EsPy package.
- **Appendix C** outlines how the simulated and experimental datasets considered in this work were generated.

- **Appendix D** is a chapter insert from the documentation of NMR-EsPy, comprising tutorials to help users gain familiarity with the software.

The following publications are related to this work:

S. G. Hulse, M. Foroozandeh, *Journal of Magnetic Resonance* **2022**, *338*, 107173

At the time of writing (24/1/2024) a draft for a communication is being being worked on which describes the 2DJ estimation work outlined in Chapter 4. It is intended for this to be submitted to *Angewandte Chemie* or *ChemComm* in the near future.

Chapter 2

Theory

This chapter provides a description of the theory behind an estimation routine which has been developed for the consideration of time-domain NMR data [69, 70, 93]. In essence, the routine consists of generating a parameter estimate using the SVD-based MPM, which is fed to an iterative non-linear programming (NLP) routine to produce the final result. The MPM is employed to form an initial guess of parameters, while the NLP routine behaves as a means of validation, by attempting to make the parameter estimate consistent with known features of the data. The technique can be thought of as a compromise between "black-box methods" [94] which require little to no prior knowledge about the data, and iterative methods like VARPRO and AMARES, which require vast amounts of user input, but are typically able to estimate complex datasets more effectively. Furthermore, after profiling the run time and memory consumption of the technique, a method for producing filtered FIDs, featuring a subset of the signals and (optionally) fewer datapoints relative to the full FID is presented, which can drastically reduce the burden on computational resources in carrying out the estimation procedure.

2.1 Outline of the Problem

By applying the theory of NMR, one can rationalise how a particular spin system, subjected to a given pulse sequence, is mapped to an FID. This is an example of a *forward problem*, in which one wishes to determine how a set of parameters leads to an observed dataset. The chemical shifts, J-couplings, dipolar couplings, etc. of the spin system may be recast as a list of signal amplitudes, phases, frequencies and damping factors, which define the form that the FID is expected to take. Sophisticated pieces of software such as Spinach [95] solve this forward problem, yielding NMR experiment simulations. Simulating an NMR experiment is a *well-posed* problem, since it satisfies the following properties:

1. The problem has a solution.

2. The solution is unique.
3. The solution's behaviour changes continuously with the parameters provided. For example, continuously changing the chemical shift of a given spin will lead to the position(s) of the signal(s) arising from it in the resulting spectrum to continuously vary. Other phenomena such as strong coupling effects may manifest as a given spin's chemical shift changes; these will evolve in a continuous fashion as well.

The estimation of FIDs is an example of an *inverse problem*; given an acquired dataset, the objective is to determine the set of parameters which went into constructing it. As with many inverse problems, FID estimation is *ill-posed* [96]; such problems do not satisfy at least one of the three properties above. For example, given an FID, it is possible to conceive of numerous signal parameter specifications which would lead to a faithful representation of the data in a least squares sense, especially when considering FIDs corrupted with noise which comprise signals of similar frequencies. Therefore, determining the "optimal" parameter set is a rather difficult challenge.

For the purposes of this work, it is always assumed that an FID to be estimated $\boldsymbol{Y} \in \mathbb{C}^{N^{(1)} \times \cdots \times N^{(D)}}$ is hypercomplex in form, meaning that it obeys Equation 1.24 with $\zeta^{(d)} = \exp(\mathrm{i}\cdot)\ \forall d \in \{1, \cdots, D\}$:

$$y_{n^{(1)},\cdots,n^{(D)}} = x_{n^{(1)},\cdots,n^{(D)}}(\boldsymbol{\theta}) + w_{n^{(1)},\cdots,n^{(D)}}, \tag{2.1a}$$

$$x_{n^{(1)},\cdots,n^{(D)}}(\boldsymbol{\theta}) = \sum_{m=1}^{M} a_m \exp(\mathrm{i}\phi_m) \prod_{d=1}^{D} \exp\left(\left(2\pi\mathrm{i}\left(f_m^{(d)} - f_{\mathrm{off}}^{(d)}\right) - \eta_m^{(d)}\right) n^{(d)} \Delta_t^{(d)}\right), \tag{2.1b}$$

$$w_{n^{(1)},\cdots,n^{(D)}} \sim \mathcal{NC}\left(0, 2\sigma^2\right), \tag{2.1c}$$

where $\Delta_t^{(d)} = 1/f_{\mathrm{sw}}^{(d)}$. Alternatively, Equation 2.1b may be expressed in terms of complex amplitudes and signals poles as follows:

$$x_{n^{(1)},\cdots,n^{(D)}}(\boldsymbol{\theta}) = \sum_{m=1}^{M} \alpha_m \prod_{d=1}^{D} {z_m^{(d)}}^{n^{(d)}}, \tag{2.2a}$$

$$\alpha_m = a_m \exp(\mathrm{i}\phi_m), \tag{2.2b}$$

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

Under this model, it is assumed that an FID consists of a summation of M damped complex sinusoids in the presence in AWGN. It should be noted that, prior to estimating the dataset, it is normalised such that the signal actually under consideration is $\boldsymbol{Y}/\|\boldsymbol{Y}\|$. To make the final result reflect the unnormalised dataset, the estimated amplitudes can be multiplied by $\|\boldsymbol{Y}\|$.

It is the goal of parametric estimation to establish the identity of all the quantities that describe the model component $\boldsymbol{X}$, which are distilled into the vector $\boldsymbol{\theta} \in \mathbb{R}^{2(D+1)M}$, given by Equation 1.23.

Due to the assumed AWGN nature of the noise array, the probability density functions (PDFs) of the real and imaginary components of an individual noise element are the following (in accordance with Equation 2.1c):

$$p\left(\Re\left(w_{n^{(1)},\cdots,n^{(D)}}\right)\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\Re\left(w_{n^{(1)},\cdots,n^{(D)}}\right)^2}{2\pi\sigma^2}\right), \tag{2.3a}$$

$$p\left(\Im\left(w_{n^{(1)},\cdots,n^{(D)}}\right)\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\Im\left(w_{n^{(1)},\cdots,n^{(D)}}\right)^2}{2\pi\sigma^2}\right). \tag{2.3b}$$

As the noise components are assumed to be independent and identically distributed, the joint PDF describing the entire noise array is given by the product of the PDFs of all its components*:

$$\begin{aligned} p\left(\boldsymbol{W}\right) &= \prod_{n^{(1)}=0}^{N^{(1)}-1} \cdots \prod_{n^{(D)}=0}^{N^{(D)}-1} \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^2 \exp\left(-\frac{\left|w_{n^{(1)},\cdots,n^{(D)}}\right|^2}{2\sigma^2}\right) \\ &= \frac{1}{\left(2\pi\sigma^2\right)^{N_{\text{tot}}}} \exp\left(-\frac{\|\boldsymbol{W}\|^2}{2\sigma^2}\right), \end{aligned} \tag{2.4}$$

where $N_{\text{tot}} := N^{(1)} \times \cdots \times N^{(D)}$. By noting that the noise array is the difference between the data and model, the likelihood function of $\boldsymbol{\theta}$ given the FID $\boldsymbol{Y}$ is

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{Y}) = \frac{1}{\left(2\pi\sigma^2\right)^{N_{\text{tot}}}} \exp\left(-\frac{\|\boldsymbol{Y} - \boldsymbol{X}(\boldsymbol{\theta})\|^2}{2\sigma^2}\right). \tag{2.5}$$

It is common to consider instead the log-likelihood function, $\ell(\boldsymbol{\theta}|\boldsymbol{Y}) := \ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{Y})$:

$$\ell(\boldsymbol{\theta}|\boldsymbol{Y}) = -N_{\text{tot}} \ln\left(2\pi\sigma^2\right) - \frac{\|\boldsymbol{Y} - \boldsymbol{X}(\boldsymbol{\theta})\|^2}{2\sigma^2}. \tag{2.6}$$

As the application of the logarithm is a monotonic transformation, the arguments of the maxima of $\mathcal{L}$ and ℓ are equivalent. Equation 2.6 implies that the optimal set of parameters $\boldsymbol{\theta}^{(*)}$, i.e. the MLE, is that which minimises the RSS between the data and the model:

$$\boldsymbol{\theta}^{(*)} = \underset{\boldsymbol{\theta}\in\mathbb{R}^{2(D+1)M}}{\arg\max}\ \ell\left(\boldsymbol{\theta}|\boldsymbol{Y}\right) \equiv \underset{\boldsymbol{\theta}\in\mathbb{R}^{2(D+1)M}}{\arg\min}\ \|\boldsymbol{Y} - \boldsymbol{X}(\boldsymbol{\theta})\|^2. \tag{2.7}$$

The application of NLP is a well-established approach to solve such a problem [93, 97]. The basic principle behind NLP is to iteratively explore, in a methodical way, how a function varies

*N.B. $\left|w_{n^{(1)},\cdots,n^{(D)}}\right|^2 \equiv \Re\left(w_{n^{(1)},\cdots,n^{(D)}}\right)^2 + \Im\left(w_{n^{(1)},\cdots,n^{(D)}}\right)^2$

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 [98, 99, 100], in scenarios where the function under consideration has well-defined, computationally tractable derivatives, the use of these can be valuable when solving optimisation problems; the problem outlined in Equation 2.7 is such an example.

As discussed already, for NLP to perform effectively, a large amount of *a priori* information is typically required, in the form of an initial guess, possibly alongside other constraints. To achieve this, the method employed in this work makes use of the MPM, the subject of the next section.

2.2 Matrix Pencil Methods†

In this section, a description of the MPM — also referred to as the generalized pencil-of-function method [102] — is provided. This work is limited to the consideration of 1D and 2D NMR data, and so both the original 1D MPM and its 2D analogue, the MMEMPM are discussed.

2.2.1 The 1D MPM

The MPM provides a route to extracting the signal poles of a 1D dataset, based on the assumption that the number of signals M that the data comprises is known [67, 68, 69]. To motivate how the MPM works, first consider a dataset which is devoid of noise, given by Equation 2.2 with $D = 1$:

$$x_n(\boldsymbol{\theta}) = \sum_{m=1}^{M} \alpha_m z_m^n \quad \forall n \in \{0, \cdots, N-1\}. \tag{2.8}$$

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

$$\boldsymbol{H_x} = \begin{bmatrix} x_0 & x_1 & \cdots & x_L \\ x_1 & x_2 & \cdots & x_{L+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N-L-1} & x_{N-L} & \cdots & x_{N-1} \end{bmatrix}. \tag{2.9}$$

This matrix comprises windowed segments of the FID, with each row featuring the segment shifted to the right by one point relative to the row above. $L \in \mathbb{N}$ is referred to as the *pencil parameter*, which dictates the size of each window. From $\boldsymbol{H_x}$, two matrices are defined: $\boldsymbol{H_{x1}}$ and $\boldsymbol{H_{x2}}$. These

†For two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{C}^{m\times n}$, their *pencil* is the set of all matrices of the form $\boldsymbol{A} - \lambda\boldsymbol{B}$, with $\lambda \in \mathbb{C}$ [101: Section 7.7]. In most applications, the matrices involved are assumed to be square (i.e. $m = n$); this is not the case in the context of the MPM however.

are formed by the removal of the last and first column of $\boldsymbol{H}_{\boldsymbol{x}}$, respectively:

$$\boldsymbol{H}_{\boldsymbol{x}1} = \begin{bmatrix} x_0 & x_1 & \cdots & x_{L-1} \\ x_1 & x_2 & \cdots & x_L \\ \vdots & \vdots & \ddots & \vdots \\ x_{N-L-1} & x_{N-L} & \cdots & x_{N-2} \end{bmatrix}, \tag{2.10a}$$

$$\boldsymbol{H}_{\boldsymbol{x}2} = \begin{bmatrix} x_1 & x_2 & \cdots & x_L \\ x_2 & x_3 & \cdots & x_{L+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N-L} & x_{N-L+1} & \cdots & x_{N-1} \end{bmatrix}. \tag{2.10b}$$

$\boldsymbol{H}_{\boldsymbol{x}1}$ and $\boldsymbol{H}_{\boldsymbol{x}2}$ can be deconstructed into the following forms involving matrices containing the M signal poles and complex amplitudes that the data comprises:

$$\boldsymbol{H}_{\boldsymbol{x}1} = \boldsymbol{Z}_{\mathrm{L}} \boldsymbol{A} \boldsymbol{Z}_{\mathrm{R}}, \tag{2.11a}$$

$$\boldsymbol{H}_{\boldsymbol{x}2} = \boldsymbol{Z}_{\mathrm{L}} \boldsymbol{A} \boldsymbol{Z}_{\mathrm{D}} \boldsymbol{Z}_{\mathrm{R}}, \tag{2.11b}$$

$$\mathbb{C}^{\left(N^{(1)}-L^{(1)}\right)\times M} \ni \boldsymbol{Z}_{\mathrm{L}} = \begin{bmatrix} \mathbf{1} & \boldsymbol{z} & \boldsymbol{z}^2 & \cdots & \boldsymbol{z}^{N-L-1} \end{bmatrix}^{\mathrm{T}}, \tag{2.11c}$$

$$\mathbb{C}^{M\times L} \ni \boldsymbol{Z}_{\mathrm{R}} = \begin{bmatrix} \mathbf{1} & \boldsymbol{z} & \boldsymbol{z}^2 & \cdots & \boldsymbol{z}^{L-1} \end{bmatrix}, \tag{2.11d}$$

$$\mathbb{C}^{M\times M} \ni \boldsymbol{Z}_{\mathrm{D}} = \mathrm{diag}\left(\boldsymbol{z}\right), \tag{2.11e}$$

$$\mathbb{C}^{M\times M} \ni \boldsymbol{A} = \mathrm{diag}\left(\boldsymbol{\alpha}\right), \tag{2.11f}$$

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

$$\boldsymbol{z} = \begin{bmatrix} z_1 & z_2 & \cdots & z_M \end{bmatrix}^{\mathrm{T}}. \tag{2.11h}$$

The matrix pencil $\boldsymbol{H}_{\boldsymbol{x}2} - \lambda \boldsymbol{H}_{\boldsymbol{x}1}$, with $\lambda \in \mathbb{C}$, can therefore be expressed as

$$\boldsymbol{H}_{\boldsymbol{x}2} - \lambda \boldsymbol{H}_{\boldsymbol{x}1} = \boldsymbol{Z}_{\mathrm{L}} \boldsymbol{A} \left(\boldsymbol{Z}_{\mathrm{D}} - \lambda \boldsymbol{I}_M\right) \boldsymbol{Z}_{\mathrm{R}}, \tag{2.12}$$

where $\boldsymbol{I}_M \in \mathbb{C}^{M\times M}$ is the identity matrix. Assuming that the condition $M \leq L \leq N - M$ is met — this ensures that both the number of rows and columns of the matrix pencil are at least M — the rank of the matrix pencil will be M. Now consider the case when the scalar λ is equal to one of the signal poles i.e. $\lambda = z_m \; \forall m \in \{1, \cdots, M\}$: the element $\left[\boldsymbol{Z}_{\mathrm{D}} - \lambda \boldsymbol{I}_M\right]_{m,m}$ becomes 0, which will lead to the matrix pencil's rank being reduced by 1. One means of extracting the signal poles is by computing the eigenvalues of the matrix $\boldsymbol{H}_{\boldsymbol{x}1}^{+} \boldsymbol{H}_{\boldsymbol{x}2}$. Deriving the corresponding complex amplitudes can then be achieved by solving the set of linear equations given by Equation 1.41, with $\boldsymbol{y}$ replaced by $\boldsymbol{x}$. Extraction of the amplitudes, phases, frequencies, and damping factors from the

signal poles and complex amplitudes can then take place:

$$\boldsymbol{a} = |\boldsymbol{\alpha}|, \tag{2.13a}$$

$$\boldsymbol{\phi} = \arctan\left(\frac{\Im(\boldsymbol{\alpha})}{\Re(\boldsymbol{\alpha})}\right), \tag{2.13b}$$

$$\boldsymbol{f} = \frac{f_{\mathrm{sw}}}{2\pi}\Im\left(\ln \boldsymbol{z}\right) + f_{\mathrm{off}}, \tag{2.13c}$$

$$\boldsymbol{\eta} = -f_{\mathrm{sw}}\Re\left(\ln \boldsymbol{z}\right). \tag{2.13d}$$

Noise corruption complicates the process of determining the M signal poles. The Hankel matrix associated with an FID corrupted by noise $\boldsymbol{H_y}$ is likely to be of full-rank, i.e. $\mathrm{rank}(\boldsymbol{H_y}) = \min(N-L, L+1)$. To minimise the influence of noise on the estimated signal poles, it is necessary to generate a rank-reduced matrix $\widetilde{\boldsymbol{H}}_{\boldsymbol{y}}$. By employing the Eckart-Young-Mirsky (EYM) theorem [101: Section 2.2], an appropriate matrix can be obtained through SVD:

$$\widetilde{\boldsymbol{H}}_{\boldsymbol{y}} = \boldsymbol{U}_M \boldsymbol{\Sigma}_M \boldsymbol{V}_M^{\dagger}, \tag{2.14a}$$

$$\mathbb{C}^{(N^{(1)}-L^{(1)})\times M} \ni \boldsymbol{U}_M = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_M \end{bmatrix}, \tag{2.14b}$$

$$\mathbb{C}^{(L^{(1)}+1)\times M} \ni \boldsymbol{V}_M = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_M \end{bmatrix}, \tag{2.14c}$$

$$\mathbb{C}^{M\times M} \ni \boldsymbol{\Sigma}_M = \mathrm{diag}\left(\sigma_1, \sigma_2, \cdots, \sigma_M\right). \tag{2.14d}$$

σ_m is the m^{th} largest singular value of $\boldsymbol{H_y}$, while $\boldsymbol{u}_m \in \mathbb{C}^{N-L}$ and $\boldsymbol{v}_m \in \mathbb{C}^{L+1}$ are the corresponding left and right singular vectors, respectively. The EYM theorem proves that $\widetilde{\boldsymbol{H}}_{\boldsymbol{y}}$ is the closest matrix of rank M to $\boldsymbol{H_y}$ in a Frobenius norm sense, i.e.

$$\widetilde{\boldsymbol{H}}_{\boldsymbol{y}} = \underset{\boldsymbol{A}:\ \mathrm{rank}(\boldsymbol{A})=M}{\arg\min} \left\| \boldsymbol{A} - \boldsymbol{H_y} \right\|. \tag{2.15}$$

The intention behind applying the SVD in the MPM and other methods like LPSVD and HSVD is to discard components in the data matrix which are associated with noise. Assuming the SNR of an FID is sufficiently high, a clear distinction is often found between the magnitudes of singular values associated with true signal components and those associated with noise (see Figure 2.2.b for an example of this).

The signal poles are finally extracted by computing the eigenvalues of $\widetilde{\boldsymbol{H}}_{\boldsymbol{y}1}^{+}\widetilde{\boldsymbol{H}}_{\boldsymbol{y}2}$, where $\widetilde{\boldsymbol{H}}_{\boldsymbol{y}1}$ and $\widetilde{\boldsymbol{H}}_{\boldsymbol{y}2}$ have the same relation to $\widetilde{\boldsymbol{H}}_{\boldsymbol{y}}$ as $\boldsymbol{H_{x}}1$ and $\boldsymbol{H_{x}}2$ do to $\boldsymbol{H_x}$. As a less expensive alternative, the same result can be achieved by computing the eigenvalues of $\boldsymbol{U}_{M1}^{+}\boldsymbol{U}_{M2}$, with

$$\boldsymbol{U}_{M1} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_{M-1} \end{bmatrix}, \tag{2.16a}$$

$$\boldsymbol{U}_{M2} = \begin{bmatrix} \boldsymbol{u}_2 & \boldsymbol{u}_3 & \cdots & \boldsymbol{u}_M \end{bmatrix}. \tag{2.16b}$$

Algorithm 2.1 provides a pseudo-code description of the MPM, while Code Listing B.3 provides an example PYTHON implementation. For optimal results, the pencil parameter should adhere to $\lfloor N/3 \rfloor \leq L \leq \lfloor 2N/3 \rfloor$ [67]. In this work, $\lfloor N/3 \rfloor$ is always used, primarily since the computational burden of the method is at a maximum when $L = N/2$[‡].

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

procedure MPM($\boldsymbol{y} \in \mathbb{C}^N, f_{\text{sw}} \in \mathbb{R}_{>0}, f_{\text{off}} \in \mathbb{R}, M \in \mathbb{N}_0$)
  $L \leftarrow \lfloor N/3 \rfloor$;
  $\boldsymbol{H}_{\boldsymbol{y}} \leftarrow \begin{bmatrix} y_0 & y_1 & \cdots & y_L \\ y_1 & y_2 & \cdots & y_{L+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N-L-1} & y_{N-L} & \cdots & y_{N-1} \end{bmatrix}$;
  $\boldsymbol{U}, \boldsymbol{\sigma}, \boldsymbol{V} \leftarrow \text{SVD}(\boldsymbol{H}_{\boldsymbol{y}})$;
  if $M = 0$ **then**
    $M \leftarrow \text{MDL}(\boldsymbol{\sigma}, L, N)$;
  end if
  $\boldsymbol{U}_{M1}, \boldsymbol{U}_{M2} \leftarrow \boldsymbol{U}[:, :M-1], \boldsymbol{U}[:, 1:M]$;
  $\boldsymbol{z} \leftarrow \text{EIGENVALUES}\left(\boldsymbol{U}_{M1}^{+} \boldsymbol{U}_{M2}\right)$;
  $\boldsymbol{Z} \leftarrow \begin{bmatrix} \boldsymbol{1} & \boldsymbol{z} & \boldsymbol{z}^2 & \cdots & \boldsymbol{z}^N \end{bmatrix}^{\mathrm{T}}$;
  $\boldsymbol{\alpha} \leftarrow \boldsymbol{Z}^{+} \boldsymbol{y}$;
  $\boldsymbol{a}, \boldsymbol{\phi}, \boldsymbol{f}, \boldsymbol{\eta} \leftarrow |\boldsymbol{\alpha}|, \arctan\left(\frac{\Im(\boldsymbol{\alpha})}{\Re(\boldsymbol{\alpha})}\right), \frac{f_{\text{sw}}}{2\pi} \Im\left(\ln \boldsymbol{z}\right) + f_{\text{off}}, -f_{\text{sw}} \Re\left(\ln \boldsymbol{z}\right)$;
  if $\boldsymbol{\eta}$ contains negative values **then**
    Remove these from $\boldsymbol{\eta}$, and remove the corresponding values from $\boldsymbol{a}$, $\boldsymbol{\phi}$, and $\boldsymbol{f}$;
  end if
  $\boldsymbol{\theta}^{(0)} \leftarrow \begin{bmatrix} \boldsymbol{a}^{\mathrm{T}} & \boldsymbol{\phi}^{\mathrm{T}} & \boldsymbol{f}^{\mathrm{T}} & \boldsymbol{\eta}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$;
  return $\boldsymbol{\theta}^{(0)}$;
end procedure

procedure MDL($\boldsymbol{\sigma} \in \mathbb{R}^{L+1}, L \in \mathbb{N}, N \in \mathbb{N}$)
  for $k = 0, \cdots, L$ **do**
    $\text{MDL}_k \leftarrow -\ln\left(\frac{\prod_{r=k}^{L-1} \sigma_{r+1}^{1/L-k}}{\frac{1}{L-k}\sum_{r=k}^{L-1} \sigma_{r+1}}\right)^{(L-k)N} + \frac{1}{2}k(2L-k)\ln N$;
    if $k > 0$ **and** $\text{MDL}_k > \text{MDL}_{k-1}$ **then**
      $M \leftarrow k - 1$;
      break;
    end if
  end for
  return M;
end procedure

[‡]With $L = \lfloor N/2 \rfloor$, the matrix $\boldsymbol{H}_{\boldsymbol{y}}$ 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 complexity $\mathcal{O}(\min(L+1, N-L+1)^2 \times \max(L+1, N-L+1))$.

2.2.2 The 2D MMEMPM

The MPM was extended for the consideration of 2D data by Hua with the *matrix enhancement and matrix pencil method* (MEMPM) [73]. The method centers around the enhanced matrix $\boldsymbol{E_Y} \in \mathbb{C}^{L^{(1)}L^{(2)}\times(N^{(1)}-L^{(1)}+1)(N^{(2)}-L^{(2)}+1)}$, a block-Hankel matrix of the form

$$\boldsymbol{E_Y} = \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{y},0} & \boldsymbol{H}_{\boldsymbol{y},1} & \cdots & \boldsymbol{H}_{\boldsymbol{y},N^{(1)}-L^{(1)}} \\ \boldsymbol{H}_{\boldsymbol{y},1} & \boldsymbol{H}_{\boldsymbol{y},2} & \cdots & \boldsymbol{H}_{\boldsymbol{y},N^{(1)}-L^{(1)}+1} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{H}_{\boldsymbol{y},L^{(1)}-1} & \boldsymbol{H}_{\boldsymbol{y},L^{(1)}} & \cdots & \boldsymbol{H}_{\boldsymbol{y},N^{(1)}-1} \end{bmatrix}, \tag{2.17a}$$

$$\boldsymbol{H}_{\boldsymbol{y},n^{(1)}} = \begin{bmatrix} y_{n^{(1)},0} & y_{n^{(1)},1} & \cdots & y_{n^{(1)},N^{(2)}-L^{(2)}} \\ y_{n^{(1)},1} & y_{n^{(1)},2} & \cdots & y_{n^{(1)},N^{(2)}-L^{(2)}+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n^{(1)},L^{(2)}-1} & y_{n^{(1)},L^{(2)}} & \cdots & y_{n^{(1)},N^{(2)}-1} \end{bmatrix}. \tag{2.17b}$$

In an equivalent fashion to Equation 2.11, $\boldsymbol{H}_{\boldsymbol{x},n^{(1)}}$, the noiseless equivalent to $\boldsymbol{H}_{\boldsymbol{y},n^{(1)}}$, can be expressed as

$$\boldsymbol{H}_{\boldsymbol{x},n^{(1)}} = \boldsymbol{Z}_{\mathrm{L}}^{(2)}\boldsymbol{A}\boldsymbol{Z}_{\mathrm{D}}^{(1)^{n^{(1)}}}\boldsymbol{Z}_{\mathrm{R}}^{(2)}. \tag{2.18}$$

This then enables the noiseless enhanced matrix to be decomposed as follows:

$$\boldsymbol{E_X} = \boldsymbol{E}_{\mathrm{L}}\boldsymbol{A}\boldsymbol{E}_{\mathrm{R}}, \tag{2.19a}$$

$$\mathbb{C}^{L^{(1)}L^{(2)}\times M} \ni \boldsymbol{E}_{\mathrm{L}} = \begin{bmatrix} \boldsymbol{Z}_{\mathrm{L}}^{(2)} \\ \boldsymbol{Z}_{\mathrm{L}}^{(2)}\boldsymbol{Z}_{\mathrm{D}}^{(1)} \\ \vdots \\ \boldsymbol{Z}_{\mathrm{L}}^{(2)}\boldsymbol{Z}_{\mathrm{D}}^{(1)^{L^{(1)}-1}} \end{bmatrix}, \tag{2.19b}$$

$$\mathbb{C}^{M\times\left(N^{(1)}-L^{(1)}+1\right)\left(N^{(2)}-L^{(2)}+1\right)} \ni \boldsymbol{E}_{\mathrm{R}} = \begin{bmatrix} \boldsymbol{Z}_{\mathrm{R}}^{(2)} & \boldsymbol{Z}_{\mathrm{D}}^{(1)}\boldsymbol{Z}_{\mathrm{R}}^{(2)} & \cdots & \boldsymbol{Z}_{\mathrm{D}}^{(1)^{N^{(1)}-L^{(1)}}}\boldsymbol{Z}_{\mathrm{R}}^{(2)} \end{bmatrix}. \tag{2.19c}$$

As was the case in the MPM, SVD can be utilised to generate a filtered matrix $\widetilde{\boldsymbol{E}}_{\boldsymbol{Y}}$ with its rank reduced to M, in accordance with the EYM theorem:

$$\widetilde{\boldsymbol{E}}_{\boldsymbol{Y}} = \boldsymbol{U}_M\boldsymbol{\Sigma}_M\boldsymbol{V}_M^{\dagger} \tag{2.20}$$

Due to the large size of the enhanced matrix, a vast improvement in the speed of the MEMPM can be realised when, rather than compute the SVD in its entirety, a *truncated* SVD is computed, in which only the first M components of the SVD are determined using iterative approaches like

the Rayleigh-Ritz/Arnoldi method [103, 104].

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

$$\boldsymbol{U}_M = \boldsymbol{E}_{\mathrm{L}} \boldsymbol{T}. \tag{2.21}$$

Now consider the following two matrices:

$$\boldsymbol{U}_{M1} = \boldsymbol{E}_{\mathrm{L1}} \boldsymbol{T}, \tag{2.22a}$$

$$\boldsymbol{U}_{M2} = \boldsymbol{E}_{\mathrm{L1}} \boldsymbol{Z}_{\mathrm{D}}^{(1)} \boldsymbol{T}, \tag{2.22b}$$

$$\mathbb{C}^{L^{(1)}\left(L^{(2)}-1\right) \times M} \ni \boldsymbol{E}_{\mathrm{L1}} = \begin{bmatrix} \boldsymbol{Z}_{\mathrm{L}}^{(2)} \\ \boldsymbol{Z}_{\mathrm{L}}^{(2)} \boldsymbol{Z}_{\mathrm{D}}^{(1)} \\ \vdots \\ \boldsymbol{Z}_{\mathrm{L}}^{(2)} \boldsymbol{Z}_{\mathrm{D}}^{(1)^{L^{(1)}-2}} \end{bmatrix}. \tag{2.22c}$$

$\boldsymbol{E}_{\mathrm{L1}}$ is derived from $\boldsymbol{E}_{\mathrm{L}}$ through the removal of its last $L^{(2)}$ rows. As such, $\boldsymbol{U}_{M1}$ and $\boldsymbol{U}_{M2}$ correspond the $\boldsymbol{U}_M$ with the last and first $L^{(2)}$ rows removed, respectively. The matrix pencil for $\boldsymbol{U}_{M1}$ and $\boldsymbol{U}_{M2}$ can be expressed as

$$\boldsymbol{U}_{M1} - \lambda \boldsymbol{U}_{M2} = \boldsymbol{E}_{\mathrm{L1}} \left(\boldsymbol{Z}_{\mathrm{D}}^{(1)} - \lambda \boldsymbol{I}_M \right) \boldsymbol{T}. \tag{2.23}$$

As seen previously, this matrix structure implies that the signal poles in the first dimension $\boldsymbol{z}^{(1)}$ are the solutions of the generalised eigenvalue problem for $\boldsymbol{U}_{M1} - \lambda \boldsymbol{U}_{M2}$, such that they are the eigenvalues of $\boldsymbol{U}_{M1}^{+} \boldsymbol{U}_{M2}$.

To extract the signal poles in the other dimension, $\boldsymbol{z}^{(2)}$, the permutation matrix $\boldsymbol{P} \in \mathbb{R}^{L^{(1)}L^{(2)} \times L^{(1)}L^{(2)}}$

is defined:

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{e}(1) \\ \boldsymbol{e}\left(1 + L^{(2)}\right) \\ \vdots \\ \boldsymbol{e}\left(1 + \left(L^{(1)} - 1\right) L^{(2)}\right) \\ \boldsymbol{e}(2) \\ \boldsymbol{e}\left(2 + L^{(2)}\right) \\ \vdots \\ \boldsymbol{e}\left(2 + \left(L^{(1)} - 1\right) L^{(2)}\right) \\ \vdots \\ \vdots \\ \boldsymbol{e}\left(L^{(2)}\right) \\ \boldsymbol{e}\left(2L^{(2)}\right) \\ \vdots \\ \boldsymbol{e}\left(L^{(1)} L^{(2)}\right) \end{bmatrix}. \tag{2.24}$$

$\boldsymbol{e}(i) \in \mathbb{R}^{L^{(1)} L^{(2)}}$ corresponds to a unit row vector comprising zeros except for $e_i = 1$. Multiplying $\boldsymbol{E}_{\mathrm{L}}$ by the permutation matrix leads to a matrix in which the roles of the two sets of signal poles are effectively swapped:

$$\boldsymbol{E}_{\mathrm{LP}} \coloneqq \boldsymbol{P}\boldsymbol{E}_{\mathrm{L}} = \begin{bmatrix} \boldsymbol{Z}_{\mathrm{L}}^{(1)} \\ \boldsymbol{Z}_{\mathrm{L}}^{(1)} \boldsymbol{Z}_{\mathrm{D}}^{(2)} \\ \vdots \\ \boldsymbol{Z}_{\mathrm{L}}^{(1)} {\boldsymbol{Z}_{\mathrm{D}}^{(2)}}^{L^{(2)}-1} \end{bmatrix}. \tag{2.25}$$

Note the similarity of Equation 2.25 with Equation 2.19b, which implies that with the same reasoning as above, $\boldsymbol{z}^{(2)}$ can be derived by extracting the eigenvalues of $\boldsymbol{U}_{M\mathrm{P}1}^{+} \boldsymbol{U}_{M\mathrm{P}2}$, where $\boldsymbol{U}_{M\mathrm{P}1}$ and $\boldsymbol{U}_{M\mathrm{P}2}$ correspond to $\boldsymbol{P}\boldsymbol{U}_M$ with the last and first $L^{(1)}$ rows removed, respectively.

In the original account on the MEMPM, the final stage employed a pairing algorithm in order to assign the uncorrelated signal poles in $\boldsymbol{z}^{(1)}$ with those in $\boldsymbol{z}^{(2)}$ [73]. The *modified* MEMPM (MMEMPM) was developed in order to overcome two issues with the pairing algorithm: (a) it is computationally expensive, and (b) it is prone to return incorrect pairings [74]. The procedure for extracting the paired poles differs slightly depending on whether all the signal poles in $\boldsymbol{z}^{(1)}$ are distinct or not. As well as the eigenvalues of $\boldsymbol{U}_{M1}^{+} \boldsymbol{U}_{M2}$, the MMEMPM requires the corresponding eigenvectors too, contained in the matrix $\boldsymbol{W}^{(1)}$. Assuming that there are no repeated poles in $\boldsymbol{z}^{(1)}$[§], the following holds:

$$\boldsymbol{Z}_{\mathrm{D}}^{(2)} \equiv \boldsymbol{G} \coloneqq {\boldsymbol{W}^{(1)}}^{-1} \boldsymbol{U}_{M\mathrm{P}1}^{+} \boldsymbol{U}_{M\mathrm{P}2} \boldsymbol{W}^{(1)}, \tag{2.26}$$

[§]For the purposes of FID estimation, it is deemed that a given pair $i, j \in \{1, \cdots, M\}$ of poles is repeated if their frequencies satisfy $\left| f_i^{(1)} - f_j^{(1)} \right| < f_{\mathrm{sw}}^{(1)} / N^{(1)}$ (recall Equation 2.13c, which states the relationship between a signal pole and its frequency).

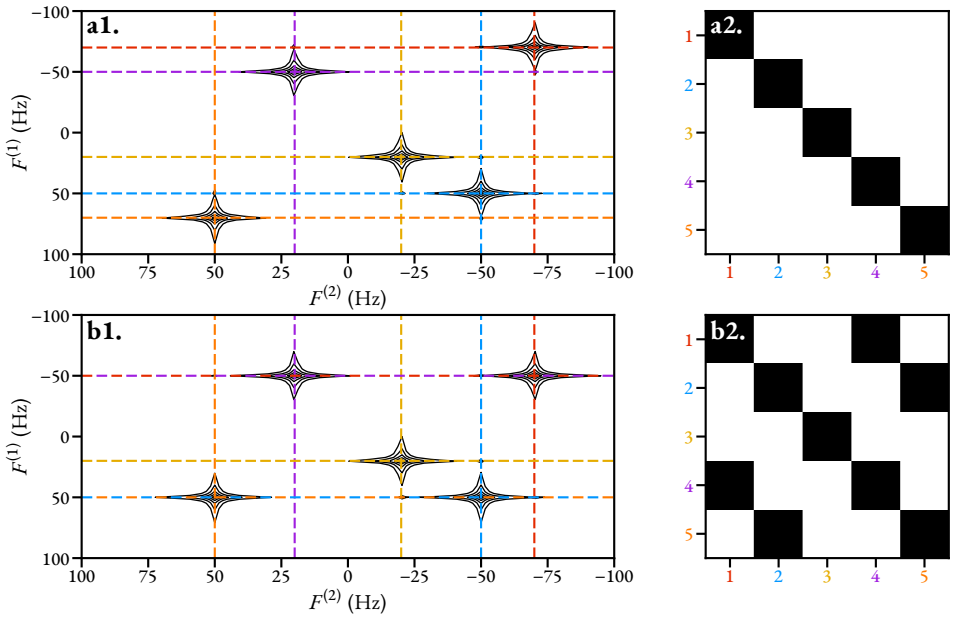


Figure 2.1: A comparison of the structure of the matrix $\boldsymbol{G}$ from the MMEMPM (Equation 2.26), for the cases of an FID with **a.** distinct signal poles in $F^{(1)}$ and **b.** repeated signal poles. **a1.** Magntiude-mode spectrum of a hypercomplex 2D FID, with $M = 5$, and all indirect-dimension frequencies having distinct values. **b1.** An equivalent spectrum with two pairs of the 5 indirect-dimension frequencies being identical ($\{1, 4\}$ and $\{2, 5\}$). **a2.** A representation of the matrix $|\boldsymbol{G}|$ for the dataset depicted in a1. All values which are greater than 10^{-10} are coloured black, while those that are less than 10^{-10} are white, though note that all the black values are $\gg 10^{-10}$. **b2.** An analogous representation for the dataset in b1. Note that $\boldsymbol{G}$ is no longer diagonal, but possesses non-zero off-diagonal elements in agreement with signals with equivalent poles. With an appropriate re-ordering of the rows and columns (i.e. swapping the rows and columns of 2 and 4), this could be recast as a block-diagonal matrix of the form in Equation 2.27, featuring 3 blocks, one which is 1×1, and two which are 2×2.

such that the second-dimension signal poles can be extracted from the main diagonal of $\boldsymbol{G}$ (see Figures 2.1.a1 and 2.1.a2). If some values in $\boldsymbol{z}^{(1)}$ are repeated, the matrix $\boldsymbol{G}$ is no longer diagonal, but instead and can be expressed as block-diagonal; assuming there are $R < M$ unique signal poles $\boldsymbol{G}$ may be expressed in the following form, allowing for a shuffling of rows and columns:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{G}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{G}_R \end{bmatrix}, \tag{2.27}$$

where $\boldsymbol{G}_r \in \mathbb{C}^{h_r \times h_r}$ is a sub-matrix corresponding to the r^{th} unique signal pole with multiplicity h_r (see Figures 2.1.b1 and 2.1.b2). Note that $\sum_{r=1}^{R} h_r = M$. When $h_r > 1$, the subset of poles $\boldsymbol{z}_r^{(2)}$ are determined by computing the eigenvalues of the matrix $\boldsymbol{G}_r$

With the signal poles in both dimensions determined, the complex amplitudes are finally computed as follows:

$$\boldsymbol{\alpha} = \operatorname{diag}\left(\boldsymbol{E}_{\mathrm{L}}^{+} \boldsymbol{E}_{\boldsymbol{Y}} \boldsymbol{E}_{\mathrm{R}}^{+}\right). \tag{2.28}$$

See Algorithm A.1 for a pseudo-code outline, and Code Listing B.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/has been predicted. It is possible that an individual inspecting the NMR 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. On top of this, scenarios may crop up where a user is unable to recognise that certain heavily overlapping peaks in the spectrum constitute more than one signal, leading to under-estimates of M. 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) [105] and minimum description length (MDL) [106, 107]. 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:

$$\mathcal{C}(k) = -c \ln\left(\mathcal{L}\left(\boldsymbol{\theta}^{(*)} | \boldsymbol{y}\right)\right) + \mathcal{P}(k) \text{ with } \boldsymbol{\theta}^{(*)} \in \mathbb{R}^{4k} \quad \forall k \in \{0, 1, \cdots\}. \tag{2.29}$$

$\mathcal{L}\left(\boldsymbol{\theta}^{(*)}|\boldsymbol{y}\right)$ is the likelihood function of the model with order k at the MLE, $c \in \mathbb{R}_{>0}$ 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 a model which also features noise components. The penalising term $\mathcal{P}(k)$, which always increases with k, is introduced in order to obtain a parsimonious model order estimate. Wax and Kailath derived an expression for the likelihood at the MLE for models comprising a summation of complex sinusoids [108]¶:

$$\mathcal{L}\left(\boldsymbol{\theta}^{(*)} \in \mathbb{R}^{4k}|\boldsymbol{y}\right) = \left(\frac{\prod_{r=k}^{L-1} \sigma_{r+1}^{1/L-k}}{\frac{1}{L-k}\sum_{r=k}^{L-1}\sigma_{r+1}}\right)^{(L-k)N} \qquad \forall k \in \{0, 1, \cdots, L-1\}. \tag{2.30}$$

$\boldsymbol{\sigma} \in \mathbb{R}^{L}$ is the set of singular values of $\boldsymbol{H}_{\boldsymbol{y}}$, in decreasing order. The explicit forms of the AIC and MDL are

$$\mathrm{AIC}(k) = -2\ln\left(\mathcal{L}\left(\boldsymbol{\theta}^{(*)}|\boldsymbol{y}\right)\right) + 2k(2L-k), \tag{2.31a}$$

$$\mathrm{MDL}(k) = -\ln\left(\mathcal{L}\left(\boldsymbol{\theta}^{(*)}|\boldsymbol{y}\right)\right) + \tfrac{1}{2}k(2L-k)\ln N. \tag{2.31b}$$

The AIC has been shown to be inconsistent since it tends to overestimate the model order as the number of samples increases [108]. For this reason, the MDL has found greater favour in signal processing applications, and is employed in this work‖:

$$M = \underset{k\in\mathbb{N}_0:\ k<L}{\arg\min}\ \mathrm{MDL}(k). \tag{2.32}$$

Applying the MDL for model order selection, and subsequently using the MPM for parameter estimation is the basis of the ITMPM from Pines and co-workers [70]. Figure 2.2 illustrates the form of the MDL for three FIDs with equivalent underlying models, comprising 7 signals, but with noise instances of differing variances. The first 14 singular values of $\boldsymbol{H}_{\boldsymbol{y}}$ are plotted in Figure 2.2.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 smaller in magnitude and more consistent, such that distinguishing the noise and signal subspaces is an easier task (cf. the yellow and red lines). As such, it

¶The expression in Wax and Kailath's paper considers the eigenvalues of the covariance matrix for the signal, rather than the singular values of $\boldsymbol{H}_{\boldsymbol{y}}$; these are equivalent.

‖In practice, rather than compute the global minimum of the MDL, the first relative minimum is determined (see Algorithm 2.1, Line 19). When k is very large, it has been noticed that the MDL can spuriously jump in value. This often occurs when FIDs with very high SNRs are analysed; the singular values involved can tend so closely to zero that issues related to the instability of floating-point arithmetic manifest.

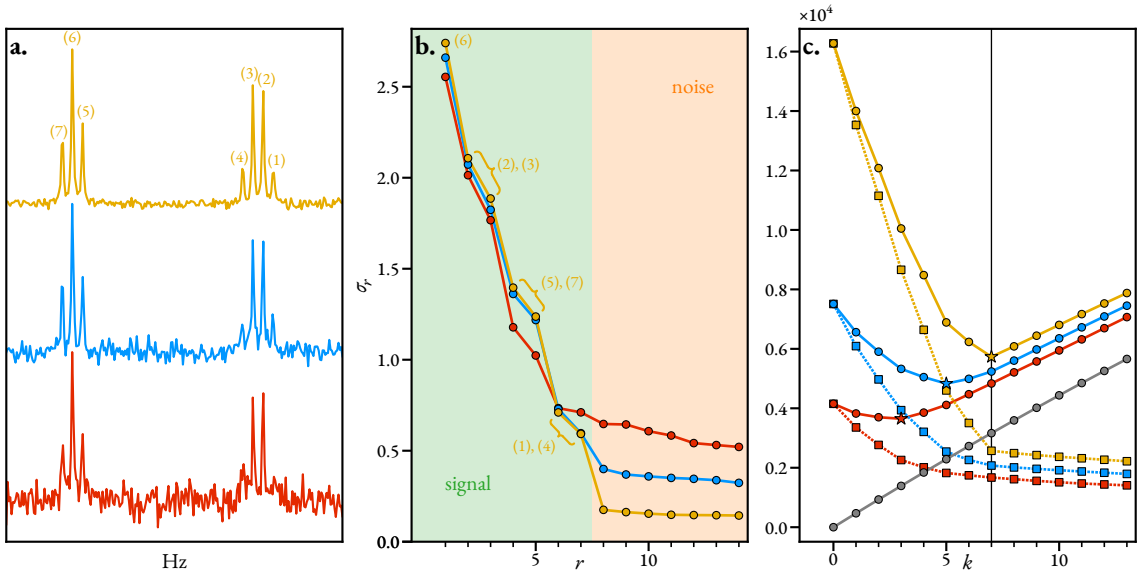


Figure 2.2: A visualisation of the behaviour of the MDL for three different FIDs comprising the same deterministic component ($\boldsymbol{x}$) but with noise instances of differing variances. The model used to construct the FIDs features 7 signals. The three SNRs used were 7 dB (red), 12 dB (blue), 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 $\boldsymbol{H_y}$, 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 2.31b. Grey line: the penalty component of the MDL, i.e. the second term in Equation 2.31b. 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 model orders are 5 and 3 for the 12 dB and 7 dB FIDs, respectively).

is perhaps unsurprising that the MDL is more likely to provide a faithful estimate of the FID's model order when the SNR is higher (see Figure 2.2.c).

2.3 Non-Linear Programming

In the devised estimation routine, the result generated using the MPM, with optional model order selection, is subsequently subjected to an NLP routine to yield the final parameter estimate. In this section, the theory behind the NLP routine is provided, along with motivation for why it is employed.

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 minimum is found:

$$\boldsymbol{\theta}^{(*)} = \underset{\boldsymbol{\theta} \in \mathbb{R}^n}{\arg\min}\, \mathcal{F}(\boldsymbol{\theta}). \tag{2.33}$$

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 for high-dimensionality problems such as the one of interest here. $\boldsymbol{\theta}^{(*)}$ is a local minimiser of $\mathcal{F}(\boldsymbol{\theta})$ if there is a neighbourhood $V \ni \boldsymbol{\theta}^{(*)}$ for which

$$\mathcal{F}\left(\boldsymbol{\theta}^{(*)}\right) \leq \mathcal{F}(\boldsymbol{\theta}) \; \forall \boldsymbol{\theta} \in V. \tag{2.34}$$

$V \subset \mathbb{R}^n$ is a continuous space such that it is possible to move a finite 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 the fidelity:

- 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 $\boldsymbol{g}\left(\boldsymbol{\theta}^{(*)}\right) := \nabla\mathcal{F}\left(\boldsymbol{\theta}^{(*)}\right)$ is the zero vector:

$$\boldsymbol{g}\left(\boldsymbol{\theta}^{(*)}\right) = \mathbf{0} \in \mathbb{R}^n \tag{2.35}$$

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

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

[‡‡]"Extremum" is used here instead of "minimum", as the first necessary condition applies to maxima of a function as well as minima.

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

$$\boldsymbol{v}^{\mathrm{T}}\boldsymbol{H}\left(\boldsymbol{\theta}^{(*)}\right)\boldsymbol{v} \geq 0 \;\forall \boldsymbol{v} \in \mathbb{R}^n. \tag{2.36}$$

- Furthermore, $\boldsymbol{\theta}^{(*)}$ is a *unique* local minimiser if the *second-order sufficient condition* is also satisfied, i.e. that the Hessian is positive definite:

$$\boldsymbol{v}^{\mathrm{T}}\boldsymbol{H}\left(\boldsymbol{\theta}^{(*)}\right)\boldsymbol{v} > 0 \;\forall \boldsymbol{v} \in \mathbb{R}^n. \tag{2.37}$$

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

$$\mathcal{F}_Q(\boldsymbol{\theta}) = \mathcal{F}\left(\boldsymbol{\theta}^{(k)}\right) + \boldsymbol{h}^{\mathrm{T}}\boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right) + \tfrac{1}{2}\boldsymbol{h}^{\mathrm{T}}\boldsymbol{H}\left(\boldsymbol{\theta}^{(k)}\right)\boldsymbol{h}, \tag{2.38}$$

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

$$\begin{aligned} \frac{\partial\mathcal{F}(\boldsymbol{\theta})}{\partial\boldsymbol{h}} &= \boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right) + \boldsymbol{H}\left(\boldsymbol{\theta}^{(k)}\right)\boldsymbol{h} \\ \implies 0 &= \boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right) + \boldsymbol{H}\left(\boldsymbol{\theta}^{(k)}\right)\left(\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\right) \\ \therefore \boldsymbol{\theta}^{(k+1)} &= \boldsymbol{\theta}^{(k)} - \boldsymbol{H}\left(\boldsymbol{\theta}^{(k)}\right)^{-1}\boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right). \end{aligned} \tag{2.39}$$

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

$$\left\|\boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right)\right\| \leq \epsilon. \tag{2.40}$$

The convergence threshold $\epsilon > 0$ can be tuned based on the desired accuracy of the result. Equation 2.39 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 does not lead to a decrease in the fidelity if the Hessian is not positive definite. Two primary strategies have emerged which are typically used instead:

- *Line search methods* [93: Chapter 3] determine an appropriate direction $\boldsymbol{p}^{(k)}$ 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 — leading to $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \alpha^{(k)}\boldsymbol{p}^{(k)}$.

- *Trust region methods* [93: Chapter 4] define a radius $\Delta^{(k)} > 0$, and determine the minimum of Equation 2.38 subject to the constraint that $\|\boldsymbol{h}\| \leq \Delta^{(k)}$.

Algorithm 2.2 The NLP routine employed in this work. This makes use of Algorithms 4.1 & 7.2 in [93], with a extra check inserted to deal with any negative-amplitude oscillators which may be generated as the routine evolves (Lines 19 to 22).

1: **procedure** NLP($\boldsymbol{Y} \in \mathbb{C}^{N^{(1)}\times\cdots\times N^{(D)}}, \boldsymbol{\theta}^{(0)} \in \mathbb{R}^{2(D+1)M}$)
2: $\quad\Delta^{(0)} \leftarrow 1/10 \left\|\boldsymbol{g}\left(\boldsymbol{\theta}^{(0)} \mid \boldsymbol{Y}\right)\right\|$;
3: $\quad\Delta_{\max} \leftarrow 16\Delta^{(0)}$;
4: $\quad$**for** $k = 0, 1, \cdots$ **do**
5: $\quad\quad\boldsymbol{p}^{(k)} \leftarrow \textsc{SteihaugToint}\left(\boldsymbol{Y}, \boldsymbol{\theta}^{(k)}, \Delta^{(k)}\right)$; ▷ See Algorithm A.2
6: $\quad\quad\rho^{(k)} \leftarrow \frac{\mathcal{F}_{\phi}\left(\boldsymbol{\theta}^{(k)}\right) - \mathcal{F}_{\phi}\left(\boldsymbol{\theta}^{(k)}+\boldsymbol{p}^{(k)}\right)}{\mathcal{F}_{\phi Q}\left(\boldsymbol{\theta}^{(k)}\right) - \mathcal{F}_{\phi Q}\left(\boldsymbol{\theta}^{(k)}+\boldsymbol{p}^{(k)}\right)}$;
7: $\quad\quad$**if** $\rho_k < 1/4$ **then**
8: $\quad\quad\quad\Delta^{(k+1)} \leftarrow 1/4\Delta^{(k)}$;
9: $\quad\quad$**else if** $\rho_k > 3/4$ **and** $\left\|\boldsymbol{p}^{(k)}\right\| = \Delta^{(k)}$ **then**
10: $\quad\quad\quad\Delta^{(k+1)} \leftarrow \min\left(2\Delta^{(k)}, \Delta_{\max}\right)$;
11: $\quad\quad$**else**
12: $\quad\quad\quad\Delta^{(k+1)} \leftarrow \Delta^{(k)}$;
13: $\quad\quad$**end if**
14: $\quad\quad$**if** $\rho^{(k)} > 3/20$ **then**
15: $\quad\quad\quad\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + \boldsymbol{p}^{(k)}$;
16: $\quad\quad$**else**
17: $\quad\quad\quad\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)}$;
18: $\quad\quad$**end if**
19: $\quad\quad$**if** $k \bmod 25 = 0$ **and** $\boldsymbol{\theta}^{(k+1)}$ contains negative amplitudes **then**
20: $\quad\quad\quad\boldsymbol{\theta}^{(0)} \leftarrow \boldsymbol{\theta}^{(k+1)}$ with negative-amplitude oscillators removed;
21: $\quad\quad\quad\boldsymbol{\theta}^{(*)}, \boldsymbol{\epsilon}^{(*)} \leftarrow \text{NLP}\left(\boldsymbol{Y}, \boldsymbol{\theta}^{(0)}\right)$;
22: $\quad\quad$**end if**
23: $\quad\quad$**if** $\left\|\boldsymbol{g}\left(\boldsymbol{\theta}^{(k+1)}\right)\right\| < 10^{-8}$ **then**
24: $\quad\quad\quad$**break**;
25: $\quad\quad$**end if**
26: $\quad$**end for**
27: $\quad\boldsymbol{\theta}^{(*)} \leftarrow \boldsymbol{\theta}^{(k+1)}$;
28: $\quad\boldsymbol{\epsilon}^{(*)} \leftarrow \sqrt{\frac{\mathcal{F}\left(\boldsymbol{\theta}^{(*)}\right) \operatorname{diag}\left(\left[\boldsymbol{H}\left(\boldsymbol{\theta}^{(*)}\right)\right]^{-1}\right)}{(N^{(1)}\cdots N^{(D)})-1}}$; ▷ See Appendix A.2.2.
29: $\quad$**return** $\boldsymbol{\theta}^{(*)}, \boldsymbol{\epsilon}^{(*)}$;
30: **end procedure**

A trust region method is applied in this work, with the typical steps involved given in Algorithm 2.2 (ignoring Lines 19 to 22, which is a custom addition, see Section 2.3.4). An initial radius for the trust region $\Delta^{(0)}$ is defined, along with a maximum permitted radius $\Delta_{\max}$, to ensure that excessively adventurous steps do not take place. For each iteration, a solution to the following subproblem is sought:

$$\boldsymbol{p}^{(k)} = \underset{\boldsymbol{p}\in\mathbb{R}^n}{\arg\min}\, \mathcal{F}\left(\boldsymbol{\theta}^{(k)}\right) + \left(\boldsymbol{\theta}^{(k)} + \boldsymbol{p}\right)^{\mathrm{T}} \boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right) + \tfrac{1}{2}\left(\boldsymbol{\theta}^{(k)} + \boldsymbol{p}\right)^{\mathrm{T}} \boldsymbol{H}\left(\boldsymbol{\theta}^{(k)}\right)\left(\boldsymbol{\theta}^{(k)} + \boldsymbol{p}\right) \quad \text{subject to } \|\boldsymbol{p}\| \leq \Delta^{(k)}. \tag{2.41}$$

This sub-problem is not usually solved exactly, but instead an efficient means of determining a sufficiently good update is used. Common approaches include computing the Cauchy point, the Dogleg method, and a truncated conjugate-gradient approach commonly called the Steihaug-Toint (ST) method [93: Chapter 7]. The latter is employed in this work (see Algorithm A.2 and Code Listing B.5). In the ST approach, iterates of the conjugate-gradient method [93: 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 $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \boldsymbol{p}^{(k)}$ is determined using the ST method, a metric is considered which indicates how well the quadratic estimate agrees with the true value of the fidelity:

$$\rho = \frac{\mathcal{F}\left(\boldsymbol{\theta}^{(k)}\right) - \mathcal{F}\left(\boldsymbol{\theta}^{(k)} + \boldsymbol{p}^{(k)}\right)}{\mathcal{F}_Q\left(\boldsymbol{\theta}^{(k)}\right) - \mathcal{F}_Q\left(\boldsymbol{\theta}^{(k)} + \boldsymbol{p}^{(k)}\right)}. \tag{2.42}$$

ρ 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 ρ is sufficiently large, the quadratic model being used to generate a new iterate is deemed to be acting well enough to warrant accepting the proposed update (Lines 14 and 15). Furthermore, if ρ 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 order to increase the rate of convergence (Lines 9 and 10). On the other hand, a small value of ρ 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 presented in this work.

