Alkali Vapor-Cell Magnetometry and its Application to Low-Field Relaxometry and Diffusometry

by

Paul Joseph Ganssle

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Chemistry

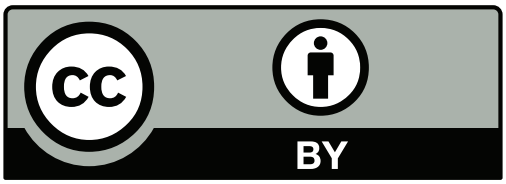
in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:
Professor Alexander Pines, Chair
Professor Dmitry Budker
Professor David Wemmer

Fall 2014

Alkali Vapor-Cell Magnetometry and its Application to Low-Field Relaxometry and Diffusometry



Abstract

Alkali Vapor-Cell Magnetometry and its Application to Low-Field Relaxometry and Diffusometry

by

Paul Joseph Ganssle
Doctor of Philosophy in Chemistry

University of California, Berkeley

Professor Alexander Pines, Chair

The availability of high-sensitivity magnetic sensors such as alkali vapor-cell magnetometers has fueled a growing body of literature on the development of inexpensive, portable and robust magnetic resonance systems which operate in ultra-low magnetic fields. This work covers the design, construction and operation of such a magnetometer, and its use as a high-sensitivity low-field detector of in nuclear magnetic resonance (NMR) experiments, including zero-field J-spectroscopy, relaxometry and diffusometry. In particular, we have attempted to demonstrate the utility of these devices for 1- and 2-dimensional relaxometry and diffusometry measurements, which are currently a significant commercial application of low-field NMR.

Dedicated to Tyren K. Barker

Contents

Acknowledgments

I owe my thanks to many people who have dedicated their time to my personal and professional development during the course of my graduate work. First of all, Alex Pines and all my colleagues in the Pines Lab during my time there, all of whom seem to have conspired to make it a pleasant, fun working and academic environment.

I tend to think through problems best by discussion, and so I'm sincerely grateful to the many people who have volunteered their time and energy as counterpoints, collaborators and mentors. This includes most everyone in the Pines lab during my time at Berkeley, and many members of Dmitry Budker's lab as well. In particular, Scott "Kremice" Seltzer[1] and Micah Ledbetter provided invaluable feedback on the design, operation and troubleshooting of alkali vapor-cell magnetometers, among many, many other topics. My early research in the Pines Lab on J-coupling and para-hydrogen induced polarization (mostly not covered in this document) was carried out alongside Thomas Theis and Gwendal Kervern, both of whom were integral to development as an NMR scientist; Mark Butler should also be added to this list, as I relied heavily on his expertise, analytical mind and supreme patience as I was learning the fundamental concepts of NMR and spin dynamics. To the list of intellectual mentors and advisors I should also add Chang Shin, who provided me advice on electronics and electrical engineering that shapes my fundamental approach to circuit design to this day.

I would also like to specifically thank Dmitry Budker for his always sage advice on many subjects both scientific and professional, he was always gracious with his time and attention and always provided extremely useful feedback at every stage of the process from initial design to publication. Similarly, I owe much to Vik Bajaj, who served as a day-to-day advisor and mentor in the Pines Lab and who significantly shaped my scientific and professional development during this time.

Finally, I would like to specifically acknowledge the contributions that Hyun "Doug" Shin made to this work during his undergraduate career at Berkeley. He was capable and dedicated, and a brilliant machinist — it took great restraint for me to not delegate all my machine work to Doug, as I knew that even though it would probably not help his career development much to spend so much time in the machine shop I also knew that he would probably do an even better job than I would if I gave him the task.

[1] While his given name is "Scott Jeffrey Seltzer", he prefers to be called Kremice, pronounced /krɛmɪs/. It is also appropriate to call him "K-Man", "Kremdog Millionaire" or "The Kremtastic Voyage".

Chapter 1

Introduction

The goal of this document is to provide a practical road map for the construction and use of alkali vapor-cell magnetometers for nuclear magnetic resonance (NMR), particularly measurements of relaxation and diffusion. While the story is told through the lens of my experience doing just that, it is intended to have somewhat more general applicability.

Chapter 2 serves as a brief introduction to the operational concepts in alkali-vapor-cell magnetometry. Chapter 3 goes into detail about the design decisions and trade offs involved in the construction of an atomic magnetometer for the purposes of making NMR measurements. Chapter 4 covers primarily the software used for controlling the experiments. Chapter 5 covers strategies for performing low-field NMR experiments that can be performed in our zero-field magnetometer, and Chapter 6 covers our relaxometry and diffusometry experiments specifically.

Chapter 2

Alkali Vapor-Cell Magnetometry

2.1 Overview

Alkali vapor-cell magnetometers are among the most sensitive magnetic sensors currently available, and in contrast to inductive coils, their sensitivity is mostly independent of the frequency of the signal[1], making them excellent sensors for low-field NMR and MRI applications. Vapor-cell magnetometers are relatively inexpensive, low-maintenance, low-power, robust to temperature, pressure and vibration, and can be microfabricated as monolithic systems; as such they are ideal candidates for next-generation high-sensitivity magnetic sensors for portable applications and as embedded in compact sensor arrays.[2,3]

Similar in principle of operation to proton precession magnetometers, alkali vapor-cell magnetometers measure magnetic field strength by observation of the resonance frequency of atomic spins.[4–6] Alkali spins are particularly suited to this task because they can be hyperpolarized by optical pumping[7,8], providing a significant enhancement in sensitivity. The precession frequency can also be measured purely optically as well, generally either by measurement of beam absorption[9] or by measurement of the Faraday rotation of an orthogonal beam (see Sec. 2.3).

2.2 Optical Pumping

Alkali vapor-cell magnetometers use the precession of an ensemble of polarized spins to measure magnetic fields, and that polarization is created by exciting an optical transition which is tied to the the atomic spin state. In the case of our device, circularly polarized 795 nm light is used to excite the D1 transition from rubidium's $5^2S_{1/2}$ state to a $5^2S_{1/2}$ excited state (the D1 transition). The electronic states are further split by their total angular momentum, $F = I + J$, where I is the nuclear spin and J is the electron spin. As both $5^2S_{1/2}$ and $5^2P_{1/2}$ are singlet states, in both cases the spin quantum number can only take values $m_f = \pm 1/2$. As circularly polarized light carries one unit of spin angular momentum, only transitions where $\Delta F = 1$ are allowed, and so spins are only pumped from the $m_f = -1/2$

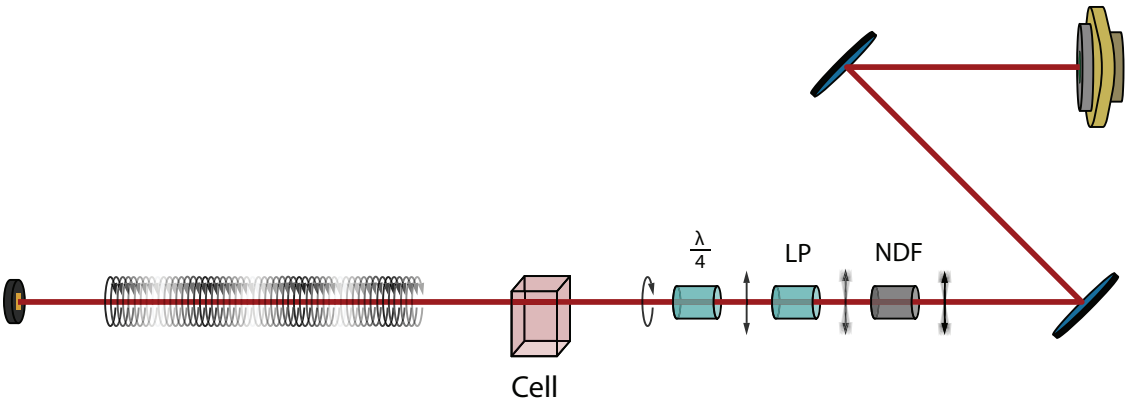


Figure 2.1: The optical path of a single-beam magnetometer.

