Estimation of NMR Signals in the Time Domain

Simon G. Hulse

Balliol College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Trinity Term, 2023

Supervisors:
Mohammadali Foroozandeh
Timothy Claridge
Fay Probert

TODO

- Introduction: describe estimation techniques: LPSVD, HSVD, VARPRO/AMARES, CRAFT, ML-based approaches
- More 1D results: cyclosporin from paper, Artemisinin.
- More amplitude-attenuated datasets: Glucose/valine.threonine mixture should yield results.
- BBQCHILI: Ask Ali for more info on dataset
- NMR-EsPy discussion

Acknowldegements

- Family: Gran, Mum, Dad, Robert, Isobel.
- Friends: Nick, Jim, Joe, Dominic, ...
- Academic: MF, TDWC, FP, AJB, Jon, Ali S, JB, David, James M

Abstract

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

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

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

Acronyms

1D one-dimensional

2D two-dimensional

3D three-dimensional

2DJ 2D J-resolved

AIC Akaike information criterion

ALPESTRE a linear predictive estimation of signal time reversal

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

API application programming interface

AR autoregressive

ARMA autoregressive moving average

AWGN additive white gaussian noise

BIRD bilinear rotation decoupling

CHORUS chirped, ordered pulses for ultra-broadband spectroscopy

COSY correlation spectroscopy

CPMG Carl-Purcell-Meiboom-Gill

CPU central processing unit

CRAFT complete reduction to amplitude frequency table

CUPID computer-assisted undiminished-sensitivity protocol for ideal decoupling

DFT density functional theory

DMSO Dimethyl sulfoxide, $(H_3C)_2SO$

DMSO-d_6 Deuterated DMSO

DOSY diffusion-ordered spectroscopy

EYM Eckart-Young-Mirsky

FID free induction decay

FFT fast Fourier transform

FS frequency-swept

FT Fourier transform

GN Gauss-Newton

GUI graphical user interface

IFT inverse Fourier transform

INEPT insensitive nuclei enhancement by polarization transfer

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

NLP non-linear programming

NMR nuclear magnetic resonance spectroscopy

NMR-EsPy NMR estimation in Python

NOESY nuclear overhauser effect spectroscopy

pdf probability density function

PFG pulsed field gradient

PGSE pulsed gradient spin echo

PGSTE pulsed gradient stimulated echo

PGSTEBP pulsed gradient stimulated echo with bipolar gradients

PSYCHE pure shift yeilded by chirp excitation

RAM random access memory

RF radio-frequency

SNR signal-to-noise ratio

ST Steihaug-Toint

SVD singular value decomposition

TSE-PSYCHE triple spin echo PSYCHE

TROSY transverse relaxation optimised spectroscopy

VE virtual echo

VARPRO variable projection

WURST wideband, uniform rate, smooth truncation

ZS Zangger-Sterk

Nomenclature

General Mathematics

$\mathcal{A} := \mathcal{B}$	$\mathcal{A}$ is defined to be equal to $\mathcal{B}$
$\lfloor a \rceil$	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 $$a \bmod b = a - \left\lfloor \frac{a}{b} \right\rfloor$$

Complex Numbers

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

Number sets

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

Probability

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

$A \perp\!\!\!\perp B$ — A and B are conditionally independent

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

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

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

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

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

Vectors, Matrices, and Arrays

$.^{\mathrm{T}}$ — Transpose

$.^{\dagger}$ — Conjugate transpose

$.^{-1}$ — Inverse

$.^{+}$ — Moore-Penrose pseudo-inverse.

$.^{\circlearrowright(d)}$ — Right circular rotation by along axis d of the array by one element. For example, for a 2D matrix $\boldsymbol{X} \in \mathbb{F}^{M \times N}$, $\boldsymbol{X}^{\circlearrowright(1)}$ is given by

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

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

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

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

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

$\mathrm{diag}(\cdot)$ Produces a diagonal matrix from a vector. 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

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

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

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

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

$\boldsymbol{x} \otimes \boldsymbol{y}$ 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

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

Calculus

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

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

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Code Listings

Introduction

1

Since its conception almost 80 years ago, NMR has become a ubiquitous technique in chemistry, biochemistry and numerous other disciplines, thanks to the unique insights into structure and dynamics it can provide. In this chapter, after a very brief historical overview of the subject, an outline of the basic theory behind the experiment – without delving into the necessary quantum mechanics for a truly formal account – is given. Details of the means by which the raw experimental data, the FID is produced, and its structure, are given. Following on from this, an account is given of techniques used to extract quantitative information from the FID, which is the central motivation of this work.

1.1 Introducing NMR

1.1.1 A Brief History of NMR

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

Russel Varian secured the first patent for a commercial NMR machine, with a 30 MHz spectrometer following soon after. The first spectrometers functioned by slowly sweeping the mag-

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

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

NMR's significance as an analytical tool is evidenced by Nobel Prizes in Chemistry being awarded for work in the field on two separate occasions. First, Ernst received the prize in 1991 "for his contributions to the development of the methodology of high resolution nuclear magnetic resonance spectroscopy"[18]. In 2002, Wüthrich was recognised "for his development of nuclear magnetic resonance spectroscopy for determining the three-dimensional structure of biological macromolecules in solution"[19].

1.1.2 Nuclear Spin and Magnetism

NMR relies on *spin*, an intrinsic property of certain nuclei (along with other elementary particles) which, along with orbital angular momentum, is one of the two sources of angular momentum in quantum mechanics. The angular momentum associated with a nuclear spin is characterised by the quantum number I, which may be integer or half-integer. 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 quantum mechanics, with many excellent texts devoted to the subject[20–22]. However, a basic appreciation can be gained using the Bloch model, a semi-classical description of the simplest of system to study: a ensemble of spin-$1/2$ nuclei.

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

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

Nucleus	I	$\gamma\,(\mathrm{rad\,T^{-1}\,s^{-1}})$	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	6.7283×10^{7}	1.07
^{14}N	1	1.9338×10^{7}	99.636
^{15}N	1/2	-2.7126×10^{7}	0.364
^{16}O	0	-	99.756
^{17}O	5/2	-3.6281×10^{7}	0.038
^{19}F	1/2	2.5162×10^{8}	100
^{31}P	1/2	1.0839×10^{8}	100

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

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

$$I_z = \hbar m, \tag{1.2}$$

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

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

γ is a proportionality constant called the *gyromagnetic ratio*, which is dependent on the nucleus of interest. Table 1.1 provides the gyromagnetic ratios for a few nuclei commonly encountered in NMR, along with a few which alas are spin-0.

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

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

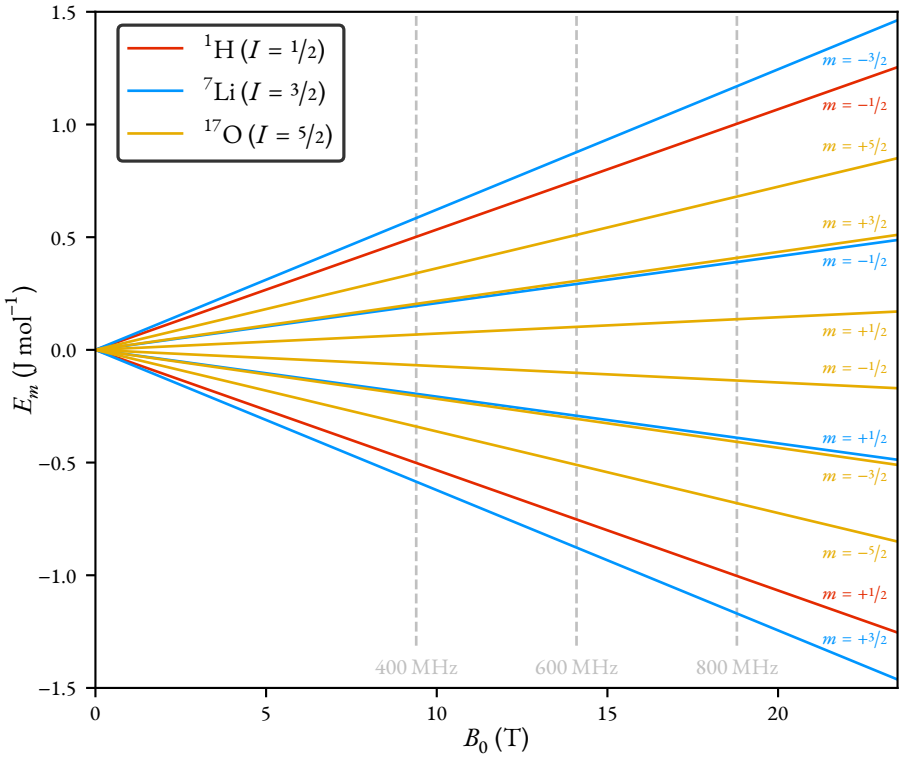


Figure 1.1: The variation of energy of the spin states of ^{1}H, ^{7}Li, and ^{17}O with external magnetic field strength (B_0), up to 23.5 T, which is approximately the strength of a 1 GHz NMR magnet. Three common field strengths for commercial NMR magnets are indicated: 9.40 T (400 MHz), 14.10 T (600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

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

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

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

NMR samples comprise a vast ensemble of equivalent spin systems, and it is the macroscopic properties of the sample that are observed. At thermal equilibrium, the various spin states will be disproportionately populated in accordance with the Boltzmann distribution, with lower energy states being more heavily populated. For example, an ensemble of non-interacting spin-1/2 nuclei with $\gamma > 0$ will have a more populated $m = +1/2$ (α) state, relative to the $m = -1/2$ (β) state. Due to the very small relative energy difference however, it should be noted that the discrepancy in state populations is very small. The ensemble acquires a net (bulk) magnetic moment $\boldsymbol{M}$, given by the

summation of all the individual spin moments:

$$\boldsymbol{M} = \sum_{n=1}^{N} \boldsymbol{\mu}_n, \tag{1.6}$$

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

$$\sum_{n=1}^{N} \mu_{x,n} = \sum_{n=1}^{N} \mu_{y,n} = 0, \tag{1.7}$$

such that the bulk magnetisation is collinear with the field direction, with a magnitude M_0, which is approximately given by

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

where k_{B} is the Boltzmann constant, and T is the sample temperature. For purposes of experiment sensitivity, it is hence desirable to utilise nuclei with (a) high natural abundance (which affects N) 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.

In the presence of a magnetic field, the bulk magnetism experiences a torque, with its of rate of change with respect to time being

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

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 from $\boldsymbol{B}_0$. The contribution to the field induced by pulses is commonly denoted $\boldsymbol{B}_1(t)$, 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 z-axis, and $\boldsymbol{B}_1 = \boldsymbol{0}$, such that

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

*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.

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

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

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

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

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

$$\frac{\mathrm{d}\tilde{\boldsymbol{M}}(t)}{\mathrm{d}t} = \tilde{\boldsymbol{M}}(t) \times \gamma\tilde{\boldsymbol{B}}(t), \tag{1.13a}$$

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

$$\Delta B_0 = -\frac{\Omega}{\gamma}, \tag{1.13c}$$

$$\Omega = -\gamma B_0 - \omega_{\mathrm{RF}}. \tag{1.13d}$$

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

Consider a scenario where a short RF pulse is applied to the system, which is then allowed to undergo free precession. (1.13) 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, $\boldsymbol{M}_{\mathrm{eq}} \equiv \tilde{\boldsymbol{M}}_{\mathrm{eq}} = [0, 0, M_0]^{\mathrm{T}}$. In the Bloch model, two processes are introduced to account for this[‡] *longitudinal* (spin-lattice) relaxation, which is responsible for the recovery of the z-component of the magnetisation to M_0, and *transverse* (spin-spin) relaxation, which leads to the x- and y-components reverting to 0. These are assigned rates of R_1 and R_2, respectively.

Everything has now been established to state the Bloch equations, which describe the evolu-

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

[‡]The introduction of relaxation is phenomenological in the Bloch model. The reversion of the spin system to equilibrium is added purely because the model agrees with observation.

tion of the bulk magnetisation of an ensemble of identical spin-1/2 nuclei in the rotating frame:

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

where $\omega_1 = -\gamma B_1$. Figure 1.2.a depicts the evolution of an off-resonance ($\Omega \neq 0\ \mathrm{rad\,s^{-1}}$) 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 $90°_y$, which indicates the *flip angle* it induces, and its phase. Assuming that negligible evolution due to the offset occurs during the pulse, the magnetisation vector will land on the x-axis, and evolve according to

$$\tilde{M}_x(t) = M_0 \cos(\Omega t) \exp(-R_2 t), \tag{1.15a}$$
$$\tilde{M}_y(t) = M_0 \sin(\Omega t) \exp(-R_2 t), \tag{1.15b}$$
$$\tilde{M}_z(t) = M_0 (1 - \exp(-R_1 t)). \tag{1.15c}$$

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

$$y(t) = c\tilde{M}_+(t), \tag{1.16a}$$
$$\tilde{M}_+(t) = \tilde{M}_x(t) + \mathrm{i}\tilde{M}_y(t) = M_0 \exp(\mathrm{i}\Omega t - R_2 t). \tag{1.16b}$$

Bloch model becomes inadequate once more complex spin systems, featuring multiple spins which interact through mechanisms such as J-couplings. However, it provides valuable insights into the form that a signal detected by an NMR experiment takes.

1.1.3 The NMR Spectrometer

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

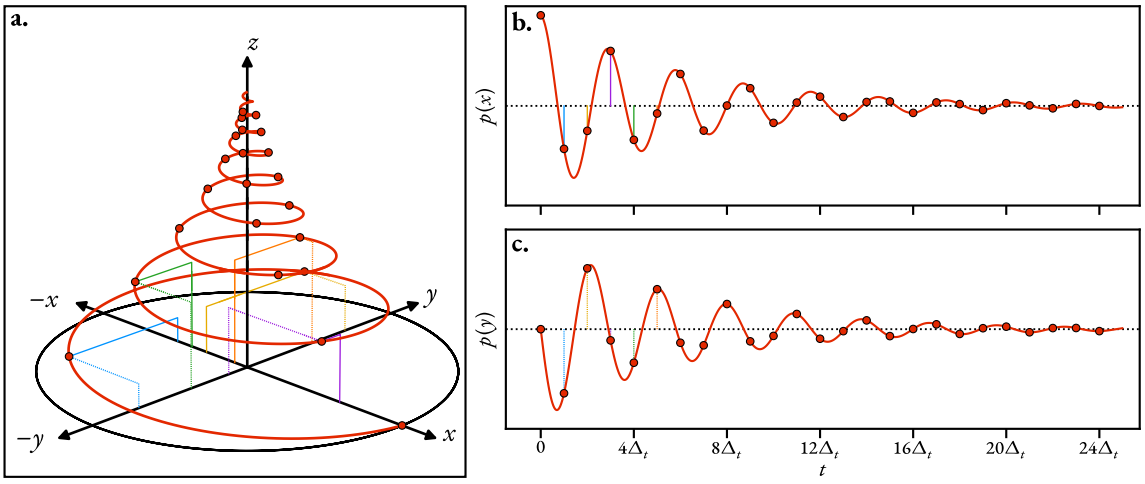


Figure 1.2: a. An illustration of the free evolution of the bulk magnetisation of an ensemble of spin-1/2 nuclei immediately after the application of a 90°$_y$ pulse according to the Bloch model. The projections of the magnetisation vector onto the x- and y-axes are plotted in panels **b.** and **c.**,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

The Magnet

The static $\boldsymbol{B}_0$ field is generated by a magnet, composed of a superconducting solenoid immersed in liquid helium. Common materials used for the solenoid include Nb-Ti alloy and 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. A bore passes through the z-direction of the magnet, which is maintained at a user-controlled temperature. Within the bore sits the probe as well as the sample. Magnets with high field strengths are desirable, as both the resolution ($\propto B_0$) and sensitivity ($\propto B_0^{3/2}$) of the data are affected. At the time of writing, commercial spectrometers which operate at and above a ^{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. For most applications, including the study of small molecules, spectrometers with more modest field strengths are typically adequate.

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

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

[§]^{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 is ubiquitous, but rarely directly studied.

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

The Probe

The probe sits inside the bore of the magnet, and features the coil(s) used to pulse the sample with RF radiation. The same coil(s) also receive the response from the sample. The principle source of noise in NMR experiments is thermal noise within the probe circuitry. For this reason, cryogenic probes[23] have become a popular development, in which the transmit/receive coil(s) and other probe electronics are maintained at a very low temperature (typically about 20K).

The Transmitter

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

The Receiver

During acquisition, the time-varying current induced in the probe coil(s) by the sample magnetisation is sent to a receiver, which comprises a series of components which convert it to the FID which is stored in computer memory. One of the processes that the receiver is responsible for is *quadrature detection*[24: Section 13.6], which enables two orthogonal transverse components of the magnetisation to be detected, and is crucial to produce spectra which are frequency discriminated (see Section 1.2.2). This is achieved by splitting the signal into two channels. Both signals, which are of very high frequency (MHz) are mixed with a reference signal of frequency ω_{RF}, such that a low-frequency signal (kHz) is generated, along with a very high frequency signal. One of the reference signals possess a phase which is shifted by 90° relative to the other one, such that the two channels constitute a quadrature pair. The signals are then sent through a low-pass filter to remove the high frequency component produced through mixing. Finally, an analogue to digital converter translates the signal to the real and imaginary components of a binary dataset which is stored in memory.

1.1.4 The structure of the FID

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

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

where $y(t)$ is the (continuous) variation of the generated signal with time, and N is the number of points sampled. The inverse of the sampling rate, $1/\Delta_t$ is the *sweep width* f_{sw} which defines how wide the range of samplable frequencies is, in accordance with the Nyquist theorem[25].

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

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

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

(1.18) 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}).

The influence the four parameters have on a signal is depicted in panels a1–d1 in Figure 1.3. An FID can therefore be parameterised by the vector $\boldsymbol{\theta} \in \mathbb{R}^{4M}$:

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{a}^{\mathrm{T}} & \boldsymbol{\phi}^{\mathrm{T}} & \boldsymbol{f}^{\mathrm{T}} & \boldsymbol{\eta}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{1.19}$$

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.

Multidimensional experiments involve incrementing one or more delay within the pulse sequence, in order to obtain an array of 1D FIDs. In a D-dimensional dataset, each contributing signal is parameterised by an amplitude and phase as before, along with D distinct frequencies and

damping factors, such that a general parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{2(1+D)M}$ is given by

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{a}^{\mathrm{T}} & \boldsymbol{\phi}^{\mathrm{T}} & \boldsymbol{f}^{(1)\mathrm{T}} & \dots & \boldsymbol{f}^{(D)\mathrm{T}} & \boldsymbol{\eta}^{(1)\mathrm{T}} & \dots & \boldsymbol{\eta}^{(D)\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{1.20}$$

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

$$Y_{n^{(1)},\cdots,n^{(D)}} = X_{n^{(1)},\cdots,n^{(D)}}(\boldsymbol{\theta}) + W_{n^{(1)},\cdots,n^{(D)}}, \tag{1.21a}$$

$$X_{n^{(1)},\cdots,n^{(D)}} = \sum_{m=1}^{M} a_m \exp(\mathrm{i}\phi_m) \prod_{d=1}^{D} \zeta^{(d)} \left(2\pi \left(f_m^{(d)} - f_{\mathrm{off}}^{(d)}\right) n^{(d)} \Delta_t^{(d)}\right) \exp\left(-\eta_m^{(d)} n^{(d)} \Delta_t^{(d)}\right), \tag{1.21b}$$

$$\zeta^{(d)}(\cdot) \begin{cases} = \exp(\mathrm{i}\cdot) & d = D \\ \in \{\cos(\cdot), \sin(\cdot), \exp(\mathrm{i}\cdot)\exp(-\mathrm{i}\cdot)\} & \text{otherwise} \end{cases}, \tag{1.21c}$$

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

$$\begin{aligned} W_{n^{(1)},\cdots,n^{(D)}} &:= w \sim \mathcal{N}_{\mathbb{C}}\left(0, 2\sigma^2\right) \\ \implies \Re(w) &\perp\!\!\!\perp \Im(w), \Re(w) \sim \mathcal{N}\left(0, \sigma^2\right), \Im(w) \sim \mathcal{N}\left(0, \sigma^2\right). \end{aligned} \tag{1.22}$$

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

$$\mathrm{SNR}(\boldsymbol{Y}) := \frac{1}{2\mathfrak{N}\sigma^2} \sum_{n^{(1)}=0}^{N^{(1)}-1} \cdots \sum_{n^{(d)}=0}^{N^{(D)}-1} \left| X_{n^{(1)},\cdots,n^{(D)}} \right|^2, \tag{1.23}$$

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

instead, in units of decibels (dB):

$$\mathrm{SNR}_{\mathrm{dB}} := 10 \log_{10} (\mathrm{SNR}) . \tag{1.24}$$

1.2 An Overview of NMR Data Analysis

To gain insights from NMR experiments on the chemical system of interest, extraction of the defining parameters $\boldsymbol{\theta}$ is necessary, though the majority of NMR users are unlikely to think about the process of NMR analysis in this way. For example:

- An understanding of the chemical environments of atoms in a molecule can be gained by considering the chemical shifts of the various peaks in the spectra, which are a proxy for the FID frequencies $\boldsymbol{f}$.
- The relative stoichiometries**does this make sense?** of a molecule can be elucidated by inspecting the integrals of spectral peaks, which are directly related to the FID amplitudes $\boldsymbol{a}$

- **Typical approach to analysing the data (FT, peak pick, integrate, baseline correction, window functions, zero-filling etc),**
- **Estimation techniques: LP, SVD techniques, iterative techniques (AMARES, VARPRO), Bayesian techniques (CRAFT), ML techniques**

1.2.1 Conventional NMR Analysis

The conventional route to interpret NMR experiments is to transform the FID into the frequency domain, producing an NMR spectrum. This is achieved through application of the *Fourier transform* (FT):

$$\text{Continuous case:} \qquad \mathrm{FT}(x(t))(F) = \int_0^\infty x(t) \exp(-2\pi \mathrm{i} t F) \mathrm{d}t \quad \forall t \in \mathbb{R}, \tag{1.25a}$$

$$\text{Discrete case:} \qquad \mathrm{FT}(\boldsymbol{x})_n = \sum_{k=0}^{N-1} x_k \exp\left(-\frac{2\pi \mathrm{i} k n}{N}\right) \quad \forall n \in \{0, \cdots, N-1\}. \tag{1.25b}$$

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

$$s(F) = \mathrm{FT}\left(a\exp(\mathrm{i}\phi)\exp\left((2\pi\mathrm{i}f-\eta)t\right)\right)(F), \tag{1.26a}$$

$$s(F) = \frac{a\exp(\mathrm{i}\phi)}{\eta+2\pi\mathrm{i}(f-F)}. \tag{1.26b}$$

When $a = \phi = 0$, this function is equivalent to the (unnormalised) probability density function (pdf) of the Cauchy distribution. The FT is a linear function, such that the FT of a summation of signals is equivalent to the summation of the FTs of each signal. A corollary is that an NMR spectrum comprises a series of Lorentzian "peaks", located at the frequencies of all the signals in the FID:

$$s(F) = \sum_{m=1}^{M}\frac{a_m\exp(\mathrm{i}\phi_m)}{\eta_m+2\pi\mathrm{i}(f_m-F)}. \tag{1.27}$$

The FT is a very attractive means of processing NMR data, as it presents the data in a format which is human-interpretable, with the basic rules describing how chemical structure is mapped to NMR spectral properties being a fundamental skill that virtually all experimental chemists require[27]. Due to innate properties of the NMR experiment, as well as features arising from analysing a discrete signal, the raw output from an NMR experiment often leads to spectra with undesirable characteristics without any extra manipulation. Extra processing steps that are frequently applied to NMR data are now outlined, in the order that they are applied to the data. Note that apodisation and zero filling are applied to the FID (prior to FT), while phase correction and baseline correction are applied to the spectrum (after FT).

Apodisation

Apodisation refers to the process of mutating a signal by multiplying it with a specified function, often called a *window function*, in order to enhance a certain property. Apodisation is employed to improve either the sensitivity or the resolution of the final spectrum, albeit at the cost of worsening the other feature. There is no such thing as a free lunch after all.

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 dampen the signal. As such, multiplying the FID with a function which is initially large and gets progressively smaller with time can be used to enhance the SNR of the FID. The most common function to achieve this is the negative exponential such that a given point is multiplied by $\exp\left(-{}^{kn}/_{N-1}\right)$. k is referred to as the line broadening factor, since the increased dampening

[¶] An upshot of only possessing data for $t \geq 0$ is that the FT of an FID features a vertical offset, such that the baseline does not sit at 0[26]. This is corrected by halving the initial point of the FID prior to FT.

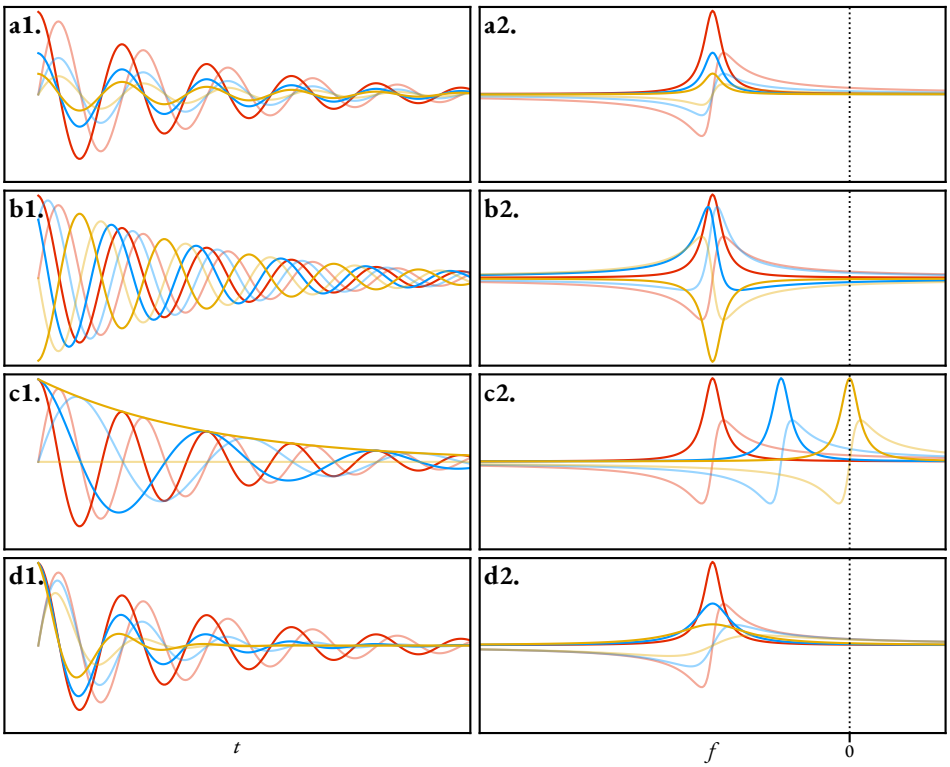


Figure 1.3: An illustration of the influence of the four parameters associated with an oscillator in both the time-domain (panels **a1.** – **d1.**) and Fourier-domain (panels **a2.** – **d2.**). The red signal is generated with the same parameters across all panels: $a = a_{\text{red}}$, $\phi = 0$,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

applied to the signal causes the linewidth of the spectral peaks to increase (see panel d of Figure 1.3). As such, peaks become more poorly resolved.

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

By being defined to decay to a sufficiently small value, or 0, at the end of the FID, window functions are also able to suppress *truncation artefacts*, which appear when the FID still possesses appreciable signal amplitude at the end of the acquisition period. Truncated FIDs produce spectra with peaks of a form that is akin to the convolution of the FT of the untruncated FID, with the FT of a box-function, which takes the form of a sinc function ($\sin(x)/x$). The resulting artefacts in spectra are often referred to as *sinc wiggles* for this reason.

Zero filling

TODO

Phase correction

The real and imaginary components of (1.27) are as follows:

$$\Re(s(F)) = \sum_m a_m(\cos(\phi_m)\mathcal{A}_m(F) + \sin(\phi_m)\mathcal{D}_m(F)), \tag{1.28a}$$

$$\Im(s(F)) = \sum_m a_m(\sin(\phi_m)\mathcal{A}_m(F) - \cos(\phi_m)\mathcal{D}_m(F)), \tag{1.28b}$$

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

$$\mathcal{A}_m(F) = \frac{\eta_m}{\eta_m^2 + 4\pi^2(f_m - F)^2}, \tag{1.29a}$$

$$\mathcal{D}_m(F) = \frac{2\pi(f_m - F)}{\eta_m^2 + 4\pi^2(f_m - F)^2}. \tag{1.29b}$$

As illustrated most clearly in panel c2 of Figure 1.3, a peak with an absorption lineshape is far more desirable than one with a dispersion lineshape for two key reasons: (a) its maximum corresponds to the oscillator frequency, while a dispersion Lorentzian has a magnitude of 0 at the oscillator frequency (b) they decay more rapidly towards 0 and are therefore more resolved. Generating a spectrum where all peaks possess absorption Lorentzians is therefore desired, which is possible if

all signal have a phase of 0°, which leads to

$$\Re(s(F)) = \sum_m a_m \mathcal{A}_m(F), \tag{1.30a}$$

$$\Im(s(F)) = -\sum_m a_m \mathcal{D}_m(F). \tag{1.30b}$$

Fortunately, for the majority of NMR experiments[‖], the contributing signals possess phases which depend linearly on their frequencies, i.e.

$$\phi_m = \Phi_0 + \Phi_1 f_m, \tag{1.31}$$

where $\Phi_0, \in (-\pi, \pi]$ and $\Phi_1 \in \mathbb{R}$ are zero- and first-order phase terms. A feature of all NMR processing platforms is the ability to perform phase-correction, in which the user determines Φ_0 and Φ_1 by inspecting the appearance of the spectrum $\boldsymbol{s}_\phi$ for different values of p_0 and p_1, according to

$$s_{\phi,n} = s_n \exp\left(\mathrm{i}\left(p_0 + \frac{p_1 n}{N-1}\right)\right), \tag{1.32}$$

in an attempt to give all peaks in the spectrum absorption-mode lineshapes.

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 is used to describe scenarios when the baseline, rather than exhibiting a flat profile with a moving average of zero, has a distorted shape instead. There a number of potential causes of baseline distortion, including acquisition starting at a time ≠ 0, "clipping" of the initial points due to excessive receiver gain, and "baseline roll" due to the transient response of audio filters[22: Section 3.3, 26]. Whatever the cause(s), it is typically the corruption of the initial points in the FID that causes baseline distortion. It is common to apply baseline correction algorithm after phase correction has been undertaken to negate any distortion. This involves multiplying the spectrum with a high-order polynomial function, whose coefficients are determined by an appropriate algorithm. A popular algorithm involves the two-step procedure of (a) determining spectral regions which are part of the baseline (b) fitting the baseline regions to a polynomial[28, 29].

[‖]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.

1.2.2 Conventional analysis for multidimensional datasets

As for 1D datasets, it is desirable that multidimensional NMR spectra feature peaks which are (a) frequency discriminated, and (b) comprise pure absorption-mode lineshapes in each dimension. Frequency discrimination describes the ability to determine whether a particular resonance is at a frequency which higher or lower than the frequency of the transmitter, which is in the middle of the spectral window.

Amplitude-modulated signals

It is clear that a cosine modulated signal, given by (1.21) with $D = 2$ and $\zeta^{(1)} = \cos(\cdot)$ cannot achieve frequency discrimination, on account of the relation

$$\cos\left(2\pi f^{(1)} t^{(1)}\right) = \tfrac{1}{2}\left(\exp\left(2\pi \mathrm{i} f^{(1)} t^{(1)}\right) + \exp\left(-2\pi \mathrm{i} f^{(1)} t^{(1)}\right)\right) \tag{1.33}$$

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, $\left(2f^{(1)}_{\text{off}} - f^{(1)}, f^{(2)}\right)$. 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*[30]. A spectrum of this form is presented in panel a of Figure 1.4. It is possible to generate pure absorption peaks by applying FT in the direct dimension, setting the imaginary component to zero, and finally applying FT in the indirect dimension. The real component of the result (panel b) is referred to as a *double absorption* spectrum.

To achieve frequency discrimination, it is necessary to also possess the analogous sine-modulated signal, for which $\zeta^{(1)} = \sin(\cdot)$, as this in effect achieves quadrature detection in the indirect dimension. For numerous multidimensional experiments, this can be achieved by repeating the pulse sequence, with careful adjustments to the phases of particular pulses, in a process referred to as *phase cycling*[24: Chapter 11]. Applying the same processing as that which achieved the double absorption spectrum for the cosine-modulated case generates a spectrum whose imaginary component features two peaks, but with opposite signs (panel c). It then becomes possible to generate a frequency discriminated spectrum by subtracting the sine spectrum from the cosine spectrum (panel d).

Phase-modulated signals

A "positive" phase-modulated signal with the form $\zeta^{(1)} = \exp(\mathrm{i}\cdot)$ (commonly referred to as *hypercomplex*) is frequency-discriminated due to its quadrature nature. However, direct FT of such a signal in both dimensions leads to a peak with a phase-twist lineshape, with no means of separating the absorption and dispersion contributions (panel e). Certain pulse sequences, including

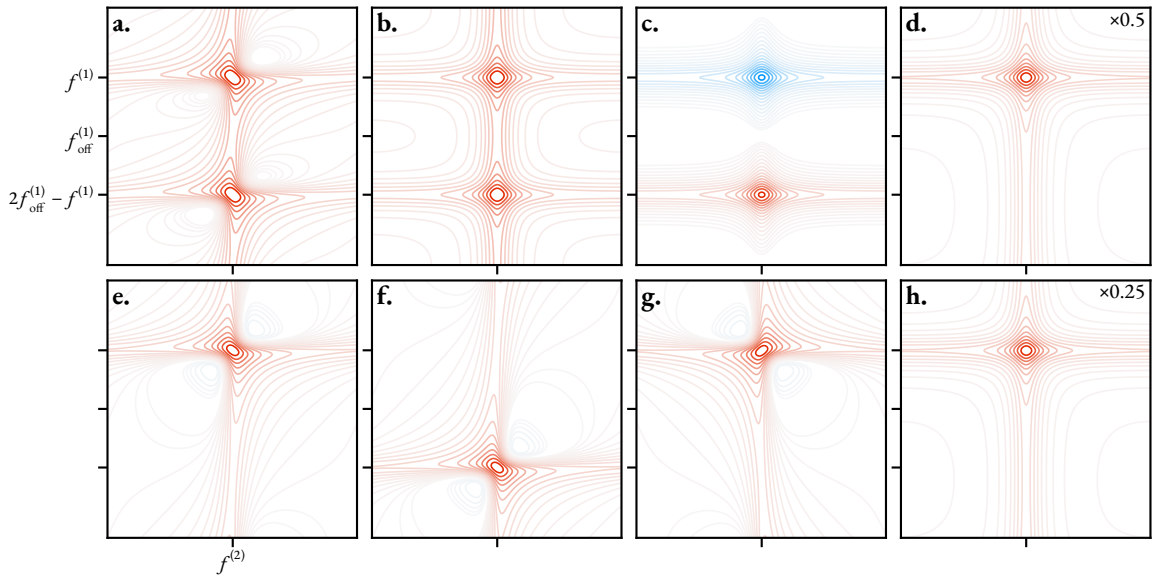


Figure 1.4: Spectra acquired from amplitude- and phase- modulated 2D signals. Red contour lines denote positive values, while blue contours denote negative values. **a.** The FT of a cosine-modulated FID, featuring peaks both at the true resonance frequency ($f^{(1)}$) and the mirrored frequency ($2f^{(1)}_{\text{off}}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

2DJ spectroscopy[31, 32] (see Chapter 4) and correlation spectroscopy (COSY)[11–13] produce hypercomplex datasets, and the conventional means of processing is simply to display the absolute value of the spectrum, which while removing the phase twist shapes, produces peaks with broad wings, due to the influence of the dispersive components. In scenarios where it is possible, it is desirable to acquire the equivalent "negative" signal ($\zeta^{(1)} = \exp(-\mathrm{i}\cdot)$), whose FT leads to a peak with the same phase twist form, but centered at $2f^{(1)}_{\mathrm{off}} - f^{(1)}$ (panel f). Inverting this spectrum in $F^{(1)}$ (panel g), and summing with the positive spectrum nullifies the dispersive contribution, generating a spectrum with an absorption lineshape[33] (panel h).

1.2.3 Estimation Techniques for NMR Analysis

Linear prediction

Linear prediction (LP)[34, 35] is a procedure which is now widely used in NMR data analysis, often with the intention of (a) propagating the FID further in time in order to reduce the presence of truncation artefacts and (b) correct the commonly corrupted initial points of the FID, as a means of improving the spectral baseline. The concept of LP stems from the idea that a deterministic signal, such as a 1D FID can be described as an autoregressive (AR) process, such that a given sample from the dataset can be described by a linear combination of an appropriate number L of previous samples:

$$y_n = \sum_{l=1}^{L} c_l y_{n-l} + e_l \tag{1.34}$$

$\forall n \in \{L, L+1, \cdots, N-1\}$, where $L \in \mathbb{Z}$ 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 account of error in the LP model. A datapoint can also be described by a linear combination of a certain number of subsequent points, using the set of *backward* LP coefficients $\boldsymbol{b} \in \mathbb{R}^L$:

$$y_n = \sum_{l=1}^{L} b_l y_{n+l} + e_l \tag{1.35}$$

$\forall n \in \{0, 1, \cdots, N-L-1\}$. Determining the LP coefficients enables the estimation of FID values beyond the data actually acquired ($n^{(1)} < 0$ and $n^{(1)} > N^{(1)} - 1$). It should be noted that (1.34) and (1.35) are only valid for FIDs without any corruption from experimental noise. Noisy datasets are instead an example of an autoregressive moving average (ARMA) process. Despite this, due to the greater simplicity of the AR model, it is far more common to employ this. The most common means of performing LP is by solving the Yule-Walker equations[36, 37], which describe the relationship between the signal autocorrelation coefficients and the LP coefficients[35: Section 3.3], with the Levinson-Durbin algorithm providing an efficient means of solving the equations[38,

39].

Stuff below here has been copy-pasted from old document. Will need editing!

Signal-Noise Subspace Separation Techniques

The Linear Prediction with Singular-Value Decomposition was proposed by Kumaresan and Tufts[40]. The core assumption of the method is that a given data point y_n may be expressed as a linear combination of L previous, or preceeding points:

$$\begin{aligned} y_n &= b_1 y_{n+1} + b_2 y_{n+2} + \cdots + b_L y_{n+L} \\ &= b_{-1} y_{n-1} + b_{-2} y_{n-2} + \cdots + b_{-L} y_{n-L} \end{aligned} \tag{1.36}$$

where $b_l (b_{-l})$, $l \in \{1, 2, \cdots, L\}$ are the backward(forward) prediction coeffiecients. For the backward prediction case, consider the follwing set of linear equations $\boldsymbol{A}\boldsymbol{b} = -\boldsymbol{b}$:

$$\begin{bmatrix} y_1^* & y_2^* & \cdots & y_L^* \\ y_2^* & y_3^* & \cdots & y_{L+1}^* \\ \vdots & \vdots & \ddots & \vdots \\ y_{N-L}^* & y_{N-L+1}^* & \cdots & y_{N-1}^* \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_L \end{bmatrix} = - \begin{bmatrix} y_0^* \\ y_1^* \\ \vdots \\ y_{N-L-1}^* \end{bmatrix} \tag{1.37}$$

This equation can be augmented into the form $\boldsymbol{A}'\boldsymbol{b}' = \boldsymbol{0}$, with $\boldsymbol{A}' = [\boldsymbol{b}|\boldsymbol{A}]$ and $\boldsymbol{b}' = [1, \boldsymbol{b}^{\mathrm{T}}]^{\mathrm{T}}$. For the noiseless data case (i.e. $y_n = x_n$), any row of $\boldsymbol{A}'$ can be written as the linear combination of J linearly independent vectors $\boldsymbol{f}_j$, $j \in \{1, 2, \cdots, J\}$[41]:

$$\boldsymbol{f}_j = \left[1, z_j^*, \left(z_j^2\right)^*, \cdots, \left(z_j^L\right)^*\right], \tag{1.38}$$

meaning that the rank of $\boldsymbol{A}'$ is J, provided it posseses at least J rows ($J \leqslant N - L$). The fact that $\boldsymbol{b}'$ lies in the null space of $\boldsymbol{A}'$ implies that $\boldsymbol{f}_j \boldsymbol{b}' = 0$, such that the following holds:

$$1 + b_1 z_j^* + b_2 \left(z_j^2\right)^* + \cdots + b_L \left(z_j^L\right)^* = 0 \tag{1.39}$$

It should be noted that L must satisfy $L \geqslant J$ to ensure the null space of $\boldsymbol{A}'$ has at least a dimension of 1. Eqn. 1.39 leads to a means of deriving the signal poles z_j via polynomial $B(\zeta)$:

$$B(\zeta) = 1 + b_1 \zeta^{-1} + b_2 \zeta^{-2} + \cdots + b_L \zeta^{-L}, \tag{1.40}$$

the zeros of which will occur whenever $\zeta = \left(z_j^{-1}\right)^*$, $j \in \{1, 2, \cdots, J\}$.

Of course, the prediction coefficient vector $\boldsymbol{b}$ still needs to be determined. Eqn. 1.37 implies

that the optimal vector can be determined by simple linear lest squares:

$$\hat{\boldsymbol{b}} = -\mathbf{A}^{+}\boldsymbol{h}. \tag{1.41}$$

However, noise components of the signal will be incorporated into the solution of $\boldsymbol{b}$. To avoid this, $\boldsymbol{A}$ is moved from the signal-noise subspace to the signal subspace via filtration using Singular-Valued Decomposition:

$$\boldsymbol{A} = \boldsymbol{U}\begin{bmatrix}\boldsymbol{\Sigma}\\ \mathbf{0}\end{bmatrix}\boldsymbol{V}^{\dagger} = \sum_{l=1}^{R}\sigma_l \boldsymbol{u}_l \boldsymbol{v}_l^{\dagger} \tag{1.42}$$

where R =
$mathrmrank(\boldsymbol{A}) = \min(L, N)$. The matrix $\tilde{\boldsymbol{A}}$ of rank J with the closest relationship to $\boldsymbol{A}$ is a Frobenius norm sense is given by[**Cadzow1988**]:

$$\tilde{\boldsymbol{A}} = \sum_{l=1}^{J}\sigma_l \boldsymbol{u}_l \boldsymbol{v}_l^{\dagger}. \tag{1.43}$$

Using the following relationship between a matrix's pseudoinverse and its SVD:

$$\boldsymbol{A}^{+} = \boldsymbol{V}\begin{bmatrix}\boldsymbol{\Sigma}^{+}\\ \mathbf{0}\end{bmatrix}\boldsymbol{U}^{\dagger}, \tag{1.44}$$

The optimal vector of prediction coeffiecients is determined by:

$$\hat{\boldsymbol{b}} = -\tilde{\boldsymbol{A}}^{+}\boldsymbol{h} = -\sum_{l=1}^{J}\sigma_l^{-1}\boldsymbol{v}_l\left[\boldsymbol{u}_l^{\dagger}\boldsymbol{h}\right] \tag{1.45}$$

Application: [42]

HSVD, MPM, only gloss over MPM as it is described in detail in Chapter 2

Iterative Methods

The variable projection (VARPRO) method is an iterative approach to parameter estimation, developed by Golub and Pereyra[43], and applied to NMR signal analysis by van der Veen et al.[44]. It relies on minimising the following quantity:

$$\hat{\theta} = \arg\min_{\theta} \|\boldsymbol{y} - \boldsymbol{x}(\theta)\|^2 \tag{1.46}$$

The vector $\boldsymbol{x}$ can be expressed in the following format:

$$\boldsymbol{x} = \begin{bmatrix} z_1^0 & z_2^0 & \cdots & z_J^0 \\ z_1^1 & z_2^1 & \cdots & z_J^1 \\ \vdots & \vdots & \ddots & \vdots \\ z_1^{N-1} & z_2^{N-1} & \cdots & z_J^{N-1} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_J \end{bmatrix} = \boldsymbol{Z}\boldsymbol{\alpha} \tag{1.47}$$

Using Eq. 1.47, Eq. 1.46 can be re-written as follows:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{\alpha}\|^2 \tag{1.48}$$

The complex amplitudes can be determined analytically as follows:

$$\hat{\boldsymbol{\alpha}} \approx \left(\boldsymbol{Z}^\dagger \boldsymbol{Z}\right)^{-1} \boldsymbol{Z}^\dagger \boldsymbol{y} \equiv \boldsymbol{Z}^+ \boldsymbol{y} \tag{1.49}$$

$\boldsymbol{Z}^+$ is the Moore-Penrose pseudoinverse of matrix $\boldsymbol{Z}$[45, 46]. The VARPRO method determines the frequencies and damping factors via an iterative minimisation of:

$$\left[\hat{\boldsymbol{f}^{\mathrm{T}}}, \boldsymbol{\eta}^{\mathrm{T}}\right]^{\mathrm{T}} = \arg\min_{[\boldsymbol{f}^{\mathrm{T}}, \boldsymbol{\eta}^{\mathrm{T}}]} \|\boldsymbol{y} - \boldsymbol{Z}\boldsymbol{Z}^+ \boldsymbol{y}\|^2 \tag{1.50}$$

which is achieved using the Levenberg–Marquardt algorithm[47, 48]. The amplitudes and phases are subsequently determined using Eq. 1.49. This means that the amplitudes and phases do not need initial guesses assocaited with them. Further specifications can be applied to the algorithm, reflecting the spectroscopist's knowledge of the signal under inspection. For example, if it is known that a certain set of resonances constitute a multiplet, the relative frequency differences and amplitudes will be known.

advanced method for accurate, robust, and efficient spectral fitting (AMARES)[49]

Bayesian Methods

complete reduction to amplitude frequency table (CRAFT)[50] 2D[51] perspective[52]

Machine Learning Methods

[53]

1.3 Overview of this work

1.3.1 Conception and motivation

Confirm whether this is accurate with Ali The initial motivation for this thesis came from discussions within the NMR Methodology group in Manchester involving my principal supervisor Dr. Mohammadali Foroozandeh and coworkers (notably Prof. Gareth Morris and Prof. Mathias Nilsson) when Dr. Foroozandeh was a Postdoctoral researcher there. There was an interest in generating pure shift NMR spectra from 2DJ spectra using an appropriate estimation and reconstruction algorithm. While little progress was made while Dr. Foroozandeh was in Manchester, he wished to continue with it after moving to Oxford to take up a research fellowship. I took on a project with a broader scope when I joined his nascent research group, starting with developing a routine and accompanying software for estimating 1D FIDs. Subsequently, a number of applications involving FID estimation were worked on, including the original 2DJ project.

1.3.2 Thesis Overview

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

- Chapter 2 discusses the theory behind routines which can be applied to determine parameter estimates which describe 1D and two-dimensional (2D) NMR datasets.
- Chapter 3 provides illustrations of the effectiveness of the estimation routine on numerous NMR datasets. Furthermore, means in which the established estimation routine can be extended for useful applications are explored, with these applications being:
  - The generation of pure shift spectra with featuring peaks with desirable lineshapes from 2DJ datasets (Section **??**).
  - Analysis of amplitude-attenuated datasets, such as those derived from diffusion and inversion recovery experiments (Section 3.2).
  - Overcoming quadratic phase behaviour and baseline distortions associated with excitation by a single frequency-swept pulse (Section 3.3).
- Chapter 5 describes the source code, called NMR-EsPy, which has been written to provide access to the described routines presented.

Theory

2

This chapter provides a detailed outline of an estimation routine which has been developed for the estimation of NMR data.