For the specific case of FID estimation using NLP, consider a general D-dimensional dataset. As established in Section 2.1, the fidelity $\mathcal{F}(\boldsymbol{\theta} \,|\, \boldsymbol{Y}) : \mathbb{C}^{N^{(1)} \times \cdots \times N^{(D)}} \times \mathbb{R}^{2(1+D)M} \to \mathbb{R}$ is given by

$$\mathcal{F}(\boldsymbol{\theta} \,|\, \boldsymbol{Y}) = \|\boldsymbol{Y} - \boldsymbol{X}(\boldsymbol{\theta})\|^2. \tag{2.43}$$

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

$$g_i = -2\Re\left\langle (\boldsymbol{Y} - \boldsymbol{X}), \frac{\partial \boldsymbol{X}}{\partial \theta_i} \right\rangle, \tag{2.44a}$$

$$h_{i,j} = 2\Re\Bigg(\underbrace{\left\langle \frac{\partial \boldsymbol{X}}{\partial \theta_i}, \frac{\partial \boldsymbol{X}}{\partial \theta_j} \right\rangle}_{①} - \underbrace{\left\langle (\boldsymbol{Y} - \boldsymbol{X}), \frac{\partial^2 \boldsymbol{X}}{\partial \theta_i \partial \theta_j} \right\rangle}_{②} \Bigg), \tag{2.44b}$$

Dimensions	# 1st derivatives	# 2nd 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	$2(1+D)MN_{\text{tot}}$	$(2D^2+5D+2)MN_{\text{tot}}$

Table 2.1: The number of first and second derivatives that are necessary to compute the gradient vector and Hessian matrix of the fidelity for 1- 2- and 3-dimensional datasets, as well as a general D-dimensional dataset.

$$\forall i, j \in \{1, \cdots, 2(1+D)M\}.$$

Computing these requires the generation of a large number of first and second partial derivatives; the complete set of these are given in Appendix A.2.1, and a PYTHON implementation to compute them is given in Code Listing B.6. Fortunately, fewer second partial derivatives need to be computed than one might initially suspect; $4(1+D)^2M^2$ of these exist per datapoint. However, all of those involving parameters associated with different oscillators are zero. The second derivatives are also symmetric, i.e.

$$\frac{\partial^2 x_{n^{(1)},\cdots,n^{(D)}}}{\partial\theta_i\partial\theta_j} \equiv \frac{\partial^2 x_{n^{(1)},\cdots,n^{(D)}}}{\partial\theta_j\partial\theta_i},$$

which further reduces the number of explicit computations that are necessary. Finally, the derivatives satisfying $\theta_i = \theta_j = a_m$ are always zero as well. The upshot of these features is that the total number of second derivatives that need to be computed is reduced to $(2D^2+5D+2)M$ per datapoint (see Table 2.1). This is still rather large, especially when $D > 1$, as will be evidenced in Section 2.4.

The Hessian matrix at convergence can be employed to determine errors associated with the parameter estimates. More detail is provided in Appendix A.2.2.

2.3.2 Approximating the Hessian

Despite many of the model second derivatives being zero, computation of those that are not zero, and subsequently using these to form the Hessian matrix, is an expensive part of the optimisation routine. Numerous optimisation problems exist where this is the case; as such, there is considerable precedent for improving the efficiency of optimisation algorithms by generating approximations of the Hessian which are less demanding to compute. Examples include the Gauss-Newton (GN) method and LM algorithm, which are specifically for RSS problems [93: Chapter 10], as well as quasi-Newton methods such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [93: Chapter 6].

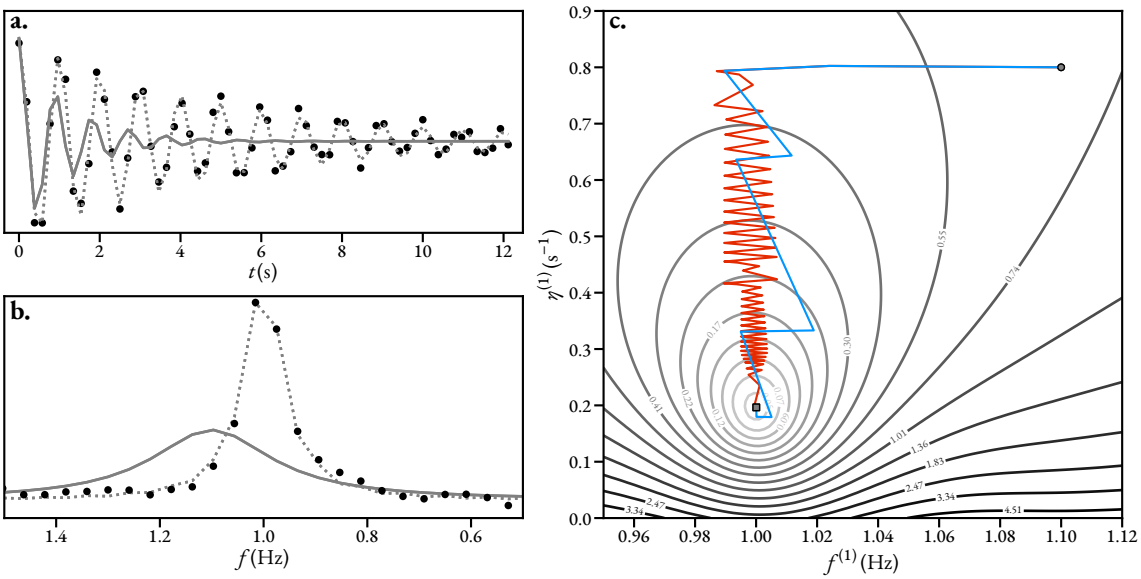


Figure 2.3: A visualisation of the trajectory of a 2-parameter optimisation involving a simulated FID comprising a single signal. **a.** and **b.** Representations of the signal in the time domain and Fourier domain, respectively. Black dots: the FID to be estimated. Solid grey line: the model generated using the initial guess $\boldsymbol{\theta}^{(0)}$. Dotted grey line: the model generated using the optimised result $\boldsymbol{\theta}^{(*)}$. **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 GN and LM approaches replace the true Hessian matrix at each iteration with the following expression:

$$h_{i,j} = 2\Re\left\langle \frac{\partial \boldsymbol{X}}{\partial \theta_i}, \frac{\partial \boldsymbol{X}}{\partial \theta_j} \right\rangle, \tag{2.45}$$

i.e. term ② in Equation 2.44b, which involves the model second derivatives, is neglected. All that needs to be generated is the *Jacobian* $\partial \boldsymbol{X}/\partial \boldsymbol{\theta}$. This can bring a very large reduction in the computational cost; no extra derivatives need to be computed for the Hessian at all since the Jacobian is already required to generate 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 a local minimum. Despite this, by invoking this approximation, the rate of convergence (i.e. the number of iterations required to reach $\boldsymbol{\theta}^{(*)}$) tends to be adversely affected, as is highlighted by the following example.

2.3.3 Visualisation of a Simple Example

Figure 2.3 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 1.24 with $M = 1$, $N = 64$,

$f_{\text{sw}} = 5.2\,\text{Hz}$ ($\Delta_t \approx 0.192\,\text{s}$), and $f_{\text{off}} = 0\,\text{Hz}$. It was constructed using the parameters $\boldsymbol{\theta} \in \mathbb{R}^4$ comprising $a = 1$, $\phi = 0\,\text{rad}$, $f = 1\,\text{Hz}$, and $\eta = 0.2\,\text{s}^{-1}$. AWGN was added to the FID to give it a target an SNR of 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 $\boldsymbol{\theta}^{(0)}$; the amplitude and phase were fixed to their true values throughout. The initial guess comprised a frequency of 1.1 Hz, and a damping factor of $0.8\,\text{s}^{-1}$, with the solid grey lines in Figures 2.3.a and 2.3.b denoting the model generated in the time- and Fourier-domains, respectively. $\boldsymbol{\theta}^{(0)}$ was subjected to NLP twice. In the first instance, the exact Hessian matrix given by Equation 2.44b was used in order to compute each update step, while in the second the Hessian approximation given by Equation 2.45 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 [109]. The trajectories of the parameter vector are denoted by blue and red lines in Figure 2.3.c. In both cases, the NLP routine successfully converged at a result $\boldsymbol{\theta}^{(*)}$ 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 trajectory) led to a far better rate of convergence compared with the approximated analogue (red trajectory), which exhibited behaviour commonly referred to as "zig-zagging"; this is often seen in gradient descent methods, in which each update occurs along the opposite direction to the gradient, and leads to the minimum being reached in an inefficient manner. 14 iterations were required to achieve convergence with the criterion $\epsilon = 10^{-8}$ 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, especially for 2D FIDs, as will be illustrated in Section 2.4.

Remark 2. *Whenever simulated NMR data is corrupted with noise in later examples in this thesis, the type of noise is always AWGN.*

2.3.4 Phase Variance Minimisation

An NLP procedure tasked with minimising the discrepancy between the model $\boldsymbol{X}$ and the observed data $\boldsymbol{Y}$ is well-suited to produce an accurate holistic representation of the data, assuming a sufficiently large model order is used. However, as has already been discussed, this does not necessarily lead to a satisfactory result due to the ill-posed nature of the estimation problem; it is desirable to produce an estimate which not only achieves a good fit to the data in an RSS sense, but which is also in agreement with the process underpinning the observation. It is for this reason that iterative procedures typically require significant quantities of prior knowledge, beyond basic assumptions of the underlying model, in order to produce meaningful estimation results. This is also why they are often able to produce results which agree better with a spectroscopist's concep-

tion of what the "correct" parameter estimate should look like, relative to other methods where such detailed information is not exploited.

While the MPM is often able to generate seemingly reasonable parameter estimates, one particular feature of these has been noticed on many occasions; oscillators in the result often exhibit "spurious phase behaviour". As has been discussed (see the description of phase correction, Section 1.2.1), NMR datasets for most experiments comprise signals whose phases depend on their frequencies to first order. This is routinely corrected in conventional NMR spectral processing, such that all signals are adjusted to acquire a phase of 0 rad. This feature can be exploited in order to overcome the aforementioned shortcoming of the MPM, through appropriate regularisation of the NLP routine. Assuming that the data has been phase corrected[§§], incorporating the variance of oscillator phases into the fidelity can lead to improved estimation results; examples of this will be provided later (Section 3.1). The updated fidelity becomes

$$\mathcal{F}_{\phi}\left(\boldsymbol{\theta} \,|\, \boldsymbol{Y}\right) = \left\|\boldsymbol{Y} - \boldsymbol{X}\left(\boldsymbol{\theta}\right)\right\|^2 + \mathrm{Var}_{\circ}(\boldsymbol{\phi}), \tag{2.46}$$

where $\mathrm{Var}_{\circ}(\boldsymbol{\phi})$ is the *circular variance* of the oscillator phases. Oscillator phases are an example of a circular variable, as all phases are wrapped within an interval of size 2π rad. Given an unwrapped phase $\widetilde{\phi} \in \mathbb{R}$, the corresponding wrapped phase $\phi \in (-\pi, \pi]$ is given by

$$\phi = (\widetilde{\phi} + \pi) \bmod 2\pi - \pi. \tag{2.47}$$

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

$$\mathrm{Var}_{\mathrm{l}}(\boldsymbol{\phi}) = \frac{1}{M}\sum_{m=1}^{M}\left(\phi_m - \mu\left(\boldsymbol{\phi}\right)\right)^2, \tag{2.48a}$$

$$\mu\left(\boldsymbol{\phi}\right) = \frac{1}{M}\sum_{m}\phi_m, \tag{2.48b}$$

unsuitable as a metric to define the variation in phases. Consider as a simple example a scenario where there are two oscillators with phases $\widetilde{\boldsymbol{\phi}} = \left[\pi + \delta \;\; \pi - \delta\right]^{\mathrm{T}}$ 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 2.47, these phases would actually be set to $\boldsymbol{\phi} = \left[-\pi + \delta \;\; \pi - \delta\right]^{\mathrm{T}}$, and the linear phase variance would be large. A definition of variance which accounts for the

[§§]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 during NLP, thus removing the need to apply phase correction in the first place. The reason why the phase variance has been chosen is two-fold. (a) Applying phase-correction to NMR data is straightforward and can be automated, meaning the user would experience minimal burden. (b) 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.

periodicity of the phases is needed; the circular variance is defined as [110: Chapter 3]

$$[0,1] \ni \mathrm{Var}_\circ(\boldsymbol{\phi}) = 1 - \frac{R}{M}, \tag{2.49a}$$

$$R = \sqrt{c_\Sigma^2 + s_\Sigma^2}, \tag{2.49b}$$

$$c_\Sigma = \sum_{m=1}^{M} \cos\phi_m, \tag{2.49c}$$

$$s_\Sigma = \sum_{m=1}^{M} \sin\phi_m. \tag{2.49d}$$

R is the length of the resultant vector produced by summing M unit vectors with 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 — this leads to there being an angle of $2\pi/M-1$ rad between all pairs of adjacent vectors — the vectors will perfectly cancel, leading to $R = 0$. In this case, the maximum possible variance (1) is obtained. The inclusion of the phase variance into the fidelity is one of the motivating reasons for normalising the data prior to estimation. Since $\mathrm{Var}_\circ(\boldsymbol{\phi})$ is constrained to the interval $[0,1]$, if the data were not normalised it is likely that $\|\boldsymbol{Y}-\boldsymbol{X}\|^2$ would dominate $\mathrm{Var}_\circ(\boldsymbol{\phi})$ in Equation 2.46, such that the influence of the phase variance would be negligible.

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

$$g_i = -2\Re\left\langle (\boldsymbol{Y}-\boldsymbol{X}), \frac{\partial \boldsymbol{X}}{\partial\theta_i}\right\rangle + \frac{\partial\,\mathrm{Var}_\circ(\boldsymbol{\phi})}{\partial\theta_i}, \tag{2.50a}$$

$$h_{i,j} = 2\Re\left(\left\langle \frac{\partial \boldsymbol{X}}{\partial\theta_i}, \frac{\partial \boldsymbol{X}}{\partial\theta_j}\right\rangle - \underbrace{\left\langle (\boldsymbol{Y}-\boldsymbol{X}), \frac{\partial^2 \boldsymbol{X}}{\partial\theta_i\partial\theta_j}\right\rangle}_{\text{Neglected if approximation used}}\right) + \frac{\partial^2\,\mathrm{Var}_\circ(\boldsymbol{\phi})}{\partial\theta_i\partial\theta_j}. \tag{2.50b}$$

The derivatives of the phase variance are given by:

$$\frac{\partial\,\mathrm{Var}_\circ(\boldsymbol{\phi})}{\partial\theta_i} = \begin{cases} \frac{1}{RM}\left(c_\Sigma \sin\phi_{i-M} - s_\Sigma\cos\phi_{i-M}\right) & M \le i < 2M \\ 0 & \text{otherwise} \end{cases} \tag{2.51a}$$

$$\frac{\partial^2 \operatorname{Var}_\circ(\boldsymbol{\phi})}{\partial\theta_i \partial\theta_j} = \begin{cases} \frac{1}{RM}\left[\frac{1}{R^2}\left(c_\Sigma \sin\phi_{i-M} - s_\Sigma \cos\phi_{i-M}\right)^2 \right. & M \leq i,j < 2M, i = j \\ \qquad \left. + c_\Sigma \cos\phi_{i-M} + s_\Sigma \sin\phi_{i-M} - 1\right] & \\ \frac{1}{RM}\left[\frac{1}{R^2}\left(c_\Sigma \sin\phi_{i-M} - s_\Sigma \cos\phi_{i-M}\right)\right. & \\ \qquad \times \left(c_\Sigma \sin\phi_{j-M} - s_\Sigma \cos\phi_{j-M}\right) & M \leq i,j < 2M, i \neq j \\ \qquad \left. - \cos\left(\phi_{i-M} - \phi_{j-M}\right)\right] & \\ 0 & \text{otherwise} \end{cases} \tag{2.51b}$$

The phase variance-regularised fidelity (Equation 2.46) is minimised according the unconstrained NLP routine described above. It is therefore possible for oscillators to acquire parameters which are unrealistic as the optimiser evolves (see the bulleted list on Page 13 for a summary of the expected ranges that the parameters reside in). With the inclusion of the variance of oscillator phases, there are situations where oscillators acquire negative amplitudes during the NLP routine. Typically, this occurs when there are oscillators in the MPM result with phases that are far from 0 rad. By acquiring a negative amplitude and a phase close to 0 rad (the expected phase of most oscillators in the parameter set if phase correction has been applied to the data) little change to the RSS term is made, while $\operatorname{Var}_\circ(\boldsymbol{\phi})$ will be reduced. The presence of such oscillators is undesirable, as they are spurious in the context of phased data. As a result, the NLP routine periodically checks for negative-amplitude oscillators (Lines 19 to 22 in Algorithm 2.2). After a given number of iterations (25 is the value given in Algorithm 2.2[¶¶]) if any oscillators have acquired negative amplitudes, they are removed from the model, and the routine continues with a reduced number of oscillators. While not infallible, this can help to to get rid of excessive oscillators in the model which do not correspond to true signals in the data. For example, in scenarios where an overestimate of model order has occurred, either by the MDL or from human error in inspecting the number of spectral peaks, noise components will be incorporated into the MPM result. These typically have greater variability in their phases; often, such oscillators gain negative amplitudes as the NLP routine evolves, yielding a parsimonious estimation result.

2.4 Profiling the Method

The routine described for FID estimation involves operations which can be computationally demanding, with the burden on resources 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

[¶¶] The choice of the number of iterations between checks for negative amplitudes is arbitrary. 25 is the default in the NMR-EsPy software, and was used to acquire the results presented in this thesis. The value was chosen as it enables the optimiser to evolve considerably before a check is made, such that any negative amplitudes that exist at the point of the first check would likely remain until convergence. Initially, the algorithm was setup to check for negative amplitudes after every iteration, but this often led to the removal of excessive numbers of oscillators from the parameter set, causing under-fits.

done by the central processing unit (CPU), and the amount of random access memory (RAM) needed to store all the required data as the routine runs. For the matrix pencil methods, the most demanding aspect is performing SVD, while for NLP, it is generation of the Hessian matrix for each iteration. Detailed accounts of the computational complexity of the MPM and MMEMPM have been presented [73, 74]. However, it is useful to consider what the actual run times of these routines are on a modern computer; a lot of relevant accounts are from decades before this work, and so the run time will have decreased considerably thanks to improvements in processing power. As an example, the account by Pines and 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 SGI Indigo workstation with 100 MHz CPU [70]. On the system used for all results generated for this work (see Remark 3) an equivalent computation takes about 100 ms. Furthermore, the consideration of larger datasets would have been impossible on the workstation used by Pines *et al.*, which supported a maximal memory capacity of 96 MiB; $N = 1024$ is the largest power of 2 for which the required amount of storage for processing using the MPM does not exceed 96 MiB, as can be seen in Figure 2.4.a2.

To derive the results presented in this section, PYTHON implementations of the aforementioned parts of the estimation routine were run, with software used to assess both the line-by-line execution times [111], and the time-dependent RAM usage [112].

Remark 3. *All results presented 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 The MPM and MMEMPM

A series of synthetic 1D FIDs was constructed, comprising 10 evenly-spaced signals, with a variable number of time-points $N \in \{512k : k \in \{1, 2, \cdots, 16\}\}$. For each FID, the MPM routine outlined in Code Listing B.3 was performed 5 times. The mean time to perform the MPM across the 5 runs is plotted as a function of N in Figure 2.4.a1, where it can be seen that for the larger values of N considered, the MPM is computed in approximately $\mathcal{O}(N^3)$ time; a fit of a cubic function of the form $aN^3 + b$ to the data satisfying $7 \leq k \leq 16$ is plotted to highlight this. The cubic dependence is realised since the rate-limiting step of the MPM is SVD of $\boldsymbol{H_y}$, whose size is to a very good approximation $\frac{2N}{3} \times \frac{N}{3}$***. For smaller values of N, a deviation away from a cubic relationship is observed. This arises because the computation of the complex amplitudes using Equation 1.41 has a comparatively significant run time relative to SVD in the low-N regime; for a 512 point signal, SVD of $\boldsymbol{H_y}$ 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%

*** The time complexity for the SVD of generic a $m \times n$ matrix is $\mathcal{O}(\min(m,n)^2 \cdot \max(m,n))$, while the space complexity is $\mathcal{O}(mn)$.

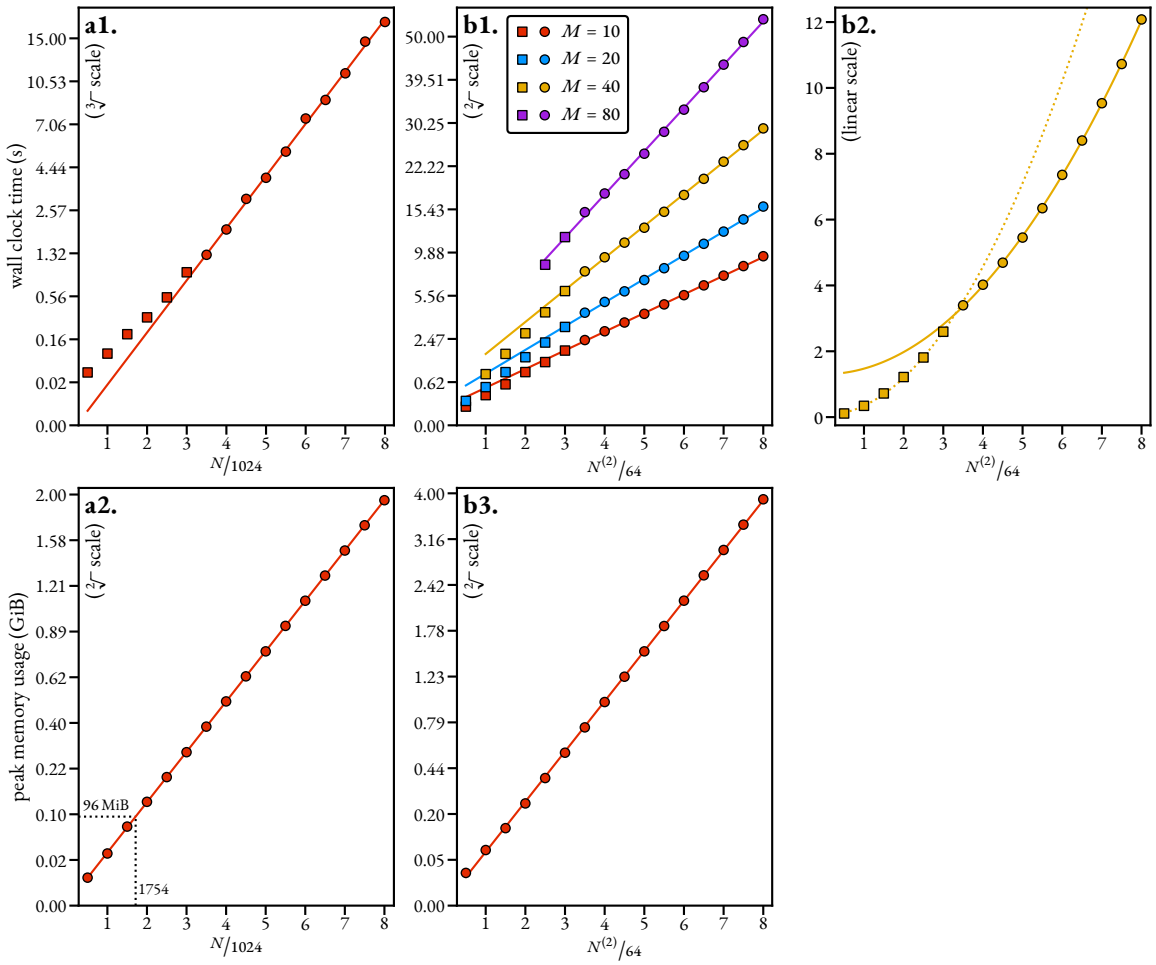


Figure 2.4: The wall clock time and peak memory usage of the MPM and MMEMPM for FIDs with differing numbers of datapoints and constituent signals. The FIDs that were used to acquire these results are described in the main text. **a1.** The amount of time required to compute the MPM, as a function of number of points. Also plotted is a cubic fit of the circular points. **a2.** Peak memory consumption in performing the MPM as a function of the number of points. The smallest number of datapoints that requires a least 96 MiB of RAM (1754) is indicated. **b1.** The run time to compute the MMEMPM of FIDs with $N^{(1)} = 64$, and variable $N^{(2)}$ and M. Circular points have been fitted to quadratic functions. **b2.** The time required to compute the SVD of $\boldsymbol{E_Y}$ for the $M = 40$ FIDs. The solid line is a quadratic fit of the circular points, while the dashed line is a quadratic fit of the square points. **b3.** Peak memory consumption in performing the MMEMPM for the $M = 40$ FIDs.

and <1%, respectively.

The MPM was run once more on each generated FID in order to assess the effect of N on the space complexity. The peak RAM consumption is plotted in Figure 2.4.a2. A clear quadratic dependence on consumption is realised as function of N, again in agreement with the expected space complexity of the SVD***.

A similar study was conducted for the consideration of the MMEMPM. A series of hypercomplex 2D FIDs were simulated, all of which all comprised $N^{(1)} = 64$. The FIDs possessed values of $N^{(2)} \in \{32k : k \in \{1, \cdots, 16\}\}$. With the MPM, since a complete SVD of the matrix $\boldsymbol{H_y}$ is computed, the model order is irrelevant in dictating the run time (at least when $M \ll N$). This is not the case for the MMEMPM; because the PYTHON implementation used (Code Listing B.4) employs a truncated SVD in order to compute only the first M components of $\boldsymbol{E_Y}$, the elected model order will have an impact on run time. Therefore, FIDs with different model orders were generated: $M \in \{10, 20, 40, 80\}$.

The MMEMPM was repeated 5 times for each FID, and the mean run times for each $N^{(2)}$ and M considered is plotted in Figure 2.4.b1. Results are only presented in cases where the MMEMPM was able to yield a satisfactory estimation result in close agreement with the model used to generate the FID; for certain FIDs with low $N^{(2)}$ and high M, appropriate estimation results could not be obtained as the constituent signals were too poorly resolved. While in the high-N regime the MPM has a cubic time dependence on the number of points, the MMEMPM can be seen to have an approximately quadratic complexity regarding $N^{(2)}$. For all combinations of $N^{(2)}$ and M, the truncated SVD was the most time consuming aspect of the routine, however other steps have notable run times too. The MMEMPM can be broken down into the following 5 steps, with the relevant lines in Code Listing B.4 given:

1. Construction of $\boldsymbol{E_Y}$. This involves building the Hankel matrices $\{\boldsymbol{H}_{\boldsymbol{y},n^{(1)}} : n^{(1)} \in \{0, \cdots, 63\}\}$, assigning them to the correct locations in $\boldsymbol{E_Y}$, and finally converting $\boldsymbol{E_Y}$ to a sparse matrix††† (Lines 44 to 62).
2. Truncated SVD of $\boldsymbol{E_Y}$ to generate $\boldsymbol{U}_M$ (Line 66).
3. Determining $\boldsymbol{z}^{(1)}$ and $\boldsymbol{W}^{(1)}$ by computing the eigenvalue decomposition of $\boldsymbol{U}_{M1}^{+}\boldsymbol{U}_{M2}$ (Lines 68 to 71).
4. Generating the second set of signal poles $\boldsymbol{z}^{(2)}$, by multiplying $\boldsymbol{U}_M$ with the permutation matrix, and extracting the diagonal from matrix $\boldsymbol{G}$, computed using Equation 2.26 (Lines 73 to

†††Truncated SVD is only available for sparse matrices in NUMPY [104]. Some experimenting was done to determine the most efficient means of generating $\boldsymbol{E_Y}$ in sparse form, and subsequently compute its SVD. It was determined that constructing $\boldsymbol{E_Y}$ using a standard NUMPY array before converting it to compressed sparse row (CSR) format [113] was optimal.

$N^{(2)}$	M	Step 1	Step 2	Step 3	Step 4	Step 5
64	10	12.2%	60.6%	2.9%	9.7%	13.8%
64	40	4%	74.6%	1.9%	8.9%	9.6%
512	10	22.1%	67.2%	0.2%	3.1%	7.2%
512	40	7.4%	81.9%	0.1%	1.3%	9.4%
512	80	3.8%	85.8%	0.1%	0.8%	9.4%

Table 2.2: A comparison of the relative times to perform the steps in the MMEMPM, for selected pairings on M and $N^{(2)}$. See the main text for a description of what each step entails.

87). N.B. The FIDs were constructed such that the signal poles were unique on all occasions, so the additional treatment of repeated signal poles was not necessary.

5. Computation of the complex amplitudes using Equation 2.28 (Lines 119 to 132).

A comparison of the relative times to perform these steps for some select pairings of M and $N^{(2)}$ is provided by Table 2.2. It can be seen that as M increases, the relative amount of time spent performing SVD increases, while that to generate $\boldsymbol{E_Y}$ decreases. This reflects the greater number of iterations required by the truncated SVD routine [104] to produce the desired number of components, while the run time to generate $\boldsymbol{E_Y}$ remains fixed.

Interestingly, the plots for a given value of M in Figure 2.4.b1 do not exhibit consistent quadratic behaviour throughout; after a certain value of $N^{(2)}$, a slight reduction in the gradient of the plots is observed (cf the square and circular points). This was found to be caused by the SVD computation, whose run times for the $M = 40$ FIDs are plotted in Figure 2.4.b2. The square and circular points both display quadratic behaviour, though the exact form of the function which describes them are different; both sets of points have been fit to curves of the form $aN^{(2)^2} + b$. After $N^{(2)}$ becomes larger than roughly 200, the SVD run time appears to enter a new regime in which a slower rate of increase is observed. The exact reason for why this is observed was not ascertained.

The peak RAM usage for the $M = 40$ FIDs as a function of $N^{(2)}$ is plotted in Figure 2.4.b3, where a quadratic complexity is observed. The variation in memory consumption barely changes as a function of the model order, since the peak usage is largely dependent on the size of $\boldsymbol{E_Y}$.

It should be noted that the run time and peak RAM consumption will also be (roughly) quadratically dependent on $N^{(1)}$, just as with $N^{(2)}$, e.g. increasing $N^{(1)}$ from 64 to 128 would cause all the run times and peak memory usages in Figures 2.4.b1 and 2.4.b3 to quadruple in value.

2.4.2 Computing the Hessian for NLP

A consideration of the burden on the CPU and RAM in computing the Hessian matrix for 1D and 2D FIDs (Equation 2.44b) is presented in Figure 2.5. It should be noted that for a typical

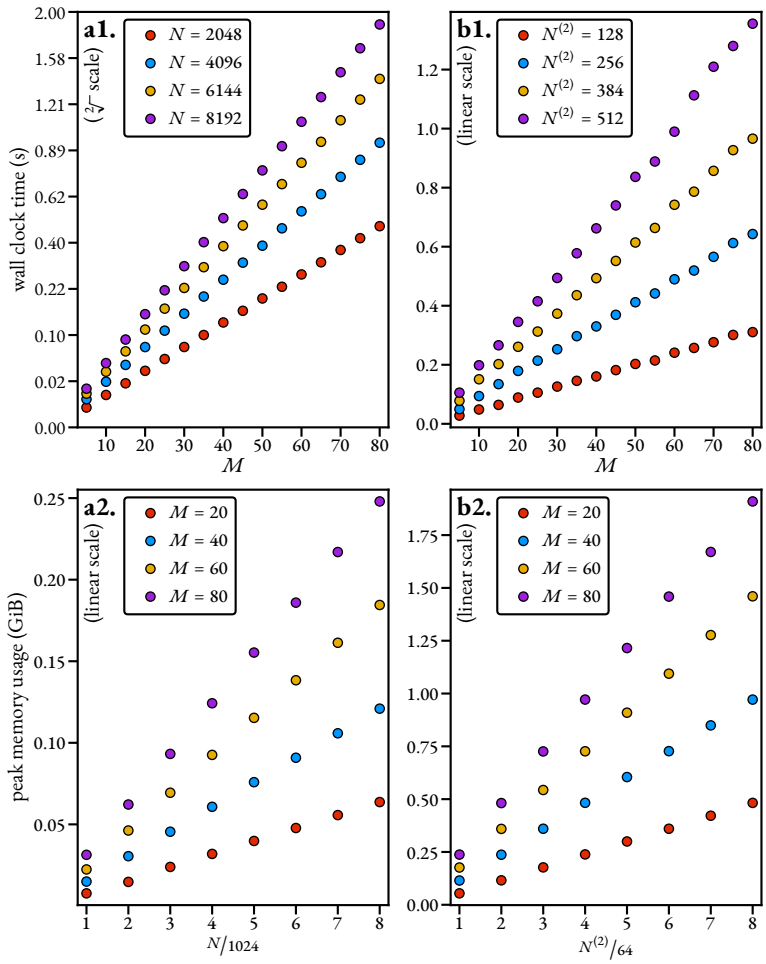


Figure 2.5: The run time and peak memory consumption in computing the Hessian matrix for FIDs with differing numbers of datapoints and signals. The FIDs that were used to acquire these results are described in the main text. **a1.** and **b1.** The amount of time required to compute the Hessian for 1D and 2D FIDs, respectively. **a2.** and **b2.** Equivalent plots showing the peak memory consumption in computing the Hessian.

NLP routine, the Hessian will need to be computed many times (on the order of magnitude of 10^2 is common) so while the times presented in the figure may seem small, a rather long total time is possible for the NLP routine if it takes seconds to compute the Hessian once. For the 1D case, simulated FIDs were generated with both a variable number of contributing signals: $M \in \{5k : k \in \{1, 2, \cdots, 16\}\}$ and datapoints: $N \in \{1024k : k \in \{1, 2, \cdots, 8\}\}$. For each FID, the Hessian was computed using a slight variant of the `obj_grad_hess_1d` function, given by Lines 50 to 135 in Code Listing B.6[‡‡‡]. There are 4 main steps in computing the Hessian matrix:

1. Generation of all the first partial derivatives (Equation A.7), which are stored in a $4M \times N$ array.
2. Generation of all the non-trivially zero second partial derivatives (Equation A.8), which are stored in a $10M \times N$ array.
3. Computation of the non-zero elements of term ② in Equation 2.44b.
4. Computation term ① in Equation 2.44b.

These steps produce an exact Hessian. Neglecting steps 2 and 3 leads to the formation of the approximated Hessian as described in Section 2.3.2. Figure 2.5.a1 shows that the time to compute the Hessian matrix scales quadratically with M, and linearly with N. The cause of this is the fact that the most time-consuming aspect of the routine is step 4, which involves multiplying the $4M \times N$ matrix of first derivatives with its $N \times 4M$ conjugate transpose, with a complexity $\mathcal{O}(M^2N)$. The relative amount of the total run time which is spent performing step 4 increases steadily as the model order increase, as seen in Table 2.3. As such, for 1D FIDs, it is common, especially when the data comprises a large number of signals, for the discrepancy in the amount of time to compute the true Hessian relative to its approximation to be small. In these cases, it may be valuable to compute the exact Hessian if it leads to a reduced number of required iterations for convergence.

To assess the effect of the number of signals and datapoints on Hessian computation for 2D FIDs, a series of signals were generated with a fixed number of datapoints in the first dimension: $N^{(1)} = 64$. FIDs were then constructed with the same values of M as the 1D case, and $N^{(2)} \in \{64k : k \in \{1, 2, \cdots, 8\}\}$. In contrast to the 1D case, it is seen that the run time in computing the Hessian now scales approximately linearly, rather than quadratically. Much more of the run time is taken up by the first three steps in comparison; computing the $6MN^{(1)}N^{(2)}$ first derivatives, $21MN^{(1)}N^{(2)}$ second derivatives, and forming term ② all scale linearly with M, $N^{(1)}$, and $N^{(2)}$. The relative times spent performing steps 2 and 3 compared with 1 and 4 imply that for the 2D case (and for signals more more than 2 dimensions), a large saving in computational time is typically achieved when the Hessian approximation is utilised over the exact form.

[‡‡‡]The variant of this function neglected the computation of the objective and gradient, and returned only the Hessian.

$N/N^{(2)}$	M	Step 1	Step 2	Step 3	Step 4
1D Hessian					
2048	20	3.7%	9.6%	6%	76.4%
8192	20	3.5%	10.7%	6.3%	75%
2048	80	1%	3%	2%	92.6%
8192	80	1.2%	3.3%	1.9%	92.4%
2D Hessian					
64	20	8.2%	32.5%	18.5%	38.5%
256	20	12.3%	50.2%	21.3%	14.3%
64	80	10.8%	41%	24.3%	21.5%
256	80	14.9%	45.4%	24.1%	13.8%

Table 2.3: A comparison of the relative times to perform the steps for computing the Hessian matrix for NLP, for selected pairings of M and N (1D)/$N^{(2)}$ (2D).

Both the time- and space-complexity of computing the Hessian is found to be roughly linear with both M and the number of datapoints in each dimension for 1D and 2D FIDs (see Figures 2.5.a2 and 2.5.b2). Most of the RAM usage comes from storing arrays of the first and second partial derivatives. In general, the typical space requirements for both 1D and 2D FIDs should be tolerable on modern computers, which tend to possess at least 8 GiB of RAM.

2.5 Frequency Filtration

A typical NMR user is unlikely to make use of the estimation routine outlined if it takes a considerable amount of time to run — on the order of several minutes or even hours — especially if the information they are interested in could be obtained by other means, such as running additional NMR experiments. The previous section highlights that FIDs which comprise smaller numbers of datapoints and signals can be estimated more rapidly. A means of generating frequency-filtered "sub-FIDs" is presented in this section, which is able to reduce both of these features, without compromising the ability of the estimation routine to parameterise the original FID. In essence, this sub-FID approach transforms the problem of FID estimation from a single large-scale estimation problem to a plurality 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. Through data filtration, all focus can be devoted to those frequency regions which can realistically be studied and which are of interest in the first place.

2.5.1 The Virtual Echo

The key steps of the filtering procedure are (a) transforming the time-domain data to the frequency domain, (b) applying a band-pass filter to the spectral region of interest, and (c) returning the spectrum back to the time-domain for estimation. For a filtered sub-FID to still be faithfully modelled by a summation of exponentially damped complex sinusoids, it is necessary that the spectral peaks of interest lie (effectively) entirely within the filter region[§§§]. Due to their narrower linewidths relative to dispersion Lorentzians, a phased 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 [114, 115, 116]. The VE is a signal with double the size of the original FID, with the key characteristic that its FT has an imaginary component of zeros. The VE concept can be applied to data of any number of dimensions. However, discussion in this section will be limited to 1D VEs; an account of the 2D VE is provided in Appendix A.3.

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

$$y_n = \xi_n(c_n + \mathrm{i}s_n) + w_n, \tag{2.52a}$$

$$\xi_n = \sum_m a_m \exp(-\eta_m n\Delta_t), \tag{2.52b}$$

$$c_n/s_n = \sum_m \cos/\sin(2\pi f_m n\Delta_t). \tag{2.52c}$$

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

$$\psi_{\pm,n} = \xi_n(c_n \pm \mathrm{i}s_n) + w_n \equiv \Re(y_n) \pm \mathrm{i}\Im(y_n) \tag{2.53}$$

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

$$\boldsymbol{t}_1 = \begin{bmatrix} \boldsymbol{\psi}_+ \\ \mathbf{0} \in \mathbb{C}^N \end{bmatrix}. \tag{2.54}$$

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

$$\boldsymbol{t}_2 = \begin{bmatrix} \mathbf{0} \in \mathbb{C}^N \\ \boldsymbol{\psi}_-^{\leftrightarrow} \end{bmatrix}^{\circlearrowright}. \tag{2.55}$$

[§§§] A Lorentzian function tends to but never explicitly reaches zero as the distance from its maximum tends to ∞ (Equation 1.30). However, as long as a sufficiently wide filter region is defined, the extent to which the "tails" of the Lorentzian fall outside the filter window can be assumed to be negligible, especially when in the presence of noise.

The VE $\boldsymbol{y}_{\mathrm{VE}}$ is then given by $\boldsymbol{t}_1 + \boldsymbol{t}_2$, with the first element divided by 2. This entire process is equivalent to

$$\boldsymbol{y}_{\mathrm{VE}} = \begin{bmatrix} \Re(y_0) & y_1 & \cdots & y_{N-1} & 0 & y_{N-1}^* & \cdots & y_1^* \end{bmatrix}^{\mathrm{T}}. \tag{2.56}$$

As alluded to already, the FT of $\boldsymbol{y}_{\mathrm{VE}}$ produces a spectrum $\boldsymbol{s}_{\mathrm{VE}}$ such that $\Im(\boldsymbol{s}_{\mathrm{VE}}) = \boldsymbol{0}$, with $\Re(\boldsymbol{s}_{\mathrm{VE}})$ featuring absorption Lorentzian peaks.

2.5.2 The Filtering Process

Remark 4. *Every result presented in this thesis which involved filtering made use of the process outlined below. However in retrospect, a more straightforward method of applying a rectangular filter, and slicing the filtered spectrum exactly at the filter boundaries would likely yield sub-FIDs with the same information content, without the requirement to incorporate synthetic noise into the filtered dataset.*

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

$$g_n = \exp\left(-2^{p+1}\left(\frac{n-c}{b}\right)^p\right). \tag{2.57}$$

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

$$c = \tfrac{1}{2}\left(l_{\mathrm{I}} + r_{\mathrm{I}}\right), \tag{2.58a}$$

$$b = r_{\mathrm{I}} - l_{\mathrm{I}}, \tag{2.58b}$$

where $l_{\mathrm{I}}, r_{\mathrm{I}} \in \{0, \cdots, 2N-1\}, r_{\mathrm{I}} > l_{\mathrm{I}}$ denote the left and right boundaries of the region of interest defined by the user, expressed as vector indices. The vector indices f_{idx} can be obtained from the corresponding spectral frequencies f_{Hz} via

$$\begin{gathered} f_{\mathrm{idx}} = \left\lfloor \frac{(2N-1)\left(f_{\mathrm{sw}} + 2\left(f_{\mathrm{off}} - f_{\mathrm{Hz}}\right)\right)}{2f_{\mathrm{sw}}} \right\rceil \\ \forall f_{\mathrm{Hz}} \in \left[f_{\mathrm{off}} - \tfrac{1}{2}f_{\mathrm{sw}}, f_{\mathrm{off}} + \tfrac{1}{2}f_{\mathrm{sw}}\right]. \end{gathered} \tag{2.59}$$

Alternatively, conversion from ppm to array indices can be achieved by replacing f_{Hz} in Equation 2.59 with $f_{\mathrm{ppm}} f_{\mathrm{sfo}}$, where f_{sfo} is the transmitter frequency (MHz) and f_{ppm} is the frequency expressed as a chemical shift.

Application of the super-Gaussian filter to $\boldsymbol{s}_{\mathrm{VE}}$ would lead to large sections of the filtered spectrum being 0. This has an undesired impact on model order prediction using 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 results from this. In order to obtain better predictions from model order selection, an array of synthetic noise is added to the filtered spectrum. To achieve this, a region in $\boldsymbol{s}_{\mathrm{VE}}$ is specified by the user (defined by the indices l_{N} and r_{N}) which contains no discernible signal peaks; this is referred to as the *noise region*. The variance of the noise 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, $\boldsymbol{w}_{\sigma^2} \in \mathbb{R}^{2N}$. The filtered spectrum is then given by

$$\widetilde{\boldsymbol{s}}_{\mathrm{VE}} = \boldsymbol{s}_{\mathrm{VE}} \odot \boldsymbol{g} + \boldsymbol{w}_{\sigma^2} \odot (\boldsymbol{1} - \boldsymbol{g}) \,. \tag{2.60}$$

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, $\widetilde{\boldsymbol{s}}_{\mathrm{VE}}$ is returned to the time-domain by inverse Fourier transform (IFT), defined for a generic frequency-domain vector $\boldsymbol{s} \in \mathbb{C}^N$ as

$$y_n = \frac{1}{N} \sum_{k=0}^{N-1} s_k \exp\left(\frac{2\pi \mathrm{i} k n}{N}\right) \quad \forall n \in \{0, \cdots, N-1\}. \tag{2.61}$$

The IFT of a real-valued spectrum generates a conjugate-symmetric signal (another VE). This is sliced so as to retain the first half, the final filtered sub-FID $\widetilde{\boldsymbol{y}} \in \mathbb{C}^N$. A depiction of the key elements involved in the filtering process is provided by Figure 2.6, while a pseudo-code description is provided by Algorithm 2.3.

Thus far, the method described is able to reduce the number of signals, though the filtered sub-FID still comprises the same number of datapoints. However, it is clear that there is a large number of points outside the region of interest in $\widetilde{\boldsymbol{s}}_{\mathrm{VE}}$ that do not possess any meaningful information. Discarding such points will then lead to a sub-FID with the same information, but in a more compressed FID. To achieve this, a slicing ratio is defined, $\chi > 1$, which dictates the left and right boundaries of a region outside of which points will be discarded:

$$l_{\mathrm{slice}} = \max\left(c - \left\lfloor \frac{b\chi}{2} \right\rfloor, 0\right), \tag{2.62a}$$

$$r_{\mathrm{slice}} = \min\left(c + \left\lceil \frac{b\chi}{2} \right\rceil, 2N - 1\right). \tag{2.62b}$$

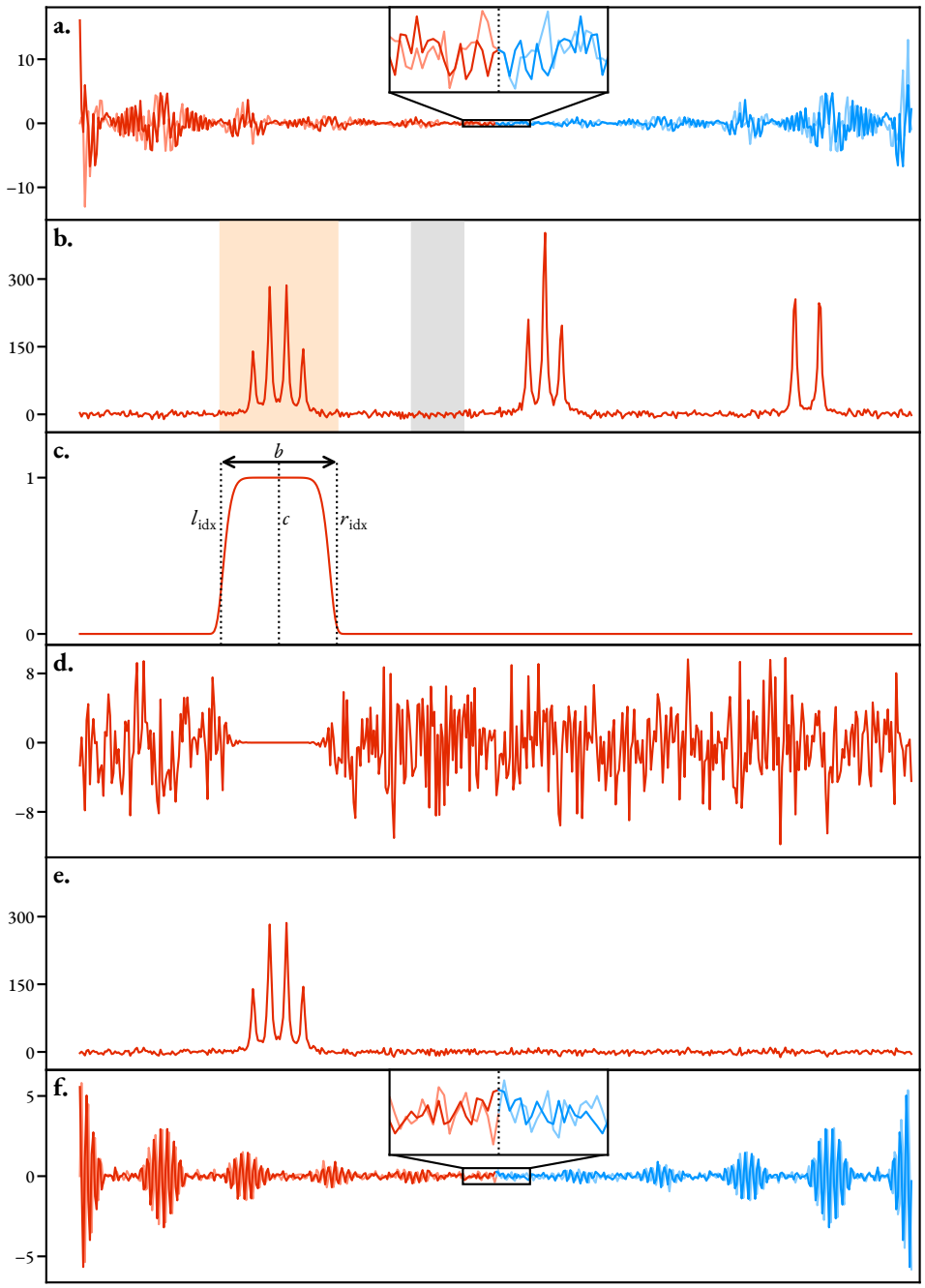


Figure 2.6: An illustration of the filtering procedure applied to a 1D FID. **a.** A VE $\boldsymbol{y}_{\text{VE}}$, with the first and last N points coloured red and blue, respectively. The middle of the VE is magnified to highlight its conjugate symmetry. **b.** The FT of the VE, $\boldsymbol{s}_{\text{VE}}$. The region of interest (orange) and noise region (grey) are denoted. **c.** A super-Gaussian function used as a band-pass filter, $\boldsymbol{g}$. **d.** Synthetic noise vector to be added to the filtered spectrum, $\boldsymbol{w}_{\sigma^2}(\mathbf{1} - \boldsymbol{g})$. **e.** The filtered spectrum $\widetilde{\boldsymbol{s}}_{\text{VE}}$, formed by applying the super-Gaussian filter, and adding the noise vector. **f.** The IFT of the filtered spectrum, $\widetilde{\boldsymbol{y}}_{\text{VE}}$, from which the final filtered signal $\widetilde{\boldsymbol{y}}$ is obtained by extracting the first N (red) points.

The filtered spectrum is then sliced accordingly:

$$\mathbb{R}^{r_{\text{slice}}-l_{\text{slice}}} \ni \widetilde{\boldsymbol{s}}_{\text{VE,slice}} = \widetilde{\boldsymbol{s}}_{\text{VE}}[l_{\text{slice}} : r_{\text{slice}} + 1]. \tag{2.63}$$

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 sub-FID 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:

$$\widetilde{\boldsymbol{y}} = \frac{r_{\text{slice}} - l_{\text{slice}}}{2N} \text{IFT}(\widetilde{\boldsymbol{s}}_{\text{VE,slice}})[0 : N_{\text{slice}}], \tag{2.64a}$$

$$N_{\text{slice}} = \left\lfloor \frac{r_{\text{slice}} - l_{\text{slice}}}{2} \right\rfloor \tag{2.64b}$$

The associated spectral 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

$$f_{\text{sw,slice}} = \frac{r_{\text{slice}} - l_{\text{slice}}}{2N - 1} f_{\text{sw}} \tag{2.65a}$$

$$f_{\text{off,slice}} = f_{\text{off}} + \frac{f_{\text{sw}}}{2}\left(1 - \frac{l_{\text{slice}} + r_{\text{slice}}}{2N - 1}\right) \tag{2.65b}$$

2.6 Summary

The MPM is well established as an effective method for the parametric estimation of signals in a number of disciplines. 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 insurmountable for complete NMR datasets, which commonly comprise thousands of points, and at least hundreds of contributing signals. Both the run time and peak memory consumption of the 1D and 2D methods increase for FIDs with more datapoints and more signals. For this reason, a method to break the estimation problem into a series of smaller, computationally tractable problems, through the construction of frequency-filtered sub-FIDs, has been introduced.