ground state into the $m_f = 1/2$ excited state. Since relaxation back to the ground state is not biased by spin state, relaxing spins are approximately equally likely to transition to either ground state; because spins already aligned along the pump beam ($m_F = 1/2$) are unaffected by pumping, this eventually leads to a buildup of polarization in the $m_F = 1/2$ state dependent upon the relaxation time of the system and the rate of optical pumping. [10,11]

An diagram of an optical pumping system is detailed in Fig. 2.1. The optical pumping rate is adjusted by changing the beam intensity - in this case a neutral density filter is used, but other methods such as crossed linear polarizers can be used for a finer adjustment. A linear polarizer is used to clean up the polarization of the beam, and then the light is passed through a quarter-waveplate with its fast-axis aligned 45° off the axis of linear polarization to produce a circularly polarized beam, which is used to pump the cell. If the polarized spins are in a magnetic field, their state will begin precessing according to $\cos\omega t \left|+\frac{1}{2}\right\rangle + \sin\omega t \left|-\frac{1}{2}\right\rangle$. Since the rate of absorption of the photons is a function of the fraction of the ensemble that is in the $\left|-\frac{1}{2}\right\rangle$ state, the laser intensity after the cell will be a function of the pumping rate (constant), the spin absorption cross-section (constant) and the precession frequency (a linear function of magnetic field); by measuring the absorption of the pump beam, it is thus possible to operate a magnetometer in a single-beam configuration. While two-beam configurations are often more sensitive, this is the simplest and likely least expensive option. [12,13]

2.3 Pump-Probe Configuration

A common way to read out the spin precession rate (and thus the magnetic field) experienced by the alkali metal atoms is to make a measurement of the Faraday rotation of off-resonant, linearly polarized light passed through the cell. When the atomic spins are polarized, alkali metal vapor is a birefringent material, with different indexes of refraction n_R and n_L for the left and right polarized states of light $|R\rangle$ and $|L\rangle$. These indices of refraction are proportional to the population of the spin ground states $\rho\left(+\frac{1}{2}_y\right)$ and $\rho(-\frac{1}{2}_y)$ in the frame of reference aligned along the direction of propagation of the probe beam (y). For the effect of the D1

transition, these indices are described by Eqn. 2.1 [10]

$$n_{\pm} = 1 + 2\rho\left(\mp\frac{1}{2_y}\right)\left(\frac{nr_e c^2 f_{D1}}{4\nu}\right) Im\left[V(\nu-\nu_{D1})\right] \tag{2.1}$$

The rotation of linearly polarized light will occur because linearly polarized light aligned along a given direction can be decomposed into an equal superposition of left and right circularly polarized light, with the angle of polarization given by a phase difference between the two:

$$|\Theta\rangle = \frac{1}{\sqrt{2}}\left[e^{-i\Theta}\,|\circlearrowleft\rangle + e^{i\Theta}\,|\circlearrowright\rangle\right] \tag{2.2}$$

Thus for a non-zero population difference $\rho\left(+\frac{1}{2_y}\right) \neq \rho\left(-\frac{1}{2_y}\right)$, the differential retardation of each component introduces some additional phase between the two circularly polarized components, tipping the polarization of the probe beam by an angle θ, given by Eqn 2.3[1]:

$$\theta = \frac{\pi}{2}\ln(r_e c)\left[\rho\left(+\frac{1}{2_y}\right) - \rho\left(-\frac{1}{2_y}\right)\right](-f_{D1} Im[V(\nu-\nu_{D1})]) \tag{2.3}$$

For a pump beam along an orthogonal direction (x), in the presence of no precession, there is no difference in polarization between the ground states along y, as the spin states polarized along x can be decomposed into an equal linear superposition of spins polarized along y. An average rotation about the z axis of the ensemble by angle $\tilde{\theta} = \gamma_S B_z R_P$ induced by a magnetic field tips the spins towards the y axis, with populations given by Eqn 2.4:

$$|\tilde{\theta}\rangle = \left(cos\tilde{\theta} + sin\tilde{\theta}\right)\left|+\frac{1}{2_y}\right\rangle + \left(cos\tilde{\theta} - sin\tilde{\theta}\right)|-\frac{1}{2_y}\rangle \tag{2.4}$$

Thus the population difference $P_y = \rho(+\frac{1}{2_y}) - \rho(-\frac{1}{2_y})$ is:

$$P_y = (\cos\tilde{\theta} + \sin\tilde{\theta}) - (\cos\tilde{\theta} - \sin\tilde{\theta}) \tag{2.5}$$

$$= 2\sin\tilde{\theta} \tag{2.6}$$

And for small angles, $2\sin\tilde{\theta} \approx 2\tilde{\theta}$. Finally, taking this result and inserting it into Eqn 2.3, the Faraday rotation angle for a given average tip angle is:

$$\theta = \frac{\pi}{2}\ln r_e c(2\sin\tilde{\theta})\,(-f_{D1} Im[V(\nu-\nu_{D1})]) \tag{2.7}$$

$$\approx \frac{\pi}{2}\ln r_e c(2\tilde{\theta})\,(-f_{D1} Im[V(\nu-\nu_{D1}]) \tag{2.8}$$

Which, for small angle and constant frequency is linearly proportional to $\tilde{\theta}$.

[1] One thing to note is that this takes into account only the effect of the D1 transitions. For a more in-depth analysis of the effect which takes into account the D2 transition (relevant in potassium magnetometers), see *Seltzer, 2008, pp 32-37* [10].

2.3.1 Balanced Polarimeter

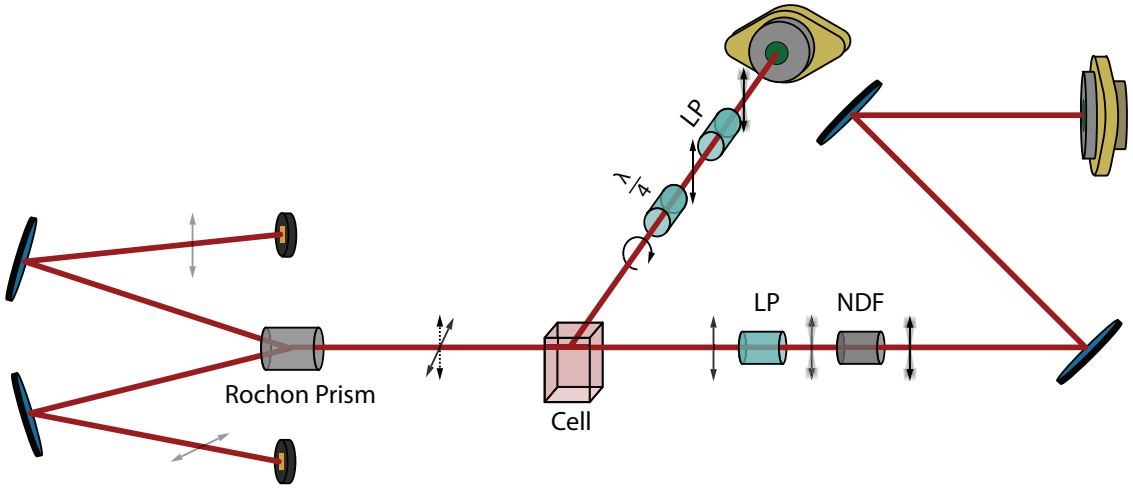


Figure 2.2: The optical path of a pump-probe magnetometer with a balanced polarimeter used for signal detection.