2.1 Outline of the Problem

For the purposes of this work, it is always assumed that an FID to be estimated $\boldsymbol{Y} \in \mathbb{C}^{N^{(1)} \times \cdots \times N^{(D)}}$ is hypercomplex in form, meaning that it obeys (1.21) with $\zeta^{(d)} = \exp(\mathrm{i}\cdot)\ \forall d \in \{1, \cdots D\}$:

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

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

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

where $\Delta_t^{(d)} = 1/f_{\mathrm{sw}}^{(d)}$. Under this model, it is assumed that an FID consists of a summation of M damped complex sinusoids in the presence in AWGN. It is the goal of parametric estimation to establish the identity of all the quantities which describe the model component $\boldsymbol{X}$, which are distilled into the vector $\boldsymbol{\theta} \in \mathbb{R}^{2(D+2)M}$, given by (1.20). An alternative, more concise, notation for (2.1c) involves the *complex amplitudes* and *signal poles* associated with the FID:

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

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

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

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

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

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

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

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

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

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

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

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

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

The application of *non-linear programming* (NLP) is a well-established approach to solve such a problem[54, 55]. The basic principle behind NLP is to iteratively explore, in a methodical way, how a function varies with its arguments. By using information about the function and optionally its derivatives, such a routine attempts to find a minimum in the function, and terminates once this has been achieved. While derivative-free approaches to NLP do exist,**(cite examples like simplex, simmulated annealing, BOBYQA?)** in scenarios where the function under consideration has well-defined, computationally tractable derivatives, the use of these can be valuable to solving optimisation problems. The problem outlined in Equation 2.7 is such an example.

In order for NLP to perform effectively, a large amount of *a priori* information is typically required, in the form of an initial guess $\boldsymbol{\theta}^{(0)}$. There is precedent for using NLP in the context of NMR data estimation, in which substantial user input defines the initial guess (see the description of VARPRO and AMARES in the previous chapter). As an alternative, a routine is presented here in which an initial guess is determined using a singular value decomposition (SVD)-based technique requiring no or minimal user input.

NLP acts in a "correcting" manner, rather than doing all the work with no initial information

Describe what is to come in this chapter...

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

2.2 Generating an Initial Guess: Matrix Pencil Method

Brief introductory text. Mention applications w. citations

2.2.1 1D Matrix Pencil Method

The matrix pencil method (MPM), developed by Hua and Sarkar[56–58], provides a route to extracting the signal poles of a 1D dataset, based on the assumption that the number or oscillators M is known. To motivate how the MPM works, first consider a dataset which is devoid of noise, given by (2.2) with $D = 1$:

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

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 comprising the segment shifted to the right by one point relative to the row above. $L \in \mathbb{N}$ is the *pencil parameter*, which dictates the size of each window. From $\boldsymbol{H_X}$, two matrices are defined, $\boldsymbol{H_{X1}}$ and $\boldsymbol{H_{X2}}$, formed by

the removal of the last or first column of $\boldsymbol{H}_{\boldsymbol{X}}$, respectively:

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

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

These matrices 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 following condition is met:

$$M \leq L \leq N - M, \tag{2.13}$$

the rank of the matrix pencil will be M. Equation (2.13) must be obeyed to ensure that both the number of rows and columns of the matrix pencil are at least M. Now consider the case when the

scalar λ is equal to one of the signal poles i.e. $\lambda = z_m \; \forall m \in \{1, \cdots, M\}$. The element $[\boldsymbol{Z}_\mathrm{D} - \lambda \boldsymbol{I}_M]_{m,m}$ will be set to 0, which will lead to the determinant of the matrix pencil being 0. The eigenvalues of the matrix pencil are the solution of the so-called *generalised eigenvalue problem*, and are defined as[59: Section 7.7]

$$\boldsymbol{z} = \{z \in \mathbb{C} : \det\left(\boldsymbol{H}_{\boldsymbol{X}2} - z\boldsymbol{H}_{\boldsymbol{X}1}\right) = 0\} \tag{2.14}$$

One means of finding the signal poles is by finding 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

$$\boldsymbol{\alpha} = \boldsymbol{Z}^{+}\boldsymbol{X}, \tag{2.15a}$$

$$\boldsymbol{Z} = \begin{bmatrix} \mathbf{1} & \boldsymbol{z} & \boldsymbol{z}^2 & \cdots & \boldsymbol{z}^{N-1} \end{bmatrix}^{\mathrm{T}}. \tag{2.15b}$$

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

$$\boldsymbol{a} = |\boldsymbol{\alpha}|, \tag{2.16a}$$

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

$$\boldsymbol{f} = \frac{f_\mathrm{sw}}{2\pi}\Im\left(\ln \boldsymbol{z}\right) + f_\mathrm{off}, \tag{2.16c}$$

$$\boldsymbol{\eta} = -f_\mathrm{sw}\Re\left(\ln \boldsymbol{z}\right). \tag{2.16d}$$

Noisy data

The presence of noise in the signal $\boldsymbol{Y}$ complicates the process of determining the M signal poles, as $\boldsymbol{H}_{\boldsymbol{Y}}$ ($\boldsymbol{H}_{\boldsymbol{X}}$'s equivalent with elements replaced by the noisy data) is likely to be full-rank ($\min(N - L, L+1)$). To cope with this, it is necessary to generate a rank-reduced matrix $\widetilde{\boldsymbol{H}}_{\boldsymbol{Y}}$. By employing the Eckart-Young-Mirsky (EYM) theorem[59: Section 2.2], an appropriate matrix is can be obtained through SVD (see Section A.1.1):

$$\widetilde{\boldsymbol{H}}_{\boldsymbol{Y}} = \boldsymbol{U}_M\boldsymbol{\Sigma}_M\boldsymbol{V}_M^{\dagger}, \tag{2.17a}$$

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

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

$$\mathbb{C}^{M\times M} \ni \boldsymbol{\Sigma}_M = \mathrm{diag}\left(\sigma_1, \sigma_2, \cdots, \sigma_M\right). \tag{2.17d}$$

σ_m is the m^{th} largest singular value of $\boldsymbol{H}_{\boldsymbol{Y}}$, $\boldsymbol{u}_m \in \mathbb{C}^{N^{(1)}-L^{(1)}}$ and $\boldsymbol{v}_m \in \mathbb{C}^{L^{(1)}+1}$ are the corresponding left and right singular vectors, respectively. The EYM 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.

$$\widetilde{\boldsymbol{H}}_{\boldsymbol{Y}} = \underset{\boldsymbol{A}:\ \mathrm{rank}(\boldsymbol{A})=M}{\arg\min} \|\boldsymbol{A} - \boldsymbol{H_Y}\|. \tag{2.18}$$

With a rank-reduced matrix produced from the noisy matrix, the signal poles can then be derived from 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 $\boldsymbol{H}_{\boldsymbol{X}1}$ and $\boldsymbol{H}_{\boldsymbol{X}2}$ do to $\boldsymbol{H_X}$. As a less expensive alternative, the same result can be achieved by computing the eigenvalues of $\boldsymbol{V}^{+}_{M1}\boldsymbol{V}_{M2}$, with

$$\boldsymbol{V}_{M1} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_{M-1} \end{bmatrix}, \tag{2.19a}$$

$$\boldsymbol{V}_{M2} = \begin{bmatrix} \boldsymbol{v}_2 & \boldsymbol{v}_3 & \cdots & \boldsymbol{v}_{M} \end{bmatrix}. \tag{2.19b}$$

Algorithm 2.1 provides a pseudo-code description of the MPM.

Discuss complexity of MPM, talking about reducing $N^{(1)}$ and $L^{(1)}$ as a means of reducing the cost.

2.2.2 2D Matrix Enhancement and Matrix Pencil Method

The MPM was extended for the consideration of 2D data by Hua with the matrix enhancement and matrix pencil method (MEMPM)[60]. The method centers around the enhanced matrix $\boldsymbol{E_Y} \in \mathbb{C}^{\left(L^{(1)}L^{(2)}\right)\times\left(N^{(1)}-L^{(1)}+1\right)\left(N^{(2)}-L^{(2)}+1\right)}$, a block Hankel matrix of the form

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

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

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

$$\boldsymbol{H}_{\boldsymbol{Y},n^{(1)}} = \boldsymbol{Z}^{(2)}_{\mathrm{L}}\boldsymbol{A}\boldsymbol{Z}^{(1)^{n^{(1)}}}_{\mathrm{D}}\boldsymbol{Z}^{(2)}_{\mathrm{R}}. \tag{2.21}$$

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

procedure MPM($\boldsymbol{y} \in \mathbb{C}^N, f_{\text{sw}} \in \mathbb{R}_{>0}, f_{\text{off}} \in \mathbb{R}, M \in \mathbb{N}_0$)
  $L \leftarrow \lfloor N/3 \rfloor$;
  $\boldsymbol{H_y} \leftarrow \begin{bmatrix} y_0 & y_1 & \cdots & y_L \\ y_1 & y_2 & \cdots & y_{L+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N-L-1} & y_{N-L} & \cdots & y_{N-1} \end{bmatrix}$;
  $\boldsymbol{U}, \boldsymbol{\sigma}, \boldsymbol{V}^\dagger \leftarrow \text{SVD}(\boldsymbol{H_y})$;
  if $M = 0$ **then**
    $M \leftarrow \text{MDL}(\boldsymbol{\sigma}, L, N)$;
  end if
  $\boldsymbol{V} \leftarrow \left[\boldsymbol{V}^\dagger\right]^\dagger$;
  $\boldsymbol{V}_M \leftarrow \boldsymbol{V}\left[:, : M\right]$; ▷ Retain first M right singular vectors.
  $\boldsymbol{V}_{M1}, \boldsymbol{V}_{M2} \leftarrow \boldsymbol{V}_M\left[:, : M-1\right], \boldsymbol{V}_M\left[:, 1 :\right]$; ▷ Remove last/first column
  $\boldsymbol{z} \leftarrow \text{Eigenvalues}\left(\boldsymbol{V}_{M1}^{+}\boldsymbol{V}_{M2}\right)$;
  $\boldsymbol{Z} \leftarrow \begin{bmatrix} \boldsymbol{1} & \boldsymbol{z} & \boldsymbol{z}^2 & \cdots & \boldsymbol{z}^N \end{bmatrix}^{\mathrm{T}}$;
  $\boldsymbol{\alpha} \leftarrow \boldsymbol{Z}^{+}\boldsymbol{Y}$;
  $\boldsymbol{a} \leftarrow |\boldsymbol{\alpha}|$;
  $\boldsymbol{\phi} \leftarrow \arctan\left(\frac{\Im(\boldsymbol{\alpha})}{\Re(\boldsymbol{\alpha})}\right)$;
  $\boldsymbol{f} \leftarrow \frac{f_{\text{sw}}}{2\pi}\Im\left(\ln \boldsymbol{z}\right) + f_{\text{off}}$;
  $\boldsymbol{\eta} \leftarrow -f_{\text{sw}}\Re\left(\ln \boldsymbol{z}\right)$;
  if $\boldsymbol{\eta}$ contains negative values **then** ▷ Purge any oscillators with negative damping
    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($\boldsymbol{\sigma} \in \mathbb{R}^{L+1}, L \in \mathbb{N}, N \in \mathbb{N}$)
  $\text{MDL}_{k-1} \leftarrow \infty$ ▷ Ensure that on first iteration ($k = 0$), the conditional does not hold
  for $k = 0, \cdots, L$ **do**
    $\text{MDL}_k \leftarrow -\ln\left(\frac{\prod_{r=k}^{L-1} \sigma_{r+1}^{1/L-k}}{\frac{1}{L-k}\sum_{r=k}^{L-1}\sigma_{r+1}}\right)^{(L-k)N}$;
    if $\text{MDL}_k > \text{MDL}_{k-1}$ **then**
      $M \leftarrow k - 1$;
      break;
    else
      $\text{MDL}_{k-1} \leftarrow \text{MDL}_k$;
    end if
  end for
  return M;
end procedure

This then leads to the enhanced matrix be expressed as

$$\boldsymbol{E}_{\boldsymbol{Y}} = \boldsymbol{E}_{\mathrm{L}} \boldsymbol{\mathcal{A}} \boldsymbol{E}_{\mathrm{R}}, \tag{2.22a}$$

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

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

As was the case in the 1D MPM, SVD can be utilised to generate a filtered matrix $\widetilde{\boldsymbol{E}}_{\boldsymbol{Y}}$ with its rank reducd to M, in accordance witht he EYM theorem:

$$\widetilde{\boldsymbol{E}}_{\boldsymbol{Y}} = \boldsymbol{U}_M \boldsymbol{\Sigma}_M \boldsymbol{V}_M^{\dagger} \tag{2.23}$$

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

$$\boldsymbol{U}_M = \boldsymbol{E}_{\mathrm{L}} \boldsymbol{T}. \tag{2.24}$$

Now consider the following two matrices:

$$\boldsymbol{U}_{M1} = \boldsymbol{E}_{\mathrm{L1}} \boldsymbol{T}, \tag{2.25a}$$

$$\boldsymbol{U}_{M2} = \boldsymbol{E}_{\mathrm{L1}} \boldsymbol{Z}_{\mathrm{D}}^{(1)} \boldsymbol{T}, \tag{2.25b}$$

$$\mathbb{C}^{L^{(1)}\left(L^{(2)}-1\right) \times M} \ni \boldsymbol{E}_{\mathrm{L1}} = \begin{bmatrix} \boldsymbol{Z}_{\mathrm{L}}^{(2)} \\ \boldsymbol{Z}_{\mathrm{L}}^{(2)} \boldsymbol{Z}_{\mathrm{D}}^{(1)} \\ \vdots \\ \boldsymbol{Z}_{\mathrm{L}}^{(2)} \left[\boldsymbol{Z}_{\mathrm{D}}^{(1)}\right]^{L^{(1)}-2} \end{bmatrix}. \tag{2.25c}$$

$\boldsymbol{U}_{M1}$ 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

$$\boldsymbol{U}_{M1} - \lambda \boldsymbol{U}_{M2} = \boldsymbol{E}_{\mathrm{L1}} \left(\boldsymbol{Z}_{\mathrm{D}}^{(1)} - \lambda \boldsymbol{I}_M \right) \boldsymbol{T}. \tag{2.26}$$

As seen previously, this matrix structure implies that the elements of $\boldsymbol{z}^{(1)}$ are the solutions to the generalised eigenvalue problem, 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 is defined:

$$\mathbb{R}^{L^{(1)}L^{(2)}\times L^{(1)}L^{(2)}} \ni \boldsymbol{P} = \begin{bmatrix} \boldsymbol{e}(0)^{\mathrm{T}} \\ \boldsymbol{e}\left(L^{(2)}\right)^{\mathrm{T}} \\ \vdots \\ \boldsymbol{e}\left(\left(L^{(1)}-1\right)L^{(2)}\right)^{\mathrm{T}} \\ \boldsymbol{e}(1)^{\mathrm{T}} \\ \boldsymbol{e}\left(1+L^{(2)}\right)^{\mathrm{T}} \\ \vdots \\ \boldsymbol{e}\left(1+\left(L^{(1)}-1\right)L^{(2)}\right)^{\mathrm{T}} \\ \vdots \\ \vdots \\ \boldsymbol{e}\left(L^{(2)}-1\right)^{\mathrm{T}} \\ \boldsymbol{e}\left(2L^{(2)}-1\right)^{\mathrm{T}} \\ \vdots \\ \boldsymbol{e}\left(L^{(1)}L^{(2)}-1\right)^{\mathrm{T}} \end{bmatrix}. \tag{2.27}$$

$\boldsymbol{e}(i) \in \mathbb{R}^{L^{(1)}L^{(2)}}$ corresponds to a unit vector comprising zeros except for $e_i = 1\,\forall i \in \{0,\cdots,L^{(1)}L^{(2)}-1\}$. 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:

$$\boldsymbol{E}_{\mathrm{LP}} := \boldsymbol{P}\boldsymbol{E}_{\mathrm{L}} = \begin{bmatrix} \boldsymbol{Z}_{\mathrm{L}}^{(1)} \\ \boldsymbol{Z}_{\mathrm{L}}^{(1)}\boldsymbol{Z}_{\mathrm{D}}^{(2)} \\ \vdots \\ \boldsymbol{Z}_{\mathrm{L}}^{(1)}\boldsymbol{Z}_{\mathrm{D}}^{(2)^{L^{(2)}-1}} \end{bmatrix}. \tag{2.28}$$

Note the similarity of Equation 2.28 with Equation 2.22b, which implies that with the same reasoning as given 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 involved employing a pairing algorithm in order to assign the uncorrelated signal poles in $\boldsymbol{z}^{(1)}$ with $\boldsymbol{z}^{(2)}$ [60]. The modified matrix enhancement and matrix pencil method (MMEMPM) was developed in order to overcome two issues with the pairing algorithm: (a) it is computationally expensive (b) it is fallible to return incorrect pairings[61]. As well as the eigenvalues of $\boldsymbol{U}_{M1}^{+}\boldsymbol{U}_{M2}$, the MMEMPM requires extraction

of the eigenvectors too, which are contained in the matrix $\boldsymbol{W}^{(1)}$. Assuming that there are no repeated poles in $\boldsymbol{z}^{(1)}$, the correctly paired second set of poles is then generated via

$$\boldsymbol{z}^{(2)} = \text{diag}\left(\boldsymbol{W}^{-1}\boldsymbol{U}_{M\text{P1}}^{+}\boldsymbol{U}_{M\text{P2}}\boldsymbol{W}\right) \tag{2.29}$$

Talk about case of paired eigenvalues. See Algorithm A.1

2.2.3 Model Order Selection

Up to this point, it has been assumed that the model order M is known, or at least has been predicted. For a typical FID, the M is unknown. It is possible that a individual inspecting the corresponding 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. **Mention use of threshold and retaining SVs that are higher than this.** 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)[62] and minimum description length (MDL)[63, 64]. 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 (2.1c), with variable M. Both the AIC and MDL take the same general form:

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

$\forall k \in \{0, 1, \cdots\}$. $\mathcal{L}\left(\boldsymbol{\theta}^{(*)}|\boldsymbol{Y}\right)$ is the likelihood function of a given model, with model order k, at the MLE $\boldsymbol{\theta}^{(*)}$, $c \in \mathbb{R}$ is a scaling constant, and $\mathcal{P}$ is a penalising function, which acts to correct for bias. As the model order increases, the likelihood function at the MLE will increase in size, as a model with more parameters will be able to fit a given dataset more accurately. However, as the model order increases, there will become a point where practically all of the deterministic part of the signal has been incorporated into the model, and increasing the model order further leads to the model also accounting for noise. The penalising term, which is larger for higher k, is required in order to estimate a model order which is parsimonious. Wax and Kailath derived an expression for the likelihood at the MLE for models comprising a summation of complex sinusoids[65]*:

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

*The expression in original paper considers the eigenvalues of the covariance matrix for the signal, rather than the singular values of $\boldsymbol{H}_{\boldsymbol{Y}}$. These are equivalent however.

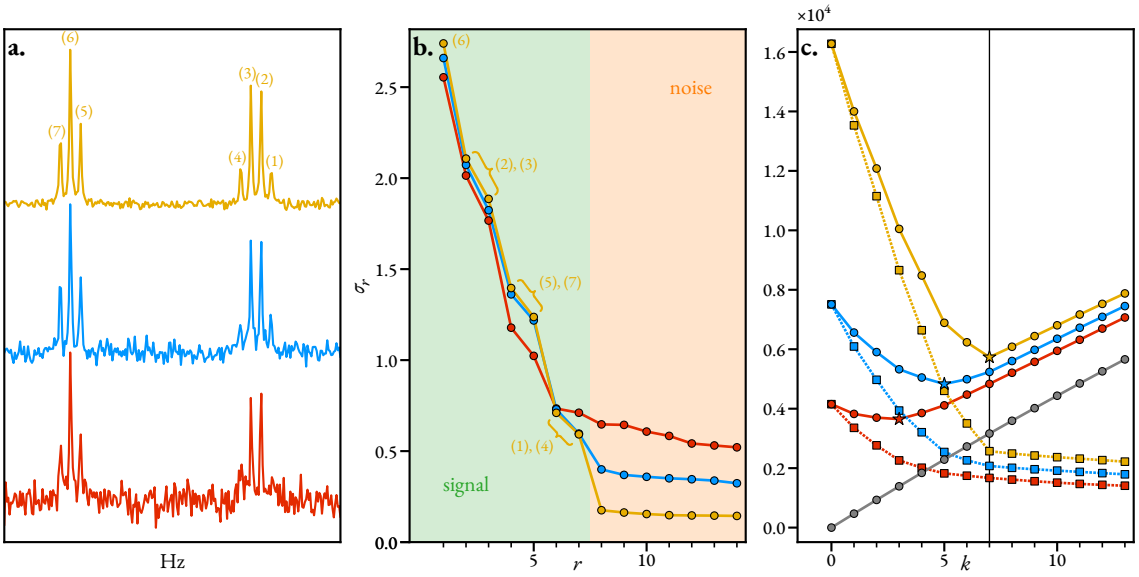


Figure 2.1: A visualisation of the behaviour of the MDL for three different FIDs comprising the same deterministic component ($\boldsymbol{x}$) but with different noise instances of differing variances. The model used to construct the FIDs feature 7 oscillators. The three SNRs used were 7 dB (red), 12 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 $\boldsymbol{H}_{\boldsymbol{Y}}$, with the pencil parameter L set to $\lfloor N/3 \rfloor = 85$. **c.** Square points with dotted lines: The negative log-likeliood at the MLE, i.e. the first term of (2.32b). Grey line: the penalty component of the MDL, given by the second term in (2.32b). Circular points with solid lines: the MDL. Stars denote the minimum of the MDL for a given FID. The 20 dB signal is correctly deemed to have a model order of 7, while the other two are underestimated (predicted models orders are 5 and 3 for the 12 dB and 7 dB FIDs, respectively).

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

$$\mathrm{AIC}(k) = -2 \ln \left(\mathcal{L} \left(\hat{\boldsymbol{\theta}} | \boldsymbol{Y} \right) \right) + 2k(2L - k), \tag{2.32a}$$

$$\mathrm{MDL}(k) = -\ln \left(\mathcal{L} \left(\hat{\boldsymbol{\theta}} | \boldsymbol{Y} \right) \right) + \tfrac{1}{2} k(2L - k) \ln N. \tag{2.32b}$$

The AIC has been shown to be inconsistent in that it tends to overestimate the model order as the number of samples increases[65]. For this reason, the MDL has found greater favour in signal processing applications. As such, by default the estimation routine employed in this work utilises the MDL:

$$M = \underset{k \in \mathbb{N}_0 :\, k < L}{\arg\min}\, \mathrm{MDL}(k). \tag{2.33}$$

Figure 2.1 illustrates the form of the MDL for three signals with equivalent underlying models, with $M = 7$, and noise instances with different variances. The first 14 singular values of $\boldsymbol{H}_{\boldsymbol{Y}}$ are plotted in panel b, where it can be seen that beyond the first 7, which account for signal components, the subsequent singular values, decrease at a far slower rate. The noise subspace for FIDs

with higher SNRs have singular values which are (a) smaller in magnitude and (b) more consistent, such that distinguishing the noise and signal subspaces is an easier task (cf. the yellow and red lines in panel b). As such, the MDL is more likely to provide a faithful estimate of the true number of components in the FID (panel c) when the SNR is higher.

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

2.3 Non-linear programming for NMR estimation

Before discussing the details of the specific NLP problem outlined above, an general overview is given.

2.3.1 An overview of non-linear programming

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

$$\boldsymbol{\theta}^{(*)} = \underset{\boldsymbol{\theta} \in \mathbb{R}^n}{\arg\min}\, \mathcal{F}(\boldsymbol{\theta}). \tag{2.34}$$

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

$$\mathcal{F}\left(\boldsymbol{\theta}^{(*)}\right) \leq \mathcal{F}(\boldsymbol{\theta}) \; \forall \boldsymbol{\theta} \in V. \tag{2.35}$$

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

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

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

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 $\boldsymbol{H}\left(\boldsymbol{\theta}^{(*)}\right) := \nabla^2\mathcal{F}\left(\boldsymbol{\theta}^{(*)}\right)$

[†]In certain applications, the interest is actually in finding the maximum of a function. However, it is trivial to transform a maximisation problem into a minimisation problem by finding the minimum of negative of the function.
[‡]A convex function is one such that a line segment through any two points of the function lies above it.

is positive semidefinite, i.e.

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

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

$$\boldsymbol{v}^{\mathrm{T}}\boldsymbol{H}\left(\boldsymbol{\theta}^{(*)}\right)\boldsymbol{v} > 0 \;\forall \boldsymbol{v} \in \mathbb{R}^n. \tag{2.38}$$

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

$$\mathcal{F}_Q\left(\boldsymbol{\theta}\right) = \mathcal{F}\left(\boldsymbol{\theta}^{(k)}\right) + \boldsymbol{h}^{\mathrm{T}}\boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right) + \frac{1}{2}\boldsymbol{h}^{\mathrm{T}}\boldsymbol{H}\left(\boldsymbol{\theta}^{(k)}\right)\boldsymbol{h}, \tag{2.39}$$

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:

$$\begin{aligned} \frac{\partial \mathcal{F}(\boldsymbol{\theta})}{\partial \boldsymbol{h}} &= \boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right) + \boldsymbol{H}\left(\boldsymbol{\theta}^{(k)}\right)\boldsymbol{h} \\ \implies 0 &= \boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right) + \boldsymbol{H}\left(\boldsymbol{\theta}^{(k)}\right)\left(\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\right) \\ \implies \boldsymbol{\theta}^{(k+1)} &= \boldsymbol{\theta}^{(k)} - \left[\boldsymbol{H}\left(\boldsymbol{\theta}^{(k)}\right)\right]^{-1}\boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right). \end{aligned} \tag{2.40}$$

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

$$\left\|\boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right)\right\| \leq \epsilon. \tag{2.41}$$

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

Line search methods[55: 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[55: Chapter 4] define a radius $\Delta^{(k)} > 0$, and determine the minimum of (2.39) subject to the constraint that $\|\boldsymbol{h}\| \leq \Delta^{(k)}$.

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

Trust Region Methods

Algorithm 2.2 Nonlinear programming routine employed in NMR-EsPy. This makes use of Algorithms 4.1 & 7.2 in [55], with a extra check inserted to deal with any negative-amplitude oscillators which may be generated as the routine evolves.

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: $\Delta^{(0)} \leftarrow {}^{1}/_{10} \left\| \boldsymbol{g}\left(\boldsymbol{\theta}^{(0)} \mid \boldsymbol{Y}\right) \right\|$;
3: $\Delta_{\max} \leftarrow 16\Delta^{(0)}$;
4: **for** $k = 0, 1, \cdots$ **do**
5: $\boldsymbol{p}^{(k)} \leftarrow \textsc{SteihaugToint}\left(\boldsymbol{Y}, \boldsymbol{\theta}^{(k)}, \Delta^{(k)}\right)$; ▷ See Algorithm A.2
6: $\rho^{(k)} \leftarrow \frac{\mathcal{F}_{\not\Phi}\left(\boldsymbol{\theta}^{(k)}\right) - \mathcal{F}_{\not\Phi}\left(\boldsymbol{\theta}^{(k)} + \boldsymbol{p}^{(k)}\right)}{\mathcal{F}_{\not\Phi Q}\left(\boldsymbol{\theta}^{(k)}\right) - \mathcal{F}_{\not\Phi Q}\left(\boldsymbol{\theta}^{(k)} + \boldsymbol{p}^{(k)}\right)}$;
7: **if** $\rho_k < {}^{1}/_{4}$ **then**
8: $\Delta^{(k+1)} \leftarrow {}^{1}/_{4}\Delta^{(k)}$;
9: **else if** $\rho_k > {}^{3}/_{4}$ **and** $\left\| \boldsymbol{p}^{(k)} \right\| = \Delta^{(k)}$ **then**
10: $\Delta^{(k+1)} \leftarrow \min\left(2\Delta^{(k)}, \Delta_{\max}\right)$;
11: **else**
12: $\Delta^{(k+1)} \leftarrow \Delta^{(k)}$;
13: **end if**
14: **if** $\rho^{(k)} > {}^{3}/_{20}$ **then**
15: $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} + \boldsymbol{p}^{(k)}$;
16: **else**
17: $\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)}$;
18: **end if**
19: **if** $k \bmod 25 = 0$ **and** $\boldsymbol{\theta}^{(k+1)}$ contains negative amplitudes **then**
20: $\boldsymbol{\theta}^{(0)} \leftarrow \boldsymbol{\theta}^{(k+1)}$ with negative-amplitude oscillators removed;
21: $\boldsymbol{\theta}^{(*)}, \boldsymbol{\epsilon}^{(*)} \leftarrow \text{NLP}\left(\boldsymbol{Y}, \boldsymbol{\theta}^{(0)}\right)$;
22: **end if**
23: **if** $\left\| \boldsymbol{g}\left(\boldsymbol{\theta}^{(k+1)}\right) \right\| < 10^{-8}$ **then**
24: **break**;
25: **end if**
26: **end for**
27: $\boldsymbol{\theta}^{(*)} \leftarrow \boldsymbol{\theta}^{(k+1)}$
28: $\boldsymbol{\epsilon}^{(*)} \leftarrow \sqrt{\frac{\mathcal{F}\left(\boldsymbol{\theta}^{(*)}\right) \operatorname{diag}\left(\left[\boldsymbol{H}\left(\boldsymbol{\theta}^{(*)}\right)\right]^{-1}\right)}{\left(N^{(1)} \cdots N^{(D)}\right) - 1}}$
29: **return** $\boldsymbol{\theta}^{(*)}, \boldsymbol{\epsilon}^{(*)}$;
30: **end procedure**

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

$$\boldsymbol{p}^{(k)} = \underset{\boldsymbol{p} \in \mathbb{R}^n}{\arg\min}\, \mathcal{F}\left(\boldsymbol{\theta}^{(k)}\right) + \boldsymbol{p}^{\mathrm{T}} \boldsymbol{g}\left(\boldsymbol{\theta}^{(k)}\right) + \tfrac{1}{2} \boldsymbol{p}^{\mathrm{T}} \boldsymbol{H}\left(\boldsymbol{\theta}^{(k)}\right) \boldsymbol{p} \quad \text{subject to } \|\boldsymbol{p}\| \leq \Delta^{(k)}. \tag{2.42}$$

This sub-problem is not usually minimised exactly, but instead an efficient means of determining

a sufficiently good value for $\boldsymbol{p}^{(k)}$ is used. Common approaches include computing the Cauchy point and the Dogleg method, and a truncated conjugate-gradient approach, also known as the Steihaug-Toint (ST) method[55: Chapter 7], which is employed in this work (Algorithm A.2). In the ST approach, iterates of the conjugate-gradient method are used, until either they go beyond the trust region, or negative curvature is discovered.

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

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

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

2.3.2 Non-linear programming applied to FID estimation

Remark 2. *Prior to estimating the dataset, it is normalised, such that the signal actually under consideration is* $\boldsymbol{Y}/\|\boldsymbol{Y}\|$. *To make the result reflect the actual dataset, the final amplitudes* $\boldsymbol{a}^{(*)}$ *are multiplied by* $\|\boldsymbol{Y}\|$.

Focussing now on the problem of FID estimation, the fidelity $\mathcal{F}\left(\boldsymbol{\theta} \,|\, \boldsymbol{Y}\right) : \mathbb{C}^{N^{(1)} \times \cdots \times N^{(D)}} \times \mathbb{R}^{2(1+D)M} \longrightarrow \mathbb{R}$ is given by

$$\mathcal{F}\left(\boldsymbol{\theta} \,|\, \boldsymbol{Y}\right) = \left\|\boldsymbol{Y} - \boldsymbol{X}\left(\boldsymbol{\theta}\right)\right\|^2. \tag{2.44}$$

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

$$g_i = -2\Re\left\langle \left(\boldsymbol{Y} - \boldsymbol{X}\right), \frac{\partial \boldsymbol{X}}{\partial \theta_i} \right\rangle, \tag{2.45a}$$

$$h_{i,j} = 2\Re\Bigg(\underbrace{\left\langle \frac{\partial \boldsymbol{X}}{\partial\theta_i}, \frac{\partial \boldsymbol{X}}{\partial\theta_j}\right\rangle}_{①} - \underbrace{\left\langle (\boldsymbol{Y}-\boldsymbol{X}), \frac{\partial^2 \boldsymbol{X}}{\partial\theta_i\partial\theta_j}\right\rangle}_{②}\Bigg). \tag{2.45b}$$

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

$$\frac{\partial x}{\partial\theta_m} \equiv \frac{\partial x}{\partial a_m} = \frac{x}{a_m}, \tag{2.46a}$$

$$\frac{\partial x}{\partial\theta_{m+M}} \equiv \frac{\partial x}{\partial\phi_m} = \mathrm{i}x, \tag{2.46b}$$

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

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

$$\frac{\partial^2 x}{\partial\theta_m^2} \equiv \frac{\partial^2 x}{\partial a_m^2} = 0, \tag{2.46e}$$

$$\frac{\partial^2 x}{\partial\theta_m\partial\theta_{m+M}} \equiv \frac{\partial^2 x}{\partial a_m\partial\phi_m} = \frac{\mathrm{i}x}{a_m}, \tag{2.46f}$$

$$\frac{\partial^2 x}{\partial\theta_m\partial\theta_{m+(d+1)M}} \equiv \frac{\partial^2 x}{\partial a_m\partial f_m^{(d)}} = \frac{2\pi\mathrm{i}\Delta_t^{(d)} n^{(d)} x}{a_m}, \tag{2.46g}$$

$$\frac{\partial^2 x}{\partial\theta_m\partial\theta_{m+(d+D+1)M}} \equiv \frac{\partial^2 x}{\partial a_m\partial\eta_m^{(d)}} = \frac{-\Delta_t^{(d)} n^{(d)} x}{a_m}, \tag{2.46h}$$

$$\frac{\partial^2 x}{\partial\theta_{m+M}^2} \equiv \frac{\partial^2 x}{\partial\phi_m^2} = -x, \tag{2.46i}$$

$$\frac{\partial^2 x}{\partial\theta_{m+M}\partial\theta_{m+(d+1)M}} \equiv \frac{\partial^2 x}{\partial\phi_m\partial f_m^{(d)}} = -2\pi\Delta_t^{(d)} n^{(d)} x, \tag{2.46j}$$

$$\frac{\partial^2 x}{\partial\theta_{m+M}\partial\theta_{m+(d+D+1)M}} \equiv \frac{\partial^2 x}{\partial\phi_m\partial\eta_m^{(d)}} = -\mathrm{i}\Delta_t^{(d)} n^{(d)} x, \tag{2.46k}$$

$$\frac{\partial^2 x}{\partial\theta_{m+(d+1)M}\partial\theta_{m+(d'+1)M}} \equiv \frac{\partial^2 x}{\partial f_m^{(d)}\partial f_m^{(d')}} = -4\pi^2\left(\Delta_t^{(d)} n^{(d)}\right)\left(\Delta_t^{(d')} n^{(d')}\right) x, \tag{2.46l}$$

$$\frac{\partial^2 x}{\partial\theta_{m+(d+1)M}\partial\theta_{m+(d'+D+1)M}} \equiv \frac{\partial^2 x}{\partial f_m^{(d)}\partial\eta_m^{(d')}} = -2\pi\mathrm{i}\left(\Delta_t^{(d)} n^{(d)}\right)\left(\Delta_t^{(d')} n^{(d')}\right) x, \tag{2.46m}$$

$$\frac{\partial^2 x}{\partial\theta_{m+(d+D+1)M}\partial\theta_{m+(d'+D+1)M}} \equiv \frac{\partial^2 x}{\partial\eta_m^{(d)}\partial\eta_m^{(d')}} = \left(\Delta_t^{(d)} n^{(d)}\right)\left(\Delta_t^{(d')} n^{(d')}\right) x, \tag{2.46n}$$

$$\frac{\partial^2 x}{\partial\theta_i\partial\theta_j} = \frac{\partial^2 x}{\partial\theta_j\partial\theta_i}, \tag{2.46o}$$

$$\frac{\partial^2 x}{\partial\theta_i\partial\theta_j} = 0 \text{ if not specified above.} \tag{2.46p}$$

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	$2(1+D)MN^{(1)}\cdots N^{(D)}$	$((1+D)(2(1+D)+1)-1)MN^{(1)}\cdots N^{(D)}$

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

(2.46p) indicates that any second derivative with respect to two parameters which do not belong to the same oscillator will always be 0. This, along with the symmetrical nature of the second derivatives, see (2.46o), drastically reduces the required number of second derivatives to compute, from $4(1+D)^2M^2$ per data-point to $(1+D)\,(3+2D)\,M$. Finally, (2.46e) indicates that another M second derivatives do not need to be computed. See Table 2.1 for the total number of derivatives that need to be computed for signals with different numbers of dimensions.

2.3.3 Approximating the Hessian

Despite many of the model second derivatives being 0, computation of those that are not zero, and subsequently using these the form the Hessian matrix, is often the most computationally expensive part of the optimisation. There are a large number of optimisation problems where this is the case, and as such there is considerable precedent for improving the efficiency of optimisation algorithms by generating approximations of the Hessian which are less expensive. Examples include the Gauss-Newton (GN) method and Levenberg-Marquardt (LM) algorithm, which are specifically for residual sum-of-squares problems[55: Chapter 10], as well as quasi-Newton methods such as the BFGS method[55: Chapter 6].

The GN and LM approaches replace the true Hessian matrix at each iteration with the following expression:

$$h_{i,j} \approx 2\Re\left\langle \frac{\partial \boldsymbol{X}}{\partial \theta_i}, \frac{\partial \boldsymbol{X}}{\partial \theta_j} \right\rangle, \tag{2.47}$$

i.e. term ② in Equation 2.45b involving the second derivatives is neglected. All that needs to be generated is the Jacobian $\boldsymbol{J} = {}^{\partial \boldsymbol{X}}/_{\partial \theta}$. This often brings a very large reduction in the computational cost, as no extra derivatives need to be computed for the Hessian at all, since the Jacobian is already required for generating the gradient vector. In situations where the residuals between the data and model are small, term ① will tend to dominate term ②, and as such these methods often enjoy a convergence rate close to that of Newton's method when close to local minima. Despite this, by

invoking this approximation, the rate of convergence, i.e. the number of iterations required to reach $\boldsymbol{\theta}^{(*)}$, tends to be adversely affected. See Section 2.3.5 for an example of this phenomenon.

2.3.4 Estimation Errors

The vector of standard errors associated with the NLP routine is related to the *observed Fisher information matrix* at convergence[66: Section 2.7]:

$$\boldsymbol{\epsilon}\left(\boldsymbol{\theta}^{(*)}\right) = \sqrt{\operatorname{diag}\left(\boldsymbol{I}\left(\boldsymbol{\theta}^{(*)}\right)^{-1}\right)}, \tag{2.48}$$

where the observed Fisher Information matrix contains the negative partial second derivatives of the log-likelihood with respect to $\boldsymbol{\theta}$:

$$\boldsymbol{I}\left(\boldsymbol{\theta}\right)_{i,j} = -\frac{\partial^2 \ell\left(\boldsymbol{\theta} \mid \boldsymbol{Y}\right)}{\partial\theta_i \partial\theta_j}. \tag{2.49}$$

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

$$\boldsymbol{I}\left(\boldsymbol{\theta}\right)_{i,j} = -\frac{1}{\sigma^2}\Re\left(\left\langle\frac{\partial \boldsymbol{X}}{\partial\theta_i}, \frac{\partial \boldsymbol{X}}{\partial\theta_j}\right\rangle - \left\langle\left(\boldsymbol{Y}-\boldsymbol{X}\right), \frac{\partial^2 \boldsymbol{X}}{\partial\theta_i \partial\theta_j}\right\rangle\right), \tag{2.50}$$

which very closely resembles the Hessian of $\boldsymbol{\theta}$:

$$\boldsymbol{I}\left(\boldsymbol{\theta}\right)_{i,j} = \frac{1}{2\sigma^2}\left(\boldsymbol{H}\left(\boldsymbol{\theta}\right)_{i,j}\right). \tag{2.51}$$

The standard errors therefore take the form

$$\boldsymbol{\epsilon}\left(\boldsymbol{\theta}^{(*)}\right) = \sqrt{2\sigma^2 \operatorname{diag}\left(\boldsymbol{H}\left(\boldsymbol{\theta}^{(*)}\right)^{-1}\right)}. \tag{2.52}$$

Considering that the mean and variance of the noise $\boldsymbol{W}$ are 0 and $2\sigma^2$, respectively:

$$2\sigma^2 = \frac{1}{\Re - 1}\left\|\boldsymbol{W}\right\|^2 = \frac{1}{\Re - 1}\left\|\boldsymbol{Y} - \boldsymbol{X}\left(\boldsymbol{\theta}^{(*)}\right)\right\|^2, \tag{2.53}$$

so that finally a useable expression for the standard errors is arrived at:

$$\boldsymbol{\epsilon}\left(\boldsymbol{\theta}^{(*)}\right) = \sqrt{\frac{\mathcal{F}\left(\boldsymbol{\theta}^{(*)}\right) \operatorname{diag}\left(\left[\boldsymbol{H}\left(\boldsymbol{\theta}^{(*)}\right)\right]^{-1}\right)}{\Re - 1}} \tag{2.54}$$

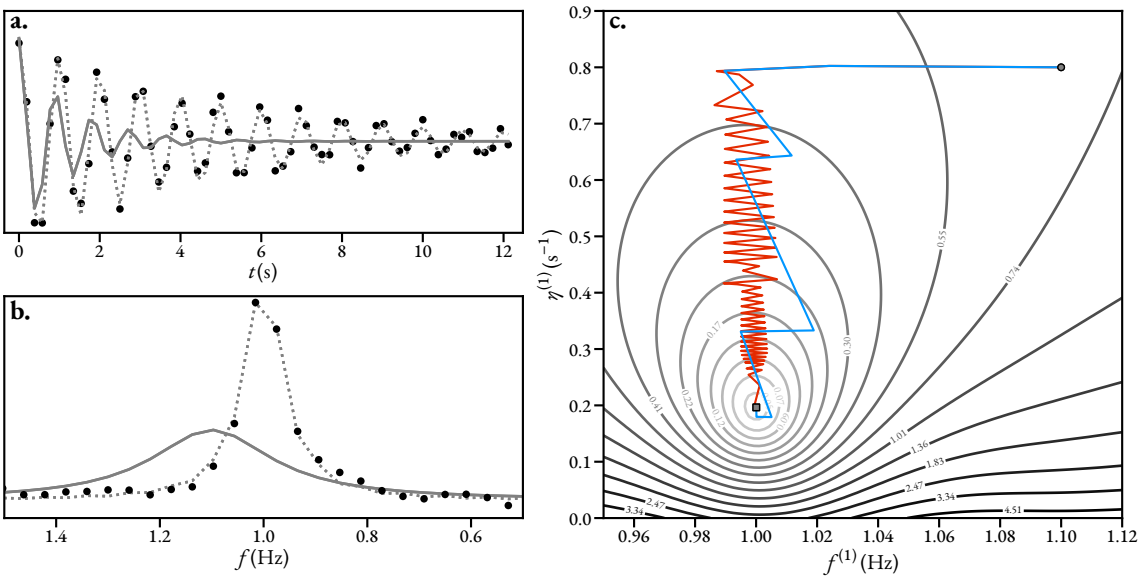


Figure 2.2: A visualisation of the trajectory of a 2-parameter optimisation involving a simulated FID comprising a single resonance. **a.** & **b.** Representations of the signal in the time domain and Fourier domain, respectively. Black dots: the signal to be estimated $\boldsymbol{Y}$. Solid grey line: the model generated using the initial guess $\boldsymbol{X}\left(\boldsymbol{\theta}^{(0)}\right)$. Dotted grey line: the model generated using the optimised result, $\boldsymbol{X}\left(\boldsymbol{\theta}^{(*)}\right)$. **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.

2.3.5 Visualisation of a simple example

Figure 2.2 provides a visualisation of numerical optimisation applied on a simulated FID comprising a single resonance. The FID was constructed using (1.21) with $D = 1$, $M = 1$, $N^{(1)} = 64$, $f_{\text{sw}}^{(1)} = 5.2\,\text{Hz}$ ($\Delta_t^{(1)} \approx 0.192\,\text{s}^{-1}$), and $f_{\text{off}}^{(1)} = 0\,\text{Hz}$. The resonance was parameterised by $\boldsymbol{\theta} \in \mathbb{R}^4$ comprising $a = 1$, $\phi = 0\,\text{rad}$, $f^{(1)} = 1\,\text{Hz}$, $\eta^{(1)} = 0.2\,\text{s}^{-1}$. White Gaussian noise was added to the FID to give it an SNR of approximately 10 dB. As the visualisation of 5D space is beyond the scope of this work, only two parameters, the frequency and damping factor were optimised from an initial guess, with the amplitude and phase being fixed to their true values. The initial guess comprised a frequency of 1.1 Hz, and a damping factor of $0.8\,\text{s}^{-1}$, with the solid grey lines in panels a & b denoting the model generated using the initial guess 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 (2.45b) was used in order to compute each update step, while in the second the Hessian approximation given by (2.47) 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[67]. The trajectories of the parameter vector are denoted as coloured lines in panel c. In both cases, the NLP routine successfully generated 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) led to a far better rate

of convergence compared with the approximated analogue (red), which exhibited "zig-zagging". This phenomenon is often seen in gradient descent methods, in which each update occurs in the opposite direction to the gradient. 14 iterations were required to reach the convergence criterion $\epsilon \leq 10^{-8}$ when the true Hessian was used, while 81 were required for the approximated case. While 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, as the cost of computing the second derivatives required for the true Hessian dominates.

2.3.6 Phase Variance Minimisation

Re-write this section While numerical optimisation procedures can return estimates $\boldsymbol{\theta}^{(*)}$ which can achieve highly accurate reconstructions of the original FID (i.e. $\boldsymbol{Y}$ and $\boldsymbol{X}\left(\boldsymbol{\theta}^{(*)}\right)$ are in close agreement), the estimate won't necessarily provide a faithful description of the physical phenomenon that has given rise to the data. For the purposes of FID estimation, the goal is to ensure that each signal contributing to the FID is described by a single oscillator in the model. In scenarios where the model order M used is greater than the true number of resonances, over-fitting of the data will occur, leading to spurious features in the estimation result, such as single resonances being fit by multiple oscillators and/or noise components being fit. To overcome this problem, it is desirable to include known information about the signal into the optimisation routine as a means of guiding the parameter vector to a more appropriate final value. One particular means of achieving this which has been to found to be effective is the incorporation of the variance of oscillator phases into the fidelity, such that it becomes

$$\mathcal{F}_{\phi}\left(\boldsymbol{\theta} \,|\, \boldsymbol{Y}\right) = \left\|\boldsymbol{Y} - \boldsymbol{X}\left(\boldsymbol{\theta}\right)\right\|^{2} + \mathrm{Var}_{\circ}\left(\boldsymbol{\phi}\right), \tag{2.55}$$

where $\mathrm{Var}_{\circ}\left(\boldsymbol{\phi}\right)$ is the *circular variance* of the oscillator phases (*vide infra*).