Algorithm 2.3 The filtering procedure for 1D data. $l/r_{\mathrm{I/N}}$ denotes the left (l)/right (r) bound of the region of interest (I)/noise region (N), as a vector index. All of these values should be members of the set $\{0, \cdots, 2N-1\}$. These would typically be provided in units of Hz or ppm by a user; conversion to indices can be carried out using Equation 2.59.

procedure FILTER1D($\boldsymbol{y} \in \mathbb{C}^N, l_{\mathrm{I}}, r_{\mathrm{I}}, l_{\mathrm{N}}, r_{\mathrm{N}}, \chi \in \mathbb{R}_{>1}$)
  $\boldsymbol{y}_{\mathrm{VE}} \leftarrow$ VIRTUALECHO1D $(\boldsymbol{y})$;
  $\boldsymbol{s}_{\mathrm{VE}} \leftarrow \mathrm{FT}\left(\boldsymbol{y}_{\mathrm{VE}}\right)$;
  $c \leftarrow {}^{(l_{\mathrm{I}}+r_{\mathrm{I}})}/_{2}$;
  $b \leftarrow r_{\mathrm{I}} - l_{\mathrm{I}}$;
  $\boldsymbol{g} \leftarrow$ SUPERGAUSSIAN1D $(2N, c, b)$;
  $\boldsymbol{s}_{\mathrm{N}} \leftarrow \boldsymbol{s}_{\mathrm{VE}}\left[l_{\mathrm{N}} : r_{\mathrm{N}} + 1\right]$;
  $\sigma^2 \leftarrow \mathrm{Var}\left(\boldsymbol{s}_{\mathrm{N}}\right)$;
  $\boldsymbol{w}_{\sigma^2} \leftarrow \mathbf{0} \in \mathbb{R}^{2N}$;
  for $n = 0, \cdots, 2N-1$ **do**
    $w_{\sigma^2,n} \leftarrow$ RANDOMSAMPLE $\left(\mathcal{N}\left(0, \sigma^2\right)\right)$;
  end for
  $\widetilde{\boldsymbol{s}}_{\mathrm{VE}} \leftarrow \boldsymbol{s}_{\mathrm{VE}} \odot \boldsymbol{g} + \boldsymbol{w}_{\sigma^2} \odot (\mathbf{1} - \boldsymbol{g})$;
  $l_{\mathrm{slice}} \leftarrow \max\left(c - \left\lfloor \frac{b\chi}{2} \right\rfloor, 0\right)$;
  $r_{\mathrm{slice}} \leftarrow \min\left(c + \left\lceil \frac{b\chi}{2} \right\rceil, 2N-1\right)$;
  $\widetilde{\boldsymbol{s}}_{\mathrm{VE,slice}} \leftarrow \widetilde{\boldsymbol{s}}[l_{\mathrm{slice}} : r_{\mathrm{slice}} + 1]$;
  $\widetilde{\boldsymbol{y}}_{\mathrm{VE}} \leftarrow \frac{r_{\mathrm{slice}} - l_{\mathrm{slice}}}{2N}\,\mathrm{IFT}\left(\widetilde{\boldsymbol{s}}_{\mathrm{VE,slice}}\right)$;
  $\widetilde{\boldsymbol{y}} \leftarrow \widetilde{\boldsymbol{y}}_{\mathrm{VE}}\left[: \left\lfloor \frac{r_{\mathrm{slice}} - l_{\mathrm{slice}}}{2} \right\rfloor\right]$;
  return $\widetilde{\boldsymbol{y}}$;
end procedure

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

procedure SUPERGAUSSIAN1D($N \in \mathbb{N}, c \in \mathbb{N}_0, b \in \mathbb{N}_0$)
  $\boldsymbol{g} \leftarrow \mathbf{0} \in \mathbb{R}^N$;
  for $n = 0, \cdots, N-1$ **do**
    $g_n \leftarrow \exp\left(-2^{41}\left(\frac{n - c_{\mathrm{idx}}}{b_{\mathrm{idx}}}\right)^{40}\right)$; ▷ p in Equation 2.57 has been set to 40.
  end for
  return $\boldsymbol{g}$
end procedure

Algorithm 2.4 provides an overview of the principal steps involved in the 1D estimation procedure.

Having established an estimation routine, the next chapter focusses on its performance in analysing 1D NMR datasets. Furthermore, two specific applications which arise from possession of the information afforded by parametric estimation are described.

Algorithm 2.4 The estimation procedure outlined in this work, for the consideration of 1D FIDs.

1: **procedure** ESTIMATE1D($\boldsymbol{y} \in \mathbb{C}^{N^{(1)}}, l_{\mathrm{I}}, r_{\mathrm{I}}, l_{\mathrm{N}}, r_{\mathrm{N}}, \chi \in \mathbb{R}_{>1}, M \in \mathbb{N}_0$)
2: $\quad \widetilde{\boldsymbol{y}} \leftarrow$ FILTER1D($\boldsymbol{y}, l_{\mathrm{I}}, r_{\mathrm{I}}, l_{\mathrm{N}}, r_{\mathrm{N}}, \chi \in \mathbb{R}_{>1},$); ▷ Algorithm 2.3
3: $\quad \boldsymbol{\theta}^{(0)} \leftarrow \mathrm{MPM}\left(\widetilde{\boldsymbol{y}}, M\right)$; ▷ Algorithm 2.1
4: $\quad \boldsymbol{\theta}^{(*)}, \boldsymbol{\epsilon}^{(*)} \leftarrow \mathrm{NLP}\left(\widetilde{\boldsymbol{y}}, \boldsymbol{\theta}^{(0)}\right)$; ▷ Algorithm 2.2
5: $\quad$ **return** $\boldsymbol{\theta}^{(*)}, \boldsymbol{\epsilon}^{(*)}$;
6: **end procedure**

Chapter 3

Results and Applications Involving 1D Estimation

This chapter showcases results generated using the 1D estimation technique outlined in the previous chapter. Furthermore, two means by which the 1D estimation procedure can be harnessed as part of specific applications are presented:

- Section 3.1 provides some examples of how the estimation routine performs in analysing 1D NMR datasets acquired using a standard 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 those from inversion recovery, Carr-Purcell-Meiboom-Gill (CPMG), 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 and baseline distortions.

All the results presented were generated using the NMR-EsPy package, described in Chapter 5. Details about the datasets generated are provided in Appendix C.

3.1 Conventional 1D Datasets

As mentioned in Section 2.3.4, one of the disadvantages of SVD-based methods like the MPM is their propensity to generate parameter estimates featuring oscillators with spurious phase behaviour. Such behaviour is most prevalent in FIDs which (a) feature signals with very similar frequencies and (b) that have a low SNR. To assess the effectiveness of including the phase variance-regularised NLP routine, comparisons are now made with results generated with the MPM in

isolation.

For the experimental datasets considered (Sections 3.1.2 and 3.1.3), the data were pre-processed using BRUKER's TOPSPIN software, using the series of commands `ft; pk; abs`; these perform FT, automatic phase correction, and baseline correction, respectively. The data was then converted back to the time-domain using IFT prior to filtering and estimation.

3.1.1 "Twenty Signals"

A series of five simulated 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 spectral 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 $\forall m \in \{1, \cdots, 20\}$: $a_m \sim \mathcal{U}(1, 5)$, $f_m \sim \mathcal{U}(-55\,\text{Hz}, 55\,\text{Hz})$, $\eta_m \sim \mathcal{U}(2\,\text{s}^{-1}, 8\,\text{s}^{-1})$. The frequencies were subjected to an additional constraint; no two signals were permitted to have frequencies that differed by less than $4f_{\text{sw}}/N \approx 0.49\,\text{Hz}$. Each FID $\boldsymbol{x}$ was then corrupted with noise, with a target SNR of 25 dB, such that the desired noise variance for each FID was given by (*cf.* Equations 1.26 and 1.27):

$$\sigma^2 = \frac{1}{20^{2.5} \times 1024} \sum_{n=0}^{1023} |x_n|^2. \tag{3.1}$$

The spectra of the simulated FIDs are presented in Figure 3.1.a, with the set of oscillator peaks which contribute to the spectrum in Figure 3.1.b. With the criteria used to constructed the datasets, it can be seen they feature signals which often suffer from severe overlap, with the high noise variance compounding the opportunity to clearly identify all contributing signals.

For each FID, the MPM was performed, assuming a model order of 30, constituting a considerable over-fit of oscillators. The MDL tended to produce under-estimates of M when applied to these FIDs, so the hard-coded value was used instead to ensure sufficient oscillators were given to the model. The under-estimates were likely due to the low SNR of the signals — recall the red signal in Figure 2.2 — in conjunction with severe signal overlap. When an excessive model order is provided to the MPM, it is typical that oscillators corresponding to the noise subspace of the data matrix $\boldsymbol{H_y}$ are characterised by small amplitudes and/or very small damping factors. For this reason, prior to subjecting the MPM result to NLP, oscillators which satisfied $a_m < 0.1$ and/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 Figure 3.1.c, along with the residual between the data and the estimated model (i.e. the sum of all oscillator peaks).

The MPM consistently generated models which agreed well with the data in a least squares sense,

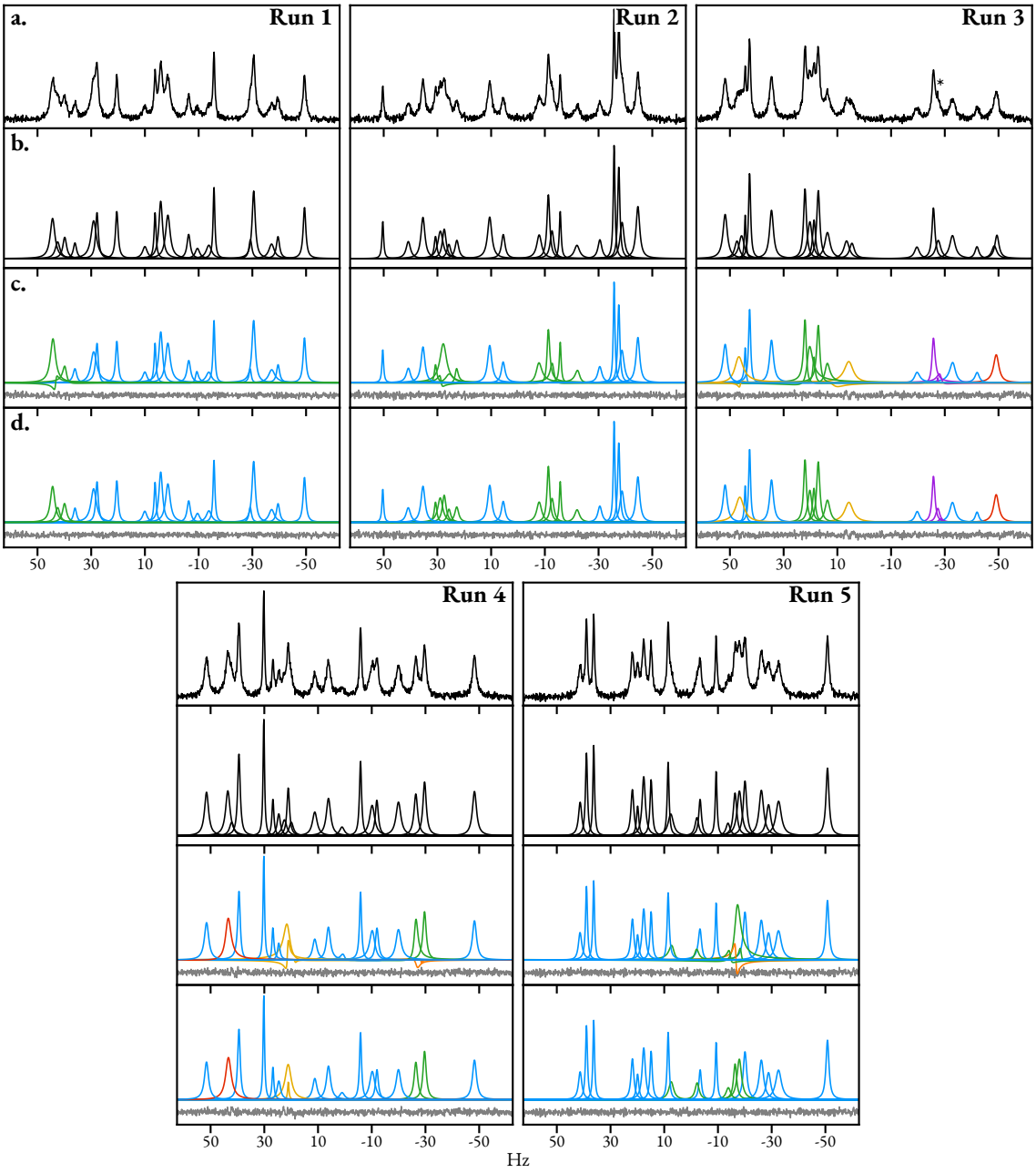


Figure 3.1: The result of estimating a series of 5 simulated FIDs comprising 20 signals. See the main text for details on how the FIDs were constructed. **a.** Spectra of the FIDs generated. **b.** Spectral lines corresponding to the set of signals used to generate each FID. **c.** Plots of peaks for each oscillator generated using the MPM. **d.** An equivalent plot for the result after applying phase variance-regularised NLP, using the MPM result as an initial guess. Also included in c and d are the residual between the data and the estimated model (grey line). Blue oscillators are those produced by the MPM which closely agree with a true signal. Green oscillators are affected notably by the NLP routine, and end up in good agreement with a true signal. Orange oscillators were excessive oscillators generated by the MPM, which were removed during the NLP routine, leading to a parsimonious fit of the remaining (green) oscillators in the frequency neighbourhood. Yellow oscillators denote cases where the number of oscillators produced by the MPM matched the number of true signals in a given frequency neighbourhood. However one of the oscillators was removed during the NLP routine, leading to an under-fit of the neighbourhood. Red oscillators denote an under-fit of a frequency neighbourhood by the MPM. Purple oscillators denote an over-fit of a frequency neighbourhood by the MPM, with none of the oscillators being removed during NLP (*cf.* orange oscillators which are removed during NLP).

as as evidenced by the residuals in Figure 3.1.c. 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 signal parameters used to construct the FIDs. Most notably, individual oscillator phases regularly stray far from 0°, and their associated amplitudes are often considerably different too. In Figure 3.1.c, blue oscillators are those which are in very close agreement with a particular true signal in the data. Oscillators with other colours are not in agreement with a true signal, with the different colourings described shortly. The desired outcome of the NLP routine is for it to adjust the parameters associated with non-blue oscillators in Figure 3.1.c such that they agree with true signals, while having little to no effect on those of the blue oscillators. The results after application of NLP are provided in Figure 3.1.d.

In discussing the outcome of the routine, it will be helpful to employ the concept of a *frequency neighbourhood*, a loose term which describes a small, continuous subset of frequencies within the spectral window. As the NLP routine involves taking small steps in parameter space over a number of iterations, 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 signals within a given frequency neighbourhood of the data, sufficient model oscillators need to present within the neighbourhood in the initial guess.

Cases where the MPM generated enough oscillators for a given frequency neighbourhood, albeit with parameters which noticeably deviate from the true signals are coloured either green or yellow. Green oscillators are those which the NLP routine was able to adjust so as to to achieve agreement with true signals; they indicate improvements to the estimation result in comparison with the MPM in isolation. Conversely, yellow oscillators denote cases where, though sufficient oscillators existed in the frequency neighbourhood in the initial guess, the NLP routine evolved such that at least one of the oscillators was driven by the phase variance constraint to acquire a negative amplitude, leading to it being removed from the parameter set. This typically occurred with oscillators that had an initial phase which was either $> \pi/2$ rad, or $< -\pi/2$ rad. Therefore, yellow oscillators indicate cases where the final result under-fit the dataset.

There are a couple of instances, denoted by red oscillators, where the MPM assigned too few oscillators to a particular frequency neighbourhood, and as such the NLP routine would not have been able to yield any major improvement. This occurred in what can safely be described as fiendishly difficult cases, where severe signal overlap and low SNR make it very difficult to associate certain spectral regions with more than one signal by eye.

The final two oscillator groupings, coloured purple and orange respectively, indicate scenarios where the MPM generated more oscillators than true signals in a given frequency neighbourhood (i.e. the data was over-fit in these regions). Orange oscillators we purged by the NLP routine

as they acquired negative amplitudes. This enabled parsimonious fits of the relevant frequency neighbourhoods by the oscillators which remained, i.e. the green oscillators in close proximity to the orange ones in Figure 3.1.c. Finally, the purple oscillators denote the one occasion (Run 3) where an over-fit occurred, and the model order was not successfully reduced by the NLP routine. One of the purple oscillators in the MPM appears to agree closely with a significant "blip" in the noise, denoted by an asterisk in Figure 3.1.a. The over-fit has therefore occurred because this noise component was incorporated into the final result; it was neither purged based on the amplitude and damping factor criteria outlined above, nor by acquiring a negative amplitude during NLP.

3.1.2 Andrographolide

Figure 3.2 illustrates the outcome of applying the estimation routine to a ^{1}H dataset of andrographolide in DMSO-d_6, acquired with a 600 MHz spectrometer. Various spectral regions were chosen for study, and for each one, a frequency-filtered sub-FID was produced using the approach in Section 2.5. The MPM was used to generate an initial guess of parameters, using the MDL to predict the model order in each case. The result of applying the MPM are presented in Figure 3.2.b. The initial guess was then subjected to NLP, giving rise to the result Figure 3.2.c.

The NLP routine was effective at resolving the spurious phase behaviour often generated by the MPM. The estimation method's ability to parametrise signals with high dynamic range and high variation of damping factors is also evidenced; a broad intense singlet from water, and a low-intensity quartet from residual ethanol in the sample, present in the same sub-FID, could both be estimated admirably for example. For some of the sub-FIDs considered, the MPM output featured oscillators, commonly with very high damping factor and/or phases far from 0°, which were removed during the NLP routine (see the red peaks in panel b). The removal of these, along with the enforcement of consistent oscillator phases, leads to parameter estimates which describe the apparent multiplet structures associated with each spin well. 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 datasets is the presence of signals that have similar frequencies* due to the influence 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 multiplet structures. Furthermore, fused systems often exhibit appreciable long-range couplings (between spins separated

*In this context, two frequencies i, j are deemed to be similar if their difference is approximately the same as the spectral resolution, i.e. $|f_i - f_j| \approx f_{sw}/N$. In theory, two signals whose difference is exactly the spectral resolution should be discernible. However in practice it is usually necessary for $|f_i - f_j|$ to be notably larger than the spectral resolution, as increased uncertainty is imposed by signal damping caused by T_2 relaxation.

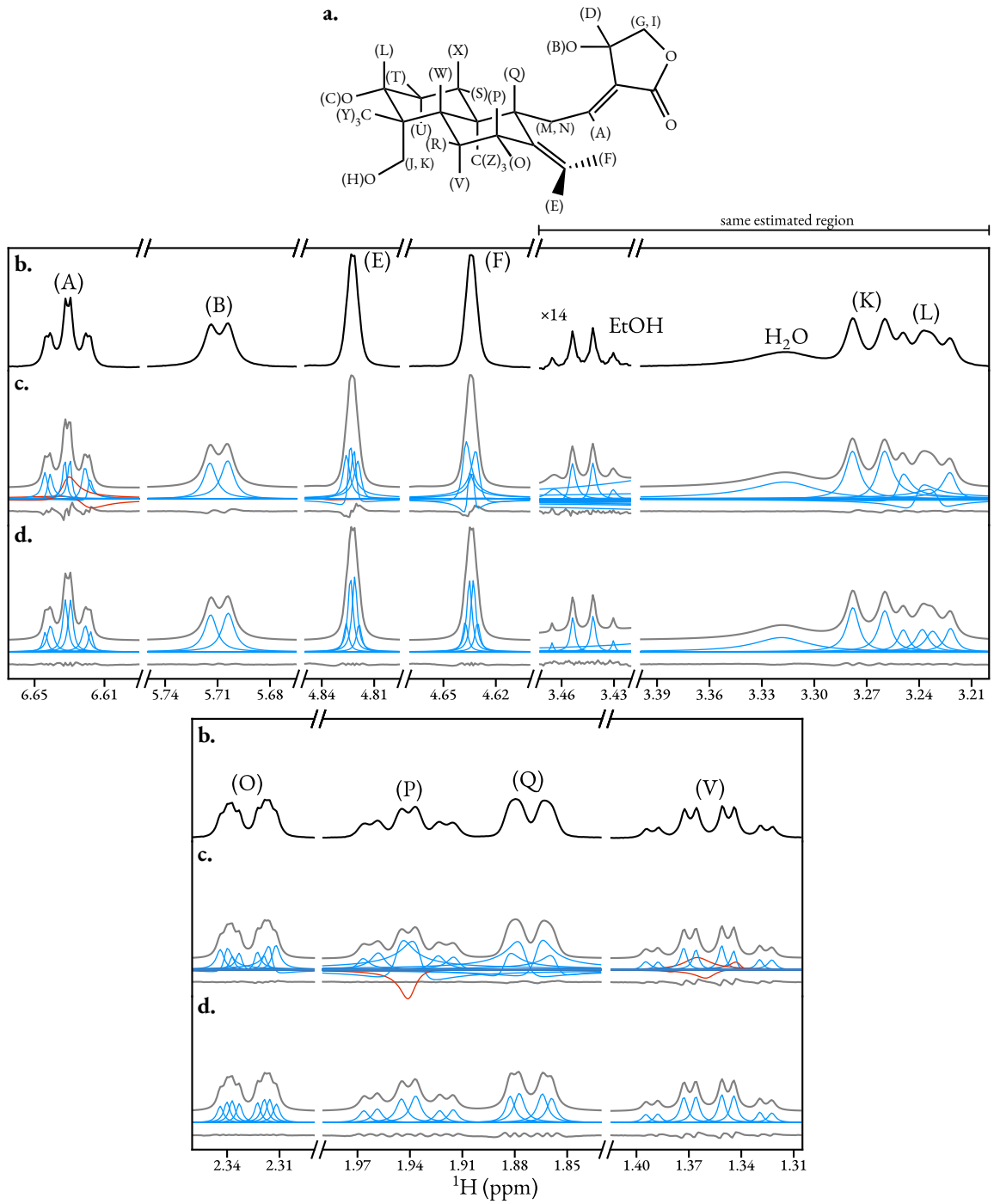


Figure 3.2: The result of applying the estimation routine to selected regions of a pulse-acquire dataset of andrographolide in DMSO-d_6. **a.** The structure of andrographolide. **b.** Spectral regions considered. **c.** The result of applying the MPM to the regions, with the model order predicted using the MDL. Blue and red lines denote individual oscillator peaks, while the grey line above is the sum of all oscillators. The grey line below is the residual between the data and the model. **d.** The result after convergence of the NLP routine, again with the model above and residual below. Red peaks in b correspond to oscillators which acquire negative amplitudes and are removed during the NLP routine. One of the estimated regions has been split in two in the figure to save space, with one half, featuring a signal from ethanol, being magnified.

Spin	Coupling partners	Apparent Multiplet structure
	Andrographolide	
(A)	(D)^{long}, (M)^{vic}, (N)^{vic}	ddd (*dt*)
(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 (*dt*)
(Q)	(M)^{vic}, (N)^{vic} ...	dd...
(V)	(O)^{60}, (P)^{180}, (R)^{g}, (W)^{180}	dddd (*dq*)
	Cyclosporin A	
(A)	diastereotopic pair on $^{\beta}$C	dd
(B)	—"—	dd
(C)	—"— & amide proton	ddd (*dt*)
(D)	proton on $^{\beta}$C & amide proton	dd
(E)	methyl protons on $^{\beta}$C & amide proton	dq
(F)	—"—	dq (*quintet*)

Table 3.1: The major coupling partners associated with spins in andrographolide and cyclosporin, considered in Figures 3.2 and 3.3 respectively, along with the 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, with one bonded to an oxygen, leading to exchange decoupling [39: Section 2.6.1.5], ^{gem} geminal coupling between two spins whose dihedral angle is not fixed, ^{long} long-range coupling, ^{vic} vicinal coupling, ^{60} geminal coupling, with a fixed dihedral angle of 60° between the spins, ^{180} geminal coupling, with a fixed dihedral angle of 180° between the spins. All cyclosporin couplings for the spins considered are geminal couplings. In cases where the observed multiplet structure is different to the true structure, the observed structure is in brackets. Ellipses denote cases where more (long-range) coupling partners are likely, based on the estimation result generated/the appearance of the spectrum, though these have not been explicitly assigned.

by four or more bonds) alongside ubiquitous two-bond (*geminal*) and three-bond (*vicinal*) couplings. Long range couplings can be particularly challenging to resolve, as they are often of a comparable magnitude to the spectral resolution (see Footnote *), making individual signals barely perceptible.

Take the multiplet structure from spin (Q) as an example of a particularly challenging estimation problem. (Q) has separate vicinal couplings to the diastereotopic protons (M) and (N); these are the couplings to (Q) of greatest magnitude. 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. However, a comparison of the data and the model indicates that there is a clear discrepancy between the two, evidenced by systematic deviations in the residual; this feature hints at an under-fit of the data. 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 frequency neighbourhood in which all contributing signals are too poorly resolved to realistically glean any further meaningful information, at least at the field strength used.

As a second illustration, the multiplet structure corresponding to spin (V) is also under-fit, this time because the presence of a number of couplings of similar magnitude leads to resonances coalescing at roughly the same frequency. A multiplet structure featuring 16 resonances forming in "dddd" structure is expected. However, 3 of the couplings are of similar magnitudes, such that many of the contributing signals coalesce to form what is apparently a quartet of doublets (dq). The estimation routine was able to resolve this dq structure, however the large deviations in the residual again imply that under-fitting has occurred, and each of the 4 "central" oscillators in the parameter set is in fact being used to fit three signals present in the FID. Again, at the field strength used to acquire the FID, it is unlikely that an accurate resolution of all 16 signals by estimation is feasible.

One potential way to overcome the under-fit of crowded frequency neighbourhoods is to retroactively adjust the parameter estimate, by "splitting" oscillators. For the spin (V) example, one could take the 8-oscillator estimate generated, and from this create a 16-oscillator estimate in which each of the original 4 central oscillators is replaced by three new ones, each with a third of the amplitude and frequencies given by $\{f_m - \Delta f, f_m, f_m + \Delta f\} \forall m \in \{3, 4, 5, 6\}$, for an appropriate initial splitting value Δf. The new parameter estimate could then be subjected to NLP to ensure it agrees well with the data. This idea was applied in some circumstances where under-fits had occurred, however unsatisfactory results were common, since the oscillators tended to acquire highly inconsistent amplitudes after unconstrained NLP. An improvement could be made by imposing further constraints on the NLP routine; this is discussed in more detail in Section 6.2. Fundamentally though, as alluded to already, in some circumstances frequency neighbourhoods may be

so crowded that there simply isn't sufficient information for each signal to be accurately resolved.

With three couplings of different magnitude, spin (O) exhibits a "ddd" multiplet structure in which all 8 signals are discernible. The NLP routine performed well in taking the initial guess from the MPM — featuring the correct number of oscillators albeit with spurious phases — and generating a well-phased set of oscillators defining the ddd structure, with a very small associated residual. This highlights that in cases where all signals present in a given frequency neighbourhood are resolvable, effective estimation results are achievable. The oscillators are not perfectly consistent in terms of their relative amplitudes, which one would predict to be equal based on the coupling scheme. This is attributable to the large error associated with significant signal overlap in the frequency neighbourhood.

3.1.3 Cyclosporin A

The estimation routine was also applied to selected regions of a ^{1}H pulse-acquire dataset of cyclosporin A, a cyclic peptide comprising 11 amino acids, in benzene-d_6. The regions considered all comprise signals arising from protons bound to C^{α} atoms in the peptide backbone [117]. The estimation procedure used was equivalent to that used for the andrographolide example.

For the most downfield region considered, featuring signals from spins (A) and (B), the MPM performed admirably, with the two well-resolved dd multiplet structures present accurately parametrised by the 8 oscillators in the model; the NLP routine hardly perturbed the initial guess as a result.

In the middle region, which features a doublet of triplets (dt) multiplet structure from spin (C), low intensity "shoulders" are noticeable; these are likely from low-concentration impurities in the sample. The MPM was able to resolve the dt structure, and characterise some low intensity signals too, though the phases of these are highly inconsistent, with a knock-on effect for the relative amplitudes of signals describing the dt structure (these are expected to abide by the ratio 1:2:1:1:2:1). The NLP routine, by enforcing low phase variance in the oscillators, subtly adjusted the parameter estimate to achieve a set of oscillator amplitudes that are more consistent with this ratio.

The most downfield region contains three separate multiplet structures featuring significant signal overlap in the Fourier domain, particularly involving those corresponding to spins (E) and (F). The MPM produced sufficient oscillators to model these structures (see Table 3.1 for an outline of the structures present). However, particularly in the most crowded section around 4.96 ppm to 4.92 ppm it can be seen that certain oscillators have phases which noticeably deviate from 0°. In applying the NLP routine, not only did the phases become more consistent, but the oscillator amplitudes also become more agreeable. For example, the highest- and lowest-frequency model oscillators associated with the 1:3:6:3:1 quintet of spin (F) (denoted by † in Figures 3.3.b and 3.3.c)

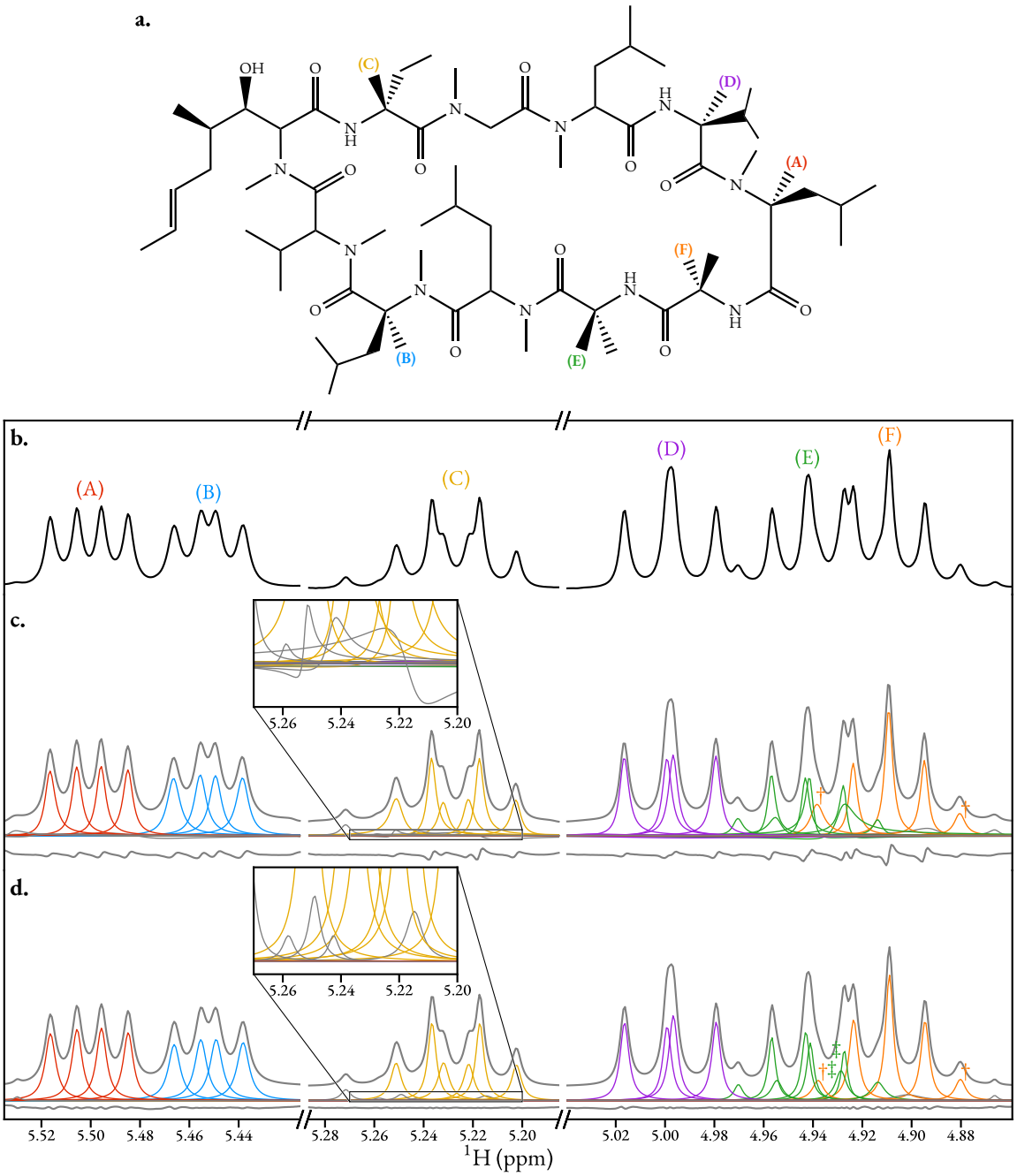


Figure 3.3: The result of applying the estimation routine to selected regions of a pulse-acquire dataset of cyclosporin A in benzene-d_6. **a.** The structure of cyclosporin A, with relevant proton environments highlighted. **b.** Spectral regions considered. **c.** The result of applying the MPM on the regions, with the model order predicted using the MDL. The grey lines above and below the individual oscillator peaks denote the sum of all oscillators and the residual between the data and the model, respectively. **d.** The result after convergence of the NLP routine, presented in the same format as panel c. Coloured oscillators have been mapped to particular cyclosporin A environments. Grey oscillators are not associated with cyclosporin A; these are likely due to impurities in the sample.

acquire amplitudes which are much closer in value after NLP, as expected. The most notable flaw in the final result is associated with the two oscillators marked by ‡ in panel c, both of which are associated with the 1:3:1:3:3:1:3:1 dq structure from spin (E). A 1:3 amplitude ratio is expected between these oscillators. However, in the final NLP result, these had an unexpected ratio of $\approx 1:1$. It is not surprising that greater deviations from the expected result occur in more heavily crowded spectral regions, since there is a larger set of values in the parameter space which will lead to acceptable fits of the data in an RSS sense. Nonetheless, armed simply with knowledge that the data is phased, the routine performs admirably in highlighting how the spectrum breaks down into its various signal components. An improved estimation result could be attained by supplying the NLP routine with more knowledge in the form of parameter constraints. While this has not been implemented in this work, it is discussed in Section 6.2 as a possible pursuit for the future.

3.2 Amplitude-Attenuated Datasets

There are a number of NMR experiments in which the variation of a particular pulse sequence parameter over a number of iterations leads to the generation of a series of FIDs with the same form except for discrepancies in their amplitudes. In this section, an extension of the established 1D estimation technique is described, facilitating the analysis of datasets acquired using these experiments. After the most prominent examples of this class of experiment have been introduced, the methodology of the technique to analyse them is described. Finally, the technique's performance when applied to both simulated and experimental datasets is showcased.

3.2.1 Relaxation Experiments

Any spin system which has been perturbed from its equilibrium position will eventually return to equilibrium due to relaxation. As introduced in Chapter 1, the simplest model of relaxation centers around two processes: longitudinal relaxation and transverse relaxation, whose rates are quantified by the times T_1 and T_2 respectively. Numerous factors affect these times including the rate at which the molecule that the spin is associated withtumbles in space, and its electronic environment. The T_1 and T_2 values of the spins in a sample provide valuable insight into the chemical system being studied, and can also help to guide spectroscopists in the determination of optimal pulse sequence parameters[†].

[†]For example, after a pulse sequence has been run, it is necessary to include a delay (called the *relaxation delay*) prior to re-running it, to enable the spin system's longitudinal magnetisation to sufficiently recover. Having an understanding of the T_1 values associated with a sample can help to determine a relaxation delay which is long enough to facilitate sufficient recovery, but that is not excessive, to avoid an experiment time which is longer than necessary.

Measuring T_1: Inversion Recovery

The inversion recovery experiment involves the simple pulse sequence $180° \rightarrow \tau \rightarrow 90° \rightarrow t^{(1)}$, where $t^{(1)}$ is the period during which FID acquisition takes place. The initial 180° pulse inverts the magnetisation, so that it is along the $-z$ axis in the context of the Bloch model. During τ, the spin system undergoes longitudinal relaxation, with the spin state populations gradually being driven back to their equilibrium configuration. The 90° pulse rotates the magnetisation into the transverse plane, enabling detection. The resultant phase and magnitude of each signal is directly related to the amount of time that longitudinal relaxation is allowed to occur. With $\tau = 0$ s, signals with maximal amplitude, but phases of 180° (equivalent to a negative amplitude) will result[‡]. At the other extreme of $\tau \gg T_1$, the spin system will have reverted back to equilibrium, such that the signals will also have maximal amplitudes, but phases of 0°. By sequentially adjusting τ in a 2D experiment, a series of FIDs will be obtained in which the intensity of a given signal, arising from a spin with longitudinal relaxation time T_1, will vary according to

$$a(\tau) = a_\infty \left(1 - 2\exp\left(-\frac{\tau}{T_1}\right)\right), \tag{3.2}$$

where a_∞ is the intensity of the signal when the spin system has returned to equilibrium. Note that $a(0) = -a_\infty$, as the magnetisation 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 a (simulated) inversion recovery experiment is given by Figure 3.5.d.

Measuring T_2: CPMG

Inspired by the inversion recovery experiment, one might assume that the pulse sequence $90° \rightarrow \tau \rightarrow t^{(1)}$ would be effective for T_2 determination. However the effects 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, $T_2^* < T_2$[§] [118]. 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 CPMG experiment [119, 120], comprising $90°_x \rightarrow \left[\tau \rightarrow 180°_y \rightarrow \tau\right]_n \rightarrow t^{(1)}$, 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. The number of CPMG cycles applied affects the intensity of the resulting signal

[‡]Here it is being assumed that a perfect $90°_y$ pulse is being applied, such that magnetisation along $-z$ will be rotated onto the $-x$ axis.

[§]T_2^* is defined by the expression $1/T_2^* := 1/T_2 + \gamma \Delta B_0$, where ΔB_0 is the variation in magnetic field strength across the sample volume.

according to

$$a(n) = a_0 \exp\left(-\frac{2\tau n}{T_2}\right), \tag{3.3}$$

where a_0 is the intensity where no spin echo cycles were employed, such that the pulse sequence is reduced to the standard pulse-acquire experiment. Recent enhancements to the CPMG approach, including the periodic refocusing of J-evolution by coherence transfer (PROJECT) pulse sequence [121] are also well-suited to T_2 determination.

3.2.2 Diffusion Experiments

NMR is well established as a means of determining the rates of translational diffusion of chemical species [122, 123]. The rate of translation of a species is given by the translational diffusion coefficient, D ($m^2\ s^{-1}$) [124: Chapter 19]. The first illustration of the determination of the diffusion coefficient using NMR came from Stejskal and Tanner, in which they described the pulsed gradient spin echo (PGSE) pulse sequence [125] (Figure 3.4.a). The PGSE sequence consists of a conventional spin-echo ($90°_x \xrightarrow{\tau} 180°_y \xrightarrow{\tau}$ 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 $z \in [-l_z/2, l_z/2]$ along the main field axis, where l_z is the length of the sample lying within the receiver coil (typically about 1.5 cm). 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 ($T\ m^{-1}$)[¶]. Assuming the gradient is applied for a time δ, the spin will precess by the angle $\alpha = -\gamma g z \delta$. After the 180° pulse, the spin's magnetisation will be as follows:

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

Suppose 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 will cause precession by the angle $\beta = -\gamma g(z + \Delta_z)\delta$:

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

Therefore the PFGs have been been employed to encode the change in z-position of the spin after a known amount of time. Extending this idea to an ensemble of many identical non-interacting 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

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

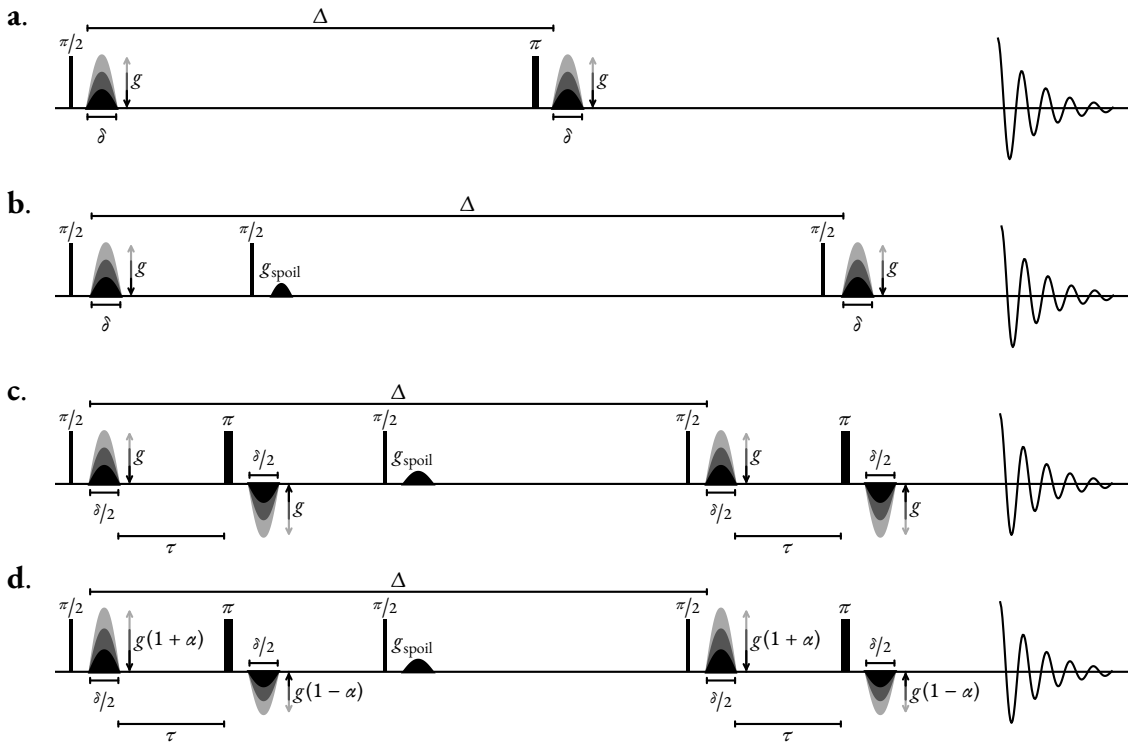


Figure 3.4: Pulse sequences used for the determination of translational diffusion constants. **a.** PGSE, **b.** PGSTE, **c.** PGSTEBP, **d.** One-shot DOSY. RF pulses are denoted by solid rectangles. Diffusion-encoding gradients are denoted by sine-bell shapes with varying shades, indicating that the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

resulting FID. For species which diffuse at a faster rate, such that typical deviations Δ_z are larger, the dephasing effect is more severe, such that a more rapid attenuation results for a given increase in g.

Through consideration of the Bloch-Torrey equations, which extend the classic Bloch equations to account for the effects of diffusion on magnetisation [126], the following equation, known as the *Stejskal-Tanner equation*, may be derived:

$$a(g) = a_0 \exp\left(-\gamma^2 \delta^2 g^2 D \left(\Delta - \frac{\delta}{3}\right)\right), \tag{3.4}$$

where a_0 is the amplitude of a given signal without the application of PFGs, δ is the duration of each PFG (s), and Δ is the delay between the PFGs, often known as the *diffusion time* (s). While Equation 3.4 is widely stated in the diffusion NMR literature, it is only strictly applicable when the PGSE sequence is used, and PFGs with rectangular amplitude profiles are applied‖. As will be

‖Rectangular PFGs (i.e. those in which there is an infinitesimal time to rise to full strength, and to fall back to zero) are technically impossible to achieve as they would require gradient coils with zero inductance, though in practice it is possible to generate gradients with behaviour close to this.

discussed soon, the exact form of the pulse sequence impacts the functional form by which $a(g)$ varies.

Tanner introduced a variant of the original PGSE experiment called pulsed gradient stimulated echo (PGSTE) [127] (Figure 3.4.b). Instead of the diffusion period being bisected by a 180° pulse, PGSTE features two 90° pulses at the beginning and end of the diffusion period. Hence, the extent of signal loss due to relaxation is affected by T_1 in PGSTE, as opposed to T_2 in PGSE. PGSTE is therefore favoured in scenarios where $T_1 \gg T_2$, as improved data sensitivity is achievable.

Both PGSE and PGSTE employ *monopolar* PFGs for diffusion encoding. Experiments also exist which employ *bipolar* gradient elements, which consist of a PFG, followed by a 180° pulse, and then a second PFG with the opposite polarity to the first [128, 129]. The pulsed gradient stimulated echo with bipolar gradients (PGSTEBP) experiment (Figure 3.4.c) can be thought of as the bipolar analogue of PGSTE. 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 [130] (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 $\alpha \in (0, 1)$; commonly, $\alpha = 0.2$.

The most common means of conducting a diffusion experiment is to run one of the pulse sequences mentioned above (or variants thereof) with g varied across increments. It is virtually always the case that the FID's signal amplitudes will abide by the following general form of the Stejskal-Tanner equation, regardless of the exact pulse sequence used:

$$a(g) = a_0 \exp\left(-cg^2 D\right), \tag{3.5}$$

for some constant c (s T^{-2}). The functional form of c depends on the type of experiment used, as well as the amplitude profile (*shape*) 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 [131]. In general, c is as follows:

$$c = \gamma^2 \delta^2 \sigma^2 \Delta'. \tag{3.6}$$

σ is the *shape factor* of the PFGs (*vide infra*), and Δ' is the effective time that diffusion is allowed

to occur. Examples of the form that Δ' takes include:

$$\Delta' = \begin{cases} \Delta + 2\left(\kappa - \lambda\right)\delta & \text{PGSE, PGSTE} \\ \Delta + \frac{(2\kappa - 2\lambda - 1)\delta}{4} - \frac{\tau}{2} & \text{PGSTEBP} \\ \Delta + \frac{(\kappa - \lambda)\left(\alpha^2 + 1\right)\delta}{2} + \frac{(\delta + 2\tau)\left(\alpha^2 - 1\right)}{4} & \text{One-shot} \end{cases} . \tag{3.7}$$

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 of the gradient amplitude 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:

$$S(\epsilon) = \int_0^{\epsilon} s\left(\epsilon'\right) \mathrm{d}\epsilon' \quad \forall \epsilon \in [0, 1]. \tag{3.8}$$

The corresponding definition of S for a discrete gradient made of N steps, $\boldsymbol{s} \in \mathbb{R}^N$, is

$$S_n = \frac{1}{n} \sum_{i=1}^{n} s_i \quad \forall n \in \{1, \cdots, N\}, \tag{3.9}$$

σ, λ, and κ are related to the shape function's cumulative distribution as follows:

$$\sigma = S(1), \tag{3.10a}$$

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

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

with their discrete counterparts being:

$$\sigma = S_N \tag{3.11a}$$

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

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

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 for the PGSE experiment (Equation 3.4) comes from inserting Equation 3.7 into Equation 3.6, assuming a rectangular shape factor for the PFG: $\sigma = 1$, $\lambda = 1/2$, and $\kappa = \frac{1}{3}$. In many situations, Δ dominates in the expression of Δ', and so ensur-

ing the correct form of c could be seen as excessive. However, especially when Δ is not orders of magnitude greater than δ, the exact form of Δ' used in Equation 3.6 will be important for accurate measurements of the diffusion coefficient.

3.2.3 Analysing the Datasets

Determining the quantity of interest from the above experiments is most typically done using the following procedure:

1. The series of FIDs are processed in an equivalent fashion, leading to a set of 1D spectra.
2. Peak-picking is performed to locate the chemical shift values which correspond to signals of interest within the spectra.
3. For each chemical shift of interest, the values of the spectra are extracted, and the relevant function describing how signal amplitudes decay (see Table 3.2) is fit to the values.

This method is *univariate* in the sense that it considers a single sample (chemical shift) in isolation from the rest of the data. It is popular to visualise the result of univariate fitting by plotting 2D pseudo-spectra, featuring axes denoting (a) chemical shift, and (b) the parameter of interest. Distributions taking into account the predicted value of interest and the associated uncertainty are plotted at each chemical shift. In the context of diffusion NMR, this form of data analysis is referred to as diffusion-ordered spectroscopy (DOSY) [123]. The DOSY approach works well in cases where peaks of interest are intense and well-resolved. However, when there is considerable peak overlap, accurately determining the parameter of interest associated with a given signal becomes more challenging, since nearby signals will influence the rate of amplitude decay. An aggregated value of the decay rates of the contributing signals will be predicted using the above approach. In such cases, it is possible to instead fit a superposition of functions to the spectral amplitudes, as a means of separating out contributions from the overlapping signals [132]. However, this process is usually only effective when the data has very high SNR, and the quantities of interest associated with the overlapping signals differ considerably.

An alternative approach is to use a *multivariate* analysis, in which entire spectrum is considered holistically, and an attempt is made to decompose the spectrum into components with different decay rates. Prominent examples of multivariate methods developed for diffusion data analysis are direct exponential curve resolution algorithm (DECRA) [133] and *speedy* component resolution (SCORE) [134], an improvement of the original CORE [135, 136]. Multivariate techniques tend to only be effective in situations where a few (2 or 3) different signal decay rates are present. In diffusion NMR, each molecule in the sample will give rise to the same decay rate across its signals (effects such as chemical exchange can lead to exceptions to this: Figure 3.6 provides an example of this). As such, only a few decay rates typically govern the form of diffusion data, making

multivariate approaches feasible. Conversely, multivariate methods are generally unsuitable for inversion recovery and CPMG datasets: each spin in a sample will possess different T_1 and T_2 values, such that a far greater number of decay rates will be present.

3.2.4 Methodology

A straightforward extension of 1D parametric estimation provides a route to estimating amplitude-attenuated datasets in a fashion with similarities to the traditional univariate (DOSY) approach. The theory of this method will now be discussed.

Suppose one of the experiments mentioned above is run with $K \in \mathbb{N}$ increments, such that there is a vector $\boldsymbol{p} \in \mathbb{R}^K$ comprising the values of the pulse sequence variable used to acquire each 1D FID in the data. The complete dataset is expected to take the form $\boldsymbol{Y} \in \mathbb{C}^{K\times N}$ in which the value of the pulse sequence variable affects the amplitudes of the contributing signals:

$$\begin{gathered} y_{k,n} = \sum_{m=1}^{M} a_{k,m} \exp(\mathrm{i}\phi_m) \exp((2\pi\mathrm{i}(f_m - f_{\mathrm{off}}) - \eta_m) n\Delta_t), \\ \forall k \in \{1, \cdots, K\},\ \forall n \in \{0, \cdots, N-1\}. \end{gathered} \tag{3.12}$$

$\boldsymbol{A} \in \mathbb{R}^{K\times M}$ is a matrix of the signal amplitudes across the FIDs:

$$\boldsymbol{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,M} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{K,1} & a_{K,2} & \cdots & a_{K,M} \end{bmatrix} \tag{3.13}$$

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

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{a}_1 & \cdots & \boldsymbol{a}_K & \boldsymbol{\phi}^{\mathrm{T}} & \boldsymbol{f}^{\mathrm{T}} & \boldsymbol{\eta}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{3.14}$$

where $\boldsymbol{a}_k \in \mathbb{R}^M$ denotes the relevant row in $\boldsymbol{A}$. The amplitudes are a function of the pulse sequence variable with the following form:

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

$\boldsymbol{a}_0 \in \mathbb{R}^M$ is a vector of the "maximal amplitudes" of the signals, and $\boldsymbol{\psi} \in \mathbb{R}^M$ is a vector of the parameter of interest (T_1, T_2, D) associated with each signal. $a_{0,m}$ can be thought of as the largest possible amplitude that could be obtained for signal m with the given pulse sequence:

- In an inversion recovery experiment, the largest amplitude is achieved as $\tau \to \infty$, as the spin

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

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

system will have returned back to equilibrium prior to the 90° pulse.

- With CPMG experiments, the largest 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.

The function $\mathcal{A}$ describes how the amplitudes of signals 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, the forms of $\mathcal{A}$ are listed in Table 3.2.

Estimating the Datasets

The close relationship between the FIDs within the dataset means that completely estimating all of them separately, without exploiting information from previously estimated FIDs, is unnecessary. Instead, after the first increment is estimated without prior knowledge, yielding $\boldsymbol{\theta}_1 \in \mathbb{R}^{4M}$, the phases, frequencies and damping factors, all of which are anticipated to remain the same across the FIDs, are fixed. Subsequent parameter estimates are determined by taking the result from the previous FID, and subjecting it to an NLP routine in which only the amplitudes are optimised. Thus, estimating each subsequent iteration is reduced to the problem**

$$\boldsymbol{a}_k = \underset{\boldsymbol{a}\in\mathbb{R}^M}{\arg\min}\, \mathcal{F}(\boldsymbol{a}\,|\,\boldsymbol{\phi}_1, \boldsymbol{f}_1, \boldsymbol{\eta}_1, \boldsymbol{y}_k) \quad \forall k \in \{2,\cdots,K\}. \tag{3.16}$$

An NLP routine can solve this very efficiently, typically in a few iterations, on account of the linear dependence of the model with respect to the oscillator amplitudes. The linear dependence means that second derivatives of the model are all zero (see Equation A.8a), 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 compute the

** $\mathcal{F}$ in Equation 3.16 reads as "the fidelity with respect to the amplitudes $\boldsymbol{a}$, given phases $\boldsymbol{\phi}_1$, frequencies $\boldsymbol{f}_1$, damping factors $\boldsymbol{\eta}_1$, and FID $\boldsymbol{y}_k$". The fidelity has exactly the same mathematical form as Equation 2.43, though it has been emphasised that the phases, frequencies and damping factors are no longer variables to be optimised, but fixed parameters.

following:

$$\boldsymbol{a}_k = \boldsymbol{Z}^{+}\boldsymbol{y}_k, \tag{3.17a}$$

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

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

An outline of the estimation procedure is provided by Algorithm 3.1.