A balanced polarimeter is a simple device for measuring linear polarization, and is the method of choice for our magnetometer due to its high sensitivity. The beam path is shown in Figure 2.2. The beam is first reflected off two dielectric mirrors to provide optimal beam angle and position control, followed by a neutral density filter (or crossed-polarizer), which is used to attenuate the beam to the appropriate value. A linear polarizer with its fast axis aligned with the vertical is used to provide clean, uniform vertical polarization, and then the beam passes through the cell, where the light undergoes a Faraday rotation. Finally, the beam is passed through a Rochon prism whose fast axis is aligned 45°off vertical, which splits the beam into its diagonal and anti-diagonal components. The beams will have intensities: [10]

$$P_D = P_0 \sin^2\left(\frac{\pi}{4} - \theta\right) \tag{2.9}$$

$$P_A = P_0 \cos^2\left(\frac{\pi}{4} - \theta\right) \tag{2.10}$$

Where θ is the angle of polarization off the vertical axis. By subtracting the current in the diagonal channel from the current in the antidiagonal channel, the result is:

$$P_D - P_A = P_0 \sin^2(\theta - \frac{\pi}{4}) - P_0 \cos^2(\theta - \frac{\pi}{4}) \tag{2.11}$$

$$= \frac{P_0\left[\right]1 + \cos\left(2\theta - \frac{\pi}{2}\right)\right]}{2} - \frac{P_0\left[1 - \cos\left(2\theta - \frac{\pi}{2}\right)\right]}{2} \tag{2.12}$$

$$= P_0 \sin\left(2\theta\right) \tag{2.13}$$

And if 2θ satisfies the small angle approximation then $\sin(2\theta) \approx 2\theta$. This gives us a signal given by:

$$P_D - P_A \approx 2P_0\theta \tag{2.14}$$

Since this applies only for small angle θ, it is necessary for the initial input polarization to be "balanced", which is to say the pre-rotation beam should be exactly 45°off the fast axis of the Rochon prism. Because rotating a Rochon prism rotates both output beams, it is often most convenient to simply rotate the linear polarizer used to clean up the beam, as the rotations are small enough that this is unlikely to cause a significant dip in the output intensity.

2.3.1.1 Practical Concerns

The operation of the balanced polarimeter is characterized by a great deal of symmetry - because it measures very small angles, what's being measured is relatively small differences in two channels with comparatively high absolute signal - the average light incident on each channel is likely to be roughly an order of magnitude larger than the range of linear response. As a result, great care should be taken to avoid asymmetric noise in the detection section between the two channels.

One such source of noise is clipping - the likely source of clipping noise is angular noise at the beam source, translated into noise on the beam position later, and as such the magnitude of the clipping noise will grow as the length of the beam path grows. As such, while clipping on an iris at the laser source may be acceptable, clipping later in the beam path can easily become a limiting source of noise. This is particularly problematic if the clipping occurs *after* the beam splitting optics, as this will cause largely independent and asymmetric noise on the absolute intensity of each beam. Ideally, the length of the beam path after the beams have been split will be kept as short as possible, and the two arms of the beam preferably kept at equal length.

Another practical issue with balanced polarimeter operation is the possibility of asymmetric response between the two detection channels, which can occur due to any number of asymmetries in the two beam paths - differences in the load resistor, diode response, bias offset voltage, etc. The likely problem with channel asymmetries is that it can lead to a "false balance", wherein zero output signal is achieved at a non-zero angle θ_e. To first order, the primary effect of a "false balance" is that the polarimeter will be operating in a non-linear region. Additional problems can arise if the beams are being operated near the linearity limit of the photodiodes - since a stronger signal is required in one of the two channels, in order to keep both channels operating in the linear region, the beam will then need to be attenuated further than necessary, reducing signal.

There are two main causes of false balance - difference in the response of the electronics and differences in response from the photodiodes themselves. Starting with the electronics, a very likely culprit for inducing false balance will be an imbalance in the load resistances. For load resistors R_{L1} and R_{L2} and deviation ϵ_R defined such that $R_{L1} = R_L + \sigma_R$ and

$R_{L2} = R_L - \sigma_R$[2], the angle θ_e which makes the channels appear balanced is given by Eqn 2.15:

$$\theta_e = \frac{1}{2}\text{asin}\left(\frac{\sigma R}{R_L}\right) \tag{2.15}$$

For the common "gold band" type resistor with tolerance ± 5%, assuming a typical quadrature addition of errors, a typical σ_R/R_L will be $0.05\sqrt{2}$, which gives $\theta_e \approx 35.4\,\text{mrad}$. This particular problem is easy to solve by placing trim-potentiometers in series with the load resistors and actively balancing the resistor channels.

Another potential cause of false balance is differences in the response of each channel's photo-diodes. While the inherent response curves of each photodiode is not necessarily something that anything can be done about, adding in symmetry between the channels and making sure the signal from each channel is individually maximized can go some way towards alleviating this problem. Depending on the photodiodes, there may be some responsivity gradient across the sensitive area, and that will certainly be true if any clipping is occurring.

In our setup, the response of our DET110 photodiodes seem somewhat sensitive to beam width and position, and so to start we try to keep beam path lengths as close to identical as possible to account for deviations from well-collimated beams (often when using smaller photodiodes, it may also be advantageous to put a lens on the input of the beam-splitting optics to focus near to the photodiode - in this case, beam path length may be a critical factor). The beams are positioned using the output signal of each individual channel - the beam is roughly centered by eye on the photodiode, and then the optimal position is found by iteratively maximizing the signal from each channel with respect to horizontal and vertical position. In this instrument, the angle of incidence is set such that the beams are roughly normal to the sensitive surface (this is estimated by eye, but it's not impossible to use some sort of signal-maximizing feedback to find this angle more precisely).

2.3.1.2 Balancing Procedure

With issues of false balance hopefully largely accounted for, finding the right balance is largely a matter of adjusting the input polarization of the light. It's best to do this while changing the minimum number of other factors which could affect the polarization, such as the length of the beam path or temperatures or positions of the optics. If the pump beam is operating, the polarization of the light will be sensitive to the magnetic field at the cell, and so the polarimeter is always balanced with the pump beam blocked.

The polarimeter can, in theory, be balanced either by adjusting the beam-splitting optics until they are exactly 45 ° off axis from the input polarization, but in many (if not all) cases this will affect the path of the output beam and not just the polarization of at least one of the two channels, and so all optics after the beamsplitter would need to be adjusted. Usually it is preferable to adjust the linear polarizer which comes before the cell; the downside of this

[2]This is always the case when R_L is the average of R_{L1} and R_{L2} and σ_R is the "standard deviation" of the two values.

method is that the output signal amplitude is a function of the beam rotation - because these rotations are often quite small it is not usually a problem, but it is worth keeping an eye on the power of the post-polarizer beam, and if it becomes too attenuated a half-waveplate can be added before the linear polarizer to tip the polarization into the right alignment. [14]

Another concern is possible birefringence introduced by the cell glass - which is likely under some strain due to the omnipresent thermal gradients. The birefringent properties of the glass will likely change both the angle of linear polarization and the ellipticity of the beam. This can be represented on the Poincarë sphere as a a rotation of an arbitrary angle (θ_B) about an arbitrary axis ($\hat{\mathbf{u}}$). Using a combination of a linear polarizer (or half-waveplate, depending on the severity of the birefringence) and a quarter-waveplate, the input polarization can be set such that the birefringence of the glass will make the light linear and aligned properly.

One thing to note is that in a balanced polarimeter, probe beam ellipticity should have no biasing effect on the signal, as elliptical components can be decomposed as an equal superposition of horizontal and vertical states (and thus subtract out). Beam ellipticity in the beam leaving the cell will thus only affect the overall signal intensity, and this will likely not be a significant problem for small deviations from linear. However, it can have a significant effect when beam *entering* the alkali vapor is elliptical, as these components will tend to pump the spins and slightly change the vector properties of the magnetometer response (including adding a phantom bias "field" along the probe direction). As such, when counter-acting beam ellipticity induced by glass birefringence, the magnetometer response is used for feedback, rather than a measure of the beam balance.