Remark 3. *The inclusion of the phase variance into the fidelity is one of the motivating reasons for normalising the data prior to estimation (see Remark 2).* $\mathrm{Var}_{\circ}\left(\boldsymbol{\phi}\right)$ *is constrained to the interval* $[0, 1]$. *If the data were not normalised, it is likely that* $\left\|\boldsymbol{Y} - \boldsymbol{X}\right\|^{2}$ *would dominate* $\mathrm{Var}_{\circ}\left(\boldsymbol{\phi}\right)$ *in Equation 2.55, such that the influence of the phase variance would be negligible.*

For the phase variance to act effectively, it is necessary to apply phase-correction to the data prior to estimation, as the assumption that the phases of all contributing resonances are equal is typically valid. **Mention phase-variation of multiplet peaks in experiments where J-coupling evolves prior to acquisition?** T_2 **for example.** The inclusion of phase variance has also been found to be effective at purging excessive oscillators that may be present in the initial guess $\boldsymbol{\theta}^{(0)}$, which be thought of in the following way:

(i) Assume that the initial guess contains more oscillators than the true number of resonances. In this circumstance, it is common to find that true resonances are fit with an acceptable oscillator, whilst extra oscillators exist with spurious phases on account of over-fitting.

(ii) Unconstrained numerical optimisation is now run, with the fidelity given by Equation 2.55. The phase variance will force oscillators with phases that are significantly different to the majority of oscillators to drastically alter their phases to match them. For some/all of these oscillators, it may be the case that the optimiser drives these to acquire a negative amplitude, such that they act as if they have a phase of π while ensuring that $\mathrm{Var}_\circ(\boldsymbol{\phi})$ is small.

(iii) This provides a criterion for detecting oscillators which are likely to be excessive. Such oscillators can be purged from $\boldsymbol{\theta}$ by periodically checking whether any amplitudes (i.e. $\boldsymbol{\theta}[:M]$) have become negative, purging these, and re-starting the optimisation.

(iv) The numerical optimisation routine is re-started each time oscillators are purged. Termination is achieved once the routine converges and no negative-amplitude oscillators exist in $\boldsymbol{\theta}$.

Circular Variance

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

$$\phi = \left(\left(\widetilde{\phi} + \pi\right) \bmod 2\pi\right) - \pi. \tag{2.56}$$

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

$$\mathrm{Var}_\shortmid(\boldsymbol{\phi}) = \frac{1}{M}\sum_{m=1}^{M}\left(\phi_m - \mu(\boldsymbol{\phi})\right)^2, \tag{2.57a}$$

$$\mu(\boldsymbol{\phi}) = \frac{1}{M}\sum_{m}\phi_m, \tag{2.57b}$$

unsuitable for phases. Consider as a simple example a scenario where there are two oscillators with phases $\widetilde{\boldsymbol{\phi}} = [\pi + \delta \;\; \pi - \delta]^\mathrm{T}$ for some small δ. The phase variance is expected to be small as these two phases are similar. However, with the inclusion of wrapping through application of (2.56), these phases would actually be set to $\boldsymbol{\phi} = [-\pi + \delta \;\; \pi - \delta]^\mathrm{T}$, and the conventional definition of phase variance would be large. It is therefore apparent that a definition of variance which accounts for the periodicity of the phases is needed. The *circular variance* is given by[68: Chapter 3]

$$[0,1] \ni \mathrm{Var}_\circ(\boldsymbol{\phi}) = 1 - \frac{R}{M}, \tag{2.58a}$$

$$R = \sqrt{c_\Sigma^2 + s_\Sigma^2}, \tag{2.58b}$$

$$c_\Sigma = \sum_m \cos\phi_m, \tag{2.58c}$$

$$s_\Sigma = \sum_m \sin\phi_m. \tag{2.58d}$$

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

The first and second derivatives of the circular variance are required for the computation of the gradient vector and Hessian matrix. These are given by

$$\frac{\partial \operatorname{Var}_\circ(\boldsymbol{\phi})}{\partial\theta_i} = \begin{cases} \frac{1}{RM}\left(c_\Sigma \sin\phi_{i-M} - s_\Sigma \cos\phi_{i-M}\right) & M \le i < 2M \\ 0 & \text{otherwise} \end{cases} \tag{2.59a}$$

$$\frac{\partial^2 \operatorname{Var}_\circ(\boldsymbol{\phi})}{\partial\theta_i\partial\theta_j} = \begin{cases} \frac{1}{RM}\left[\frac{1}{R^2}\left(c_\Sigma \sin\phi_{i-M} - s_\Sigma \cos\phi_{i-M}\right)^2 \right. & \\ \qquad \left. + c_\Sigma \cos\phi_{i-M} + s_\Sigma \sin\phi_{i-M} - 1\right] & M \le i,j < 2M, i = j \\ \frac{1}{RM}\left[\frac{1}{R^2}\left(c_\Sigma \sin\phi_{i-M} - s_\Sigma \cos\phi_{i-M}\right)\right. & \\ \qquad \times \left(c_\Sigma \sin\phi_{j-M} - s_\Sigma \cos\phi_{j-M}\right) & M \le i,j < 2M, i \ne j \\ \qquad \left. - \cos\left(\phi_{i-M} - \phi_{j-M}\right)\right] & \\ 0 & \text{otherwise} \end{cases} \tag{2.59b}$$

2.4 Profiling the MPM and NLP

The routine described for FID estimation involves operations which can be computationally demanding. This is the case both in terms of the amount of work done by the central processing unit (CPU), and the amount of random access memory (RAM) needed to store all the required information as the routine runs. For the MPM, the most demanding aspect is SVD calculations while for numerical optimisation, it is generation of the Hessian matrix at each iteration. Detailed accounts of the computational complexity of the MPM and MMEMPM have been presented[60, 61]. However, it is useful to consider what the actual running times of these routines are on a modern computer. A lot of the accounts on running the MPM are from decades before this work, and so the time required for the routine to run will have decreased a lot thanks to improvements in processing power. For example, the account by Pines a coworkers from 1997 states that a signal comprising 1024 time-points would take about 4.5 min to be processed by the MDL and MPM, using a 100 MHz CPU[69]. On the system used for all results generated for this work (see Remark 4), an equivalent computation takes about 100 ms.

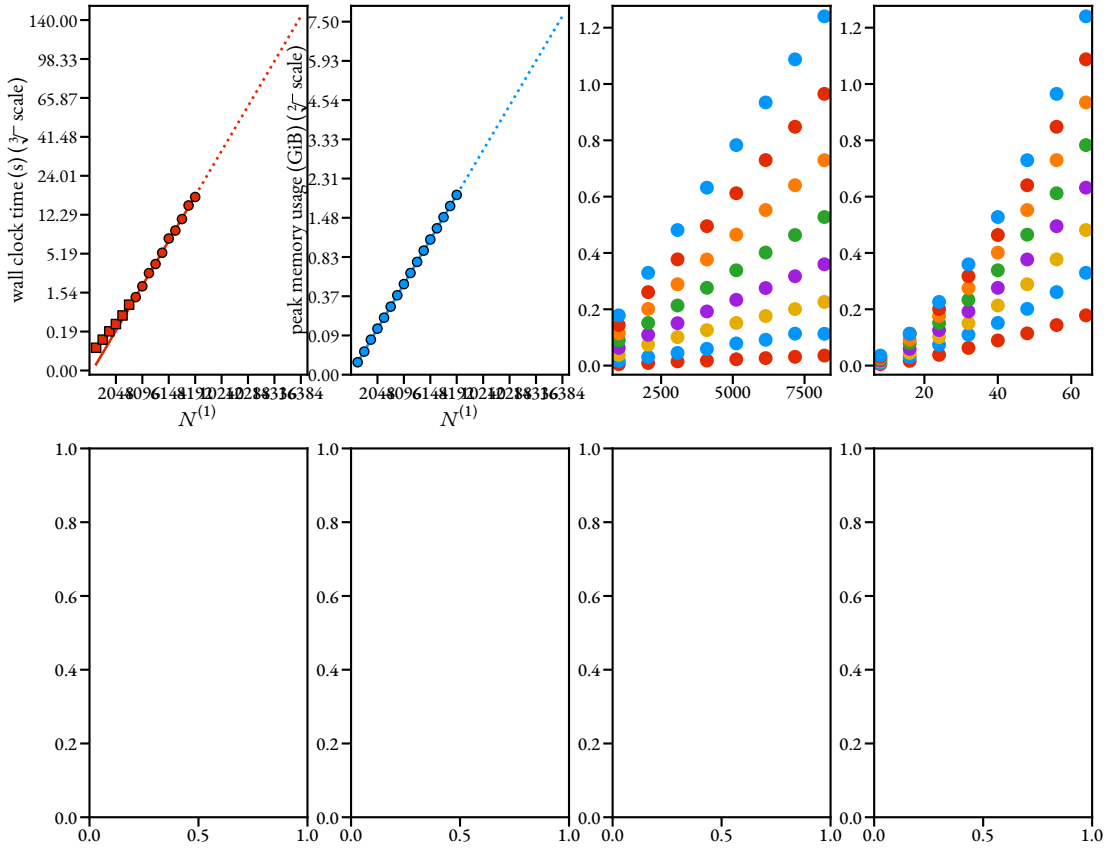


Figure 2.3: TODO

Remark 4. *All results generated in this work were acquired using a workstation featuring a Intel® Core™ i9-10900X CPU @ 3.7 GHz, and 32 GiB of RAM.*

Appendix: mention use of `timeprof` and `memory_profiler`

2.4.1 1D MPM

An series of FIDs were constructed with a variable number of time-points $N \in \{512k \mid k \in \{1, 2, \cdots, 16\}\}$. For each FID, the MPM was performed 5 times, with a pencil parameter $L = \lfloor N/3 \rfloor$. A Python implementation of the MPM was timed using a line-by-line profiler. The mean complete time to run the MPM is plotted as a function of N in panel a of Figure 2.3, where it can be seen that for sufficiently large N the MPM is computed in approximately $\mathcal{O}(N^3)$ time. This is due to the most time-consuming aspect of the MPM being the computation of the SVD of $\boldsymbol{H}_Y$, whose size is approximately $\frac{2N}{3} \times \frac{N}{3}$[§]. For small values of N (i.e. the square points in the plot), a deviation away from a cubic relationship is observed. This is since the computation of the complex amplitudes

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

in accordance with (2.15) has a comparatively significant run time[¶]. The figure shows the fit of a cubic, of the form $aN^3 + b$, to the circular points, corresponding to values of N for which all steps other than the SVD of $\boldsymbol{H_Y}$ are negligible in comparison. This function has been extrapolated to 16k points, to give an idea of the required run-time over a wide range of N (of course, the run time will vary depending on the hardware used).

2.5 Frequency Filtration

The previous section provides motivation for finding ways to reduce the number of points in the signal and also the number of oscillators that the signal contains. This has led to work on a procedure for generating frequency-filtered "sub-FIDs" from the original data. A detailed description of the filtering procedure is presented in this section.

2.5.1 The virtual echo

In brief, the filtering procedure consists taking the FT of the FID, applying a band-pass filter on the spectral data to discard parts of not being considered, and returning the spectrum back to the time-domain by an inverse Fourier transform (IFT). For a filtered FID to be faithfully described by the model of a summation of damped complex sinusoids, it is necessary that the spectral peaks of interest lie effectively entirely within the filter region[‖]. For absorptive Lorentzians, due to their characteristically narrow linewidths, this is straightforward. However, for broader dispersive Lorentzians, this is far more challenging. For this reason, generating a spectrum in which only the real component is retained is desired. Assuming the data has been phase-corrected, this will produce a spectrum comprising only absorptive Lorentzians. The virtual echo (VE) has been employed here, which has found application in the field of compressed sensing NMR[70–72]. This is a signal with double the size as the original FID, with the key characteristic that its FT has a real component which is equivalent to its counterpart derived from an unaltered FID (except it has double the points), and an imaginary component of 0s. The concept of a virtual echo can be applied to data of any number of dimensions. However, only 1D virtual echoes are employed in the work outlined in this thesis. An account of the 2D virtual echo is provided in the Appendix (Section A.2.

[¶]For a 512 point signal, the 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% and <1%, respectively.

[‖]Lorentzian lineshapes tend to, but don't reach zero, as the distance from the maximum tends to ∞[26]. However, as long as a sufficiently wide filtering is employed, the regions of the Lorentzian which do not pass through the filter can be assumed to be negligible.

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

$$y_n = \xi_n(c_n + \mathrm{i}s_n) + w_n, \tag{2.60a}$$

$$\xi_n = \sum_m a_m \exp(-\eta_m n \Delta_t), \tag{2.60b}$$

$$c_n/s_n = \sum_m \cos/\sin(2\pi f_m n \Delta_t). \tag{2.60c}$$

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

$$\psi_{\pm,n} = \xi_n(c_n \pm \mathrm{i}s_n) + w_n \equiv \Re(y_n) \pm \mathrm{i}\Im(y_n) \tag{2.61}$$

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:

$$\boldsymbol{t}_1 = \begin{bmatrix} \boldsymbol{\psi}_+ \\ \mathbf{0} \in \mathbb{C}^N \end{bmatrix}. \tag{2.62}$$

$\boldsymbol{t}_2$ 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 ($\cdot^{\circlearrowright}$):

$$\boldsymbol{t}_2 = \begin{bmatrix} \mathbf{0} \in \mathbb{C}^N \\ \boldsymbol{\psi}_-^{\leftrightarrow} \end{bmatrix}^{\circlearrowright}. \tag{2.63}$$

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

$$\boldsymbol{y}_{\mathrm{ve}} = \begin{bmatrix} \Re(y_0) & y_1 & \cdots & y_{N-1} & 0 & y_{N-1}^* & \cdots & y_1^* \end{bmatrix}^{\mathrm{T}}. \tag{2.64}$$

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

2.5.2 The filtering process

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 $b \in \mathbb{N} : b < 2N$ in each dimension:

$$g_n = \exp\left(-2^{p+1}\left(\frac{n-c}{b}\right)^p\right). \tag{2.65}$$

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

$$\widetilde{\boldsymbol{s}}_{\text{ve}} = \boldsymbol{s}_{\text{ve}} \odot \boldsymbol{g} + \boldsymbol{w}_{\sigma^2} \odot (\boldsymbol{1} - \boldsymbol{g}) . \tag{2.66}$$

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

After filtering, $\widetilde{\boldsymbol{s}}_{\text{ve}}$ is returned to the time-domain by IFT. The IFT of a real-valued spectrum generates a conjugate-symmetric signal, which is also a VE. This is sliced so as to retain the first half, which is the final filtered sub-FID $\widetilde{\boldsymbol{y}} \in \mathbb{C}^N$:

$$\widetilde{\boldsymbol{y}}_{\text{ve}} = \text{IFT}(\widetilde{\boldsymbol{s}}_{\text{ve}}), \tag{2.67a}$$

$$\widetilde{\boldsymbol{y}} = \widetilde{\boldsymbol{y}}_{\text{ve}}[0 : N]. \tag{2.67b}$$

A summary of the filtering process is provided by Figure 2.4.

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

$$c = \tfrac{1}{2} \left(l_{\text{idx}} + r_{\text{idx}} \right), \tag{2.68a}$$

$$b = l_{\text{idx}} - r_{\text{idx}}, \tag{2.68b}$$

where l_{idx} and r_{idx} denote the desired indices where the filter's left and right bounds are located, respectively. Array indices can be obtained from the corresponding spectral frequencies $f_{\text{Hz}}^{(d)}$ via

$$f_{\text{idx}} = \left\lfloor \frac{(2N-1)\left(f_{\text{sw}} + 2\left(f_{\text{off}} - f_{\text{Hz}}\right)\right)}{2f_{\text{sw}}} \right\rceil \qquad \forall f_{\text{Hz}} \in \left[f_{\text{off}} - \tfrac{1}{2} f_{\text{sw}}, f_{\text{off}} + \tfrac{1}{2} f_{\text{sw}} \right]. \tag{2.69}$$

Conversion from ppm to array indices can be achieved by replacing f_{Hz} in (2.69) with $f_{\text{ppm}} f_{\text{sfo}}$, where f_{sfo} is the transmitter frequency (MHz) and f_{ppm} is the frequency expressed as a chemical

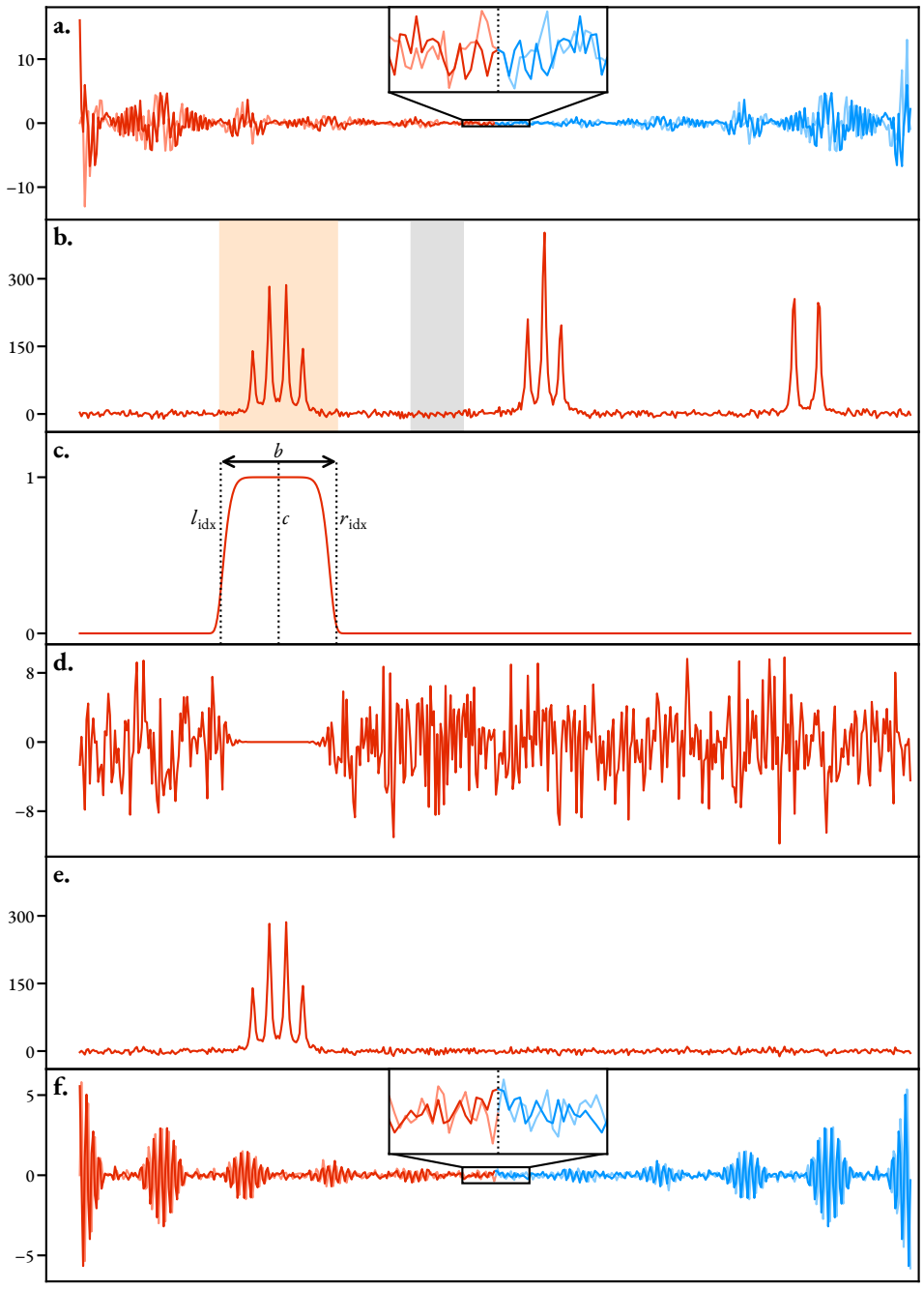


Figure 2.4: An illustration of the filtering procedure applied to a 1D FID. **a.** A VE $\boldsymbol{y}_{\text{ve}}$, with the first and last N points coloured red and blue, respectively. The middle of the VE is magnified to highlight its conjugate symmetry. **b.** The FT of the VE, $\boldsymbol{s}_{\text{ve}}$. 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.** AWGN vector to be added to the filtered spectrum. The magnitude of the signal at each point is dependent on the corresponding super-Gaussian value. **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 points.

shift.

Spectrum slicing

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

$$l_{\mathrm{slice}} = \begin{cases} c - \left\lfloor \frac{b\chi}{2} \right\rfloor & \text{if } \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.70a}$$

$$r_{\mathrm{slice}} = \begin{cases} c + \left\lceil \frac{b\chi}{2} \right\rceil & \text{if } \leq 2N - 1 \\ 2N - 1 & \text{otherwise} \end{cases} \tag{2.70b}$$

The filtered spectrum is then sliced accordingly:

$$\widetilde{\boldsymbol{s}}_{\mathrm{ve,slice}} = \widetilde{\boldsymbol{s}}_{\mathrm{ve}}[l_{\mathrm{slice}} : r_{\mathrm{slice}} + 1]. \tag{2.71}$$

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

$$\widetilde{\boldsymbol{y}} = \frac{r_{\mathrm{slice}} - l_{\mathrm{slice}}}{2N} \mathrm{IFT}(\widetilde{\boldsymbol{s}}_{\mathrm{ve,slice}})[0 : N_{\mathrm{slice}}], \tag{2.72a}$$

$$N_{\mathrm{slice}} = \left\lfloor \frac{r_{\mathrm{slice}} - l_{\mathrm{slice}}}{2} \right\rfloor \tag{2.72b}$$

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

$$f_{\mathrm{sw,slice}} = \frac{r_{\mathrm{slice}} - l_{\mathrm{slice}}}{2N - 1} f_{\mathrm{sw}} \tag{2.73a}$$

$$f_{\mathrm{off,slice}} = f_{\mathrm{off}} + \frac{f_{\mathrm{sw}}}{2} \left(1 - \frac{l_{\mathrm{slice}} + r_{\mathrm{slice}}}{2N - 1} \right) \tag{2.73b}$$

2.6 Summary

The matrix pencil-based methods described above are well established as effective procedures for parametric estimation of time-domain signals in a number of disciplines, including radar detection[73], acoustics[**TODO**], and NMR[69], with specific application to relaxometry being a recently introduced application [74, 75]. Due to considerable advances computational processing power since the introduction of the technique, estimates can be acquired from NMR signals in reasonable times. One notable downside of the technique that has been realised while assessing its effectiveness in parametrising NMR FIDs is its propensity to return oscillators with unexpected phase behaviour, especially in scenarios involving resonances with close frequencies. For this reason, estimating phase-corrected FIDs, using the result of the MPM as an initial guess to feed into a phase-variance penalised NLP routine was proposed as a means of improving parameter estimates. The theory underpinning the procedure has been explored in this chapter.

The computational burden of running the procedure is large and often intractable for complete NMR signals, which often comprise thousands of samples, and at least hundreds of contributing resonances. This has been illustrated through profiling both the CPU time and the peak memory requirements for the 1D and 2D methods. **Say more when completed?** For this reason, a method to break FIDs into frequency-filtered sub-signals, by filtering spectra derived using virtual echoes, is introduced. This method enables to formation of signals with far fewer resonances and samples as compared to the full signal.

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

1D Results and Applications

3

This chapter showcases numerous results generated using the 1D estimation technique outlined in Chapter 2. Furthermore, two means by which the standard 1D estimation procedure can be extended for use in specific applications are presented.

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

3.1 Conventional 1D datasets

3.1.1 "Twenty signals"

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

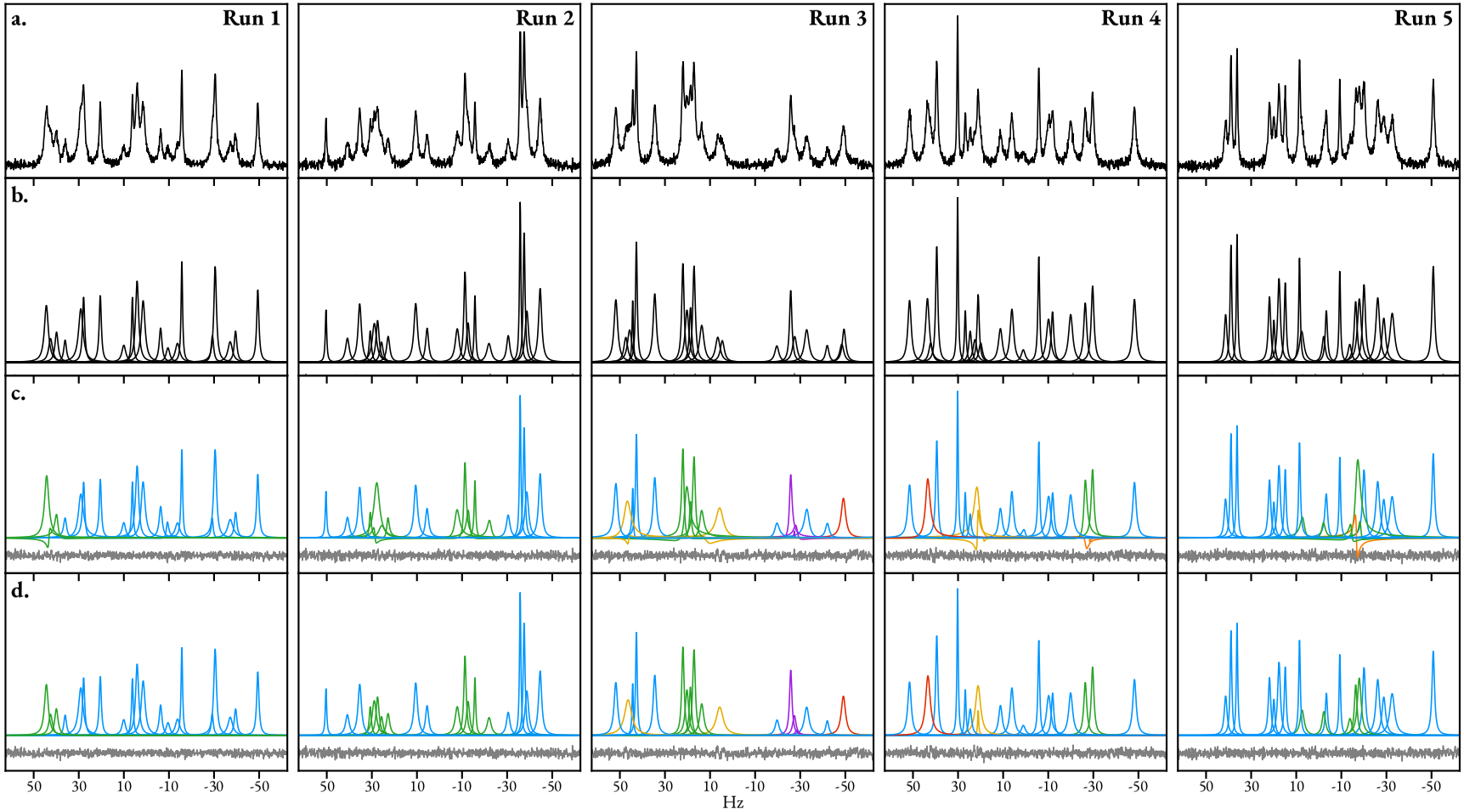


Figure 3.1: The result of estimating a series of 5 simulated signals comprising 20 oscillators (see the main text for details on how the datasets were constructed). **a.** Spectra of the datasets generated. **b.** Spectral lines corresponding to the true set of oscillators used to generate each dataset.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

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

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

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

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

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

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

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

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

3.1.2 Andrographolide

Figure 3.2 illustrates the outcome of applying the estimation routine to selected regions of a ^{1}H dataset of andrographolide (Figure C.1.e) in DMSO-d_6 acquired with a 600 MHz spectrometer. The NLP routine is effective at resolving the spurious phase-behaviour often generated by the MPM (cf. panels b and c). It's ability to estimate parameters from resonances with high dynamic range and high variation of damping factors is also evidenced by the fact that it was able to assign an intense broad singlet, corresponding to water which has entered the sample over time, alongside a nearby quartet corresponding to the methylene of some residual ethanol in the sample.

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

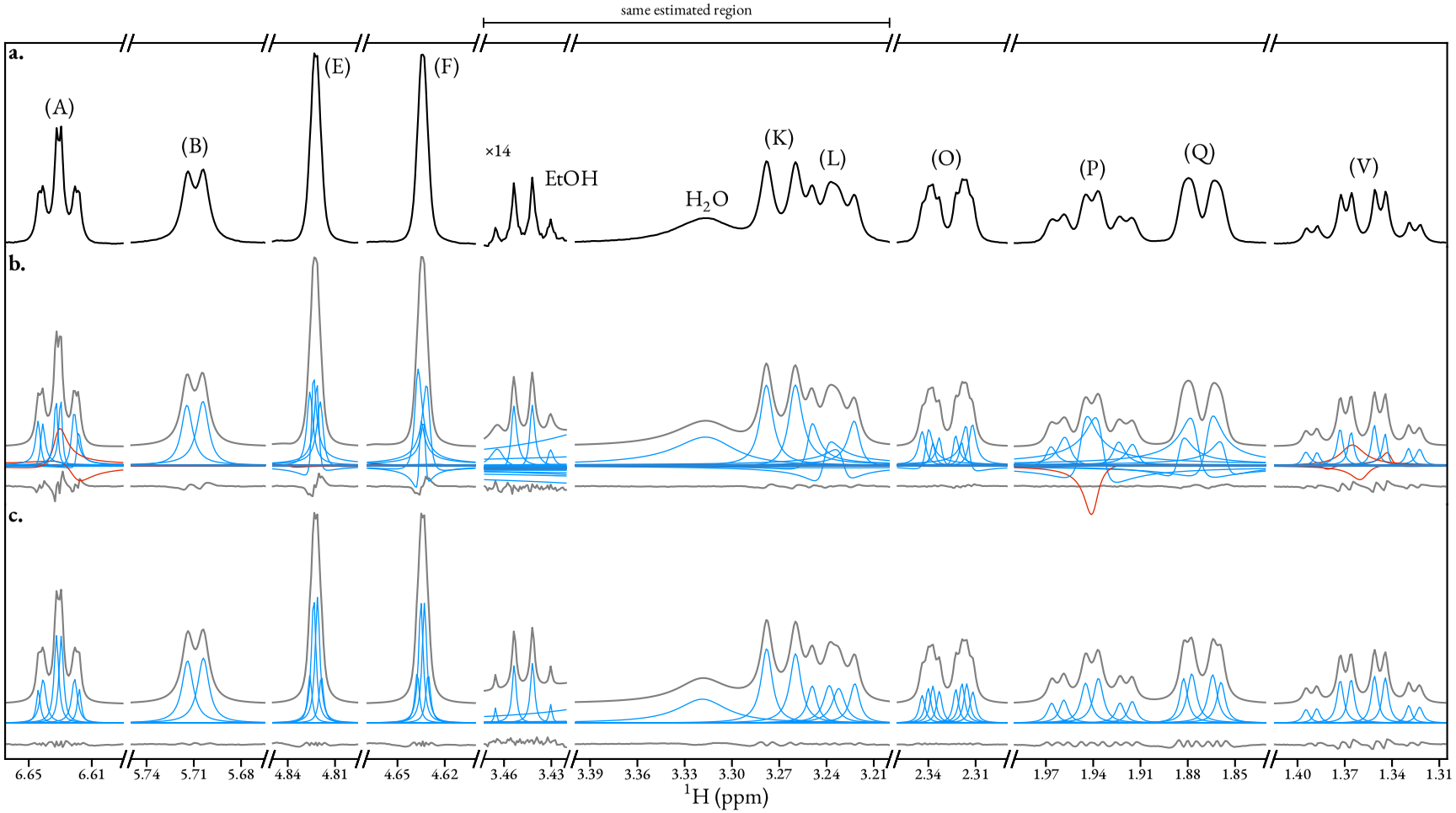


Figure 3.2: Result of applying the estimation routine to selected regions of a pulse-acquire dataset of andrographolide in DMSO-d_6. **a.** Spectral data corresponding to the regions considered. **b.** The result of applying the MPM to the regions, with the model order predicted with the MDL. Blue/red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

Spin	Coupling partners	Apparent multiplet structure
(A)	$(D)^{long}$, $(M)^{vic}$, $(N)^{vic}$	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)	$(O)^{gem}$, $(R)^{60}$, $(V)^{180}$	dt
(Q)	$(M)^{vic}$, $(N)^{v}$	dd
(V)	$(O)^{60}$, $(P)^{180}$, $(R)^{g}$, $(W)^{180}$	dq

Table 3.1: Major coupling partners associated with spins in andrographolide whose signals are considered in Figure 3.2, along with the aparent multiplet structures that arise. Coupling partners are labelled as follows: ^{vinyl} geminal coupling between two vinylic protons, ^{ex} geminal coupling between two protons, in which one if bonded to an oxygen, leading to exchange decoupling, ^{gem} geminal coupling, ^{long} long-range coupling, ^{vic} vicinal coupling, ^{60} geminal coupling, with a fixed dihedral angle of 60°, ^{180} geminal coupling, with a fixed dihedral angle of 180°.

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

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

As a second illustration, the signal corresponding to spin (V) is also under-fit, this time because the presence of a number of couplings of similar magnitudes leads to resonances coalescing at roughly the same frequency. For (V), a multiplet structure featuring 16 resonances forming in a "dddd" structure is expected, however 3 of the couplings are of similar magnitudes, such that individual resonances coalesce to form what is apparently a quartet of doublets (dq). The estima-

tion routine was able to resolve this dq structure, however the large wobbles in the residual again signal that under-fitting has occurred, and each oscillator is in fact being used to fit two or more resonances present in the FID. Again, at the field strength used to acquire the FID, it is unlikely that an accurate resolution of all 16 oscillators by estimation is feasible.

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

Spin (L) also has 3 major couplings, though only a dd structure is apparent, which is what has been assigned by the estimation routine. The small residual also implies that an under-fit has not occurred, even though a ddd structure would be anticipated at first glance. However, one of the coupling partners is a labile hydroxyl proton, which rapidly undergoes exchange with other protons on the sample*. Chemical exchange at a sufficiently high rate leads to a "decoupling" of the spins, such that the two resonances associated with a doublet coalesce into a broad singlet[76: Section 2.6.1.5].

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

3.1.3 Cyclosporin

TODO: result from Newton Meets Ockham paper

3.2 Amplitude-attenuated Datasets

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

3.2.1 Relaxation experiments

Any spin system which has been perturbed from its equilibrium position will eventually return to equilibrium by relaxation. As introduced in Chapter 1, the simplest model of relaxation centers

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

around two quantities: the longitudinal relaxation rate and the transverse relaxation rate, quantified by the times T_1 and T_2 respectively. Though a rigorous picture of relaxation in NMR requires the use quantum mechanics [22: Chapter 5, 77, 78], T_1 and T_2 are still valuable concepts which provide insight into the chemical system being studied[†]. Numerous factors affect these quantities, including the rate at which the spin tumbles in space, and its electronic environment. Well-known experiments exist to quantify both of these quantities.

PS figure

Measuring T_1: Inversion recovery

The inversion recovery experiment involves the simple pulse sequence $180° \rightarrow \tau \rightarrow 90° \rightarrow t^{(1)}$. The initial 180° pulse inverts the magnetisation, so that it is along $-z$. During τ, the spin system undergoes longitudinal relaxation, with the spin state populations at equilibrium gradually being restored. The 90° pulse rotates the magnetisation into the transverse plane, enabling detection. The resultant phase and magnitude of the signal is directly related to the amount of time that longitudinal relaxation is allowed to occur. With $\tau = 0\,\mathrm{s}$, a signal with maximal amplitude, but a phase of 180° will result[‡]. At the other extreme of very long τ, the spin system will have reverted back to equilibrium, such that a signal which also has maximal amplitude, but a phase of 0° will be realised. By sequentially adjusting τ in a 2D experiment, a series of spectra will be obtained in which the intensity of each peak 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 peak acquired when the spin system has returned to equilibrium. Note that $a(0) = -a_\infty$, as the spin system has been completely inverted, and no time has been allowed for longitudinal relaxation to take place. An example of a series of spectra acquired using an inversion recovery experiment is given by panel d in Figure 3.4.

Measuring T_2: CPMG

In an analogous fashion to the inversion recovery experiment, by creating transverse magnetisation and leaving it to evolve for different amounts of time, it is possible to determine T_2. This might lead one to believe that the pulse sequence $90° \rightarrow \tau \rightarrow t^{(1)}$ would be effective for T_2 determination. However the presence of J-modulation would generate undesirable spectra with phase-modulated

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

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

peaks. As well as this the presence of field inhomogeneities will cause relaxation at a faster rate than anticipated. The effect of field inhomogeneities is incorporated into the "observed" transverse relaxation time, T_2^*. The effects of J-modulation and field inhomogeneity can be nullified if rapid refocussing is applied, by subjecting the spin system to a train of spin echoes. The classic route to T_2 measurement is the Carl-Purcell-Meiboom-Gill (CPMG) experiment[79, 80], 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. T_2 attenuates the intensity of the resulting signal 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 reduces to a standard pulse-acquire experiment.

3.2.2 Diffusion experiments

NMR is well established as a means of determining the rates of diffusion of chemical species[81, 82]. The first showcase for determining translational diffusion coefficients came from Stejskal and Tanner, in which they described the pulsed gradient spin echo (PGSE) pulse sequence[83] (Figure 3.3.a). The PGSE sequence consists of a conventional spin-echo ($90° \xrightarrow{\tau} 180° \xrightarrow{\tau}$ acquire), with pulsed field gradients (PFGs) applied after each of the RF pulses. As a simple overview of how the pulse sequence works, consider a single spin on resonance with the transmitter (i.e. its rotating frame frequency is zero) in a sample tube at position z along the axis collinear with the main field. After the 90° pulse, the magnetisation will be $-M_y$. During the first PFG, the spin's resonance frequency will become $\omega_{\mathrm{PFG}} = -\gamma g z$, where g is the strength of the PFG. Assuming the gradient is applied for a time δ, the spin will precess by an angle of $\alpha = -\gamma g z \delta$. After the 180° pulse, the spin's magnetisation is as follows:

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

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

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

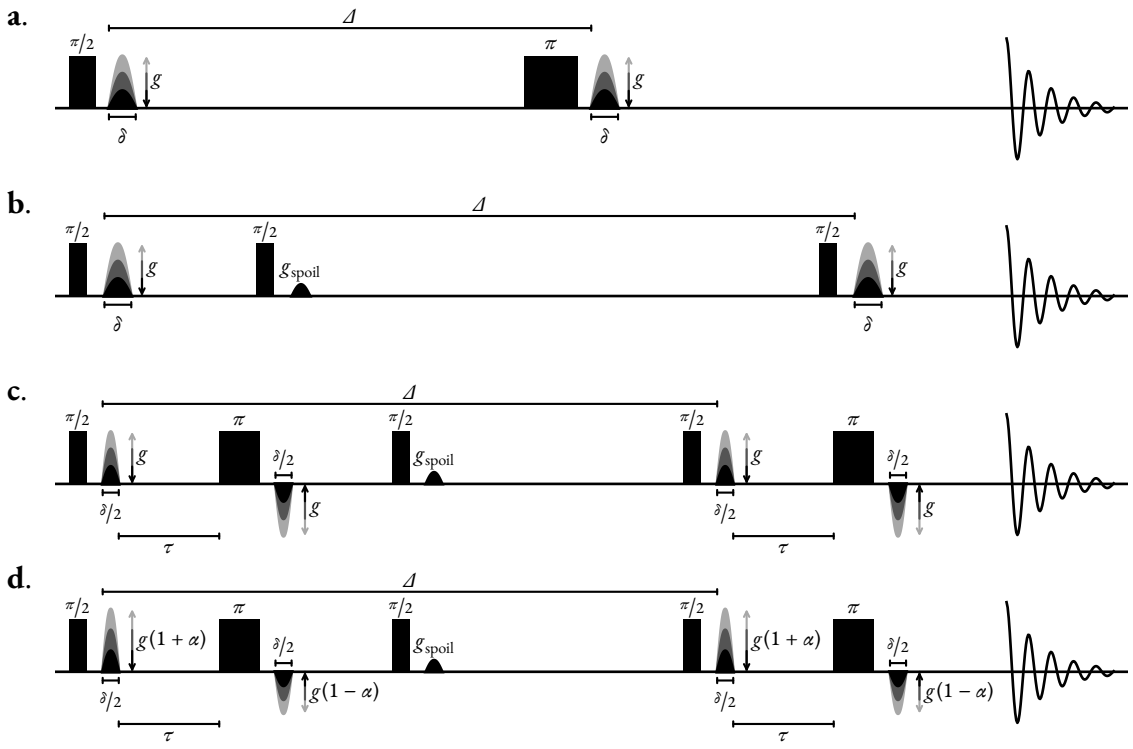


Figure 3.3: Pulse sequences used for the determination of translational diffusion constants. **a.** PGSE, **b.** PGSTE, **c.** PGSTEBP, **d.** One-shot DOSY. RF pulses are denoted by solid rectangles. Diffusion-encoding gradients are denoted by sine-bell shapes with varying shades, indicating that the intensity is incremented to create a 2D dataset. Spoiler gradients are denoted by solid black sine-bell shapes.

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

Through consideration of the Bloch-Torrey equations[84], which extend the classic Bloch equations to account for the effects of diffusion on magnetisation, the following equation, known as the *Stejskal-Tanner equation*, may be derived:

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

where $a_0 = \lim_{g \to 0} a$, γ is the gyromagnetic ratio of the target nucleus (rad MHz T^{-1}), g is the

gradient strength ($\mathrm{T\,m^{-1}}$)[§], δ is the duration of each PFG (s), Δ is the delay between the PFGs, often known as the diffusion time (s), and D is the translational diffusion constant of the species giving rise to the signal ($\mathrm{m^2\,s^{-1}}$). While (3.4) is widely stated in the literature, it is only strictly applicable when the PGSE sequence is used, and PFGs with rectangular amplitude profiles are applied[¶].

Tanner introduced a variant of the original PGSE experiment called pulsed gradient stimulated echo (PGSTE)[85] (Figure 3.3.b). Instead of the diffusion period including a 180° pulse, PGSTE features two 90° pulses, with the first being applied shortly after the initial PFG, and the second being applied just before the second PFG. The key difference between this and the PGSE experiment is that decoherence during the diffusion time is dictated by longitudinal relaxation (T_1) rather than transverse relaxation (T_2). PGSTE is therefore favoured in scenarios where $T_1 \ll T_2$, as improved sensitivity is attainable.

Both PGSE and PGSTE employ *monopolar* PFGs for diffusion encoding, in the sense that both diffusion-encoding PFGs are polarised in a single direction. Experiments also exist which employ *bipolar* gradient elements[86, 87], which consist of a PFG, followed by a 180° pulse, and then a second PFG with the opposite polarity to the first. A well-known example is the pulsed gradient stimulated echo with bipolar gradients (PGSTEBP) experiment (Figure 3.3.c). Bipolar gradients are useful in circumstances where it is important to purge the effects of static gradients in the sample, caused by field inhomogeneities. Morris and coworkers have developed the *one-shot* experiment[88] (Figure 3.3.d), which requires a single transient per gradient strength (i.e. there is no requirement for a phase-cycling scheme). This is achieved through the use of bipolar gradients which comprise asymmetrical PFGs with relative powers $1 + \alpha : 1 - \alpha$ for some $0 > \alpha > 1$ (a common value is 0.2).

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

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

for some constant c ($\mathrm{T\,s^{-2}}$). The functional form of c depends on the type of experiment used, as well as the amplitude profile of the PFGs. A consideration of the Bloch-Torrey equations for a given experiment is necessary, with an extensive overview provided by Sinnaeve for most diffusion NMR experiments[89]. In general, c is as follows:

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

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

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

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

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

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

$$\text{One-shot} \quad \Delta + \frac{(\kappa - \lambda)(\alpha^2 + 1)\delta}{2} + \frac{(\delta + 2\tau)(\alpha^2 - 1)}{4}. \tag{3.7c}$$

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

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

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

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

The three factors are given by

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

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

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

with their discrete counterparts being

$$\sigma = S_{N_g} \tag{3.11a}$$

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

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

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 (3.4) comes from plugging (3.7a) into (3.6), with parameters for rectangular PFGs: $\sigma = 1$, $\lambda = 1/2$, and $\kappa = \frac{1}{3}$. In many situations, Δ dominates in

the expression of Δ', and so ensuring 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 extremely important for accurate measurements of D.

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

3.2.3 Methodology

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

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

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

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

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

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

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

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

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

Experiment	p	ψ	$\mathcal{A}(\psi\|p)$	$\frac{\partial\mathcal{A}(\psi\|p)}{\partial\psi}$	$\frac{\partial^2\mathcal{A}(\psi\|p)}{\partial\psi^2}$
Inversion Recovery	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 its 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 greatest amplitude will occur when $n = 0$, as no time is designated for transverse relaxation to take place.
- For diffusion experiments, the largest amplitude is achieved when $g = 0$, since no diffusion-induced dephasing of spins will have occurred.

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

Estimating amplitude-attenuated datasets

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

$$\boldsymbol{a}_k = \underset{\boldsymbol{a}\in\mathbb{R}^M}{\arg\min}\, \mathcal{F}\left(\boldsymbol{a} \mid \boldsymbol{\phi}_{k=1}, \boldsymbol{f}_{k=1}, \boldsymbol{\eta}_{k=1}, \boldsymbol{y}_k\right). \tag{3.16}$$

An NLP routine can solve this very efficiently, typically in few iterations, on account of the linear dependence of the model with respect to the oscillator amplitudes. The linear dependence also means that second derivatives of the model are all zero (see (2.46e)), such that only first derivatives need to be computed to derive an exact Hessian matrix of the fidelity.

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

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

procedure EstimateAmpAttenuated($\boldsymbol{Y} \in \mathbb{C}^{K\times N}, \boldsymbol{r}_{\text{interest}} \in \mathbb{R}^2, \boldsymbol{r}_{\text{noise}} \in \mathbb{R}^2, M \in \mathbb{N}_0$)
  $\boldsymbol{\theta}_1, \boldsymbol{\epsilon}_1 \leftarrow$ Estimate1D $(\boldsymbol{y}_1, \boldsymbol{r}_{\text{interest}}, \boldsymbol{r}_{\text{noise}}, M)$; ▷ Estimate first increment
  $M \leftarrow {}^{\text{len}(\boldsymbol{\theta}_1)}/_4$;
  $\boldsymbol{\theta}, \boldsymbol{\epsilon} \leftarrow \boldsymbol{0} \in \mathbb{R}^{(K+3)M}, \boldsymbol{0} \in \mathbb{R}^{(K+3)M}$; ▷ 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 held constant across increments.
  for $k = 2, \cdots, K$ **do**
    $\widetilde{\boldsymbol{y}} \leftarrow$ Filter1D $(\boldsymbol{y}_k, \boldsymbol{r}_{\text{interest}}, \boldsymbol{r}_{\text{noise}})$; ▷ Algorithm A.3.
    $\boldsymbol{\theta}_k, \boldsymbol{\epsilon}_k \leftarrow$ NLPAmp $(\widetilde{\boldsymbol{y}}, \boldsymbol{\theta}_k)$;
    $\boldsymbol{\theta}[kM:(k+1)M], \boldsymbol{\epsilon}[kM:(k+1)M] \leftarrow \boldsymbol{\theta}_k[:M], \boldsymbol{\epsilon}_k[:M]$; ▷ Extract amplitudes.
  end for
  return $\boldsymbol{\theta}, \boldsymbol{\epsilon}$;
end procedure

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

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

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

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

Determining the parameter of interest

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

$$\begin{bmatrix} a_{0,m}^{(*)} \\ \psi_{0,m}^{(*)} \end{bmatrix} = \underset{[a_0 \;\; \psi]^{\mathrm{T}} \in \mathbb{R}^2}{\arg\min} \left\| \boldsymbol{a}_m - a_0 \mathcal{A}(\psi \mid \boldsymbol{p}) \right\|^2, \tag{3.18}$$

where $\boldsymbol{a}_m$ corresponds to the m^{th} column of $\boldsymbol{A}$. This is another example of a residual sum-of-squares problem, and can also be solved using an NLP routine. The gradient vector and Hessian matrix of the fidelity take very similar functional forms to those for FID estimation (see (2.45a)

and (2.45b)), and are as follows $\forall i, j \in \{0, 1\}$:

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

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

$$\mathbb{R}^2 \ni \boldsymbol{\vartheta} = \begin{bmatrix} a_0 & \psi \end{bmatrix}^{\mathrm{T}}, \tag{3.19c}$$

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 for the different experiments of interest of $\mathcal{A}$ are given in Table 3.2.