Algorithm 3.1 The proposed routine for estimating a sequence of amplitude-attenuated FIDs. NLPAmp denotes a routine which is akin to NLP (Algorithm 2.2), except only amplitudes are optimised; phases, frequencies and damping factors are fixed.

procedure EstimateAmpAttenuated($\boldsymbol{Y} \in \mathbb{C}^{K\times N}, l_{\mathrm{I}}, r_{\mathrm{I}}, l_{\mathrm{N}}, r_{\mathrm{N}}, \chi \in \mathbb{R}_{>1}, M \in \mathbb{N}_0$)
  $\boldsymbol{\theta}_1, \boldsymbol{\epsilon}_1 \leftarrow$ Estimate1D $(\boldsymbol{y}_1, l_{\mathrm{I}}, r_{\mathrm{I}}, l_{\mathrm{N}}, r_{\mathrm{N}}, \chi, M)$; ▷ Estimate first increment, see Algorithm 2.4.
  $M \leftarrow {}^{\mathrm{len}(\boldsymbol{\theta}_1)}/_4$;
  $\boldsymbol{\theta}, \boldsymbol{\epsilon} \leftarrow \boldsymbol{0} \in \mathbb{R}^{(K+3)M}, \boldsymbol{0} \in \mathbb{R}^{(K+3)M}$; ▷ Initialise complete parameter vector.
  $\boldsymbol{\theta}[:M], \boldsymbol{\epsilon}[:M] \leftarrow \boldsymbol{\theta}_1[:M], \boldsymbol{\epsilon}_1[:M]$; ▷ Amplitudes for first increment.
  $\boldsymbol{\theta}[KM:], \boldsymbol{\epsilon}[KM:] \leftarrow \boldsymbol{\theta}_1[M:], \boldsymbol{\epsilon}_1[M:]$; ▷ Phases, frequencies and damping factors, which are fixed.
  for $k = 2, \cdots, K$ **do**
    $\widetilde{\boldsymbol{y}} \leftarrow$ Filter1D $(\boldsymbol{y}_k, l_{\mathrm{I}}, r_{\mathrm{I}}, l_{\mathrm{N}}, r_{\mathrm{N}}, \chi,)$; ▷ Algorithm 2.3.
    $\boldsymbol{\theta}_k, \boldsymbol{\epsilon}_k \leftarrow$ NLPAmp $(\widetilde{\boldsymbol{y}}, \boldsymbol{\theta}_{k-1})$; ▷ Similar to Algorithm 2.2, though see the caption.
    $\boldsymbol{\theta}[kM:(k+1)M], \boldsymbol{\epsilon}[kM:(k+1)M] \leftarrow \boldsymbol{\theta}_k[:M], \boldsymbol{\epsilon}_k[:M]$; ▷ Extract amplitudes.
  end for
  return $\boldsymbol{\theta}, \boldsymbol{\epsilon}$;
end procedure

Determining the Parameter of Interest

Having generated a parameter estimate for each FID in the dataset, focus now moves to determining the parameter of interest for each signal. For each oscillator $m \in \{1, \cdots, M\}$, the maximal amplitude and parameter of interest are determined by solving the following problem:

$$\boldsymbol{\vartheta}_m^{(*)} = \underset{\boldsymbol{\vartheta}\in\mathbb{R}^2}{\arg\min} \left\| \boldsymbol{a}_m - a_0 \mathcal{A}\left(\psi \mid \boldsymbol{p}\right) \right\|^2, \tag{3.18a}$$

$$\boldsymbol{\vartheta} = \begin{bmatrix} a_0 & \psi \end{bmatrix}^{\mathrm{T}} \implies \boldsymbol{\vartheta}_m^{(*)} = \begin{bmatrix} a_{0,m}^{(*)} & \psi_m^{(*)} \end{bmatrix}^{\mathrm{T}}, \tag{3.18b}$$

where $\boldsymbol{a}_m$ corresponds to the m^{th} column of $\boldsymbol{A}$. This is another example of an RSS problem, which can be solved using an NLP routine as already described. The gradient vector and Hessian matrix of the fidelity take very similar functional forms to those for FID estimation for this reason (see

Equations 2.44a and 2.44b), and are as follows $\forall i, j \in \{1, 2\}$:

$$g(\vartheta)_i = -2\left\langle \boldsymbol{a}_m - a_0\mathcal{A}(\psi|\boldsymbol{p}), \frac{\partial a_0\mathcal{A}(\psi|\boldsymbol{p})}{\partial\vartheta_i}\right\rangle, \tag{3.19a}$$

$$h(\vartheta)_{i,j} = 2\left(\left\langle \frac{\partial a_0\mathcal{A}(\psi|\boldsymbol{p})}{\partial\vartheta_i}, \frac{\partial a_0\mathcal{A}(\psi|\boldsymbol{p})}{\partial\vartheta_j}\right\rangle - \left\langle \boldsymbol{a}_m - a_0\mathcal{A}(\psi|\boldsymbol{p}), \frac{\partial^2 a_0\mathcal{A}(\psi|\boldsymbol{p})}{\partial\vartheta_i\partial\vartheta_j}\right\rangle\right), \tag{3.19b}$$

$$\tag{3.19c}$$

with explicit expressions for the requisite first and second derivatives being

$$\frac{\partial a_0\mathcal{A}(\psi|\boldsymbol{p})}{\partial a_0} = \mathcal{A}(\psi|\boldsymbol{p}), \tag{3.20a}$$

$$\frac{\partial a_0\mathcal{A}(\psi|\boldsymbol{p})}{\partial\psi} = a_0\frac{\partial\mathcal{A}(\psi|\boldsymbol{p})}{\partial\psi}, \tag{3.20b}$$

$$\frac{\partial^2 a_0\mathcal{A}(\psi|\boldsymbol{p})}{\partial a_0^2} = 0, \tag{3.20c}$$

$$\frac{\partial^2 a_0\mathcal{A}(\psi|\boldsymbol{p})}{\partial\psi^2} = a_0\frac{\partial^2\mathcal{A}(\psi|\boldsymbol{p})}{\partial\psi^2}, \tag{3.20d}$$

$$\frac{\partial^2 a_0\mathcal{A}(\psi|\boldsymbol{p})}{\partial a_0\partial\psi} = \frac{\partial^2 a_0\mathcal{A}(\psi|\boldsymbol{p})}{\partial\psi\partial a_0} = \frac{\partial\mathcal{A}(\psi|\boldsymbol{p})}{\partial\psi}. \tag{3.20e}$$

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

Displaying Results

It is possible to visualise the estimation results acquired by this method in a similar fashion to DOSY analysis. For each oscillator in the estimated model, a 2D array is generated, corresponding to the outer product of the FT of the model oscillator ($\boldsymbol{s}_m$) and a distribution describing the predicted value of ψ ($\boldsymbol{d}_m$).

$$\mathbb{R}^{R\times N} \ni \boldsymbol{S} = \sum_{m=1}^{M} \boldsymbol{d}_m \otimes \boldsymbol{s}_m, \tag{3.21a}$$

$$\boldsymbol{s}_m = \Re\left(\mathrm{FT}\left(\boldsymbol{x}_m\right)\right), \tag{3.21b}$$

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

where $R \in \mathbb{N}$ is the number of samples to generate the distribution from. The exact form that the distribution $\boldsymbol{d}_m$ should take is arbitrary, though it should indicate two key pieces of information:

- Its maximum should coincide with the predicted value of ψ_m.

- Its broadness 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 broadening factor which can be chosen to ensure clear visibility of the peaks[††]:

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

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

$p_{\min}$ and $p_{\max}$ specify the range of values over which to generate the distribution. The errors associated with the parameter of interest can be determined as follows:

$$\epsilon_m = \sqrt{\frac{\left\|\boldsymbol{a}_m - a_{0,m}^{(*)}\mathcal{A}\left(\psi_m^{(*)}|\boldsymbol{p}\right)\right\|^2 \left[\boldsymbol{H}\left(\boldsymbol{\vartheta}_m^{(*)}\right)^{-1}\right]_{2,2}}{K-1}}, \tag{3.23}$$

where $\boldsymbol{H}\left(\boldsymbol{\vartheta}_m^{(*)}\right) \in \mathbb{C}^{2\times 2}$ is the Hessian at convergence, comprising elements of the form $h\left(\boldsymbol{\vartheta}_m^{(*)}\right)_{i,j}$ (see Equation 3.19b). The derivation of Equation 3.23 is akin to that for the errors associated with FID estimation; more detail on FID estimation errors is provided in Appendix A.2.2.

3.2.5 Results

"Five Multiplets"

Figure 3.5 shows the result achieved by applying the method outlined in Algorithm 3.1 to three simulated inversion recovery datasets featuring 5 overlapping ddd multiplet structures. Each dataset was simulated using Spinach, using spin systems which were defined such that within a known region of the generated spectrum (0.15 ppm to −0.15 ppm) five ddd multiplet structures with significant overlap, abiding by the weak coupling approximation, were present. Constraints were placed on the shifts and couplings of the spin system to ensure that no two signals 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 $\mathcal{U}(0.2\,\text{s}, 0.6\,\text{s})$. Relaxation phenomena were modelled using the "extended T_1/T_2 approximation" [137] (see Appendix C.1.3 for more information). Each FID generated dataset was corrupted by noise with a target SNR of 40 dB. The MDL was used for

[††]In cases where the error in estimating ψ_m is very small, it may be that the distribution derived from Equation 3.22 satisfies $\max(\boldsymbol{d}_m) = 0$ if the peak of the distribution does not closely coincide with any of the sampled values. The affected oscillator will not appear in the DOSY-like spectrum under these circumstances. Broadening the distribution (i.e. setting $c > 1$) resolves this.

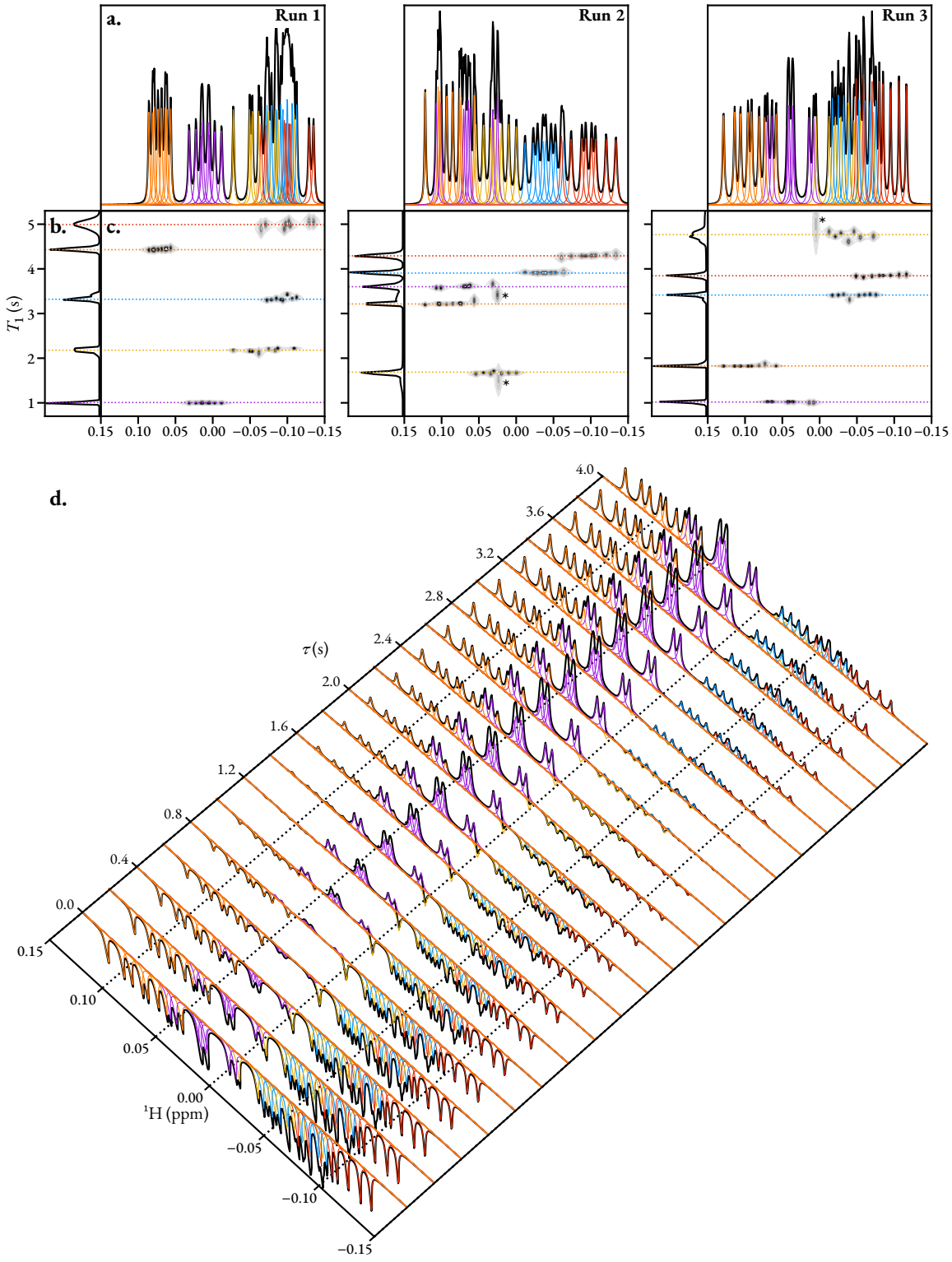


Figure 3.5: Three examples of results generated when considering simulated inversion recovery datasets comprising five ddd multiplet structures. **a.** Plot of the result generated for the first increment (τ = 0 s), with all the plots multiplied by −1. Black: spectrum of the data. Coloured lines: peaks corresponding to oscillators in the estimated model. Oscillators with the same colour are components of the same multiplet. **b.** Distribution of T_1 values, generated by projecting the array in panel c onto the y-axis. **c.** DOSY-style representation of the result, generated using Equation 3.21, with p_{min} = 0.7 s, p_{max} = 5.3 s, c = 40, R = 128. Dashed horizontal lines denote the true T_1 values for each spin. **d.** Estimation result of each increment for Run 3, illustrating the evolution of the amplitudes of each oscillator as a function of τ.

model order prediction; it successfully predicted that 40 signals were present on each occasion.

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 T_1 of the spin giving rise to it (see Figure 3.5.b). As is to be expected, in scenarios where little overlap existed between signals with different decay rates, T_1 predictions tended to be more accurate, with smaller associated errors (see for example the purple and orange multiplets in Run 1. Nevertheless, good estimates could still be obtained in cases of severe overlap between signals from different spins (see the red, blue and yellow multiplets in Run 1). Reassuringly, particular oscillators for which the estimate of T_1 was notably far from the true value tended to be associated with large errors, with examples denoted with an asterisk in Figure 3.5.c.

Andrographolide Diffusion

Figure 3.6 shows the result of applying the estimation technique on a oneshot DOSY dataset of andrographolide in unfresh DMSO-d_6 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 $3.7 \times 10^{-10}\,\mathrm{m^2\,s^{-1}}$. 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 [138]. The diffusion profile generated suggests a diffusion constant of andrographolide of $2.07 \times 10^{-10}\,\mathrm{m^2\,s^{-1}}$. The predicted diffusion constant for each estimated oscillator, especially those with greater intensity, shows decent consistency. Lower intensity oscillators, especially those which significantly overlap with others, tended to be associated with less consistent diffusion constants and larger errors with examples of this phenomenon apparent in Figure 3.6.d. A few oscillators also show a significant deviation at around 2.5 ppm. This is due to the presence of a 1:2:3:2:1 quintet from partially protonated DMSO, due to proton exchange with the residual water[‡‡]. As the data are insufficiently resolved to enable the separation of andrographoilde and DMSO signals, oscillators exist in the model which have amplitude profiles influenced by both species, leading to an aggregated diffusion constant. As $D_{\mathrm{DMSO}} > D_{\mathrm{andrographoilide}}$, the affected oscillators possess larger predicted diffusion constants.

Glucose/Valine/Threonine Diffusion

Another diffusion example is provided by Figure 3.7, where the estimation routine was applied to a dataset derived from a sample comprising the molecules L-valine ($\mathcal{M}_r = 117.148\,\mathrm{g\,mol^{-1}}$), L-

[‡‡]This multiplet structure arises from the ^{1}H in DMSO coupling to two equivalent spin-1 ^{2}H nuclei. The relative amplitudes of the signals can be deduced by considering the "trinomial triangle", an extension of the more familiar Pascal's triangle, the latter being applicable to couplings involving equivalent spin-1/2 nuclei.

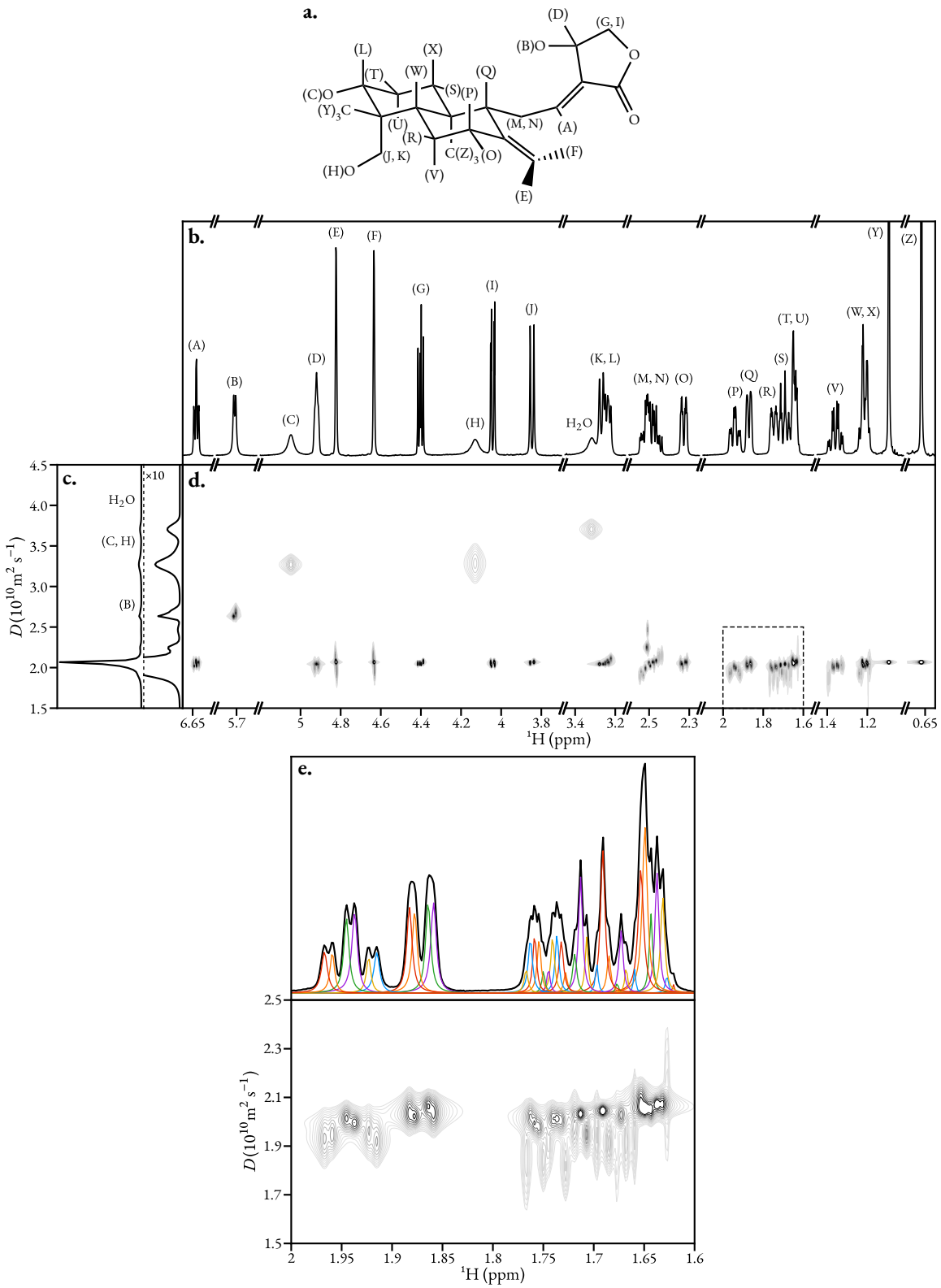


Figure 3.6: The result of estimating a diffusion dataset of andrographolide in unfresh DMSO-d_6. **a.** Structure of andrographolide. **b.** 1D spectrum. **c.** Diffusion profile obtained by projecting the contour plot in c onto the y-axis. **d.** Contour plot mapping estimated oscillators to diffusion constants, with $p_{\text{min}} = 1.5 \times 10^{-10}$ m^2 s^{-1}, $p_{\text{max}} = 4.5 \times 10^{-10}$ m^2 s^{-1}, $c = 2.5$, $R = 128$. **e.** Magnified view of the 2 ppm to 1.6 ppm spectral range (see the dashed box in panel d) with estimated oscillator peaks plotted.

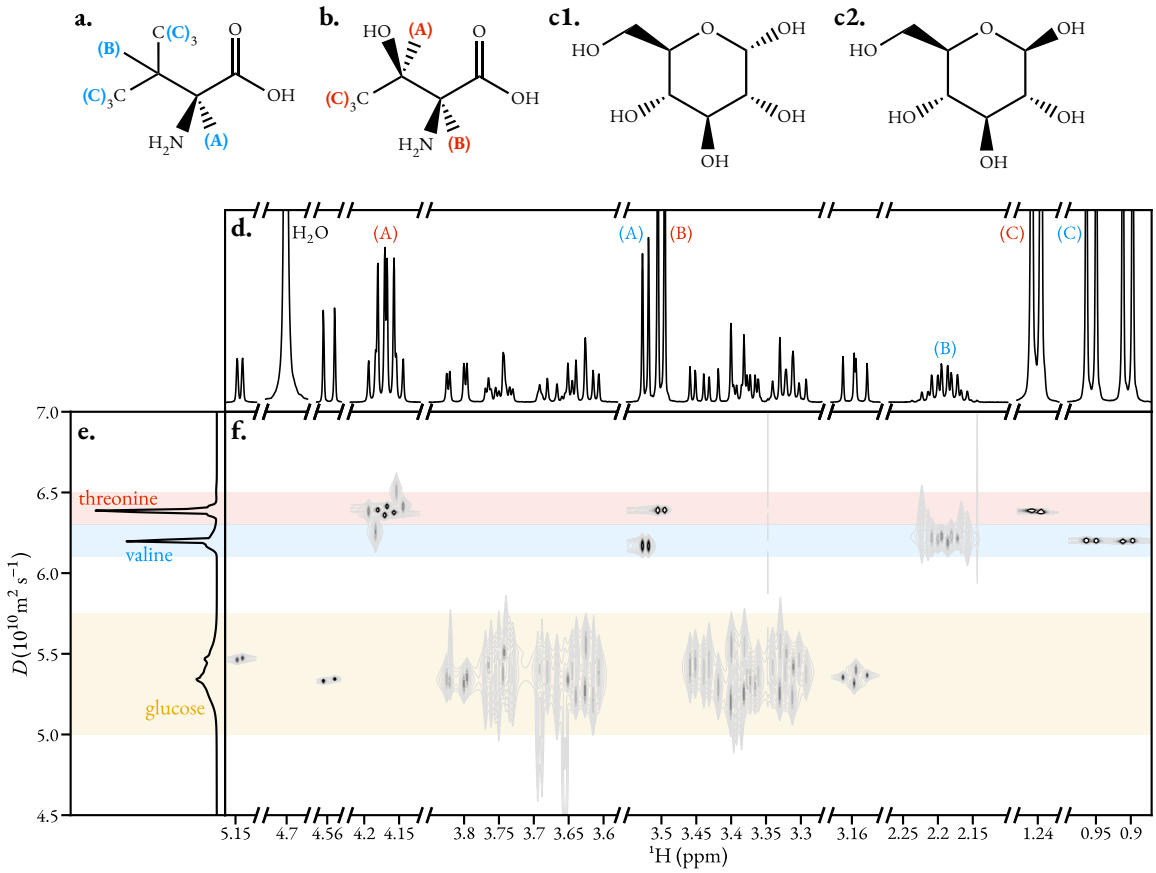


Figure 3.7: The result of estimating a diffusion dataset for a mixture of L-threonine, L-valine and D-(+)-glucose in D_2O. **a.** Structure of L-valine. **b.** Structure of L-threonine. **c1.** Structure of α-D-glucopyroanose. **c2.** Structure of β-D-glucopyroanose. **d.** 1D spectrum, taken from the first FID of the diffusion dataset. **e.** Diffusion coefficient distribution. **f.** DOSY-style plot of chemical shifts vs diffusion constant, generated using Equation 3.22, with $p_{\text{min}} = 4.5 \times 10^{-10}$ m^2 s^{-1}, $p_{\text{max}} = 7 \times 10^{-10}$ m^2 s^{-1}, $R = 256$, and $c = 1.5$.

threonine (M_r = 119.120 g mol^{-1}), and D-(+)-glucose (M_r = 180.156 g mol^{-1}) dissolved in D_2O at 298 K. Not included in the figure is the result for the water signal, for which a diffusion constant of 1.88 × 10^{-9} m^2 s^{-1} was determined. The routine achieved decent separation of the three species in the sample. Predicted diffusion coefficients for valine and threonine were 6.20 × 10^{-10} m^2 s^{-1} and 6.39 × 10^{-10} m^2 s^{-1}, respectively. For glucose, the situation is complicated by the presence of two major anomeric forms: α-D-glucopyroanose and β-D-glucopyroanose[§§] [139: Chapter 3]. There is some evidence of separation of these anomers, principally due to the downfield doublets at 5.15 ppm (α-anomer) and 4.56 ppm (β-anomer), which have estimated diffusion coefficients of 5.47 × 10^{-10} m^2 s^{-1} and 5.34 × 10^{-10} m^2 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 species. The small difference in diffusion constants associated with the anomers is likely related to how effectively they are able to hydrogen-bond to the solvent.

As is to be expected, signals which are of greater intensity, and which are more clearly resolved, enable the determination of diffusion coefficients with smaller associated errors. A clear example of this behaviour can be recognised when considering the valine result (see the area shaded blue in Figure 3.7.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. Similar arguments can be made for the signals which correspond to the other species in the mixture. Application of multivariate methods on the dataset was unsuccessful at extracting diffusion information for the separate components; both DECRA and SCORE could extract the water signal from the rest of the dataset, with the two components having associated diffusion coefficients of 6.31 × 10^{-10} m^2 s^{-1} (an aggregate of the valine, threonine, and glucose signals) and 1.88 × 10^{-9} m^2 s^{-1}, respectively.

3.3 Ideal Spectra from Single-Chirp Excitation

There are numerous nuclei of interest in NMR with very wide chemical shift ranges, including ^{19}F (of particular interest in the pharmaceutical industry), ^{31}P, and ^{195}Pt. Attaining spectra covering the entire chemical shift range of such nuclei 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 that of the transmitter [23: Section 3.4.1]. One popular means of achieving ultra-broadband excitation, in which a consistent amplitude- and phase-profile across a spectral

[§§] The equilibrium mixture in water comprises 38% of the α isomer and 62% of the β isomer. 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. A tiny amount of the open-chain form will also be present, though in a negligible quantity.

window of tens or even hundreds of kHz the achieved, is to use frequency-swept (FS) pulses, during which the frequency of RF irradiation varies with time [140]. One of the most common classes of FS pulses are those where the variation of excitation frequency with time is linear, with such pulses commonly referred to as *chirp* pulses. The application of a single 90° chirp pulse to achieve ultra-broadband excitation, while involving a simple and short pulse sequence, 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 [141, 142, 143, 144, 145].

With knowledge of the form of the chirp pulse, the expected phase of a particular signal 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 a far shorter pulse sequence than state of the art methods such as chirped, ordered pulses for ultra-broadband spectroscopy (CHORUS) [144, 145], where both a 90° chirp pulse, and two 180° chirp pulses are applied; spectra with improved signal intensity could therefore be realised. Here, a description of the technique is presented, followed by an illustration of its performance on both a simulated and an experimental dataset.

3.3.1 Chirp Excitation

Focus is limited here to chirp pulses which sweep from low to high frequencies; they are parameterised by their duration τ_{p} (s), excitation bandwidth ΔF (Hz), and RF "amplitude" ω_{RF} (Hz). The frequencies that a pulse sweeps through are in the range $[f_{\mathrm{off}} - {}^{1}\!/_{2}\Delta F, f_{\mathrm{off}} + {}^{1}\!/_{2}\Delta F]$, and the rate at which the frequency of the chirp is increased (the sweep rate) is given by ${}^{\Delta F}\!/_{\tau_{\mathrm{p}}}$. 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, and which is routinely corrected for by phase correction. The various chirp pulse parameters are inter-related as follows [145, 146]:

$$\omega_{\mathrm{RF}} = \sqrt{\frac{\Delta F Q}{2\pi\tau_{\mathrm{p}}}}, \tag{3.24}$$

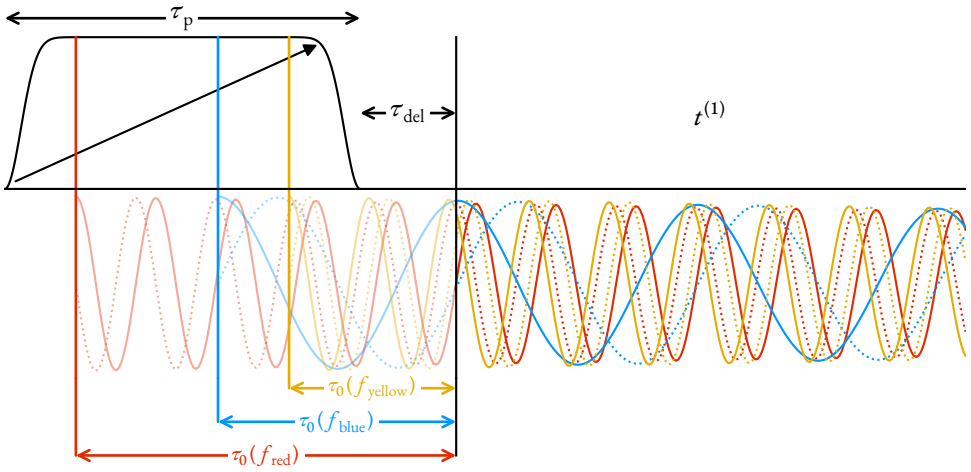


Figure 3.8: An illustration of an experiment comprising a single chirp pulse sweeping low to high frequencies of duration τ_p, followed by a pre-scan delay period of time τ_{del}, prior to acquisition. The fate of three resonances with different frequencies is denoted, with $f_{red} < f_{blue} < f_{yellow}$. Each resonance is excited at different points in time, with lower frequency resonances being excited earlier, such that each of these evolves for different amounts of time prior to acquisition (τ_0). The resulting FID comprises signals whose phases depend on their frequencies quadratically. Coloured oscillations denote the evolution of each resonance, with solid and dashed lines representing real and imaginary components, respectively. It is assumed that the pulse rotates each resonance so as to be in phase with the receiver at the point of excitation.

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

$$Q = \frac{2}{\pi} \ln\left(\frac{2}{\cos(\beta) + 1}\right), \tag{3.25}$$

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 $Q \approx 0.441$. The combination used in examples in this work is $\Delta F = 400\,\text{kHz}$, $\tau_p = 100\,\mu\text{s}$, and $\omega_{RF} \approx 16.8\,\text{kHz}$. For a pulse with sufficiently low ω_{RF} — this 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 depicted 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

$$\tau_0(f) = \tau_{del} + \frac{\tau_p}{2} - \frac{(f - f_{off})\tau_p}{2\Delta F}. \tag{3.26}$$

$\tau_{del} + \tau_p/2$ 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 a frequency smaller than that of 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 of a signal a function of its

frequency can be approximated as [145]

$$\phi(f) = \phi_0 + 2\pi\left(\tau_{\text{del}} + \frac{\tau_{\text{p}}}{2}\right)(f - f_{\text{off}}) - 2\pi\left(\frac{\tau_{\text{p}}}{2\Delta F}\right)(f - f_{\text{off}})^2. \tag{3.27}$$

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

$$s_\phi(f) = s(f)\exp(-\mathrm{i}\phi(f)). \tag{3.28}$$

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:

1. The "complete" signal, which starts at the time of excitation.
2. 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 comprises a broad sinc function with its maximum at the signal 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 in the low-frequency region. Figures 3.9.b and 3.10.b both provide examples 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 a synthetic 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 is given by $-\tau_0$, with τ_0 defined in Equation 3.26. The resulting corrected FID $\boldsymbol{y} \in \mathbb{C}^N$ is defined as

$$y_n = \sum_{m=1}^{M} a_m \exp(\mathrm{i}\phi_m)\exp\big((2\pi\mathrm{i}(f_m - f_{\text{off}}) - \eta_m)(n\Delta_t - \tau_0(f_m))\big) \quad \forall n \in \{0, \cdots, N-1\}. \tag{3.29}$$

This concept has similarities to the use of LP in order to back-propagate an FID to correct for corrupted initial points. However, a holistic approach such as LP cannot be used in this application,

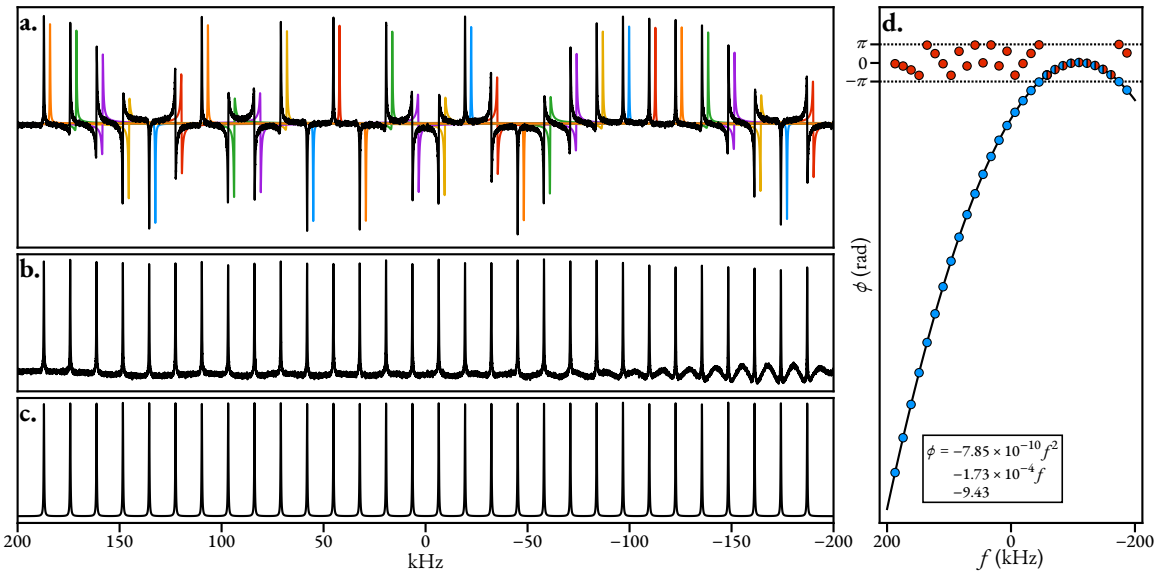


Figure 3.9: A comparison of quadratic phase correction vs frequency-dependent back-propagation in treating simulated single-chirp excitation data. **a.** Simulated spectrum for a spin system comprising 30 spins with uniformly-separated resonance frequencies. The data were generated with $N = 2^{14}$, $f_{\text{sw}} = 400\,\text{kHz}$, $f_{\text{off}} = 0\,\text{Hz}$, $\tau_{\text{p}} = 100\,\mu\text{s}$, $\tau_{\text{del}} = 0\,\text{s}$, $\Delta F = 400\,\text{kHz}$. Coloured lines depict individual oscillators generated using the MPM; these have been slightly offset for clarity. **b.** Spectrum generated using quadratic phase correction (Equation 3.28). **c.** Spectrum generated using the back-propagation approach. **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 their quadratic dependence on frequency. Black curve: quadratic fit of the blue points. The equation of the quadratic is stated.

as each signal component must be treated differently according to its frequency.

Thus far in this work, it has been assumed that all signals which make up a dataset are of the same phase. Of course this isn't the case for FIDs generated from single-chirp excitation. As such, it is inappropriate to incorporate the variance of oscillator phases in the fidelity for NLP. For the examples presented in this section, NLP was not applied at all; 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 the former, a simulated dataset is considered, comprising 30 evenly-spaced signals, and generated using Equation 3.29 with $-\tau_0(f_m)$ replaced with $+\tau_0(f_m)$. Other relevant parameters used are stated in the caption. A very large damping factor ($1000\,\text{s}^{-1}$) was assigned to each oscillator, as this augments the baseline distortions in the spectrum. Noise was added to the FID, with a target SNR of 25 dB.

The spectrum after quadratic phase correction (Figure 3.9.b) exhibits the undesired baseline distortions as discussed, with more intense, narrower baseline distortions associated with lower-frequency signals. The MPM was used to estimate the FID's parameters, with 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 Figures 2.4.a1 and 2.4.a2). Estimating a truncated FID comprising the first 2048 points of the data is justifiable here, since all signal 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 Figure 3.9.c, which is well-phased, and does not suffer from baseline distortion. The variation of the estimated oscillator phases against their frequencies is plotted in Figure 3.9.d, with the quadratic dependence clearly illustrated; fitting the blue points to a quadratic function yielded a second-order coefficient of -7.85×10^{-10} rad s^2, in agreement with the expected value of $-2\pi\,(\tau_\mathrm{p}/2\Delta F)$.

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 in each increment. The resulting dataset was a series of FIDs, each comprising a single H_2O signal. When summed, these produce an FID with numerous signals at frequencies defined by the selected transmitter frequencies. The resulting spectrum is presented in Figure 3.10.a.

To produce the phased spectrum in Figure 3.10.b, only the second-order phase was automatically corrected; afterwards, manual first-order phase correction was applied to the spectrum. This was done because, as stated above, the length of the spectrometer delay τ_del is not trivial to determine. As with the simulated case, severe baseline distortions exist in the spectrum, making it unsuitable for quantitative applications. Analogously, to generate the spectrum via back-propagation, only the contribution to τ_0 which is dependent on the signal frequency was included, i.e. τ_0 in Equation 3.29 was replaced with

$$\tau_0' = -\frac{(f - f_\mathrm{off})\tau_\mathrm{p}}{2\Delta F}. \tag{3.30}$$

First-order phase correction was then applied to the spectrum produced from the back-propagated FID to yield the result presented in Figure 3.10.c. In comparison with the quadratic phase-correction approach, it is evident that a far cleaner spectral baseline is achieved. The phases are not perfectly consistent across all signals; most notably, the highest- and second-lowest-frequency signals have phases which stray noticeably from 0°; it is likely that this resulted from error associated with the MPM. Regardless, it is undeniable that the spectrum produced by back-propagation achieves a far more desirable result relative to the quadratic phasing approach.

[¶¶] The paramagnetic species Gd^{III} is a popular contrast agent used in magnetic resonance imaging (MRI), owing to its ability to decrease the T_1 of nearby spins. This typically has an influence on T_2 too, with it being shortened at high enough concentrations. The signal from H_2O therefore decays at a more rapid rate, such that the baseline distortions observed in the phased spectrum of Figure 3.10.b are more pronounced.

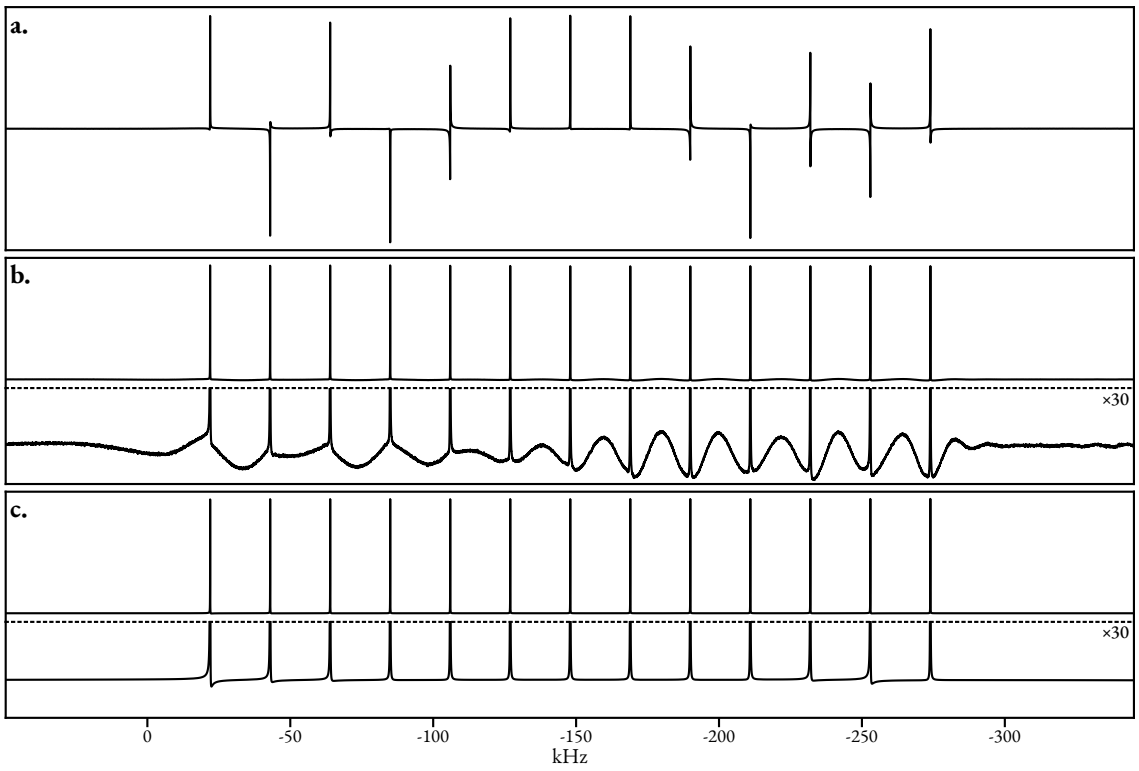


Figure 3.10: A comparison of quadratic phase correction vs frequency-dependent back-propagation in treating experimental single-chirp excitation data generated from a sample of 1% Gd-doped H_2O in D_2O. **a.** Spectrum generated directly from the acquired FID. **b.** Spectrum generated using quadratic phase correction. **c.** Spectrum generated using the back-propagation approach.

3.4 Summary

In this chapter, numerous results have been presented, all of which feature 1D FID estimation. In Section 3.1, it was illustrated that extending the MPM, with the added step of a phase variance-regularised NLP routine, can aid in the generation of parameter estimates which can agree better with the underlying assumptions of the data structure. Oscillators in the MPM often possess spurious phases; this feature can frequently be rectified by the NLP method. Beyond simply providing a description of the signals which make up a dataset, it has been shown that parametric estimation can be applied to further useful ends; two such examples have been presented in Sections 3.2 and 3.3, respectively.

There are scenarios in which 1D estimation can perform admirably, enhancing the information accessible to a spectroscopist. However, 1D datasets are typically the most challenging form of NMR data to estimate due to their inherent complexity; with all information about a sample being distilled into a single data dimension, it is very common (particularly with ^{1}H datasets) for FIDs to be so densely populated with signals that extracting meaningful information at the per-signal level is futile. It should therefore be appreciated that estimation methods such as the one presented in this work — requiring little to no prior knowledge about the dataset to operate — can only be applied effectively to a limited range of FIDs.

Through the separation of signals into more than one detection dimension, multidimensional experiments can yield datasets which have sufficiently well-resolved signals for estimation to be applicable for a wider range of samples. The 2DJ experiment is such an example, and the estimation of datasets acquired using it is the focus of the next chapter.

Chapter 4

2DJ Estimation for Pure Shift NMR

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 [147] (sensitivity $\propto B_0^{3/2}$) and cryogenic probes [33], as well as simply increasing the number of scans (sensitivity $\propto \mathrm{NS}^{1/2}$). However, few means of achieving better resolution exist beyond increased field strengths (resolution $\propto B_0$), and ensuing that a homogeneous magnetic field is used through the use of shimming. 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 gleaned. While it is commonplace to decouple heteronuclear couplings at the point of FID acquisition [148, 149, 150], homonuclear decoupling is far more challenging. At the time of writing, there are a number of approaches for acquiring pure shift spectra; the most popular modern approach involves running a suitable 2D pulse sequence, and concatenating the initial sections of each FID, in a process referred to as "chunking" [151, 152, 153]. 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 to that which is incorporated into the dataset.

In this chapter, a method for deriving pure shift spectra indirectly via the estimation of 2DJ datasets is presented, which has been 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 with desirable absorption-mode lineshapes can be produced without the signal loss associated with experimental pure shift methods.

4.1 Pure Shift NMR

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

4.1.1 The 2DJ Experiment

The 2DJ experiment [45, 46] provided the first means to achieve broadband homodecoupling. It has the following simple pulse sequence:

$$90° \longrightarrow {}^{t^{(1)}}\!/_{2} \longrightarrow 180° \longrightarrow {}^{t^{(1)}}\!/_{2} \longrightarrow t^{(2)}.$$

After excitation of magnetisation onto the transverse plane, the indirect-dimension evolution consists of a spin echo, with acquisition following immediately afterwards. The result is an FID in which only scalar couplings evolve in $t^{(1)}$, as the chemical shifts are refocussed by the spin echo, while both chemical shifts and scalar couplings evolve during $t^{(2)}$. An FID generated by the 2DJ experiment is hypercomplex, taking the form of Equation 1.24 with $D = 2$ and $\zeta^{(1)} = \exp(\mathrm{i}\cdot)$, i.e.

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

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 from a spin system with S distinct spins. Under the assumption of the *weak coupling approximation* [23: Section 2.5.2], the signals which arise from a particular spin $s \in \{1, \cdots, S\}$ form a grouping $G_s \subset \{1, \cdots, M\}$ (note that $M \equiv \sum_{s=1}^{S} |G_s|$). All the signals in G_s will have frequencies given by

$$f_m^{(1)} = d_m, \tag{4.2a}$$

$$f_m^{(2)} = f_{0,s} + d_m, \tag{4.2b}$$

$$\forall m \in G_s,$$

where $f_{0,s}$ is the Larmor frequency of the spin, and d_m is the displacement of the signal from $f_{0,s}$, as a result of J-couplings*. Due to this relationship, all peaks in the Fourier domain which are part of the same multiplet will lie along a line which makes 45° angles to both the $F^{(1)}$ and $F^{(2)}$ axes, as depicted in Figure 4.2.a.

*d_m will be a linear combination of all the J-couplings associated with the spin, with all the coefficients being $\pm 1/2$.

One limitation of the 2DJ experiment is the fact that peaks comprising pure absorption Lorentzian character cannot be produced. This is since, due to the absence of a mixing period, it is not possible to acquire a complementary pair of phase-modulated FIDs; such a pair could be processed so as to nullify the dispersive contributions (see Section 1.2.2). The FT of a 2DJ FID produces a spectrum with phase-twist peaks; an example is provided in Figure 4.2.a. As with other experiments which produce hypercomplex signals, such as COSY, the data is conventionally displayed in *magnitude-mode* (Figure 4.2.b) in which the absolute value of each complex point in the spectrum is plotted.

When present, another disadvantageous trait of 2DJ datasets are *strong coupling artefacts*, which arise due to coherence transfers induced by the 180° pulse [154, 155]. As stressed in [155], 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. An illustration of the influence of strong coupling effects is provided for a system of two coupled spins AB in Figure 4.1. A rigorous description of the theory behind a 2DJ experiment for an AB system is given in [155]. As well as four *first-order* signals (those that are be accounted for when the weak-coupling approximation is invoked) four additional signals appear in the dataset due to strong coupling effects. The frequencies of and relative amplitudes of the eight signals are provided in Table 4.1.

a	$f^{(1)}$	$f^{(2)}$
First order signals		
1	$\frac{J_{AB}}{2}$	$f_A + \frac{J_{AB}}{2}$
1	$-\frac{J_{AB}}{2}$	$f_A - \frac{J_{AB}}{2}$
1	$\frac{J_{AB}}{2}$	$f_B + \frac{J_{AB}}{2}$
1	$-\frac{J_{AB}}{2}$	$f_B - \frac{J_{AB}}{2}$
Strong coupling artefacts		
ρ_{AB}	$\frac{J_{AB}}{2} + \xi_{AB}$	$f_A + \frac{J_{AB}}{2}$
ρ_{AB}	$-\frac{J_{AB}}{2} + \xi_{AB}$	$f_A - \frac{J_{AB}}{2}$
ρ_{AB}	$\frac{J_{AB}}{2} - \xi_{AB}$	$f_B + \frac{J_{AB}}{2}$
ρ_{AB}	$-\frac{J_{AB}}{2} - \xi_{AB}$	$f_B - \frac{J_{AB}}{2}$

Table 4.1: The frequencies and relative amplitudes of signals which feature in the 2DJ spectrum of an AB spin system. The quantities ξ_{AB} and ρ_{AB} are defined in Equations 4.3 and 4.4, respectively.

Each of the strong coupling artefacts has a direct-dimension frequency that matches that of one of the first-order signals. However, each pair of signals that abide by this are displaced in the indirect dimension by ξ_{AB}, give by

$$\xi_{AB} = \frac{1}{2}\sqrt{(f_A - f_B)^2 + J_{AB}^2}.. \tag{4.3}$$

The magnitude of the displacement is such that strong coupling artefacts are frequently aliased in the indirect dimension, which tends to have a small spectral width ($\approx$ 50 Hz). Due to their displacement in the indirect dimension, strong coupling artefacts lie along different 45° cross sections to the first-order signals they are associated with. The intensities of the strong coupling artefacts relative to the first-order signals is given by:

$$\rho_{AB} = \tan^{-1}\left(\frac{J_{AB}}{f_A - f_B}\right). \tag{4.4}$$

From this expression, it can be seen that as the difference in chemical shift increases, the magnitude of the strong coupling artefacts diminishes, until their effect is negligible, such that the spin system may be deemed as weakly coupled.

The Shear and Projection Approach

A pure shift spectrum can be acquired from a 2DJ spectrum by performing a 45° shear on the spectrum array; each direct-dimension vector in the spectrum is subjected to a right circular rotation according to

$$s^{\text{shear}}_{n^{(1)},n^{(2)}} = s_{n^{(1)},n^{(2)\prime}}, \tag{4.5a}$$

$$n^{(2)\prime} = \left(n^{(2)} + \left\lfloor \frac{f^{(1)}_{\text{sw}} N^{(2)}}{f^{(2)}_{\text{sw}} N^{(1)}} \left(\frac{N^{(1)}}{2} - n^{(1)}\right)\right\rceil\right) \bmod N^{(2)}. \tag{4.5b}$$

In the limit of continuous sampling, this achieves the relationship

$$s^{\text{shear}}\left(F^{(1)}, F^{(2)}\right) \equiv s(F^{(1)}, F^{(2)} - F^{(1)}).$$

By performing a 45° shear, chemical shifts and scalar couplings are separated onto orthogonal axes, such that the peaks of the sheared spectrum which arise from a given spin all reside at the same value of $F^{(2)}$ (Figure 4.2.c). The effectiveness of the shear is maximised when both $f^{(2)}_{\text{sw}}/f^{(1)}_{\text{sw}}$ and $N^{(2)}/N^{(1)}$ are powers of 2. After shearing, projecting the spectrum onto $F^{(2)}$ (i.e. summing along the indirect dimension) leads to the 1D pure shift spectrum. If the spectrum wasn't converted to magnitude-mode prior to shearing and projecting, the absorptive and dispersive components of the spectrum would cancel each other out, such that a vector devoid of any signal would be obtained (Figure 4.2.a). With a magnitude-mode spectrum, the process leads to undesirable pure shift spectra with broad "wings" on account of the presence of dispersive character, as well as non-linearities. These effects can be suppressed by appropriate processing to make the FID envelope symmetric in both dimensions; sine-bell apodisation and pseudo-echo reshaping [156] are common methods. However, these both cause a significant reduction in sensitivity being incurred, along with

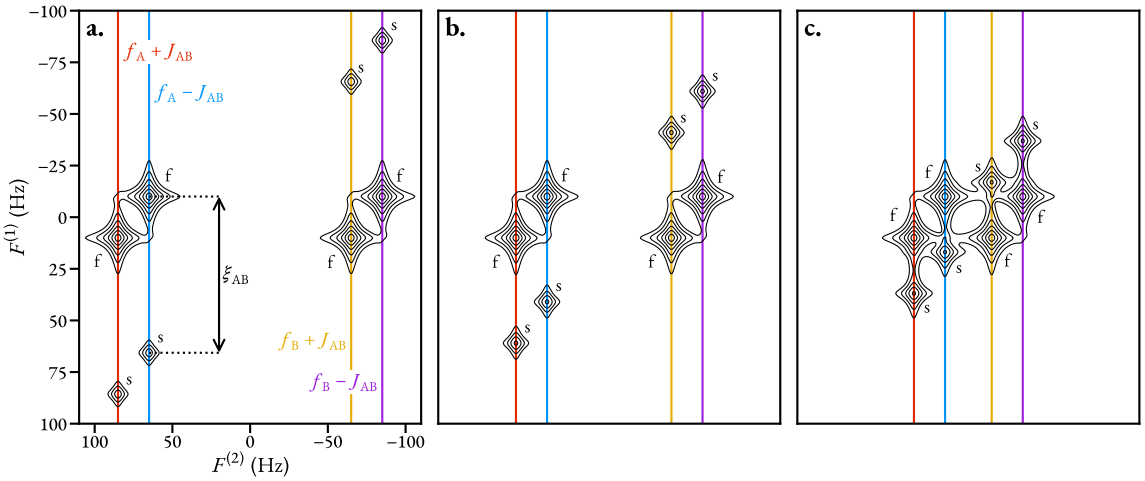


Figure 4.1: The appearance of magnitude-mode spectra 2DJ associated with AB spin systems with varying chemical shift differences. For each spin system, J_{AB} = 20 Hz, and the resonance frequencies of the two spins were set to be $f_A = \Delta f/2$ and $f_B = -\Delta f/2$, with Δf given by 150 Hz (panel a), 100 Hz (panel b), and 50 Hz (panel c). Each spectrum features 8 signals, with their frequencies and relative amplitudes defined in accordance with [155: Table 1]. Along with the 4 first-order signals (labelled "f"), 4 strong coupling artefacts are present in each spectrum (labelled "s"). The artefacts have the following amplitudes relative to the first order signals, in accordance with Equation 4.4: 0.133 (panel a), 0.197 (panel b), 0.381 (panel c). Each of these have a direct-dimension frequency which matches that of a first-order signal; in the indirect dimension the frequencies of each signal pair differ by the quantity ξ_{AB} (Equation 4.3): 75.7 Hz (panel a), 51.0 Hz (panel b), 26.9 Hz (panel c). N.B. the sweep width in the indirect dimension (200 Hz) is considerably larger than typical values used for ^{1}H 2DJ experiments; this as been done to ensure the strong coupling artefacts are not aliased, which occurs commonly in real datasets.