There are two main methods for accounting for beam ellipticity. The primary method looks at the magnetometer signal with the pump beam blocked - some fraction of the spins become polarized along the direction of the probe beam (y), and thus the magnetometer becomes sensitive to fields along x and z. In this method, an oscillating magnetic field is induced[3] and the quarter waveplate and linear polarizer are iteratively adjusted until the oscillations are no longer detectable and the DC level is balanced.

The other method for accounting for beam ellipticity is to look at the lineshape of the magnetometer response. A slow ($\approx$ 1-3 Hz) triangle wave is applied with an amplitude much greater than the magnetometer's region of linear response - this should result in a Lorentzian lineshape in the magnetometer response. This lineshape is affected by many factors - making it a somewhat less specific optimization parameter, but still possibly more sensitive - and an elliptical probe beam will induce asymmetry between the Lorentzian lobes.

2.3.2 Photoelastic Modulator-based Polarimeter

Another method for measuring the probe beam polarization uses a photo-elastic modulator (PEM)-based polarimeter, wherein the input light's ellipticity is modulated by the PEM and the linear component at the characteristic frequency is measured (Figure 2.3). As the

[3] Usually this field much larger than the test signal - our typical test signal is 62.7 pT_{RMS} for sensitivity testing and 6.27 or 62.7 nT_{RMS} for detection of ellipticity

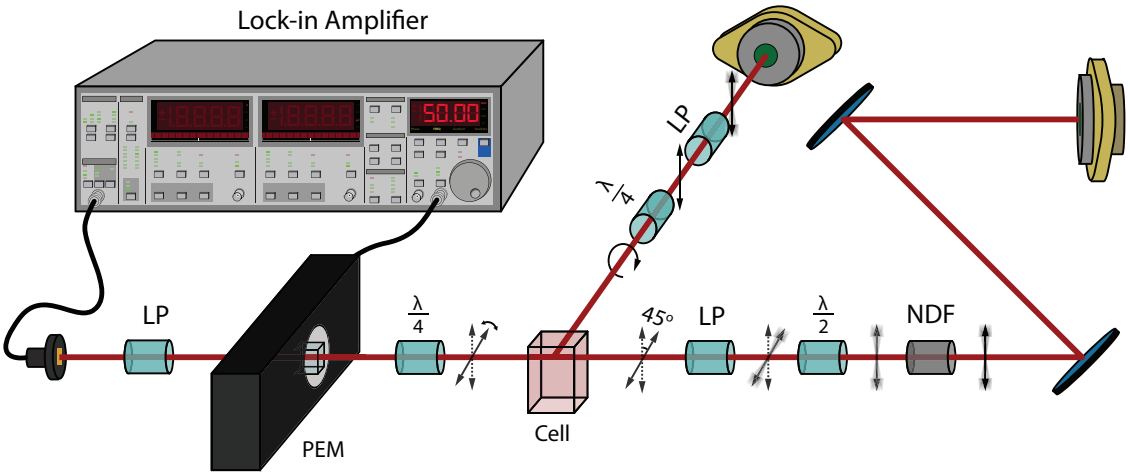


Figure 2.3: The optical path of a pump-probe magnetometer with a PEM-based polarimeter.

measurement is made at higher frequency, this has the potential advantage of avoiding some forms of low-frequency noise that may be present in other methods of polarimetry.

A Poincaré sphere representation of the beam polarization during the experiment is shown in Figure 2.4. The probe beam light passes through two mirrors to give maximum control over the positioning of the light, and as before, passes through a neutral density filter (NDF), which is used to set the optimal beam attenuation. Next a half-waveplate with its fast axis set at 22.5 ° off vertical is used to rotate the light to 45 ° off vertical, and the beam is then cleaned up with a linear polarizer set 45 ° off vertical (Figure 2.4b). Note that because this polarizer is present, if the half-waveplate is not present, the only consequence is that the beam will be further attenuated by a factor of approximately 2.

Next, the 45 ° linearly polarized light passes through the cell and undergoes a small-angle Faraday rotation (Figure 2.4c), after which it is passed through a quarter-waveplate with its fast axis aligned 45 ° off vertical (aligned with un-rotated light) — this tips the polarization into the $|D\rangle$-$|R\rangle$ plane (Figure 2.4d), thus translating linear rotation into ellipticity. After this, the light passes through the PEM, which modulates the ellipticity of the beam at the characteristic reference frequency (Figure 2.4e). Finally, the beam passes through a linear polarizer aligned 45 ° off vertical, selecting a projection of the light onto the $|D\rangle$ axis, which is then fed into a photodiode signal, the intensity of which is given by

$$\sin^2\left(\delta\theta + \beta\sin(\omega t)\right), \tag{2.16}$$

where $\delta\theta$ is the angle of the Faraday rotation and β and ω are the amplitude and frequency of the PEM modulation, respectively. The signal is fed into a lock-in amplifier synchronized with the PEM, and so the output of the lock-in amplifier, for small angles, is linearly proportional to $\delta\theta$, and insensitive to noise not resonant with the frequency ω (in our case, the frequency used was ≈50 kHz).