Displaying results

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

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

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

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

where $R \in \mathbb{N}$ is the number of samples to generate the distribution from. The exact form that the distribution should take is not set in stone, though it should indicate two key pieces of information: (i) its maximum should coincide with the predicted value of ψ_m and (ii) its width should indicate the level of uncertainty associated with the prediction. In this work, the distribution used

is a Gaussian distribution with mean ψ_m and standard deviation $c\epsilon_m$, where ϵ_m is the estimation error associated with oscillator m's parameter of interest,** and $c \in \mathbb{R}_{>0}$ is an arbitrary linewidth factor which can be chosen to ensure clear visibility of the peaks.††

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

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

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

3.2.4 Results

"Five multiplets"

Figure 3.4 shows the result of the described method in determining the T_1 values of resonances in a simulated inversion recovery datasets featuring 5 overlapping ddd multiplet structures. The spin systems used to generate the datasets were constructed in a similar way to that described for the "Four Multiplets" example in Section 4.3. In this case however, 5 estimated spins were present instead of 4. Constraints were also placed on the shifts and couplings to ensure that no two oscillators would have frequencies with a difference less than $f_{\text{sw}}^{(1)}/N^{(1)}$ **Describe in the appendix**. For each spin, a T_1 value was sampled from $\mathcal{U}(1\,\text{s}, 5\,\text{s})$, and a T_2 value was sampled from $\mathcal{U}(0.2\,\text{s}, 0.6\,\text{s})$. The inversion recovery experiment was simulated using SPINACH, with the relaxation phenomena described by the "extended T_1/T_2 approximation" (see Appendix **REF**). Each dataset was provided AWGN such that the target SNR of the datasets as a whole was 40 dB. The estimation routine outlined by Algorithm 3.1 was applied to the generate a parameter estimate of the region which contained signals from the estimated spins.

Despite heavy overlap between peaks, the routine was successful at assigning each signal in the dataset with a T_1 value that closely agreed with the true value (see panel b.). As is to be expected, in scenarios where little multiplet overlap existed, T_1 predictions tended to be more accurate, with smaller associated errors (see for example the purple and orange multiplets in Run 1. Nevertheless, adequate estimates could still be obtained in cases of severe overlap, especially when the predicted T_1s for all oscillators associated with a given multiplet are averaged. (see the red, blue and yellow multiplets in Run 1). Particular oscillators for which the estimate of T_1 is particularly far from the

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

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

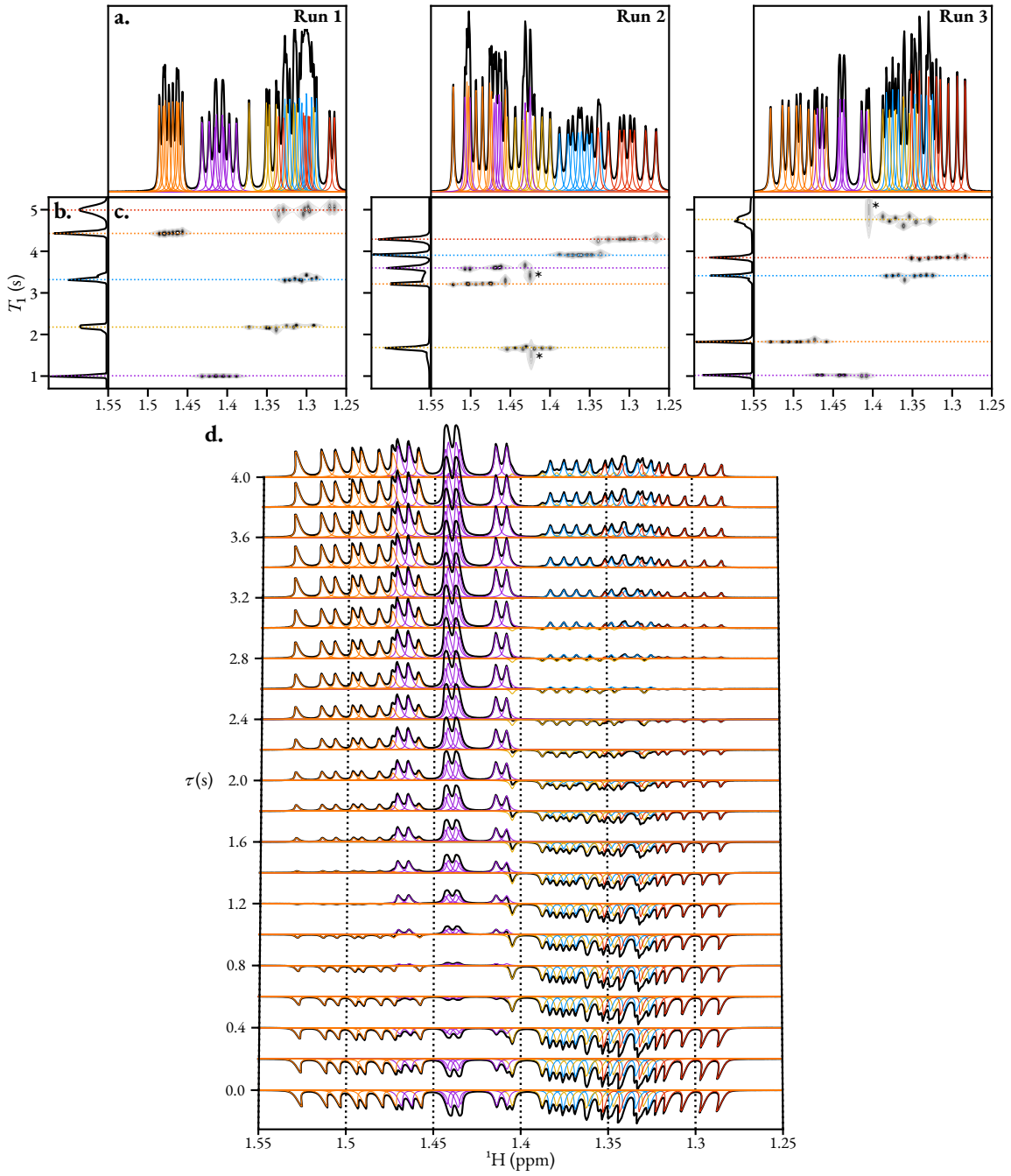


Figure 3.4: Three examples of results generated on simulated inversion recovery datasets comprising five ddd multiplet structures. **a.** Plot of the result generated for the first increment ($\tau = 0$ s), with each plot multiplied by −1. Black: spectrum of the data. Coloured lines: spectra of individual oscillators generated by the estimation routine. Oscillators with the same colour are components of the same multiplet. **b.** Distribution of T_1 values, generated using (3.22), with $p_{\min} = 0.7$ s, $p_{\max} = 5.3$ s, $c = 40$, $R = 128$. **c.** DOSY-style contour plot of the result, generated using (3.21). Dashed horizontal lines denote the true T_1 values for each spin. **c.** Estimation result for each increment for Run 3, illustrating the evolution of the amplitudes of each oscillator with τ.

true value tend to be associated with large errors, with examples denoted with an asterisk.

Andrographolide Diffusion

TODO: Re-run figure generation, with sigma set to 1. Compare with dynamic center Figure 3.5 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 4.57×10^{-10} 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[90]. The diffusion profile generated suggests a diffusion constant of andrographolide of 2.54×10^{-10} m^2 s^{-1}. The predicted diffusion constant for each estimated oscillator shows decent consistency, especially with oscillators of greater intensity. Lower intensity oscillators - especially those which significantly overlap with other oscillators - tended to be associated with less consistent diffusion constants and larger errors with examples of this phenomenon apparent in panel d of the figure. A few oscillators also show a significant deviation at around 2.5 ppm. This is likely due to the presence of signals that make up a 1:2:3:2:1 quintet due to the presence of partially protonated DMSO. As the data is insufficiently resolved to enable the separation of andrographoilde and Dimethyl sulfoxide, $(H_3C)_2SO$ (DMSO) signals in the estimation result, oscillators exist which will have an amplitude profile influenced by both species, leading to an aggregated diffusion constant. As $D_{\text{DMSO}} > D_{\text{andrographoilide}}$, the affected oscillators show larger apparent diffusion constants.

Glucose/valine/threonine diffusion

TODO: get result from dynamic center and compare Another example is provided by Figure 3.6, where the estimation routine was applied to a diffusion dataset derived from a sample comprising the molecules L-valine (M_r = 117.148 g mol^{-1}), L-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, from which a diffusion constant of 1.88×10^{-9} m^2 s^{-1} was determined. The routine was able to achieve separation of the three species in the sample. Predicted diffusion coefficients for valine and threonine were 6.20×10^{-10} m^2 s^{-1} and 6.39×10^{-10} m^2 s^{-1}. For glucose, the situation is complicated by the presence of two major forms, α-D-glucopyroanose and β-D-glucopyroanose[‡‡], due to anomerisation[91: Chapter 3]. There is some evidence of separation of these anomers, principally due to the downfield doublets at 5.15 ppm (α) and 4.56 ppm (β), which have estimated diffusion coefficients of 5.47×10^{-10} m^2 s^{-1} and 5.34×10^{-10} m^2 s^{-1},

[‡‡]The equilibrium mixture in water comprises 38% of the α isomer and 62% of the β isomer. A tiny amount of the open-chain form will also be present, though in a negligible quantity. This is evidenced by the spectrum in Figure 3.6.a, where the relative integrals of the doublets at 5.15 ppm and 4.56 ppm agree with this ratio.

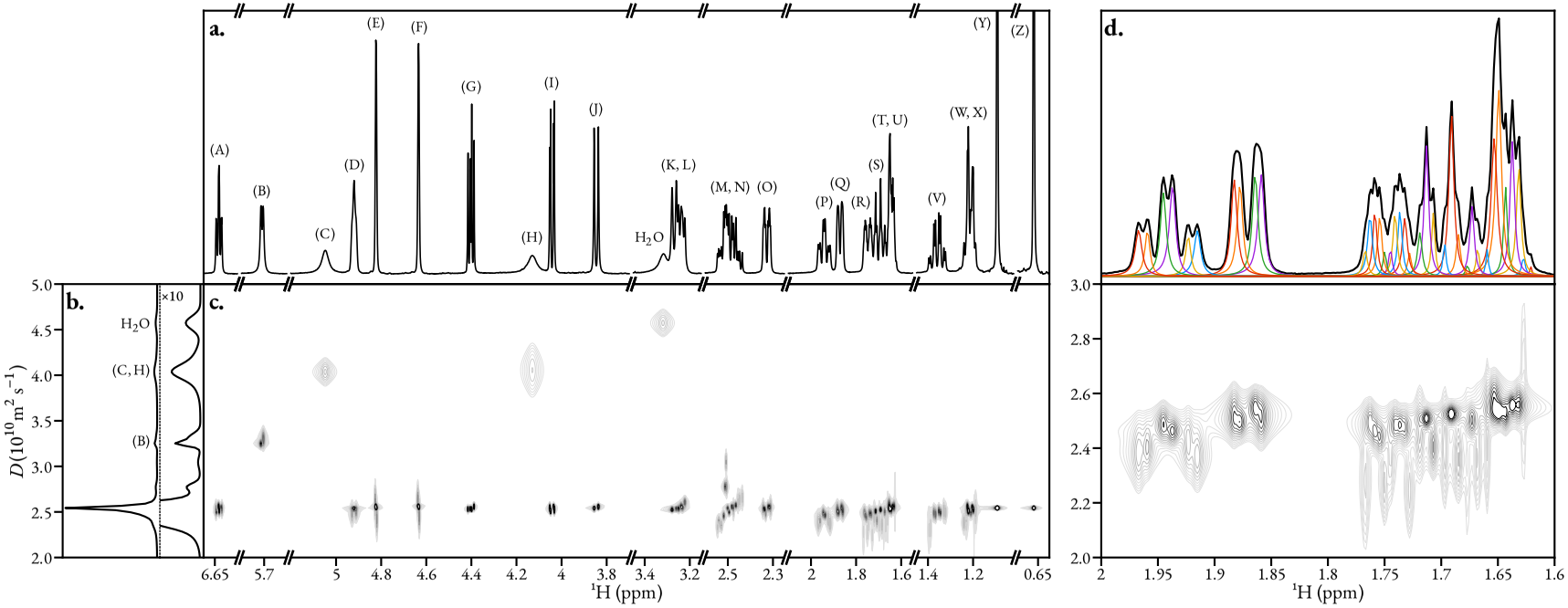


Figure 3.5: Result of estimating a Oneshot DOSY dataset of andrographolide in unfresh Deuterated DMSO (DMSO-d_6). **a.** 1D spectrum. **b.** Diffusion profile obtained by summing the contour plot in c. along the *x*-axis. **c.** Contour plot mapping estimated oscillators to diffusion constants, with $p_{\min} = 2 \times 10^{-10}\ \mathrm{m^2\ s^{-1}}$, $p_{\max} = 5 \times 10^{-10}\ \mathrm{m^2\ s^{-1}}$, $c = 2.5$, $R = 128$. **d.** Magnified view of the 2 ppm to 1.6 ppm spectral range, with estimated oscillator peaks plotted.

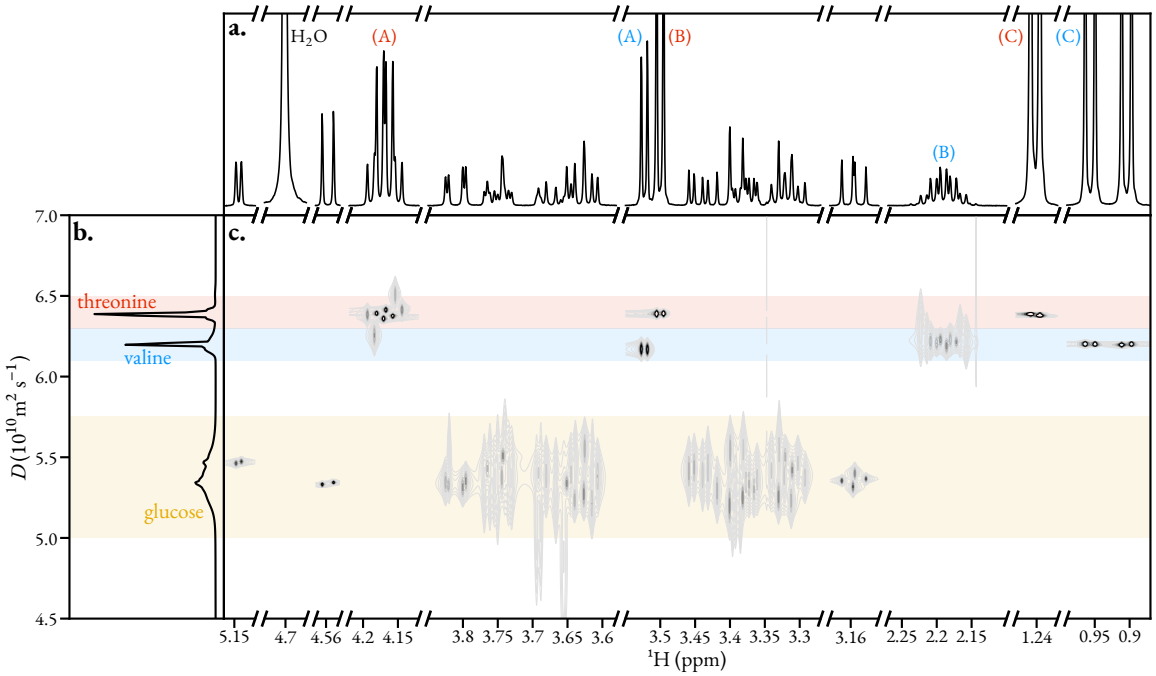


Figure 3.6: Result of estimating a diffusion dataset for a mixture of L-threonine, L-valine and D-(+)-glucose in D_2O. **a.** 1D spectrum, taken from the first FID of the diffusion dataset. **b.** Diffusion coefficient distribution. **c.** DOSY-style plot of chemical shifts vs diffusion constant, generated using (3.22), with $p_{\min} = 4.5 \times 10^{-10}$ m^2 s^{-1}, $p_{\max} = 7 \times 10^{-10}$ m^2 s^{-1}, $R = 256$, and $c = 1.5$.

respectively. Beyond these however, the other estimated oscillators corresponding to glucose have associated errors which are too large for clear resolution of the two anomers.

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

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

3.3 Phased broadband spectra from chirp excitation

The are numerous nuclei of value in NMR with very wide chemical shift ranges, including ^{13}C, ^{19}F (of particular interest in the pharmaceutical industry) ^{31}P, ^{195}Pt. Attaining spectra covering the entire chemical shift range of such spins for use in quantitative applications is challenging due to off-resonance effects, which severely alter the amplitudes and phases of resonances with frequencies far from the transmitter frequency[22: Section 3.4.1]. One popular means of achieving broadband excitation, in which a consistent amplitude- and phase-profile across a spectral window of tens or even hundreds of kHz the achieved, is to use frequency-swept (FS) pulses, during which the frequency of RF irradiation varies with time[92]. One of the most common classes of FS pulses are those where the variation of frequency with time is linear, with such pulses commonly referred to as *chirp* pulses. The application of a single 90° chirp pulse to achieve broadband excitation, while simple, yields spectra with undesirable phase behaviour, on account of resonances with different frequencies being excited at different moments in time. There are well-established methods for overcoming this using pulse sequences featuring an initial excitation, followed by one or more refocussing FS pulses[93–97].

With knowledge of the form of the chirp pulse, the expected phase of a particular resonance is determinable, and in this section, it will be shown that well-phased spectra can be obtained from excitation with a single FS pulse when appropriate post-processing of the FID is employed. The main advantage of being able to derive spectra with desirable features from a single chirp excitation experiment is the fact that ultra-broadband spectra can be generated using with a far shorter pulse sequence than state of the art methods such as chirped, ordered pulses for ultra-broadband spectroscopy (CHORUS)[96, 97], where both a 90° chirp pulse, and two 180° chirp pulses are applied. Spectra with greater intensity, as which could include broad resonances from slowly tumbling species could therefore be realised. Here, a description of the technique is presented, followed by an illustration of its performance on a simulated and an experimental dataset. This work is fairly nascent, and while the initial results presented here show promise, there is yet to be consideration on samples of greater interest.

3.3.1 Chirp excitation

Here, focus is limited to chirp pulses which sweep from low to high frequencies. Such a pulse is parameterised by its duration τ_p (s), excitation bandwidth ΔF (Hz), and RF "amplitude" ω_{RF} (Hz). The frequencies that the pulse sweeps through are in the range $[f_{off} - {}^1/_2\Delta F, f_{off} + {}^1/_2\Delta F]$, and the rate at which the frequency of the chirp is increased (the sweep rate) is given by ${}^{\Delta F}/_{\tau_p}$. Figure 3.7 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

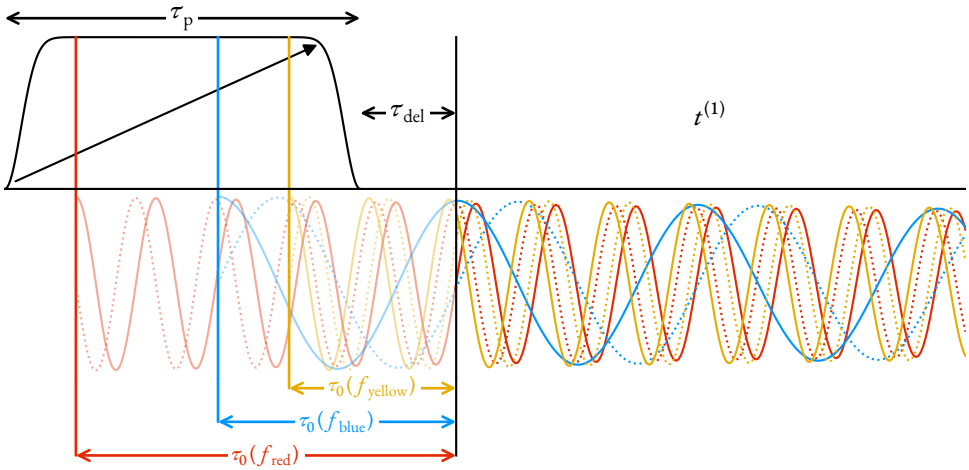


Figure 3.7: An illustration of an experiment comprising a single chirp pulse sweeping low to high frequencies of duration τ_p, followed by a pre-scan delay period or τ_{del}, prior to acquisition. The fate of three resonances with different frequencies is denoted, with $f_{red} < f_{blue} < f_{yellow}$. Each resonance is excited at different points in time, with lower frequency resonances being excited earlier, such that each resonance is allowed to evolve for different amounts of time prior to acquisition (τ_0). The resulting FID possesses quadratic phase behaviour. Coloured oscillations denote the evolution of each resonance, with solid and dashed lines representing real and imaginary components, respectively. It is assumed that the 90° chirp rotates each resonance to be initially in phase with the receiver.

the start of acquisition. While the pre-scan delay can be determined if an intimate knowledge of the spectrometer hardware is known, this varies from instrument to instrument, and is not trivial to ascertain. It is this delay which induces a first-order phase shift in NMR experiments. The various pulse parameters are inter-related as follows[97, 98]:

$$\omega_{RF} = \sqrt{\frac{\Delta F Q}{2\pi\tau_p}}, \tag{3.23}$$

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

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

such that an appropriate pulse to achieve a flip angle of 90° requires selecting a combination of ω_{RF}, ΔF, and τ_p which satisfies $Q \approx 0.441$[§§]. For a pulse with sufficiently low ω_{RF} — which requires a sufficiently large pulse duration for a given excitation bandwidth — it is reasonable to assume that the chirp pulse induces an instantaneous 90° rotation at the point of resonance, as illustrated in Figure 3.7. As such, resonances with different Larmor frequencies evolve for different amounts of time prior to the start of acquisition, according to **Double check this with Ali: his draft has +**

[§§]The combination used in examples in this work are $\omega_{RF} \approx 16.8$ kHz, $\Delta F = 400$ kHz, $\tau_p = 100$ µs

for last term, rather than −. I have −, as this makes τ_0 larger for frequencies less than the transmitter, as I would expect for a low-to-high sweep.

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

$\tau_{\text{del}} + \tau_{\text{p}}/2$ is the amount of time between excitation and detection for the on-resonance case, in which excitation occurs exactly halfway through the pulse. Resonances with an frequency smaller than the transmitter are excited earlier and hence have a larger τ_0, while the converse is true for resonances with greater frequencies. The resulting overall phase as a function of frequency can be approximated as[97] **Double check sign for quadratic term**

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

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

$$s_\phi(f) = s(f)\exp(-\mathrm{i}\phi(f)). \tag{3.27}$$

While the quadratic phase behaviour of peaks is corrected by doing this, another issue with the dataset is not addressed. For any resonance, the signal that is detected can be thought of as the difference between two signals: (a) the "complete" signal, which starts at the time of excitation, and (b) a "truncated" signal which is identical to the complete signal before acquisition, and which comprises zeros once acquisition has begun. The linear nature of the FT dictates that the resulting delayed-acquisition spectrum comprises the difference between the FTs of the complete signal and the truncated signal. The FT of a severely truncated FID is well approximated as a broad sinc function with its maximum at the resonance frequency. The appearance of the sinc "wiggle" depends on the gap between excitation and acquisition. Resonances of lower frequencies, will exhibit deeper, narrower artefacts since the signal is more significantly truncated. The result of applying quadratic phase correction is therefore a spectrum of well-phased peaks, but with major baseline distortions, particularly to the low-frequency end. Figures 3.8.b & 3.9.b both provide an example of this phenomenon.

3.3.2 Methodology

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

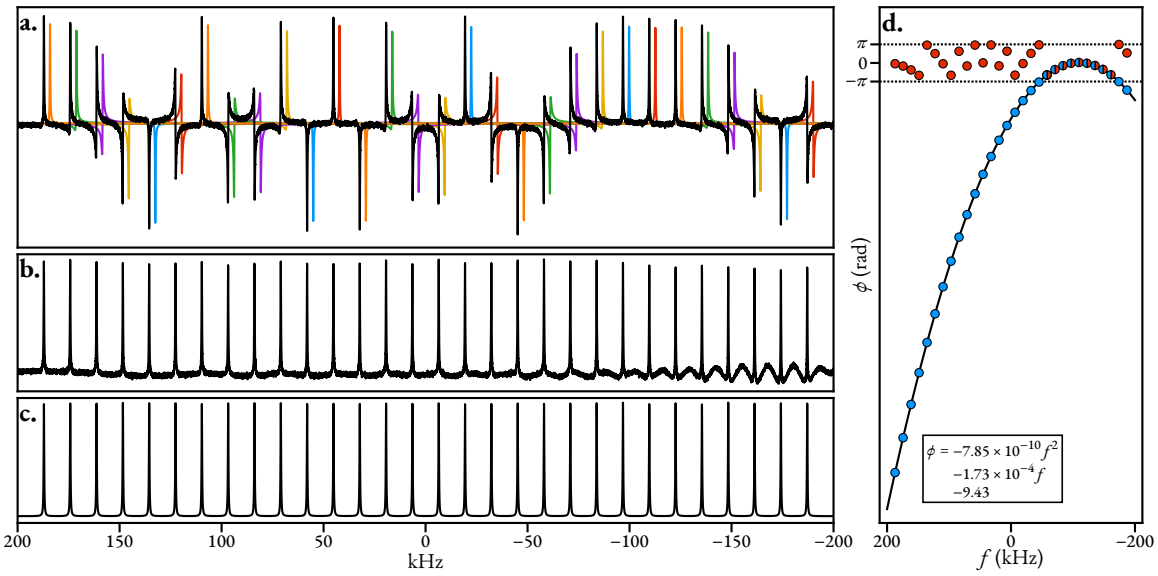


Figure 3.8: Comparison of quadratic phase correction vs frequency-dependent back-propagation in treating simulated single-chirp excitation data. **a.** Simulated spectrum for a spin system comprising 30 spins with uniformly-separated resonance frequencies. The data was generated with $N = 2^{14}$, $f_{\text{sw}} = 400\,\text{kHz}$, $f_{\text{off}} = 0\,\text{Hz}$, $\tau_{\text{p}} = 100\,\mu\text{s}$, $\tau_{\text{del}} = 0\,\text{s}$, $\Delta F = 400\,\text{kHz}$. Coloured lines depict individual signals generated using the MPM. **b.** Spectrum generated using quadratic phase correction, with (3.27). **c.** Spectrum generated from estimation using the MPM, and back-propagation. **d.** Estimated phases of each oscillator as a function of frequency. Red points: phases wrapped within the range $(-\pi, \pi]$. Blue points: the same phases, adjusted by addition of a suitable multiple of 2π to each red point in order to display the quadratic dependence of the phases. Black curve: quadratic fit of the blue points.

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

$$y_n = \sum_{m=1}^{M} a_m \exp(\mathrm{i}\phi_m) \exp\big((2\pi\mathrm{i}(f_m - f_{\text{off}}) - \eta_m)(n\Delta_t - \tau_0(f_m))\big). \tag{3.28}$$

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

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

3.3.3 Results

Figures 3.8 & 3.9 present comparisons between the application of quadratic phase correction and the proposed back-propagation procedure. In Figure 3.8, a simulated dataset is considered, comprising 30 evenly-spaced signals, and generated using (3.28) with $-\tau_0(f_m)$ replaced with $+\tau_0(f_m)$. Other relevant parameters used are stated in the caption. For the purposes of clarity, a very large damping factor ($1000\ \mathrm{s}^{-1}$) was assigned to each oscillator, as this augments the baseline distortions in the spectrum. AWGN was added to the FID, with a target SNR of 25 dB.

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

Figure 3.9 features an experimental dataset, acquired from a sample of 1% Gd-doped H_2O in D_2O. The dataset was acquired using a 2D experiment, in which the transmitter offset was adjusted for each increment. As such, FIDs with a single resonance of differing frequency from H_2O are produced across the increments, which when summed lead to the spectrum in panel a of Figure 3.9.

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

$$\tau_0' = -\frac{(f - f_\mathrm{off})\tau_\mathrm{p}}{2\Delta F}. \tag{3.29}$$

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

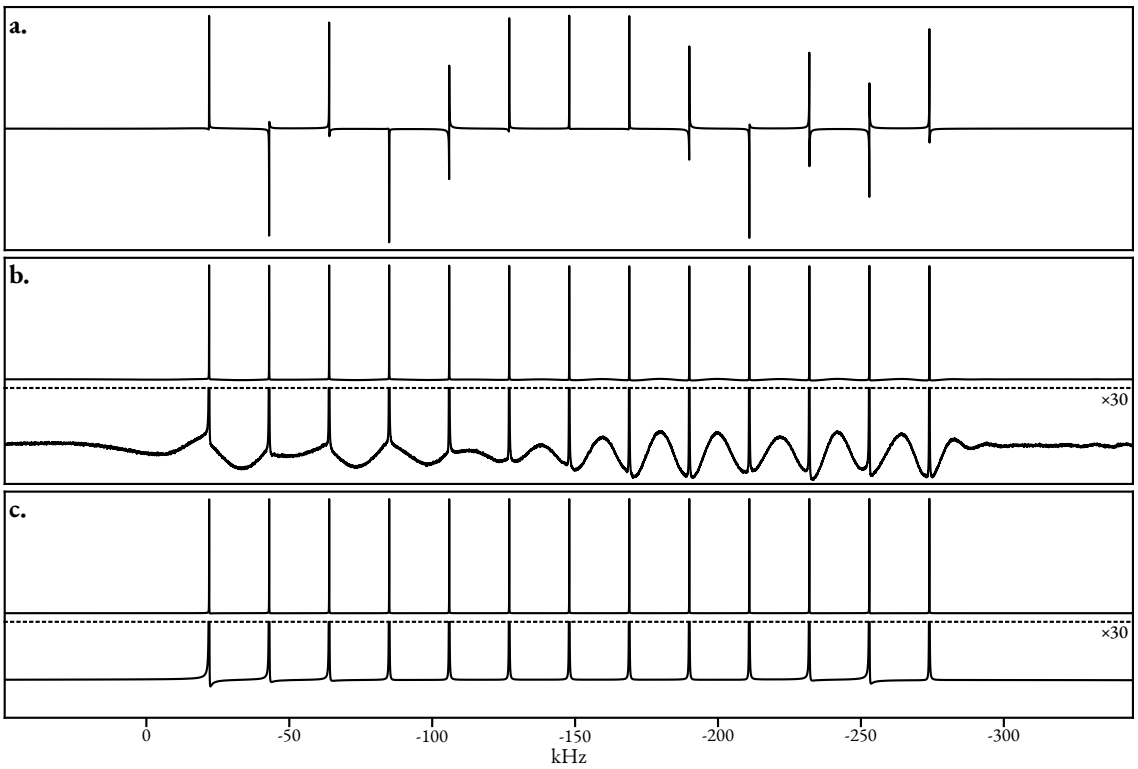


Figure 3.9: Comparison of quadratic phase correction vs frequency-dependent back-propagation in treating experimental single-chirp excitation data generated from a sample of 1% Gd-doped H_2O in D_2O. **a.** Spectrum generated directly from the acquired FID. **b.** Spectrum generated using quadratic phase correction. **c.** Spectrum generated from estimation using the MPM, followed by frequency-dependent back-propagation.

3.4 Summary

TODO

Pure shift spectra from 2D J-resolved estimation

4

Two key features of the NMR experiment for which improvements are constantly being sought are sensitivity and resolving power. There are numerous means of enhancing sensitivity, including technological advancements such as production of superconducting magnets with higher field strengths[99] (sensitivity $\propto B_0^{7/4}$) and cryogenic probes[23], as well as simply increasing the number of scans (sensitivity $\propto \sqrt{\text{no. scans}}$). However, few means of achieving better resolution exist beyond increased field strengths (resolution $\propto B_0$). Significant interest has therefore been given to the development of techniques which generate broadband homodecoupled (*pure shift*) spectra, in which the effects of homonuclear scalar couplings are absent from the data. While often valuable for structural assignment purposes, the influence of scalar couplings can lead to spectra which are too crowded for meaningful insights to be gleamed. While it is commonplace to decouple heteronuclear couplings at the point of FID acquisition[100–102], homonuclear decoupling is far more challenging. At the time of writing, there are a number of of well-established pure shift experiments, which involve running a 2D pulse sequence, and concatenating the initial sections of each FID, in a process referred to as *chunking*[103–105]. 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 subset of the available spin magnetisation contributes that which is detectable.

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

4.1 Pure Shift NMR

4.1.1 The 2D J-resolved Experiment

The 2DJ experiment[31, 32] provided the first means of achieving pure shift spectra. It has a simple pulse sequence, presented in Figure **??**.a; after excitation of magnetisation onto the transverse plane, the indirect dimension evolution consists of a spin echo, with acquisition following immediately afterwards. Fourier transformation in both dimensions leads to a spectrum in which only scalar couplings contribute in $F^{(1)}$, as the chemical shifts are refocussed by the spin echo, while both scalar couplings and chemical shifts contribute in $F^{(2)}$. An FID generated by the 2DJ experiment is hypercomplex, taking the form of (1.21) 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 by a spin system with S distinct spins. The signals giving rise to a particular spin $s \in \{1, \cdots, S\}$ form a grouping $G_s \subset \{1, \cdots, M\}$. All of the signals in G_s have (angular) frequencies given by

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

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

$\forall m \in G_s$, where $\omega_{0,s}$ is the Larmor frequency of the spin, and $\Delta\omega_m$ is the displacement of the signal from $\omega_{0,s}$, as a result of J-couplings*. Due to the relationship between the direct- and indirect-dimension frequencies, all signals which are part of the same multiplet lie along a line which bisects (i.e. makes a 45° angle with) both the $F^{(1)}$ and $F^{(2)}$ axes, as depicted in Figure 4.1. One limitation of the 2DJ experiment is the fact that spectra with pure absorption lineshapes cannot be produced. This is since, due to the absence of a mixing period, it is not possible to produce a complementary pair of phase- or amplitude-modulated FIDs, which are required to nullify dispersive contributions (see Section 1.2.2). The FT of a 2DJ FID produces a spectrum with phase-twist peaks (Figure 4.1.a). As with other experiments which produce hypercomplex signals, such as the COSY pulse sequence, the data is conventionally displayed in "magnitude-mode", in which the absolute value of each point in the spectrum is plotted.

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

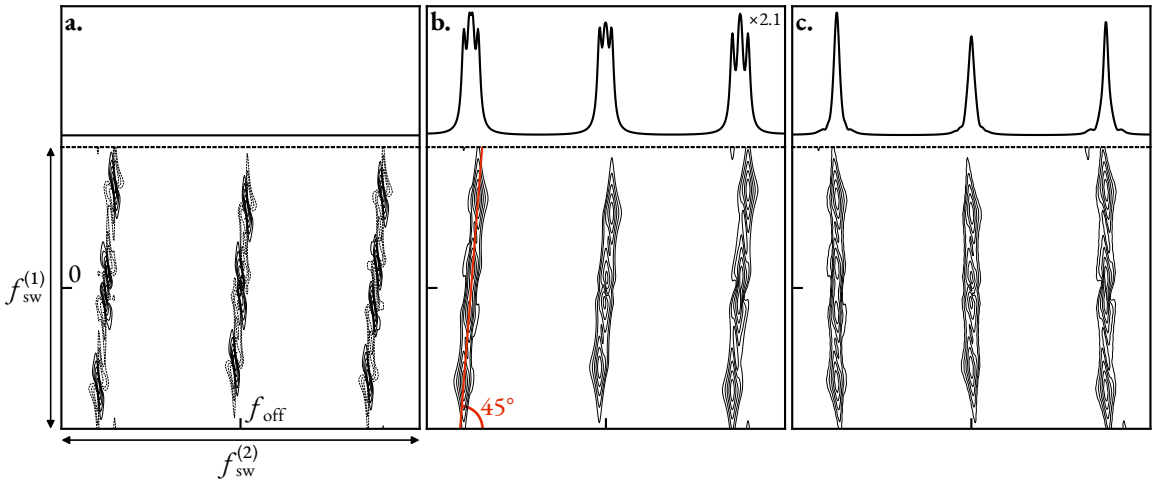


Figure 4.1: Example of a simple 2DJ spectrum derived from an AMX spin system. Each panel showcases the appearance of the spectrum following different processing procedures. Below: contour plots of the spectrum. Above: the summation of the spectrum along the indirect axis. **Put 45 line on panel a** **a.** Spectrum produced from sine-bell apodisation followed by FT in both dimensions, which features phase-twist peaks, and which sums to zero along the indirect dimension. Each multiplet lies along a line at 45° to the $F^{(1)}$ and $F^{(2)}$ axes. Note that this line often appears to make an angle that is greater than 45° with the $F^{(2)}$ axis when viewing spectra, due to the relative scaling of the frequency axes (typically, $f_{sw}^{(2)} \gg f_{sw}^{(1)}$). **b.** Magnitude-mode spectrum produced from **a.** by subjecting each point to $\sqrt{\Re(x)^2 + \Im(x)^2}$. **c.** Spectrum generated after application of a 45° shear to **b**. The summation produces a pure shift spectrum, though this features peaks with broad wings, and significant non-linearities.

A pure shift spectrum is generated from the 2DJ spectrum by performing a 45° shear (often called a tilt) on the spectrum array, leading to the separation of chemical shifts and scalar couplings onto orthogonal axes (Figure 4.1.b). Each slice in $F^{(2)}$ is subjected to a right circular rotation such that

$$s^{\text{tilt}}_{n^{(1)},n^{(2)}} = s_{n^{(1)},n^{(2)\prime}}, \tag{4.3a}$$

$$n^{(2)\prime} = \left(n^{(2)} + \left\lfloor \frac{f_{sw}^{(1)} N^{(2)}}{f_{sw}^{(2)} N^{(1)}} \left(\frac{N^{(1)}}{2} - n^{(1)} \right) \right\rceil \right) \bmod N^{(2)}. \tag{4.3b}$$

This achieves the mapping $s\left(F^{(1)}, F^{(2)}\right) \rightarrow s(F^{(1)}, F^{(2)} - F^{(1)})$, which leads to a spectrum in which all peaks belonging to a particular spin being positioned at the same direct-dimension frequency. The effectiveness of the shear is maximised when both $f_{sw}^{(2)}/f_{sw}^{(1)}$ and $N^{(2)}/N^{(1)}$ are powers of 2**check this**. Summing the sheared spectrum along $F^{(1)}$ leads to the pure shift spectrum. If the spectrum wasn't in magnitude-mode, shearing and summing would lead to the absorptive and dispersive components of the spectrum cancelling each other out, such that the zero vector would be obtained. With a magnitude-mode spectrum, the process leads to undesirable pure shift spectra with broad "wings" on account of the presence of dispersive character. The dispersive component

can be suppressed by appropriate processing to make the FID envelope symmetric in both dimensions, such as with sine-bell apodisation or pseudo-echo reshaping[106], though this results in a significant reduction in sensitivity being incurred.

4.1.2 The Zangger-Sterk Method

Zangger and Sterk introduced a pulse sequence element which achieves *slice-selective excitation*, by applying a low RF power (weak) 180° pulse[†] in the presence of a PFG along the z-axis[108]. Such an element excites a given spin only in a narrow range of heights in the sample, as the PFG induces a shift in resonance frequency according to $\Delta\omega(z) = \gamma g z$, where g is the magnitude of the PFG. By placing a hard 180° pulse adjacent to the selective pulse, the "active" spin in a given slice is rotated by 360° (i.e. no net rotation), while all other ("passive") spins are only rotated by 180°. Placing such a element in the middle of the $t^{(1)}$ evolution therefore achieves refocussing of the J-couplings associated with the active spin[109]. In order to achieve effective decoupling of any given pair of spins, it is necessary that the bandwidth of the selective π-pulse is smaller than the difference in their Larmor frequencies. However, with more selective pulses, a smaller proportion of the available spin magnetisation will contribute to the final FID, and hence sensitivity will be diminished [‡]. Therefore a trade-off exists between effective decoupling of all spins, and achieving the greatest sensitivity possible. In the case of strong coupling, the Zangger-Sterk (ZS) method tends to perform poorly relative to other options for this reason. The ZS element has utilised in order to generate 2DJ datasets comprising phase-modulated pairs, enabling the generation of pure absorption-mode spectra[110]. Pure shift spectra with far more desirable lineshapes can be achieved relative to using a typical magnitude-mode spectrum 2DJ, though with a significant loss of sensitivity.

4.1.3 The BIRD Method

The bilinear rotation decoupling (BIRD) pulse sequence element[111, 112] also takes advantage of the idea of selectively inverting passive spins, while leaving active spins unaffected. However the active spins are those which are directly bound to a low natural abundance heteronucleus, with the two most common heteronuclei used being ^{13}C (1.1% abundance) and ^{15}N (0.37% abundance) The passive spins are those bound to far more abundant nucleus (i.e. ^{12}C or ^{14}N). The reduction in sensitivity of the experiment relative to a full-sensitivity experiment is therefore known and constant across samples. In scenarios where strong coupling exists, BIRD can achieve improved sensitivity over ZS, since with the latter a very weak selective pulse would be required to ensure

[†]Conventionally, a R-SNOB pulse is used[107].

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

it is of a sufficiently small bandwidth. The BIRD method is particularly attractive in scenarios where the sensitivity penalty due to the involvement of a low-abundance nucleus has already been paid, for example in sequences where an insensitive nuclei enhancement by polarization transfer (INEPT) block is present.**CITATIONS**

4.1.4 PSYCHE

TODO pure shift yeilded by chirp excitation (PSYCHE) Original paper[113], tutorial paper[114], PSYCHE-2DJ[115, 116]: note that this is a 3D experiment therefore long run time.

4.1.5 Pure shift spectra from 2DJ estimation

Procedures based on the estimation of 2DJ datasets have been presented previously. Nuzillard introduced a linear predictive estimation of signal time reversal (ALPESTRE)[117, 118], in which the parameters of each indirect-dimension FID are estimated using linear prediction singular value decomposition (LPSVD), such that a set of parameters $\boldsymbol{\Theta} \in \mathbb{R}^{N^{(2)} \times 4M}$ is generated.

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

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

$$\begin{aligned} y^{\mathrm{full}}_{n^{(1)},n^{(2)}} = \sum_{m=1}^{M} a_{n^{(2)},m} \exp(\mathrm{i}\phi_{n^{(2)},m}) \exp\left(\left(2\pi \mathrm{i} f_{n^{(2)},m} n^{(1)} - \eta_{n^{(2)},m} \left|n^{(1)}\right|\right) \Delta_t^{(1)}\right), \\ \forall n^{(1)} \in \{-N^{(1)}+1, \cdots, 0, \cdots, N^{(1)}-1\},\ \forall n^{(2)} \{0, \cdots, N^{(2)}-1\}. \end{aligned} \tag{4.5}$$

FT of (4.5) generates a spectrum whose real component comprises absorption-mode Lorentzian character in both dimensions. This opens up the means of producing pure-shift spectra from the 2DJ experiment with sharp lineshapes and without signal loss. A similar approach proposed by Mutzenhardt et al. instead constructs full echoes via LP of each direct-dimension FID, and generates a full echo by propagating into $-t^{(2)}$ [119].

4.2 Methodology

4.2.1 The −45° signal

CUPID aims to generate pure shift spectra by utilising the result of parametric estimation of 2DJ data, assumed to take the functional form of (4.1). Instead of estimating successive 1D FIDs, as proposed by Nuzillard and Mutzenhardt et al., the entire 2DJ signal is estimated as a whole, giving

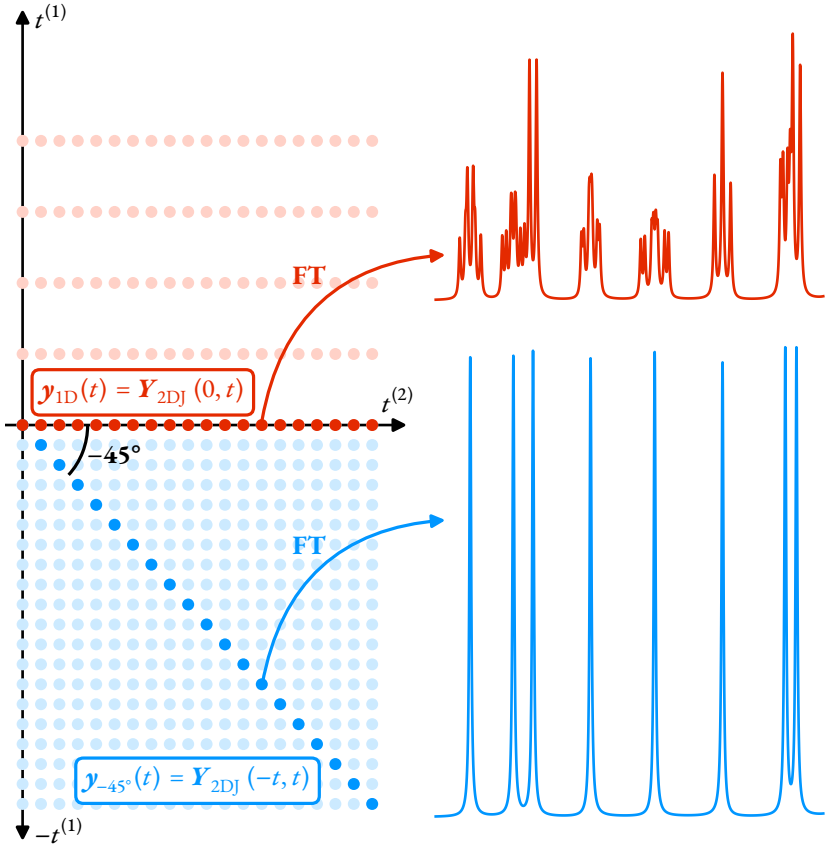


Figure 4.2: An illustration of the reasoning behind the name "−45° signal" used to generate pure shift spectra. The pale red dots denote a typical 2DJ FID, where the amount and rate of sampling in the direct dimension is greater than in the indirect dimension (i.e. $N^{(1)} \ll N^{(2)}$ and $f_{\text{sw}}^{(1)} \ll f_{\text{sw}}^{(2)}$). The bright red dots correspond to the first direct-dimension signal $\boldsymbol{Y}_{2DJ}(0, t^{(2)})$, 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 to the $t^{(2)}$ axis, yields an FID $\boldsymbol{y}_{-45°}$ which is homodecoupled. Note that there is a slight discrepancy between (4.6) and this description, in that the indirect-dimension damping factors $\boldsymbol{\eta}^{(1)}$ are neglected in the former case.

access to 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 than constructing a full-echo 2DJ signal, and subsequently shearing and summing it. The desired signal is named the "−45° signal", whose name is explained in Figure 4.2, and has the form:

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

$\forall n^{(2)} \in \{0, \cdots, N^{(2)} - 1\}$. The −45° signal takes the form of a 1D FID expected from a pulse-acquire experiment, except that the frequency of each oscillator, which would usually be $f_m^{(2)}$, is replaced with $f_m^{(2)} - f_m^{(1)}$, such that oscillators belonging to a given multiplet all provide a contribution with the frequency $\omega_{0,s}$ to the −45° signal. 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.