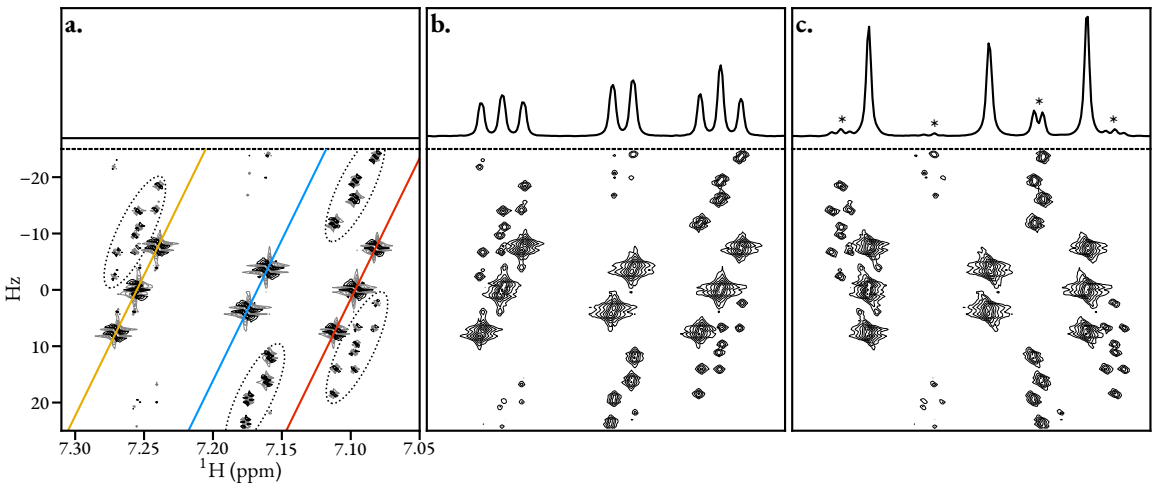


Figure 4.2: A region of a simulated 2DJ spectrum of strychnine. Each panel depicts the spectrum following different processing procedures. Below: contour plots of the spectrum. Black contours have positive values; grey contours have negative values. Above: the projection of the spectrum onto the direct dimension ($F^{(2)}$). **a.** Spectrum produced by applying sine-bell apodisation followed by FT in both dimensions. Coloured lines denote 45° cross-sections along which first-order multiplet structures lie. Prominent strong coupling artefacts have been annotated with dotted ellipses. **b.** Magnitude-mode spectrum. **c.** Spectrum generated after application of a 45° shear on the magnitude-mode spectrum. Peaks marked with an asterisk originate from strong coupling artefacts in the 2DJ spectrum.

distortions in relative peak amplitudes such that the data are rendered unsuitable for quantitative applications. Examples of experimental pure shift spectra produced via sine-bell apodisation can be found in Figures 4.8.a and 4.9.a, where severe discrepancies in peak integrals exist.

When strong coupling artefacts feature in the 2DJ data, spectra produced by shearing and projecting onto $F^{(2)}$ will feature additional low-intensity nuisance signals — recall from above that these tend to lie along different 45° cross-sections to first-order signals in the 2DJ spectrum — that compound the task of interpreting the spectrum (see peaks marked with asterisks in Figures 4.2.c, 4.8.a, and 4.9.a.

Pure Shift Spectra from 2DJ Estimation

Beyond the shearing and projection method, 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) [157, 158], 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:

$$\boldsymbol{\theta}_{n^{(2)}} = \begin{bmatrix} \boldsymbol{a}^{\mathrm{T}}_{n^{(2)}} & \boldsymbol{\phi}^{\mathrm{T}}_{n^{(2)}} & \boldsymbol{f}^{\mathrm{T}}_{n^{(2)}} & \boldsymbol{\eta}^{\mathrm{T}}_{n^{(2)}} \end{bmatrix}^{\mathrm{T}}. \tag{4.6}$$

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

$$\begin{aligned} y^{\mathrm{full}}_{n^{(1)},n^{(2)}} = \sum_{m=1}^{M} a_{n^{(2)},m} \exp\left(\mathrm{i}\phi_{n^{(2)},m}\right) \exp\left(\left(2\pi \mathrm{i} f_{n^{(2)},m} n^{(1)} - \eta_{n^{(2)},m} \left|n^{(1)}\right|\right) \Delta_t^{(1)}\right), \\ \forall n^{(1)} \in \{-N^{(1)}+1, \cdots, 0, \cdots, N^{(1)}-1\},\ \forall n^{(2)} \{0, \cdots, N^{(2)}-1\}. \end{aligned} \tag{4.7}$$

FT of Equation 4.7 generates a spectrum whose real component comprises absorption 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 $-t^{(2)}$ [159]. Further to these LP based methods, Mandelshtam and co-workers have also illustrated how the filter diagonalisation method (FDM) can be applied for parametric estimation, followed by pure shift spectrum construction [160, 161].

4.1.2 Chunking Methods

A popular set of experimental techniques for pure shift NMR comprises 2D pulse sequences in which a short initial section (chunk) of the FID at each increment is retained [152]. The acquired chunks are subsequently concatenated to yield a 1D pure shift signal. All of the sequences are

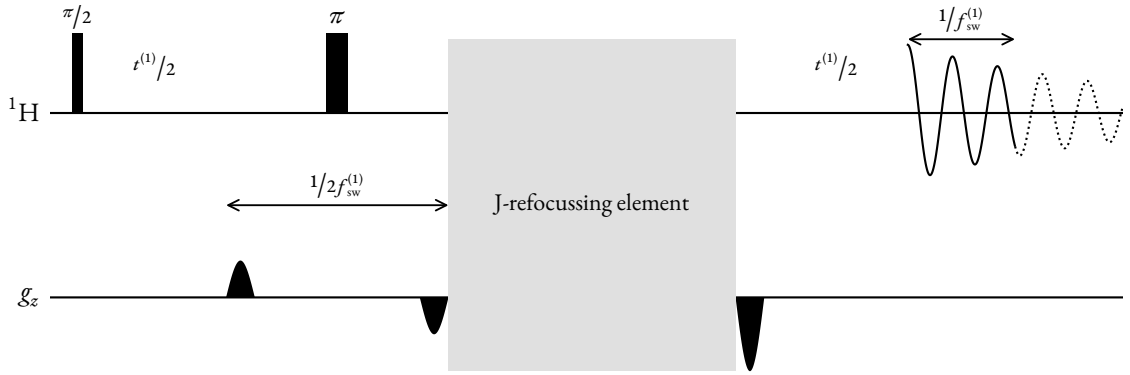


Figure 4.3: The general form of a pulse sequence used for pure shift NMR via the chunking approach. Appropriate examples of J-refocussing elements include BIRD, the Zangger-Sterk element, and the PSYCHE element. The initial chunk of the FID (solid line) is retained and concatenated with other acquired chunks, while the rest (dotted line) is discarded.

designed such that they refocus the effects J-couplings at a certain known point in time; the time at which refocussing occurs is adjusted by incrementing the indirect-dimension ($t^{(1)}$), so as to acquire different chunks of the pure shift FID. It is impossible to refocus the couplings associated with all spins in a sample simultaneously. Instead, the chunking methods operate by ensuring that J-refocussing is achieved for a subset of the spins (the *active spins*) while the rest (the *passive spins*) are manipulated to ensure J-refocussing of the active spins occurs; these do not contribute to the final FID.

A generalised pulse sequence for the chunking approach is depicted in Figure 4.3. In the middle of the sequence is a 180° hard pulse, which inverts all spins. Following this is an element which is designed to invert the active spins again, while leaving the passive spin coherences unaffected. As such, the active spins experience no net effect (they undergo a 360° rotation) while the passive spins experience a net inversion. This results in the J-couplings associated with the active spins being refocussed at the point that acquisition starts[†]. Rather than acquiring a plurality of points, one might envisage that only a single point should be acquired at each increment to ensure that the effects of J-couplings are completely negated from the concatenated signal. However, since the rate of J-coupling evolution is slow (on the order of Hz), it is far more efficient to acquire a chunk of points at each iteration for little loss of data quality.

The primary drawback of the chunking approach is that regardless of the J-refocussing element used, a significant reduction is sensitivity is incurred, since only a subset of the magnetisation present in the sample contributes to the detected signal. However, since they are able to produce clean spectra with absorption lineshapes, these methods have largely superseded the original 2DJ

[†]In practice, the pulse sequence is usually set up to ensure that the exact point of J-refocussing occurs in the middle of each acquired chunk, rather that at the start of the chunk. This enables double the amount of points to be acquired per increment without noticeable influence from J-couplings.

shear-and-project approach. A summary of some of the most prominent J-refocussing elements is now provided.

Zangger-Sterk (ZS)

The ZS element employs the concept of "slice-selective excitation" to distinguish between active and passive spins [162, 163]. It consists of the simultaneous application of a low RF power 180° pulse — conventionally, a r-SNOB pulse is used [164] — and a weak PFG along the laboratory z-axis. The PFG induces a change in a given spin's Larmor frequency as a function of its position in the sample, as was described in Section 3.2.2. For a low-power RF pulse to excite a given spin, its frequency must be at or very close to its Larmor frequency. For spins with different resonance frequencies, the sample region where excitation occurs differs since their Larmor frequencies must be perturbed by different extents to match the pulse frequency. The net effect of the ZS element is to invert a particular spin in only a narrow range of heights in the sample; the spins which are inverted belong to the pool of active spins, while the remainder are the passive spins. In order to achieve effective decoupling of a given pair of spins, it is required that the bandwidth of the selective 180° pulse is smaller than the difference in their Larmor frequencies. However, with more selective pulses, a smaller proportion of the spins will be in the active subset, and hence the FID's sensitivity will be diminished[‡]. Therefore a trade-off exists between effective decoupling of all spins, and achieving good sensitivity. When strong coupling is present, the ZS element tends to perform poorly relative to other options for this reason. It possible to incorporate the ZS element into the 2DJ pulse sequence, enabling the generation of 2DJ datasets comprising phase-modulated pairs [165]. With this approach, pure shift spectra with far more desirable lineshapes can be generated via the 45° shear-and-project approach relative to using a magnitude-mode spectrum.

Bilinear Rotation Decoupling (BIRD)

The BIRD element acts to invert spins bound to a low-abundance heteronuclear isotope, while the remaining spins bound to a high-abundance isotope are unaffected [166, 167]. The two most common heteronuclei exploited when using BIRD are ^{13}C (1.1% abundance) and ^{15}N (0.37% abundance). The loss of sensitivity is therefore known and constant across samples. In scenarios where strong coupling exists, BIRD can achieve improved sensitivity over ZS, due to the requirement for a highly selective 180° in the ZS element as discussed above. 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 [168]. One of BIRD's primary

[‡]The sensitivity reduction suffered using the ZS element is $\propto {}^{\Delta F}/_{\gamma g l_z}$, where ΔF is the selective pulse bandwidth, g is the strength of the PFG, and l_z is the length of the sample lying within the receiver coil (≈ 1.5 cm).

drawbacks is the fact that geminal protons cannot be decoupled from each other, since such protons are always in the same active/passive subset. Doublets rather than singlets will arise in such cases.

Pure Shift Yielded by Chirp Excitation (PSYCHE)

The most recent major development in pure shift spectroscopy is the PSYCHE experiment [169, 170], which has been hailed for its ability to generate clean pure shift spectra with better sensitivity relative to alternative methods. PSYCHE is inspired by the anti z-COSY experiment [171], and involves the application of two low flip-angle ($\beta \ll 90°$) pulses to achieve active/passive subset creation via coherence selection. The two low flip-angle pulses are usually either chirp pulses, with the sweep direction of the second being the opposite of the first, or *saltire* pulses, which are a superposition of low-to-high and high-to-low chirp pulses. When applied in the presence of a PFG, it has been illustrated that this pair of pulses is very effective at achieving J-refocussing of the active set of spins, while simultaneously refocussing other undesired (i.e. non-single quantum) coherences. The proportions of active and passive spins are dependent on the flip angle of the chirp pulses; these are $\sin^2 \beta$ and $\cos^2 \beta$, respectively. With larger flip angles, more signal from unwanted coherences is present in the final dataset. As with the ZS element, this means there is a compromise that needs to be addressed in order to produce pure shift spectra that have both acceptable sensitivity and that feature minimal artefacts; as a rule of thumb, optimal results are typically achieved when $\beta \approx 20°$.

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 [172, 173]. Involving a three-dimensional (3D) pulse sequence, the PSYCHE-2DJ experiment requires long experiment times (typically tens of hours) in order to produce a spectrum with well-resolved multiplet structures in the indirect dimension.

4.2 Methodology

CUPID, a new method developed for pure shift NMR, fits into the category of techniques which utilise 2DJ estimation. In this section, a description of it is given.

4.2.1 The Estimation Routine

Recall the assumption that a 2DJ FID takes the functional form of Equation 4.1. The primary steps involved in estimating a 2DJ dataset, for a given spectral region of interest, are:

1. Generation of a frequency-filtered sub-FID corresponding to the region of interest (*vide in-*

fra: Section 4.2.3).

2. Prediction of the sub-FID's model order, either by applying the MDL on the first increment in the direct dimension[§](Section 2.2.3) or by manually specifying a value.
3. Generation of an initial parameter estimate using the MMEMPM (Section 2.2.2).
4. Subjection of the initial parameter estimate to phase variance-regularised NLP (Section 2.3).

Instead of estimating successive 1D FIDs, as proposed by Nuzillard and Mutzenhardt *et al.*, 2DJ sub-FIDs are considered holistically; a number of benefits are realised because of this. First, multiplet structures which heavily overlap in a conventional 1D dataset can become separated in the 2DJ dataset, assuming that the Larmor frequencies of the relevant spins are sufficiently different. Accurate resolution of the FID's constituent signals 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 additional resolution advantage relative to the estimation of *direct* dimension 1D 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 is not corrected, such that damping occurs at a faster rate, characterised by T_2^* (see Footnote § in Section 3.2). As such, multiplet structures in the indirect dimension exhibit better resolution (assuming $f_{\text{sw}}^{(1)}/N^{(1)}$ and $f_{\text{sw}}^{(2)}/N^{(2)}$ are comparable). A further benefit comes with having access to the frequencies of signals in *both* dimensions, since this opens up the opportunity to group together those which belong to the same multiplet, as will be discussed in Section 4.2.4. Similar information can be obtained by extracting cross-sections of a sheared magnitude-mode 2DJ spectrum at appropriate values of $F^{(2)}$, though the lineshapes of peaks suffer from the undesirable characteristics described above. The ZS-2DJ [165] and PSYCHE-2DJ [172, 173] experiments are also able to generate individual multiplet structures, though with long 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 on a 2D FID is not desirable, since a complete SVD computation would need to be undertaken on the typically very large block-Hankel matrix $\boldsymbol{E}_{\boldsymbol{Y}}$. 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 sometimes realised when the 1D MDL is used for model order selection. The presence of strong coupling artefacts leads to additional nuisance peaks featuring in spectra produced by the shear-and-project approach. Exactly the same phenomenon would occur when estimation is employed, assuming that the strong coupling artefacts are incorporated into the parameter set. However,

[§]The first increment of a 2DJ experiment, for which $t^{(1)} = 0$ s, effectively takes the same form as an FID derived from a pulse-acquire experiment.

since the artefacts have direct-dimension frequencies which are identical to those of first-order signals in the dataset, it is incredibly challenging to resolve these using the 1D MDL approach. Therefore, the MDL is often found to predict 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 expected to be quantified, whereas the weaker strong coupling artefacts will be neglected. It should be noted that this concept is not infallible; it has simply been observed on a number of occasions. There are instances where strong coupling artefacts are incorporated into the estimation result, either because the prediction of M was excessive, or certain artefacts possess particularly large amplitudes; example of these situations will be given later.

4.2.2 The −45° Signal

The 2DJ estimation routine yields the 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 proceeding via the full-echo approach of Nuzillard and Mutzenhardt *et al*. The desired synthetic FID produced is named the *−45° signal* $\boldsymbol{y}_{-45^\circ} \in \mathbb{C}^{N^{(2)}}$:

$$y_{-45^\circ,n^{(2)}} = \sum_{m=1}^{M} a_m \exp(\mathrm{i}\phi_m) \exp\left(\left(2\pi\mathrm{i}\left(f_m^{(2)} - f_m^{(1)} - f_{\text{off}}\right) - \eta_m^{(2)}\right) n^{(2)} \Delta_t^{(2)}\right), \tag{4.8}$$

with the reasoning behind the name provided by Figure 4.4. The −45° signal takes the form of a conventional 1D FID, except that the frequency of each oscillator, which would be $f_m^{(2)}$ in a pulse-acquire experiment, is replaced with $f_m^{(2)} - f_m^{(1)}$. Oscillators belonging to the same multiplet s will all provide a contribution to the FID with the angular 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, the generation of frequency-filtered sub-FIDs is limited to the direct dimension in CUPID. The filtering procedure applied to 2DJ data is an extension of that for 1D data described in Section 2.5:

1. The array $\boldsymbol{Y}_{\text{VE}} \in \mathbb{C}^{N^{(1)} \times 2N^{(2)}}$, is constructed, which contains the VE of each direct-dimension

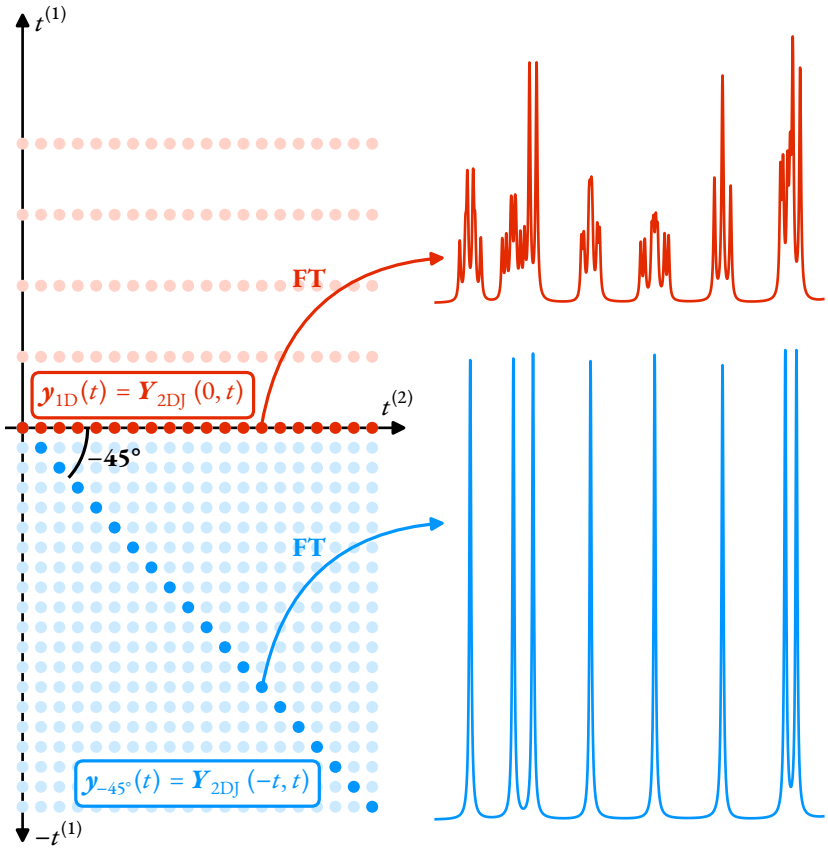


Figure 4.4: An illustration of the reasoning behind the name "−45° signal", which is used to generate pure shift spectra as part of CUPID. The pale red dots denote a typical 2DJ FID, where the amount and rate of sampling in the direct dimension is greater than in the indirect dimension (i.e. $N^{(1)} \ll N^{(2)}$ and $f_{sw}^{(1)} \ll f_{sw}^{(2)}$). The bright red dots correspond to the first direct-dimension signal $y_{2DJ}(0, t^{(2)})$, 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 with the $t^{(2)}$ axis — the convention of angles being defined through an anticlockwise rotation starting on the $t^{(2)}$-axis has been applied here — yields an FID $\boldsymbol{y}_{-45°}$ which is homodecoupled. Note that there is a slight discrepancy between Equation 4.8 and this description, in that the indirect dimension damping factors $\boldsymbol{\eta}^{(1)}$ are neglected in the former case.

FID, i.e. each row of the array is given by

$$\boldsymbol{y}_{\mathrm{VE},n^{(1)}} = \left[\Re\left(y_{n^{(1)},0}\right) \quad y_{n^{(1)},1} \quad \cdots \quad y_{n^{(1)},N^{(2)}-1} \quad 0 \quad y^*_{n^{(1)},N^{(2)}-1} \quad \cdots \quad y^*_{n^{(1)},1}\right], \quad \forall n^{(1)} \in \{0,\cdots,N^{(1)}-1\}. \tag{4.9}$$

2. $\boldsymbol{Y}_{\mathrm{VE}}$ is subjected to FT along the direct dimension to produce the spectrum $\boldsymbol{S}_{\mathrm{VE}}$ (Figure 4.5.a); this has an imaginary component of zeros.
3. A super-Gaussian $\boldsymbol{G} \in \mathbb{R}^{N^{(1)}\times 2N^{(2)}}$ is constructed (Figure 4.5.b):

$$\boldsymbol{G} = \boldsymbol{1} \otimes \boldsymbol{g}^{(2)}, \tag{4.10}$$

where $\boldsymbol{1} \in \mathbb{R}^{N^{(1)}}$ is a vector of ones, and $\boldsymbol{g}^{(2)} \in \mathbb{R}^{2N^{(2)}}$ is a super-Gaussian vector given by Equation 2.57.

4. A matrix of additive noise is generated by extracting the variance σ^2 of a direct-dimension strip of $\boldsymbol{S}_{\mathrm{VE}}$ which is devoid of peaks, and generating an array $\boldsymbol{W}_{\sigma^2} \in \mathbb{R}^{N^{(1)}\times 2N^{(2)}}$ with values independently sampled from $\mathcal{N}(0,\sigma^2)$ (Figure 4.5.c).
5. The spectrum is filtered (Figure 4.5.d):

$$\widetilde{\boldsymbol{S}}_{\mathrm{VE}} = \boldsymbol{S}_{\mathrm{VE}} \odot \boldsymbol{G} + \boldsymbol{W}_{\sigma^2} \odot (\boldsymbol{1} - \boldsymbol{G}). \tag{4.11}$$

6. $\widetilde{\boldsymbol{S}}_{\mathrm{VE}}$ is subjected to IFT and is sliced in half in the direct dimension, yeilding the final filtered signal $\widetilde{\boldsymbol{Y}}$.

As with the 1D case, this method can also be extended to incorporate spectral slicing, acting to reduce the number of points present in the filtered sub-FID.

4.2.4 Multiplet Prediction

CUPID's ability to group oscillators present in a parameter set into multiplet structures relies on simultaneously knowing both the indirect- and direct-dimension frequencies of each model oscillator. As has already been established, for oscillators which are associated with the same multiplet grouping G_s, the quantities $f^{(2)}_{m_1} - f^{(1)}_{m_1}$ and $f^{(2)}_{m_2} - f^{(1)}_{m_2}$ 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:

$$\left|\left(f^{(2)}_{m_1} - f^{(1)}_{m_1}\right) - \left(f^{(2)}_{m_2} - f^{(1)}_{m_2}\right)\right| < \epsilon. \tag{4.12}$$

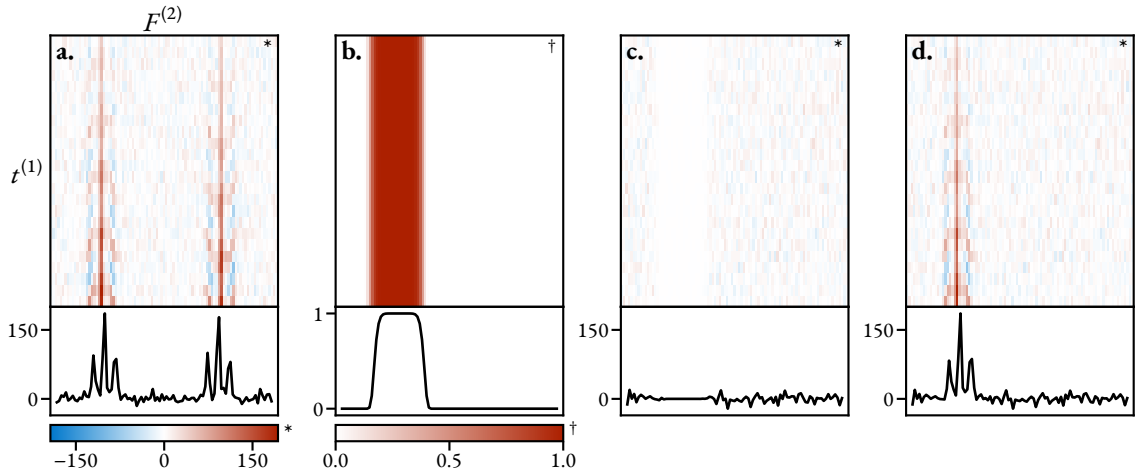


Figure 4.5: An illustration of the filtering procedure for 2DJ data. Each panel consists of a heat-map of the complete 2D array, as well as a plot of the first slice of the array in the direct dimension (x-axis). **a.** The spectrum $\boldsymbol{S}_{\mathrm{VE}}$, **b.** Super-Gaussian filter $\boldsymbol{G}$, **c.** Additive noise, attenuated by the super-Gaussian, $\boldsymbol{W}_{\sigma^2} \odot (\mathbf{1} - \boldsymbol{G})$, **d.** Filtered spectrum $\widetilde{\boldsymbol{S}}_{\mathrm{VE}}$ Figures 4.5.a to 4.5.d are analogous to Figures 4.5.b to 4.5.e for the 1D case.

$\epsilon \in \mathbb{R}_{>0}$ is a threshold to account for uncertainty in the estimation result. A lower bound on ϵ is the separation between adjacent points in the better-resolved dimension of the spectrum, i.e. $\min\left(f_{\mathrm{sw}}^{(1)}/N^{(1)}, f_{\mathrm{sw}}^{(2)}/N^{(2)}\right)$. However, limitations in resolution due to relaxation-induced signal damping and field inhomogeneities can require ϵ to be increased beyond this for effective assignments to be achieved. Code Listing B.8 provides a PYTHON routine that can be used for multiplet prediction.

There are certain circumstances where it is safe to assume that a particular oscillator in the estimation result is not associated with a first-order signal in the dataset; no oscillator which was derived from a first-order signal will abide by both of the following:

1. The oscillator is not grouped with any other oscillator as part of the multiplet assignment.
2. The magnitude of the indirect dimension frequency of the oscillator is appreciably greater than 0 Hz.

If a first-order 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 automatically discarded from the final parameter estimate.

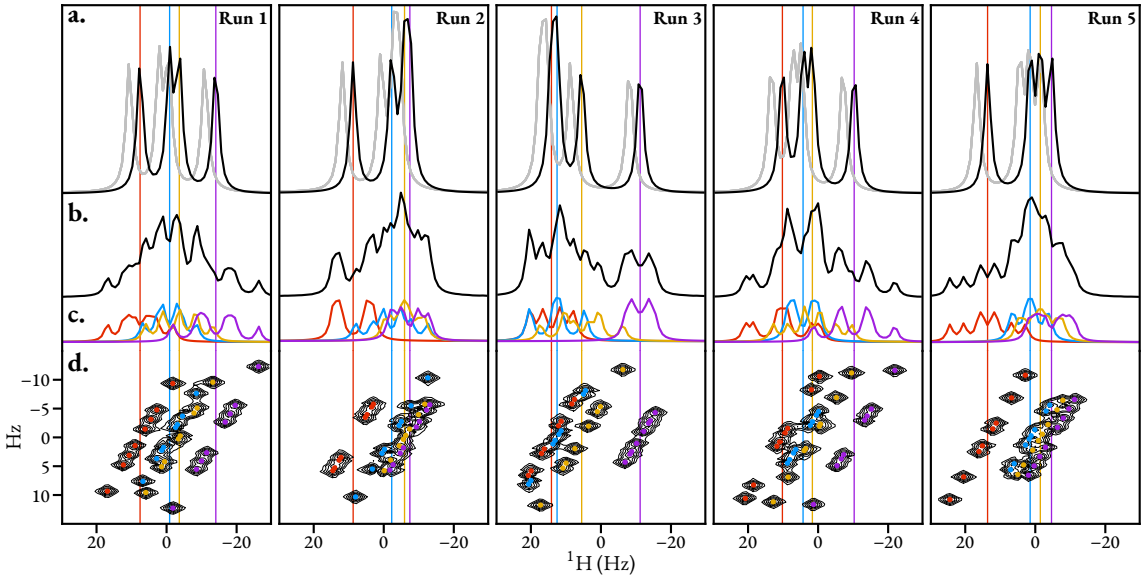


Figure 4.6: The result of applying CUPID on 5 instances of simulated 2DJ datasets with 4 heavily overlapping multiplet structures. **a.** Black: pure shift spectrum generated by CUPID (via the −45° signal). Grey: 1D spectrum simulated with Spinach, using the same spin system as was used to produce the 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 the threshold $\epsilon = f_{sw}^{(2)}/N^{(2)} \approx 0.98$ Hz. **d.** Contour plot of the 2DJ spectrum in magnitude-mode, with coloured points denoting the frequencies of oscillators in the estimation result. The coloured vertical lines denote the predicted central frequencies of each multiplet structure.

4.3 Results

A number of examples of the application of CUPID is now presented. For details relating to generation of the presented datasets, see Appendices C.1, C.2.3 and C.2.4. Useful metrics, such as run times and the number of oscillators present at different stages of the estimation routine are provided in Table C.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, four ddd multiplet structures with significant overlap were present. Noise was added to each FID, with a target SNR of 30 dB. Filtering was applied to the datasets to generate sub-FIDs containing the signals in the region of interest (−30 Hz to 30 Hz). Figure 4.6 provides an illustration of CUPID's performance when applied to the datasets. Each of the sub-FIDs comprised 32 (4×2^3) signals. As can be seen in Figure 4.6.b, the first direct-dimension FIDs are too crowded for reasonable estimates of model order to be made using the MDL, so a value was manually provided; for each dataset, a random integer from the range $[33, 40]$ was selected as the initial number of oscillators. Hence, the initial guess from the MMEMPM comprised a slightly excessive

number of oscillators in each case. For each FID generated, CUPID was able to produce a 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 through their acquisition of negative amplitudes. For 2 of the 5 datasets, the result after NLP comprised 33 oscillators, with a single oscillator being associated with noise remaining. These were automatically detected and removed using the first-order signal detection criteria described in Section 4.2.4.

With simulated examples such as this, it is possible to confirm that the pure shift spectrum generated using CUPID agrees with the expected result; the "true" pure shift spectrum can be obtained by simulating a pulse-acquire experiment, using a spin system with same chemical shifts, but with all J-couplings set to 0 Hz. As seen in Figure 4.6.a. the spectra produced using CUPID agree well with these. Beyond pure shift spectrum generation, the multiplet assignment was also successful; each of the ddd structures were resolved, and these are plotted in Figure 4.6.c.

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. Noise was included with a target SNR of 20 dB. CUPID was applied to filtered sub-FIDs such that the signals arising from all spins were considered, with the result presented in Figure 4.7. 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 FIDs in order to predict their model orders. 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" effect associated with strongly coupled spins (*cf.* Figures 4.7.b and 4.7.c). The clearest examples of this are associated with the pair (U) and (V), as well as the trio (A), (B) and (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, except that scalar couplings are set to 0 Hz.

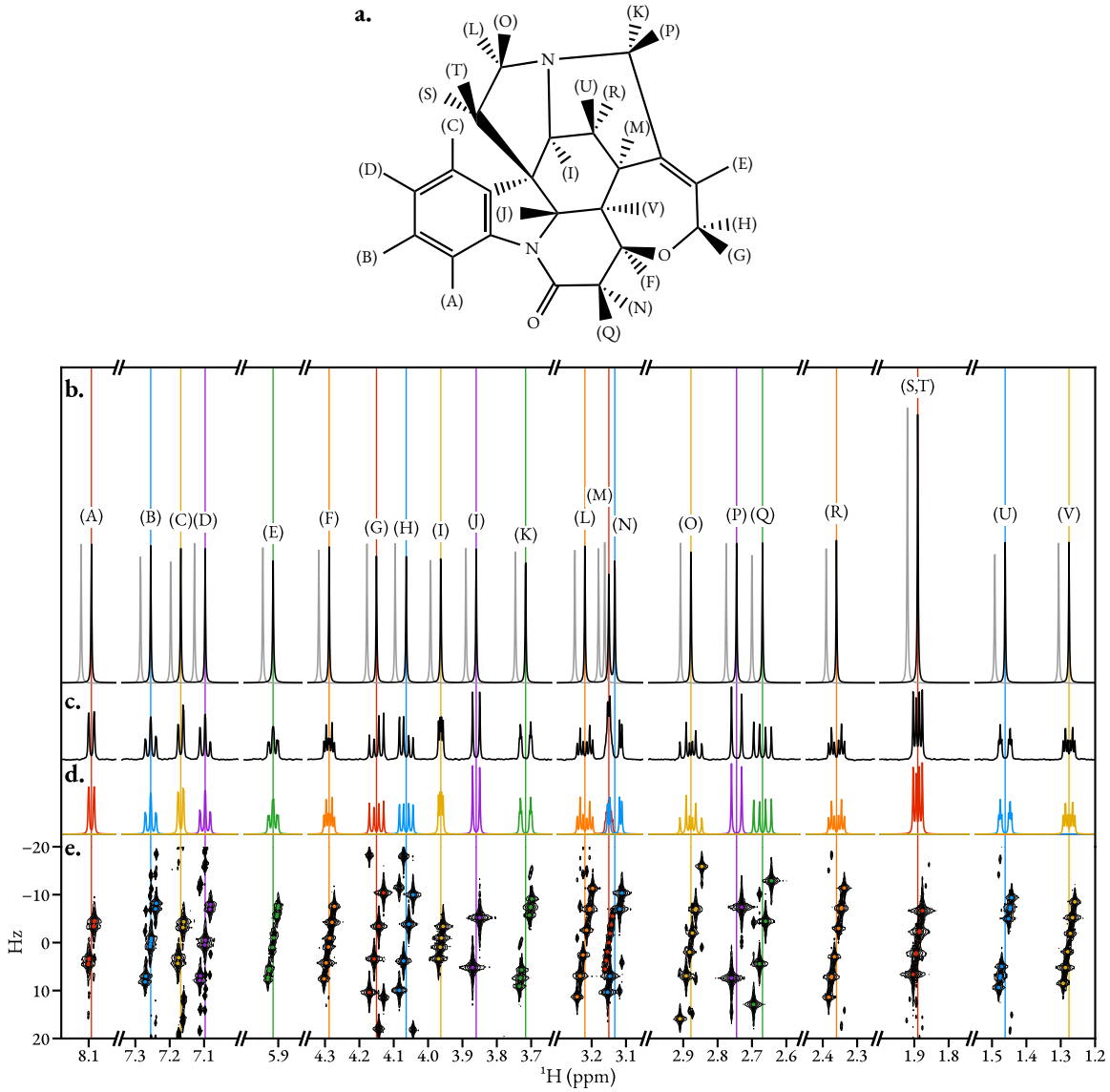


Figure 4.7: The application of CUPID on a simulated strychnine 2DJ dataset. **a.** Structure of strychnine; see Table C.1 for an outline of the chemical shifts and scalar couplings used to generate the dataset. **b.** Black: pure shift spectrum generated by CUPID (via the −45° signal). Grey: 1D spectrum 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. **c.** 1D spectrum of the dataset. **d.** Multiplet structures predicted, using the threshold $\epsilon = f_{\text{sw}}^{(2)}/N^{(2)} \approx 0.39$ Hz. **e.** Contour plot of the 2DJ spectrum in magnitude-mode, with coloured points denoting the frequencies of oscillators in the estimation result.

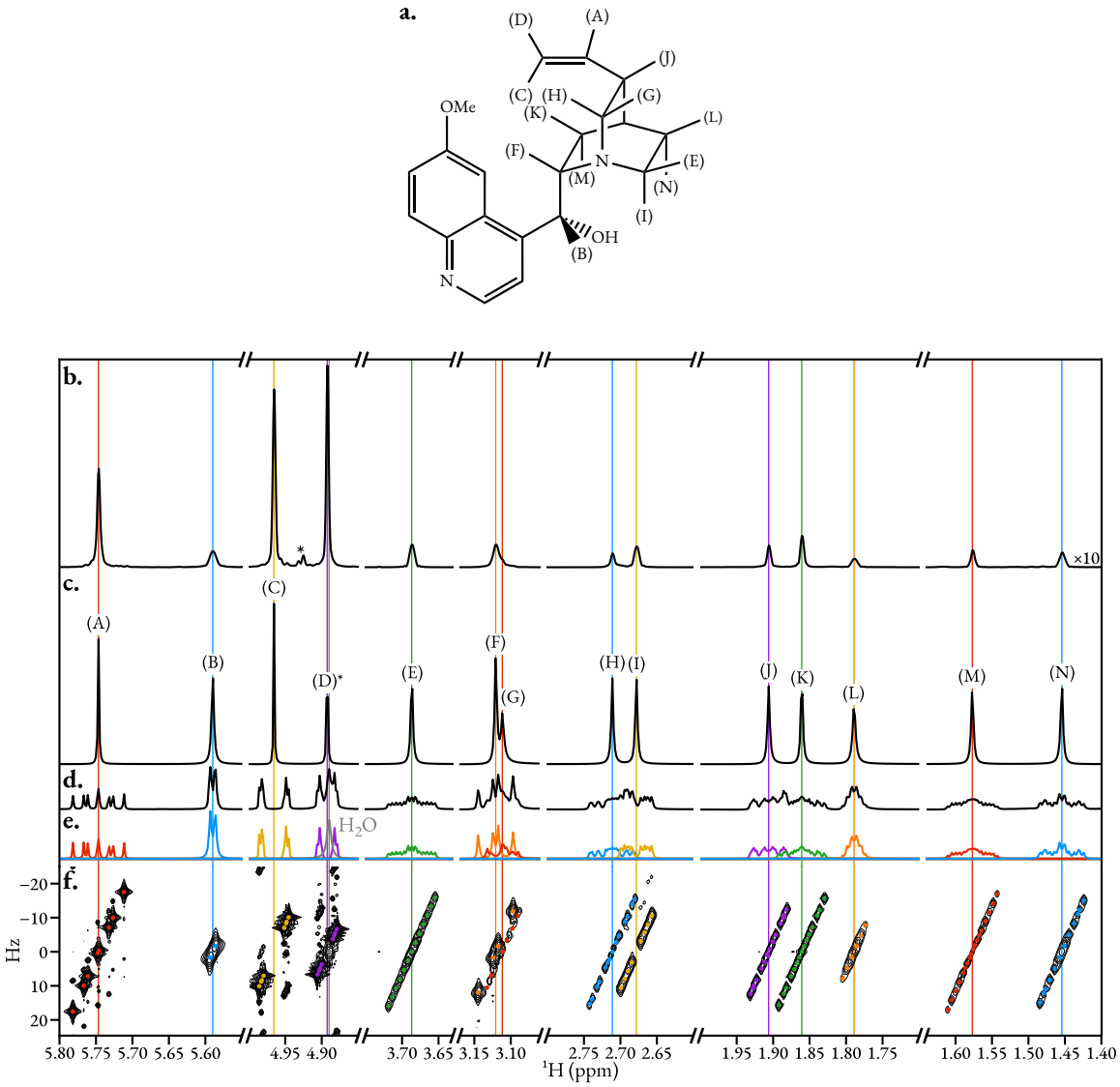


Figure 4.8: The application of CUPID on the non-aromatic regions of a quinine 2DJ dataset. **a.** Structure of quinine. **b.** Spectrum produced using the 45° shear and projection approach. The peaks denoted by an asterisk originate from strong coupling artefacts. **c.** The spectrum generated from FT of the −45° signal, with the signal arising from H_2O (grey, close to 4.9 ppm) neglected. **d.** Spectrum of the first direct-dimension signal in the 2DJ FID. **e.** Multiplet structures assigned ($\epsilon = f_{\mathrm{sw}}^{(2)}/N^{(2)} \approx 0.92\,\mathrm{Hz}$). **f.** Contour plot of the 2DJ spectrum in magnitude-mode, with the locations of assigned oscillators given as coloured points.

4.3.3 Quinine

Figure 4.8 illustrates the result of applying CUPID on a dataset generated from a sample comprising quinine in CD_3OD, with all signals arising from non-aromatic protons considered.

For comparison with CUPID, Figure 4.8.a shows the spectrum produced using the shear-and-project method. Because sine-bell apodisation was applied to the FID in order to suppress dispersive contributions, the relative amplitudes of the pure-shift peaks are drastically different. In particular, the signals from the alkenyl spins (A), (C) and (D) are attenuated less by apodisation relative to the aliphatic signals due to their 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).

CUPID successfully generated a pure shift spectrum with distinct peaks for each ^1H environment, which possess sharper lines and more consistent integrals relative to the shear-and-project spectrum (Figure 4.8.b). Furthermore, the strong coupling artefacts between (C) and (D) were not incorporated into the estimation result, and as such a clean baseline between (C) and (D) was achieved. The multiplet grouping procedure was able to resolve the signals corresponding to spin (D) (purple) from the singlet corresponding to residual water in the sample (grey). To obtain a clean signal for spin (D), without heavy overlap with the water signal, the oscillator corresponding to the water 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 [174]. These methods operate by assuming that the most significant component(s) in the data are derived from the solvent; these are subtracted from the dataset to remove their influence. The process of removing the water signal from the quinine data is slightly different, since it was suppressed manually through inspection of the CUPID result. A knowledgeable user would be able to locate the water signal and discard it. However, 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 certain multiplet structures 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 has compensated by giving said oscillators large amplitudes and damping factors, so that they can reasonably fit multiple similar-frequency signals. In these situations, pure shift peaks generated via the $-45°$ signal will possess augmented linewidths. This behaviour is also exhibited to a lesser ex-

[¶] Spins with longer T_2s produce signals which decay less rapidly. Sine-bell apodisation heavily diminishes the intensities 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.

tent 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 of the doublets, culminating in a slightly broadened pure shift peak.

4.3.4 Camphor

The application of CUPID to the non-methyl signals of a 2DJ dataset of camphor in DMSO-d_6 is presented in Figure 4.9. As with the quinine example, a spectrum generated through the shear-and-project approach 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 Figure 4.9.b results. Again, achieving this requires manual intervention, with the user requiring knowledge about the sample to distinguish between first-order signals and strong coupling artefacts.

4.3.5 Dexamethasone

Figure 4.10 shows the result of applying CUPID on a dataset acquired from a sample of dexamethasone in DMSO-d_6. A pure shift spectrum was also acquired using the triple spin echo PSYCHE (TSE-PSYCHE) experiment [170, 172] for comparison. Overall, excellent agreement is achieved between the PSYCHE spectrum, and that generated using the −45° signal. The estimation routine performed admirably even in cases where heteronuclear coupling to ^{19}F exists. 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. However, the very small though perceptible heteronuclear coupling between ^{19}F and (D) could not be resolved.

In this example, there are a few cases where oscillators which correspond to strong coupling artefacts exist in the final parameter set, even after applying the first-order signal criteria. These persist because either they have an indirect frequency ≈ 0 Hz (these exist close to 1.4 ppm, 1.6 ppm, and

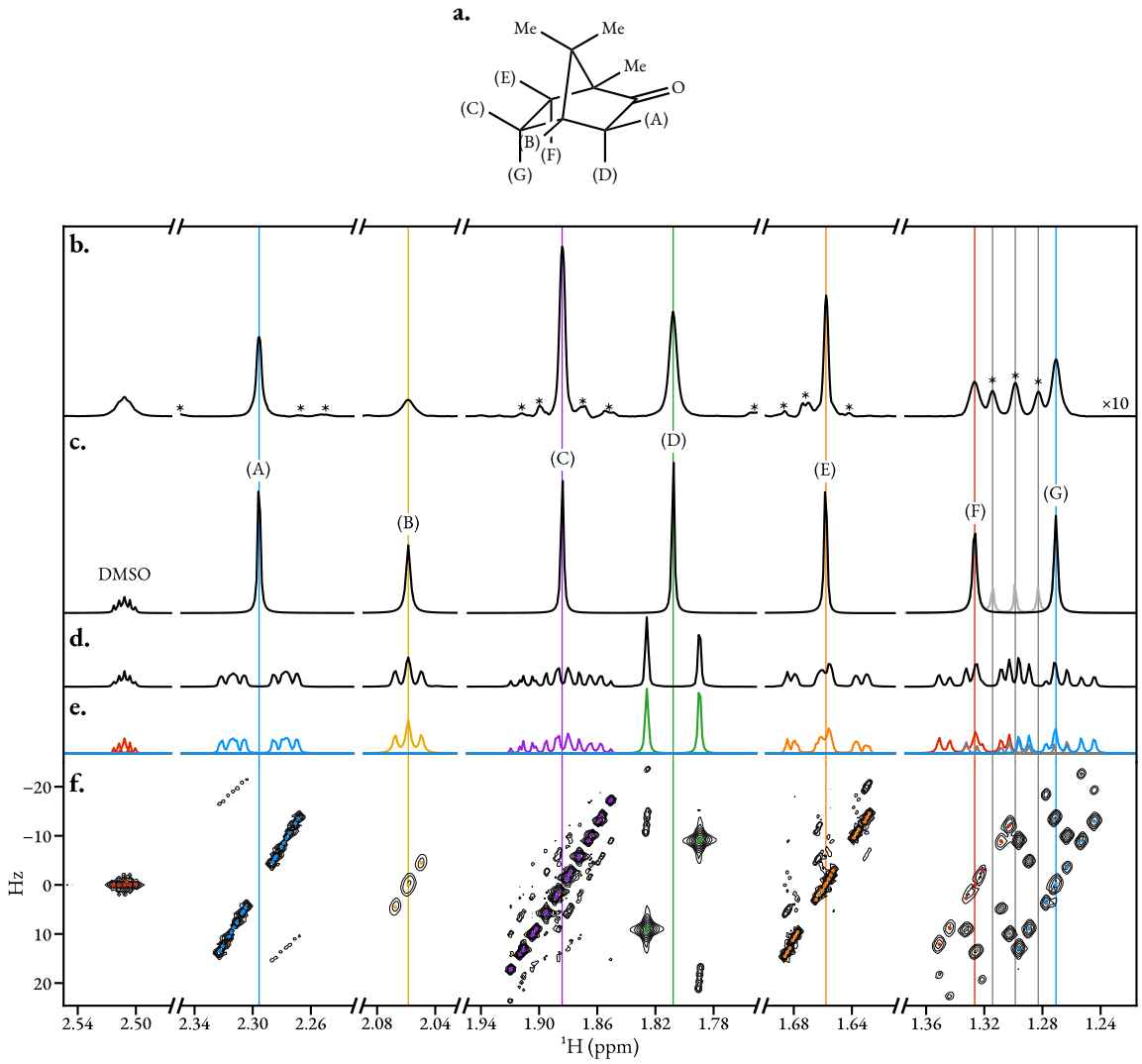


Figure 4.9: The application of CUPID on a camphor 2DJ dataset. **a.** Structure of camphor. **b.** Spectrum produced using the 45° shear and projection approach. The peaks denoted by an asterisk originate from strong coupling artefacts. **c.** Black: spectrum generated from FT of the −45° signal, by neglecting the strong coupling artefacts associated with spins (F) and (G). Grey: equivalent spectrum, but with the strong coupling artefacts incorporated. **d.** Spectrum of the first direct-dimension signal in the 2DJ FID. **e.** Multiplet structures assigned ($\epsilon = 2f_{\mathrm{sw}}^{(2)}/N^{(2)} \approx 1.23$ Hz). **f.** Contour plot of the 2DJ spectrum in magnitude-mode, with the locations of assigned oscillators given as coloured points. Points which are grey do not correspond to the first-order signals associated with camphor; these are present in the parameter estimate through the incorporation of strong coupling artefacts.

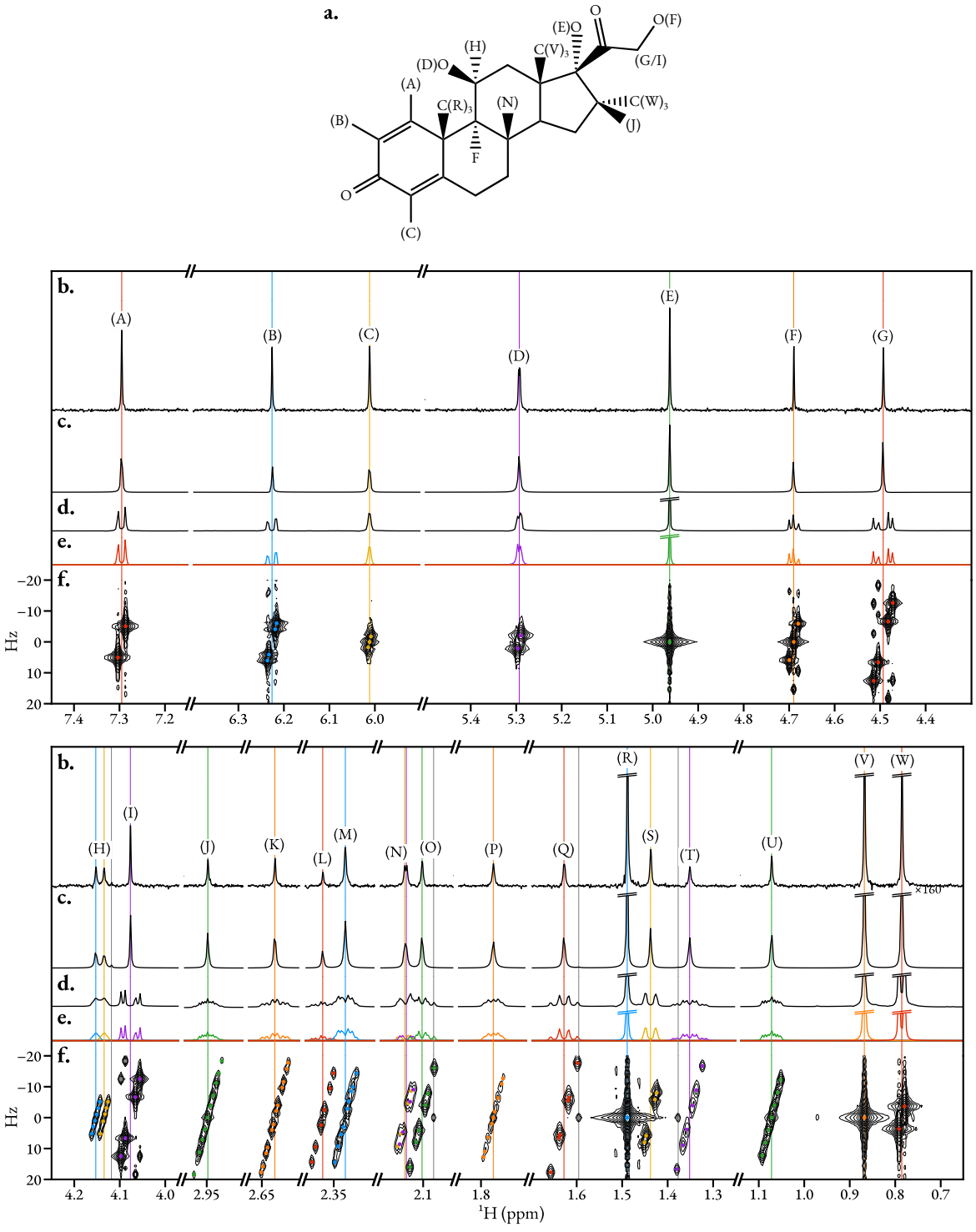


Figure 4.10: The application of CUPID on a 2DJ dataset of dexamethasone in DMSO-d_6. **a.** The structure of dexamethasone. Unlabelled proton environments could not confidently be assigned. **b.** TSE-PSYCHE spectrum. **c.** The spectrum generated from FT of the −45° signal. **d.** Conventional 1D spectrum. **e.** Multiplet structures assigned ($\epsilon = f_{\text{sw}}^{(2)}/N^{(2)} \approx 0.92$ Hz). **f.** Magnitude-mode 2DJ spectrum, with the locations of assigned oscillators given as coloured points. Points which are grey do not correspond to the first-order signals associated with dexamethasone.

2.1 ppm), or because at least two parameterised artefacts are grouped together as part of the multiplet assignment (this occurs close to 4.1 ppm). 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 to illustrate their effect on the final pure shift spectrum.

4.3.6 Estradiol

A final illustration of CUPID's performance is provided by Figure 4.11, where a low concentration (2 mM) sample of 17β-estradiol in DMSO-d_6 is considered. This is the most challenging example presented due to (a) the low SNR of the data, and (b) the presence of incredibly complex regions for a small molecule spectrum, featuring many overlapping multiplet structures. In fact, because of the considerable complexity of the estradiol 1D spectrum, the molecule was used to showcase the PSYCHE experiment in the original work describing it [169].

The PSYCHE experiment produced a spectrum with very poor SNR, such that it is barely sensitive enough for pure shift peaks to be distinguished from the noise. However due to the innately higher sensitivity associated with the 2DJ experiment it was still possible to generate a reasonable parameter estimate of the 2DJ dataset, and subsequently to yield an agreeable pure shift spectrum via the −45° signal. Due to the severe signal overlap in the first direct-dimension FID, model order estimates had to be manually specified. It is clear from inspection of the most downfield region that certain strong coupling artefacts have been incorporated into the estimation result, manifesting in the appearance of artefacts in the pure shift spectrum (see the positions marked by the grey vertical lines in Figure 4.11.b). The time elapsed to run the estimation routine was rather long, at 10.25 min for all regions considered (see Table C.7 for a more detailed run-down of timings). This is largely attributable to the high model orders required in estimating the spectral regions considered, particularly for 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 construction of pure shift spectra via the holistic estimation of 2DJ datasets, is presented. Such spectra possess myriad beneficial features relative to alternative methods, albeit with the requirement of a complex post-processing procedure.