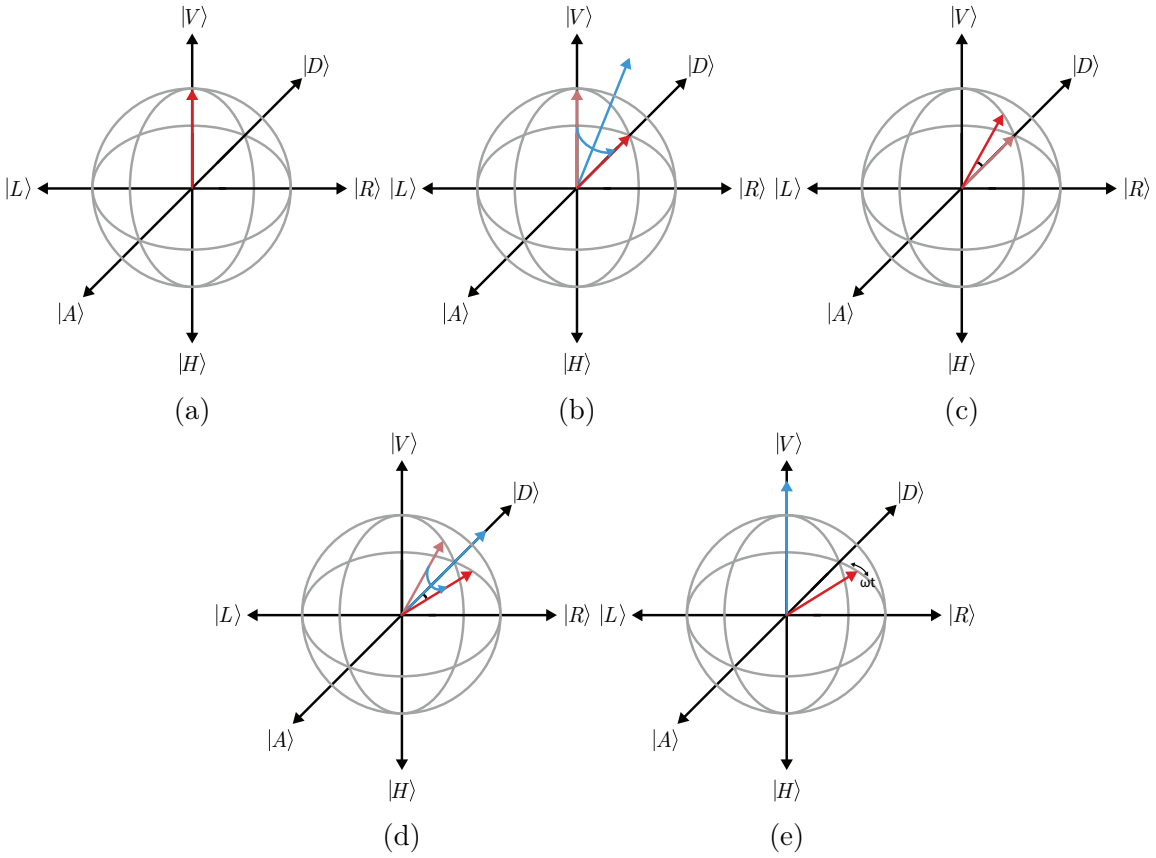


Figure 2.4: A Poincaré sphere representation of the polarization of the laser beam throughout the experiment. Light is initially linearly polarized vertical (2.4a), the half-waveplate tips it 45°to lie along the diagonal (2.4b). Passing through the cell causes the light to tip linearly some small angle (2.4c). Passing the light through a quarter-waveplate with fast axis aligned with the diagonal, the light is tipped into the $|D\rangle$-$|R\rangle$ plane (2.4d). Finally the PEM modulates the ellipticity of the beam at frequency ω (2.4e), and for small Faraday rotations θ, the amplitude second harmonic of the projection onto the anti-diagonal should be linearly proportional to θ.

Chapter 3

Magnetometer Design

3.1 Overview

Our magnetometer was designed as a versatile detector for low- and zero-field NMR experiments on relatively small samples. During the course of of the design, two types of cell were used, microfabricated cells provided by the Atomic Devices and Instrumentation group in NIST's Time and Frequency Division, and larger "cuvette"-style cells purchased frAlso,om Twinleaf Precision Sensors. The magnetometer was configured in a pump-probe configuration (see Sec. 2.3) tuned to the D1 resonance of ^{87}Rb, operating at zero field in a spin-exchange-relaxation-free (SERF) configuration[15]. The cells were heated using resistive heating coils wound on ceramic substrates.

The magnetometer used 4 layers of μ-metal shielding using progressive gaps[16] to achieve a total shielding factor of 10^{-6}. The remaining magnetic field after shielding was removed using a set of three-axis electromagnetic shim coils, the design of which is described in Section 3.4; no gradient shim coils were found to be necessary.

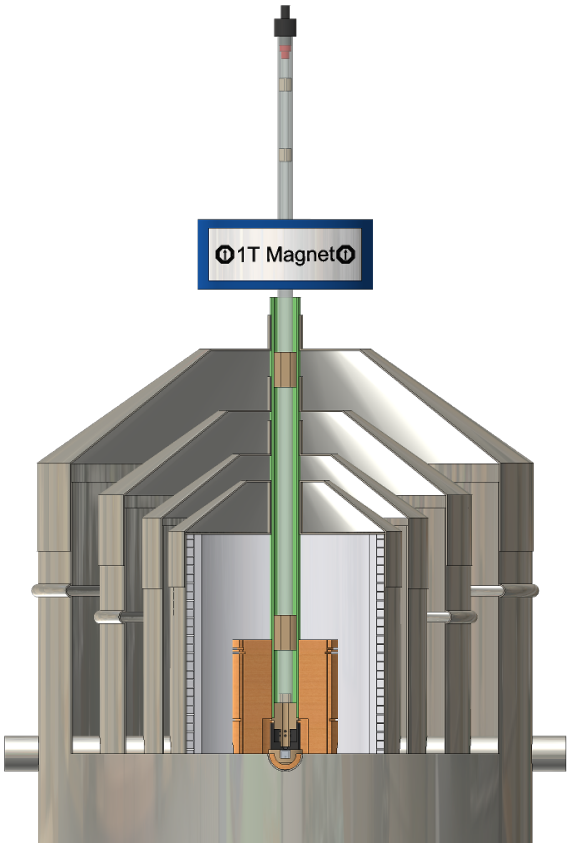


Figure 3.1: A render of our magnetometer.

Samples were contained in standard 5 mm NMR tubes widely used in high field experiments. The sample spins were pre-polarized[1] by pneumatically shuttling the NMR tube between a 1.8 T magnet and an air-cooled detection region. NMR pulses and gradients were provided

[1] More on sample polarization in Sec. 5.2

by coils wound on an 3D-printed substrate. surrounding the cell and detector.

3.2 Pneumatic Shuttling

There are three main ways for a sample to be polarized in a bias field other than the bias field in which the signal is detected. Either a temporary field can be generated during the pre-polarization stage (using an electromagnetic coil or otherwise), a permanent magnet can be temporarily placed in proximity to the sample, or the sample can be temporarily placed in proximity of the permanent magnet. In shielded magnetometers, the application of strong magnetic fields can cause the shields to become magnetized, and as such, the primary method of pre-polarization is by sample shuttling.

In previous magnetometry-based NMR applications, liquid samples have been transferred between polarization and detection regions by pressurized liquid flow between chambers in a cell, driven either by back pressure of gas or using syringe pumps. In these experiments, however, a pneumatic shuttling system has been devised to transfer samples in commercially-available NMR tubes. This has several advantages: it is easier to switch between samples, it can be much faster, it works with non-liquid or extremely viscous samples and often it is much less expensive. The primary disadvantage of the pneumatic sample system is that it is necessary to dissipate much more kinetic energy and design problems can lead to destruction of NMR tubes and — worse yet — magnetometer cells.

For lack of a better word, the sample containment and transfer apparatus is herein called a "probe", and the sample reception region is called the "probe head", by analogy to the high field probe apparatus. A rendering of the probe components can be found in Fig 3.2. The outer casing of the probe is $\frac{1}{16}$ " 1 " OD G-10 fiberglass, so chosen for its ability to withstand high temperatures. This should be concentric with the pulse coil substrate (see Sec. 3.3.4) and loose-fitting in the shield access ports (in practice, shield access ports often are not perfectly concentric and parallel tubes, and as such it would be impossible to thread a snug-fitting tube through them). The sample transfer tube is a length of 9 mm ID 1 mm thickness pyrex glass tubing kept concentric in the outer casing with PEEK tube adapters (see Fig. 3.3). These spacers are placed ≈ 0.5 " (1.27 cm) from the bottom of the glass tube and the top of the outer casing, while the glass tube extends beyond the outer casing, long enough that the bottom of a standard 7 " NMR tube sits just below the center of the pre-polarizing magnet when the sample is in the pre-polarization

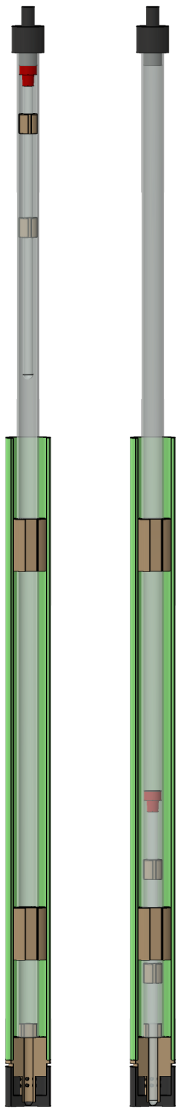


Figure 3.2: The sample probe - the sample is pneumatically shuttled between the pre-polarizing region and the detection region.

position. Between the two spacers is a solenoid wound of 30 AWG enamel-coated copper wire, which is used to ensure adiabatic sample transfer (see Sec. 3.2.1 below).