In using a holistic 2D estimation routine, as opposed to iteratively estimating successive 1D FIDs, a number of benefits are realised. First and foremost, multiplet structures which heavily overlap in a conventional 1D dataset become separated in the 2DJ dataset (assuming that the Larmor frequencies of the relevant spins are sufficiently different). As such, accurate resolution of the signal components in more crowded spectral regions is far more likely to be successful with a full 2D estimation. On top of this, there is an extra resolution advantage relative to the estimation of *direct* dimension FIDs. Due to the presence of a spin echo during $t^{(1)}$, signal damping effects caused by field inhomogeneities are nullified, such that damping is dictated solely by transverse relaxation (T_2). During $t^{(2)}$ however, the influence of field inhomogeneities are not corrected, such that damping occurs at a faster rate, characterised by T_2^*. As such, multiplet structures in the indirect dimension exhibit better resolution (assuming $f_{\mathrm{sw}}^{(1)}/N^{(1)}$ and $f_{\mathrm{sw}}^{(2)}/N^{(2)}$ are comparable). A further benefit comes with having access to the frequencies of each oscillator in *both* dimensions, since this allows one to group together those which belong to the same multiplet (see Section 4.2.3). Similar information can be obtained by extracting slices of a tilted 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[110] and PSYCHE-2DJ[115, 116] experiments are also able to generate individual multiplet structures, though with long three-dimensional (3D) pulse sequences, and with reduced sensitivity relative to a conventional 2DJ experiment.

4.2.2 Filtration of 2DJ data

Unlike the direct-dimension, which can often comprise sparsely distributed peaks in the Fourier domain, the indirect dimension of 2DJ datasets tends to be densely populated since all multiplet

structures are centered at 0 Hz, and rarely span beyond ±50 Hz. As such, generation of frequency-filtered sub-FIDs is limited to consideration of the direct dimension. The filtering procedure applied to 2DJ data is an extension of that procedure for 1D data described in Section 2.5, and is depicted in Figure 4.3:

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

$$\boldsymbol{y}_{\text{ve},n^{(1)}} = \left[\Re\left(\boldsymbol{y}_{n^{(1)},0}\right) \quad \boldsymbol{y}_{n^{(1)},1} \quad \cdots \quad \boldsymbol{y}_{n^{(1)},N^{(2)}-1} \quad 0 \quad \boldsymbol{y}^{*}_{n^{(1)},N^{(2)}-1} \quad \cdots \quad \boldsymbol{y}^{*}_{n^{(1)},1}\right]. \tag{4.7}$$

(ii) $\boldsymbol{Y}_{\text{ve}}$ is subjected to FT along the direct dimension to produce the spectrum $\boldsymbol{S}_{\text{ve}}$ (panel a of Figure 4.3). This has an imaginary component of zero.

(iii) A super-Gaussian $\boldsymbol{G} \in \mathbb{R}^{N^{(1)} \times 2N^{(2)}}$ is constructed (panel b):

$$\boldsymbol{G} = \boldsymbol{1} \otimes \boldsymbol{g}^{(2)}, \tag{4.8}$$

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 (2.65).

(iv) A matrix of additive noise is generated by extracting the variance σ^2 of a direct-dimension strip of $\boldsymbol{S}_{\text{ve}}$ 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 a normal distribution with mean 0 and variance σ^2.

(v) The spectrum is filtered (panel d):

$$\widetilde{\boldsymbol{S}}_{\text{ve}} = \boldsymbol{S}_{\text{ve}} \odot \boldsymbol{G} + \boldsymbol{W}_{\sigma^2} \odot (\boldsymbol{1} - \boldsymbol{G}). \tag{4.9}$$

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

4.2.3 Multiplet Prediction

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

$$\left|\left(f^{(2)}_{m_1} - f^{(1)}_{m_1}\right) - \left(f^{(2)}_{m_2} - f^{(1)}_{m_2}\right)\right| < \epsilon. \tag{4.10}$$

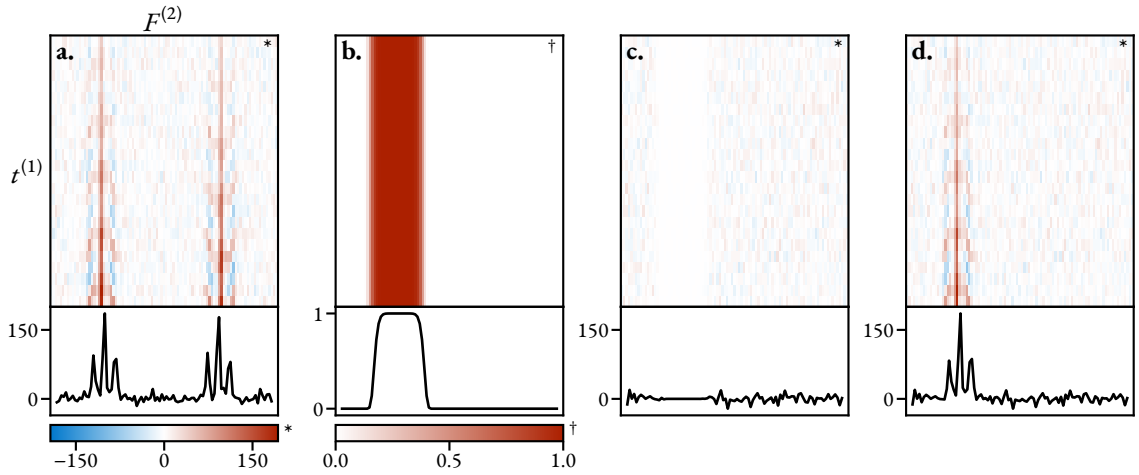


Figure 4.3: An illustration of the filtering procedure for 2DJ data. For each panel is a heat-map of the full 2D signal, as well as a plot underneath of the first slice of the signal in the direct dimension. **a.** The spectrum $\boldsymbol{S}_{\text{ve}}$, **b.** Super-Gaussian filter $\boldsymbol{G}$,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

$\epsilon \in \mathbb{R}_{>0}$ is a suitable threshold to account for error in the estimation result. A lower bound on the threshold is the separation between adjacent points in the better resolved dimension of the spectrum, i.e. $\epsilon = \min\left(f_{\text{sw}}^{(1)}/N^{(1)}, f_{\text{sw}}^{(2)}/N^{(2)}\right)$. However, limitations in resolution due to signal damping and field inhomogeneities can mean that ϵ has to be increased beyond this for reasonable multiplet assignments to be achieved. Algorithm A.6 provides a routine that can be used for multiplet prediction **Remove algorithm and include python script instead**.

Rethink this bit Spurious oscillators:

- Very broad, with $f^{(1)} \approx 0$ Hz. Caused in cases where very intense signal i.e. from solvent/residual water has a tail which "breaks into" the region of interest.
- Low intensity, random indirect frequency: overfit: probably fitting a noise component
- Around region boundary: likely due to artefacts induced by filtering

The ability to predict multiplet groupings can also assist in scenarios where the estimation result contains some oscillators with a spurious nature, typically due to overfitting. These typically possess either a very large damping factor or low amplitude, and are not associated with discernible peaks in the spectrum. Part of the reason that the variance of phases is included in the fidelity for NLP is to try and purge these oscillators, however this method is not infallible, and undesired oscillators can end up in the final result. An appreciable number of these can be removed in an automated fashion by noting that there should not be any oscillators in the estimation result of a 2DJ dataset which satisfy both of the following:

(i) The oscillator is not grouped with any other oscillator as part of the multiplet assignment.

(ii) The magnitude of the indirect dimension frequency of the oscillator is appreciably greater than 0 Hz.

These criteria are borne out of the fact that it should not be possible to have oscillators in a 2DJ dataset which are not part of a multiplet structure, unless such oscillators are singlets. As no scalar couplings contribute, these singlets should have an indirect dimension frequency of 0 Hz.

4.2.4 An overview of CUPID

The estimation routine used in CUPID is as follows:

(i) Perform appropriate pre-processing on the 2DJ dataset in the direct dimension: phase correction, and baseline correction if deemed to be necessary.

(ii) Generate frequency-filtered sub-FIDs for each direct-dimension spectral region of interest.

(iii) For each sub-FID, determine a parameter estimate by applying the MMEMPM to generate a initial guess, followed by phase variance-regularised NLP. Optionally, application of the MDL on the first direct-dimension increment of the FID can be used to estimate the model order. If the MDL is not used, an estimation of the model order must be hard-coded.

(iv) The pure shift spectrum is generated by combining the parameter estimates of each sub-FID, generating the $-45°$ signal with (4.6), and performing FT.

(v) If of interest, individual multiplet structures can also be generated.

4.3 Results

A number of examples of the application of CUPID are now provided. Initially, a few results are presented on datasets simulated using Spinach. After this, examples are provided with experimental data. In a couple of these, comparison of the result acquired using CUPID is made with a spectrum acquired using PSYCHE. For details relating to generation of the datasets, see Sections C.1 and C.2.3 in the Appendix.

4.3.1 "Four Multiplets"

A series of simulated ^{1}H 2DJ datasets were generated such that within a known region of the spectrum (-30 Hz to 30 Hz), four ddd multiplet structures with significant overlap exist. To achieve this, a spin-system with 7 spins was formed, with the spins divided into 2 subsets:

- Four of the spins (the "estimated spins") were assigned random resonance frequencies sampled from $\mathcal{U}(-20\,\text{Hz}, 20\,\text{Hz})$.

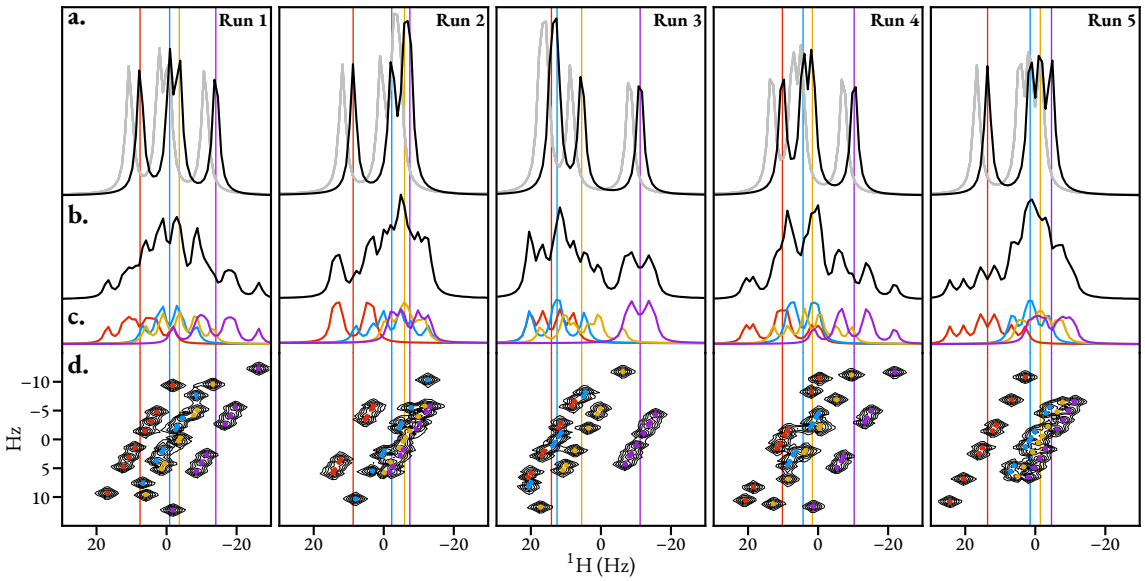


Figure 4.4: The result of applying CUPID to 5 instances of simulated 2DJ datasets with 4 heavily overlapping multiplet structures. **a.** Black: pure shift spectrum generated by CUPID (via the −45° signal). Grey: 1D spectrum simulated with Spinach, using the same spin system as was used to produce the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

- The remaining three spins (the "coupling spins"), were coupled to each of the estimated spins, with the values of the couplings randomly sampled from $\mathcal{U}(-10\,\text{Hz}, 10\,\text{Hz})$. The coupling spins were given chemical shifts such that they lay far from the estimated spins in the spectrum (i.e. their frequencies were $\gg 20$ Hz).

AWGN noise was added to the FID, with a target SNR of 30 dB. A filtered sub-FID containing only the signals from the estimated spins was then generated using the filtering procedure described above, with $l^{(2)}_{\text{Hz}} = 30$ Hz, $r^{(2)}_{\text{Hz}} = -30$ Hz. The resulting sub-FID was expected to comprise 32 (4×2^3) oscillators. To assess the estimation procedure's ability, a random integer from the range $[33, 40]$ was selected as the initial number of oscillators. Hence, the initial guess from the MMEMPM would comprise a excessive number of oscillators. Due to the severe overlap of the multiplets, application of the MDL on the first direct-dimension signal would be ineffective and return an under-estimate of the model order. Each FID was subjected to estimation, yielding the result vector $\boldsymbol{\theta}^{(*)}$. Spurious oscillators were checked for, using the criteria outlined above, with the threshold for multiplet assignment set to the spectral resolution in the direct dimension: $\epsilon = f^{(2)}_{\text{sw}}/N^{(2)}$. If spurious oscillators were found, these were removed, and NLP was run on the updated set of parameters.

Figure 4.4 illustrates the result achieved for 5 separate runs of this procedure. For each FID generated, the method was effective at producing an estimation result with 32 oscillators, as desired, despite the excessive number that were present in $\boldsymbol{\theta}^{(0)}$. Most of the excessive oscillators were purged from $\boldsymbol{\theta}^{(0)}$ through the NLP procedure. When spurious oscillators did remain[§], they were then detected when checking for spurious oscillators and subsequently removed. With simulated examples, it is easy to confirm that the pure shift spectrum generated using CUPID agrees with the expected pure shift spectrum; it is possible to generate the "true" pure shift spectrum by simulating a pulse-acquire experiment with Spinach, using a spin system with same chemical shifts, but all scalar couplings set to 0 Hz. As seen in panel a of Figure 4.4, the spectra produced using CUPID agree well with these. Panel c indicates that CUPID effectively resolved the multiplet structures associated with the dataset.

4.3.2 Sucrose simulated

Maybe not so impressive? Perhaps try strychnine? As a second example of applying CUPID on simulated data, the chemical shifts and isotropic scalar couplings associated with a Gaussian[120] density functional theory (DFT) calculation of sucrose in a vacuum [¶] were used to construct a 2DJ dataset. AWGN was added with a target SNR of 20 dB. The CUPID procedure was applied to

[§] for 2 of the 5 datasets, the result after NLP comprised 33 oscillators

[¶] It is well known that isotropic chemical shift calculations using DFT are typically very inaccurate. The resulting spectrum is not typical of sucrose in the liquid state, though this doesn't really matter for assessing the performance of CUPID.

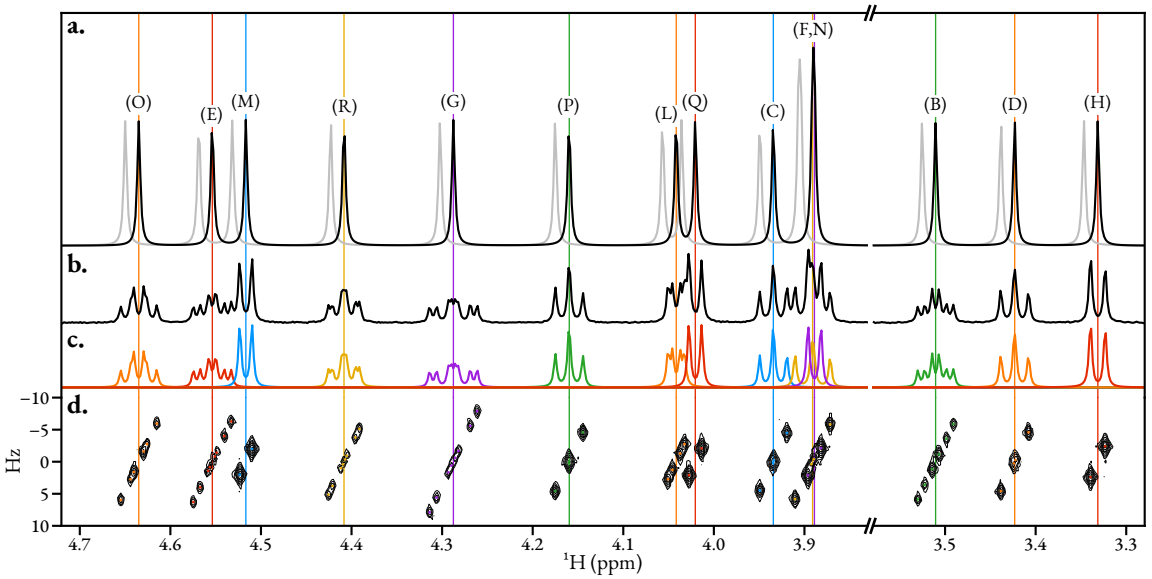


Figure 4.5: Application of CUPID on a simulated sucrose 2DJ dataset. **a.** Black: the spectrum generated from FT of the −45° signal. Grey: the spectrum of a simulated dataset with the same chemical shifts, with all scalar couplings set to 0 Hz. **b.** Conventional 1D spectrum. **c.** Multiplet structures assigned ($\epsilon \approx 0.27$ Hz). **d.** Contour plot of the absolute value mode 2DJ spectrum, with the locations of assigned oscillators given as coloured points.

filtered sub-FIDs such that the signals arising from all 22 spins were considered, though only the regions of the dataset with the most interesting multiplet structures are presented in Figure 4.5.

The estimation technique successfully assigned multiplet structures for all 22 multiplets in the dataset, including structures derived from two spins (F & N) with a 0.6 Hz difference in resonance frequency, approaching the spectral resolution in the direct dimension (0.537 Hz). The pure-shift spectrum generated via the −45° signal again showed close agreement with a 1D spectrum simulated using the same chemical shifts, with scalar couplings set to 0 Hz. There are particular multiplets where the number of oscillators fit using the estimation routine was less than the true number. Examples of this phenomenon are exhibited in the estimates of the multiplets for spins B & O, which are both ddd structures. The scalar couplings involved meant that certain oscillators were of such similar frequencies that they were separated by significantly less than the spectral resolution, and thus resolving these was unrealistic. For example, there are two pairs of peaks in the spin-B multiplet which lie only 0.085 Hz apart. Under-fitting in this case had a negligible impact on the final pure shift spectrum. However there are circumstances which will be seen in the experimental examples below where more blatant cases of under-fitting lead to the generation of peaks in the pure shift spectrum which are noticeably broadened.

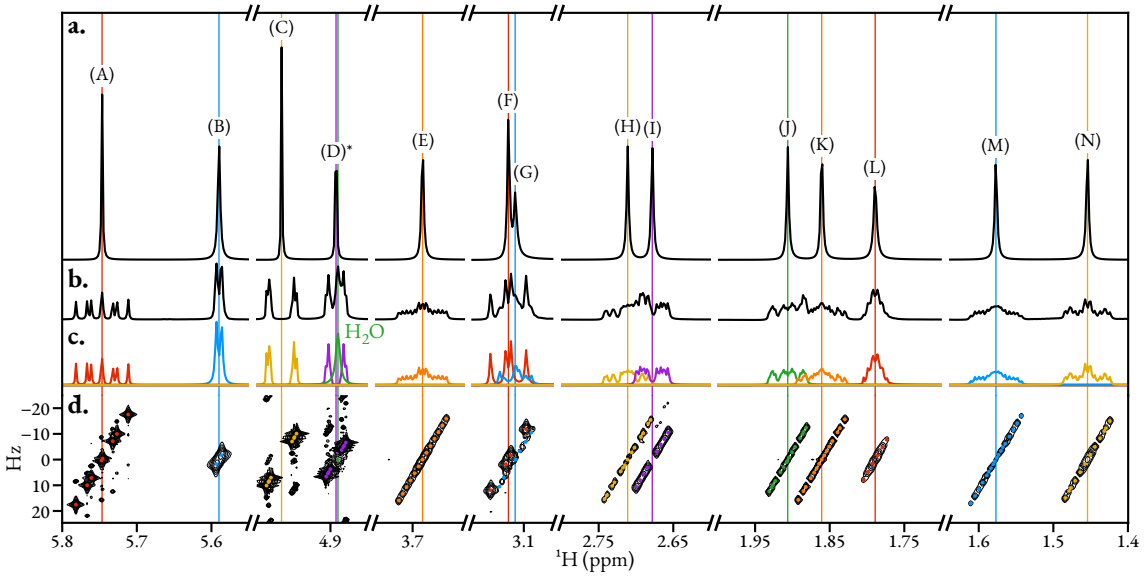


Figure 4.6: Application of CUPID on the non-aromatic regions of a quinine 2DJ dataset. **a.** The spectrum generated from FT of the −45° signal, with the signal arising from H_2O (green, close to 4.9 ppm neglected). **b.** Spectrum of the first direct-dimension signal in the 2DJ FID. **c.** Multiplet structures assigned ($\epsilon = f_{sw}^{(2)}/N^{(2)} \approx 0.92$ Hz). **d.** Contour plot of the absolute value mode 2DJ spectrum, with the locations of assigned oscillators given as coloured points.

4.3.3 Quinine

Figure 4.6 illustrates the result of applying CUPID on a dataset generated from a sample comprising quinine (Figure C.1.a) in CD_3OD, with all signals arising from non-aromatic protons considered. The method successfully generated a pure shift spectrum with distinct peaks for each ^1H environment. This example also highlights an added benefit of using CUPID; it provides the opportunity to suppress "nuisance" signals in the pure shift spectrum. In this example, an intense, broad singlet at around 4.89 ppm was detected (see the green peak at this frequency in panel c). The singlet was due to the presence of water in the sample and was a hindrance due to it overlapping heavily with the multiplet structure corresponding to spin (D). To obtain a clean singlet for spin (D) in the pure shift spectrum, the oscillator corresponding to the water signal was simply neglected from the parameter set used to generate the −45° signal. This concept of neglecting nuisance signals through post-processing has been employed extensively, with the most prominent use case being solvent suppression. The most significant component in the data (assumed to be the solvent peak), is determined through SVD or some other approach, and is automatically subtracted from the FID[121]. In this case, the water signal is not automatically purged from the parameter set to construct the final pure shift spectrum, but a knowledgeable user would be able to locate the water signal, determine that it is undesirable to include in the pure shift spectrum, and neglect it.

As eluded to already, a few of the peaks in the pure-shift spectrum are rather broad on account

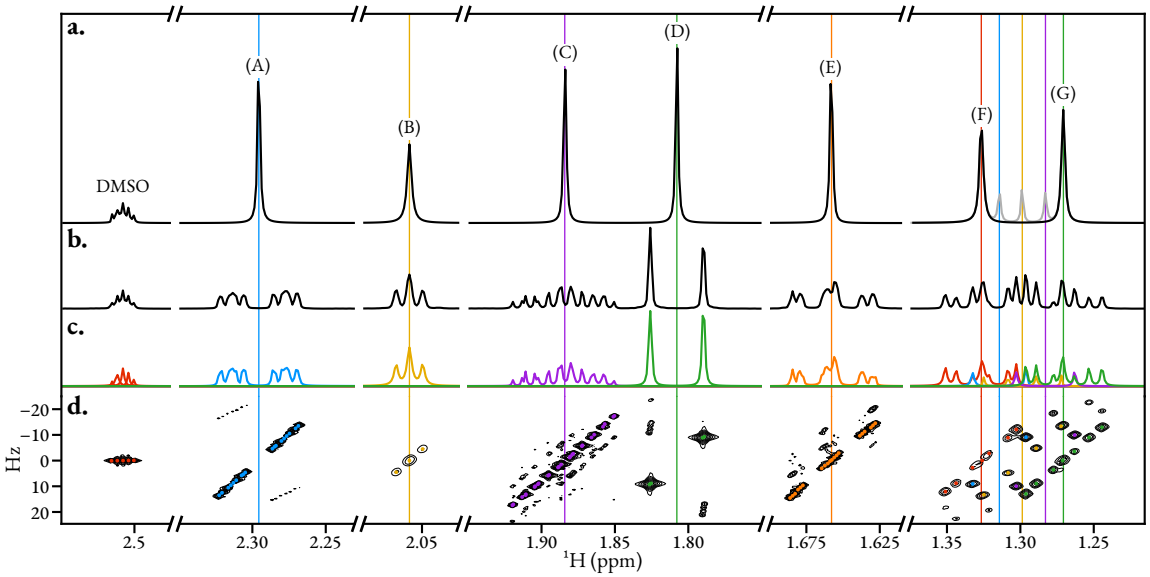


Figure 4.7: Application of CUPID on camphor 2DJ dataset. **a.** Black: the spectrum generated from FT of the −45° signal. Oscillators associated with strong coupling artefacts between spins (F) and (G) were neglected. Grey: spectrum generated without neglecting oscillators associated with strong coupling artefacts. **b.** 1D spectrum produced from the first direct-dimension FID in the dataset. Note that, unlike a conventional pulse-acquire spectrum, strong coupling artefacts are present. **c.** Multiplet structures assigned ($\epsilon = 2f_{sw}^{(2)}/N^{(2)} \approx 1.23$ Hz). **d.** Contour plot of the absolute value mode 2DJ spectrum, with the locations of assigned oscillators given as coloured points.

of the estimation routine under-fitting the relevant multiplet structure. The most notable example of this phenomenon in the quinine example comes from the peak for spin (G), where close proximity with the spin (F) multiplet has likely compounded the task of accurately estimating the relevant oscillators. With fewer oscillators than the true number of contributing signals fitting a given multiplet structure, the NLP routine will compensate by giving the oscillators it does have at its disposal large amplitudes and damping factors, so that they can reasonably fit multiple similar-frequency signals. This behaviour is also exhibited by the multiplet for spin (B), which comprises two pairs of very close signals in a dd structure. A single oscillator is fit to each pair of signals, culminating in a broadened pure shift peak. While linewdiths are affected by under-fitting hard-to-resolve multiplets, the integrals of the pure shift peaks are not typically perturbed significantly.
COMPUTE INTEGRALS

4.3.4 Camphor

Are the signals that I neglect in the final pure shift spectrum definitely due to strong coupling? The application of CUPID to the non-methyl regions of a 2DJ dataset of camphor (Figure C.1.c) in DMSO-d_6 is presented in Figure 4.7. As in the quinine case, there are instances of underfitting poorly resolved multiplets, resulting in broadened pure shift peaks. The peak associated

with spin (B) is the most drastic case here, where 4 vicinal couplings to protons with dihedral angles of 60° are present, along with potentially more contributions from long range couplings. This example highlights the ability of CUPID to remove another class of nuisance peak: *strong coupling artefacts*‖, which arise due to mixing effects induced by the 180° pulse in the 2DJ sequence[122, 123]. The effects of strong coupling lead to the presence of extra unexpected signals which do not agree with the chemical shift of any spin associated with camphor. An example of this is found between 1.35 ppm to 1.25 ppm in the 2DJ spectrum, where artefacts associated with spins (F) and (G) are located. The estimation routine was able to determine parameters for the more intense signals which comprise the strong coupling artefacts (these are coloured blue, yellow, and purple, while oscillators associated with the true multiplet structures for (F) and (G) are coloured red and green, respectively). Inclusion of all oscillators extracted by the estimation routine generates the spectrum in panel a, with the low-intensity grey peaks associated with the strong coupling effects included. However, in much the same way as the water signal in the quinine example could be neglected, it is trivial to construct the −45° signal with the oscillators associated with strong coupling artefacts left out, which produces the black spectrum.

Double check mp thold

4.3.5 Dexamethasone

Figure 4.8 shows the result of applying CUPID on a dataset acquired from a sample dexamethasone in DMSO-d_6. A pure-shift spectrum was also acquired using the triple spin echo PSYCHE (TSE-PSYCHE) experiment[114, 115] for comparison. CUPID generated a pure-shift spectrum with overall excellent agreement with the TSE-PSYCHE spectrum. Certain multiplet structures in the spectrum exhibit splitting in the direct dimension, on account of heteronuclear couplings to ^{19}F. Most notable are those derived from spins (D), (H) & (O). For the (D) multiplet, the magnitude of the heterocoupling is very small such that assigning these to separate oscillators was not achievable. For the spin (N) multiplet, two separate structures were successfully assigned (see the orange and green multiplets around 2.1 ppm). The estimation routine was unsuccessful at accurately estimating the structure associated with spin (H), where a severe under-fitting occurred. An under-fitting of this structure even occurred when the estimation was re-run using considerable over-estimation of the model order, with most oscillators in the initial guess being purged during the NLP procedure. The spin (H) multiplet provides an extreme example line-broadening in the pure shift spectrum on account of under-fitting. The most downfield peaks in the CUPID spectrum (corresponding to aromatic and hydroxyl protons) also appear to be noticeably broadened relative to their PSYCHE equivalents. This is also probably due to under-fitting of the relevant

‖As stressed in [122], 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.

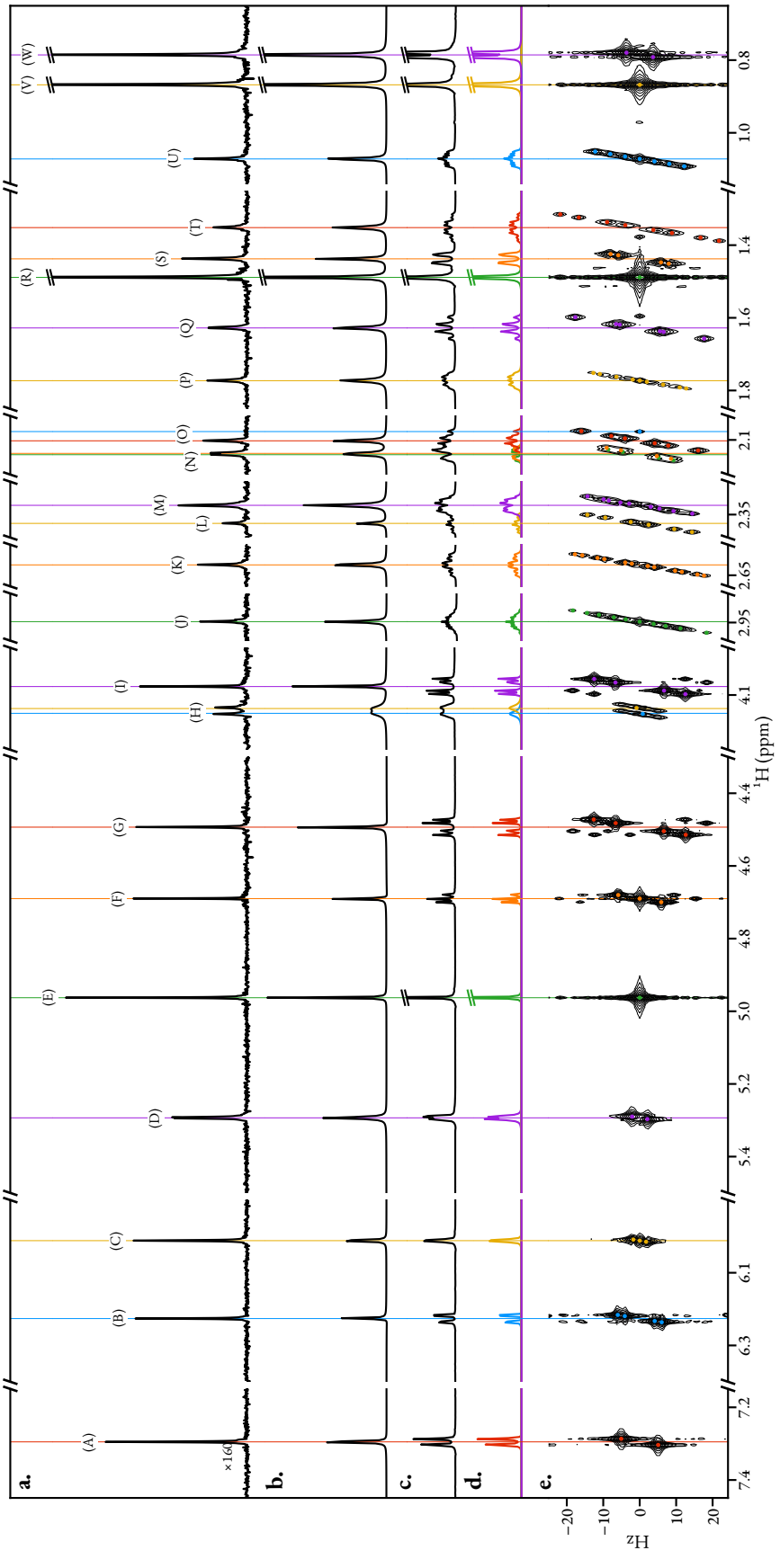


Figure 4.8: Fix magnification label. Application of CUPID on dexamethasone 2DJ dataset. **a.** TSE-PSYCHE spectrum of the sample. **b.** The spectrum generated from FT of the −45° signal. **c.** Conventional 1D spectrum. **.** Multiplet structures assigned ($\epsilon = f_{sw}^{(2)}/N^{(2)} \approx 0.92\,\text{Hz}$). **d.** Contour plot of the absolute value mode 2DJ spectrum, with the locations of assigned oscillators given as coloured points.

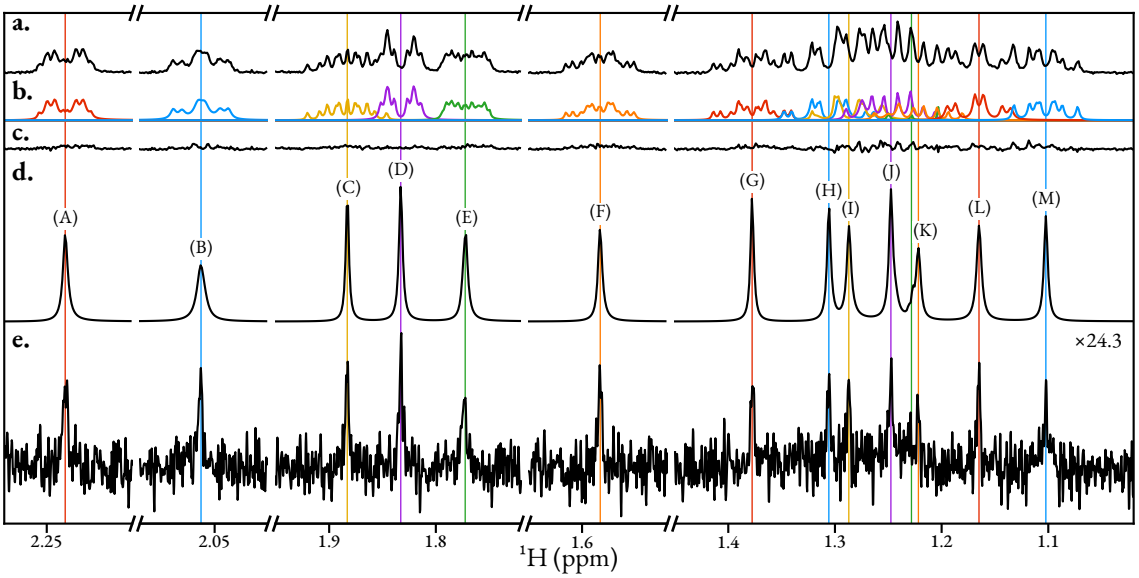


Figure 4.9: Application of CUPID on 2DJ dataset of 17β-estradiol in DMSO-d_6. **a.** Spectrum of the first direct-dimension FID in the 2DJ dataset. **b.** Multiplet structures assigned ($\epsilon = f_{sw}^{(2)}/N^{(2)} \approx 2$ Hz). **c.** The residual between the spectrum in panel a and the lines in panel b. **d.** The pure shift spectrum generated using CUPID. **e.** PSYCHE spectrum of the sample see Figure C.4 for details on the pulse sequence. The spectrum has been scaled such that the maximum is of the same magnitude as the corresponding point in the CUPID spectrum.

multiplet structures, though to a far less noticeable extent than for spin H. **Any other reason why this might be so?**

4.3.6 Estradiol

A final showcase of CUPID is provided by Figure 4.9, where a low concentration (2 mM) sample of 17β-estradiol (Figure C.1.

4.4 Summary

TODO

Software

5

The estimation routine and applications emerging from it that have been presented in the previous three chapters are accessible via the NMR-EsPy package. NMR-EsPy aims to provide a feature-rich yet simple interface in order perform estimation on datasets of interest. In this chapter, a short overview of the package is given, along with the accompanying graphical user interface (GUI). A rigorous description of the usage of NMR-EsPy, including details on installation, and a reference for the application programming interface (API) are given in the documentation for NMR-EsPy. The most up-to-date version of the documentation can be found at:

HTML: `https://foroozandehgroup.github.io/NMR-EsPy/`

PDF: ???

The source code for NMR-EsPy is hosted on the Foroozandeh group's GitHub page at `https://github.com/foroozandehgroup/NMR-EsPy`. One chapter of the documentation, which provides tutorials on the basic usage of NMR-EsPy can be found in the appendix (Chapter D).

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, beyond the fact that the candidate had a decent familiarity with it when starting their PhD:

- It has a large user-base, particularly within the scientific community.
- The SciPy ecosystem[124], including the packages NumPy[125] and Matplotlib[126] is a powerful tool which enables high-performance scientific computation in Python, despite the language's reputation for performing slowly.

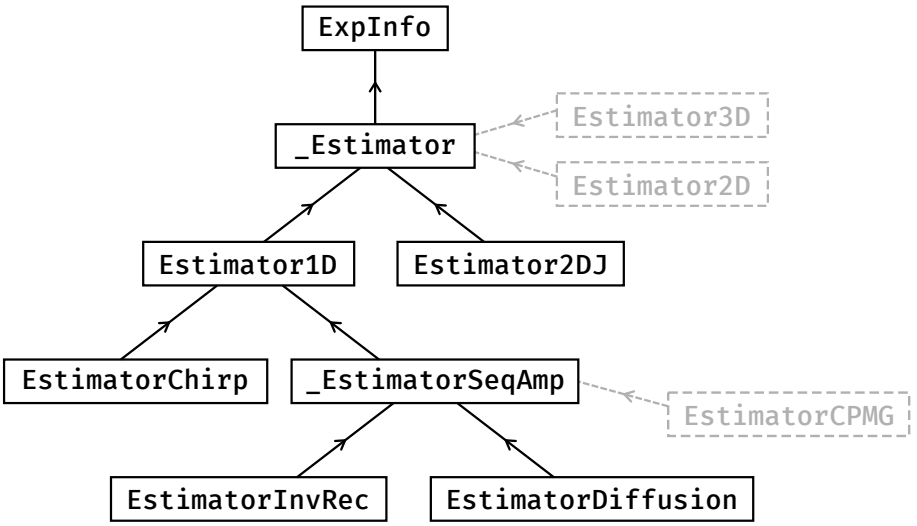


Figure 5.1: Inheritance tree for estimation classes in the NMR-EsPy package. Arrows are directed from child classes to the parent class they directly inherit from. Classes in grey are objects that could be added to the API in the future.

- Being a scripting language makes Python ideal for exploring datasets in a step-by-step fashion. This is useful in the context of NMR-EsPy, as a user will want to (i) inspect and pre-process then data, (ii) determine the regions they wish to estimate setup the estimation routine, (iii) output the estimation 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, such as multiple levels of inheritance. This is exploited in NMR-EsPy in order to create numerous objects designed with a specific NMR data types in mind.

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

5.1.2 Estimator objects

The fundamental user-facing objects (or classes) that NMR-EsPy provides are *estimators*. Instances of these estimator objects contain relevant *attributes* (the FID, experiment parameters like

the sweep width, transmitter offset etc), as well as *methods* which perform useful functions like estimation, figure generation etc. Thanks to Python's support for multiple levels of inheritance, it is possible to build numerous objects with certain shared attributes and methods, but that also have their own bespoke features that are solely of relevance to the type of NMR data being considered. Figure 5.1 is an inheritance diagram for the estimators in NMR-EsPy.

`ExpInfo`: Stores parameters used in a particular NMR experiment. `ExpInfo` also has methods for the generation of the timepoints/chemical shifts sampled by the experiment, and for producing synthetic FIDs, if given a set of oscillator parameters.

`_Estimator`: Also possesses the NMR data as an attribute on top of the experiment information. This class is designed to contain the functionality which can be generalised across all NMR data types considered. It does not however possess all the features necessary to useful as a standalone object, and as such the user is not permitted to use it directly (such an object is referred to as an *abstract class*). For child classes of `_Estimator`, the main feature that requires methods to be defined is the means by which the data is imported (experimental data) or generated (simulated data).

`Estimator1D`: 1D datasets, such as those in Section 3.1 are analysed using this class.

`Estimator2DJ`: Estimation of 2DJ data is performed with this class. It also provides features to generate ure shift spectra using CUPID (Chapter 4).

`_EstimatorSeqAmp`: An abstract class which enables the estimation of sequential 1D datasets as outlined in Section 3.2. Analysis of the dataset varies depending on the type of experiment considered, such as the function used for fitting amplitudes (Table 3.2), and the means by which that data should be imported/generated. As such, this is not designed for direct use. One of the classes which inherit it should be used instead.

`EstimatorInvRec`: For the estimation of inversion recovery (T_1) datasets.

`EstimatorDiffusion`: For the estimation of diffusion datasets.

Other estimator objects which are yet to be implemented, but are potential future additions to the package are also depicted in Figure 5.1. `Estimator2D` and `Estimator3D` would be for the estimation of datasets comprising sets of 2D or 3D amplitude- or phase-modulated FIDs, respectively. A discussion of the feasibility of implementing these is made in Section 6.2.

5.2 The NMR-EsPy GUI

Coloured rectangles in the plot indicate regions of the spectrum to generate frequency-filtered FIDs from, the grey rectangle defines the noise region.

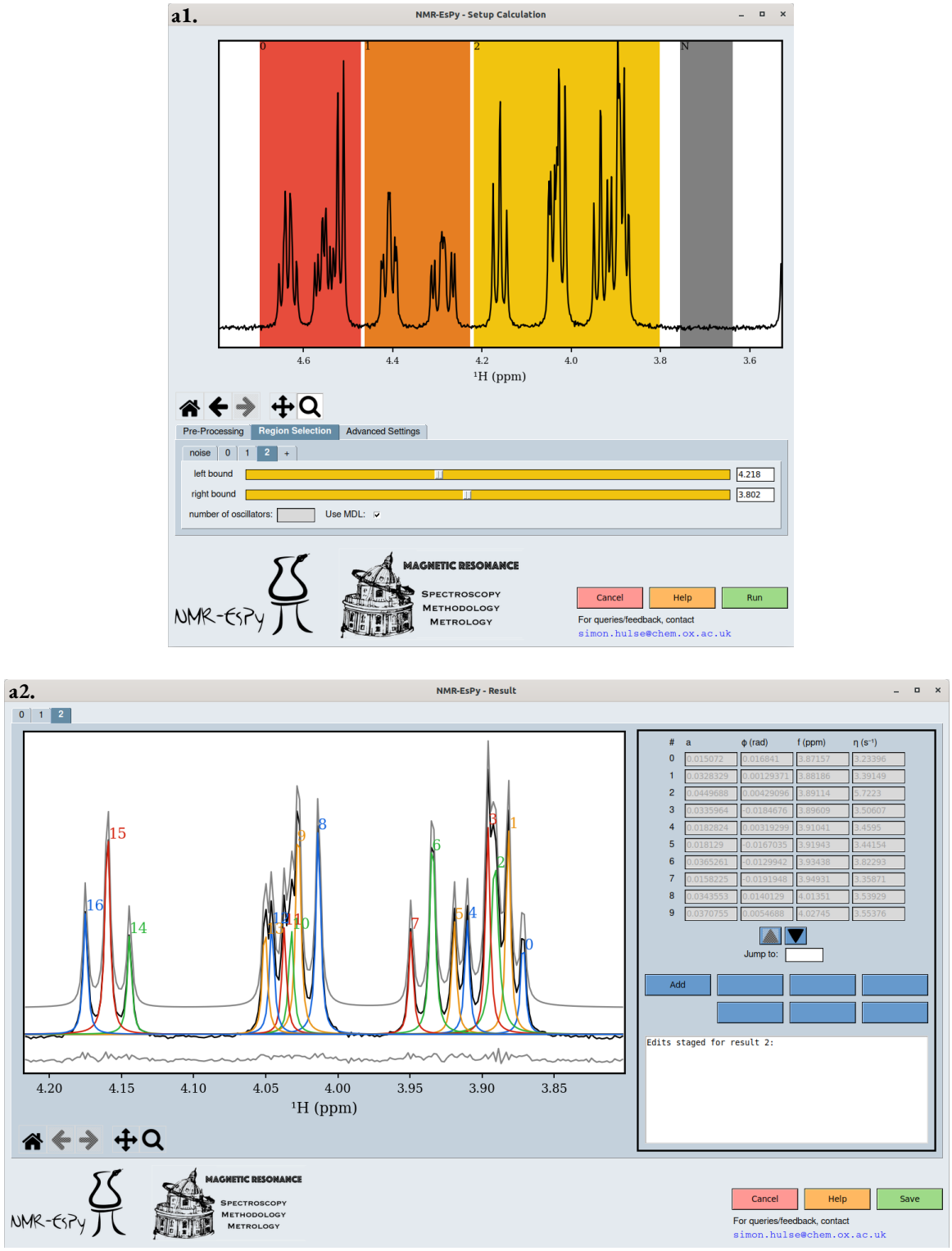
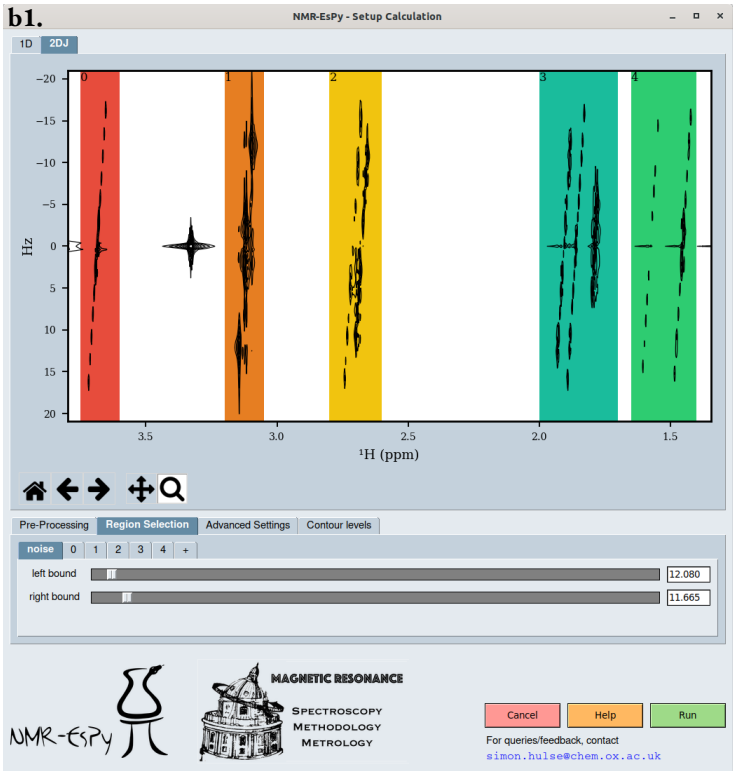
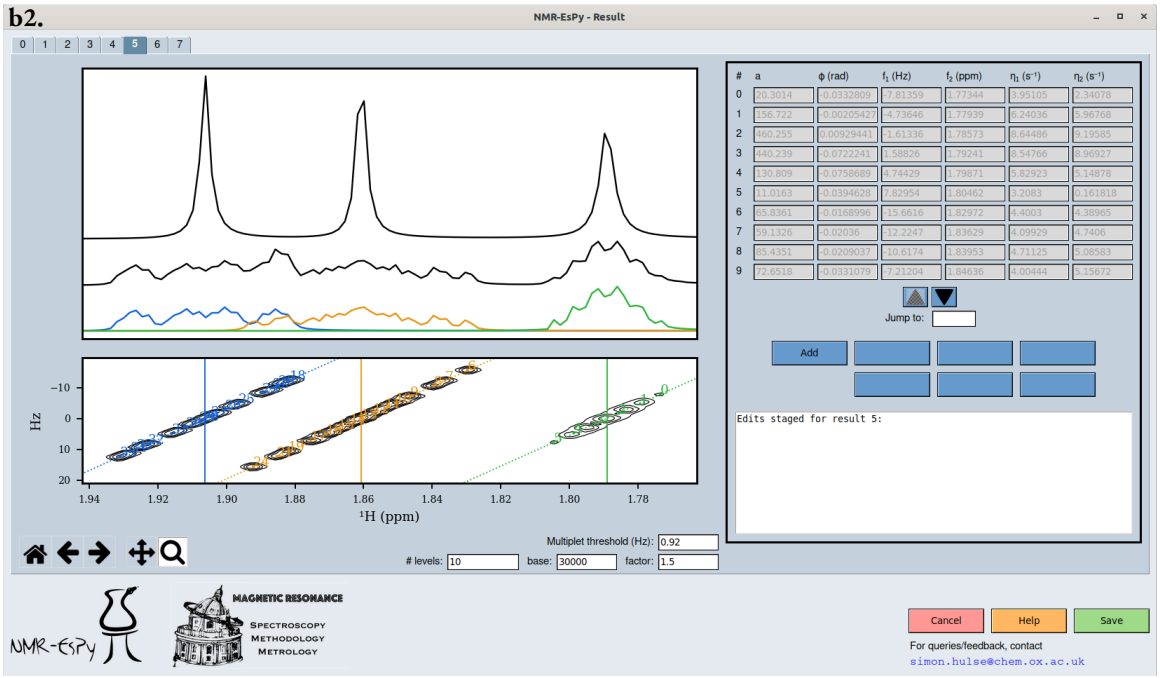


Figure 5.2: Screenshots of windows that form part of the NMR-EsPy GUI for 1D estimation (**a.**) and 2DJ estimation (**b.**). For both data types, the windows used to setup the estimation routine (**1.**) and inspect the result (**2.**) are shown. (Continues on the next page)





Continuation of **Figure 5.2**.