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. CUPID is able to perform admirably even when state of the art techniques like PSYCHE produce spectra with such low SNRs as to render them unusable. Frequently, CUPID is able to automatically discard

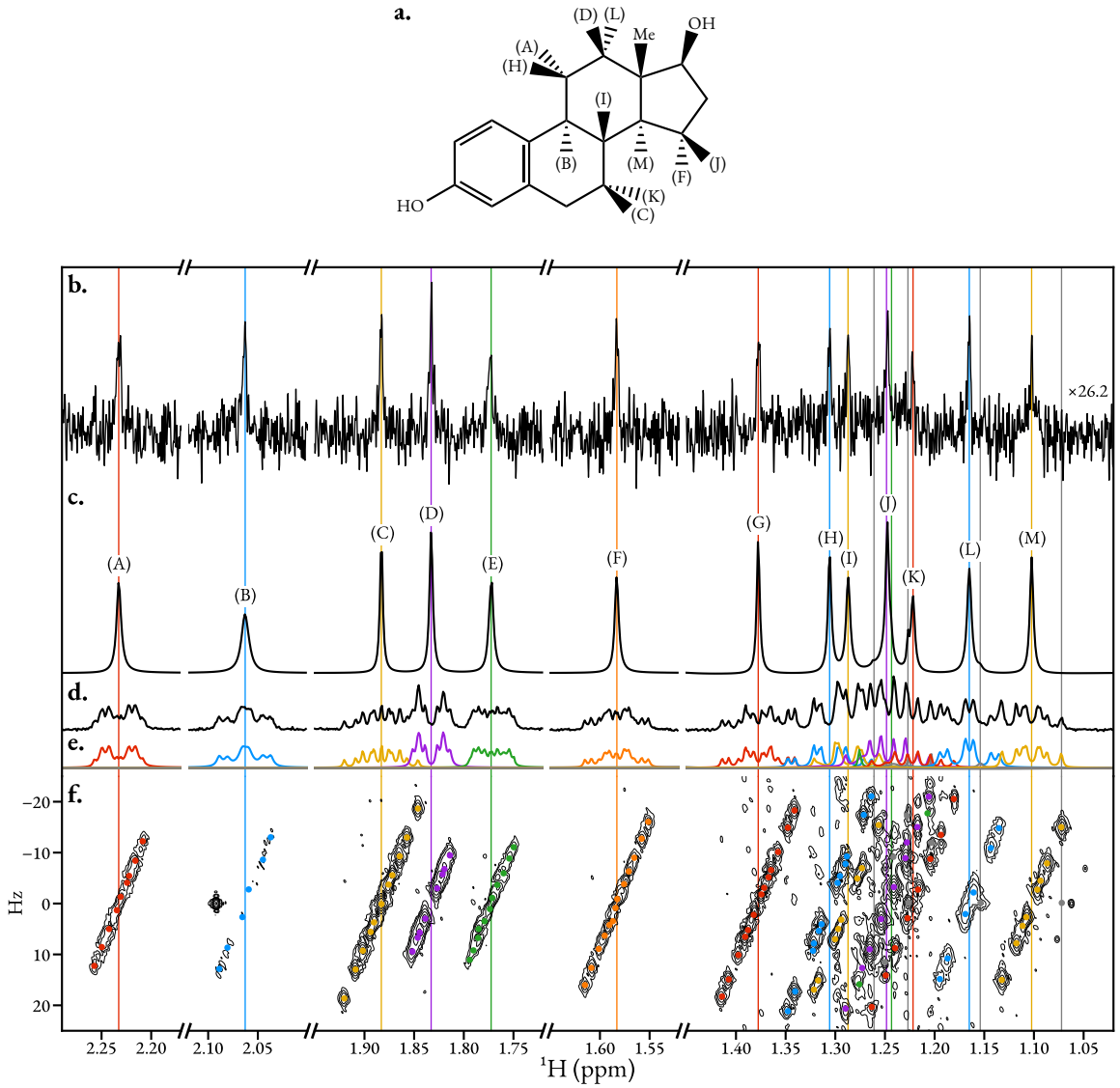


Figure 4.11: The application of CUPID on a 2DJ dataset of 17β-estradiol in DMSO-d_6. **a.** Structure of estradiol. **b.** PSYCHE spectrum. The spectrum has been scaled such that its maximum is of the same magnitude as the CUPID spectrum. **c.** Pure shift spectrum generated via the −45° signal. **d.** Conventional 1D spectrum. **e.** Multiplet structures assigned (ϵ = 2 Hz). **f.** Magnitude-mode 2DJ spectrum, with coloured points denoting the frequencies of oscillators assigned using estimation. Points which are grey do not correspond to the first-order signals associated with estradiol; these arise due to strong coupling artefacts in the 2DJ dataset, and result in artefacts appearing in the panel c. N.B. The 2DJ spectrum was produced using a more concentrated sample of estradiol, since the 2DJ dataset which CUPID was applied on was too insensitive to produce a viable spectrum after sine-bell apodisation.

oscillators present in the model which either correspond to strong coupling artefacts or noise, leading to simplified spectra which appear to adhere to the weak coupling regime. There are cases where strong coupling artefacts do end up in the estimation result. It has been shown that there are circumstances where these can be manually neglected from the parameter set to prevent them from having 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 the level of the estimation routine's accuracy. On numerous occasions within 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 is not attainable when this is so. However, such multiplet structures can still provide valuable insights into the sample being studied.

CUPID is limited by the complexity of the dataset of interest. The reasons for this are two-fold. First, for datasets comprising progressively more peaks in a given spectral region, the difficulty in generating accurate parameter estimates becomes harder. Second, with an increased model order required to estimate the dataset, the computation time increases 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 is anticipated that CUPID will perform admirably on datasets derived from small molecules, though datasets derived from large molecules such as proteins are likely to be too complex and demanding for good results.

Chapter 5

Software

The estimation routine and applications 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 to facilitate analysing 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 [175], including the packages NumPy [176] and Matplotlib [177] enables high-performance scientific computation and data 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 (a) inspect and pre-process the data of interest, (b) determine the regions they wish to estimate and set up the estimation routine, and (c) 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 levels of inheritance. This is exploited in NMR-EsPy to facilitate the creation of numerous objects designed with specific NMR data types in mind (*vide infra*).

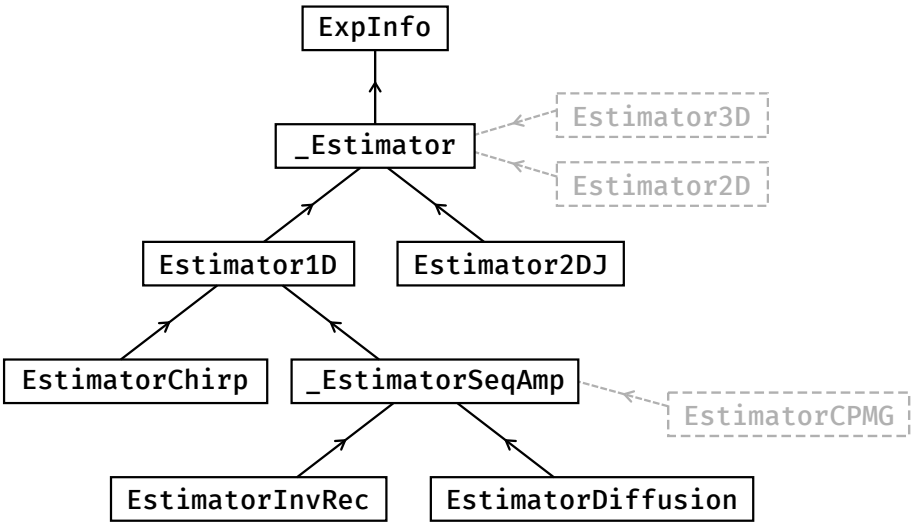


Figure 5.1: An Inheritance tree of the estimation classes in the NMR-EsPy package. Arrows are directed from child classes to the parent class they directly inherit from. Classes in grey are objects that could be added to the application programming interface (API) in the future.

Probably the largest disadvantage in using Python is its slow performance; some of the features which makes Python user-friendly lead to significant runtime overheads*. 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 higher-performance languages is typically a lot greater those with a higher level of abstraction like Python.

5.1.2 Estimator Objects

The fundamental user-facing objects (*classes*) that NMR-EsPy provides are called *estimators*. Instances of these estimator classes contain relevant pieces of data called *attributes* (e.g. the FID and experiment parameters including the spectral width and transmitter offset), as well as *methods* which perform routine applicable to the class 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, outlining the relationship between different estimators

*Python is interpreted rather than compiled, dynamically typed rather than statically typed, and it relies on garbage collection for memory management, rather than requiring the programmer to do this manually. All of these features are slower than the alternatives which are mentioned; these alternatives feature in low-level languages like C, C++, and Rust.

in NMR-EsPy. Basic overviews of each of these are as follows:

- **`ExpInfo`** stores the parameters that were used to run a particular NMR experiment. `ExpInfo` also has methods for the generation of the timepoints and chemical shifts sampled based on these parameters, and for producing synthetic FIDs if a set of signal parameters is provided.
- **`_Estimator`** inherits experimental parameters from `ExpInfo`, but also possesses the NMR dataset itself. 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 be useful as a standalone object, and as such the user is not supposed to use it directly (such an object is often referred to as an abstract class; this is emphasised by starting the object's name with an underscore).
- **`Estimator1D`** handles conventional 1D datasets, such as those in Section 3.1.
- **`_EstimatorSeqAmp`** is an abstract class which enables the estimation of sequential 1D datasets such as those described in Section 3.2. Analysis of a given dataset varies depending on the type of experiment considered, since the function used for fitting amplitudes (Table 3.2) and the means by which that data should be imported varies. As such, this is not designed for direct use; one of the classes which inherit it should be used instead. Under the hood, `_EstimatorSeqAmp` stores a number of `Estimator1D` objects; each of these contains one of the FIDs in the sequential dataset. Estimation simply comprises iterating through these estimators, giving the parameter estimate of the previous estimator to next as its initial guess for NLP.
- **`EstimatorInvRec`** is for the estimation of inversion recovery datasets.
- **`EstimatorDiffusion`** is for the estimation of diffusion datasets.
- **`EstimatorChirp`** is for the estimation of FIDs acquired using single-chirp excitation, as described in Section 3.3.
- **`Estimator2DJ`** is for the estimation of 2DJ data. This class provides features to generate pure shift spectra and assign multiplet structures using CUPID, as described in 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 enable the analysis of CPMG/PROJECT datasets, while `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 the 2D and 3D estimators is given in Section 6.2.

5.1.3 Example Usage

Code Listing 5.1: An example script which makes use of the NMR-EsPy package for the consideration of a 2DJ dataset using CUPID.

```
1  from pathlib import Path
2  import nmrespy as ne
3
4  datapath = Path("~/data/camphor/").expanduser()
5  estimator = ne.Estimator2DJ.new_bruker(datapath / "1")
6  estimator.phase_data(p0=5.238, p1=-6.262)
7  estimator.baseline_correction()
8
9  regions = [
10     (2.55, 2.475), (2.35, 2.23), (2.09, 2.025),
11     (1.95, 1.75), (1.7, 1.61), (1.375, 1.215),
12 ]
13 for region in regions:
14     estimator.estimate(
15         region=region, noise_region=(5.25, 5.21), region_unit="ppm",
16         nlp_trim=2048,
17     )
18
19 estimator.to_pickle(datapath / "camphor_2dj_espy")
20 estimator.write_cupid_to_bruker(datapath, expno=2)
21 thold = 2. * estimator.sw()[1] / estimator.default_pts[1]  # ε = 2f_sw^(2) / N^(2)
22 fig, _ = estimator.plot_result(multiplet_thold=thold)
23 fig.savefig("camphor_2dj.pdf")
```

Code Listing 5.1 provides an example of NMR-EsPy being used for the consideration of 2DJ data; it is similar to the script which was to generate the camphor result presented in Figure 4.9. The script carries out the following:

1. An `Estimator2DJ` object is initialised, which imports and stores the Bruker-formatted camphor dataset, located at the path `$HOME/data/camphor/1`[†] (Line 5).
2. The 2DJ FID is pre-processed to ensure it is phased and features a flat spectral baseline (Lines 6 and 7).
3. The regions of interest are defined, in units of ppm (Lines 9 to 12). Each region is given by a two-element tuple containing its left- and right-boundaries.
4. For each region, parameter estimates are determined using the `Estimator2D.estimate` method, which constructs a filtered sub-FID, and subsequently analyses it (Lines 13 to 17). Most ar-

[†]`$HOME` denotes the current user's home directory on a UNIX-based system. In many scenarios, the tilde symbol (~) can be used to denote the home directory as well.

guments given are self-explanatory; `nlp_trim` can be set to truncate the sub-FID in the direct dimension, to reduce the computational burden on the NLP routine. By default the MDL is used for model order selection.

5. After estimation is complete, the estimator object is saved to a byte stream using Python's "pickling" protocol [178] (Line 19). This enables the estimator to be recovered at a later time.
6. A pure shift spectrum is produced via the $-45°$ signal, and stored to the path `$HOME/data/camphor/2` in a format that enables it to be read by software that supports Bruker datasets, such as TopSpin (Line 20).
7. Finally, a figure depicting the estimation result is produced (making use Matplotlib) and saved (Lines 22 and 23). The form the figure takes is reminiscent of Figure 4.9, though a lot of aesthetic customisation has been applied to the latter.

5.2 The NMR-EsPy GUI

Along with the NMR-EsPy API, an accompanying GUI, created using Python's in-built Tkinter toolkit [179], ships with the package. The GUI can be accessed either via the command line, or within Bruker's TopSpin 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 are provided in Figure 5.2. The GUI comprises two primary windows; the first enables the estimation routine to be set up (Figures 5.2.a1 and 5.2.b1), while the second is for inspecting the result and exporting information about it to the desired format(s) (Figures 5.2.a2 and 5.2.b2).

The set-up window allows the following actions to be carried out:

- The *Pre-Processing* tab facilitates phase correction, application of exponential line-broadening[‡], and baseline correction. If the imported data have already been processed by some other software such as TopSpin[§], this step can be skipped.
- The regions of interest and the noise region are defined with the *Region Selection* tab. Examples of the appearance of the GUI after a number of regions have been defined by the user are given in Figures 5.2.a1 and 5.2.b1; regions of interest are denoted by various colours, while the noise region is grey.

[‡]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`).

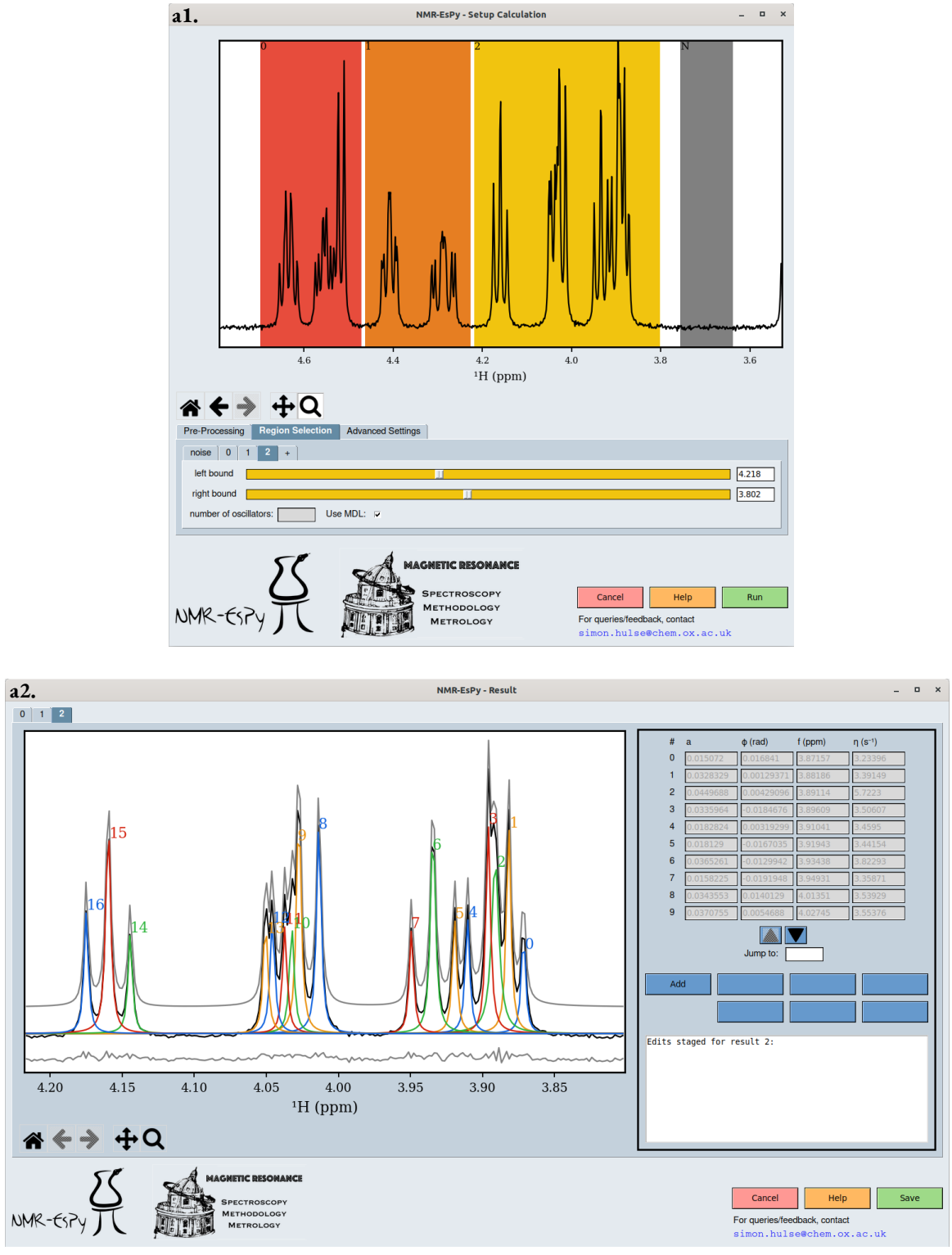
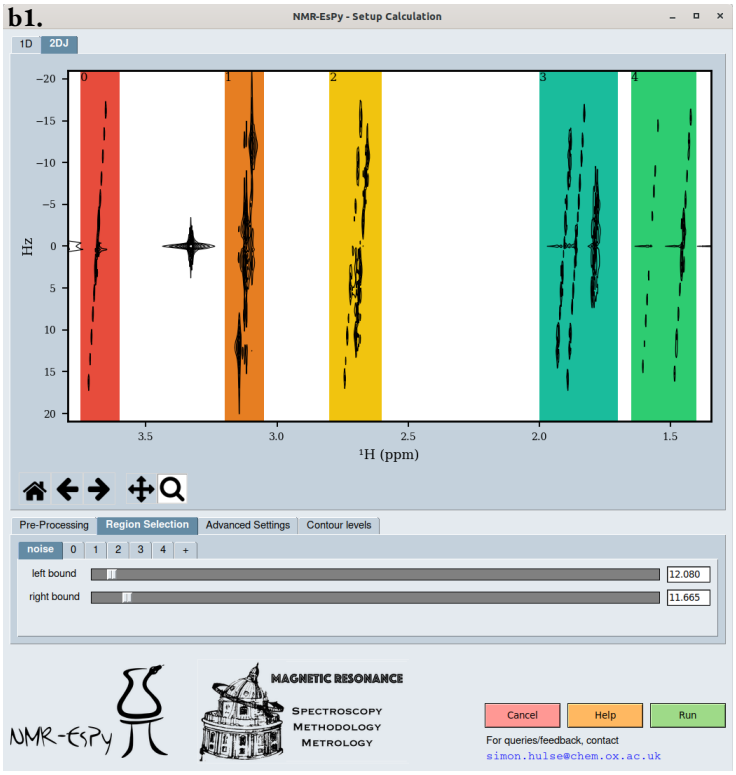
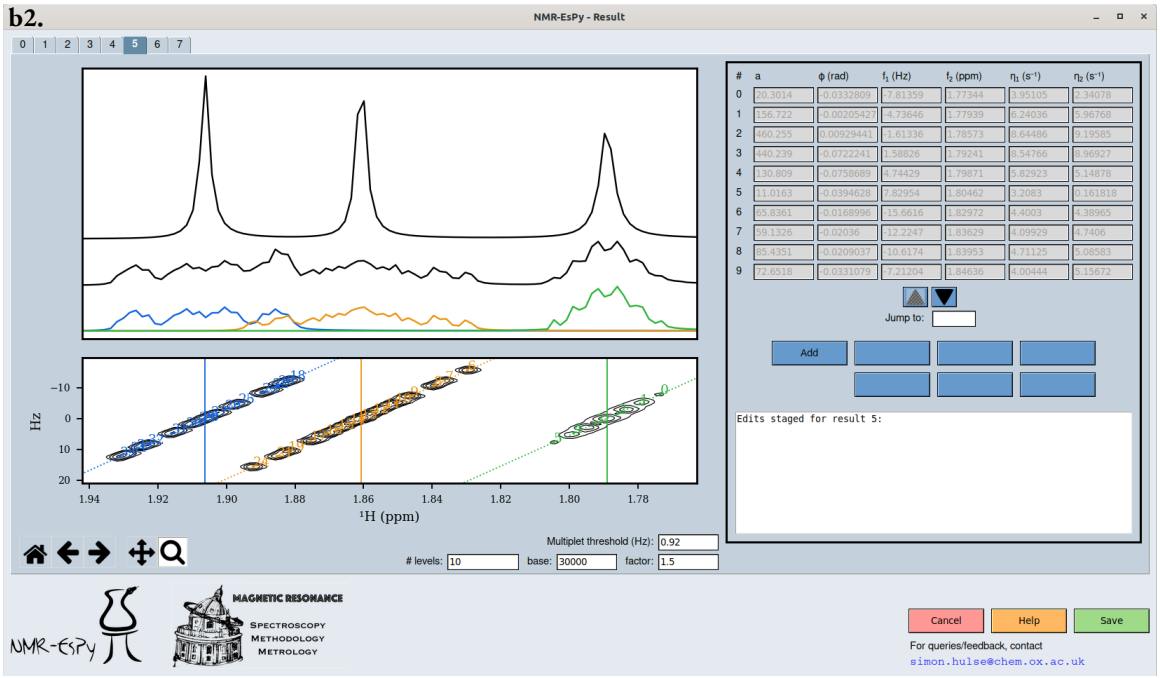


Figure 5.2: Screenshots of windows that form part of the NMR-EsPy GUI for 1D estimation (**a.**) and 2DJ estimation (**b.**). For both data types, the windows used to setup the estimation routine (**1.**) and inspect the result (**2.**) are shown. (Continues on the next page)





Continuation of **Figure 5.2**.

- *Additional Settings* allows features related of 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 before the NLP routine terminates.

The result window features a figure depicting the outcome of the estimation routine with a table of all the estimated parameters. The final stage in using the GUI involves specifying the formats to output the result to. Figures and parameter tables summarising the result can be exported to various formats. Simulated signals constructed using the parameter estimate, such as the pure shift spectrum and individual multiplets structures produced by CUPID, may also be created.

5.3 Summary

In this chapter, an overview of NMR-EsPy, the software associated with this project, has been provided. There are more resources available for further information on NMR-EsPy. A rigorous description of the usage of NMR-EsPy, including details on installation, and a reference for its API are given in the package's documentation, found at https://foroozandehgroup.github.io/NMR-EsPy/. One chapter of the documentation, which provides tutorials to help users gain familiarity with the API, is included in this work, and found in Appendix D.1. 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.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis has focussed on the use of parametric estimation as a means of extracting quantitative information from NMR FIDs. The initial motivation for the work arose from an interest in developing a technique which could be applied to 2DJ datasets, facilitating the generation of broadband homodecoupled NMR spectra with desirable characteristics. To achieve this, the MMEMPM, a 2D extension of the original MPM, was proposed as a suitable estimation routine for considering 2DJ datasets in a holistic fashion. However, initial evaluations of results generated by the MPM on 1D FIDs indicated that a particular flaw frequently arose: the phases of oscillators in the estimated model were commonly inconsistent with expectations based on the data structure. For this reason, a routine was developed which used the result of the MPM as the initial guess of an iterative NLP algorithm; the fidelity to be optimised included the variance of oscillator phases in order to overcome the undesirable phase behaviour. Through a number of examples, it has been shown that improved parameter estimates can be obtained relative to the MPM in isolation. An even more robust means of estimation is possible when routines which incorporate large amounts of user-provided information are used, such as AMARES. However, the uptake of these methods is minimal in NMR, due to the impracticality in supplying all the information required. The routine described in this thesis aims to be a compromise between the MPM, requiring no user information, and those like AMARES.

To alleviate the high computational burden of the proposed method, a frequency-filtration procedure has been developed. The filtration method produces sub-FIDs with reduced numbers of both constituent signals and datapoints, breaking the estimation problem down into a number of smaller-scale ones; drastic reductions in both CPU work and RAM usage are possible by doing this. Furthermore, data filtration enables the user to ignore spectral regions that are not of interest or simply too crowded for meaningful information to be obtained. A specification of the regions

of interest is the primary piece of information that must be supplied manually by the user in order to apply the estimation routine. Work which hasn't been discussed in this thesis was carried out in an attempt to automate this process, though with minimal success*. Because of this — alongside the requirement to sometimes specify a prediction of the model order when the MDL performs inadequately — the routine is not fully automated; some user intervention is required to run it.

Impressive parametrisations of hypercomplex 2DJ FIDs were achieved using the estimation routine, which forms a part of CUPID, a new protocol that is able to generate pure shift spectra featuring no loss of signal (*cf.* "chunking" methods), and peaks with sharp absorption Lorentzian lineshapes (*cf.* the classic 2DJ shear-and-project approach). Furthermore, through a knowledge of the signal frequencies in both dimensions of the dataset, CUPID can decompose 1D spectra into their constituent multiplet structures. Acquiring the information that CUPID can generate with a simple 2DJ dataset usually requires bespoke 3D pulse sequences which have very large run-times.

Beyond to the original goal of 2DJ estimation, two other applications have been developed and presented. The first is an alternative means for determining valuable information about molecular species, including the T_1 and T_2 times of spins, and the diffusion coefficients of molecules, via datasets which comprise a series of FIDs with varying amplitudes. Whereas typical univariate approaches for analysing these datasets involve fitting single datapoints in frequency space, the showcased method instead aims to glean the information from each signal explicitly, in order to overcome the aggregating effect frequently encountered due to peak overlap. While the method works admirably, it couldn't be shown to perform noticeably better than other techniques; in cases where there is severe signal crowding, the difficulty/impossibility of resolving each component leads to similar signal aggregation issues. In these circumstances, the use of a pure shift element in combination with the relevant pulse sequence, as exemplified by the 3D PSYCHE-iDOSY pulse sequence [180], is likely the best course of action for resolving crowded signals (assuming J-couplings are the cause of said crowding).

The last application discussed in this work involves generating ultra-broadband NMR spectra from an experiment involving excitation by a single chirp pulse. With appropriate processing, where each signal is estimated using the MPM, and appropriately back-propagating it based on its frequency, phased spectra with flat baselines are achievable. This work is fairly nascent, and while the initial results presented here show promise, the method is yet to be applied to datasets of greater interest. If further results indicate the method is effective, this could lead to improved sensitivity for ultra-broadband spectra compared with prominent pulse sequences like CHORUS.

*The devised automation method was based on the concept of classifying points which belong to the spectral baseline, as employed in baseline correction methods [42]. Any section of the spectrum which comprised a sufficiently large number of consecutive points which were classified as "non-baseline" was deemed to be a region of interest. Alas, this approach proved to be temperamental, with considerable tweaking of hyperparameters necessary to yield reasonable results. For this reason, the idea was discarded.

The software package NMR-EsPy has been developed to facilitate use of the procedures described in this thesis. The package's API follows a straightforward object-oriented paradigm, which should make it easily accessible to all members of the NMR community that have experience with scientific computing using Python. Furthermore, the availability of an accompanying GUI widens the accessibility of the package further, meaning NMR-EsPy can be used without writing a single line of code.

It is important for an NMR practitioner interested in applying parametric estimation to appreciate its limitations, as well as the benefits that it can offer. Most notably, an understanding is needed of when meaningful information can be extracted from a given dataset. Datasets which abide by the "Goldilocks principle" — being neither too simple nor too complex — are suitable candidates for analysis with parametric estimation. If a dataset is very simplistic, featuring well-separated spectral peaks with high SNR, there is unlikely to be much information to be gained through the application of estimation that cannot be acquired via basic integration and peak-picking. Conversely, if the dataset features many signals with incredibly similar frequencies, it is possible that there simply is not enough information to accurately disentangle all of the individual signal contributions reliably. Despite this constraint, it has hopefully been shown that NMR data estimation is a valuable pursuit; it can often provide richer information than that which is typically available with the conventional FT method, and it opens the way to many fruitful applications which can enhance the value of the NMR experiment.

6.2 Future Work

Any future work related to this project would be centered around extending and improving the NMR-EsPy package. Some possible pursuits include the following:

Improving performance Improvements in the speed of the estimation routine could be realised if the most computationally demanding parts of it were written in a low-level compiled language like C/C++. This may not be the case for 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 packages. However, the NLP routine could benefit from significant improvements if the current Python implementation were ported to one of the aforementioned languages.

There are certain aspects of the routine which could be made more efficient through the use of *multiprocessing*: the act of running distinct parts of a program simultaneously. Most modern computer systems are equipped with multi-core CPUs, with the cores being able to perform sepa-

rate tasks concurrently[†]. In situations where different parts of a program are independent of each other, large time-savings can be achieved by running each part in separate sub-programs across different cores. PYTHON's `multiprocessing` module provides a straightforward interface to achieve this [181].

At the time of writing (version 2.0.0), multiprocessing is not utilised by NMR-EsPy. However, there are some aspects of the estimation routine which could benefit from it. One of these is the computation of the model first- and second-derivatives, required to yield the gradient and Hessian of the fidelity. In Lines 71 to 81 of Code Listing B.6 (a program which generates the fidelity and its derivatives for the 1D case) the second derivatives are computed in accordance with Equations A.8b to A.8j, found in Appendix A.2.1. Each line computes $MN^{(1)}$ of the required $9MN^{(1)}$ gradients, with each line being run in succession. Running each of these in separate sub-programs — across 9 CPU cores, if available — may well be more efficient. A 9-fold increase in speed will not be realised however, since there is an overhead in setting up a "pool" of sub-programs; only in situations where significant computational work is being carried out in each sub-program will performance improvements be realised. It is unlikely that much benefit will be seen for most 1D FIDs, given that (a) the number of derivatives to compute is generally not particularly large, and (b) computing the model gradients isn't the most time-consuming aspect of 1D Hessian computation anyway (recall Table 2.3). However, 2D FID estimation could well be sped up considerably with the use of multiprocessing, given that generating the $20MN^{(1)}N^{(2)}$ second derivatives takes up a considerable amount of the total Hessian computation time (again, see Table 2.3). This would be particularly true if the CPU being used featured at least 20 cores, which is commonplace for modern workstations.

A general-purpose platform 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 parameter tables and figures. Incorporating other estimation methods into the software would therefore be rather straightforward, such that NMR-EsPy could become a general-purpose platform for using and comparing the different methods.

2D and 3D datasets Another potential future venture would be to extend the routine for the consideration of multidimensional datasets that comprise amplitude- or phase-modulated pairings. As shown in Chapter 4, the routine performs capably on hypercomplex 2D datasets, so pro-

[†]N.B. multiprocessing is a distinct concept to *multithreading*; the latter describes the process by which a single CPU core runs a number of separate programs (*threads*) by dedicating a certain amount of time to one, before continuing to work on a different thread etc. In essence, this means that a CPU core doesn't have to run one program to completion before starting a new program.

vided appropriate data treatment is applied, general 2D datasets should be well within the scope of NMR-EsPy. 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 sensing applications, in which the direction (i.e. azimuth and elevation angles) as well as the frequency of waves arriving at an antenna are sought [182]. However, the computational burden required would likely be too high, both in terms of CPU usage and memory requirements, for the routine to be practicable beyond the simplest 3D datasets made of few datapoints and constituent signals.

2DJ datasets are rather anomalous among multidimensional datasets in that they have very densely populated indirect dimensions, due to the chemical shift evolution not occurring in $t^{(1)}$. For this reason, it is not necessary to be concerned about filtering the data in the indirect dimension. However, for experiments like heteronuclear single quantum coherence spectroscopy (HSQC), where chemical shift evolution occurs in both dimensions, 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 heavily truncated, usually as a time-saving measure. The FT of such signals without any treatment would produce spectra with considerable truncation artefacts; attempting to filter would lead to unusable sub-FIDs due to the influence of these artefacts.

Treating truncated datasets could possibly be handled in a few ways, beyond running the experiment with a larger number of indirect-dimension increments. The first option would be to propagate the FID further in time using LP. Another option would be to apply apodisation in order to reshape the indirect FIDs, and remove truncation artefacts. Exponential apodisation would lead to spectra with broader lineshapes, such that wider spectral regions would need to be selected; furthermore the resolution of the data would be decreased. Resolution enhancement methods like sine-bell apodisation can produce FIDs without truncation artefacts and with good resolution. However, applying a non-exponential weighting to the dataset would render it incompatible with the estimation routine in its current form; a modified routine with a suitable model would need to be implemented.

"CUPID-DOSY" The PSYCHE-iDOSY experiment offers better signal resolution to diffusion NMR datasets. A similar 3D pulse sequence could be developed which instead produces a series of 2DJ datasets whose amplitudes are attenuated though diffusion. With such a dataset, each 2DJ FID could be estimated using a method analogous to that for 1D data, presented in Section 3.2.4. After estimation, each 2DJ FID increment could then be recast as a pure shift spectrum via the $-45°$ signal. In effect, this procedure would be an extension of CUPID for a series of inter-related 2DJ spectra.

Incorporating constraints into the NLP routine The estimation routine in its current form is designed to require as little user input as possible while producing faithful parameter estimates. However, in particularly challenging circumstances, most notably when there is considerable overlap between numerous signals, the inclusion of oscillator phase variance alone can be insufficient to produce estimates which closely agree with known features of the data. When this is the case, it would be useful to enable users to specify additional knowledge, incorporated into the routine as regularising terms, *after* the initial estimation has been performed. This concept, while similar to AMARES, differs in that AMARES requires very large quantities or prior knowledge *before* estimation. A lot of the heavy lifting towards an accurate parameter estimate could be achieved using phase variance-regularised NLP, with the user subsequently indicating erroneous features.

Take the cyclosporin A result (Figure 3.3) as an example. As discussed already, due to severe signal overlap, the assigned oscillators related to spin (E), while at the correct frequencies, do not have appropriate amplitudes for a dq multiplet structure. As such, re-running NLP with a new fidelity of

$$\underbrace{\mathcal{F}_{\phi}\left(\boldsymbol{\theta} \,|\, \boldsymbol{Y}\right)}_{\text{Equation 2.46}} + \mathrm{Var}\left(a_{\mathrm{E}1}, \tfrac{1}{3}a_{\mathrm{E}2}, a_{\mathrm{E}3}, \tfrac{1}{3}a_{\mathrm{E}4}, \tfrac{1}{3}a_{\mathrm{E}5}, a_{\mathrm{E}6}, \tfrac{1}{3}a_{\mathrm{E}7}, a_{\mathrm{E}8}\right)$$

would likely lead to an improved result. $\{\mathrm{E}1, \cdots, \mathrm{E}8\} \subset \{1, \cdots, M\}$ are the indices of the oscillators corresponding to spin (E), i.e. the green oscillators in the Figure 3.3. Similar terms could be included for spins (D) and (F) as well.

Bibliography

[1] F. Bloch, W. W. Hansen, M. Packard, *Physical Review* **1946**, *69*, 127–127.

[2] E. M. Purcell, H. C. Torrey, R. V. Pound, *Physical Review* **1946**, *69*, 37.

[3] E. D. Becker, *Analytical Chemistry* **1993**, *65*, 295A–302A.

[4] *Nature (London)* **1952**, *170*, 911–912.

[5] G. R. Eaton, S. S. Eaton, K. M. Salikhov, *Foundations of modern EPR*, World Scientific, Singapore; London, **1998**.

[6] W. D. Knight, *Physical Review* **1949**, *76*, 1259–1260.

[7] W. G. Proctor, F. C. Yu, *Physical Review* **1950**, *77*, 717–717.

[8] W. C. Dickinson, *Physical Review* **1950**, *77*, 736–737.

[9] R. R. Ernst, W. A. Anderson, *Review of Scientific Instruments* **1966**, *37*, 93–102.

[10] R. Freeman, G. A. Morris, *Journal of Magnetic Resonance* **2015**, *250*, 80–84.

[11] J. W. Cooley, J. W. Tukey, *Mathematics of Computation* **1965**, *19*, 297–301.

[12] J. Jeener, *Ampere International Summer School Basko Polje Yugoslavia* **1971**.

[13] J. Jeener, G. Alewaeters, *Progress in Nuclear Magnetic Resonance Spectroscopy* **2016**, *94-95*, 75–80.

[14] W. P. Aue, E. Bartholdi, R. R. Ernst, *The Journal of Chemical Physics* **1976**, *64*, 2229–2246.

[15] M. P. Williamson, T. F. Havel, K. Wüthrich, *Journal of Molecular Biology* **1985**, *182*, 295–315.

[16] D. Marion, P. C. Driscoll, L. E. Kay, P. T. Wingfield, A. Bax, A. M. Gronenborn, G. M. Clore, *Biochemistry* **1989**, *28*, 6150–6156.

[17] L. Kay, G. Clore, A. Bax, A. Gronenborn, *Science* **1990**, *249*, 411–414.

[18] K. Pervushin, R. Riek, G. Wider, K. Wüthrich, *Proceedings of the National Academy of Sciences* **1997**, *94*, 12366–12371.

[19] R. R. Ernst, *Angewandte Chemie International Edition* **1992**, *31*, 805–823.

[20] K. Wüthrich, *Angewandte Chemie International Edition* **2003**, *42*, 3340–3363.

[21] P. J. Hore, J. A. Jones, S. Wimperis, *NMR: the Toolkit: How Pulse Sequences Work*, Second Edition, Oxford University Press, Oxford, **2015**.

[22] M. H. Levitt, *Spin dynamics: Basics of Nuclear Magnetic Resonance*, Second Edition, John Wiley & Sons, Ltd, Chichester, **2008**.

[23] J. Cavanagh, W. J. Fairbrother, A. G. Palmer, M. Rance, N. J. Skelton, *Protein NMR Spectroscopy: Principles and Practice*, Second Edition, Academic Press, Burlington, Massachusetts, **2007**.

[24] M. Goldman, *Quantum Description of High-Resolution NMR in Liquids*, First Edition, Clarendon Press, Oxford, **1988**.

[25] A. Abragam, *The Principles of Nuclear Magnetism*, First Paperback Edition, Clarendon Press, Oxford, **1985**.

[26] I. Kuprov, *Spin : from basic symmetries to quantum optimal control*, First Edition, Springer Nature, Cham, Switzerland, **2023**.

[27] N. J. Stone, *Table of Recommended Nuclear Magnetic Dipole Moments: Part I - Long-lived States*, INDC(NDS)-0794, IAEA, **2019**.

[28] E. Tiesinga, P. J. Mohr, D. B. Newell, B. N. Taylor, *Rev. Mod. Phys.* **2021**, *93*, 025010.

[29] D. D. Traficante, *Concepts in Magnetic Resonance* **1991**, *3*, 171–177.

[30] M. Goldman, *Journal of Magnetic Resonance* **2001**, *149*, 160–187.

[31] I. Kuprov, N. Wagner-Rundell, P. Hore, *Journal of Magnetic Resonance* **2007**, *184*, 196–206.

[32] J. Giberson, J. Scicluna, N. Legge, J. Longstaffe in *Annual Reports on NMR Spectroscopy*, *Vol. 102*, (Ed.: G. A. Webb), Academic Press, Amsterdam, **2021**, pp. 153–246.

[33] H. Kovacs, R. Kuemmerle, D. Moskau, B. Perrone in *Encyclopedia of Biophysics*, (Eds.: G. Roberts, A. Watts), Springer, Berlin, Heidelberg, **2020**, pp. 1–8.

[34] J. Keeler, *Understanding NMR spectroscopy*, Second Edition, Wiley, Chichester, **2010**.

[35] C. E. Shannon, *Proceedings of the IRE* **1949**, *37*, 10–21.

[36] G. A. Morris, H. Barjat, T. J. Home, *Progress in Nuclear Magnetic Resonance Spectroscopy* **1997**, *31*, 197–257.

[37] C. Tang, *Journal of Magnetic Resonance Series A* **1994**, *109*, 232–240.

[38] P. J. Hore, *Nuclear Magnetic Resonance*, Second Edition, Oxford University Press, Oxford, **2015**.

[39] T. D. W. Claridge, *High-Resolution NMR Techniques in Organic Chemistry*, Third Edition, Elsevier, Amsterdam, **2016**.

[40] E. Bartholdi, R. Ernst, *Journal of Magnetic Resonance (1969)* **1973**, *11*, 9–19.

[41] A. Webb in *Magnetic Resonance Technology: Hardware and System Component Design*, The Royal Society of Chemistry, **2016**.

[42] W. Dietrich, C. H. Rüdel, M. Neumann, *Journal of Magnetic Resonance (1969)* **1991**, *91*, 1–11.

[43] J. Carlos Cobas, M. A. Bernstein, M. Martín-Pastor, P. G. Tahoces, *Journal of Magnetic Resonance* **2006**, *183*, 145–151.

[44] J. Keeler, D. Neuhaus, *Journal of Magnetic Resonance (1969)* **1985**, *63*, 454–472.

[45] W. Aue, J. Karhan, R. Ernst, *The Journal of Chemical Physics* **1976**, *64*, 4226–4227.

[46] G. A. Morris in *eMagRes*, (Eds.: R. K. Harris, R. L. Wasylishen), John Wiley & Sons, Ltd, **2009**.

[47] A. L. Davis, J. Keeler, E. D. Laue, D. Moskau, *Journal of Magnetic Resonance (1969)* **1992**, *98*, 207–216.

[48] M. A. Bernstein, Ed., *Magnetic Resonance in Chemistry Vol. 54.6* (**2016**): *Reaction monitoring using NMR*.

[49] A.-H. Emwas, R. Roy, R. T. McKay, L. Tenori, E. Saccenti, G. A. N. Gowda, D. Raftery, F. Alahmari, L. Jaremko, M. Jaremko, D. S. Wishart, *Metabolites* **2019**, *9*.

[50] D. S. Stephenson, *Progress in Nuclear Magnetic Resonance Spectroscopy* **1988**, *20*, 515–626.

[51] P. Koehl, *Progress in Nuclear Magnetic Resonance Spectroscopy* **1999**, *34*, 257–299.

[52] D. Marion, A. Bax, *Journal of Magnetic Resonance (1969)* **1989**, *83*, 205–211.

[53] G. Zhu, A. Bax, *Journal of Magnetic Resonance (1969)* **1992**, *98*, 192–199.

[54] G. U. Yule, *Philosophical Transactions of the Royal Society of London. Series A* **1927**, *226*, 267–298.

[55] G. T. Walker, *Proceedings of the Royal Society of London. Series A.* **1931**, *131*, 518–532.

[56] N. Levinson, *Journal of Mathematics and Physics* **1946**, *25*, 261–278.

[57] J. Durbin, *Revue de l'Institut International de Statistique* **1960**, *28*, 233.

[58] R. Kumaresan, D. Tufts, *IEEE Transactions on Acoustics Speech and Signal Processing* **1982**, *30*, 833–840.

[59] R. Kumaresan, *IEEE Transactions on Acoustics Speech and Signal Processing* **1983**, *31*, 217–220.

[60] S. Y. Kung, K. S. Arun, D. V. B. Rao, *J. Opt. Soc. Am.* **1983**, *73*, 1799–1811.

[61] H. Barkhuijsen, R. de Beer, W. Bovée, D. van Ormondt, *Journal of Magnetic Resonance (1969)* **1985**, *61*, 465–481.

[62] H. Barkhuijsen, R. De Beer, W. M. M. J. Bovee, J. H. N. Creyghton, D. Van Ormondt, *Magnetic Resonance in Medicine* **1985**, *2*, 86–89.

[63] H. Barkhuijsen, R. de Beer, D. van Ormondt, *Journal of Magnetic Resonance (1969)* **1987**, *73*, 553–557.

[64] R. De Beer, D. van Ormondt, W. Pijnappel, J. van der Veen, *Israel Journal of Chemistry* **1988**, *28*, 249–261.

[65] W. Pijnappel, A. van den Boogaart, R. de Beer, D. van Ormondt, *Journal of Magnetic Resonance (1969)* **1992**, *97*, 122–134.

[66] G. R. de Prony, *Journal Polytechnique ou Bulletin du Travail fait a l'Ecole Centrale des Travaux Publics* **1795**.

[67] Y. Hua, T. Sarkar, *IEEE Transactions on Acoustics Speech and Signal Processing* **1990**, *38*, 814–824.

[68] Y. Hua, T. Sarkar, *Signal Processing* **1990**, *21*, 195–198.

[69] Y. Hua, T. Sarkar, *IEEE Transactions on Signal Processing* **1991**, *39*, 892–900.

[70] Y.-Y. Lin, P. Hodgkinson, M. Ernst, A. Pines, *J. Magn. Reson.* **1997**, *128*, 30–41.

[71] S. Fricke, J. Seymour, M. Battistel, D. Freedberg, C. Eads, M. Augustine, *Journal of Magnetic Resonance* **2020**, *313*, 106704.

[72] D. Wörtge, M. Parziale, J. Claussen, B. Mohebbi, S. Stapf, B. Blümich, M. Augustine, *Journal of Magnetic Resonance* **2023**, *351*, 107435.

[73] Y. Hua, *IEEE Transactions on Signal Processing* **1992**, *40*, 2267–2280.

[74] F.-J. Chen, C. Fung, C.-W. Kok, S. Kwong, *IEEE Transactions on Signal Processing* **2007**, *55*, 718–724.

[75] G. H. Golub, V. Pereyra, *SIAM Journal on Numerical Analysis* **1973**, *10*, 413–432.

[76] J. W. C. van der Veen, R. de Beer, P. R. Luyten, D. van Ormondt, *Magnetic Resonance in Medicine* **1988**, *6*, 92–98.

[77] C. Decanniere, P. Vanhecke, F. Vanstapel, H. Chen, S. Vanhuffel, C. Vandervoort, B. Vantongeren, D. Vanormondt, *Journal of magnetic resonance. Series B* **1994**, *105*, 31–37.

[78] K. Levenberg, *Quarterly of Applied Mathematics* **1944**, *2*, 164–168.

[79] D. W. Marquardt, *Journal of the Society for Industrial and Applied Mathematics* **1963**, *11*, 431–441.

[80] L. Vanhamme, A. van den Boogaart, S. Van Huffel, *Journal of Magnetic Resonance* **1997**, *129*, 35–43.

[81] K. Krishnamurthy, *Magnetic Resonance in Chemistry* **2013**, *51*, 821–829.

[82] K. Krishnamurthy, *Magnetic Resonance in Chemistry* **2021**, *59*, 757–791.

[83] G. L. Bretthorst, *Journal of Magnetic Resonance (1969)* **1990**, *88*, 533–551.

[84] G. L. Bretthorst, *Journal of Magnetic Resonance (1969)* **1990**, *88*, 552–570.

[85] G. L. Bretthorst, *Journal of Magnetic Resonance (1969)* **1990**, *88*, 571–595.

[86] G. Bretthorst, *Journal of Magnetic Resonance (1969)* **1991**, *93*, 369–394.

[87] G. Bretthorst, *Journal of Magnetic Resonance (1969)* **1992**, *98*, 501–523.

[88] K. Krishnamurthy, A. M. Sefler, D. J. Russell, *Magnetic Resonance in Chemistry* **2017**, *55*, 224–232.

[89] D.-W. Li, A. L. Hansen, C. Yuan, L. Bruschweiler-Li, R. Brüschweiler, *Nature communications* **2021**, *12*, 5229–5229.

[90] 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, D. Wilhelm, *Journal of Magnetic Resonance* **2023**, *347*, 107357.

[91] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, *Proceedings of the IEEE* **1998**, *86*, 2278–2324.

[93] J. Nocedal, S. J. Wright, *Numerical Optimization*, Second Edition, Springer, New York, **2006**.

[94] J.-B. Poullet, D. M. Sima, S. Van Huffel, *Journal of Magnetic Resonance* **2008**, *195*, 134–144.

[95] H. Hogben, M. Krzystyniak, G. Charnock, P. Hore, I. Kuprov, *Journal of magnetic resonance (1997)* **2011**, *208*, 179–194.
[96] S. I. Kabanikhin, *Journal of Inverse and Ill-Posed Problems* **2008**, *16*, 317–357.
[97] R. Fletcher, *Practical Methods of Optimization*, Second Edition, Wiley, Chichester, **1987**.
[98] J. A. Nelder, R. Mead, *The Computer Journal* **1965**, *7*, 308–313.
[99] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, *Science* **1983**, *220*, 671–680.
[100] M. J. Powell, *Cambridge NA Report NA2009/06 University of Cambridge Cambridge* **2009**, *26*.
[101] G. Golub, C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, **2013**.
[102] Y. Hua, T. Sarkar, *IEEE Transactions on Antennas and Propagation* **1989**, *37*, 229–234.
[103] W. E. Arnoldi, *Quarterly of Applied Mathematics* **1951**, *9*, 17–29.
[104] `scipy.sparse.linalg.svds`, https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.svds.html, Accessed 15/8/2023.
[105] H. Akaike, *IEEE Transactions on Automatic Control* **1974**, *19*, 716–723.
[106] G. Schwarz, *Annals of Statistics* **1978**, *6*, 461–464.
[107] J. Rissanen, *Automatica* **1978**, *14*, 465–471.
[108] M. Wax, T. Kailath, *IEEE Transactions on Acoustics Speech and Signal Processing* **1985**, *33*, 387–392.
[109] N. I. Gould, D. Orban, A. Sartenaer, P. L. Toint, *4OR* **2005**, *3*, 227–241.
[110] N. I. Fisher, *Statistical Analysis of Circular Data*, Cambridge University Press, **1993**.
[111] `pyutils/line_profiler` repository, https://github.com/pyutils/line_profiler, Accessed 14/8/2023.
[112] `pythonprofilers/memory_profiler` repository, https://github.com/pythonprofilers/memory_profiler, Accessed 14/8/2023.
[113] `scipy.sparse.csr_array`, https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_array.html#scipy.sparse.csr_array, Accessed 15/8/2023.
[114] M. Mayzel, K. Kazimierczuk, V. Y. Orekhov, *Chemical Communications* **2014**, *50*, 8947–8950.
[115] D. Gołowicz, P. Kasprzak, K. Kazimierczuk, *Sensors (Basel)* **2020**, *20*, 1325.
[116] J. Luo, Q. Zeng, K. Wu, Y. Lin, *Journal of Magnetic Resonance* **2020**, *317*, 106772.
[117] A. Verma, S. Bhattacharya, B. Baishya, *RSC Advances* **2018**, *8*, 19990–19999.
[118] G. B. Chavhan, P. S. Babyn, B. Thomas, M. M. Shroff, E. M. Haacke, *RadioGraphics* **2009**, *29*, 1433–1449.
[119] H. Y. Carr, E. M. Purcell, *Phys. Rev.* **1954**, *94*, 630–638.
[120] S. Meiboom, D. Gill, *Review of Scientific Instruments* **1958**, *29*, 688–691.
[121] J. A. Aguilar, M. Nilsson, G. Bodenhausen, G. A. Morris, *Chemical Communications* **2012**, *48*, 811–813.

[122] C. Johnson, *Progress in Nuclear Magnetic Resonance Spectroscopy* **1999**, *34*, 203–256.
[123] G. A. Morris in *eMagRes*, John Wiley & Sons, Ltd, **2009**.
[124] P. W. Atkins, J. De Paula, *Atkins' Physical Chemistry*, Tenth Edition, Oxford University Press, Oxford, **2014**.
[125] E. O. Stejskal, J. E. Tanner, *The Journal of Chemical Physics* **1965**, *42*, 288–292.
[126] H. C. Torrey, *Phys. Rev.* **1956**, *104*, 563–565.
[127] J. E. Tanner, *The Journal of Chemical Physics* **1970**, *52*, 2523–2526.
[128] R. Cotts, M. Hoch, T. Sun, J. Markert, *Journal of Magnetic Resonance (1969)* **1989**, *83*, 252–266.
[129] D. Wu, A. Chen, C. Johnson, *Journal of Magnetic Resonance Series A* **1995**, *115*, 260–264.
[130] M. D. Pelta, G. A. Morris, M. J. Stchedroff, S. J. Hammond, *Magnetic Resonance in Chemistry* **2002**, *40*, S147–S152.
[131] D. Sinnaeve, *Concepts in Magnetic Resonance Part A* **2012**, *40A*, 39–65.
[132] M. Nilsson, M. A. Connell, A. L. Davis, G. A. Morris, *Analytical Chemistry (Washington)* **2006**, *78*, 3040–3045.
[133] W. Windig, J. Hornak, B. Antalek, *Journal of magnetic resonance (1997)* **1998**, *132*, 298–306.
[134] M. Nilsson, G. A. Morris, *Analytical chemistry (Washington)* **2008**, *80*, 3777–3782.
[135] P. Stilbs, K. Paulsen, *Review of Scientific Instruments* **1996**, *67*, 4380–4386.
[136] P. Stilbs, K. Paulsen, P. C. Griffiths, *Journal of Physical Chemistry (1952)* **1996**, *100*, 8180–8189.
[137] Spinach relaxation theory parameters, https://www.spindynamics.org/wiki/index.php?title=Relaxation_theory_parameters, Accessed 24/7/2023.
[138] A. Chen, C. S. Johnson, M. Lin, M. J. Shapiro, *Journal of the American Chemical Society* **1998**, *120*, 9094–9095.
[139] B. G. Davis, A. J. Fairbanks, *Carbohydrate chemistry*, Oxford University Press, Oxford, **2002**.
[140] M. Foroozandeh, *Journal of Magnetic Resonance* **2020**, *318*, 106768.
[141] J.-M. Bohlen, M. Rey, G. Bodenhausen, *Journal of Magnetic Resonance (1969)* **1989**, *84*, 191–197.
[142] J. Bohlen, G. Bodenhausen, *Journal of Magnetic Resonance Series A* **1993**, *102*, 293–301.
[143] K. E. Cano, M. A. Smith, A. Shaka, *Journal of Magnetic Resonance* **2002**, *155*, 131–139.
[144] J. E. Power, M. Foroozandeh, R. W. Adams, M. Nilsson, S. R. Coombes, A. R. Phillips, G. A. Morris, *Chemical Communications* **2016**, *52*, 2916–2919.
[145] M. Foroozandeh, M. Nilsson, G. A. Morris, *Journal of Magnetic Resonance* **2019**, *302*, 28–33.