3.2.1 Adiabatic Transfer

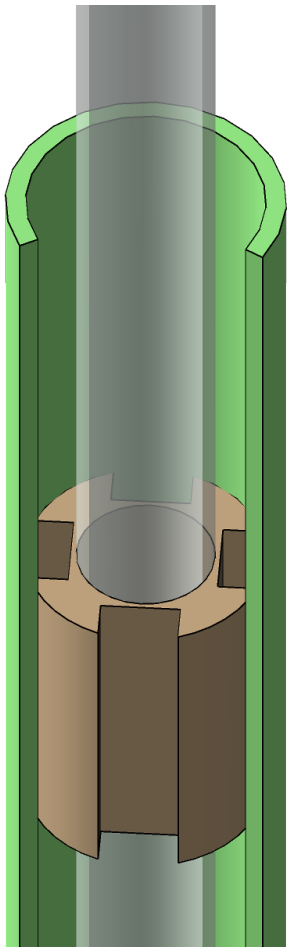


Figure 3.3: Tube diameter adapters used in the probe. The grooves are deep enough to accommodate $\frac{1}{8}$" tubing used to carry cool nitrogen to the probe head.

In general, the density matrix of a quantum mechanical ensemble will precess transverse to the direction of the Hamiltonian in the Louisville space representing that Hamiltonian's eigenbasis, with a frequency given by the energy level splitting between the Hamiltonian's eigenstates. This is a more general statement of the principle underlying spin precession transverse to a magnetic field - wherein the eigenstates are aligned parallel and antiparallel with the magnetic field and so the eigenbasis corresponds (to first order) to a physical frame of reference.

When such an ensemble encounters a time-dependent Hamiltonian, the resulting state depends on the rate of the rate of change of the Hamiltonian and the strength of the energy level splitting - when the the rate of change of the Hamiltonian is slow compared to the energy level splittings, the state change is said to occur *adiabatically*, whereas when the Hamiltonian changes quickly with respect to the energy level splittings, the state is said to change *non-adiabatically*. In the non-adiabatic condition (which occurs in NMR experiments when applying pulses), the state of the system is unchanged in response to a sudden change in the Hamiltonian, and if it is not a stationary state of the new Hamiltonian, it will begin to precess in the new Louisville space. In the adiabatic condition, however, precession about the Hamiltonian during periods of incremental change average out over the course of the change in the Hamiltonian, and so the density matrix is "dragged" along with the Hamiltonian, maintaining its relative angle in the Louisville space. As a result, stationary states of the initial Hamiltonian will be transformed into stationary states of the final Hamiltonian, and states transverse to the initial Hamiltonian will continue to precess about the final Hamiltonian.

In many cases, the polarization field will not exactly align with either the sensitive direction of the magnetometer or the z direction of the pulsed bias coils. Additionally, as spins are transferred from the polarization to detection region, if they pass through regions in which there is a large gradient (relative to the size of the offset bias field in that location) of fields transverse to the direction of polarization, the spins could decohere during the the sample transfer, killing the signal. Both of these problems are fixed by applying a bias offset field leading from the prepolarization field to the sample detection

region which is aligned with the sensitive direction of the magnetometer - this ensures that the Hamiltonian smoothly (and adiabatically) transitions between the polarization and detection regions, and hopefully truncates any small gradients along the way.

In this experimental design, this leading field is provided by a solenoid wound about the inner, glass "transfer" tube, and ends several inches above the detection region. The largest practical worry about this is that it might magnetize the shields if the coils are too close and too strong. However, often times the field needs to be *much* stronger directly below the prepolarizing magnet than along the remainder of the path, as the field is shifting rapidly from one on the order of a few Tesla to one on the order of a few Gauss. As a result, in some experiments it is desirable to have both a full leading solenoid which might operate at a few Gauss as well as a larger solenoid positioned further away from the shields but closer to the fringe field of the magnet, which might generate a field of tens to hundreds of Gauss. We have found that for our 2 T setup, this does not contribute significantly to the adiabatic transfer and it has been removed, but the necessity of this step is highly dependent on transfer speed and setup geometry and so no general recommendation either way can be made.

To ensure that, in the end, the field is at least aligned exactly with the bias field coils (which are geometrically constrained to be orthogonal to the x and y directions), the bias field coil is also energized during the transfer. This also helps to ensure the homogeneity of the field as the sample travels through the fringe of the solenoid - which is several inches from the cell, both for practical reasons and to reduce magnetic field noise associated with the leading coil.

3.2.2 Probe Head Design

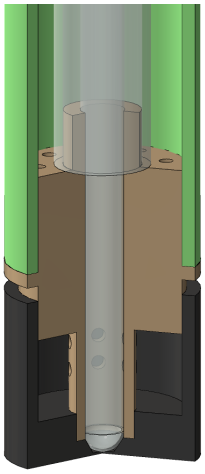


Figure 3.4: The probe head is designed for thermal insulation and minimization of physical shock.

The probe head (Fig. 3.4) has two primary design goals: to thermally insulate the sample from the cell and to protect the sample and the cell from a high-speed collision during sample shuttling. Thermal insulation is necessary because the cell runs at an elevated temperature, a side effect of which is that the inside of the magnetometer reaches thermal equilibrium at some significantly elevated temperature (on the order of 60-100 °C depending on cell temperature, air flow, etc), which could have a deleterious effect on the sample, or otherwise interfere with desired measurements. The probe head thermally isolates the samples both to minimize the impact of the elevated temperature in the detection and to allow for the use of cool, dry nitrogen flow to cool the sample without causing undue temperature gradients very close to the cell.

The probe head is divided into two parts - the probe head cap and the main body, both PEEK pieces which are designed to "slip-fit" together sufficiently tightly that an airtight seal is formed. Because both pneumatic shuttling and sample cooling (more on which in Sec 3.2.3) cause

gusts of wind which introduce significant noise into the system, it is important to ensure that the probe head is well-sealed from the detection region. Generally slip-fit designs can form seals sufficient to this goal because the high-temperature operating environment cause thermal expansion of the components, forming tighter seals than might be possible with room-temperature tolerances. In lower-temperature magnetometers, such as Cs-vapor cell magnetometers, it might be advisable to switch to a system using O-rings or rubber gaskets to prevent leaks into the detection region.

The probe head cap is 0.75 " (1.905 cm) long with an outer diameter set to just below the outer diameter of the outer G-10 probe casing and an inner diameter set such that it will slip-fit over the probe head body. A square-bottomed hole is milled or lathed out of the center to a depth sufficient to attach solidly to the probe head body while still leaving room for a hollow chamber through which air can flow over the sample for cooling, while also leaving enough thickness at the bottom of the cap to insulate the probe head well from the cell heater. Generally 0.25 " (0.635 cm) is chosen for this value. Greater thickness along the bottom of the cap allows for greater thermal insulation and ability to absorb kinetic energy from dropped samples, but may contribute to a larger thermal gradient across the sample - which can give rise to convection.

Figure 3.5: An older version of the probe head, with a flat-bottomed cap.

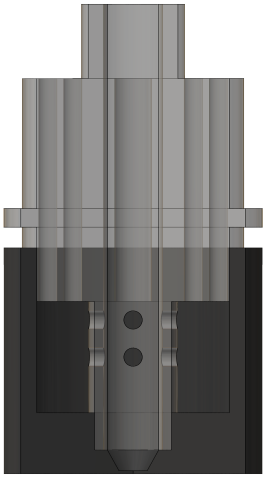


Figure 3.6: A render of the new probe head cap, with a tapered cap bottom.

A smaller hole is then milled out of the center of this cap with a diameter sufficient to accommodate the sample transfer tube extruding from the bottom of the probe head body, and milled sufficiently thin that the bottom of the sample sits no more than $\approx$ 100-500 μm from the bottom of the cap. Because the magnetic field generated by the sample dies off proportional to $1/r^3$ (where r is the distance between the sample and the sensitive magnetometer volume), it is extremely desirable to minimize the thickness of this landing spot as much as possible without significantly compromising the thermal insulation or the physical protection afforded by the probe head.