Conclusions and Future Work

6

6.1 Conclusions

6.2 Future Work

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

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

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

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

which the direction (i.e. azimuth and elevation angles) as well as the frequency of waves arriving at an antenna are sought[127]. However, there is a key issue with multidimensional NMR signals which would likely hamper the proposed method in this thesis from being effective. 2DJ datasets are rather anomalous in that they have very densely populated indirect dimensions. For this reason, it was necessary to be concerned about filtering the data in the indirect dimension. However, for most 2D datasets, it would be desirable to filter the data in both dimensions. While the filtering procedure presented works well on direct-dimension FIDs, which have usually decayed to noise at the end of acquisition, indirect dimension FIDs are often truncated. The FT of such signals without any treatment would produce spectra with considerable truncation artefacts. Filtering such a spectrum would incorporate undesirable artefacts into the final filtered FID, rendering it worthless. Compensating for the truncation could be achieved, by applying exponential apodisation prior to FT. However, this would then lead to spectra with broader lineshapes, such that wider spectral regions need to be selected to ensure practically all of the peaks of interest lie within it. Dealing with truncated signals could possibly be handled in one of two ways, beyond running the experiment to acquire the data with a huge number of increments. One possibility could be to propagate the FID further in time using LP. Another option might be to apply conventional apodisation, such as a sine-bell function to the FID, producing a FID without truncation artefact, and with acceptable resolution. Applying a non-exponential weighting to the dataset would however render the data incompatible with the estimation routine in its current form, such that a modified routine with a suitable model would need to be implemented. For 3D estimation, the computational burden required would also likely be too high for the routine to be practicable. **Once 2D profiling has been done, perhaps this could be used to guestimate typical 3D running times?**

Bibliography

[1] F. Bloch, W. W. Hansen, and M. Packard. "Nuclear Induction". In: *Phys. Rev.* 69 (3-4 Feb. 1946), pp. 127–127.

[2] E. M. Purcell, H. C. Torrey, and R. V. Pound. "Resonance absorption by nuclear magnetic moments in a solid". In: *Physical review* 69.1-2 (1946), p. 37.

[3] E. D. Becker. "A brief history of Nuclear Magnetic Resonance". In: *Analytical Chemistry* 65.6 (1993), 295A–302A.

[4] G. R. Eaton, S. S. Eaton, and K. M. Salikhov. *Foundations of modern EPR*. Singapore ; London: World Scientific, 1998.

[5] W. D. Knight. "Nuclear Magnetic Resonance Shift in Metals". In: *Phys. Rev.* 76 (8 Oct. 1949), pp. 1259–1260.

[6] W. G. Proctor and F. C. Yu. "The Dependence of a Nuclear Magnetic Resonance Frequency upon Chemical Compound". In: *Phys. Rev.* 77 (5 Mar. 1950), pp. 717–717.

[7] W. C. Dickinson. "Dependence of the ^{19}F Nuclear Resonance Position on Chemical Compound". In: *Phys. Rev.* 77 (5 Mar. 1950), pp. 736–737.

[8] R. R. Ernst and W. A. Anderson. "Application of Fourier Transform Spectroscopy to Magnetic Resonance". In: *Review of Scientific Instruments* 37.1 (1966), pp. 93–102.

[9] R. Freeman and G. A. Morris. "The Varian story". In: *Journal of Magnetic Resonance* 250 (2015), pp. 80–84.

[10] J. W. Cooley and J. W. Tukey. "An Algorithm for the Machine Calculation of Complex Fourier Series". In: *Mathematics of Computation* 19.90 (1965), pp. 297–301.

[11] J. Jeener. "Ampere international summer school". In: *Basko Polje, Yugoslavia* 197.1 (1971).

[12] J. Jeener and A. G. ""Pulse pair technique in high resolution NMR" a reprint of the historical 1971 lecture notes on two-dimensional spectroscopy". In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 94-95 (2016), pp. 75–80.

[13] W. P. Aue, E. Bartholdi, and R. R. Ernst. "Two-dimensional spectroscopy. Application to nuclear magnetic resonance". In: *The Journal of Chemical Physics* 64.5 (1976), pp. 2229–2246.

[14] M. P. Williamson, T. F. Havel, and K. Wüthrich. "Solution conformation of proteinase inhibitor IIA from bull seminal plasma by 1H nuclear magnetic resonance and distance geometry". In: *Journal of Molecular Biology* 182.2 (1985), pp. 295–315.

[15] D. Marion, P. C. Driscoll, L. E. Kay, P. T. Wingfield, A. Bax, A. M. Gronenborn, and G. M. Clore. "Overcoming the overlap problem in the assignment of proton NMR spectra of larger proteins by use of three-dimensional heteronuclear proton-nitrogen-15 Hartmann-Hahn-multiple quantum coherence and nuclear Overhauser-multiple quantum coherence spectroscopy: application to interleukin 1.beta." In: *Biochemistry* 28.15 (1989), pp. 6150–6156.

[16] L. Kay, G. Clore, A. Bax, and A. Gronenborn. "Four-dimensional heteronuclear triple-resonance NMR spectroscopy of interleukin-1 beta in solution". In: *Science* 249.4967 (1990), pp. 411–414.

[17] K. Pervushin, R. Riek, G. Wider, and K. Wüthrich. "Attenuated T2 relaxation by mutual cancellation of dipole–dipole coupling and chemical shift anisotropy indicates an avenue to NMR structures of very large biological macromolecules in solution". In: *Proceedings of the National Academy of Sciences* 94.23 (1997), pp. 12366–12371.

[18] R. R. Ernst. "Nuclear Magnetic Resonance Fourier Transform Spectroscopy (Nobel Lecture)". In: *Angewandte Chemie International Edition in English* 31.7 (1992), pp. 805–823.

[19] K. Wüthrich. "NMR Studies of Structure and Function of Biological Macromolecules (Nobel Lecture)". In: *Angewandte Chemie International Edition* 42.29 (2003), pp. 3340–3363.

[20] P. Abragam and A. Abragam. *The Principles of Nuclear Magnetism*. Comparative Pathobiology - Studies in the Postmodern Theory of Education. Clarendon Press, 1961.

[21] M. Goldman. *Quantum description of high-resolution NMR in liquids*. eng. International series of monographs on chemistry ; 15. Oxford: Clarendon Press, 1988.

[22] J. Cavanagh. *Protein NMR spectroscopy : principles and practice*. eng. 2nd ed. Burlington, Mass. ; London: Academic Press, 2007.

[23] H. Kovacs, R. Kuemmerle, D. Moskau, and B. Perrone. "Cryogenic NMR Probes". In: *Encyclopedia of Biophysics*. Ed. by G. Roberts and A. Watts. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, pp. 1–8.

[24] J. Keeler. *Understanding NMR spectroscopy*. eng. 2nd ed. Chichester: Wiley, 2010.

[25] C. E. Shannon. "Communication in the Presence of Noise". In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21.

[26] C. Tang. "An Analysis of Baseline Distortion and Offset in NMR Spectra". In: *Journal of Magnetic Resonance, Series A* 109.2 (1994), pp. 232–240.

[27] P. J. Hore. *Nuclear magnetic resonance*. eng. Second edition. Oxford chemistry primers. Oxford, 2015.

[28] W. Dietrich, C. H. Rüdel, and M. Neumann. "Fast and precise automatic baseline correction of one- and two-dimensional nmr spectra". In: *Journal of Magnetic Resonance (1969)* 91.1 (1991), pp. 1–11.

[29] J. Carlos Cobas, M. A. Bernstein, M. Martín-Pastor, and P. G. Tahoces. "A new general-purpose fully automatic baseline-correction procedure for 1D and 2D NMR data". In: *Journal of Magnetic Resonance* 183.1 (2006), pp. 145–151.

[30] J. Keeler and D. Neuhaus. "Comparison and evaluation of methods for two-dimensional NMR spectra with absorption-mode lineshapes". In: *Journal of Magnetic Resonance (1969)* 63.3 (1985), pp. 454–472.

[31] W. Aue, J. Karhan, and R. Ernst. "Homonuclear broad band decoupling and two-dimensional J-resolved NMR spectroscopy". In: *The Journal of Chemical Physics* 64.10 (1976), pp. 4226–4227.

[32] G. A. Morris. "Two-Dimensional J -Resolved Spectroscopy". In: *eMagRes*. John Wiley & Sons, Ltd, 2009.

[33] A. L. Davis, J. Keeler, E. D. Laue, and D. Moskau. "Experiments for recording pure-absorption heteronuclear correlation spectra using pulsed field gradients". In: *Journal of Magnetic Resonance (1969)* 98.1 (1992), pp. 207–216.

[34] D. S. Stephenson. "Linear prediction and maximum entropy methods in NMR spectroscopy". In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 20.6 (1988), pp. 515–626.

[35] P. Koehl. "Linear prediction spectral analysis of NMR data". In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 34.3 (1999), pp. 257–299.

[36] G. U. YULE. "On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers". eng. In: *Philosophical transactions of the Royal Society of London. Series A, Containing papers of a mathematical or physical character* 226 (1927), pp. 267–298.

[37] G. T. Walker. "On periodicity in series of related terms". eng. In: *Proceedings of the Royal Society of London. Series A, Containing papers of a mathematical and physical character* 131.818 (1931), pp. 518–532.

[38] N. Levinson. "The Wiener (Root Mean Square) Error Criterion in Filter Design and Prediction". In: *Journal of Mathematics and Physics* 25.1-4 (1946), pp. 261–278.

[39] J. Durbin. "The Fitting of Time-Series Models". eng. In: *Revue de l'Institut international de statistique* 28.3 (1960), p. 233.

[40] R. Kumaresan and D. Tufts. "Estimating the parameters of exponentially damped sinusoids and pole-zero modeling in noise". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 30.6 (1982), pp. 833–840.

[41] R. Kumaresan. "On the zeros of the linear prediction-error filter for deterministic signals". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 31.1 (1983), pp. 217–220.

[42] H. Barkhuijsen, R. De Beer, W. M. M. J. Bovee, J. H. N. Creyghton, and D. Van Ormondt. "Application of linear prediction and singular value decomposition (LPSVD) to determine NMR frequencies and intensities from the FID". In: *Magnetic Resonance in Medicine* 2.1 (1985), pp. 86–89.

[43] G. H. Golub and V. Pereyra. "The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate". In: *SIAM Journal on Numerical Analysis* 10.2 (1973), pp. 413–432.

[44] J. W. C. van der Veen, R. de Beer, P. R. Luyten, and D. van Ormondt. "Accurate quantification of in vivo 31P NMR signals using the variable projection method and prior knowledge". In: *Magnetic Resonance in Medicine* 6.1 (1988), pp. 92–98.

[45] R. Penrose. "A generalized inverse for matrices". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 51.3 (1955), pp. 406–413.

[46] G. Strang. *Linear Algebra and Its Applications*. 5 ed. Florence: Cengage Learning, Inc, 2018.

[47] K. Levenberg. "A method for the solution of certain non-linear problems in least squares". In: *Quarterly of Applied Mathematics* 2.2 (1944), pp. 164–168.

[48] D. W. Marquardt. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.

[49] L. Vanhamme, A. van den Boogaart, and S. Van Huffel. "Improved Method for Accurate and Efficient Quantification of MRS Data with Use of Prior Knowledge". In: *Journal of Magnetic Resonance* 129.1 (1997), pp. 35–43.

[50] K. Krishnamurthy. "CRAFT (complete reduction to amplitude frequency table) – robust and time-efficient Bayesian approach for quantitative mixture analysis by NMR". In: *Magnetic Resonance in Chemistry* 51.12 (2013), pp. 821–829.

[51] K. Krishnamurthy, A. M. Sefler, and D. J. Russell. "Application of CRAFT in two-dimensional NMR data processing". In: *Magnetic Resonance in Chemistry* 55.3 (2017), pp. 224–232.

[52] K. Krishnamurthy. "Complete Reduction to Amplitude Frequency Table (CRAFT)—A perspective". In: *Magnetic Resonance in Chemistry* 59.8 (2021), pp. 757–791.

[53] N. Schmid, S. Bruderer, F. Paruzzo, G. Fischetti, G. Toscano, D. Graf, M. Fey, A. Henrici, V. Ziebart, B. Heitmann, H. Grabner, J. Wegner, R. Sigel, and D. Wilhelm. "Deconvolution of 1D NMR spectra: A deep learning-based approach". In: *Journal of Magnetic Resonance* 347 (2023), p. 107357.

[54] R. Fletcher. *Practical methods of optimization*. eng. 2nd ed. Chichester: Wiley, 1987.

[55] J. Nocedal and S. J. Wright. *Numerical optimization*. 2nd ed. Springer series in operations research. New York: Springer, 2006.

[56] Y. Hua and T. Sarkar. "Matrix pencil method for estimating parameters of exponentially damped/undamped sinusoids in noise". In: *IEEE Trans. on Acoust. Speech Signal Process.* 38.5 (1990), pp. 814–824.

[57] Y. Hua and T. Sarkar. "Matrix pencil and system poles". In: *Signal Processing* 21.2 (1990), pp. 195–198.

[58] Y. Hua and T. Sarkar. "On SVD for estimating generalized eigenvalues of singular matrix pencil in noise". In: *IEEE Trans. Signal Process.* 39.4 (1991), pp. 892–900.

[59] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013.

[60] Y. Hua. "Estimating two-dimensional frequencies by matrix enhancement and matrix pencil". eng. In: *IEEE transactions on signal processing* 40.9 (1992), pp. 2267–2280.

[61] F.-J. Chen, C. Fung, C.-W. Kok, and S. Kwong. "Estimation of Two-Dimensional Frequencies Using Modified Matrix Pencil Method". eng. In: *IEEE transactions on signal processing* 55.2 (2007), pp. 718–724.

[62] H. Akaike. "A new look at the statistical model identification". In: *IEEE Trans. Automat. Control* 19.6 (1974), pp. 716–723.

[63] G. Schwarz. "Estimating the Dimension of a Model". In: *Ann. Stat.* 6.2 (1978), pp. 461–464.

[64] J. Rissanen. "Modeling by shortest data description". In: *Automatica* 14.5 (1978), pp. 465–471.

[65] M. Wax and T. Kailath. "Detection of signals by information theoretic criteria". In: *IEEE Trans. Acoust. Speech Signal Process.* 33.2 (1985), pp. 387–392.

[66] Y. Pawitan. *In All Likelihood: Statistical Modelling and Inference Using Likelihood*. Oxford science publications. OUP Oxford, 2001.

[67] N. I. Gould, D. Orban, A. Sartenaer, and P. L. Toint. "Sensitivity of trust-region algorithms to their parameters". eng. In: *4OR* 3.3 (2005), pp. 227–241.

[68] N. I. Fisher. *Statistical Analysis of Circular Data*. Cambridge University Press, 1993.

[69] Y.-Y. Lin, P. Hodgkinson, M. Ernst, and A. Pines. "A Novel Detection–Estimation Scheme for Noisy NMR Signals: Applications to Delayed Acquisition Data". In: *J. Magn. Reson.* 128.1 (1997), pp. 30–41.

[70] M. Mayzel, K. Kazimierczuk, and V. Y. Orekhov. "The causality principle in the reconstruction of sparse NMR spectra". In: *Chem. Commun.* 50.64 (2014), pp. 8947–8950.

[71] D. Gołowicz, P. Kasprzak, and K. Kazimierczuk. "Enhancing Compression Level for More Efficient Compressed Sensing and Other Lessons from NMR Spectroscopy". In: *Sensors (Basel)* 20.5 (2020), p. 1325.

[72] J. Luo, Q. Zeng, K. Wu, and Y. Lin. "Fast reconstruction of non-uniform sampling multidimensional NMR spectroscopy via a deep neural network". In: *Journal of Magnetic Resonance* 317 (2020), p. 106772.

[73] Y. Hua. "High resolution imaging of continuously moving object using stepped frequency radar". In: *Signal Processing* 35.1 (1994), pp. 33–40.

[74] S. Fricke, J. Seymour, M. Battistel, D. Freedberg, C. Eads, and M. Augustine. "Data processing in NMR relaxometry using the matrix pencil". In: *Journal of Magnetic Resonance* 313 (2020), p. 106704.

[75] D. Wörtge, M. Parziale, J. Claussen, B. Mohebbi, S. Stapf, B. Blümich, and M. Augustine. "Quantitative stray-field T1 relaxometry with the matrix pencil method". In: *Journal of Magnetic Resonance* 351 (2023), p. 107435.

[76] T. D. Claridge. *High-Resolution NMR Techniques in Organic Chemistry: Third Edition*. eng. 2016.

[77] M. Goldman. "Formal Theory of Spin–Lattice Relaxation". In: *Journal of Magnetic Resonance* 149.2 (2001), pp. 160–187.

[78] I. Kuprov, N. Wagner-Rundell, and P. Hore. "Bloch-Redfield-Wangsness theory engine implementation using symbolic processing software". In: *Journal of Magnetic Resonance* 184.2 (2007), pp. 196–206.

[79] H. Y. Carr and E. M. Purcell. "Effects of Diffusion on Free Precession in Nuclear Magnetic Resonance Experiments". In: *Phys. Rev.* 94 (3 May 1954), pp. 630–638.

[80] S. Meiboom and D. Gill. "Modified Spin-Echo Method for Measuring Nuclear Relaxation Times". In: *Review of Scientific Instruments* 29.8 (Aug. 1958), pp. 688–691.

[81] C. Johnson. "Diffusion ordered nuclear magnetic resonance spectroscopy: principles and applications". In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 34.3 (1999), pp. 203–256.

[82] G. A. Morris. "Diffusion-Ordered Spectroscopy". In: *eMagRes*. John Wiley & Sons, Ltd, 2009.

[83] E. O. Stejskal and J. E. Tanner. "Spin Diffusion Measurements: Spin Echoes in the Presence of a Time-Dependent Field Gradient". In: *The Journal of Chemical Physics* 42.1 (1965), pp. 288–292.

[84] H. C. Torrey. "Bloch Equations with Diffusion Terms". In: *Phys. Rev.* 104 (3 Nov. 1956), pp. 563–565.

[85] J. E. Tanner. "Use of the Stimulated Echo in NMR Diffusion Studies". In: *The Journal of Chemical Physics* 52.5 (1970), pp. 2523–2526.

[86] R. Cotts, M. Hoch, T. Sun, and J. Markert. "Pulsed field gradient stimulated echo methods for improved NMR diffusion measurements in heterogeneous systems". In: *Journal of Magnetic Resonance (1969)* 83.2 (1989), pp. 252–266.

[87] D. Wu, A. Chen, and C. Johnson. "An Improved Diffusion-Ordered Spectroscopy Experiment Incorporating Bipolar-Gradient Pulses". In: *Journal of Magnetic Resonance, Series A* 115.2 (1995), pp. 260–264.

[88] M. D. Pelta, G. A. Morris, M. J. Stchedroff, and S. J. Hammond. "A one-shot sequence for high-resolution diffusion-ordered spectroscopy". In: *Magnetic Resonance in Chemistry* 40.13 (2002), S147–S152.

[89] D. Sinnaeve. "The Stejskal–Tanner equation generalized for any gradient shape—an overview of most pulse sequences measuring free diffusion". In: *Concepts in Magnetic Resonance Part A* 40A.2 (2012), pp. 39–65.

[90] A. Chen, C. S. Johnson, M. Lin, and M. J. Shapiro. "Chemical Exchange in Diffusion NMR Experiments". eng. In: *Journal of the American Chemical Society* 120.35 (1998), pp. 9094–9095.

[91] B. G. Davis and A. J. Fairbanks. *Carbohydrate chemistry*. eng. Oxford chemistry primers ; 99. Oxford: Oxford University Press, 2002.

[92] M. Foroozandeh. "Spin dynamics during chirped pulses: applications to homonuclear decoupling and broadband excitation". In: *Journal of Magnetic Resonance* 318 (2020), p. 106768.

[93] J.-M. Bohlen, M. Rey, and G. Bodenhausen. "Refocusing with chirped pulses for broadband excitation without phase dispersion". In: *Journal of Magnetic Resonance (1969)* 84.1 (1989), pp. 191–197.

[94] J. Bohlen and G. Bodenhausen. "Experimental Aspects of Chirp NMR Spectroscopy". In: *Journal of Magnetic Resonance, Series A* 102.3 (1993), pp. 293–301.

[95] K. E. Cano, M. A. Smith, and A. Shaka. "Adjustable, Broadband, Selective Excitation with Uniform Phase". In: *Journal of Magnetic Resonance* 155.1 (2002), pp. 131–139.

[96] J. E. Power, M. Foroozandeh, R. W. Adams, M. Nilsson, S. R. Coombes, A. R. Phillips, and G. A. Morris. "Increasing the quantitative bandwidth of NMR measurements". In: *Chem. Commun.* 52 (14 2016), pp. 2916–2919.

[97] M. Foroozandeh, M. Nilsson, and G. A. Morris. "Improved ultra-broadband chirp excitation". In: *Journal of Magnetic Resonance* 302 (2019), pp. 28–33.

[98] E. Kupce and R. Freeman. "Adiabatic Pulses for Wideband Inversion and Broadband Decoupling". In: *Journal of Magnetic Resonance, Series A* 115.2 (1995), pp. 273–276.

[99] H. Maeda and Y. Yanagisawa. "Future prospects for NMR magnets: A perspective". In: *Journal of Magnetic Resonance* 306 (2019), pp. 80–85.

[100] A. Shaka, J. Keeler, T. Frenkiel, and R. Freeman. "An improved sequence for broadband decoupling: WALTZ-16". In: *Journal of Magnetic Resonance (1969)* 52.2 (1983), pp. 335–338.

[101] A. Shaka, J. Keeler, and R. Freeman. "Evaluation of a new broadband decoupling sequence: WALTZ-16". In: *Journal of Magnetic Resonance (1969)* 53.2 (1983), pp. 313–340.

[102] A. Shaka, P. Barker, and R. Freeman. "Computer-optimized decoupling scheme for wideband applications and low-level operation". In: *Journal of Magnetic Resonance (1969)* 64.3 (1985), pp. 547–552.

[103] N. H. Meyer and K. Zangger. "Simplifying Proton NMR Spectra by Instant Homonuclear Broadband Decoupling". In: *Angewandte Chemie International Edition* 52.28 (2013), pp. 7143–7146.

[104] R. W. Adams. "Pure Shift NMR Spectroscopy". In: *eMagRes*. John Wiley & Sons, Ltd, 2014, pp. 295–310.

[105] K. Zangger. "Pure shift NMR". In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 86-87 (2015), pp. 1–20.

[106] A. Bax, R. Freeman, and G. A. Morris. "A simple method for suppressing dispersion-mode contributions in NMR spectra: The "pseudo echo"". In: *Journal of Magnetic Resonance (1969)* 43.2 (1981), pp. 333–338.

[107] E. Kupce, J. Boyd, and I. Campbell. "Short Selective Pulses for Biochemical Applications". In: *Journal of Magnetic Resonance, Series B* 106.3 (1995), pp. 300–303.

[108] K. Zangger and H. Sterk. "Homonuclear Broadband-Decoupled NMR Spectra". In: *Journal of Magnetic Resonance* 124.2 (1997), pp. 486–489.

[109] J. A. Aguilar, S. Faulkner, M. Nilsson, and G. A. Morris. "Pure Shift 1H NMR: A Resolution of the Resolution Problem?" In: *Angewandte Chemie International Edition* 49.23 (2010), pp. 3901–3903.

[110] A. J. Pell and J. Keeler. "Two-dimensional J-spectra with absorption-mode lineshapes". In: *Journal of Magnetic Resonance* 189.2 (2007), pp. 293–299.

[111] J. Garbow, D. Weitekamp, and A. Pines. "Bilinear rotation decoupling of homonuclear scalar interactions". In: *Chemical Physics Letters* 93.5 (1982), pp. 504–509.

[112] A. Bax. "Broadband homonuclear decoupling in heteronuclear shift correlation NMR spectroscopy". In: *Journal of Magnetic Resonance (1969)* 53.3 (1983), pp. 517–520.

[113] M. Foroozandeh, R. W. Adams, N. J. Meharry, D. Jeannerat, M. Nilsson, and G. A. Morris. "Ultrahigh-Resolution NMR Spectroscopy". In: *Angewandte Chemie International Edition* 53.27 (2014), pp. 6990–6992.

[114] M. Foroozandeh, G. A. Morris, and M. Nilsson. "PSYCHE Pure Shift NMR Spectroscopy". In: *Chemistry – A European Journal* 24.53 (2018), pp. 13988–14000.

[115] M. Foroozandeh, R. Adams, P. Kiraly, M. Nilsson, and G. Morris. "Measuring couplings in crowded NMR spectra: Pure shift NMR with multiplet analysis". eng. In: *Chemical communications (Cambridge, England)* 51.84 (2015), pp. 15410–15413.

[116] P. Kiraly, M. Foroozandeh, M. Nilsson, and G. A. Morris. "Anatomising proton NMR spectra with pure shift 2D J-spectroscopy: A cautionary tale". In: *Chemical Physics Letters* 683 (2017). Ahmed Zewail (1946-2016) Commemoration Issue of Chemical Physics Letters, pp. 398–403.

[117] J.-M. Nuzillard. "Time-Reversal of NMR Signals by Linear Prediction. Application to Phase-Sensitive HomonuclearJ-Resolved Spectroscopy". In: *Journal of Magnetic Resonance, Series A* 118.1 (1996), pp. 132–135.

[118] A. Martinez, F. Bourdreux, E. Riguet, and J.-M. Nuzillard. "High-resolution and high-sensitivity 2D homonuclear J-resolved NMR spectroscopy". In: *Magnetic Resonance in Chemistry* 50.1 (2012), pp. 28–32.

[119] P. Mutzenhardt, F. Guenneau, and D. Canet. "A Procedure for Obtaining Pure Absorption 2D J-Spectra: Application to Quantitative Fully J-Decoupled Homonuclear NMR Spectra". In: *Journal of Magnetic Resonance* 141.2 (1999), pp. 312–321.

[120] M. J. Frisch et al. *Gaussian03 Revision E.01*. Gaussian Inc. Wallingford CT. 2003.

[121] G. Zhu, D. Smith, and Y. Hua. "Post-Acquisition Solvent Suppression by Singular-Value Decomposition". In: *Journal of Magnetic Resonance* 124.1 (1997), pp. 286–289.

[122] M. J. Thrippleton, R. A. Edden, and J. Keeler. "Suppression of strong coupling artefacts in J-spectra". In: *Journal of Magnetic Resonance* 174.1 (2005), pp. 97–109.

[123] G. Wider, R. Baumann, K. Nagayama, R. R. Ernst, and K. Wüthrich. "Strong spin-spin coupling in the two-dimensional J-resolved 360-MHz 1H NMR spectra of the common amino acids". In: *Journal of Magnetic Resonance (1969)* 42.1 (1981), pp. 73–87.

[124] P. Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nat. Methods* 17 (2020), pp. 261–272.

[125] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, and E. Wieser. "Array programming with NumPy." In: *Nature* 585.7825 (2020), p. 357.

[126] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[127] N. Yilmazer, R. Fernandez-Recio, and T. K. Sarkar. "Matrix pencil method for simultaneously estimating azimuth and elevation angles of arrival along with the frequency of the incoming signals". In: *Digital Signal Processing* 16.6 (2006), pp. 796–816.

[128] H. Hogben, M. Krzystyniak, G. Charnock, P. Hore, and I. Kuprov. "Spinach – A software library for simulation of spin dynamics in large spin systems". eng. In: *Journal of magnetic resonance (1997)* 208.2 (2011), pp. 179–194.

[129] *Spinach Documentation*. https://spindynamics.org/wiki/index.php?title=Main_Page. Accessed 11-05-2023.

[130] S. Berger and S. Braun. *200 and more NMR experiments : a practical course*. eng. [3rd rev. and expanded ed.] Weinheim: Wiley-VCH, 2004.

[131] C. P. Butts, C. R. Jones, and J. N. Harvey. "High precision NOEs as a probe for low level conformers-a second conformation of strychnine". eng. In: 47.4 (2011), pp. 1193–1195.

[132] L. A. O'Dell. "The WURST kind of pulses in solid-state NMR". In: *Solid State Nuclear Magnetic Resonance* 55-56 (2013), pp. 28–41.

Additional Theory

A

A.1 Mathematical definitions

Descriptions of linear algebra and statics concepts that are referred to in this thesis are provided here. More detail can be found in numerous relevant texts[46, 66].

A.1.1 Linear algebra

TODO: rank, range

Singular value decomposition

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

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\dagger}. \tag{A.1}$$

The matrices that make up the decomposition are as follows

- $\boldsymbol{\Sigma} \in \mathbb{C}^{m \times n}$ is a rectangular diagonal matrix with diagonal elements comprising the *singular values* in descending order of magnitude. The singular values are the square roots of the non-zero eigenvalues of both $\boldsymbol{A}^{\dagger}\boldsymbol{A}$ and $\boldsymbol{A}\boldsymbol{A}^{\dagger}$.
- $\boldsymbol{U} \in \mathbb{C}^{m \times m}$ is a unitary matrix whose columns comprise the *left singular vectors*. The left singular vectors are the eigenvectors of the matrix $\boldsymbol{A}\boldsymbol{A}^{\dagger}$.
- $\boldsymbol{V} \in \mathbb{C}^{n \times n}$ is a unitary matrix whose columns comprise the *right singular vectors*. The right singular vectors are the eigenvectors of the matrix $\boldsymbol{A}^{\dagger}\boldsymbol{A}$.

One fundamental property of the SVD is that the number of non-zero singular values is equivalent to the rank of the matrix. As the EYM theorem highlights, the SVD is valuable in constructing

low-rank approximations of matrices, with applications in various fields such as signal processing, (see Section 2.2.1) and data compression.

Special matrices

A *Hankel matrix* is a matrix in which each ascending diagonal from left to right possesses identical elements. While Hankel matrices are often defined to be square, in this work such a restriction is not applied. Given a matrix $\boldsymbol{X} \in \mathbb{F}^{M\times N}$, the matrix is Hankel if

$$x_{m,n} = x_{m+1,n-1} \tag{A.2}$$

$\forall m \in \{1,\cdots,M-1\}\ \forall n \in \{2,\cdots,N\}$. Similarly, a *Toeplitz matrix* is a matrix in which every descending diagonal from left to right possesses identical elements, i.e.

$$x_{m,n} = x_{m+1,n+1} \tag{A.3}$$

$\forall m \in \{1,\cdots,M-1\}\ \forall n \in \{1,\cdots N-1,\}$.

A.1.2 Statistics and probability

TODO: Likelihood function, MLE

Probability density function

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 that a random sample obeying a known distribution lies withing the range $[x_a, x_b]$ is given by the integral

$$P(x_a \leq x \leq x_b) = \int_{x_a}^{x_b} p(x)\mathrm{d}x. \tag{A.4}$$

The integral of $p(x)$ over the entire sample space is defined to be unity. Also, the pdf of a specific value is always 0, since the width of the region of integration is 0.

Likelihood function

A.2 Multidimensional virtual echos

The VE concept (Section 2.5.1) can be generalised to any number of dimensions, assuming that a pair of amplitude-modulated signals exist for each indirect-dimension. Thus a set of 2^{D-1} signals is required for a D-dimensional FID. For the 2D case, this corresponds to the pair of signals

$\{\boldsymbol{Y}^{\cos}, \boldsymbol{Y}^{\sin}\}$, given by (1.21) with $D = 2$ and $\zeta = \{\cos(\cdot), \sin(\cdot)\}$, taking the forms (with noise neglected)

$$y^{\cos}_{n^{(1)},n^{(2)}} = \xi_{n^{(1)},n^{(2)}} c^{(1)}_{n^{(1)},n^{(2)}} \left(c^{(2)}_{n^{(1)},n^{(2)}} + \mathrm{i} s^{(2)}_{n^{(1)},n^{(2)}} \right), \tag{A.5a}$$

$$y^{\sin}_{n^{(1)},n^{(2)}} = \xi_{n^{(1)},n^{(2)}} s^{(1)}_{n^{(1)},n^{(2)}} \left(c^{(2)}_{n^{(1)},n^{(2)}} + \mathrm{i} s^{(2)}_{n^{(1)},n^{(2)}} \right), \tag{A.5b}$$

$$\xi_{n^{(1)},n^{(2)}} = \sum_m a_m \exp\left(-\eta^{(1)}_m n^{(1)} \Delta^{(1)}_t - \eta^{(2)}_m n^{(2)} \Delta^{(2)}_t \right), \tag{A.5c}$$

$$(c/s)^{(1/2)}_{n^{(1)},n^{(2)}} = \sum_m \cos / \sin\left(2\pi f^{(1/2)} n^{(1/2)} \Delta^{(1/2)} \right). \tag{A.5d}$$

Four matrices $\boldsymbol{\psi}_{\pm\pm}$ are then constructed of the form

$$\begin{aligned} \psi_{\pm\pm,n^{(1)},n^{(2)}} &= \xi_{n^{(1)},n^{(2)}} \left(c^{(1)}_{n^{(1)},n^{(2)}} \pm^{(1)} \mathrm{i} s^{(1)}_{n^{(1)},n^{(2)}} \right) \left(c^{(2)}_{n^{(1)},n^{(2)}} \pm^{(2)} \mathrm{i} s^{(2)}_{n^{(1)},n^{(2)}} \right) \\ &\equiv \Re\left(y^{\cos}_{n^{(1)},n^{(2)}} \right) \pm^{(1)} \pm^{(2)} - \Im\left(y^{\sin}_{n^{(1)},n^{(2)}} \right) + \mathrm{i} \left(\pm^{(1)} \Re\left(y^{\sin}_{n^{(1)},n^{(2)}} \right) \pm^{(2)} \Im\left(y^{\cos}_{n^{(1)},n^{(2)}} \right) \right), \end{aligned} \tag{A.6}$$

from which the matrices $\boldsymbol{T}_{1\to 4} \in \mathbb{C}^{2N^{(1)} \times 2N^{(2)}}$ are generated:

$$\boldsymbol{T}_1 = \begin{bmatrix} \boldsymbol{\Psi}_{++} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \tag{A.7a}$$

$$\boldsymbol{T}_2 = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{\Psi}^{\leftrightarrow(1)}_{-+} & \boldsymbol{0} \end{bmatrix}^{\circlearrowleft(1)}, \tag{A.7b}$$

$$\boldsymbol{T}_3 = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{\Psi}^{\leftrightarrow(2)}_{+-} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}^{\circlearrowleft(2)}, \tag{A.7c}$$

$$\boldsymbol{T}_4 = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Psi}^{\leftrightarrow(1,2)}_{--} \end{bmatrix}^{\circlearrowleft(1,2)}. \tag{A.7d}$$

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.4.

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 (1.21) with $D = 2$ and $\zeta = \{\exp(\mathrm{i}\cdot), \exp(-\mathrm{i}\cdot)\}$. These can be used to generate an amplitude modulated pair via

$$\boldsymbol{Y}^{\cos} = \frac{\boldsymbol{Y}^{\mathrm{pos}} + \boldsymbol{Y}^{\mathrm{neg}}}{2}, \tag{A.8a}$$

$$\boldsymbol{Y}^{\sin} = \frac{\boldsymbol{Y}^{\mathrm{pos}} - \boldsymbol{Y}^{\mathrm{neg}}}{2\mathrm{i}}. \tag{A.8b}$$

A.3 Additional algorithms

Algorithm A.1 The MMEMPM.

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**

$$\boldsymbol{H}_{\boldsymbol{Y},n^{(1)}} \leftarrow \begin{bmatrix} \boldsymbol{Y}\left[n^{(1)}, 0\right] & \boldsymbol{Y}\left[n^{(1)}, 1\right] & \cdots & \boldsymbol{Y}\left[n^{(1)}, N^{(2)} - L^{(2)}\right] \\ \boldsymbol{Y}\left[n^{(1)}, 1\right] & \boldsymbol{Y}\left[n^{(1)}, 2\right] & \cdots & \boldsymbol{Y}\left[n^{(1)}, N^{(2)} - L^{(2)} + 1\right] \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{Y}\left[n^{(1)}, L^{(2)} - 1\right] & \boldsymbol{Y}\left[n^{(1)}, L^{(2)}\right] & \cdots & \boldsymbol{Y}\left[n^{(1)}, N^{(2)} - 1\right] \end{bmatrix}$$

end for;

$$\boldsymbol{E}_{\boldsymbol{Y}} \leftarrow \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{Y},0} & \boldsymbol{H}_{\boldsymbol{Y},1} & \cdots & \boldsymbol{H}_{\boldsymbol{Y},N^{(1)}-L^{(1)}} \\ \boldsymbol{H}_{\boldsymbol{Y},1} & \boldsymbol{H}_{\boldsymbol{Y},2} & \cdots & \boldsymbol{H}_{\boldsymbol{Y},N^{(1)}-L^{(1)}+1} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{H}_{\boldsymbol{Y},L^{(1)}-1} & \boldsymbol{H}_{\boldsymbol{Y},L^{(1)}} & \cdots & \boldsymbol{H}_{\boldsymbol{Y},N^{(1)}-1} \end{bmatrix};$$

$\boldsymbol{U}_M, \boldsymbol{\Sigma}_M, \boldsymbol{V}_M^{\dagger} \leftarrow \textsc{TruncatedSVD}\left(\boldsymbol{E}_{\boldsymbol{Y}}, M\right)$;

$\boldsymbol{P} \leftarrow \boldsymbol{0} \in \mathbb{C}^{L^{(1)}L^{(2)} \times L^{(1)}L^{(2)}}$;

$r \leftarrow 0$

for $i = 0, \cdots, L^{(2)} - 1$ **do**

  for $j = 0, \cdots, L^{(1)} - 1$ **do**

    $c \leftarrow i + jL^{(2)}$;

    $\boldsymbol{P}\left[r, c\right] \leftarrow 1$;

    $r = r + 1$;

  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

$\boldsymbol{z}^{(1)}, \boldsymbol{W}^{(1)} \leftarrow \textsc{Eigendecomposition}\left(\boldsymbol{U}_{M1}^{+}\boldsymbol{U}_{M2}\right)$;

$\boldsymbol{f}^{(1)}, \boldsymbol{\eta}^{(1)} \leftarrow \left(f_{\text{sw}}^{(1)}/2\pi\right)\Im\left(\ln \boldsymbol{z}^{(1)}\right) + f_{\text{off}}^{(1)}, -f_{\text{sw}}^{(1)}\Re\left(\ln \boldsymbol{z}^{(1)}\right)$;

$\boldsymbol{U}_{M\text{P}} \leftarrow \boldsymbol{P}\boldsymbol{U}_M$;

$\boldsymbol{U}_{M\text{P}1}, \boldsymbol{U}_{M\text{P}2} \leftarrow \boldsymbol{U}_{M\text{P}}\left[: (L^{(1)} - 1)L^{(2)}\right], \boldsymbol{U}_{M\text{P}}\left[L^{(1)} :\right]$; ▷ Last/First $L^{(1)}$ rows deleted

$\boldsymbol{z}^{(2)} \leftarrow \text{diag}\left(\left[\boldsymbol{W}^{(1)}\right]^{-1}\boldsymbol{U}_{M\text{P}1}^{+}\boldsymbol{U}_{M\text{P}2}\boldsymbol{W}^{(1)}\right)$;

$\boldsymbol{f}^{(2)}, \boldsymbol{\eta}^{(2)} \leftarrow \left(f_{\text{sw}}^{(2)}/2\pi\right)\Im\left(\ln \boldsymbol{z}^{(2)}\right) + f_{\text{off}}^{(2)}, -f_{\text{sw}}^{(2)}\Re\left(\ln \boldsymbol{z}^{(2)}\right)$;

$\boldsymbol{Z}_{\text{L}}^{(2)} = \begin{bmatrix} \boldsymbol{1} & \boldsymbol{z}^{(2)} & \boldsymbol{z}^{(2)^2} & \cdots & \boldsymbol{z}^{(2)^{L^{(2)}-1}} \end{bmatrix}^{\text{T}}$;

$\boldsymbol{Z}_{\text{R}}^{(2)} \leftarrow \begin{bmatrix} \boldsymbol{1} & \boldsymbol{z}^{(2)} & \boldsymbol{z}^{(2)^2} & \cdots & \boldsymbol{z}^{(2)^{N^{(2)}-L^{(2)}}} \end{bmatrix}$;

$\boldsymbol{Z}_{\text{D}}^{(1)} \leftarrow \text{diag}\left(\boldsymbol{z}^{(1)}\right)$;

$$\boldsymbol{E}_{\text{L}} \leftarrow \begin{bmatrix} \boldsymbol{Z}_{\text{L}}^{(2)} \\ \boldsymbol{Z}_{\text{L}}^{(2)}\boldsymbol{Z}_{\text{D}}^{(1)} \\ \vdots \\ \boldsymbol{Z}_{\text{L}}^{(2)}\left[\boldsymbol{Z}_{\text{D}}^{(1)}\right]^{L^{(1)}-1} \end{bmatrix};$$

$\boldsymbol{E}_{\text{R}} \leftarrow \begin{bmatrix} \boldsymbol{Z}_{\text{R}}^{(2)} & \boldsymbol{Z}_{\text{D}}^{(1)}\boldsymbol{Z}_{\text{R}}^{(2)} & \cdots & \left[\boldsymbol{Z}_{\text{D}}^{(1)}\right]^{N^{(1)}-L^{(1)}}\boldsymbol{Z}_{\text{R}}^{(2)} \end{bmatrix}$;

$\boldsymbol{\alpha} \leftarrow \text{diag}\left(\boldsymbol{E}_{\text{L}}^{+}\boldsymbol{E}_{\boldsymbol{Y}}\boldsymbol{E}_{\text{R}}^{+}\right)$;

$\boldsymbol{a}, \boldsymbol{\phi} \leftarrow |\boldsymbol{\alpha}|, \arctan\left(\frac{\Im(\boldsymbol{\alpha})}{\Re(\boldsymbol{\alpha})}\right)$;

$\boldsymbol{\theta}^{(0)} \leftarrow \begin{bmatrix} \boldsymbol{a}^{\text{T}} & \boldsymbol{\phi}^{\text{T}} & \left[\boldsymbol{f}^{(1)}\right]^{\text{T}} & \left[\boldsymbol{f}^{(2)}\right]^{\text{T}} & \left[\boldsymbol{\eta}^{(1)}\right]^{\text{T}} & \left[\boldsymbol{\eta}^{(2)}\right]^{\text{T}} \end{bmatrix}^{\text{T}}$;

return $\boldsymbol{\theta}^{(0)}$

end procedure

Algorithm A.2 Steihaug-Toint method for determining an update for nonlinear programming. This is equivalent to Algorithm 7.2 in [55].