[146] E. Kupce, R. Freeman, *Journal of Magnetic Resonance Series A* **1995**, *115*, 273–276.
[147] H. Maeda, Y. Yanagisawa, *Journal of Magnetic Resonance* **2019**, *306*, 80–85.
[148] A. Shaka, J. Keeler, T. Frenkiel, R. Freeman, *Journal of Magnetic Resonance (1969)* **1983**, *52*, 335–338.
[149] A. Shaka, J. Keeler, R. Freeman, *Journal of Magnetic Resonance (1969)* **1983**, *53*, 313–340.
[150] A. Shaka, P. Barker, R. Freeman, *Journal of Magnetic Resonance (1969)* **1985**, *64*, 547–552.
[151] N. H. Meyer, K. Zangger, *Angewandte Chemie International Edition* **2013**, *52*, 7143–7146.
[152] R. W. Adams in *eMagRes*, John Wiley & Sons, Ltd, **2014**, pp. 295–310.
[153] K. Zangger, *Progress in Nuclear Magnetic Resonance Spectroscopy* **2015**, *86-87*, 1–20.
[154] G. Wider, R. Baumann, K. Nagayama, R. R. Ernst, K. Wüthrich, *Journal of Magnetic Resonance (1969)* **1981**, *42*, 73–87.
[155] M. J. Thrippleton, R. A. Edden, J. Keeler, *Journal of Magnetic Resonance* **2005**, *174*, 97–109.
[156] A. Bax, R. Freeman, G. A. Morris, *Journal of Magnetic Resonance (1969)* **1981**, *43*, 333–338.
[157] J.-M. Nuzillard, *Journal of Magnetic Resonance Series A* **1996**, *118*, 132–135.
[158] A. Martinez, F. Bourdreux, E. Riguet, J.-M. Nuzillard, *Magnetic Resonance in Chemistry* **2012**, *50*, 28–32.
[159] P. Mutzenhardt, F. Guenneau, D. Canet, *Journal of Magnetic Resonance* **1999**, *141*, 312–321.
[160] V. A. Mandelshtam, H. S. Taylor, *The Journal of Chemical Physics* **1997**, *107*, 6756–6769.
[161] V. A. Mandelshtam, Q. N. Van, A. J. Shaka, *Journal of the American Chemical Society* **1998**, *120*, 12161–12162.
[162] K. Zangger, H. Sterk, *Journal of Magnetic Resonance* **1997**, *124*, 486–489.
[163] J. A. Aguilar, S. Faulkner, M. Nilsson, G. A. Morris, *Angewandte Chemie International Edition* **2010**, *49*, 3901–3903.
[164] E. Kupce, J. Boyd, I. Campbell, *Journal of Magnetic Resonance Series B* **1995**, *106*, 300–303.
[165] A. J. Pell, J. Keeler, *Journal of Magnetic Resonance* **2007**, *189*, 293–299.
[166] J. Garbow, D. Weitekamp, A. Pines, *Chemical Physics Letters* **1982**, *93*, 504–509.
[167] A. Bax, *Journal of Magnetic Resonance (1969)* **1983**, *53*, 517–520.
[168] L. Paudel, R. W. Adams, P. Király, J. A. Aguilar, M. Foroozandeh, M. J. Cliff, M. Nilsson, P. Sándor, J. P. Waltho, G. A. Morris, *Angewandte Chemie International Edition* **2013**, *52*, 11616–11619.

[169] M. Foroozandeh, R. W. Adams, N. J. Meharry, D. Jeannerat, M. Nilsson, G. A. Morris, *Angewandte Chemie International Edition* **2014**, *53*, 6990–6992.

[170] M. Foroozandeh, G. A. Morris, M. Nilsson, *Chemistry – A European Journal* **2018**, *24*, 13988–14000.

[171] M. J. Thrippleton, J. Keeler, *Angewandte Chemie International Edition* **2003**, *42*, 3938–3941.

[172] M. Foroozandeh, R. Adams, P. Kiraly, M. Nilsson, G. Morris, *Chemical Communications* **2015**, *51*, 15410–15413.

[173] P. Kiraly, M. Foroozandeh, M. Nilsson, G. A. Morris, *Chemical Physics Letters* **2017**, *683*, 398–403.

[174] G. Zhu, D. Smith, Y. Hua, *Journal of Magnetic Resonance* **1997**, *124*, 286–289.

[175] P. Virtanen et al., *Nature Methods* **2020**, *17*, 261–272.

[176] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, *Nature* **2020**, *585*, 357.

[177] J. D. Hunter, *Computing in Science & Engineering* **2007**, *9*, 90–95.

[178] `pickle`: Python object serialization, https://docs.python.org/3/library/pickle.html, Accessed 21/9/2023.

[179] `tkinter`: Python interface to Tcl/Tk, https://docs.python.org/3/library/tkinter.html, Accessed 20/9/2023.

[180] M. Foroozandeh, L. Castañar, L. G. Martins, D. Sinnaeve, G. D. Poggetto, C. F. Tormena, R. W. Adams, G. A. Morris, M. Nilsson, *Angewandte Chemie International Edition* **2016**, *55*, 15579–15582.

[181] `multiprocessing` – Process-based parallelism, https://docs.python.org/3/library/multiprocessing.html, Accessed 21/1/2024.

[182] N. Yilmazer, R. Fernandez-Recio, T. K. Sarkar, *Digital Signal Processing* **2006**, *16*, 796–816.

[183] G. Strang, *Linear Algebra and Its Applications*, Fifth Edition, Cengage Learning, Inc, Florence, **2018**.

[184] Y. Pawitan, *In All Likelihood: Statistical Modelling and Inference Using Likelihood*, Oxford University Press, Oxford, **2001**.

[185] Y. Jaradat, M. Masoud, I. Jannoud, A. Manasrah, M. Alia in 2021 International Conference on Information Technology, **2021**, pp. 769–772.

[186] Matlab engine for Python, https://uk.mathworks.com/help/matlab/matlab-engine-for-python.html, Accessed 24/7/2023.

[187] S. Berger, S. Braun, *200 and more NMR Experiments: a Practical Course*, Third Edition, Wiley-VCH, Weinheim, **2004**.

[188] L. Castañar, G. D. Poggetto, A. A. Colbourne, G. A. Morris, M. Nilsson, *Magnetic Resonance in Chemistry* **2018**, *56*, 546–558.

[189] L. A. O'Dell, *Solid State Nuclear Magnetic Resonance* **2013**, *55-56*, 28–41.

Appendix A

Additional Theory

A.1 Mathematical definitions

Descriptions of linear algebra and statistics concepts that are referred to in this thesis are provided here. More detail can be found in numerous relevant texts [183, 184].

A.1.1 Linear algebra

Matrix Rank

The rank of a matrix is defined as the dimension of the vector space spanned by its rows; this is identical to the dimension of the vector space spanned by its columns. Equivalently, rank can be thought of as the number of linearly independent row/column vectors present in the matrix. For a matrix $\boldsymbol{X} \in \mathbb{F}^{m \times n}$, the largest possible rank — often referred to as "full rank" — is $\min(m, n)$.

Singular value decomposition (SVD)

SVD is a generalisation of eigendecomposition — applicable only to square, diagonalisable matrices — for any general matrix $\boldsymbol{X} \in \mathbb{C}^{m \times n}$. The SVD is a factorisation given by

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\dagger}. \tag{A.1}$$

The matrices that make up the factorisation are as follows

- $\boldsymbol{\Sigma} \in \mathbb{C}^{m \times n}$ is a rectangular diagonal matrix with diagonal elements comprising the *singular values* of $\boldsymbol{X}$ in descending order of magnitude. The singular values are the square roots of the non-zero eigenvalues of both $\boldsymbol{X}^{\dagger}\boldsymbol{X}$ and $\boldsymbol{X}\boldsymbol{X}^{\dagger}$.
- $\boldsymbol{U} \in \mathbb{C}^{m \times m}$ is a unitary matrix whose columns comprise the *left singular vectors* of $\boldsymbol{X}$. The left singular vectors are the eigenvectors of the matrix $\boldsymbol{X}\boldsymbol{X}^{\dagger}$.

- $\boldsymbol{V} \in \mathbb{C}^{n\times n}$ is a unitary matrix whose columns comprise the *right singular vectors* or $\boldsymbol{X}$. These are the eigenvectors of the matrix $\boldsymbol{X}^\dagger\boldsymbol{X}$.

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 [185].

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 $\boldsymbol{X} \in \mathbb{F}^{M\times N}$, the matrix is Hankel if

$$x_{m,n} = x_{m+1,n-1} \quad \forall m \in \{1,\cdots,M-1\},\ \forall n \in \{2,\cdots,N\}. \tag{A.2}$$

Similarly, a *Toeplitz matrix* is a matrix in which every *descending* diagonal from left to right possesses identical elements, i.e.

$$x_{m,n} = x_{m+1,n+1} \quad \forall m \in \{1,\cdots,M-1\},\ \forall n \in \{1,\cdots N-1\}. \tag{A.3}$$

A *Vandermonde matrix* is a matrix with rows or columns that feature a geometric progression, i.e.

$$x_{m,n} = x_{m,1}^n \text{ or } x_{m,n} = x_{1,n}^m \quad \forall m \in \{1,\cdots,M\},\ \forall n \in \{1,\cdots N\}. \tag{A.4}$$

A.1.2 Statistics and Probability

Probability Density Function (PDF)

The PDF $p(x) : \mathbb{R} \to \mathbb{R}$ is a function over a continuous sample space which provides relative likelihoods between potential values of x. The probability $P \in [0,1]$ that a random sample obeying a known distribution lies withing the range $[x_a, x_b]$ is given by the integral

$$P(x_a \le x \le x_b) = \int_{x_a}^{x_b} p(x)\mathrm{d}x. \tag{A.5}$$

The integral of $p(x)$ over the entire sample space is defined to be unity. The probability of a specific value $P(x)$ is always 0 by definition, since the width of the region of integration is 0. The PDF can be extended for the consideration of multiple variables; in this case $p(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ is commonly referred to as a *multivariate* PDF.

Likelihood Function and Maximum Likelihood Estimate (MLE)

The likelihood function $\mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{Y})$ is the probability density of some observed data, when viewed from the perspective of the parameters of a model (i.e. the data is taken as a fixed parameter, and the model parameters are taken as variables). The MLE is the set of parameters which maximises the likelihood function:

$$\boldsymbol{\theta}^{(*)} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{Y}). \tag{A.6}$$

A.2 Further Information about NLP

A.2.1 Model Derivatives

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

First derivatives

$$\frac{\partial x}{\partial \theta_m} \equiv \frac{\partial x}{\partial a_m} = \frac{x}{a_m}, \tag{A.7a}$$

$$\frac{\partial x}{\partial \theta_{m+M}} \equiv \frac{\partial x}{\partial \phi_m} = \mathrm{i}x, \tag{A.7b}$$

$$\frac{\partial x}{\partial \theta_{m+(d+1)M}} \equiv \frac{\partial x}{\partial f_m^{(d)}} = 2\pi\mathrm{i}\Delta_t^{(d)} n^{(d)} x, \tag{A.7c}$$

$$\frac{\partial x}{\partial \theta_{m+(d+D+1)M}} \equiv \frac{\partial x}{\partial \eta_m^{(d)}} = -\Delta_t^{(d)} n^{(d)} x. \tag{A.7d}$$

Second derivatives

$$\frac{\partial^2 x}{\partial \theta_m^2} \equiv \frac{\partial^2 x}{\partial a_m^2} = 0, \tag{A.8a}$$

$$\frac{\partial^2 x}{\partial \theta_m \partial \theta_{m+M}} \equiv \frac{\partial^2 x}{\partial a_m \partial \phi_m} = \frac{\mathrm{i}x}{a_m}, \tag{A.8b}$$

$$\frac{\partial^2 x}{\partial \theta_m \partial \theta_{m+(d+1)M}} \equiv \frac{\partial^2 x}{\partial a_m \partial f_m^{(d)}} = \frac{2\pi\mathrm{i}\Delta_t^{(d)} n^{(d)} x}{a_m}, \tag{A.8c}$$

$$\frac{\partial^2 x}{\partial \theta_m \partial \theta_{m+(d+D+1)M}} \equiv \frac{\partial^2 x}{\partial a_m \partial \eta_m^{(d)}} = \frac{-\Delta_t^{(d)} n^{(d)} x}{a_m}, \tag{A.8d}$$

$$\frac{\partial^2 x}{\partial \theta_{m+M}^2} \equiv \frac{\partial^2 x}{\partial \phi_m^2} = -x, \tag{A.8e}$$

$$\frac{\partial^2 x}{\partial\theta_{m+M}\partial\theta_{m+(d+1)M}} \equiv \frac{\partial^2 x}{\partial\phi_m\partial f_m^{(d)}} = -2\pi\Delta_t^{(d)} n^{(d)} x, \tag{A.8f}$$

$$\frac{\partial^2 x}{\partial\theta_{m+M}\partial\theta_{m+(d+D+1)M}} \equiv \frac{\partial^2 x}{\partial\phi_m\partial\eta_m^{(d)}} = -\mathrm{i}\Delta_t^{(d)} n^{(d)} x, \tag{A.8g}$$

$$\frac{\partial^2 x}{\partial\theta_{m+(d+1)M}\partial\theta_{m+(d'+1)M}} \equiv \frac{\partial^2 x}{\partial f_m^{(d)}\partial f_m^{(d')}} = -4\pi^2\left(\Delta_t^{(d)} n^{(d)}\right)\left(\Delta_t^{(d')} n^{(d')}\right) x, \tag{A.8h}$$

$$\frac{\partial^2 x}{\partial\theta_{m+(d+1)M}\partial\theta_{m+(d'+D+1)M}} \equiv \frac{\partial^2 x}{\partial f_m^{(d)}\partial\eta_m^{(d')}} = -2\pi\mathrm{i}\left(\Delta_t^{(d)} n^{(d)}\right)\left(\Delta_t^{(d')} n^{(d')}\right) x, \tag{A.8i}$$

$$\frac{\partial^2 x}{\partial\theta_{m+(d+D+1)M}\partial\theta_{m+(d'+D+1)M}} \equiv \frac{\partial^2 x}{\partial\eta_m^{(d)}\partial\eta_m^{(d')}} = \left(\Delta_t^{(d)} n^{(d)}\right)\left(\Delta_t^{(d')} n^{(d')}\right) x, \tag{A.8j}$$

$$\frac{\partial^2 x}{\partial\theta_i\partial\theta_j} = \frac{\partial^2 x}{\partial\theta_j\partial\theta_i}, \tag{A.8k}$$

$$\frac{\partial^2 x}{\partial\theta_i\partial\theta_j} = 0 \text{ if not specified above.} \tag{A.8l}$$

A.2.2 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 [184: Section 2.7]:

$$\epsilon\left(\theta^{(*)}\right) = \sqrt{\mathrm{diag}\left(\boldsymbol{I}\left(\theta^{(*)}\right)^{-1}\right)}, \tag{A.9}$$

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

$$\boldsymbol{I}(\theta)_{i,j} = -\frac{\partial^2 \ell(\theta \mid \boldsymbol{Y})}{\partial\theta_i\partial\theta_j}. \tag{A.10}$$

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

$$\boldsymbol{I}(\theta)_{i,j} = -\frac{1}{\sigma^2}\Re\left(\left\langle\frac{\partial\boldsymbol{X}}{\partial\theta_i}, \frac{\partial\boldsymbol{X}}{\partial\theta_j}\right\rangle - \left\langle(\boldsymbol{Y}-\boldsymbol{X}), \frac{\partial^2\boldsymbol{X}}{\partial\theta_i\partial\theta_j}\right\rangle\right), \tag{A.11}$$

which very closely resembles the Hessian of θ:

$$\boldsymbol{I}(\theta)_{i,j} = \frac{1}{2\sigma^2}\boldsymbol{H}(\theta)_{i,j}. \tag{A.12}$$

The standard errors therefore take the form

$$\boldsymbol{\epsilon}\left(\boldsymbol{\theta}^{(*)}\right) = \sqrt{2\sigma^2 \operatorname{diag}\left(\boldsymbol{H}\left(\boldsymbol{\theta}^{(*)}\right)^{-1}\right)}. \tag{A.13}$$

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

$$2\sigma^2 = \frac{1}{N_{\text{tot}} - 1}\left\|\boldsymbol{W}\right\|^2 = \frac{1}{N_{\text{tot}} - 1}\left\|\boldsymbol{Y} - \boldsymbol{X}\left(\boldsymbol{\theta}^{(*)}\right)\right\|^2. \tag{A.14}$$

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

$$\boldsymbol{\epsilon}\left(\boldsymbol{\theta}^{(*)}\right) = \sqrt{\frac{\mathcal{F}\left(\boldsymbol{\theta}^{(*)}\right)\operatorname{diag}\left(\boldsymbol{H}\left(\boldsymbol{\theta}^{(*)}\right)^{-1}\right)}{N_{\text{tot}} - 1}} \tag{A.15}$$

A.3 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 $\{\boldsymbol{Y}^{\cos}, \boldsymbol{Y}^{\sin}\}$, given by Equation 1.24 with $D = 2$ and $\zeta = \{\cos(\cdot), \sin(\cdot)\}$, taking the forms (with noise neglected)

$$y^{\cos}_{n^{(1)},n^{(2)}} = \xi_{n^{(1)},n^{(2)}} c^{(1)}_{n^{(1)},n^{(2)}}\left(c^{(2)}_{n^{(1)},n^{(2)}} + \mathrm{i}s^{(2)}_{n^{(1)},n^{(2)}}\right), \tag{A.16a}$$

$$y^{\sin}_{n^{(1)},n^{(2)}} = \xi_{n^{(1)},n^{(2)}} s^{(1)}_{n^{(1)},n^{(2)}}\left(c^{(2)}_{n^{(1)},n^{(2)}} + \mathrm{i}s^{(2)}_{n^{(1)},n^{(2)}}\right), \tag{A.16b}$$

$$\xi_{n^{(1)},n^{(2)}} = \sum_m a_m \exp\left(-\eta^{(1)}_m n^{(1)}\Delta^{(1)}_t - \eta^{(2)}_m n^{(2)}\Delta^{(2)}_t\right), \tag{A.16c}$$

$$(c/s)^{(1/2)}_{n^{(1)},n^{(2)}} = \sum_m \cos/\sin\left(2\pi f^{(1/2)} n^{(1/2)}\Delta^{(1/2)}\right), \tag{A.16d}$$

$$\forall M \in \{1, \cdots M\},\ \forall n^{(1)} \in \{0, \cdots N^{(1)} - 1\},\ \forall n^{(2)} \in \{0, \cdots N^{(2)} - 1\},$$

where, as with the 1D case in the main text, it has been assumed the data is phased, such that $\phi_m = 0\ \forall m$. Four matrices $\boldsymbol{\psi}_{\pm\pm}$ are then constructed of the form

$$\begin{aligned}\psi_{\pm\pm,n^{(1)},n^{(2)}} &= \xi_{n^{(1)},n^{(2)}}\left(c^{(1)}_{n^{(1)},n^{(2)}} \pm^{(1)} \mathrm{i}s^{(1)}_{n^{(1)},n^{(2)}}\right)\left(c^{(2)}_{n^{(1)},n^{(2)}} \pm^{(2)} \mathrm{i}s^{(2)}_{n^{(1)},n^{(2)}}\right) \\ &\equiv \Re\left(y^{\cos}_{n^{(1)},n^{(2)}}\right) \pm^{(1)} \pm^{(2)} - \Im\left(y^{\sin}_{n^{(1)},n^{(2)}}\right) + \mathrm{i}\left(\pm^{(1)}\Re\left(y^{\sin}_{n^{(1)},n^{(2)}}\right) \pm^{(2)} \Im\left(y^{\cos}_{n^{(1)},n^{(2)}}\right)\right),\end{aligned} \tag{A.17}$$

from which the matrices $\boldsymbol{T}_{1\to4} \in \mathbb{C}^{2N^{(1)}\times 2N^{(2)}}$ are generated:

$$\boldsymbol{T}_1 = \begin{bmatrix} \boldsymbol{\Psi}_{++} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{A.18a}$$

$$\boldsymbol{T}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \boldsymbol{\Psi}^{\leftrightarrow(1)}_{-+} & \mathbf{0} \end{bmatrix}^{\circlearrowleft(1)}, \tag{A.18b}$$

$$\boldsymbol{T}_3 = \begin{bmatrix} \mathbf{0} & \boldsymbol{\Psi}^{\leftrightarrow(2)}_{+-} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}^{\circlearrowleft(2)}, \tag{A.18c}$$

$$\boldsymbol{T}_4 = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Psi}^{\leftrightarrow(1,2)}_{--} \end{bmatrix}^{\circlearrowleft(1,2)}. \tag{A.18d}$$

The virtual echo is then given by $\boldsymbol{Y}_{\mathrm{VE}} = \sum_{i=1}^{4} \boldsymbol{T}_i$, with the first row and column divided by two. For a full outline of the 2D filtering procedure, see Algorithm A.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 $\{\boldsymbol{Y}^{\mathrm{pos}}, \boldsymbol{Y}^{\mathrm{neg}}\}$, given by Equation 1.24 with $D = 2$ and $\zeta = \{\exp(\mathrm{i}\cdot), \exp(-\mathrm{i}\cdot)\}$. These can be used to generate an amplitude modulated pair via

$$\boldsymbol{Y}^{\cos} = \frac{\boldsymbol{Y}^{\mathrm{pos}} + \boldsymbol{Y}^{\mathrm{neg}}}{2}, \tag{A.19a}$$

$$\boldsymbol{Y}^{\sin} = \frac{\boldsymbol{Y}^{\mathrm{pos}} - \boldsymbol{Y}^{\mathrm{neg}}}{2\mathrm{i}}, \tag{A.19b}$$

which can then be processed as described above to form $\boldsymbol{Y}_{\mathrm{VE}}$.

A.4 Additional algorithms

Algorithm A.1 The MMEMPM. TruncatedSVD is a routine which computes the first M SVD components of a matrix, using some iterative approach such as the Rayleigh-Ritz method.

procedure MMEMPM($\boldsymbol{Y} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}, M \in \mathbb{N}$)
  $L^{(1)}, L^{(2)} \leftarrow \left\lfloor N^{(1)}/2 \right\rfloor, \left\lfloor N^{(2)}/2 \right\rfloor$;
  for $n^{(1)} \leftarrow \{0, \cdots, N^{(1)} - 1\}$ **do**
    $$\boldsymbol{H}_{\boldsymbol{y},n^{(1)}} \leftarrow \begin{bmatrix} y_{n^{(1)},0} & y_{n^{(1)},1} & \cdots & y_{n^{(1)},N^{(2)}-L^{(2)}} \\ y_{n^{(1)},1} & y_{n^{(1)},2} & \cdots & y_{n^{(1)},N^{(2)}-L^{(2)}+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n^{(1)},L^{(2)}-1} & y_{n^{(1)},L^{(2)}} & \cdots & y_{n^{(1)},N^{(2)}-1} \end{bmatrix};$$
  end for
  $$\boldsymbol{E}_{\boldsymbol{Y}} \leftarrow \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{y},0} & \boldsymbol{H}_{\boldsymbol{y},1} & \cdots & \boldsymbol{H}_{\boldsymbol{y},N^{(1)}-L^{(1)}} \\ \boldsymbol{H}_{\boldsymbol{y},1} & \boldsymbol{H}_{\boldsymbol{y},2} & \cdots & \boldsymbol{H}_{\boldsymbol{y},N^{(1)}-L^{(1)}+1} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{H}_{\boldsymbol{y},L^{(1)}-1} & \boldsymbol{H}_{\boldsymbol{y},L^{(1)}} & \cdots & \boldsymbol{H}_{\boldsymbol{y},N^{(1)}-1} \end{bmatrix};$$
  $\boldsymbol{U}_M, \boldsymbol{\Sigma}_M, \boldsymbol{V}_M^\dagger \leftarrow$ TruncatedSVD $(\boldsymbol{E}_{\boldsymbol{Y}}, M)$;
  $\boldsymbol{P} \leftarrow \boldsymbol{0} \in \mathbb{C}^{L^{(1)}L^{(2)} \times L^{(1)}L^{(2)}}$;
  $r \leftarrow 0$
  for $i = 0, \cdots, L^{(2)} - 1$ **do**
    for $j = 0, \cdots, L^{(1)} - 1$ **do**
      $c \leftarrow i + jL^{(2)}$;
      $p_{r,c} \leftarrow 1$;
      $r \leftarrow r + 1$;
    end for
  end for
  $\boldsymbol{U}_{M1}, \boldsymbol{U}_{M2} \leftarrow \boldsymbol{U}_M \left[: L^{(1)}(L^{(2)} - 1)\right], \boldsymbol{U}_M \left[L^{(2)} :\right]$; ▷ Last/First $L^{(2)}$ rows deleted
  $\boldsymbol{z}^{(1)}, \boldsymbol{W}^{(1)} \leftarrow$ Eigendecomposition $\left(\boldsymbol{U}_{M1}^{+} \boldsymbol{U}_{M2}\right)$;
  $\boldsymbol{f}^{(1)}, \boldsymbol{\eta}^{(1)} \leftarrow \left(f_{\text{sw}}^{(1)}/2\pi\right) \Im\left(\ln \boldsymbol{z}^{(1)}\right) + f_{\text{off}}^{(1)}, -f_{\text{sw}}^{(1)} \Re\left(\ln \boldsymbol{z}^{(1)}\right)$;
  $\boldsymbol{U}_{M\text{P}} \leftarrow \boldsymbol{P}\boldsymbol{U}_M$;
  $\boldsymbol{U}_{M\text{P}1}, \boldsymbol{U}_{M\text{P}2} \leftarrow \boldsymbol{U}_{M\text{P}} \left[: (L^{(1)} - 1)L^{(2)}\right], \boldsymbol{U}_{M\text{P}} \left[L^{(1)} :\right]$; ▷ Last/First $L^{(1)}$ rows deleted
  $\boldsymbol{G} \leftarrow {\boldsymbol{W}^{(1)}}^{-1} \boldsymbol{U}_{M\text{P}1}^{+} \boldsymbol{U}_{M\text{P}2} \boldsymbol{W}^{(1)}$;
  if all values in $\boldsymbol{f}^{(1)}$ are distinct **then** ▷ See Footnote ∗, Page 73
    $\boldsymbol{z}^{(2)} \leftarrow \text{diag}(\boldsymbol{G})$;
  else
    $R \leftarrow$ number of distinct frequencies in $\boldsymbol{f}^{(1)}$;
    for $r = 1, \cdots, R$ **do**
      $\mathbb{C}^{h_r} \ni \boldsymbol{z}_r^{(2)} \leftarrow$ Eigenvalues$(\boldsymbol{G}_r)^*$; ▷ See Equation 2.27.
    end for
  end if
  $\boldsymbol{f}^{(2)}, \boldsymbol{\eta}^{(2)} \leftarrow \left(f_{\text{sw}}^{(2)}/2\pi\right) \Im\left(\ln \boldsymbol{z}^{(2)}\right) + f_{\text{off}}^{(2)}, -f_{\text{sw}}^{(2)} \Re\left(\ln \boldsymbol{z}^{(2)}\right)$;

Continues on the next page...

*N.B. Here it is assumed that $\boldsymbol{G}$ is block-diagonal, in accordance with Equation 2.27. In practice $\boldsymbol{G}$ may not be block-diagonal, as whether it is or not is dependent on the ordering of the rows and columns in the matrix. A more explicit depiction of how $\boldsymbol{G}_r$ can be extracted from $\boldsymbol{G}$ is provided by Code Listing B.4 (Lines 89 to 116).

Continuation of **Algorithm A.1**.

$\boldsymbol{Z}_{\mathrm{L}}^{(2)} = \begin{bmatrix} \mathbf{1} & \boldsymbol{z}^{(2)} & \boldsymbol{z}^{(2)^2} & \dots & \boldsymbol{z}^{(2)^{L^{(2)}-1}} \end{bmatrix}^{\mathrm{T}}$;

$\boldsymbol{Z}_{\mathrm{R}}^{(2)} \leftarrow \begin{bmatrix} \mathbf{1} & \boldsymbol{z}^{(2)} & \boldsymbol{z}^{(2)^2} & \dots & \boldsymbol{z}^{(2)^{N^{(2)}-L^{(2)}}} \end{bmatrix}$;

$\boldsymbol{Z}_{\mathrm{D}}^{(1)} \leftarrow \operatorname{diag}\left(\boldsymbol{z}^{(1)}\right)$;

$\boldsymbol{E}_{\mathrm{L}} \leftarrow \begin{bmatrix} \boldsymbol{Z}_{\mathrm{L}}^{(2)} \\ \boldsymbol{Z}_{\mathrm{L}}^{(2)}\boldsymbol{Z}_{\mathrm{D}}^{(1)} \\ \vdots \\ \boldsymbol{Z}_{\mathrm{L}}^{(2)}\left[\boldsymbol{Z}_{\mathrm{D}}^{(1)}\right]^{L^{(1)}-1} \end{bmatrix}$;

$\boldsymbol{E}_{\mathrm{R}} \leftarrow \begin{bmatrix} \boldsymbol{Z}_{\mathrm{R}}^{(2)} & \boldsymbol{Z}_{\mathrm{D}}^{(1)}\boldsymbol{Z}_{\mathrm{R}}^{(2)} & \dots & \left[\boldsymbol{Z}_{\mathrm{D}}^{(1)}\right]^{N^{(1)}-L^{(1)}}\boldsymbol{Z}_{\mathrm{R}}^{(2)} \end{bmatrix}$;

$\boldsymbol{\alpha} \leftarrow \operatorname{diag}\left(\boldsymbol{E}_{\mathrm{L}}^{+}\boldsymbol{E}_{\boldsymbol{Y}}\boldsymbol{E}_{\mathrm{R}}^{+}\right)$;

$\boldsymbol{a}, \boldsymbol{\phi} \leftarrow |\boldsymbol{\alpha}|, \arctan\left(\frac{\Im(\boldsymbol{\alpha})}{\Re(\boldsymbol{\alpha})}\right)$;

$\boldsymbol{\theta}^{(0)} \leftarrow \begin{bmatrix} \boldsymbol{a}^{\mathrm{T}} & \boldsymbol{\phi}^{\mathrm{T}} & \left[\boldsymbol{f}^{(1)}\right]^{\mathrm{T}} & \left[\boldsymbol{f}^{(2)}\right]^{\mathrm{T}} & \left[\boldsymbol{\eta}^{(1)}\right]^{\mathrm{T}} & \left[\boldsymbol{\eta}^{(2)}\right]^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$;

return $\boldsymbol{\theta}^{(0)}$

end procedure

Algorithm A.2 The ST method for determining an update in NLP. This is equivalent to Algorithm 7.2 in [93].

procedure SteihaugToint($\boldsymbol{Y} \in \mathbb{C}^{N^{(1)} \times \cdots \times N^{(D)}}, \boldsymbol{\theta}^{(k)} \in \mathbb{R}^{2(1+D)M}, \Delta^{(k)} \in \mathbb{R}_{>0}$)
  $\boldsymbol{g} \leftarrow \nabla \mathcal{F}_{\phi}\left(\boldsymbol{\theta}^{(k)} \mid \boldsymbol{Y}\right)$; ▷ Grad vector: Equation 2.44a
  $\boldsymbol{H} \leftarrow \nabla^2 \mathcal{F}_{\phi}\left(\boldsymbol{\theta}^{(k)} \mid \boldsymbol{Y}\right)$; ▷ Hessian matrix, either exact: Equation 2.44b or approximate: Equation 2.45
  $\epsilon^{(k)} \leftarrow \min\left(1/2, \sqrt{\|\boldsymbol{g}\|}\right) \|\boldsymbol{g}\|$;
  $\boldsymbol{z}^{(0)} \leftarrow \boldsymbol{0} \in \mathbb{R}^{6M}$;
  $\boldsymbol{r}^{(0)} \leftarrow \boldsymbol{g}$;
  $\boldsymbol{d}^{(0)} \leftarrow -\boldsymbol{r}^{(0)}$;
  if $\left\|\boldsymbol{r}^{(0)}\right\| < \epsilon^{(k)}$ **then**
    return $\boldsymbol{z}^{(0)}$;
  end if
  for $j = \{0, 1, \cdots\}$ **do**
    if $\boldsymbol{d}^{(j)\mathrm{T}} \boldsymbol{H} \boldsymbol{d}^{(j)} \leq 0$ **then**
      Find τ such that $\boldsymbol{p}^{(k)} = \boldsymbol{z}^{(j)} + \tau \boldsymbol{d}^{(j)}$ minimises $\mathcal{F}_{\phi Q}\left(\boldsymbol{\theta}^{(k)} + \boldsymbol{p}^{(k)}\right)$, subject to $\left\|\boldsymbol{p}^{(k)}\right\| = \Delta^{(k)}$;
      return $\boldsymbol{p}^{(k)}$;
    end if
    $\alpha^{(j)} \leftarrow \dfrac{\boldsymbol{r}^{(j)\mathrm{T}} \boldsymbol{r}^{(j)}}{\boldsymbol{d}^{(j)\mathrm{T}} \boldsymbol{H} \boldsymbol{d}^{(j)}}$;
    $\boldsymbol{z}^{(j+1)} \leftarrow \boldsymbol{z}^{(j)} + \alpha^{(j)} \boldsymbol{d}^{(j)}$;
    if $\left\|\boldsymbol{z}^{(j+1)}\right\| < \epsilon^{(k)}$ **then**
      Find $\tau \in \mathbb{R}_{>0}$ such that $\boldsymbol{p}^{(k)} = \boldsymbol{z}^{(j)} + \tau \boldsymbol{d}^{(j)}$ satisfies $\left\|\boldsymbol{p}^{(k)}\right\| = \Delta^{(k)}$;
      return $\boldsymbol{p}^{(k)}$;
    end if
    $\boldsymbol{r}^{(j+1)} \leftarrow \boldsymbol{r}^{(j)} + \alpha^{(j)} \boldsymbol{H} \boldsymbol{d}^{(j)}$;
    if $\left\|\boldsymbol{r}^{(j+1)}\right\| < \epsilon^{(k)}$ **then**
      return $\boldsymbol{z}^{(j+1)}$;
    end if
    $\beta^{(j+1)} \leftarrow \dfrac{\boldsymbol{r}^{(j+1)\mathrm{T}} \boldsymbol{r}^{(j+1)}}{\boldsymbol{r}^{(j)\mathrm{T}} \boldsymbol{r}^{(j)}}$;
    $\boldsymbol{d}^{(j+1)} \leftarrow -\boldsymbol{r}^{(j+1)} + \beta^{(j+1)} \boldsymbol{d}^{(j)}$;
  end for
end procedure

Algorithm A.3 The filtering procedure for 2D data. The arguments $l/r^{(1/2)}_{\mathrm{I/N}}$ denote the left (l)/right (r) bound of the region of interest (I)/noise region (N) in dimension 1/2. These arguments should be members of the set $\{0, \cdots, N^{(1)} - 1/N^{(2)} - 1\}$.

procedure FILTER2D($\boldsymbol{Y}_{\cos} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}, \boldsymbol{Y}_{\sin} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}, l^{(1)}_{\mathrm{I}}, r^{(1)}_{\mathrm{I}}, l^{(2)}_{\mathrm{I}}, r^{(2)}_{\mathrm{I}}, l^{(1)}_{\mathrm{N}}, r^{(1)}_{\mathrm{N}}, l^{(2)}_{\mathrm{N}}, r^{(2)}_{\mathrm{N}}$)
  $\boldsymbol{Y}_{\mathrm{VE}} \leftarrow$ VIRTUALECHO2D $(\boldsymbol{Y}_{\cos}, \boldsymbol{Y}_{\sin})$;
  $\boldsymbol{S}_{\mathrm{VE}} \leftarrow \mathrm{FT}(\boldsymbol{Y}_{\mathrm{VE}})$;
  for $d = 1, 2$ **do**
    $c^{(d)} \leftarrow \left(l^{(d)}_{\mathrm{I}} + r^{(d)}_{\mathrm{I}}\right)/2$;
    $b^{(d)} \leftarrow r^{(d)}_{\mathrm{I}} - l^{(d)}_{\mathrm{I}}$;
    $\boldsymbol{g}^{(d)} \leftarrow$ SUPERGAUSSIAN1D $\left(2N^{(d)}, c^{(d)}, b^{(d)}\right)$; ▷ See Algorithm 2.3.
    $\boldsymbol{G} \leftarrow \boldsymbol{g}^{(1)} \otimes \boldsymbol{g}^{(2)}$;
  end for
  $\boldsymbol{S}_{\mathrm{N}} \leftarrow \boldsymbol{S}_{\mathrm{VE}}\left[l^{(1)}_{\mathrm{N}} : r^{(1)}_{\mathrm{N}} + 1, l^{(2)}_{\mathrm{N}} : r^{(2)}_{\mathrm{N}} + 1\right]$
  $\sigma^2 \leftarrow \mathrm{Var}(\boldsymbol{S}_{\mathrm{N}})$;
  $\boldsymbol{W}_{\sigma^2} \leftarrow \boldsymbol{0} \in \mathbb{R}^{2N^{(1)} \times 2N^{(2)}}$;
  for $n^{(1)} = 0, \cdots, 2N^{(1)} - 1$ **do**
    for $n^{(2)} = 0, \cdots, 2N^{(2)} - 1$ **do**
      $w_{\sigma^2, n^{(1)}, n^{(2)}} \leftarrow$ RANDOMSAMPLE $\left(\mathcal{N}\left(0, \sigma^2\right)\right)$;
    end for
  end for
  $\widetilde{\boldsymbol{S}}_{\mathrm{VE}} \leftarrow \boldsymbol{S}_{\mathrm{VE}} \odot \boldsymbol{G} + \boldsymbol{W}_{\sigma^2} \odot (\boldsymbol{1} - \boldsymbol{G})$;
  $\widetilde{\boldsymbol{Y}}_{\mathrm{VE}} \leftarrow \mathrm{IFT}\left(\widetilde{\boldsymbol{S}}_{\mathrm{VE}}\right)$;
  $\widetilde{\boldsymbol{Y}} \leftarrow \widetilde{\boldsymbol{Y}}_{\mathrm{VE}}\left[: N^{(1)}, : N^{(2)}\right]$;
  return $\widetilde{\boldsymbol{Y}}$;
end procedure

procedure VIRTUALECHO2D($\boldsymbol{Y}_{\cos} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}, \boldsymbol{Y}_{\sin} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}$)
  $\boldsymbol{\Psi}_{++} \leftarrow \Re(\boldsymbol{Y}_{\cos}) - \Im(\boldsymbol{Y}_{\sin}) + \mathrm{i}(\Im(\boldsymbol{Y}_{\cos}) + \Re(\boldsymbol{Y}_{\sin}))$;
  $\boldsymbol{\Psi}_{+-} \leftarrow \Re(\boldsymbol{Y}_{\cos}) + \Im(\boldsymbol{Y}_{\sin}) + \mathrm{i}(\Re(\boldsymbol{Y}_{\sin}) - \Im(\boldsymbol{Y}_{\cos}))$;
  $\boldsymbol{\Psi}_{-+} \leftarrow \Re(\boldsymbol{Y}_{\cos}) + \Im(\boldsymbol{Y}_{\sin}) + \mathrm{i}(\Im(\boldsymbol{Y}_{\cos}) - \Re(\boldsymbol{Y}_{\sin}))$;
  $\boldsymbol{\Psi}_{--} \leftarrow \Re(\boldsymbol{Y}_{\cos}) - \Im(\boldsymbol{Y}_{\sin}) - \mathrm{i}(\Im(\boldsymbol{Y}_{\cos}) + \Re(\boldsymbol{Y}_{\sin}))$;
  $\boldsymbol{Z} \leftarrow \boldsymbol{0} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}$
  $\boldsymbol{T}_1 \leftarrow \begin{bmatrix} \boldsymbol{\Psi}_{++} & \boldsymbol{Z} \\ \boldsymbol{Z} & \boldsymbol{Z} \end{bmatrix}$;
  $\boldsymbol{T}_2 \leftarrow \begin{bmatrix} \boldsymbol{Z} & \boldsymbol{\Psi}^{\leftrightarrow(2)}_{+-} \\ \boldsymbol{Z} & \boldsymbol{Z} \end{bmatrix}^{\circlearrowleft(2)}$;
  $\boldsymbol{T}_3 \leftarrow \begin{bmatrix} \boldsymbol{Z} & \boldsymbol{Z} \\ \boldsymbol{\Psi}^{\leftrightarrow(1)}_{-+} & \boldsymbol{Z} \end{bmatrix}^{\circlearrowleft(1)}$;
  $\boldsymbol{T}_4 \leftarrow \begin{bmatrix} \boldsymbol{Z} & \boldsymbol{Z} \\ \boldsymbol{Z} & \boldsymbol{\Psi}^{\leftrightarrow(1,2)}_{--} \end{bmatrix}^{\circlearrowleft(1,2)}$;
  $\boldsymbol{Y}_{\mathrm{VE}} \leftarrow \boldsymbol{T}_1 + \boldsymbol{T}_2 + \boldsymbol{T}_3 + \boldsymbol{T}_4$;
  for $n^{(1)} = 0, \cdots, 2N^{(1)} - 1$ **do**
    $y_{\mathrm{VE}, n^{(1)}, 0} \leftarrow y_{\mathrm{VE}, n^{(1)}, 0}/2$;
  end for
  for $n^{(2)} = 0, \cdots, 2N^{(2)} - 1$ **do**
    $y_{\mathrm{VE}, n^{(2)}, 0} \leftarrow y_{\mathrm{VE}, n^{(2)}, 0}/2$;
  end for
  return $\boldsymbol{Y}_{\mathrm{VE}}$;
end procedure

Appendix B

Python Listings

Presented here are listings of Python implementations for a number of the routines described in this text. These can be thought of minimalist versions of code present in NMR-EsPy package. These listings require Python version 3.8 or higher, and also require the `numpy` and `scipy` packages to be installed from the Python Package Index (PyPI). As a preliminary, Code Listing B.1 outlines all the necessary imports for the subsequent listings in this chapter.

Code Listing B.1: The required imports for the subsequent listings in this chapter. The `itertools.product` function is used in the MMEMPM routine (Code Listing B.4). The imports from the `typing` module are used to annotate what the expected argument- and return-types are. The `numpy` and `scipy` modules are ubiquitous, 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
```

B.1 Matrix Pencil Methods

B.1.1 MDL

Code Listing B.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 20; see Footnote ‖ in Section 2.2.3).

```
1 def mdl(sigma: np.ndarray, N: int, L: int) -> int:
2     """Compute the Minimum Description Length
3
4     Parameters
```

```
    ----------
    sigma
        Vector of singular values associated with the data matrix.
    N
        Number of points the signal comprises.
    L
        Pencil parameter.
    """
    mdl_vec = np.zeros(L)
    for k in range(L):
        mdl_vec[k] = (
            -N * np.sum(np.log(sigma[k:])) +
            N * (L - k) * np.log(np.sum(sigma[k:]) / (L - k)) +
            k * np.log(N) * (2 * L - k) / 2
        )
    M = sp.signal.argrelextrema(mdl_vec, np.less)[0][0]
    return M
```

B.1.2 MPM

Code Listing B.3: The MPM for 1D FID estimation, with the option of predicting the model order using the MDL.

```
# mpm.py
# Simon Hulse
# simonhulse@protonmail.com
# Last Edited: Tue 09 Jan 2024 17:38:34 EST

def mpm(y: np.ndarray, sw: float, offset: float, M: int = 0) -> np.array:
    """Matrix Pencil Method for estimating a 1D FID.

    Parameters
    ----------
    y
        FID.
    sw
        Spectral width (Hz).
    offset
        Transmitter offset (Hz).
    M
        Number of oscillators. If 0, this is estimated using the MDL.
    """
    N = y.size  # N
    L = N // 3  # L: pencil parameter
    norm = np.linalg.norm(y)  # ||y||
```

```
    y /= norm  # Y/‖y‖

    # SVD of H_y
    col = Y[: N - L]  # First column of H_y
    row = Y[N - L - 1 :]  # Last row of H_y
    Hy = sp.linalg.hankel(col, row)
    U, sigma, _ = np.linalg.svd(Hy)  # Determine σ and U

    # Optional model order selection
    if M == 0:
        M = mdl(sigma, N, L)

    # Eigendecomposition of U_M1^+ U_M2 to get signal poles, z
    UM = U[:, :M]
    UM1 = Um[:-1, :]
    UM2 = Um[1:, :]
    UM1inv = np.linalg.pinv(UM1)
    UM1invUM2 = UM1inv @ UM2
    z, _ = np.linalg.eig(UM1invUM2)

    # Determine complex amplitudes, α
    Z = np.power.outer(z, np.arange(N)).T  # Vandermonde pole matrix
    alpha = np.linalg.pinv(Z) @ y

    # Determine a, ϕ, f, and η, and store in M × 4 array
    M = z.size
    theta = np.zeros((M, 4), dtype="float64")
    theta[:, 0] = np.abs(alpha) * norm
    theta[:, 1] = np.arctan2(np.imag(alpha), np.real(alpha))
    theta[:, 2] = (sw / (2 * np.pi)) * np.imag(np.log(z)) + offset
    theta[:, 3] = -sw * np.real(np.log(z))
    theta = theta[np.argsort(theta[:, 2])]  # order by f

    # Remove oscillators with negative damping factors
    neg_damp_idx = np.nonzero(damp < 0.)[0]
    theta = np.delete(theta, neg_damp_idx, axis=0)

    return theta
```

B.1.3 MMEMPM

Code Listing B.4: The MMEMPM for 2D hypercomplex FID estimation. Due to the very large size of the Hankel matrix $\boldsymbol{E}_{\boldsymbol{Y}}$, 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; see Footnote ††† in Section 2.4.1).

```python
# mmempm.py
# Simon Hulse
# simonhulse@protonmail.com
# Last Edited: Tue 09 Jan 2024 17:38:53 EST

def mmempm(
    Y: np.ndarray, sw1: float, sw2: float,
    off1: float, off2: float, M: int = 0,
) -> np.ndarray:
    """Modified Matrix Enhancment Matrix Pencil Method for estimating a
   2D FID.

    Parameters
    ----------
    Y
        FID.
    sw1
        Spectral width in first (indirect) dimension (Hz).
    sw2
        Spectral width in second (direct) dimension (Hz).
    off1
        Transmitter offset in first dimension (Hz).
    off2
        Transmitter offset in second dimension (Hz).
    M
       Number of oscillators. If 0, this is estimated by applying the MDL
        to the first direct dimension FID (i.e. Y[0])
    """
    N1, N2 = Y.shape  # N^(1), N^(2)
    L1, L2 = N1 // 2, N2 // 2  # L^(1), L^(2): pencil parameters
    norm = np.linalg.norm(Y)  # ||Y||
    Y /= norm  # Y/||Y||

    # Optional model orer selection on first direct-dimension signal
    if M == 0:
        L_mdl = N2 // 3
        y_mdl = Y[0]
        col_mdl = y_mdl[: N2 - L_mdl]
```

```
        row_mdl = y_mdl[N2 - L_mdl - 1 :]
        Hy_mdl = sp.linalg.hankel(col_mdl, row_mdl)
        _, sigma_mdl, _ = np.linalg.svd(Hy_mdl)
        M = mdl(sigma_mdl, N2, L_mdl)

    # Construct block Hankel E_Y
    row_size = L2
    col_size = N2 - L2 + 1
    EY = np.zeros(
        (L1 * L2, (N1 - L1 + 1) * (N2 - L2 + 1)),
        dtype="complex",
    )
    for n1 in range(N1):
        # Construct H_{Y,n^(1)}, and assign to appropriate positions in E_Y
        col = Y[n1, :L2]
        row = Y[n1, L2 - 1:]
        HYn1 = sp.linalg.hankel(col, row)
        for n in range(n1 + 1):
            r, c = n, n1 - n
            if r < L1 and c < N1 - L1 + 1:
                EY[
                    r * row_size : (r + 1) * row_size,
                    c * col_size : (c + 1) * col_size
                ] = HYn1

    # SVD of E_Y to get U_M
    EY = sp.sparse.csr_matrix(EY)  # Make E_Y sparse
    UM, *_ = sp.sparse.linalg.svds(EY, k=M)

    # Eigendecomposition of U_{M1}^+ U_{M2} to get poles z^(1) and matrix W^(1)
    UM1 = UM[: L1 * (L2 - 1)]  # Last L^(2) rows deleted
    UM2 = UM[L2:]  # First L^(2) rows deleted
    z1, W1 = np.linalg.eig(np.linalg.pinv(UM1) @ UM2)

    # Construct permutation matrix P
    P = np.zeros((L1 * L2, L1 * L2))
    r = 0
    for l2 in range(L2):
        for l1 in range(L1):
            c = l1 * L2 + l2
            P[r, c] = 1
            r += 1

    # Compute G
    UMP = P @ UM
```

```
    UMP1 = UMP[: (L1 - 1) * L2]  # Last L^(1) rows deleted
    UMP2 = UMP[L1:]  # First L^(1) rows deleted
    G = np.linalg.inv(W1) @ np.linalg.pinv(UMP1) @ UMP2 @ W1
    z2 = np.diag(G).copy()  # Provisional signal poles z^(2)

    # Check for and deal with similar values in f^(1)
    freq1 = (0.5 * sw1 / np.pi) * np.imag(np.log(z1)) + off1
    threshold = sw1 / N1  # f_sw^(1)/N^(1)
    groupings = {}
    # Iterate through values in f^(1) and group those with
    # similar frequencies together
    for idx, f1 in enumerate(freq1):
        assigned = False
        for group_f1, indices in groupings.items():
            if np.abs(f1 - group_f1) < threshold:
                indices.append(idx)
                n = len(indices)
                indices = sorted(indices)
                # Get new mean value of the group
                new_group_f1 = (n * group_f1 + f1) / (n + 1)
                groupings[new_group_f1] = groupings.pop(group_f1)
                assigned = True
                break
        if not assigned:
            groupings[f1] = [idx]

    for indices in groupings.values():
        n = len(indices)
        if n != 1:
            Gr_slice = tuple(zip(*product(indices, repeat=2)))
            Gr = G[Gr_slice].reshape(n, n)
            new_group_z2, _ = np.linalg.eig(Gr)
            z2[indices] = new_group_z2

    # Compute complex amplitudes α
    ZL2 = np.power.outer(z2, np.arange(L2)).T  # Z_L^(2)
    ZR2 = np.power.outer(z2, np.arange(N2 - L2 + 1))  # Z_R^(2)
    Z1D = np.diag(z1)  # Z_D^(1)
    EL = np.zeros((L1 * L2, M), dtype="complex")
    Z2LZ1D = ZL2
    for i in range(L1):
        EL[i * row_size : (i + 1) * row_size] = Z2LZ1D
        Z2LZ1D = Z2LZ1D @ Z1D
    ER = np.zeros((M, (N1 - L1 + 1) * (N2 - L2 + 1)), dtype="complex")
    Z1DZ2R = ZR2
```

```
129     for i in range(N1 - L1 + 1):
130         ER[:, i * col_size : (i + 1) * col_size] = Z1DZ2R
131         Z1DZ2R = Z1D @ Z1DZ2R
132     alpha = np.diag(np.linalg.pinv(EL) @ EY @ np.linalg.pinv(ER))
133 
134     # Determine a, ϕ, f(1), f(2), η(1), and η(2), and store in M×6 array
135     theta = np.zeros((M, 6), dtype="float64")
136     theta[:, 0] = np.abs(alpha) * norm  # a
137     theta[:, 1] = np.arctan2(np.imag(alpha), np.real(alpha))  # ϕ
138     theta[:, 2] = freq1  # f(1) (already computed)
139     theta[:, 3] = (0.5 * sw2 / np.pi) * np.imag(np.log(z2)) + off2  # f(2)
140     theta[:, 4] = -sw1 * np.real(np.log(z1))  # η(1)
141     theta[:, 5] = -sw2 * np.real(np.log(z2))  # η(2)
142     theta = theta[np.argsort(theta[:, 3])]  # order by f(2)
143 
144     # Remove oscillators with negative damping factors
145     neg_damp_idx1 = list(np.nonzero(damp1 < 0.)[0])
146     neg_damp_idx2 = list(np.nonzero(damp2 < 0.)[0])
147     neg_damp_idx = list(set(neg_damp_idx1 + neg_damp_idx2))
148     theta = np.delete(theta, neg_damp_idx, axis=0)
149 
150     return theta
```

B.2 Non-Linear Programming

B.2.1 Trust Region Algorithm

Code Listing B.5: The 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 103–109).

```
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), assuming a 1D FID.
12    function_factory
13        Object for computing the objective, gradient and Hessian.
```

```
    args
        Extra arguments required for computing the objective and its
        derivatives.