In an earlier version of the probe head cap, this landing spot was a 100-200 μm-thick flat panel, which eventually lead to the destruction of at least two magnetometer cells. Although the probe head does not sit directly on the cells (it sits several hundred μm above, on small PEEK spacers), the flat-bottomed design does not provide sufficient strength to absorb the kinetic energy of a dropped sample without deformation. In the earliest versions of the probe head cap, samples were able to cause this panel to break off (and the sample to crash through, thus destroying the cell). In later versions, the sample caused this flat region to "bow out", allowing the sample to come into contact with the cell. In the current version, however, this

problem is solved by using a ball-end mill to match the contours of this "landing spot" with the contours of an NMR sample tube. This adds strength and thermal insulation to the cap without compromising on distance, as the sample can only sit such that the bottom part is touching the cap, and so by keeping the nadir of the landing region at 100-200 μm thick and following the contours of the NMR sample tube, the total distance is unchanged.

The probe head body is designed to adapt between the glass sample transfer tube, the G-10 casing and the probe head cap, and to allow a strong flow of cooling and shuttling gases through the system. As with the junction between the cap and body, the junction between the probe head body and outer probe casing must be airtight to prevent noise in the sensor region, and as before this is achieved fairly easily with a slip-fit design at elevated temperature. Running through the center of the probe head body is a through-hole with diameter $\lessapprox 0.217$ " (5.5 mm), intended to allow 5 mm NMR sample tubes to pass unimpeded. The hole is flat on the bottom, but has a roughly 30 ° chamfer on top, which allows slightly mis-aligned samples coming in to both smoothly become re-centered in the guidance tube, and to lose some of their kinetic energy prior to hitting the bottom, possibly making for a more gentle and elastic collision at its eventual destination (as it happens, properly made spacers will likely prevent much contact with the side-walls of the probe head, and so in order to take advantage of these collisions, it might be preferable to use looser fit spacers and a narrower channel in the probe head body).

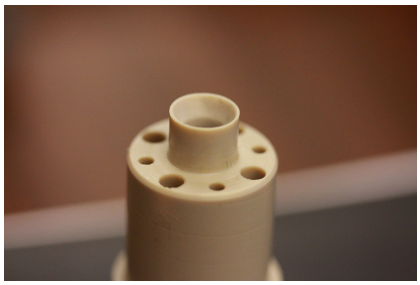


Figure 3.7: The top of the probe head is chamfered to prevent falling NMR tubes from landing askew or breaking.

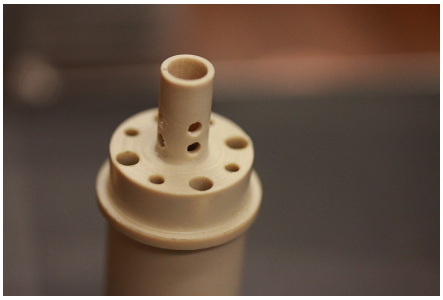


Figure 3.8: Vents are placed in the bottom of the probe head to allow air displaced by the falling sample tube to escape rapidly; without these, air is compressed below the NMR tube, slowing or stopping entirely its descent before it reaches the measurement region.

At the bottom of the probe head body is a 0.3 " diameter, 0.5 " extrusion which is loose to slip fit in the landing channel of the probe head cap (no seal is necessary or possibly even desirable, but a tighter fit tends to force the two to remain concentrically aligned). Eight $1/16$ " holes are drilled high enough that, when the probe head is assembled, they will all be exposed to the hollow sample chamber. These act as vents to allow the cooling air to flow over the summer, and to allow the air under the falling sample to be pushed out of the main body of the probe. At the top of the probe is a 0.25 " extrusion with 0.35 " (8.89 mm) diameter - this should fit loosely into the glass sample transfer tube, but not slip-fit; because the sample transfer tube has brittle ends, a tight fit is likely to result in the ends of the glass tubes becoming broken either due to minor mis-alignment between the glass tube and the outer casing or thermal expansion of the materials.

Drilled longitudinally through the main body of the probe head body are eight radially distributed holes, six $1/16$ " diameter and two $1/8$ " holes. The $1/16$ " holes serve as vents from the hollow sample chamber, allowing air to flow through the system. The $1/8$ " holes are

designed to accommodate 1/8 " OD tubing, which brings in the cool, dry nitrogen for sample cooling. Depending on the need for flow, either one or two tubes can be used for this purpose.

The air pumped into the probe head can escape from two channels - either it can escape out of the vents into the probe body or it can push past the sample and escape out the sample tube, which generally buffets the sample tube upwards. We can take advantage of this for sample shuttling by blocking the open end of the probe body and sealing it with parafilm, preventing any air from escaping from the probe body. Vent holes are then drilled into the probe body above the level of the outermost shield, but before the end of the probe - where the vents are most accessible. These vents can then be covered to prevent air from escaping from the probe body, or uncovered to prevent air from buffeting the sample. This is particularly useful when changing samples, as this cannot be done with the pneumatically actuated vacuum in place. Without pushing the sample out with a back-pressure of nitrogen in this way, removing the vacuum source from the transfer tube will cause the sample to drop. It is also not desirable to leave the probe out-flow blocked at all times, as this will cause samples to move during the experiments, leading to noise and uncertainty.

3.2.3 Active Sample Cooling

When using cooling nitrogen (which is often at or slightly below room temperature), the sample will soon reach an equilibrium temperature which depends on the temperature of the cell, the equilibrium temperature of the sensitive region of the magnetometer, the nitrogen flow and the exact geometry of the probe head, among other things. However, often this equilibrium temperature will still be slightly above room temperature, and it certainly not be lower than room temperature. When experiments require a sample to be measured at a specific temperature, it is necessary to use a more active cooling approach.

Aiming for greater temperature control, we introduced a liquid nitrogen cooled coil into the cooling line, to start the nitrogen at a significantly lower temperature. After leaving the liquid nitrogen cooling line, the delivery line is wound with a resistive heater attached to a thermocouple, allowing more heat to be added back in as needed. This system is somewhat wasteful, however, and it might be preferable to replace it with a temperature-controlled oil or water bath with a cooling unit. The main concern with most cooling apparatus is that they may introduce some vibrations into the experiment, and so it is important to keep them vibrationally isolated - this is also the case with liquid nitrogen cooling, where the LN2 boil-off can be quite volatile. In our experiments, a dewar was placed on vibration-damping foam, which seemed to remove most if not all of the vibrational noise from boil-off.

Our experiments in this front were not particularly successful, and as they were not particularly necessary to any one project, they were abandoned after early failures. The primary issues encountered were slow response times (it took quite a while for the system to come into equilibrium after each change, and often the LN2 would boil off quickly on this time scale) and sensitive calibration. Without very dry liquid nitrogen, water and (to some extent liquid oxygen) condenses in the lines, preventing any cooling whatsoever. Even then, leaks can develop as plastic tubes and junctions contract in response to low temperatures

— this can introduce some moisture and again cause plugs to form. Finally, when all these practical issues are addressed, often times our samples would freeze in place, cracking the sample tubes and preventing their removal from the probe head (which also tends to contract, making it more difficult for samples to reach the landing spot).

In future cooling system designs, we would recommend the use of a long copper or aluminum wound coil of tubing in a refrigerated bath (a saltwater bath is likely sufficient). This will not help with the slow response times, but it will prevent coolant boil-off from limiting the long-term use of the system. Additionally, either evacuating or strongly insulating all the lines carrying the cooling air is ideal, as significant condensation has been observed when using liquid nitrogen cooling. Finally, the use of cooled nitrogen gas could very well lead the probe head body and cap to contract and break their airtight seal, leading to noise in the magnetometer. In future designs for active cooling, it is likely that the use of low-temperature O-rings or gaskets would be advisable to maintain all seals. This is especially important as sudden temperature shifts at the point of the cell itself can cause immense strain and if any cold liquid were to drip onto the cell (such as condensation on the outside of the probe head cap), the cell would likely be destroyed.