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: (2.45a)
  $\boldsymbol{H} \leftarrow \nabla^2 \mathcal{F}_{\phi}\left(\boldsymbol{\theta}^{(k)} \mid \boldsymbol{Y}\right)$; ▷ Hessian matrix, either exact: (2.45b) or approximate: (2.47)
  $\epsilon^{(k)} \leftarrow \min\left(1/2, \sqrt{\|\boldsymbol{g}\|}\right) \|\boldsymbol{g}\|$;
  $\boldsymbol{z}^{(0)} \leftarrow \boldsymbol{0} \in \mathbb{R}^{6M}$;
  $\boldsymbol{r}^{(0)} \leftarrow \boldsymbol{g}$;
  $\boldsymbol{d}^{(0)} \leftarrow -\boldsymbol{r}^{(0)}$;
  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 Filtering procedure for 1D data. $\boldsymbol{r}_{\text{interest}}$ is a vector of length 2 containing the indices of the left and right bounds of the region of interest. These would typically be provided in units of Hz or ppm by a user. Conversion to array indices can be carried out using (2.69). $\boldsymbol{r}_{\text{noise}}$ contains the left and right bounds of the region used to estimate the noise variance. RandomSample indicates taking a random sample from the given distribution.

procedure Filter1D($\boldsymbol{y} \in \mathbb{C}^{N^{(1)}}, \boldsymbol{r}_{\text{interest}} \in \mathbb{N}_0^2, \boldsymbol{r}_{\text{noise}} \in \mathbb{N}_0^2$)
  $\boldsymbol{y}_{\text{ve}} \leftarrow$ VirtualEcho1D $(\boldsymbol{y})$;
  $\boldsymbol{s}_{\text{ve}} \leftarrow \text{FT}\left(\boldsymbol{y}_{\text{ve}}\right)$;
  $l_{\text{idx}}^{(1)}, r_{\text{idx}}^{(1)} \leftarrow \boldsymbol{r}_{\text{interest}}[0], \boldsymbol{r}_{\text{interest}}[1]$;
  $l_{\text{idx,noise}}^{(1)}, r_{\text{idx,noise}}^{(1)} \leftarrow \boldsymbol{r}_{\text{noise}}[0], \boldsymbol{r}_{\text{noise}}[1]$;
  $c_{\text{idx}}^{(1)} \leftarrow \left(l_{\text{idx}}^{(1)} + r_{\text{idx}}^{(1)}\right)/2$;
  $b_{\text{idx}}^{(1)} \leftarrow r_{\text{idx}}^{(1)} - l_{\text{idx}}^{(1)}$;
  $\boldsymbol{g} \leftarrow$ SuperGaussian1D $\left(N^{(1)}, c_{\text{idx}}^{(1)}, b_{\text{idx}}^{(1)}\right)$;
  $\boldsymbol{s}_{\text{noise}} \leftarrow \boldsymbol{s}_{\text{ve}}\left[l_{\text{idx,noise}}^{(1)} : r_{\text{idx,noise}}^{(1)} + 1\right]$;
  $\sigma^2 \leftarrow \text{Var}\left(\boldsymbol{s}_{\text{noise}}\right)$;
  $\boldsymbol{w}_{\sigma^2} \leftarrow \boldsymbol{0} \in \mathbb{R}^{2N^{(1)}}$;
  for $n^{(1)} = 0, \cdots, N^{(1)} - 1$ **do**
    $\boldsymbol{w}_{\sigma^2}\left[n^{(1)}\right] \leftarrow$ RandomSample $\left(\mathcal{N}\left(0, \sigma^2\right)\right)$;
  end for
  $\widetilde{\boldsymbol{s}}_{\text{ve}} \leftarrow \boldsymbol{s}_{\text{ve}} \odot \boldsymbol{g} + \boldsymbol{w}_{\sigma^2} \odot (\boldsymbol{1} - \boldsymbol{g})$;
  $\widetilde{\boldsymbol{y}}_{\text{ve}} \leftarrow \text{IFT}\left(\widetilde{\boldsymbol{s}}_{\text{ve}}\right)$;
  $\widetilde{\boldsymbol{y}} \leftarrow \widetilde{\boldsymbol{y}}_{\text{ve}}\left[: N^{(1)}\right]$;
  return $\widetilde{\boldsymbol{y}}$;
end procedure

procedure VirtualEcho1D($\boldsymbol{y} \in \mathbb{C}^{N^{(1)}}$)
  $\boldsymbol{t}_1 \leftarrow \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{0} \in \mathbb{C}^{N^{(1)}} \end{bmatrix}$;
  $\boldsymbol{t}_2 \leftarrow \begin{bmatrix} \boldsymbol{0} \in \mathbb{C}^{N^{(1)}} \\ \boldsymbol{y}^{*\leftrightarrow(1)} \end{bmatrix}^{\circlearrowleft(1)}$;
  $\boldsymbol{y}_{\text{ve}} \leftarrow \boldsymbol{t}_1 + \boldsymbol{t}_2$;
  $\boldsymbol{y}_{\text{ve}}[0] \leftarrow \boldsymbol{y}_{\text{ve}}[0]/2$;
  return $\boldsymbol{y}_{\text{ve}}$;
end procedure

procedure SuperGaussian1D($N \in \mathbb{N}, c_{\text{idx}} \in \mathbb{R}_{>0}, b_{\text{idx}} \in \mathbb{N}$)
  $\boldsymbol{g} \leftarrow \boldsymbol{0} \in \mathbb{R}^N$;
  for $n = 0, \cdots, N - 1$ **do**
    $\boldsymbol{g}[n] \leftarrow \exp\left(-2^{41}\left(\frac{n - c_{\text{idx}}}{b_{\text{idx}}}\right)^{40}\right)$; ▷ p in (2.65) has been set to 40.
  end for
  return $\boldsymbol{g}$
end procedure

Algorithm A.4 Filtering procedure for 2D data.

procedure FILTER2D($\boldsymbol{Y}_{\cos} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}, \boldsymbol{Y}_{\sin} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}, \boldsymbol{R}_{\text{interest}} \in \mathbb{N}_0^{2\times 2}, \boldsymbol{R}_{\text{noise}} \in \mathbb{N}_0^{2\times 2}$)
  $\boldsymbol{Y}_{\text{ve}} \leftarrow$ VIRTUALECHO2D $(\boldsymbol{Y}_{\cos}, \boldsymbol{Y}_{\sin})$;
  $\boldsymbol{S}_{\text{ve}} \leftarrow \text{FT}\left(\boldsymbol{Y}_{\text{ve}}\right)$;
  $l_{\text{idx}}^{(1)}, r_{\text{idx}}^{(1)}, l_{\text{idx}}^{(2)}, r_{\text{idx}}^{(2)} \leftarrow \boldsymbol{R}_{\text{interest}}[0,0], \boldsymbol{R}_{\text{interest}}[0,1], \boldsymbol{R}_{\text{interest}}[1,0], \boldsymbol{R}_{\text{interest}}[1,1]$;
  for $d = 1, 2$ **do**
    $c_{\text{idx}}^{(d)} \leftarrow \left(l_{\text{idx}}^{(d)} + r_{\text{idx}}^{(d)}\right)/2$;
    $b_{\text{idx}}^{(d)} \leftarrow r_{\text{idx}}^{(d)} - l_{\text{idx}}^{(d)}$;
    $\boldsymbol{g}^{(d)} \leftarrow$ SUPERGAUSSIAN1D $\left(2N^{(d)}, c_{\text{idx}}^{(d)}, b_{\text{idx}}^{(d)}\right)$;
    $\boldsymbol{G} \leftarrow \boldsymbol{g}^{(1)} \otimes \boldsymbol{g}^{(2)}$;
  end for
  $l_{\text{idx,noise}}^{(1)}, r_{\text{idx,noise}}^{(1)}, l_{\text{idx,noise}}^{(2)}, r_{\text{idx,noise}}^{(2)} \leftarrow \boldsymbol{R}_{\text{noise}}[0,0], \boldsymbol{R}_{\text{noise}}[0,1], \boldsymbol{R}_{\text{noise}}[1,0], \boldsymbol{R}_{\text{noise}}[1,1]$;
  $\boldsymbol{S}_{\text{noise}} \leftarrow \boldsymbol{S}_{\text{ve}}\left[l_{\text{idx,noise}}^{(1)} : r_{\text{idx,noise}}^{(1)} + 1, l_{\text{idx,noise}}^{(2)} : r_{\text{idx,noise}}^{(2)} + 1\right]$
  $\sigma^2 \leftarrow \text{Var}\left(\boldsymbol{S}_{\text{noise}}\right)$;
  $\boldsymbol{W}_{\sigma^2} \leftarrow \boldsymbol{0} \in \mathbb{R}^{2N^{(1)} \times 2N^{(2)}}$;
  for $n^{(1)} = 0, \cdots, 2N^{(1)} - 1$ **do**
    for $n^{(2)} = 0, \cdots, 2N^{(2)} - 1$ **do**
      $\boldsymbol{W}_{\sigma^2}\left[n^{(1)}, n^{(2)}\right] \leftarrow$ RANDOMSAMPLE $\left(\mathcal{N}\left(0, \sigma^2\right)\right)$;
    end for
  end for
  $\widetilde{\boldsymbol{S}}_{\text{ve}} \leftarrow \boldsymbol{S}_{\text{ve}} \odot \boldsymbol{G} + \boldsymbol{W}_{\sigma^2} \odot (\boldsymbol{1} - \boldsymbol{G})$;
  $\widetilde{\boldsymbol{Y}}_{\text{ve}} \leftarrow \text{IFT}\left(\widetilde{\boldsymbol{S}}_{\text{ve}}\right)$;
  $\widetilde{\boldsymbol{Y}} \leftarrow \widetilde{\boldsymbol{Y}}_{\text{ve}}\left[: N^{(1)}, : N^{(2)}\right]$;
  return $\widetilde{\boldsymbol{Y}}$;
end procedure

procedure VIRTUALECHO2D($\boldsymbol{Y}_{\cos} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}, \boldsymbol{Y}_{\sin} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}$)
  $\boldsymbol{\Psi}_{++} \leftarrow \Re\left(\boldsymbol{Y}_{\cos}\right) - \Im\left(\boldsymbol{Y}_{\sin}\right) + \text{i}\left(\Im\left(\boldsymbol{Y}_{\cos}\right) + \Re\left(\boldsymbol{Y}_{\sin}\right)\right)$;
  $\boldsymbol{\Psi}_{+-} \leftarrow \Re\left(\boldsymbol{Y}_{\cos}\right) + \Im\left(\boldsymbol{Y}_{\sin}\right) + \text{i}\left(\Re\left(\boldsymbol{Y}_{\sin}\right) - \Im\left(\boldsymbol{Y}_{\cos}\right)\right)$;
  $\boldsymbol{\Psi}_{-+} \leftarrow \Re\left(\boldsymbol{Y}_{\cos}\right) + \Im\left(\boldsymbol{Y}_{\sin}\right) + \text{i}\left(\Im\left(\boldsymbol{Y}_{\cos}\right) - \Re\left(\boldsymbol{Y}_{\sin}\right)\right)$;
  $\boldsymbol{\Psi}_{--} \leftarrow \Re\left(\boldsymbol{Y}_{\cos}\right) - \Im\left(\boldsymbol{Y}_{\sin}\right) - \text{i}\left(\Im\left(\boldsymbol{Y}_{\cos}\right) + \Re\left(\boldsymbol{Y}_{\sin}\right)\right)$;
  $\boldsymbol{Z} \leftarrow \boldsymbol{0} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}$
  $\boldsymbol{T}_1 \leftarrow \begin{bmatrix} \boldsymbol{\Psi}_{++} & \boldsymbol{Z} \\ \boldsymbol{Z} & \boldsymbol{Z} \end{bmatrix}$;
  $\boldsymbol{T}_2 \leftarrow \begin{bmatrix} \boldsymbol{Z} & \boldsymbol{\Psi}_{+-}^{\leftrightarrow(2)} \\ \boldsymbol{Z} & \boldsymbol{Z} \end{bmatrix}^{\circlearrowleft(2)}$;
  $\boldsymbol{T}_3 \leftarrow \begin{bmatrix} \boldsymbol{Z} & \boldsymbol{Z} \\ \boldsymbol{\Psi}_{-+}^{\leftrightarrow(1)} & \boldsymbol{Z} \end{bmatrix}^{\circlearrowleft(1)}$;
  $\boldsymbol{T}_4 \leftarrow \begin{bmatrix} \boldsymbol{Z} & \boldsymbol{Z} \\ \boldsymbol{Z} & \boldsymbol{\Psi}_{--}^{\leftrightarrow(1,2)} \end{bmatrix}^{\circlearrowleft(1,2)}$;
  $\boldsymbol{Y}_{\text{ve}} \leftarrow \boldsymbol{T}_1 + \boldsymbol{T}_2 + \boldsymbol{T}_3 + \boldsymbol{T}_4$;
  for $n^{(1)} = 0, \cdots, 2N^{(1)} - 1$ **do**
    $\boldsymbol{Y}_{\text{ve}}\left[n^{(1)}, 0\right] \leftarrow \boldsymbol{Y}_{\text{ve}}\left[n^{(1)}, 0\right]/2$;
  end for
  for $n^{(2)} = 0, \cdots, 2N^{(2)} - 1$ **do**
    $\boldsymbol{Y}_{\text{ve}}\left[0, n^{(2)}\right] \leftarrow \boldsymbol{Y}_{\text{ve}}\left[0, n^{(2)}\right]/2$;
  end for
  return $\boldsymbol{Y}_{\text{ve}}$;
end procedure

Algorithm A.5 Filtering procedure for 2DJ data.

procedure Filter2DJ($\boldsymbol{Y} \in \mathbb{C}^{N^{(1)} \times N^{(2)}}, \boldsymbol{r}_{\text{interest}} \in \mathbb{N}_0^2, \boldsymbol{r}_{\text{noise}} \in \mathbb{N}_0^2$)
  $\boldsymbol{Y}_{\text{ve}} \leftarrow \boldsymbol{0} \in \mathbb{C}^{N^{(1)} \times 2N^{(2)}}$;
  for $n^{(1)} = 0, \cdots, N^{(1)} - 1$ **do**
    $\boldsymbol{Y}_{\text{ve}}\left[n^{(1)}, :\right] \leftarrow$ VirtualEcho1D $\left(\boldsymbol{Y}\left[n^{(1)}, :\right]\right)$;
  end for
  $\boldsymbol{S}_{\text{ve}} \leftarrow \text{FT}^{(2)}\left(\boldsymbol{Y}_{\text{ve}}\right)$;
  $l_{\text{idx}}^{(2)}, r_{\text{idx}}^{(2)} \leftarrow \boldsymbol{r}_{\text{interest}}[0], \boldsymbol{r}_{\text{interest}}[1]$;
  $l_{\text{idx,noise}}^{(2)}, r_{\text{idx,noise}}^{(2)} \leftarrow \boldsymbol{r}_{\text{noise}}[0], \boldsymbol{r}_{\text{noise}}[1]$;
  $c_{\text{idx}}^{(2)} \leftarrow \left(l_{\text{idx}}^{(2)} + r_{\text{idx}}^{(2)}\right)/2$;
  $b_{\text{idx}}^{(2)} \leftarrow r_{\text{idx}}^{(2)} - l_{\text{idx}}^{(2)}$;
  $\boldsymbol{g}^{(1)} \leftarrow \boldsymbol{1} \in \mathbb{R}^{N^{(1)}}$;
  $\boldsymbol{g}^{(2)} \leftarrow$ SuperGaussian1D $\left(N^{(2)}, c_{\text{idx}}^{(2)}, b_{\text{idx}}^{(2)}\right)$;
  $\boldsymbol{G} \leftarrow \boldsymbol{g}^{(1)} \otimes \boldsymbol{g}^{(2)}$;
  $\boldsymbol{S}_{\text{noise}} \leftarrow \boldsymbol{S}_{\text{ve}}\left[:, l_{\text{idx,noise}}^{(2)} : r_{\text{idx,noise}}^{(2)} + 1\right]$;
  $\sigma^2 \leftarrow \text{Var}\left(\boldsymbol{S}_{\text{noise}}\right)$;
  $\boldsymbol{W}_{\sigma^2} \leftarrow \boldsymbol{0} \in \mathbb{R}^{N^{(1)} \times 2N^{(2)}}$;
  for $n^{(1)} = 0, \cdots, N^{(1)} - 1$ **do**
    for $n^{(2)} = 0, \cdots, 2N^{(2)} - 1$ **do**
      $\boldsymbol{W}_{\sigma^2}\left[n^{(1)}, n^{(2)}\right] \leftarrow$ RandomSample $\left(\mathcal{N}\left(0, \sigma^2\right)\right)$;
    end for
  end for
  $\widetilde{\boldsymbol{S}}_{\text{ve}} \leftarrow \boldsymbol{S}_{\text{ve}} \odot \boldsymbol{G} + \boldsymbol{W}_{\sigma^2} \odot (\boldsymbol{1} - \boldsymbol{G})$;
  $\widetilde{\boldsymbol{Y}}_{\text{ve}} \leftarrow \text{IFT}^{(2)}\left(\widetilde{\boldsymbol{S}}_{\text{ve}}\right)$;
  $\widetilde{\boldsymbol{Y}} \leftarrow \widetilde{\boldsymbol{Y}}_{\text{ve}}\left[:, : N^{(2)}\right]$;
  return $\widetilde{\boldsymbol{Y}}$;
end procedure

Algorithm A.6 An algorithm for multiplet assignment of a 2DJ estimation result.

procedure MultipletAssign($\boldsymbol{\theta} \in \mathbb{R}^{6M}, \epsilon \in \mathbb{R}_{>0}$)
  $\boldsymbol{f}^{(1)}, \boldsymbol{f}^{(2)} \leftarrow \boldsymbol{\theta}[2M : 3M], \boldsymbol{\theta}[3M : 4M]$;
  Map $\leftarrow$ HashMap $[\mathbb{R}, \text{Vector}[\mathbb{N}_0]]$;
  for $m = \{0, \cdots, M - 1\}$ **do**
    $f_{\text{c}} = \boldsymbol{f}^{(2)}[m] - \boldsymbol{f}^{(1)}[m]$;
    Assigned $\leftarrow$ False;
    for $f_{\text{mp}}, \boldsymbol{i}$ **in** Map **do**
      if $\left|f_{\text{c}} - f_{\text{mp}}\right| < \epsilon$ **then**
        $\boldsymbol{i} \leftarrow \left[\boldsymbol{i}^{\text{T}} \;\; m\right]^{\text{T}}$;
        Assigned $\leftarrow$ True;
        break;
      end if
    end for
    if Assigned = False **then**
      Map.insert $\left(f_{\text{c}}, [m]\right)$;
    end if
  end for
  return Map;
end procedure

Code Listings

B

NEED TO ADD TEXT TO DESCRIBE THINGS The following is all the required imports:

Code Listing B.1: Required imports for the subsequent listings in this chapter. The `product` function, is employed in the MMEMPM routine (Listing B.4). The imports from the `typing` module are used to annotate what the expected types are for each argument, and what the return type is. The `numpy` and `scipy` modules are ubiquitous in the following listings, providing access to efficient routines for numerical computations.

```
1 from itertools import product
2 from typing import Any, Iterable, Optional, Tuple, Union
3 import numpy as np
4 import scipy as sp
```

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 22), since the presence of very small singular values when `k` is large can lead to errors involving floating-point arithmetic.

```
1 def mdl(sigma: np.ndarray, N: int, L: int) -> int:
2     """Compute the Minimum Description Length
3 
4     Parameters
5     ----------
6     sigma
7         Vector of singular values associated with the data matrix.
8 
```

```
    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 estimation of a 1D FID, with the option of estimating the model order using the MDL.

```
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
        Sweep width (Hz).

    offset
        Transmitter offset (Hz).

    M
        Number of oscillators. If 0, this is estimated using the MDL.
    """
    N = Y  # N^(1)
    L = N // 3  # L^(1): pencil parameter
    norm = np.linalg.norm(Y)  # ||Y||
    Y /= norm  # Y/||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)  # H_Y
    _, S, Vh = np.linalg.svd(HY)  # σ and V^†
    V = Vh.T  # V

    if M == 0:
        M = mdl(sigma, N, L)

    VM = V[:, :M]  # V_M
    VM1 = Vm[:-1, :]  # V_M1
    VM2 = Vm[1:, :]  # V_M2
    VM1inv = np.linalg.pinv(VM1)  # V_M1^+
    VM1invVM2 = VM1inv @ VM2  # V_M1^+ V_M2
    z, _ = np.linalg.eig(VM1invVM2)  # z^(1): signal poles
    Z = np.power.outer(z, np.arange(N)).T  # Z^(1)
    alpha = np.linalg.pinv(Z) @ Y  # α: complex amplitudes

    # Extract amplitude, phase, frequency and damping factor
    amp = np.abs(alpha) * norm  # a
    phase = np.arctan2(np.imag(alpha), np.real(alpha))  # ϕ
    freq = (sw / (2 * np.pi)) * np.imag(np.log(z)) + offset  # f^(1)
    damp = -sw * np.real(np.log(z))  # η^(1)
    theta = np.vstack((amp, phase, freq, damp)).T  # θ, as a M×4 array

    # Remove negative damping factors
    neg_damp_idx = np.nonzero(damp < 0.)[0]
    theta = np.delete(theta, neg_damp_idx, axis=0)

    theta = theta[np.argsort(theta[:, 2])]  # order by f^(1)
    return theta
```

B.1.3 MMEMPM

Code Listing B.4: The MMEMPM for estimation of a 2D hypercomplex FID. Due to the very large size of the Hankel matrix $\boldsymbol{E}_Y$, a truncated SVD routine is employed, which determines only the first M components of the decomposition. This is only available to arrays stored in sparse form in SciPy (lines 65–66).

```
def mmempm(
    Y: np.ndarray, sw1: float, sw2: float,
    offset1: float, offset2: float, M: int = 0,
) -> np.ndarray:
    """Modified Matrix Enhancment Matrix Pencil Method for estimating a
   2D FID.

```

```
    Parameters
    ----------
    Y
        FID.

    sw1
        Sweep width in first (indirect) dimension (Hz).

    sw2
        Sweep width in second (direct) dimension (Hz).

    offset1
        Transmitter offset in first dimension (Hz).

    offset2
        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||

    if M == 0:
        # Extract first direct dimension signal, and estimate model order
        # with the MDL
        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):
```

```
53          # Construct H_{Y,n^(1)}, and assign to appropriate positions in E_Y
54          col = Y[n1, :L2]
55          row = Y[n1, L2 - 1:]
56          HYn1 = sp.linalg.hankel(col, row)
57          for n in range(n1 + 1):
58              r, c = n, n1 - n
59              if r < L1 and c < N1 - L1 + 1:
60                  EY[
61                      r * row_size : (r + 1) * row_size,
62                      c * col_size : (c + 1) * col_size
63                  ] = HYn1
64
65      EY = sp.sparse.csr_matrix(EY)  # Make E_Y sparse
66      UM, *_ = sp.sparse.linalg.svds(EY, k=M)  # U_M
67
68      # === Construct permutation matrix P ===
69      P = np.zeros((L1 * L2, L1 * L2))
70      r = 0
71      for l2 in range(L2):
72          for l1 in range(L1):
73              c = l1 * L2 + l2
74              P[r, c] = 1
75              r += 1
76
77      UM1 = UM[: L1 * (L2 - 1)]  # Last L^(2) rows deleted: U_M1
78      UM2 = UM[L2:]  # First L^(2) rows deleted: U_M2
79      z1, W1 = np.linalg.eig(np.linalg.pinv(UM1) @ UM2)  # z^(1), W^(1)
80
81      UMP = P @ UM  # U_MP
82      UMP1 = UMP[: (L1 - 1) * L2]  # Last L^(1) rows deleted: U_MP1
83      UMP2 = UMP[L1:]  # First L^(1) rows deleted: U_MP2
84      Z2 = np.linalg.inv(W1) @ np.linalg.pinv(UMP1) @ UMP2 @ W1  # Z^(2)
85      z2 = np.diag(Z2).copy()  # z^(2): copy needed as slice is readonly
86
87      # === Check for and deal with similar frequencies ===
88      freq1 = (0.5 * sw1 / np.pi) * np.imag(np.log(z1)) + offset1  # f^(1)
89      threshold = sw1 / N1  # f_sw^(1) / N^(1)
90      groupings = {}
91      # Iterate through values in f^(1) and group any with
92      # similar frequencies together
93      for idx, f1 in enumerate(freq1):
94          assigned = False
95          for group_f1, indices in groupings.items():
96              if np.abs(f1 - group_f1) < threshold:
97                  indices.append(idx)
```

```
                n = len(indices)
                indices = sorted(indices)
                # Get new mean freq 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:
            A_slice = tuple(zip(*product(indices, repeat=2)))
            A = Z2[A_slice].reshape(n, n)
            new_group_z2, _ = np.linalg.eig(A)
            z2[indices] = new_group_z2

    # === Construct E_L and E_R ===
    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
    for i in range(N1 - L1 + 1):
        ER[:, i * col_size : (i + 1) * col_size] = Z1DZ2R
        Z1DZ2R = Z1D @ Z1DZ2R
    # α = diag(E_L^+ E_Y E_R^+)
    alpha = np.diag(np.linalg.pinv(EL) @ EY @ np.linalg.pinv(ER))

    amp = np.abs(alpha) * norm  # a
    phase = np.arctan2(np.imag(alpha), np.real(alpha))  # φ
    freq2 = (0.5 * sw2 / np.pi) * np.imag(np.log(z2)) + offset2  # f^(2)
    damp1 = -sw1 * np.real(np.log(z1))  # η^(1)
    damp2 = -sw2 * np.real(np.log(z2))  # η^(2)
    # θ, as a M×6 array
    theta = np.vstack((amp, phase, freq1, freq2, damp1, damp2)).T

```

```
142     # Remove negative damping factors
143     neg_damp_idx1 = list(np.nonzero(damp1 < 0.)[0])
144     neg_damp_idx2 = list(np.nonzero(damp2 < 0.)[0])
145     neg_damp_idx = list(set(neg_damp_idx1 + neg_damp_idx2))
146     theta = np.delete(theta, neg_damp_idx, axis=0)
147
148     theta = theta[np.argsort(theta[:, 3])]  # order by f^(2)
149     return theta
```

B.2 NLP

B.2.1 Trust Region Algorithm

Code Listing B.5: Steihaug-Toint trust region algorithm. Included is a check for oscillators with negative amplitudes, which causes the routine to terminate, in order for said oscillators to be purged (lines 107–113).

```
1  def trust_steihaug_toint(
2      theta0: np.ndarray,
3      function_factory: FunctionFactory,
4      args: Iterable[Any] = (),
5  ) -> Tuple[np.ndarray, np.ndarray, bool]:
6      """Trust Region algorithm with Steihaug-Toint subroutine.
7
8      Parameters
9      ----------
10     theta0
11         Initial guess, with shape (M, 4).
12
13     function_factory
14         Object for computing the objective, gradient and Hessian.
15
16     args
17         Extra arguments required for computing the objective and its
18         derivatives.
19
20     Returns
21     -------
22     theta
23         Parameter vector at termination.
24
25     errors
26         Errors associated with parameter vector.
```

```
27
28      negative_amps
29          Flag indicating whether or not termination occurred because
30          negative amplitudes were detected.
31      """
32      theta = theta0
33      M = theta.shape[0] // 4
34      factory = function_factory(theta, *args)
35
36      # === Define relevent parameters ===
37      # These have been hard-coded, though in NMR-EsPy they are all
38      # configurable.
39      eta = 0.15,
40      initial_trust_radius = 0.1 * factory.gradient_norm
41      max_trust_radius = 16 * initial_trust_radius
42      epsilon: float = 1.e-8,
43      max_iterations: int = 200,
44      check_neg_amps_every: int = 25,
45
46      k = 0
47      while True:
48          # === Steihaug-Toint ===
49          epsi = min(0.5, np.sqrt(factory.gradient_norm)) *
             ↪  factory.gradient_norm
50          z = np.zeros_like(theta)
51          r = factory.gradient
52          d = -r
53          while True:
54              Bd = factory.hessian @ d
55              dBd = d.T @ Bd
56              if dBd <= 0:
57                  ta, tb = get_boundaries(z, d, trust_radius)
58                  pa = z + ta * d
59                  pb = z + tb * d
60                  p = min(factory.model(pa), factory.model(pb))
61                  hits_boundary = True
62                  break
63              r_sq = r.T @ r
64              alpha = r_sq / dBd
65              z_next = z + alpha * d
66              if sp.linalg.norm(z_next) >= trust_radius:
67                  _, tb = get_boundaries(z, d, trust_radius)
68                  p = z + tb * d
69                  hits_boundary = True
70                  break
```

```
71              r_next = r + alpha * Bd
72              r_next_sq = r_next.T @ r_next
73              if np.sqrt(r_next_sq) < epsi:
74                  hits_boundary = False
75                  p = z_next
76                  break
77              beta_next = r_next_sq / r_sq
78              d_next = -r_next + beta_next * d
79              z = z_next
80              r = r_next
81              d = d_next
82
83          # === Assess effectiveness of update ===
84          predicted_value = factory.model(p)
85          theta_proposed = theta + p
86          factory_proposed = function_factory(theta_proposed, *args)
87          actual_reduction = factory.objective - factory_proposed.objective
88          predicted_reduction = factory.objective - predicted_value
89          if predicted_reduction <= 0:
90              # No improvement could be found: terminate
91              negative_amps = False
92              break
93          rho = actual_reduction / predicted_reduction
94          if rho < 0.25:
95              # Quadratic model perfomring poorly: reduce TR
96              trust_radius *= 0.25
97          elif rho > 0.75 and hits_boundary:
98              # Quadratic model perfomring well: increase TR
99              trust_radius = min(2 * trust_radius, max_trust_radius)
100         if rho > eta:
101             # Accept update: new iteration
102             theta = theta_proposed
103             factory = factory_proposed
104             k += 1
105
106         # === Check for termination criteria ===
107         if (k % check_neg_amps_every == 0):
108             neg_amps = np.where(theta[amp_slice] <= 0)[0]
109             print(neg_amps)
110             if neg_amps.size > 0:
111                 # Negative amps found: this run in order to purge
112                 negative_amps = True
113                 break
114
115         if factory.gradient_norm < epsilon:
```

```
116             # Convergence
117             negative_amps = False
118             break
119 
120         if k == max_iterations:
121             # Maximum allowed iterations reached
122             negative_amps = False
123             break
124 
125     # Routine terminated: compute errors and return parameter array
126     errors = np.sqrt(
127         factory.objective *
128         np.abs(np.diag(np.linalg.inv(factory.hessian)))
129     )
130     return theta, errors, negative_amps
131 
132 
133 def get_boundaries(
134     z: np.ndarray, d: np.ndarray, trust_radius: float
135 ) -> Tuple[float, float]:
136     """Determine the intersections of the search direction and the trust
137     region."""
138     a = d.T @ d
139     b = 2 * z.T @ d
140     c = (z.T @ z) - (trust_radius ** 2)
141     aux = b + np.copysign(
142         np.sqrt(b * b - 4 * a * c),
143         b,
144     )
145     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 the generation of the fidelity for NLP applied to 1D estimation, as well as the gradient and (approximated) Hessian. The `FunctionFactory` object accepts a parameter set (`theta`) and function (`fun`), which computes the objective, gradient and Hessian. The first time a quantity is requested from the factory, the function is run, and the objective and its derivatives are cached (memoised), such that the next time a quantity is requested, the cached result is used, rather than the expensive function being re-computed. `FunctionFactoryGaussNewton1D` (lines 41–43) inherits from the base class, for specific use in 1D estimation, with an approximated Hessian.

```
1 class FunctionFactory:
2     """Object which computes and memoises the objective, gradient and
3     hessian for a given set of parameters."""
```

```
4       def __init__(self, theta: np.ndarray, fun: callable, *args) -> None:
5           self.theta = theta
6           self._new_theta = True
7           self.fun = fun
8           self.obj = None
9           self.grad = None
10          self.hess = None
11          self.args = args
12
13      def _compute_if_needed(self):
14          """Compute the obj, grad and Hess if they haven't been yet"""
15          if self.obj is None:
16              self.obj, self.grad, self.hess = self.fun(self.theta,
                ↪  *self.args)
17
18      def model(self, p) -> float:
19          return self.objective + self.gradient @ p + 0.5 * (p.T @
            ↪  self.hessian @ p)
20
21      @property
22      def objective(self) -> float:
23          self._compute_if_needed()
24          return self.obj
25
26      @property
27      def gradient(self) -> np.ndarray:
28          self._compute_if_needed()
29          return self.grad
30
31      @property
32      def gradient_norm(self) -> float:
33          return sp.linalg.norm(self.gradient)
34
35      @property
36      def hessian(self) -> np.ndarray:
37          self._compute_if_needed()
38          return self.hess
39
40
41  class FunctionFactoryGaussNewton1D(FunctionFactory):
42      def __init__(self, theta: np.ndarray, *args) -> None:
43          super().__init__(theta, obj_grad_gauss_newton_hess_1d, *args)
44
45
46  def obj_grad_gauss_newton_hess_1d(
```

```
    theta: np.ndarray, *args: Tuple[int, int, np.ndarray],
) -> Tuple[float, np.ndarray, np.ndarray]:
    Y, tp = args  # Unpack args: FID, timepoints
    N = Y.shape[0]
    M = theta.shape[0] // 4
    X_per_osc = np.exp(
        np.outer(tp, (2j * np.pi * theta[2 * M : 3 * M] - theta[3 * M :]))
    ) * (theta[:M] * np.exp(1j * theta[M : 2 * M]))

    # Jacobian: all first partial derivatives, ∂X/∂θ
    jac = np.zeros((N, 4 * M), dtype="complex")
    jac[:, :M] = X_per_osc / theta[:M]  # ∂X/∂a
    jac[:, M : 2 * M] = 1j * X_per_osc  # ∂X/∂ϕ
    jac[:, 2 * M : 3 * M] = \
        np.einsum("ij,i->ij", X_per_osc, 2j * np.pi * tp)  # ∂X/∂f^(1)
    jac[:, 3 * M :] = np.einsum("ij,i->ij", X_per_osc, -tp)  # ∂X/∂η^(1)

    X = np.einsum("ij->i", X_per_osc)
    Y_minus_X = Y - X

    obj = (Y_minus_X.conj().T @ Y_minus_X).real  # ℱ(θ)
    grad = -2 * (Y_minus_X.conj().T @ jac).real  # ∇ℱ(θ)
    hess = 2 * (jac.conj().T @ jac).real  # ∇²ℱ(θ)

    # === Determine phase variance and derivatives ===
    phi = theta[M : 2 * M]
    cos_phi = np.cos(phi)
    cos_sum = np.sum(cos_phi)
    sin_phi = np.sin(phi)
    sin_sum = np.sum(sin_phi)
    R = np.sqrt(cos_sum ** 2 + sin_sum ** 2)
    pv_obj = 1 - (R / M)
    pv_grad = (sin_phi * cos_sum - cos_phi * sin_sum) / (M * R)
    x = (sin_phi * cos_sum) - (cos_phi * sin_sum)
    term_1 = np.outer(x, x) / (R ** 2)
    phi_array = np.zeros((M, M))
    phi_array[:] = phi
    term_2 = -np.cos(phi_array.T - phi_array)
    pv_hess = term_1 + term_2
    pv_hess[np.diag_indices(M)] += cos_phi * cos_sum + sin_phi * sin_sum
    pv_hess /= M * R
    obj += pv_obj  # ℱ_ϕ(θ)
    grad[M : 2 * M] += pv_grad  # ∇ℱ_ϕ(θ)
    hess[M : 2 * M, M : 2 * M] += pv_hess  # ∇²ℱ_ϕ(θ)

```

```
92      return obj, grad, hess
```

B.2.3 The main routine

Code Listing B.7: Hello

```
1  def nlp(
2      Y: np.ndarray,
3      sw: float,
4      offset: float,
5      theta0: np.ndarray,
6  ) -> Tuple[np.ndarray, np.ndarray]:
7      """Nonlinear programming routine for 1D FID estimation.
8
9      Parameters
10     ----------
11     Y
12         FID.
13
14     sw
15         Sweep width (Hz).
16
17     offset
18         Tranmitter offset (Hz).
19
20     theta0
21         Initial guess, of shape (M, 4)
22
23     Returns
24     -------
25     theta
26         Parameter estimate.
27
28     errors
29         Errors associated with theta.
30     """
31     norm = np.linalg.norm(Y)
32     Y /= norm
33     N = Y.shape[0]
34     M = theta0.shape[0]
35     # Flatten parameter array: Fortran (column-wise) ordering
36     theta0_vec = theta0.flatten(order="F")
37     theta0_vec[:M] /= norm  # Normalise amplitudes
38     # Remove transmitter offset from frequencies
```

```
    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: Hello

```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
```

```
39                 break
40         # No match: create new multiplet group
41         if not assigned:
42             multiplets[fc] = [m]
43 
44     return multiplets
```

Information on Datasets

C

C.1 Simulated datasets

Many of the simulated datasets presented in this work were generated using the Spinach Matlab® package[128]. In each case, the dataset was generated via a call to the `new_spinach` method associated with the relevant `Estimator` object in NMR-EsPy. `new_spinach` works by instantiating a Matlab engine from Python, 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.

C.1.1 2DJ

The simulated datasets for the "Four multiplets" and Sucrose results in Sections 4.3.1 and 4.3.2 were generated using NMR-EsPy's `Estimator2DJ.new_spinach` method, with key parameters supplied to the simulation provided by Table C.1.

Parameter	Four Multiplets	Sucrose
$f_{\text{bf}}^{(1)}$ (MHz)	500	300
$f_{\text{off}}^{(2)}$ (Hz)	0	1000
$f_{\text{off}}^{(2)}$ (ppm)	0	3.333
$f_{\text{sw}}^{(1)}$ (Hz)	40	30
$f_{\text{sw}}^{(2)}$ (Hz)	1000	2200
$f_{\text{sw}}^{(2)}$ (ppm)	2	7.333
$N^{(1)}$	128	64
$N^{(2)}$	1024	4096

Table C.1: Experiment parameters 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 as follows:

CODE LISTING C.1: Hello