    Returns
    -------
    theta
        Parameter vector at termination.
    errors
        Errors associated with parameter vector.
    negative_amps
        Flag indicating whether or not termination occurred because
        negative amplitudes were detected.
    """
    theta = theta0
    M = theta.shape[0] // 4
    factory = function_factory(theta, *args)

    # Define relevent parameters
    # These have been hard-coded, though in NMR-EsPy they are all
    # configurable.
    eta = 0.15,
    initial_trust_radius = 0.1 * factory.gradient_norm
    max_trust_radius = 16 * initial_trust_radius
    epsilon = 1.e-8,
    max_iterations = 200,
    check_neg_amps_every = 25,

    k = 0
    while True:
        # === Steihaug-Toint ===
        epsi = min(0.5, np.sqrt(factory.gradient_norm)) * \
                factory.gradient_norm
        z = np.zeros_like(theta)
        r = factory.gradient
        d = -r
        while True:
            Bd = factory.hessian @ d
            dBd = d.T @ Bd
            if dBd <= 0:
                ta, tb = get_boundaries(z, d, trust_radius)
                pa = z + ta * d
                pb = z + tb * d
                p = min(factory.model(pa), factory.model(pb))
                hits_boundary = True
```

```
59                 break
60             r_sq = r.T @ r
61             alpha = r_sq / dBd
62             z_next = z + alpha * d
63             if sp.linalg.norm(z_next) >= trust_radius:
64                 _, tb = get_boundaries(z, d, trust_radius)
65                 p = z + tb * d
66                 hits_boundary = True
67                 break
68             r_next = r + alpha * Bd
69             r_next_sq = r_next.T @ r_next
70             if np.sqrt(r_next_sq) < epsi:
71                 hits_boundary = False
72                 p = z_next
73                 break
74             beta_next = r_next_sq / r_sq
75             d_next = -r_next + beta_next * d
76             z = z_next
77             r = r_next
78             d = d_next
79
80         # Assess effectiveness of update
81         predicted_value = factory.model(p)
82         theta_proposed = theta + p
83         factory_proposed = function_factory(theta_proposed, *args)
84         actual_reduction = factory.objective - factory_proposed.objective
85         predicted_reduction = factory.objective - predicted_value
86         if predicted_reduction <= 0:
87             # No improvement could be found: terminate
88             negative_amps = False
89             break
90         rho = actual_reduction / predicted_reduction
91         if rho < 0.25:
92             # Quadratic model perfomring poorly: reduce TR
93             trust_radius *= 0.25
94         elif rho > 0.75 and hits_boundary:
95             # Quadratic model perfomring well: increase TR
96             trust_radius = min(2 * trust_radius, max_trust_radius)
97         if rho > eta:
98             # Accept update and increment iteration count
99             theta = theta_proposed
100            factory = factory_proposed
101            k += 1
102
103        # Check for termination criteria
```

```
104         if (k % check_neg_amps_every == 0):
105             neg_amps = np.where(theta[amp_slice] <= 0)[0]
106             if neg_amps.size > 0:
107                 # Negative amps found
108                 negative_amps = True
109                 break
110 
111         if factory.gradient_norm < epsilon:
112             # Convergence
113             negative_amps = False
114             break
115 
116         if k == max_iterations:
117             # Maximum allowed iterations reached
118             negative_amps = False
119             break
120 
121     # Routine terminated: compute errors and return parameter array
122     errors = np.sqrt(
123         factory.objective *
124         np.abs(np.diag(np.linalg.inv(factory.hessian)))
125     )
126     return theta, errors, negative_amps
127 
128 
129 def get_boundaries(
130     z: np.ndarray, d: np.ndarray, trust_radius: float
131 ) -> Tuple[float, float]:
132     """Determine the intersections of the search direction and the trust
133     region."""
134     a = d.T @ d
135     b = 2 * z.T @ d
136     c = (z.T @ z) - (trust_radius ** 2)
137     aux = b + np.copysign(
138         np.sqrt(b * b - 4 * a * c),
139         b,
140     )
141     return sorted([-aux / (2 * a), -(2 * c) / aux])
```

B.2.2 Computing $\mathcal{F}_\phi$, $\nabla\mathcal{F}_\phi$, and $\nabla^2\mathcal{F}_\phi$

Code Listing B.6: Code for generating the fidelity, gradient and Hessian as part of the non-linear programming routine for 1D FIDs. 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 objective and its derivatives are computed and cached (memoised), such that the next time a quantity is requested, the cached result is returned, rather than requiring the expensive function to be re-run. `FunctionFactory1D` (Lines 45–47) inherits from the base class, for specific use in 1D estimation. For the equivalent code for 2D estimation, and/or for computing the approximated Hessian instead, the reader is directed to the `nlp/_funcs.py` module in the NMR-EsPy package.

```
1  # obj_grad_hess.py
2  # Simon Hulse
3  # simonhulse@protonmail.com
4  # Last Edited: Fri 19 Jan 2024 17:56:53 EST
5
6  class FunctionFactory:
7      """Object which computes and memoises the objective, gradient and
8      Hessian for a given set of parameters."""
9      def __init__(self, theta: np.ndarray, fun: callable, *args) -> None:
10         self.theta = theta
11         self.fun = fun
12         self.obj = None
13         self.grad = None
14         self.hess = None
15         self.args = args
16
17     def _compute_if_needed(self):
18         """Compute the obj, grad and Hess if they haven't been yet"""
19         if self.obj is None:
20             self.obj, self.grad, self.hess = self.fun(self.theta,
                ↪  *self.args)
21
22     def model(self, p) -> float:
23         return self.objective + self.gradient @ p + 0.5 * (p.T @
            ↪  self.hessian @ p)
24
25     @property
26     def objective(self) -> float:
27         self._compute_if_needed()
28         return self.obj
29
30     @property
31     def gradient(self) -> np.ndarray:
32         self._compute_if_needed()
```

```
33          return self.grad
34
35      @property
36      def gradient_norm(self) -> float:
37          return sp.linalg.norm(self.gradient)
38
39      @property
40      def hessian(self) -> np.ndarray:
41          self._compute_if_needed()
42          return self.hess
43
44
45  class FunctionFactory1D(FunctionFactory):
46      def __init__(self, theta: np.ndarray, *args) -> None:
47          super().__init__(theta, obj_grad_hess_1d, *args)
48
49
50  def obj_grad_hess_1d(
51      theta: np.ndarray, *args: Tuple[int, int, np.ndarray],
52  ) -> Tuple[float, np.ndarray, np.ndarray]:
53      Y, tp = args  # Unpack args: FID, timepoints
54      N = Y.shape[0]
55      M = theta.shape[0] // 4
56      X_per_osc = np.exp(
57          np.outer(tp, (2j * np.pi * theta[2 * M : 3 * M] - theta[3 * M :]))
58      ) * (theta[:M] * np.exp(1j * theta[M : 2 * M]))
59
60      # First partial derivatives
61      d1 = np.zeros((N, 4 * M), dtype="complex")
62      d1[:, :M] = X_per_osc / theta[:M]  # ∂x/∂a_m
63      d1[:, M : 2 * M] = 1j * X_per_osc  # ∂x/∂ϕ_m
64      d1[:, 2 * M : 3 * M] = \  # ∂x/∂f_m
65          np.einsum("ij,i->ij", X_per_osc, 2j * np.pi * tp)
66      d1[:, 3 * M :] = np.einsum("ij,i->ij", X_per_osc, -tp)  # ∂x/∂η_m
67
68      # Second partial derivatives
69      d2 = np.zeros((N, 10 * M), dtype="complex")
70      # Note ∂²x/∂a_m² = 0 always.
71      d2[:, M : 2 * M] = 1j * d1[:, M : 2 * M]  # ∂²x/∂ϕ_m²
72      d2[:, 2 * M : 3 * M] = \  # ∂²x/∂f_m²
73          np.einsum("ij,i->ij", d1[:, 2 * M : 3 * M], 2j * np.pi * tp)
74      d2[:, 3 * M : 4 * M] = np.einsum("ij,i->ij", d2[: 3 * M :], -tp)  #
        ↪ ∂²x/∂η_m²
75      d2[:, 4 * M : 5 * M] = 1j * d1[:, :M]  # ∂²x/∂a_m∂ϕ_m
76      d2[:, 5 * M : 6 * M] = 1j * d1[:, 2 * M : 3 * M]  # ∂²x/∂ϕ_m∂f_m
```

```
77      d2[:, 6 * M : 7 * M] = \  # ∂²x/∂f_m∂η_m
78          np.einsum("ij,i->ij", d2[: 2 * M : 3 * M], -tp)
79      d2[:, 7 * M : 8 * M] = d1[:, 2 * M : 3 * M] / theta[:M]  # ∂²x/∂a_m∂f_m
80      d2[:, 8 * M : 9 * M] = 1j * d1[:, 3 * M : 4 * M]  # ∂²x/∂ϕ_m∂η_m
81      d2[:, 9 * M :] = d1[:, 3 * M : 4 * M] / theta[:M]  # ∂²x/∂a_m∂η_m
82
83      X = np.einsum("ij->i", X_per_osc)
84      Y_minus_X = Y - X
85
86      # Objective
87      obj = np.real(Y_minus_X.conj().T @ Y_minus_X)
88
89      # Gradient
90      grad = -2 * np.real(Y_minus_X.conj().T @ d1)
91
92      # Hessian
93      hess = np.zeros((4 * M, 4 * M))
94      # Compute non-zero elements of term ② of the Hessian,
95      # and assign to upper right triangle.
96     hess_term_2 = -2 * np.real(np.einsum("ji,j->i", d2.conj(), Y_minus_X))
97      term_2_row_idxs = []
98      term_2_col_idxs = []
99      for i in range(4):
100         rows, cols = _diag_idx(4 * M, i * M)
101         term_2_row_idxs.extend(rows)
102         term_2_col_idxs.extend(cols)
103     term_2_idxs = (term_2_row_idxs, term_2_col_idxs)
104     hess[term_2_idxs] = hess_term_2
105     # Add the transpose to fill left botton triangle
106     # Halve the main diagonal before transposing to avoid doubling
107     main_diag = _diag_idx(4 * M, 0)
108     hess[main_diag] *= 0.5
109     hess += hess.T
110     # Compute term ① of the Hessian
111     hess_term_1 = 2 * np.real(np.einsum("ki,kj->ij", d1.conj(), d1))
112     hess += hess_term_1
113
114     # Phase variance and derivatives
115     phi = theta[M : 2 * M]
116     cos_phi = np.cos(phi)
117     cos_sum = np.sum(cos_phi)
118     sin_phi = np.sin(phi)
119     sin_sum = np.sum(sin_phi)
120     R = np.sqrt(cos_sum ** 2 + sin_sum ** 2)
121     pv_obj = 1 - (R / M)
```

```
122     pv_grad = (sin_phi * cos_sum - cos_phi * sin_sum) / (M * R)
123     x = (sin_phi * cos_sum) - (cos_phi * sin_sum)
124     term_1 = np.outer(x, x) / (R ** 2)
125     phi_array = np.zeros((M, M))
126     phi_array[:] = phi
127     term_2 = -np.cos(phi_array.T - phi_array)
128     pv_hess = term_1 + term_2
129     pv_hess[np.diag_indices(M)] += cos_phi * cos_sum + sin_phi * sin_sum
130     pv_hess /= M * R
131     obj += pv_obj  # \mathcal{F}_\phi(\theta)
132     grad[M : 2 * M] += pv_grad  # \nabla\mathcal{F}_\phi(\theta)
133     hess[M : 2 * M, M : 2 * M] += pv_hess  # \nabla^2\mathcal{F}_\phi(\theta)
134
135     return obj, grad, hess
136
137
138 def _diag_idx(size: int, disp: int) -> Tuple[np.ndarray, np.ndarray]:
139     """Return the indices of an array's kth diagonal.
140
141     Parameters
142     ----------
143     size
144         The size of the sqaure matrix.
145     disp
146         Displacement from the main diagonal.
147
148     Returns
149     -------
150     rows
151         0-axis coordinates of indices.
152     cols
153         1-axis coordinates of indices.
154     """
155     rows, cols = np.diag_indices(size)
156     if disp < 0:
157         return rows[-disp:], cols[:disp]
158     elif disp > 0:
159         return rows[:-disp], cols[disp:]
160     else:
161         return rows, cols
```

B.2.3 The Main Routine

Code Listing B.7: Code which runs the NLP routine for FID estimation. The routine consists of running the ST algorithm (Code Listing B.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.

```python
# nlp_routine.py
# Simon Hulse
# simonhulse@protonmail.com
# Last Edited: Tue 09 Jan 2024 17:39:49 EST

def nlp(
    y: np.ndarray,
    sw: float,
    offset: float,
    theta0: np.ndarray,
) -> Tuple[np.ndarray, np.ndarray]:
    """Nonlinear programming routine for 1D FID estimation.

    Parameters
    ----------
    y
        FID.
    sw
        Spectral width (Hz).
    offset
        Tranmitter offset (Hz).
    theta0
        Initial guess, of shape (M, 4)

    Returns
    -------
    theta
        Parameter estimate.
    errors
        Errors associated with theta.
    """
    norm = np.linalg.norm(y)
    y /= norm
    N = y.shape[0]
    M = theta0.shape[0]
    # Flatten parameter array: Fortran (column-wise) ordering
    theta0_vec = theta0.flatten(order="F")
    theta0_vec[:M] /= norm  # Normalise amplitudes
```

```
    # Remove transmitter offset from frequencies
    theta0_vec[2 * M : 3 * M] -= offset

    # Extra arguments needed to compute the objective, grad, and Hessian:
    # FID and timepoints sampled
    opt_args = [y, np.linspace(0, float(N - 1) / sw, N)]

    while True:
        theta_vec, errors_vec, negative_amps = trust_ncg(
            theta0=theta0_vec,
            function_factory=FunctionFactoryGaussNewton1D,
            args=opt_args,
        )

        if negative_amps:
            # Negative amps exist: remove these
            negative_idx = list(np.where(theta_vec[:M] <= 0.)[0])
            slice_ = []
            for idx in range(negative_idx):
                slice_.extend([i * M + idx for i in range(4)])
            theta_vec = np.delete(theta_vec, slice_)
            M -= len(negative_idx)  # New model order

        else:
            # No negative amps: routine complete
            break

    # Reshape parameter array back to (M, 4)
    theta = theta_vec.reshape((M, 4), order="F")
    errors = errors_vec.reshape((M, 4), order="F")
    errors_vec /= N - 1
    theta[:, 2] += offset  # Re-add transmitter offset to frequencies
    theta[:, 0] *= norm  # Re-scale amplitudes
    errors[:, 0] *= norm
    theta[:, 1] = (theta[:, 1] + np.pi) % (2 * np.pi) - np.pi  # Wrap
    ↪ phases

    return theta, errors
```

B.3 CUPID

B.3.1 Assigning Multiplet Structures

Code Listing B.8: Code which performs multiplet assignment as part of CUPID.

```python
def predict_multiplets(
    params: np.ndarray,
    thold: float,
) -> Dict[float, Iterable[int]]:
    """
    Parameters
    ----------
    params
        Estimated parameter array with shape (M, 6) such that each row
        provides the parameters of a particular oscillator, in the order
        [a, φ, f1, f2, η1, η2].
    thold
        Frequency threshold ε > 0.

    Returns
    -------
    A dictionary with the multiplet central frequencies as keys and
    oscillator indices as values
    """
    multiplets = {}
    for m, osc in enumerate(params):
        assigned = False
        f1, f2 = osc[2], osc[3]  # Extract f^(1) and f^(2)
        fc = osc[3] - osc[2]  # Central frequency for oscillator: f_m^(2) − f_m^(1)
        # Check whether central frequency agrees with any
        # alread-established multiplet grouping
        for fmp in multiplets:
            if fmp - thold < fc < fmp + thold:
                # Update grouping:
                # Add m to list of oscillators
                # Update central frequency: mean of all oscillators
                ms = multiplets.pop(fmp)
                ms.append(m)
                mp_size = len(ms)
                new_fmp = ((mp_size - 1) * fmp + fc) / mp_size
                multiplets[new_fmp] = ms
                assigned = True
                break
```

```
39          # No match: create new multiplet group
40          if not assigned:
41              multiplets[fc] = [m]
42
43      return multiplets
```

Appendix C

Information on Datasets and Results

C.1 Simulated datasets

C.1.1 Spinach Spin Systems

Many of the simulated datasets presented in this work were generated using the Spinach Matlab package [95]. 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 initialising a Matlab engine from Python [186], 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 relevant information about the simulation.

Table C.1 provides a specification of the chemical shifts and scalar couplings that made up the spin systems used for the Spinach simulations considered. The relevant the spin systems were constructed as follows:.

"Five Multiplets" The spin systems used to generate the "five multiplets" inversion recovery datasets featured 8 spins (^{1}H), comprising a subset of 5 "estimated spins" and another subset of 3 "coupling spins":

- The estimated spins were each assigned chemical shifts randomly sampled from the distribution $\mathcal{U}(-0.1\,\text{ppm}, 0.1\,\text{ppm})$.
- The coupling spins were assigned non-zero J-couplings to each of the estimated spins, with the couplings randomly sampled from $\mathcal{U}(-20\,\text{Hz}, 20\,\text{Hz})$. They were given chemical shifts $\gg 0.15$ ppm, thus ensuring the spin systems abides by the weak coupling regime.

As a result, a region of the dataset, centred at 0 ppm, would feature 5 ddd multiplets. A further constraint on the shifts and couplings was applied, to ensure that the 40 (5×2^3) signals generated

by the 5 estimated spins would have sufficiently separated frequencies, such that they could feasibly be resolved by estimation. Each spin was assigned longitudinal and transverse relaxation times, randomly sampled from $T_1 \sim \mathcal{U}(1\,\mathrm{s}, 5\,\mathrm{s})$, and $T_2 \sim \mathcal{U}(0.2\,\mathrm{s}, 0.6\,\mathrm{s})$. Relaxation of the spin system was defined to adhere to the *extended T_1/T_2 approximation* [137]. Under this model, all longitudinal states associated with a given spin relax exponentially 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.

"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 $\mathcal{U}(-0.03\,\mathrm{ppm}, 0.03\,\mathrm{ppm})$. J-couplings between the estimated and coupling spins were sampled from $\mathcal{U}(-10\,\mathrm{Hz}, 10\,\mathrm{Hz})$. The system was not subjected to any relaxation.

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 [187: Appendix 5]. The system was not subjected to any relaxation.

Table C.1: 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

Continues on next page...

Spin	δ (ppm)	ω_0 (Hz)	J (Hz)	T_1 (s)
(C)	-3.33×10^{-2}	-16.66	(F): 8.758, (G): 16.689, (H): 12.956	4.766
(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	–
Strychinine				
(A)	8.092	4046.0	(C): 0.230, (D): 0.980, (B): 7.900	–
(B)	7.255	3627.5	(C): 1.080, (D): 7.440, (A): 7.900	–
(C)	7.167	3583.5	(D): 7.490, (B): 1.080, (A): 0.230	–
(D)	7.098	3549.0	(C): 7.490, (B): 7.440, (A): 0.980	–
(E)	5.915	2957.5	(M): 0.470, (K): 1.790, (G): 7.000, (H): 6.100	–

Continues on next page...

Spin	δ (ppm)	ω_0 (Hz)	J (Hz)	T_1 (s)
(F)	4.288	2144.0	(N): 3.340, (Q): 8.470, (V): 3.300	–
(G)	4.148	2074.0	(E): 7.000, (H): −13.800	–
(H)	4.066	2033.0	(E): 6.100, (G): −13.800	–
(I)	3.963	1981.5	(R): 4.330, (U): 2.420	–
(J)	3.860	1930.0	(V): 10.410	–
(K)	3.716	1858.0	(M): 1.610, (P): −14.800, (E): 1.790	–
(L)	3.219	1609.5	(T): 5.500, (S): 3.200, (O): −13.900	–
(M)	3.150	1575.0	(V): 3.290, (R): 4.110, (U): 1.960, (K): 1.610, (E): 0.470	–
(N)	3.132	1566.0	(Q): −17.340, (F): 3.340	–
(O)	2.878	1439.0	(T): 7.200, (S): 10.700, (L): −13.900	–
(P)	2.745	1372.5	(K): −14.800	–
(Q)	2.670	1335.0	(N): −17.340, (F): 8.470	–
(R)	2.360	1180.0	(M): 4.110, (U): −14.350, (I): 4.330	–
(S)	1.890	945.0	(T): −13.900, (L): 3.200, (O): 10.700	–
(T)	1.890	945.0	(S): −13.900, (L): 5.500, (O): 7.200	–
(U)	1.462	731.0	(M): 1.960, (R): −14.350, (I): 2.420	–
(V)	1.276	638.0	(J): 10.410, (F): 3.300, (M): 3.290	–

C.1.2 2DJ Datasets

The simulated datasets of 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 C.2.

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 C.2: The experiment parameters used for 2DJ simulations run using Spinach.

A 2DJ pulse sequence simulation does not ship with Spinach. Therefore, an in-house one was written, which is given by Code Listing C.1.

Code Listing C.1: A MATLAB function for simulating 2DJ experiments using SPINACH.

```
function fid = jres_seq(spin_system, parameters, H, R, K)
    % Compose Liouvillian
    L = H + 1i * R + 1i * K; clear('H', 'R', 'K');
    % Coherent evolution timestep
    d1 = 1 / parameters.sweep(1);
    d2 = 1 / parameters.sweep(2);
    % Number of points
    pts1 = parameters.npoints(1);
    pts2 = parameters.npoints(2);
    % Targeted nucleus
    nuc = parameters.spins{1}
    % Initial state
    rho = state(spin_system, 'Lz', nuc);
    % Detection state
    coil = state(spin_system, 'L+', nuc);
    % Get the pulse operators
    Lp = operator(spin_system, 'L+', nuc);
    Lx = (Lp + Lp') / 2;
    % First pulse: 90 about x
    rho = step(spin_system, Lx, rho, pi / 2);
    % First half of t1 evolution
    rho = evolution(spin_system, L, [], rho, d1 / 2, pts1 - 1,
     ↪ 'trajectory');
    % Select "-1" coherence
    rho = coherence(spin_system, rho, {{nuc, -1}});
    % Second pulse: 180 about x
    rho = step(spin_system, Lx, rho, pi);
    % Select "+1" coherence
    rho = coherence(spin_system, rho, {{nuc, +1}});
    % Second half of t1 evolution
    rho = evolution(spin_system, L, [], rho, d1 / 2, pts1 - 1, 'refocus');
    % Run the F2 evolution
    fid = evolution(spin_system, L, coil, rho, d2, pts2 - 1,
     ↪ 'observable');
end
```

Exponential apodisation was applied to the FIDs after simulation (recall that relaxation phenomena were switched off for these spin systems). The exponential used was such that the final point in a given dimension of the dataset would be attenuated by a factor of 1000.

C.1.3 Inversion Recovery Datasets

The simulated datasets for the "five multiplets" results in Section 3.2.5 was generated using NMR-EsPy's `EsimatorInvRec.new_spinach` method, with parameters used in the simulation provided by Table C.3. 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
$f_{\text{bf}}^{(1)}$ (MHz)	500
$f_{\text{off}}^{(1)}$ (Hz)	2500
$f_{\text{off}}^{(1)}$ (ppm)	5
$f_{\text{sw}}^{(1)}$ (Hz)	5000
$f_{\text{sw}}^{(1)}$ (ppm)	10
$N^{(1)}$	16384

Table C.3: Parameters for the "five multiplets" inversion recovery simulation run using Spinach.

C.2 Experimental datasets

C.2.1 1D Datasets

	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 C.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. PLW1 for the cyclosporin dataset could not be found.

Both the andrographolide and cyclosporin A pulse-acquire experiments (Section 3.1) were acquired using 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`. Noteworthy experiment parameters are provided by Table C.4.

C.2.2 Diffusion Datasets

	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 C.5: Noteworthy experiment parameters for the diffusion 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.

Experiment parameters related to the diffusion datasets presented in Section 3.2 are provided by Table C.5. The andrographolide diffusion dataset (Figure 3.6) was acquired using the one-shot DOSY pulse sequence [130] (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 C.1. The glucose/valine/threonine diffusion dataset (Figure 3.7) was acquired using Bruker's `ledbpgp2s` 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 [129]. The pulse sequence is displayed in Figure C.2. Analysis of the dataset using the multivariate methods DECRA and SCORE was facilitated through use of The University of Manchester's general NMR analysis toolbox (GNAT) software [188].

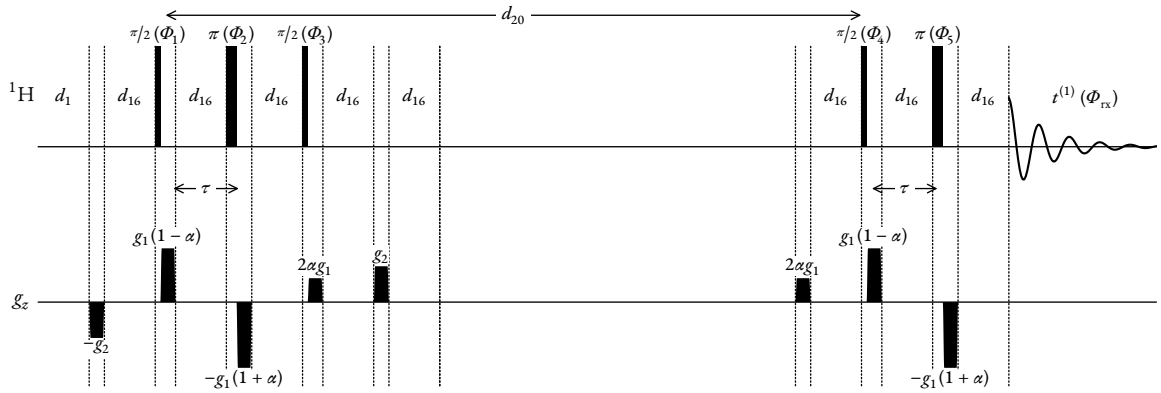


Figure C.1: The oneshot DOSY pulse sequence, which was used for the acquisition of andrographolide data (Figure 3.6). All delays are included, though they are not to scale. Delays: d_1 (relaxation delay): 1.5 s, d_{16} (gradient recovery delay): 200 µs, d_{20} (diffusion time): 0.1 s. Hard pulses had a power of 24 W, with the duration of the $\pi/2$ pulse being 12 µs. 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 µs, and a strength which was 75% of g_1. The phase cycling scheme used was: $\Phi_1 : 2 \times (0°, 180°)$; $\Phi_2 : 4 \times 0°$; $\Phi_3 : 4 \times 0°$; $\Phi_4 : 2 \times 0°, 2 \times 180°$; $\Phi_5 : 4 \times 0°$; $\Phi_{rx} : 0°, 180°, 180°, 0°$.

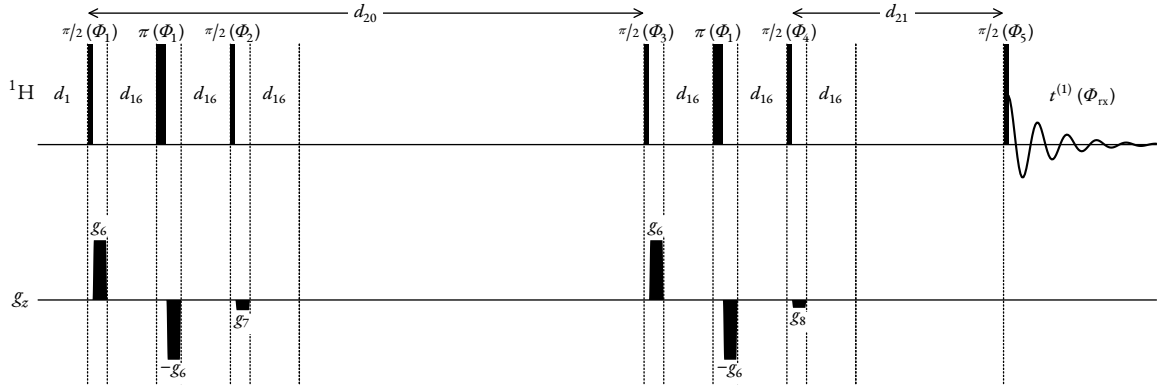


Figure C.2: The pulse sequence used for the acquisition of glucose/threonine/valine diffusion data (Figure 3.7). All delays are included, though they are not to scale. Delays: d_1 (relaxation delay): 6 s, d_{16} (gradient recovery delay): 200 µs, d_{20} (diffusion time): 0.1 s, d_{21} (eddy-current delay): 5 ms. Hard pulses had a power of 18.204 W, with the duration of the $\pi/2$ pulse being 10 µs. 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. The value used for g_6 were (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 µs. The gradients had the relative strengths $g_7 = -0.1713 g_6$ and $g_8 = -0.1317 g_6$. The following phase cycling scheme used was: $\Phi_1 : 32 \times 0°$; $\Phi_2 : 8 \times (2 \times 0°, 2 \times 180°)$; $\Phi_3 : 2 \times (4 \times 0°, 4 \times 180°, 4 \times 90°, 4 \times 270°)$; $\Phi_4 : 2 \times (2 \times (0°, 180°), 2 \times (180°, 0°), 2 \times (90°, 270°), 2 \times (270°, 90°))$; $\Phi_5 : 2 \times (4 \times 0°, 4 \times 180°, 4 \times 90°, 4 \times 270°)$; $\Phi_{rx} : 2 \times (0°, 180°, 180°, 0°, 180°, 0°0°, 180°, 270°, 90°, 90°, 270°, 90°, 270°, 270°, 90°)$.

C.2.3 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 $\pi/2(\Phi_1) \rightarrow t^{(1)}/2 \rightarrow \pi(\Phi_2) \rightarrow t^{(1)}/2 \rightarrow t^{(2)}(\Phi_{rx})$, with the EXORCYCLE phase-cycling scheme used [34: Section 11.6]:

$$\begin{array}{lllll} \Phi_1: & 0° & 0° & 0° & 0° \\ \Phi_2: & 0° & 90° & 180° & 270° \\ \Phi_{rx}: & 0° & 180° & 0° & 180° \end{array}$$

Key experiment parameters are provided in Table C.6.

	Quinine	Dexamethasone	Camphor	Estradiol
f_{bf} (MHz)	500.13	600.18	500.13	500.3
$f_{off}^{(2)}$ (Hz)	2500	2815.4	1000	2501.5
$f_{off}^{(2)}$ (ppm)	4.9987	4.691	1.9995	5
$f_{sw}^{(1)}$ (Hz)	50	50	50	100
$f_{sw}^{(2)}$ (Hz)	7500	7211.5	5000	5000
$f_{sw}^{(2)}$ (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 C.6: Noteworthy experiment parameters used for the 2DJ and PSYCHE experiments. 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.

C.2.4 PSYCHE Datasets

The pure shift spectrum of dexamethasone (Figure 4.10.a) was acquired using a TSE-PSYCHE experiment (Figure C.3). 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`.

The pulse sequence used for the acquisition of the estradiol PSYCHE spectrum (Figure 4.11) is presented in Figure C.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 C.6.

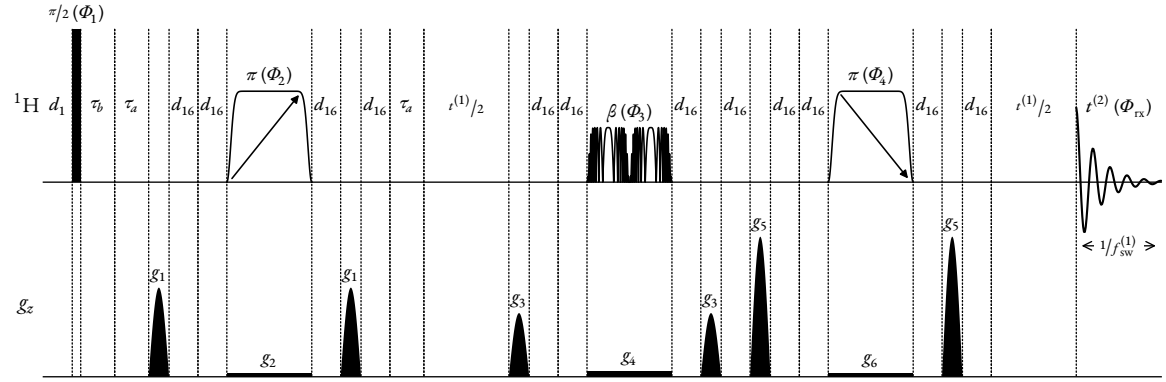


Figure C.3: The TSE-PSYCHE pulse sequence used for the acquisition of dexamethasone data (Figure 4.10.a). Delays: d_1 (relaxation delay): 2 s, d_{16}: (gradient recovery delay): 200 µs, τ_a: 5 ms (= $1/4f_{sw}^{(1)}$). The hard $\pi/2$ pulse had a duration of 12 µs, and a power of 24 W. The two π pulses 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 $\beta = 15°$, and featured two saltire chirp pulses. Both saltire pulses had a wideband, uniform rate, smooth truncation (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: $\Phi_1 : 8 \times 0°$; $\Phi_2 : 2 \times (2 \times 0°, 2 \times 180°)$; $\Phi_3 : 2 \times (0°, 90°), 2 \times (180°, 270°)$; $\Phi_4 : 8 \times 0°$; $\Phi_{rx} : 4 \times (0°, 180°)$.

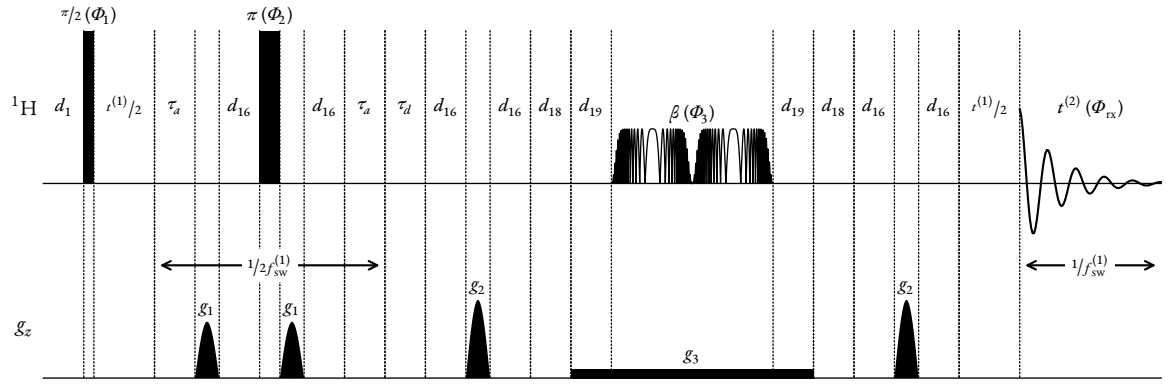


Figure C.4: The PSYCHE pulse sequence used for the acquisition of estradiol data (Figure 4.11). All delays are included, though they are not to scale. Delays: d_1 (relaxation delay): 1 s, d_{16}: (gradient recovery delay): 200 µs, d_{18}: 200 µs, d_{19}: 1 ms, τ_a: 1.3 ms, τ_d: 18.9 ms. The PSYCHE element featured two saltire chirp pulses with a WURST [189] amplitude envelope, with a target flip angle $\beta = 20°$. 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: $\Phi_1 : 2 \times (0°, 180°)$; $\Phi_2 : 4 \times 0°$; $\Phi_3 : 2 \times 0°, 2 \times 90°$; $\Phi_{rx} : 0°, 180°, 180°, 0°$.

C.3 CUPID result metrics

Table C.7: Metrics for all the results generated using CUPID (Section 4.3). *Initial M* specifies the number of oscillators given to the MMEMPM. 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 *M after MMEMPM* column indicates how many oscillators were present in the initial guess $\boldsymbol{\theta}^{(0)}$. This can differ from *Initial M*, as any oscillators possessing negative damping factors in the MMEMPM result were purged. *M after NLP* indicates how many oscillators were present in $\boldsymbol{\theta}^{(*)}$, the result of NLP. When this value is smaller than *M after MMEMPM*, 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 M	MMEMPM time (s)	M after MMEMPM	NLP time (s)	NLP iterations	M after NLP	Purged oscillators
Four Multiplets (Run 1)							
0.06 – -0.06	34†	3.2	34	45.7	65	33	1
Four Multiplets (Run 2)							
0.06 – -0.06	38†	4.2	38	76.9	105	32	None
Four Multiplets (Run 3)							
0.06 – -0.06	39†	4.3	38	27.6	35	32	None
Four Multiplets (Run 4)							
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
Strychnine Simulated							
8.15 – 8.02	6*	1.2	6	4.7	41	4	None
7.34 – 7	18*	6.9	18	28.5	90	18	None
6 – 5.83	10*	2.5	10	11.2	77	10	None

Continues on next page...

Region (ppm)	Initial M	MMEMPM time (s)	M after MMEMPM	NLP time (s)	NLP iterations	M after NLP	Purged oscillators
4.35 – 4	15*	6.6	14	8.2	26	14	None
4 – 3.64	14*	7.6	14	11.3	45	14	None
3.3 – 3.05	19*	6.83	19	30.6	88	17	None
3 – 2.56	13*	6.9	13	5.6	20	13	None
2.45 – 2.25	10*	3.4	10	11.9	61	8	None
2 – 1.74	8*	5.5	8	11.5	90	6	None
1.55 – 1.35	9*	3.2	9	6.4	28	9	1
1.35 – 1.2	10*	2.4	10	5.3	27	8	None
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
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

Continues on next page...

Region (ppm)	Initial M	MMEMPM time (s)	M after MMEMPM	NLP time (s)	NLP iterations	M after NLP	Purged oscillators
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
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
Estradiol							
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
1.45 – 1.02	90†	65.6	86	271.3	150	80	8

Appendix D

NMR-EsPy Tutorials

This appendix comprises in insert from a chapter of the NMR-EsPy documentation. The chapter features tutorials describing how to use the package's API for the consideration of 1D and 2DJ NMR datasets. These tutorials provide a short description of the key features associated with the package, and is an ideal initial point of reference for new users to gain familiarity with NMR-EsPy.

CHAPTER

TWO

TUTORIALS

In this chapter, tutorials 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 an FID, 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 will be 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 convienient 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

to phase and baseline correct. However, **you should not apply any window function to the data other than exponential line broadening.**

```
>>> import nmrespy 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 signal 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 a signal's amplitude, phase (rad), frequency (Hz), and damping factor (s^{-1}).

```
>>> import nmrespy 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,  # FID made with 2048 points
>>>     sw=1.2 * sfo,  # sweep width set to 1.2 ppm
>>>     offset=4.1 * sfo,  # transmitter offset set to 4.1 ppm
>>>     sfo=sfo,  # transmitter frequecy set to 500 MHz
>>>     snr=40.,  # signal-to-noise ratio of the FID set to 40 dB
>>> )
```

Note: For the rest of this section, 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 estimator 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 shifts, 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".**

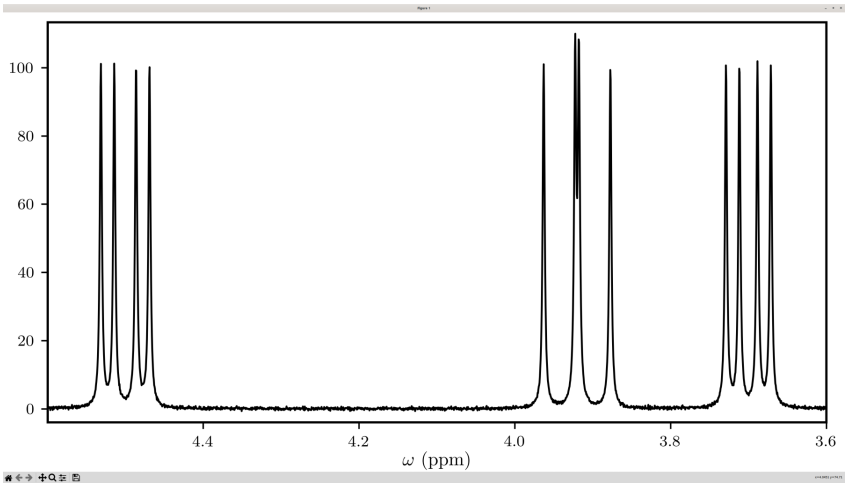
It can take some time to run this function is it involves (a) starting up MATLAB and (b) running a simulation of the experiment.

```
>>> import nmrespy as ne
>>> # 2,3-Dibromopropanoic acid
>>> shifts = [3.7, 3.92, 4.5]
>>> couplings = [(1, 2, -10.1), (1, 3, 4.3), (2, 3, 11.3)]
>>> sfo = 500.
>>> estimator = ne.Estimator1D.new_spinach(
>>>     shifts=shifts,
>>>     couplings=couplings,
>>>     pts=2048,
>>>     sw=1.2 * sfo,
>>>     offset=4.1 * sfo,
>>>     sfo=sfo,
>>> )
```

2.1.2 Viewing and accessing the dataset

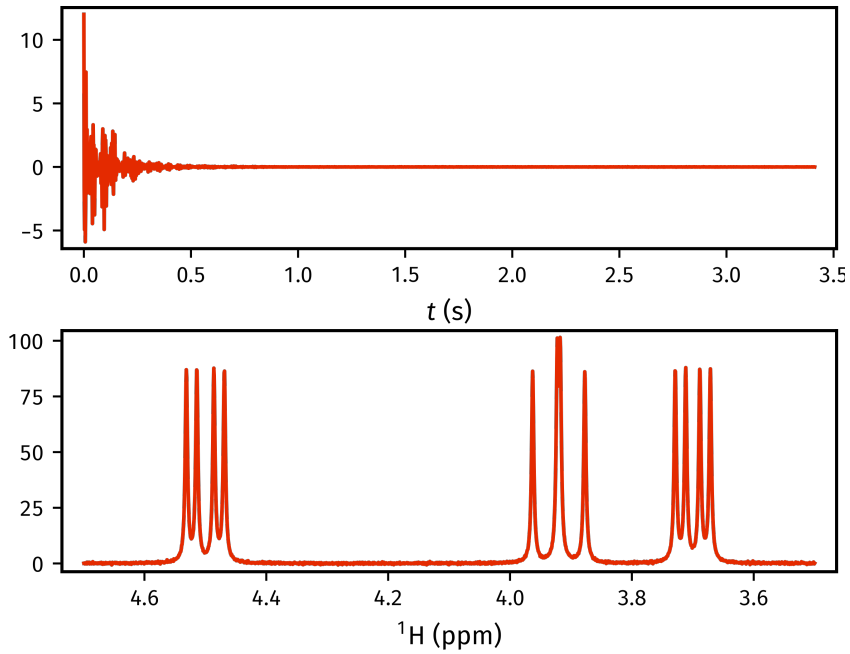
You can inspect the data associated with the estimator with `view_data()`, which loads an interactive matplotlib figure:

```
>>> estimator.view_data(freq_unit="ppm")
```



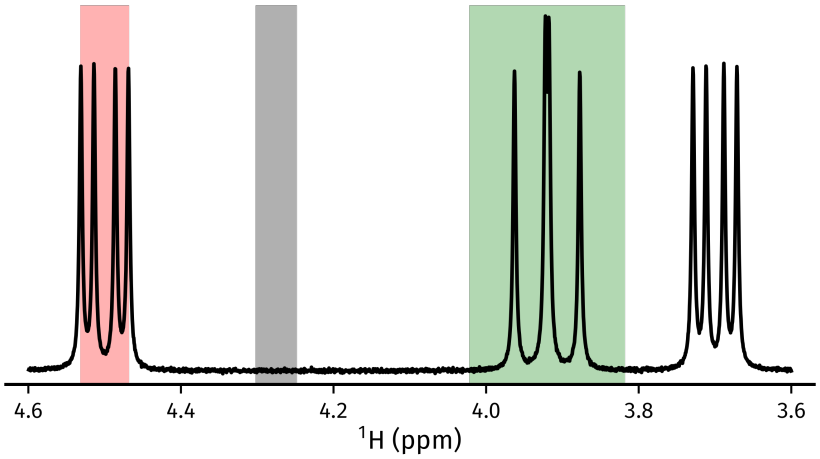
You can access the time-domain data with the `data()` property, and the associated time-points can be retrieved using `get_timepoints()`. The spectral data is accessed with `spectrum()`, and the corresponding chemical shifts with `get_shifts()`.

```
>>> 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("$^1$H (ppm)")
>>> plt.show()
```



2.1.3 Estimating the dataset

The generation of parameter estimates is facilitated using the `estimate()` method. In most scenarios, it will not be computationally feasible to estimate the entire FID at once, due to the number of constituent datapoints and signals. 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 in which all signals of interest fully reside within its bounds. As well as this, a region that is devoid of signals (the "noise region") must be indicated. In the figure below, the red region would be inappropriate as the signals clearly do not reside within it fully. The green region is acceptable, as the signals do abide by this. Finally, the grey region is a suitable noise region as it only comprises points in the baseline.



For our dataset, we will estimate three regions, encompassing each multiplet structure in the spectrum. Each region must 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 containing all the generated results. For many methods that make use of 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 the signal 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)

(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*.

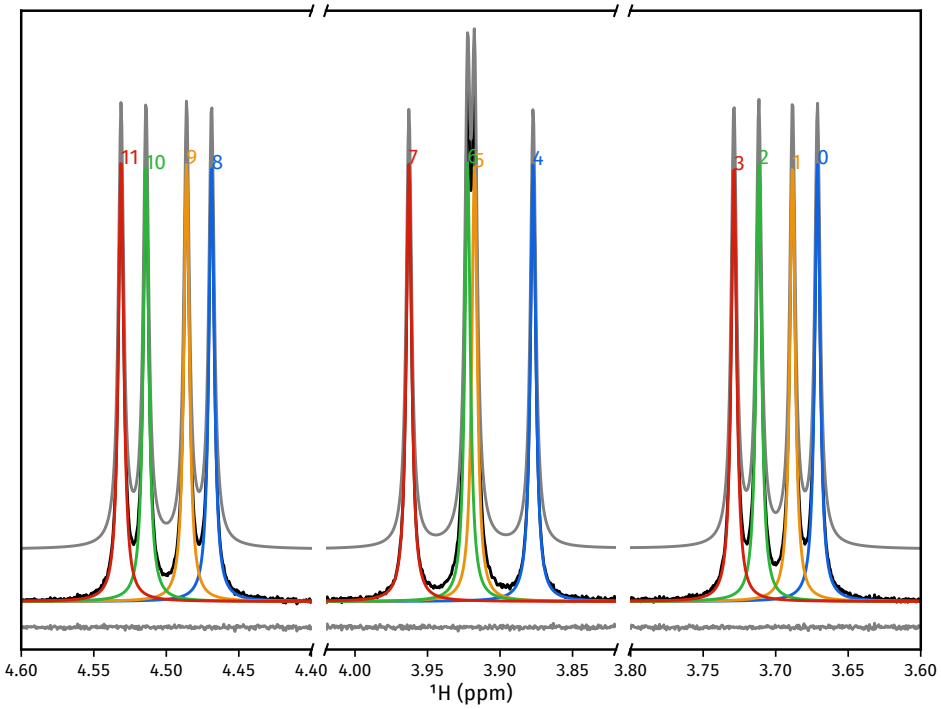
```
>>> for fmt in ("txt", "pdf"):
>>>     estimator.write_result(
>>>         path="tutorial_1d",
>>>         fmt=fmt,
>>>         description="Simulated 2,3-Dibromopropanoic acid signal.",
>>>     )
Saved file tutorial_1d.txt.
Saved file tutorial_1d.tex.
Saved file tutorial_1d.pdf.
You can view and 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 assign to the `estimator_cp` variable
>>> estimator_cp = ne.Estimator1D.from_pickle("tutorial_1d")
```

Saving a logfile

A logfile listing all the methods called on the estimator can be created 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 2D J-Resolved (2DJ) spectroscopy datasets. This facilitates use of **CUPID** (**C**omputer-assisted **U**ndiminished-sensitivity **P**rotocol for **I**deal **D**ecoupling) which can be used to generate homodecoupled (*pure shift*) spectra and to predict multiplet structures.

Many methods in this class have analogues in `Estimator1D`. You are advised to read through the *1D tutorial* before continuing, as minimal descriptions will be provided of concepts 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>/<expno>/"`. There should be a `ser` file, an `acqus` file, and an `acqu2s` file directly under this directory. Phasing in the direct-dimension will be needed. If deemed necessary, baseline correction can be performed too.

```
>>> import nmrespy as ne
>>> estimator = ne.Estimator2DJ.new_bruker("/home/simon/nmr_data/qunine/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                      │¹H   │¹H
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 DFT calculation shifts and couplings
>>> # Note that the dataset generated will not be reminiscent of what
>>> # the actual solution-state dataset for sucrose looks like!
>>> 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.,
... )
```

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

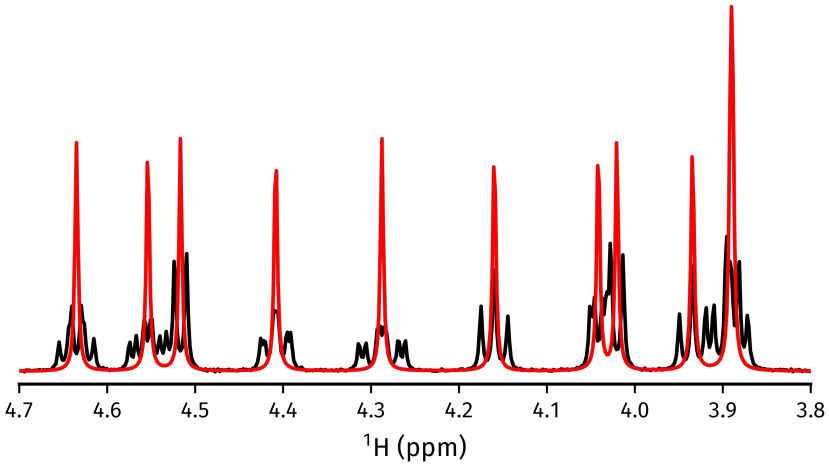
The procedure for estimating 2DJ data is very similar to that of 1D data. You need to specify regions in the direct dimension that are of interest for generating filtered sub-FIDs. No filtering is done in the indirect dimension. In our example, it turns out that for a couple of the regions selected, the number of oscillators automatically predicted is slightly smaller that the "true" number, and so the true number has been hard-coded (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)
>>> # It is a good idea to pickle the estimator after estimation
>>> estimator.to_pickle("sucrose")
```

2.2.3 Acquiring a pure shift spectrum

The `cupid_spectrum()` method produces a pure shift spectrum using the 2DJ parameter estimate. In the code snippet below, a figure is made which compares the pure shift spectrum with the spectrum of the first direct-dimension slice in the 2DJ data, i.e. 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 FID any pair of signals i, j should satisfy the following:

$$\left|\left(f_i^{(2)} - f_i^{(1)}\right) - \left(f_j^{(2)} - f_j^{(1)}\right)\right| < \epsilon,$$

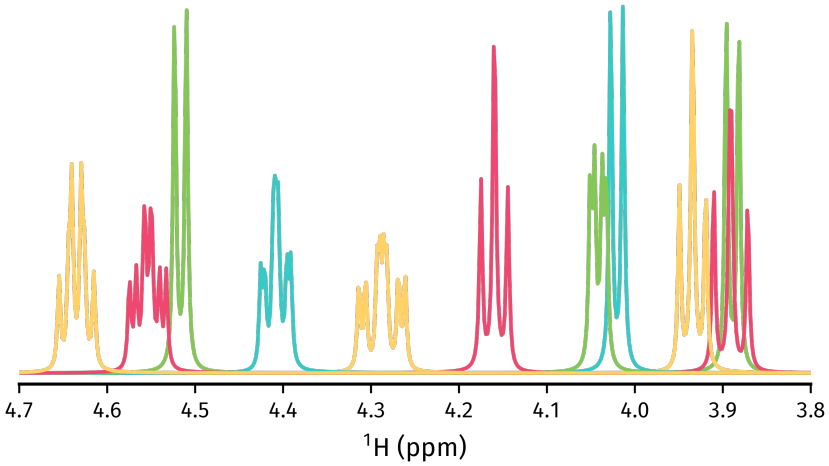
where ϵ is an error threshold, and $f^{(1)}$ and $f^{(2)}$ are the estimated indirect- and direct-dimension frequencies, respectively. `predict_multiplets()` generates groups of oscillator indices satisfying the above criterion. A key parameter for this is `thold`, which sets the error threshold ϵ. By default, this is set to be $\min\left(f_{sw}^{(1)}/N^{(1)}, f_{sw}^{(2)}/N^{(2)}\right)$, 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 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.

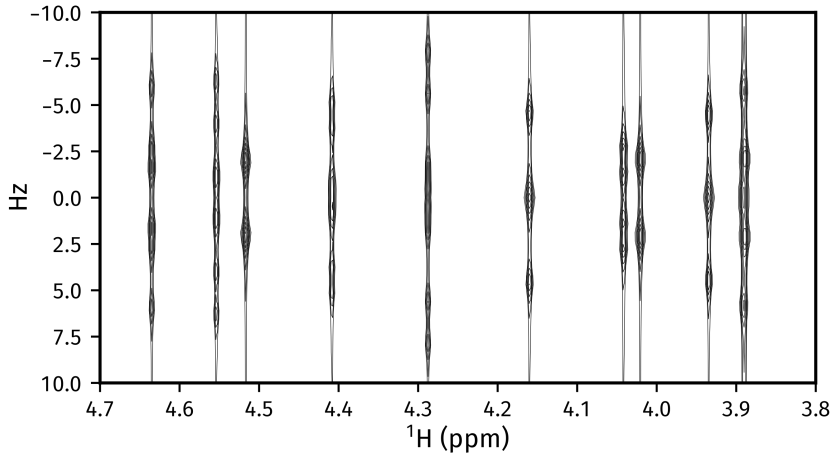
```
>>> # Direct-dimension shifts
>>> shifts_f2 = estimator.get_shifts(unit="ppm", meshgrid=False)[-1]
>>> fids = estimator.construct_multiplet_fids(indices=indices)
>>> # Create an iterator which cycles through values infinitely
>>> from itertools import cycle
>>> colors = cycle(["#84c757", "#ef476f", "#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"):
...     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° shear (commonly called a 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 a parameter estimate of an FID, a synthetic signal with these adjusted frequencies can be constructed. As well as this, generating a pair of phase- or amplitude-modulated FIDs enables the construction of absorption-mode spectra (cf the issues involved in generating nice spectra from hypercomplex 2DJ datasets). Use `nmrespy.Estimator2DJ.sheared_signal()`, with `indirect_modulation` set to either `"amp"` or `"phase"` to generate the desired spectrum. Then, use either `nmrespy.sig.proc_phase_modulated()` or `nmrespy.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(
...     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")
```

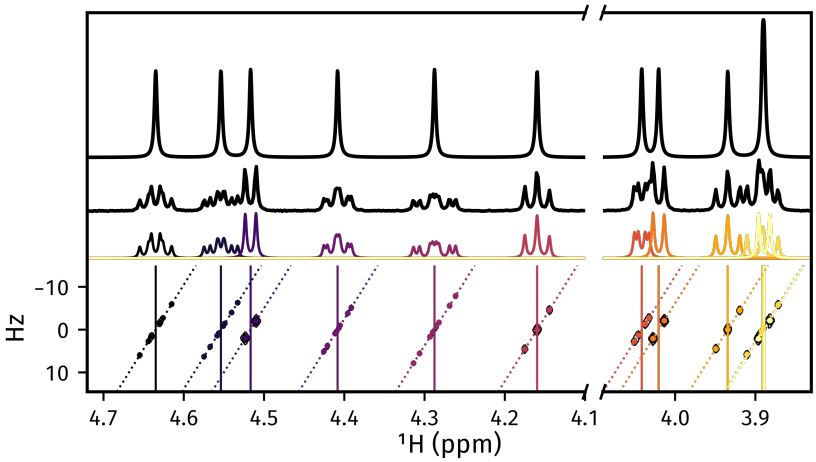


2.2.6 Plotting result figures

The `plot_result()` method enables the generation of a figure which gives 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 generated from the first direct-dimension FID in 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),
...     # There is a lot of scope for editing the colours of
...     # multiplets. See the reference!
...     # Here I specify the name of a colormap in matplotlib.
...     multiplet_colors="inferno",
...     # Argumnets 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 that is done 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,
...     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* tutorial.