```
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
```

The chemical shifts and couplings for sucrose were extracted from a GAUSSIAN[120] log-file which ships with SPINACH at the path `SPINACHROOT/examples/standard_systems/sucrose`.

`log`. Isotropic chemical shifts were determined for each spin i via

$$\delta_i = \frac{\mathrm{Tr}\left(\boldsymbol{\sigma}_i\right)}{3}, \tag{C.1}$$

where $\boldsymbol{\sigma}_i \in \mathbb{R}^{3\times3}$ is the computed chemical shift tensor of spin i.

The chemical shifts and couplings associated with the "four multiplets" datasets were generated as described in the main text.

C.1.2 Inversion recovery

As described in the Spinach documentation[129], longitudinal spin states are assigned the rate $1/T_1$ and transverse states are assigned the rate $1/T_2$. Multi-spin states are assigned rates which are the summation of each spin's relevant rate.

C.1.3 Spinach Simulations

Table C.2 provides a specification of the chemical shifts and scalar couplings that made up the spin systems used to construct simulated data with Spinach. Tables C.1 and **??** specify key parameters used in each 2DJ and inversion recovery simulation, respectively.

Spin Systems

Sucrose The chemical shifts and couplings for sucrose were extracted from a Gaussian[120] logfile which ships with Spinach at the path `SPINACHROOT/examples/standard_systems/sucrose.log`. Isotropic chemical shifts were determined for each spin i via

$$\delta_i = \frac{\mathrm{Tr}\left(\boldsymbol{\sigma}_i\right)}{3}, \tag{C.2}$$

where $\boldsymbol{\sigma}_i \in \mathbb{R}^{3\times3}$ is the computed chemical shift tensor of spin i.

Strychnine The strychnine spin system was derived from the Spinach function `<SPINACHROOT>/etc/strychnine.` which returns a spin system specification using chemical shifts and scalar couplings from[130: Appendix 5], and atomic coordinates from[131: Supplementary Material]. To determine T_1 and T_2 values for each spin, the relaxation superoperator $\boldsymbol{R}$ was constructed according to Redfield theory, under the assumption that the molecule was undergoing spherical isotropic rotation with a rotational correlation time of 200 ps, in a magnetic field of 700 MHz. Individual T_1s and T_2s were

then extracted using

$$T_{(1/2),i} = \frac{1}{R_{(1/2),i}}, \tag{C.3a}$$

$$R_{1,i} = -\Re\left(\boldsymbol{l}_{z,i}^{\dagger}\boldsymbol{R}\boldsymbol{l}_{z,i}\right), \tag{C.3b}$$

$$R_{2,i} = -\Re\left(\boldsymbol{l}_{+,i}^{\dagger}\boldsymbol{R}\boldsymbol{l}_{+,i}\right), \tag{C.3c}$$

where $\boldsymbol{l}_{z,i}$ is the state vector of the Hilbert-space operator $\hat{L}_z$ for spin i, and $\boldsymbol{l}_{+,i}$ is the corresponding vector for the $\hat{L}_+$ operator.

Table C.2: The isotropic chemical shifts (∂), corresponding rotating frame frequencies (ω_{rot}), scalar couplings (J), and relaxation times (T_1, T_2, if applicable) associated with spin systems used in Spinach simulations.

Spin	∂ (ppm)	ω_{rot} (Hz)	J (Hz)	T_1 (s)	T_2 (s)
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	–	–

Continues on next page...

Spin	δ (ppm)	ω_{rot} (Hz)	J (Hz)	T_1 (s)	T_2 (s)
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	–	–
			Sucrose		
A	6.005	1801.6	**B**: 2.285	–	–
B	3.510	1053.1	**A**: 2.285, **C**: 4.657, **H**: 4.828	–	–
C	3.934	1180.2	**B**: 4.657, **D**: 4.326	–	–
D	3.423	1027.0	**C**: 4.326, **E**: 4.851	–	–
E	4.554	1366.1	**D**: 4.851, **F**: 5.440, **G**: 2.288	–	–
F	3.891	1167.4	**E**: 5.440, **G**: −6.210	–	–
G	4.287	1286.2	**E**: 2.288, **F**: −6.210, **K**: 7.256	–	–
H	3.332	999.5	**B**: 4.828	–	–
I	1.908	572.3	–	–	–
J	1.555	466.6	–	–	–
K	0.644	193.3	**G**: 7.256	–	–
L	4.042	1212.5	**M**: −4.005, **S**: 1.460	–	–
M	4.517	1355.0	**L**: −4.005	–	–
N	3.889	1166.7	**O**: 4.253	–	–
O	4.635	1390.4	**N**: 4.253, **P**: 4.448, **U**: 3.221	–	–
P	4.160	1248.0	**O**: 4.448, **R**: 4.733	–	–
Q	4.021	1206.2	**R**: −4.182	–	–
R	4.408	1322.4	**P**: 4.733, **Q**: −4.182, **V**: 1.350	–	–
S	0.311	93.3	**L**: 1.460	–	–
T	1.334	400.2	–	–	–
U	0.893	267.9	**O**: 3.221	–	–
V	0.150	45.0	**R**: 1.350	–	–

Continues on next page...

Spin	δ (ppm)	ω_{rot} (Hz)	J (Hz)	T_1 (s)	T_2 (s)
Five Multiplets, Run 1					
A	1.33	665.99	**F:** 10.965, **G:** 12.657, **H:** 17.070	2.178	–
B	1.47	735.44	**F:** 3.610, **G:** 2.543, **H:** 8.448	4.430	–
C	1.31	653.87	**F:** 10.630, **G:** 6.282, **H:** 3.012	3.319	–
D	1.41	705.01	**F:** 8.101, **G:** 4.589, **H:** 9.068	1.007	–
E	1.30	650.23	**F:** 3.014, **G:** 16.537, **H:** 15.587	4.992	–
Five Multiplets, Run 2					
A	1.47	733.07	**F:** 19.488, **G:** 18.279, **H:** 3.147	3.600	–
B	1.36	681.25	**F:** 11.924, **G:** 8.400, **H:** 5.515	3.905	–
C	1.30	651.61	**F:** 13.672, **G:** 6.543, **H:** 16.275	4.291	–
D	1.43	713.48	**F:** 12.007, **G:** 5.141, **H:** 9.981	1.687	–
E	1.49	744.53	**F:** 8.715, **G:** 14.309, **H:** 9.805	3.214	–
Five Multiplets, Run 3					
A	1.32	658.87	**F:** 4.984, **G:** 18.119, **H:** 10.642	3.846	–
B	1.44	719.50	**F:** 13.518, **G:** 14.381, **H:** 3.074	1.018	–
C	1.37	683.34	**F:** 8.758, **G:** 16.689, **H:** 12.956	4.766	–
D	1.35	676.98	**F:** 17.648, **G:** 7.514, **H:** 3.918	3.414	–
E	1.49	746.76	**F:** 16.396, **G:** 7.455, **H:** 11.352	1.827	–
Strychinine					
A	7.167	2150.1	**B**: 7.490, **C**: 1.080, **D**: 0.230	1.74	1.26
B	7.098	2129.4	**A**: 7.490, **C**: 7.440, **D**: 0.980	2.47	1.78
C	7.255	2176.5	**A**: 1.080, **B**: 7.440, **D**: 7.900	2.50	1.80
D	8.092	2427.6	**A**: 0.230, **B**: 0.980, **C**: 7.900	5.12	3.69
E	3.860	1158.0	**I**: 10.410	1.26	0.91
F	3.132	939.6	**G**: −17.340, **H**: 3.340	0.52	0.37
G	2.670	801.0	**F**: −17.340, **H**: 8.470	0.55	0.39
H	4.288	1286.4	**F**: 3.340, **G**: 8.470, **I**: 3.300	0.89	0.64
I	1.276	382.8	**E**: 10.410, **H**: 3.300, **J**: 3.290	0.97	0.70
J	3.150	945.0	**I**: 3.290, **K**: 4.110, **L**: 1.960, **R**: 1.610, **T**: 0.470	1.07	0.77

Continues on next page...

Spin	δ (ppm)	ω_{rot} (Hz)	J (Hz)	T_1 (s)	T_2 (s)
K	2.360	708.0	**J**: 4.110, **L**: −14.350, **M**: 4.330	0.42	0.30
L	1.462	438.6	**J**: 1.960, **K**: −14.350, **M**: 2.420	0.41	0.30
M	3.963	1188.9	**K**: 4.330, **L**: 2.420	1.15	0.83
N	1.890	567.0	**O**: −13.900, **P**: 5.500, **Q**: 7.200	0.48	0.35
O	1.890	567.0	**N**: −13.900, **P**: 3.200, **Q**: 10.700	0.47	0.34
P	3.219	965.7	**N**: 5.500, **O**: 3.200, **Q**: −13.900	0.49	0.36
Q	2.878	863.4	**N**: 7.200, **O**: 10.700, **P**: −13.900	0.42	0.30
R	3.716	1114.8	**J**: 1.610, **S**: −14.800, **T**: 1.790	0.47	0.34
S	2.745	823.5	**R**: −14.800	0.45	0.33
T	5.915	1774.5	**J**: 0.470, **R**: 1.790, **U**: 7.000, **V**: 6.100	1.48	1.06
U	4.148	1244.4	**T**: 7.000, **V**: −13.800	0.48	0.34
V	4.066	1219.8	**T**: 6.100, **U**: −13.800	0.55	0.40

Table of spinach experiment parameters Description of 2DJ and Invrec simulations (i.e. relaxation model used, approximations to basis used etc.

C.2 Experimental datasets

C.2.1 Structures

C.2.2 Diffusion datasets

The andrographolide diffusion dataset (Figure 3.5) was acquired using the one-shot DOSY pulse sequence[88] (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.2.

The glucose/valine/threonine diffusion dataset (Figure 3.6) 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[87]. The pulse sequence is displayed in Figure C.3.

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 $\pi/2(\Phi_1) \rightarrow t^{(1)}/2 \rightarrow \pi(\Phi_2) \rightarrow$

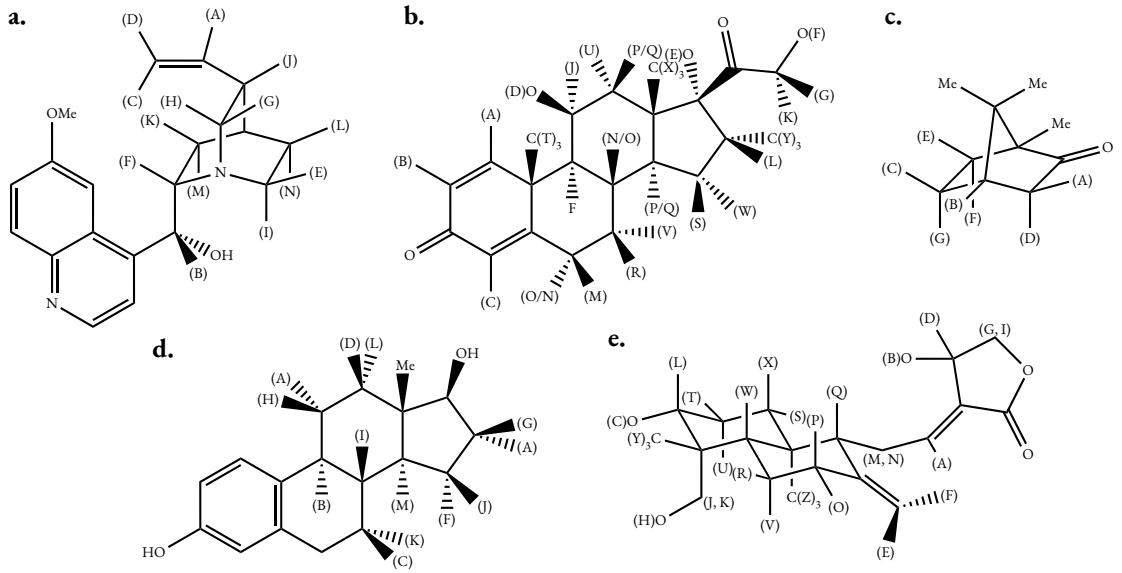


Figure C.1: The molecular structures of species giving rise to the experimental NMR datasets considered in this work. **a.** Quinine, **b.** Dexamethasone, **c.** Camphor, **d.** 17β-estradiol, **e.** Andrographolide. Proton environments giving rise to signals which are considered in this work are denoted with bracketed alphabetical characters. Non-bracketed alphabetical characters denote chemical symbols. Me denotes the methyl group, equivalent to CH_3. **TODO: check assignments (esp. estradiol). If more structures need adding, edit the ChemDraw file on Chive called `simon_stuff/thesis_structures.cdxml`. Use EB-Garamond for atom labels. One a structure is made, scale to 75% of original size, and set font to 7pt. Then in inkscape, rescale this by multiplying by 0.8.**

$t^{(1)}/2 \rightarrow t^{(2)}\,(\Phi_{\mathrm{rx}})$, with the EXORCYCLE phase-cycling scheme[24: Section 11.6]:

$$\begin{array}{lllll} \Phi_1: & 0° & 0° & 0° & 0° \\ \Phi_2: & 0° & 90° & 180° & 270° \\ \Phi_{\mathrm{rx}}: & 0° & 180° & 0° & 180° \end{array}$$

Key experiment parameters are provided in Table C.3.

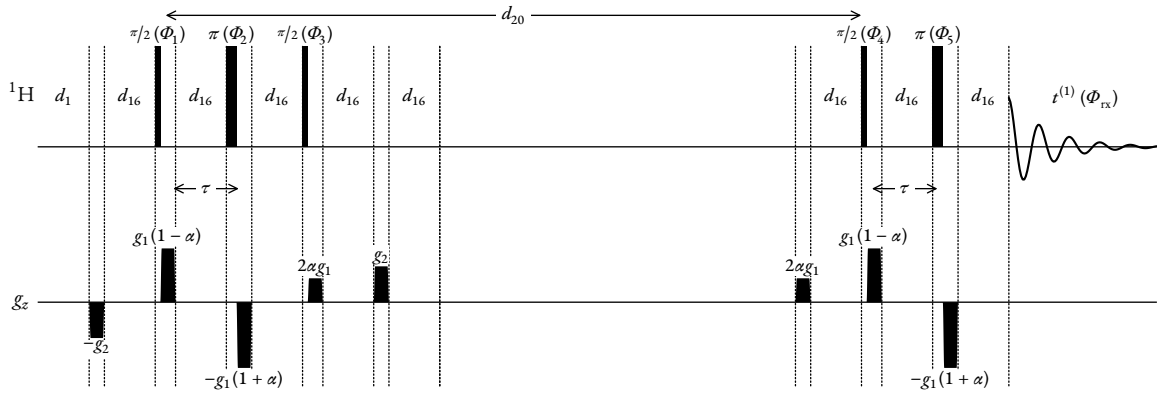


Figure C.2: Oneshot DOSY pulse sequence used for the acquisition of andrographolide data (Figure 3.5). All delays are included, though they are not to scale. Delays: d_1 (relaxation delay): 1.5 s, d_{16} (gradient recovery delay): 200 µs, d_{20} (diffusion time, equivalent to Δ): 0.1 s. Hard pulses had 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: $\Phi_1 : 2 \times (0°, 180°)$; $\Phi_2 : 4 \times 0°$; $\Phi_3 : 4 \times 0°$; $\Phi_4 : 2 \times 0°, 2 \times 180°$; $\Phi_5 : 4 \times 0°$; $\Phi_{rx} : 0°, 180°, 180°, 0°$.

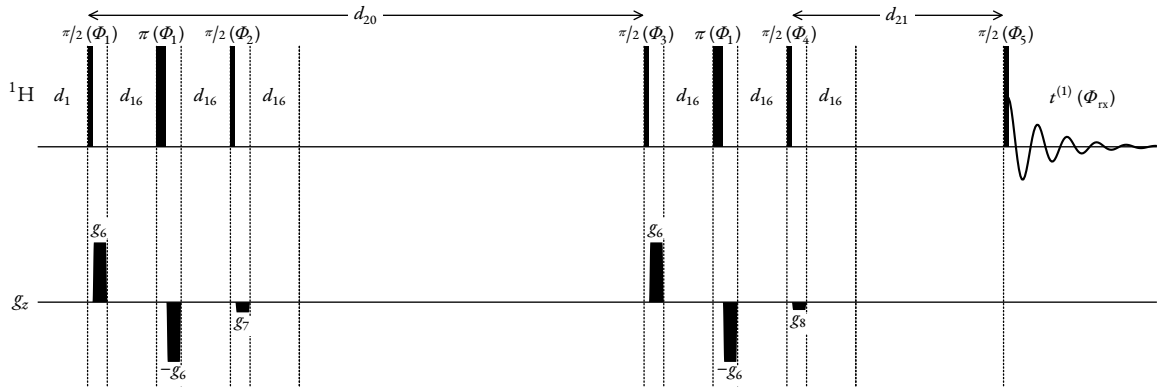


Figure C.3: Pulse sequence used for the acquisition of glucose/threonine/valine diffusion data (Figure 3.6). All delays are included, though they are not to scale. Delays: d_1 (relaxation delay): 6 s, d_{16} (gradient recovery delay): 200 µs, d_{20} (diffusion time, equivalent to Δ): 0.1 s, d_{21} (eddy-current delay): 5 ms. Hard pulses had a power of 18.204 W, with the duration of the $\pi/2$ pulse being 10 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, with the values used being (G cm^{-1}): 4.500, 12.728, 17.428, 21.107, 24.233, 27.000, 29.508, 31.820, 33.974, 36.000. Spoiler gradients had a duration of 600 µs. The gradients had the relative strengths $g_7 = -0.1713 g_6$ and $g_8 = -0.1317 g_6$. The phase cycling scheme used was: $\Phi_1 : 32 \times 0°$; $\Phi_2 : 8 \times (2 \times 0°, 2 \times 180°)$; $\Phi_3 : 2 \times (4 \times 0°, 4 \times 180°, 4 \times 90°, 4 \times 270°)$; $\Phi_4 : 2{\times}(2{\times}(0°, 180°), 2{\times}(180°, 0°), 2{\times}(90°, 270°), 2{\times}(270°, 90°))$; $\Phi_5 : 2{\times}(4{\times}0°, 4{\times}180°, 4{\times}90°, 4{\times}270°)$; $\Phi_{rx} : 2 \times (0°, 180°, 180°, 0°, 180°, 0°0°, 180°, 270°, 90°, 90°, 270°, 90°, 270°, 270°, 90°)$.

	Quinine	Dexamethasone	Camphor	Estradiol
f_{bf} (MHz)	500.13	600.18	500.13	500.3
$f_{\mathrm{off}}^{(2)}$ (Hz)	2500	2815.4	1000	2501.5
$f_{\mathrm{sw}}^{(1)}$ (Hz)	50	50	50	100
$f_{\mathrm{sw}}^{(2)}$ (Hz)	7500	7211.5	5000	5000
$f_{\mathrm{sw}}^{(2)}$ (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.3: Noteworthy experiment parameters for the 2D J-Reolved and PSYCHE experiments run. NS: Number of scans, DS: Number of dummy scans, PLW1: Hard pulse power (W), P1: Duration of $\pi/2$ pulse, D1: Duration of relaxation delay.

C.2.4 PSYCHE datasets

The pulse sequence used for the acquisition of the estradiol PSYCHE spectrum (Figure 4.9) 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.3.

The pure shift spectrum of dexamethasone (Figure 4.8.a) was acquired using a TSE-PSYCHE experiment (Figure C.5). The pulse sequence file `UoM_1d_if_tsepsyche_ts4x` can be obtained via the link `https://research.manchester.ac.uk/en/datasets/manchester-pure-shift-nmr-workshop-bruker-software`.

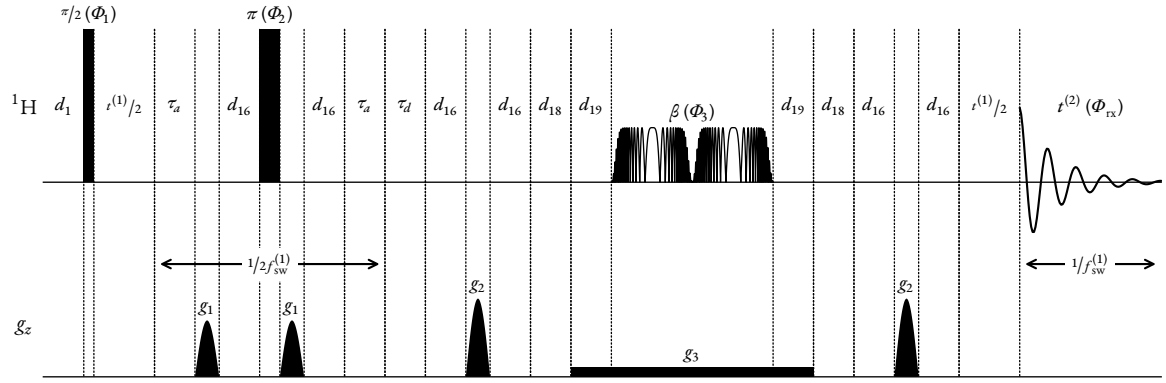


Figure C.4: PSYCHE pulse sequence used for the acquisition of estradiol data. All delays are included, though they are not to scale. Delays: d_1 (relaxation delay): 1 s, d_{16}: (gradient recovery delay): 200 µs, d_{18}: 200 µs, d_{19}: 1 ms, τ_a: 1.3 ms, τ_d: 18.9 ms. The PSYCHE element featured two saltire chirp pulses with a wideband, uniform rate, smooth truncation (WURST)[132] 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: $\Phi_1 : 2 \times (0°, 180°)$; $\Phi_2 : 4 \times 0°$; $\Phi_3 : 2 \times 0°, 2 \times 90°$; $\Phi_{rx} : 0°, 180°, 180°, 0°$.

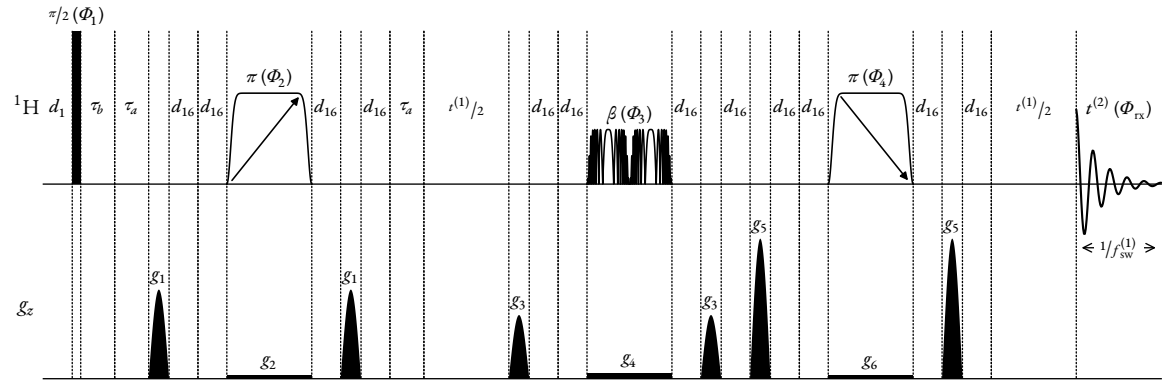


Figure C.5: TSE-PSYCHE pulse sequence used for the acquisition of dexamethasone data (Figure 4.8.a). Delays: d_1 (relaxation delay): 2 s, d_{16}: (gradient recovery delay): 200 µs, τ_a: 5 ms (= $1/4f_{sw}^{(1)}$). The hard $\pi/2$ pulse had a duration of 12 µs, and a power of 24 W. The two π pulses were unidirectional frequency-swept (chirped) pulses, with the first pulse sweeping from low to high frequencies, and the second pulse sweeping from high to low. These each had a WURST amplitude envelope, lasted a duration of 40 ms, and had a power of 11.05 mW. The PSYCHE element had a target flip angle $\beta = 15°$, and featured two saltire chirp pulses. Both saltire pulses had a WURST amplitude envelope, a duration of 15 ms, and a power of 1.28 mW. g_1, g_3 and g_5 were gradients for coherence order selection. Each comprised a 100-point sine profile, and lasted 1 ms. g_2, g_4 and g_6 were weak rectangular gradients which were applied at the same time as the chirped pulses. The magnitudes of gradients g_1 to g_6 as a percentage of the maximum permitted gradient were, respectively: 49%, 2%, 35%, 3%, 77%, 2%. The phase cycling scheme used was: $\Phi_1 : 8 \times 0°$; $\Phi_2 : 2 \times (2 \times 0°, 2 \times 180°)$; $\Phi_3 : 2 \times (0°, 90°), 2 \times (180°, 270°)$; $\Phi_4 : 8 \times 0°$; $\Phi_{rx} : 4 \times (0°, 180°)$.

NMR-EsPy walkthroughs

D

The remainder of this thesis is an insert from the documentation of version 2.0 of NMR-EsPy. The section of the documentation included provides walkthroughs describing how to use the package's API for the consideration of 1D and 2DJ NMR datasets **Sequential data too?**. These walkthroughs provide a short description of the key features associated with the package, and is an ideal first place to get up-and-running with using it.

Also included in the full documentation are instructions on installation and using the GUIs for 1D and 2DJ estimation, and a complete reference for the complete API. The most up-to-date version of the documentation can be found at:

HTML: `https://foroozandehgroup.github.io/NMR-EsPy/`

PDF: ???

The source code for NMR-EsPy is hosted on the Foroozandeh group's GitHub page at `https://github.com/foroozandehgroup/NMR-EsPy`.

CHAPTER

TWO

WALKTHROUGHS

In this chapter, walkthroughs are provided to help you get up-and-running with the main features of the NMR-EsPy API. For a rigorous description of the API, you should consult the *Reference* afterwards.

2.1 Using `Estimator1D`

The `nmrespy.Estimator1D` class is provided for the consideration of 1D NMR data.

2.1.1 Generating an instance

There are a few ways to create a new instance of the estimator depending on the source of the data

Bruker data

It is possible to load both raw FID data and processed spectral data from Bruker using `new_bruker()`. All that is needed is the path to the dataset:

1. If you wish to import FID data, set the path as `"<path_to_data>/<expno>/"`. There should be an `fid` file and an `acqus` file directly under this directory. The data in the `fid` file will be imported, and the artefact from digital filtering is removed by a first-order phase shift.

   Note: If you import FID data, there is a high chance that you will need to phase the data, and apply baseline correction before proceeding to run estimation. Look at `phase_data()` and `baseline_correction()`, respectively.

```
>>> import nmrespy as ne
>>> estimator = ne.Estimator1D.new_bruker("/home/simon/nmr_data/andrographolide/1")
>>> estimator.phase_data(p0=2.653, p1=-5.686, pivot=13596)
>>> estimator.baseline_correction()
```

2. To import processed data, set the path as `"<path_to_data>/<expno>/pdata/<procno>"`. There should be a `1r` file and a `procs` file directly under this directory. The data in `1r` will be Inverse Fourier Transformed, and the resulting time-domain signal is sliced so that only the first half is retained.

   Note: It can be more 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 oscillator parameters

You can create an estimator with synthetic data constructed from known parameters using `new_from_parameters()`. The parameters must be provided as a 2D NumPy array with `params.shape[1] == 4`. Each row should contain an oscillator's amplitude, phase (rad), frequency (Hz), and damping factor (s^{-1}).

```
>>> import 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,
>>>     sw=1.2 * sfo,  # 1ppm
>>>     offset=4.1 * sfo,  # 4.1ppm
>>>     sfo=sfo,
>>>     snr=40.,
>>> )
```

Note: For the rest of this tutorial, we will be using the estimator created in the above code snippet.

Simulated data from Spinach

Assuming you have installed the *relevant requirements*, you can create an instance with data simulated using Spinach with `new_spinach()`. The spin system is defined by a specification of isotropic chemical shifts, and scalar couplings:

- For the chemical shift, a list of floats is required.
- For J-couplings, a list with 3-element tuples of the form `(spin1, spin2, coupling)` is required. **N.B. the spin indices start at "1" rather than "0"**.

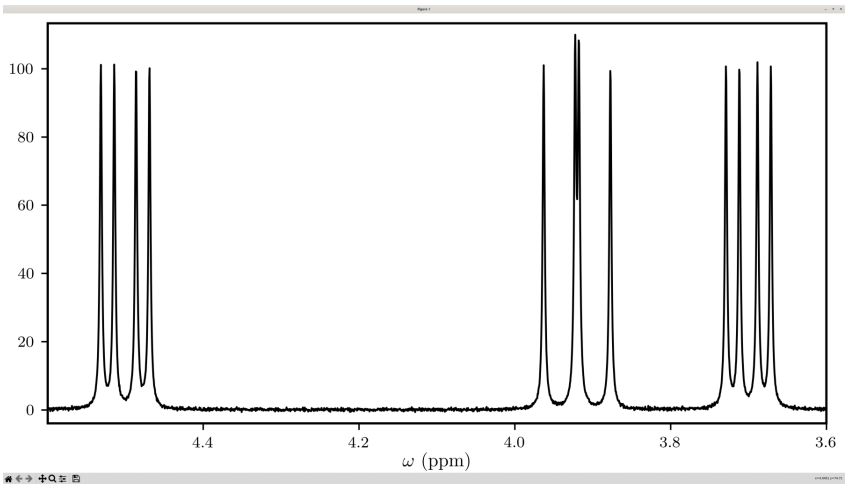
It can take some time to run this function is it involves (a) starting up MATLAB and (b) running a simulation of the experiment.

```
>>> 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.
>>> offset = 4.1 * sfo  # Hz
>>> sw = 1.2 * sfo
>>> estimator = ne.Estimator1D.new_spinach(
>>>     shifts=shifts,
>>>     couplings=couplings,
>>>     pts=2048,
>>>     sw=sw,
>>>     offset=offset,
>>>     sfo=sfo,
>>> )
```

2.1.2 Viewing and accessing the dataset

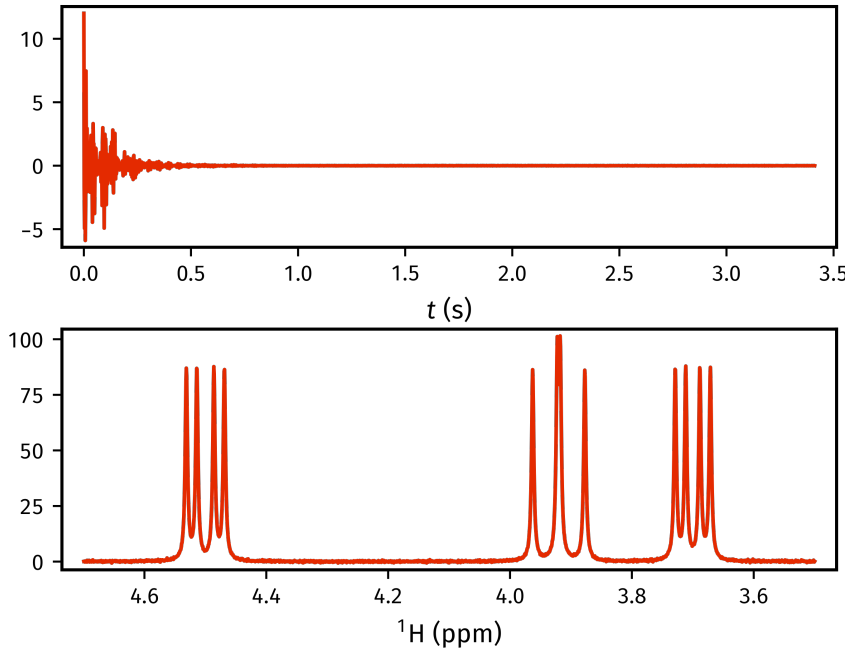
You can inspect the data associated with the estimator with `view_data()`, which loads an interactive matplotlib figure:

```
>>> estimator.view_data(freq_unit="ppm")
```



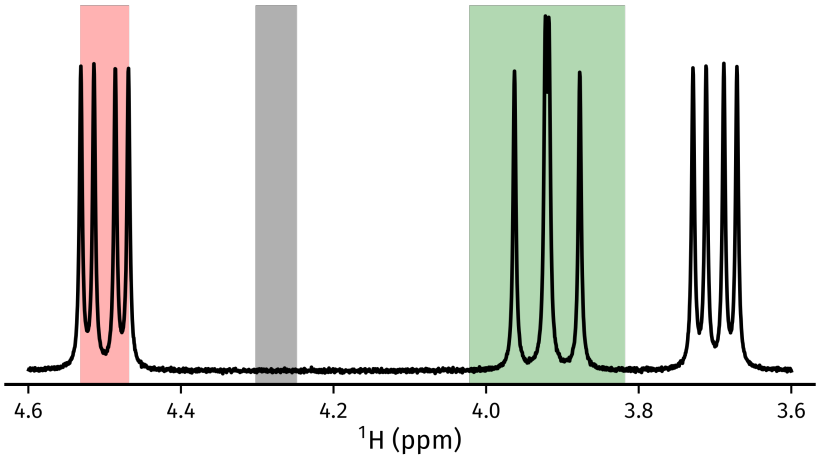
You can access the time-domain data with the `data()` property, and the associated time-points can be retrieved using `get_timepoints()`. The spectral data is accessed with `spectrum()`, and the corresponding chemical shifts with `get_shifts()`.

```
>>> 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 for the dataset is facilitated using the `estimate()` method. In most scenarios, your dataset will possess too many oscillators for it to be feasible computationally to estimate the entire signal at once. For this reason, NMR-EsPy generates frequency-filtered "sub-FIDs" to break the problem down into more manageable chunks. To create suitable sub-FIDs, it is important to select regions where the bounds are placed at points that comprise the baseline. As well as this, a region that comprises just the baseline must be indicated. In the figure below, the red region would be inappropriate as it slices through signal. The green region is acceptable, as the bounds are located on the baseline. Finally, the grey region is a suitable noise region as it contains only baseline.



For our dataset, we will estimate three regions, comprising each multiplet structure in the spectrum. A region should be given as a tuple of 2 floats, specifying the left and right boundaries of the region of interest (the order of these doesn't matter). By default, these are assumed to be given in Hz, unless `region_unit` is set to `"ppm"`.

```
>>> regions = [(4.6, 4.4), (4.02, 3.82), (3.8, 3.6)]
>>> noise_region = (4.3, 4.25)
>>> for region in regions:
>>>     estimator.estimate(
>>>         region=region, noise_region=noise_region, region_unit="ppm",
>>>     )
```

2.1.4 Inspecting estimation results

Note: Result Indices

Each time the `estimate()` method is called, the result is appended to a list of all generated results. For many methods that use estimation results, an argument called `indices` exists. This lets you specify the results you are interested in. By default (`indices = None`) all results will be used. A subset of the results can be considered by including a list of integers. For example `indices = [0, 2]` would mean only the 1st and 3rd results acquired with the estimator are considered.

A NumPy array of the generated results can be acquired using `get_params()`. The corresponding errors associated with each parameters are obtained with `get_errors()`.

```
>>> # All params, frequencies in Hz:
>>> estimator.get_params()
[[ 1.0018e+00  1.5921e-03  1.8356e+03  7.0187e+00]
 [ 1.0003e+00  2.4881e-03  1.8442e+03  6.9968e+00]
 [ 1.0024e+00  1.5817e-03  1.8558e+03  7.0281e+00]
 [ 1.0008e+00  9.1591e-04  1.8644e+03  7.0007e+00]
 [ 1.0022e+00  7.1936e-04  1.9386e+03  7.0109e+00]
 [ 9.9470e-01 -7.4609e-04  1.9588e+03  6.9866e+00]
 [ 1.0080e+00 -1.0112e-03  1.9612e+03  7.0448e+00]
 [ 1.0009e+00 -7.1398e-04  1.9814e+03  7.0131e+00]
 [ 1.0003e+00  1.1306e-03  2.2344e+03  7.0095e+00]
 [ 1.0011e+00  6.0150e-04  2.2430e+03  7.0011e+00]
```

(continues on next page)

(continued from previous page)

```
 [ 9.9902e-01  2.8231e-04  2.2570e+03  6.9856e+00]
 [ 1.0004e+00 -1.8229e-03  2.2656e+03  7.0057e+00]]
>>> # All errors, frequencies in Hz
>>> estimator.get_errors()
[[0.0013 0.0013 0.0019 0.0121]
 [0.0014 0.0014 0.002  0.0124]
 [0.0014 0.0014 0.002  0.0125]
 [0.0013 0.0013 0.0019 0.012 ]
 [0.0012 0.0012 0.0018 0.0114]
 [0.0036 0.0036 0.0034 0.0212]
 [0.0036 0.0036 0.0034 0.0213]
 [0.0012 0.0012 0.0018 0.0114]
 [0.0013 0.0013 0.0019 0.0116]
 [0.0013 0.0013 0.0019 0.0118]
 [0.0013 0.0013 0.0019 0.0118]
 [0.0013 0.0013 0.0018 0.0116]]
>>> # Params for first region, frequencies in ppm
>>> estimator.get_params(indices=[0], funit="ppm")
[[ 1.0003e+00  1.1306e-03  4.4688e+00  7.0095e+00]
 [ 1.0011e+00  6.0150e-04  4.4860e+00  7.0011e+00]
 [ 9.9902e-01  2.8231e-04  4.5140e+00  6.9856e+00]
 [ 1.0004e+00 -1.8229e-03  4.5312e+00  7.0057e+00]]
>>> # Params for second and third regions, split up
>>> estimator.get_params(indices=[1, 2], merge=False, funit="ppm")
[array([[ 1.0022e+00,  7.1936e-04,  3.8772e+00,  7.0109e+00],
       [ 9.9470e-01, -7.4609e-04,  3.9176e+00,  6.9866e+00],
       [ 1.0080e+00, -1.0112e-03,  3.9224e+00,  7.0448e+00],
       [ 1.0009e+00, -7.1398e-04,  3.9628e+00,  7.0131e+00]]), array([[1.0018e+00, 1.5921e-03,
   3.6712e+00, 7.0187e+00],
       [1.0003e+00, 2.4881e-03, 3.6884e+00, 6.9968e+00],
       [1.0024e+00, 1.5817e-03, 3.7116e+00, 7.0281e+00],
       [1.0008e+00, 9.1591e-04, 3.7288e+00, 7.0007e+00]])]
```

Writing result tables

Tables of parameters can be saved to `.txt` and `.pdf` formats. using `write_result()`. For PDF generation, you will need a working LaTeX installation. See the *installation instructions*.

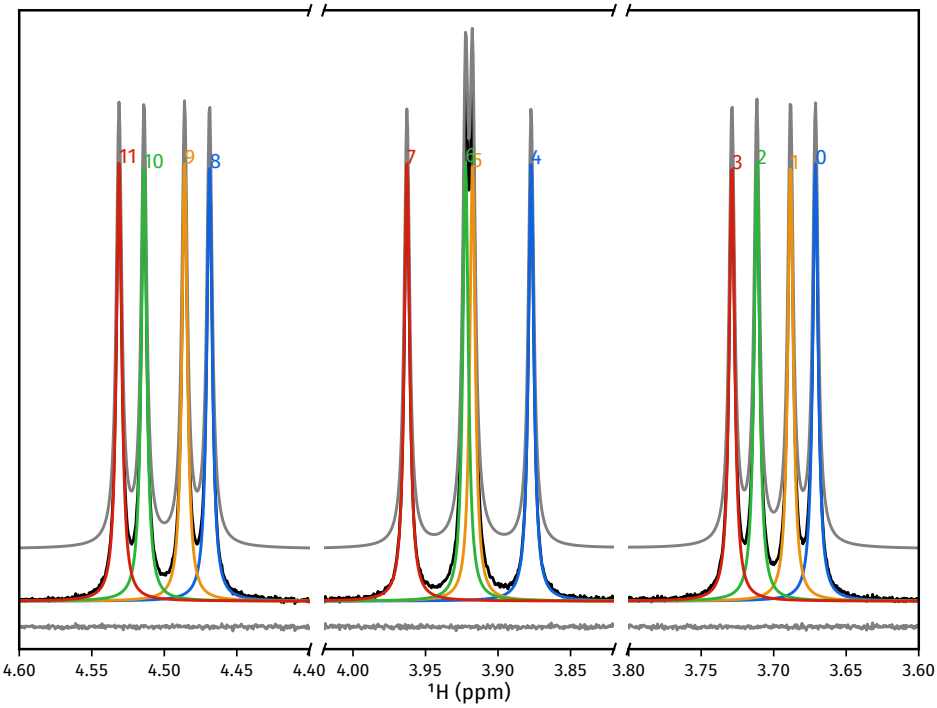
```
>>> for fmt in ("txt", "pdf"):
>>>     estimator.write_result(
>>>         path="tutorial_1d",
>>>         fmt=fmt,
>>>         description="Simulated 2,3-Dibromopropanoic acid signal.",
>>>     )
Saved file tutorial_1d.txt.
Saved file tutorial_1d.tex.
Saved file tutorial_1d.pdf.
You can view and customise the corresponding TeX file at tutorial_1d.tex.
```

Creating result plots

Figures giving an overview of the estimation result can be generated using `plot_result()`.

```
>>> for (txt, indices) in zip(("complete", "index_1"), (None, [1])):
>>>     fig, ax = estimator.plot_result(
>>>         indices=indices,
>>>         figure_size=(4.5, 3.),
>>>         region_unit="ppm",
>>>         axes_left=0.03,
>>>         axes_right=0.97,
>>>         axes_top=0.98,
>>>         axes_bottom=0.09,
>>>     )
>>>     fig.savefig(f"tutorial_1d_{txt}_fig.pdf")
```

Below is the figure `tutorial_1d_complete_fig.pdf`:



Saving the estimator

The estimator object itself can be saved and reloaded for future use with the `to_pickle()` and `from_pickle()` methods, respectively:

```
>>> estimator.to_pickle("tutorial_1d")
Saved file tutorial_1d.pkl.
>>> # Load the estimator and save to the `estimator_cp` variable
>>> estimator_cp = ne.Estimator1D.from_pickle("tutorial_1d")
```

Saving a logfile

A logfile listing all the methods called on the estimator can be saved using `save_log()`:

```
>>> estimator.save_log("tutorial_1d")
Saved file tutorial_1d.log.
```

2.2 Using `Estimator2DJ`

The `nmrespy.Estimator2DJ` class enables the estimation of J-Resolved (2DJ) spectroscopy datasets. This facilitates use of **CUPID** (**C**omputer-assisted **U**ltrahigh-resolution **P**rotocol for **I**deal **D**ecoupling) which can be used to generate homodecoupled spectra and to predicting multiplet structures.

Many methods in this class have analogues in `Estimator1D`. You are advised to read through the *1D walkthrough* before continuing, as I will be providing minimal descriptions of things covered in that.

2.2.1 Generating an instance

Bruker data

Use `new_bruker()`. Unlike `Estimator1D`, you must import time-domain 2DJ data. The path should be set as `"<path_to_data>/<expno>/"`. There should be a `ser` file, an `acqus` file, and an `acqu2s` file directly under this directory. Again, phasing in the direct-dimension will be needed. If deemed necessary, baseline correction is possible too.

```
>>> import nmrespy as ne
>>> estimator = ne.Estimator2DJ.new_bruker("/home/simon/nmr_data/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 shifts and couplings
>>> shifts = [
>>>    6.005,
>>>    3.510,
>>>    3.934,
>>>    3.423,
>>>    4.554,
>>>    3.891,
>>>    4.287,
>>>    3.332,
>>>    1.908,
>>>    1.555,
>>>    0.644,
>>>    4.042,
>>>    4.517,
>>>    3.889,
>>>    4.635,
>>>    4.160,
>>>    4.021,
>>>    4.408,
>>>    0.311,
>>>    1.334,
>>>    0.893,
>>>    0.150,
>>> ]
>>> couplings = [
>>>     (1, 2, 2.285),
>>>     (2, 3, 4.657),
>>>     (2, 8, 4.828),
>>>     (3, 4, 4.326),
>>>     (4, 5, 4.851),
>>>     (5, 6, 5.440),
>>>     (5, 7, 2.288),
>>>     (6, 7, -6.210),
>>>     (7, 11, 7.256),
>>>     (12, 13, -4.005),
>>>     (12, 19, 1.460),
>>>     (14, 15, 4.253),
>>>     (15, 16, 4.448),
>>>     (15, 21, 3.221),
>>>     (16, 18, 4.733),
>>>     (17, 18, -4.182),
>>>     (18, 22, 1.350),
>>> ]
>>> estimator = ne.Estimator2DJ.new_spinach(
>>>     shifts=shifts,
>>>     couplings=couplings,
>>>     pts=(64, 4096),
>>>     sw=(30., 2200.),
>>>     offset=1000.,
>>>     field=300.,
>>>     field_unit="MHz",
>>>     snr=20.,
>>> )
```

Note: We will be using the estimator generated above for the rest of this tutorial. If you do not have access to MATLAB/Spinach, you can construct the estimator by using an FID I made earlier:

```
>>> import nmrspy as ne
>>> from pathlib import Path
>>> import pickle
>>> fid_path = Path(ne.__file__).expanduser().parents[1] \
...     / "samples/jres_sucrose_sythetic/sucrose_jres_fid.pkl"
>>> with open(fid_path, "rb") as fh:
...     fid = pickle.load(fh)
...
>>> expinfo = ne.ExpInfo(
...     dim=2,
...     sw=(30., 2200.),
...     offset=(0., 1000.),
...     sfo=(None, 300.),
...     nucleus=(None, "1H"),
...     default_pts=(64, 4096),
... )
>>> estimator = ne.Estimator2DJ(fid, expinfo)
```

2.2.2 Estimating the dataset

The procedure for estimating 2DJ data is very similar to that of 1D data. You need to specify regions in the direct dimension that are of interest for generating filtered sub-FIDs. No filtering is done in the indirect dimension. In our example, it turns out that for a couple of the regions selected, the number of oscillators automatically generated is slightly smaller that the "true" number, and so we force the optimiser to us the true number (see the lines involving `initial_guesses`).

```
>>> regions = (
...     (6.08, 5.91),
...     (4.72, 4.46),
...     (4.46, 4.22),
...     (4.22, 4.1),
...     (4.09, 3.98),
...     (3.98, 3.83),
...     (3.58, 3.28),
...     (2.08, 1.16),
...     (1.05, 0.0),
... )
>>> n_regions = len(regions)
>>> initial_guesses = n_regions * [None]
>>> initial_guesses[1:3] = [16, 16]
>>> # kwargs common to estimation of each region
>>> common_kwargs = {
...     "noise_region": (5.5, 5.33),
...     "region_unit": "ppm",
...     "max_iterations": 200,
...     "phase_variance": True,
... }
>>> for init_guess, region in zip(initial_guesses, regions):
...     kwargs = {**{"region": region, "initial_guess": init_guess}, **common_kwargs}
...     estimator.estimate(**kwargs)
```

(continues on next page)

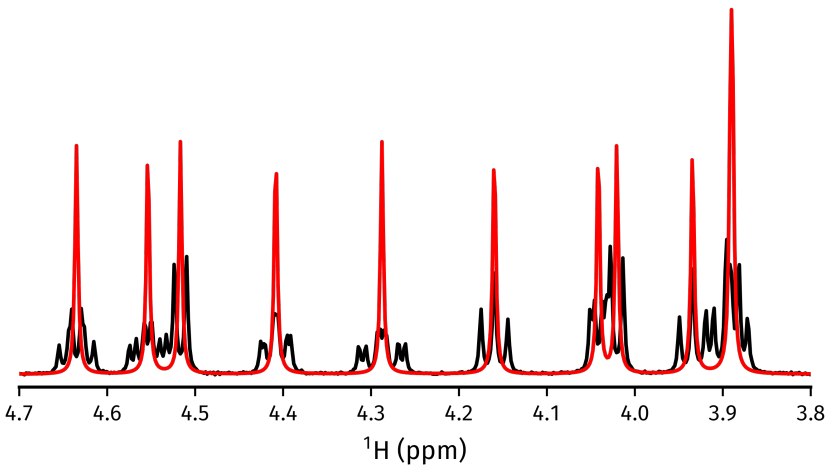
(continued from previous page)

```
>>> # It is a good idea to pickle the estimator after estimation
>>> estimator.to_pickle("sucrose")
```

2.2.3 Acquiring a homodecoupled spectrum

The `cupid_spectrum()` method produces a homodecoupled spectrum using the estimation parameters. In the code snippet below, a figure is made comparing the homodecoupled spectrum with the spectrum of the first direct-dimension slice in the 2DJ data, which is a normal 1D spectrum.

```
>>> # Normal 1D spectrum
>>> init_spectrum = estimator.spectrum_zero_t1.real
>>> # Homodecoupled spectrum produced using CUPID
>>> cupid_spectrum = estimator.cupid_spectrum().real
>>> # Get direct-dimension shifts
>>> shifts = estimator.get_shifts(unit="ppm", meshgrid=False)[-1]
>>> import matplotlib.pyplot as plt
>>> fig, ax = plt.subplots(figsize=(4.5, 2.5))
>>> ax.plot(shifts, init_spectrum, color="k")
>>> ax.plot(shifts, cupid_spectrum, color="r")
>>> # The most interesting region of the spectrum
>>> ax.set_xlim(4.7, 3.8)
>>> # ========================
>>> # These lines are just for plot aesthetics
>>> for x in ("top", "left", "right"):
>>>     ax.spines[x].set_visible(False)
>>> ax.set_xticks([4.7 - 0.1 * i for i in range(10)])
>>> ax.set_yticks([])
>>> ax.set_position([0.03, 0.175, 0.94, 0.83])
>>> ax.set_xlabel(f"{estimator.latex_nuclei[1]} (ppm)")
>>> # ========================
>>> fig.savefig("cupid_spectrum.png")
```



2.2.4 Multiplet prediction

Oscillators belonging to the same multiplet can be predicted based on the fact that in a 2DJ signal any pair should satisfy the following:

$$\left|\left(f_i^{(2)} - f_i^{(1)}\right) - \left(f_j^{(2)} - f_j^{(1)}\right)\right| < \epsilon$$

where ϵ is an error threshold. $f^{(1)}$ and $f^{(2)}$ are the indirect- and direct-dimension frequencies, respectively. The `predict_multiplets()` generates groups of oscillator indices satisfying the above criterion. A key parameter for this is `thold`, which sets the error threshold ϵ. By default, this is set to be $\max\left(f_{sw}^{(1)}/N^{(1)}, f_{sw}^{(2)}/N^{(2)}\right)$, 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 slightly larger threshold.

In the example below, multiplet groups are determined for regions with indices 1-5 (covering the region plotted above).

```
>>> indices = [1, 2, 3, 4, 5]
>>> multiplets = estimator.predict_multiplets(indices=indices)
>>> for (freq, idx) in multiplets.items():
...     print(f"{freq / estimator.sfo[1]:.4f}ppm: {idx}")
...
3.8890ppm: [1, 4]
3.8910ppm: [0, 2, 3, 5]
3.9344ppm: [6, 7, 8]
4.0205ppm: [9, 10]
4.0416ppm: [11, 12, 13, 14]
4.1598ppm: [15, 16, 17]
4.2876ppm: [18, 19, 20, 21, 22, 23, 24, 25]
4.4083ppm: [26, 27, 28, 29, 30, 31, 32, 33]
4.5167ppm: [34, 35]
4.5537ppm: [36, 37, 38, 39, 40, 41, 42, 43]
4.6349ppm: [44, 45, 46, 47, 48, 49]
```

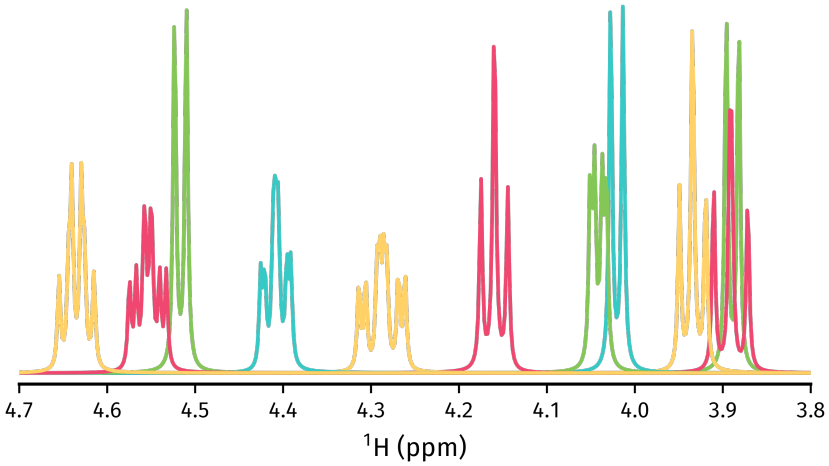
To generate FIDs corresponding to each multiplet structure, use the `construct_multiplet_fids()` method. In the following code snippet, each generated FID undergoes FT, with all the spectra being plotted.

```
>>> # Direct-dimension shifts
>>> shifts_f2 = estimator.get_shifts(unit="ppm", meshgrid=False)[-1]
>>> fids = estimator.construct_multiplet_fids(indices=indices)
>>> # Create an iterator which cycles through values infinitely
>>> from itertools import cycle
>>> colors = cycle(["#84c757", "#ef476f", "#ffd166", "#36c9c6"])
>>> fig, ax = plt.subplots(figsize=(4.5, 2.5))
>>> for fid in fids:
...     # Halve first point prior to FT to prevent vertical baseline shift
...     fid[0] *= 0.5
...     # FT and retrieve real component
...     spectrum = ne.sig.ft(fid).real
...     ax.plot(shifts_f2, spectrum, color=next(colors))
...
>>> ax.set_xlim(4.7, 3.8)
>>> # ========================
>>> # These lines are just for plot aesthetics
>>> for x in ("top", "left", "right"):
```

(continues on next page)

(continued from previous page)

```
...     ax.spines[x].set_visible(False)
...
>>> ax.set_xticks([4.7 - 0.1 * i for i in range(10)])
>>> ax.set_yticks([])
>>> ax.set_position([0.03, 0.175, 0.94, 0.83])
>>> ax.set_xlabel(f"{estimator.latex_nuclei[1]} (ppm)")
>>> # ========================
>>> fig.savefig("multiplets.png")
```



2.2.5 Generating tilted spectra

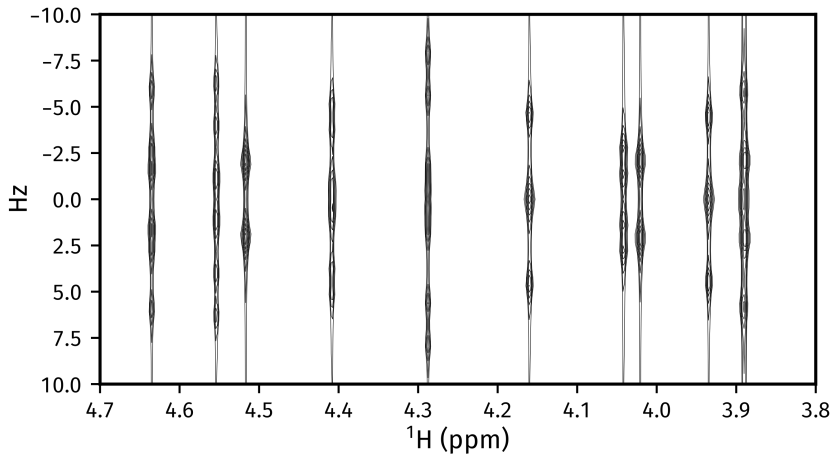
The well-known 45° "tilt" that is applied to 2DJ spectra for orthogonal separation of chemical shifts and scalar couplings effectively maps the frequencies in the direct dimension $f^{(2)}$ to $f^{(2)} - f^{(1)}$. Armed with an estimation result, a signal with these adjusted frequencies can easily be constructed. As well as this, generating a pair of phase- or amplitude-modulated FIDs enables the construction of absorption-mode spectra. Use `nmrespy.Estimator2DJ.sheared_signal()`, with `indirect_modulation` set to either `"amp"` or `"phase"` to generate the desired spectra. Then, you can 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(
```

(continues on next page)

(continued from previous page)

```
...     shifts_f2,
...     shifts_f1,
...     spectrum_2rr,
...     colors="k",
...     levels=levels,
...     linewidths=0.3,
... )
>>> ax.set_xlim(4.7, 3.8)
>>> ax.set_ylim(10., -10.)
>>> # ========================
>>> # These lines are just for plot aesthetics
>>> ax.set_xticks([4.7 - 0.1 * i for i in range(10)])
>>> ax.set_position([0.12, 0.175, 0.85, 0.79])
>>> ax.set_xlabel(f"{estimator.latex_nuclei[1]} (ppm)", labelpad=1)
>>> ax.set_ylabel("Hz", labelpad=1)
>>> # ========================
>>> fig.savefig("sheared_spectrum.png")
```



2.2.6 Plotting result figures

The `plot_result()` method enables the generation of a figure giving an overview of the estimation result. The figure comprises the following, from top to bottom:

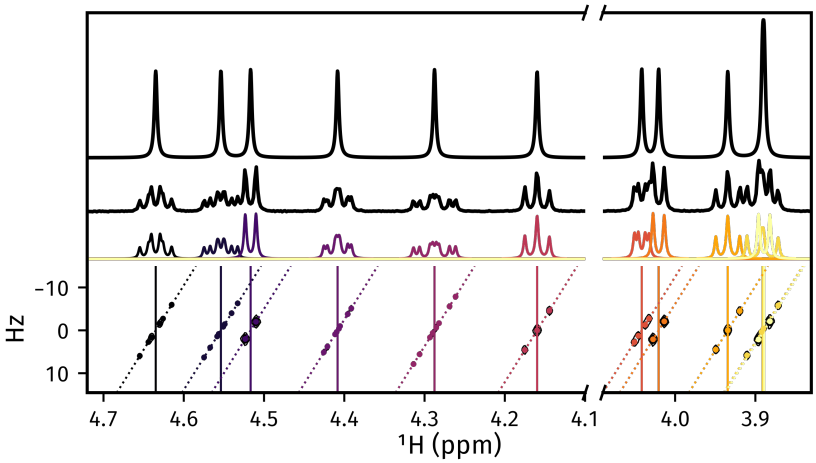
- The homodecoupled spectrum generated using `cupid_spectrum()`.
- The 1D spectrum corresponding to the 2DJ dataset.
- The multiplet structures predicted. Note that to get decent multiplet assignments, you may need to increase the value of the `multiplet_thold` argument manually.
- A contour plot of the 2DJ spectrum, with points indicating the positions of estimated peaks.

```
>>> fig, axs = estimator.plot_result(
...     indices=[1, 2, 3, 4, 5],
...     region_unit="ppm",
...     marker_size=5.,
...     figsize=(4.5, 2.5),
...     # Number of points to construct homodecoupled signal
```

(continues on next page)

(continued from previous page)

```
...     # and multiplet structures from
...     high_resolution_pts=16384,
...     # There is a lot of scope for editing the colours of
...     # multiplets. See the reference!
...     # Here I specify a recognised name of a colourmap in
...     # matplotlib.
...     multiplet_colors="inferno",
...     # 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 I do to get the spectrum from the FID is halve the initial point and FT (there is no apodisation).

Multiplets

To write individual multiplet structures to separate Bruker experiments, you can use `write_multiplets_to_bruker()`. It is a good idea to set a prefix for the experiment numbers, especially if the directory you are saving to already has data directories, so you can easily remember which of the directories correspond to multiplet structures. In the example below, as 22 multiplets were resolved, and `expno_prefix` was set to `99`, the directories `9901/`, `9902/`, ..., `9922/` are created.

```
>>> estimator.write_multiplets_to_bruker(
...     path=".",
...     expno_prefix=99,
...     pts=16384,
```

(continues on next page)

(continued from previous page)

```
...     force_overwrite=True,
... )
Saved multiplets to folders ./[9901-9922]/
```

CUPID data

To save the homodecoupled signal generated by our CUPID method, use `write_cupid_to_bruker()`:

```
>>> estimator.write_cupid_to_bruker(
...     path=".",
...     expno=1111,
...     pts=16384,
... )
Saved CUPID signal to ./1111/
```

2.2.8 Miscellaneous

For writing result tables to text and PDF files, saving estimators to binary files for later use, and saving log files, look at the relevant sections in the *Using Estimator1D* walkthrough. The process is identical.