3.3 Pulse Coil Design

3.3.1 NMR Pulses

Most if not all NMR experiments involve the application of sudden, non-adiabatic on-resonance pulses for control of spins. When a field orthogonal to the direction of spin polarization is turned on non-adiabatically, this induces a precession of frequency $\omega_1 = \gamma \mathbf{B_1}$ where $\mathbf{B_1}$ is the amplitude of the applied field. This fact can be used to control spin evolution using pulsed magnetic fields - for example, spins aligned along z can be made to align along y by applying a pulse along x for $t_{\pi/2} = \pi/2\gamma B_1$[2]. In a more general form, the tip angle from applying a (possibly time-dependent) magnetic field with amplitude $B_1(t)$ transverse to the spins is given by the integral of the precession during the pulse:

$$\theta_p = \int_0^{t_p} \gamma B_1(t) t \mathrm{d}t \tag{3.1}$$

For a stationary pulse applied to spins at zero field, B_1 is constant, and so $\theta_p = \gamma B_1 t$, but in the presence of a bias offset field $B_0 \gg B_1$ along $\vec{\mathbf{z}}$, spins tipped transverse to the field will precess at frequency ω_0, and so the transverse amplitude varies as a function of time. The natural coordinate system for analyzing the spin evolution of such a system is not the laboratory frame but a rotating frame of reference in which unperturbed spins are stationary. This can be achieved by defining time-dependent axes $\vec{\mathbf{x}'}$ and $\vec{\mathbf{y}'}$:

[2]Here γ is in units of angular frequency per magnetic field ($^{\mathrm{rad\cdot Hz}}/_{\mathrm{G}}$, $^{\mathrm{rad\cdot MHz}}/_{\mathrm{T}}$, etc). For $\bar{\gamma} = \gamma/2\pi$ in units of angular frequency ($^{\mathrm{Hz}}/_{\mathrm{G}}$, $^{\mathrm{MHz}}/_{\mathrm{T}}$, etc), the value is $t_{\pi/2} = 1/4\bar{\gamma}B_1$

$$\vec{\mathbf{x}'} = \vec{\mathbf{x}} \cos(-\omega_0 t) - \vec{\mathbf{y}} \sin(-\omega_0 t) \tag{3.2}$$

$$\vec{\mathbf{y}'} = \vec{\mathbf{x}} \sin(-\omega_0 t) + \vec{\mathbf{y}} \cos(-\omega_0 t) \tag{3.3}$$

In the lab frame, a static pulse along $\vec{\mathbf{x}}$ is described by Eqn 3.4

$$B_x(t) = B_1 \vec{\mathbf{x}} \tag{3.4}$$

However, when transformed into the rotating frame, the same field is described by Eqn 3.5:

$$B_x(t) = B_1 \left[\vec{\mathbf{x}'} \cos(\omega_0 t) - \vec{\mathbf{y}'} \sin(\omega_0 t) \right] \tag{3.5}$$

So for spins along $\vec{\mathbf{y}'}$, applying the pulse $B_x(t)$ gives a tip angle given by the integral of the amplitude along the transverse $\vec{\mathbf{x}'}$ direction (Eqn 3.6):

$$\theta_p = \gamma B_1 \int_0^{t_p} \cos(\omega_0 t)\, \mathrm{d}t \tag{3.6}$$

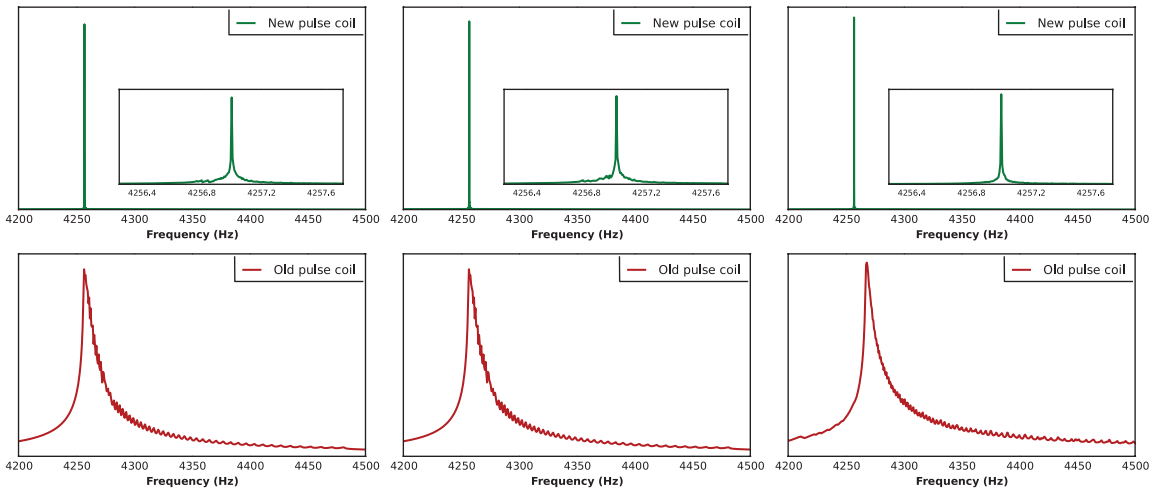
For $\omega_0 \gg t_p$, this integral evaluates to 0, and so such a pulse would yield no net tip of the spins. Modulating B_1 at frequency ω_0, however, gives a tip angle:

$$\theta_p = \gamma B_1 \int_0^{t_p} \cos(\omega_0 t)^2\, \mathrm{d}t \tag{3.7}$$

And for $\omega_0 \gg t_p$ this evaluates to $\frac{1}{2}\gamma B_1 t_p$.

3.3.2 DC Pulsing

Our experiments all take place at zero magnetic field. This is a special case of the rotating frame where $\omega_0 = 0$, and so DC pulses can be used for spin manipulation. Because the spins do not precess with respect to the pulse coils, the rotating-frame pulse "phase" is replaced with a physical, lab-frame, direction, and as such separate orthogonal coils must be used for pulse application. However, because a DC pulse at zero field is effectively a full quadrature pulse (whereas a single coil in a rotating frame only applies half the quadrature pulse), the tip angle at zero field is twice what it would be for the same amplitude in the RF context. This has consequences for device voltage, as the RF transmitter power is proportional to the waveform root-mean-squared, while the DC power is proportional directly to the amplitude - as such to apply RF pulses of the same power as a DC pulse requires voltage to be stepped up by a factor of $\sqrt{2}$.



(a) X Coil (b) Y Coil (c) Z Coil

Figure 3.9: A comparison of the predicted pulse coil inhomogeneity. These spectra are generated by simulations of the proton precession frequency over the sample volume, weighted by $1/r^3$ where r is the distance between a given point in the sample volume and the center of the sensitive magnetometer volume.

3.3.3 Coil Designs

Two sets of pulse coil designs have been used in these experiments - a smaller set of 3 coils wound about a Macor substrate, and a larger set of 4 coils wound about a 3D printed epoxy-infused plaster substrate. In both cases, 30 AWG high-temperature magnet wires were wound in grooves machined (or printed) into cylindrical substrates. In the smaller coils, saddle coils are wound about x and y with a Helmholtz pair about z. In the larger coils, a "g" (gradient) coil is wound as an anti-Helmholtz pair.

The larger coils were designed in response to homogeneity concerns about the smaller coils. The smaller set of coils was made from a 2 " diameter 3 " height cylinder, with orthogonal 0.5 " holes centered longitudinally (1 " from the top and bottom). Grooves $1/16$ " deep were used for all coil windings. Helmholtz pair geometry is constrained such that the coil separation $\Delta z = r$, where r is the radius of each loop. Assuming that a single layer of 30 AWG wire is used in a $1/16$ " groove, the separation between the centers of the Helmholtz grooves was set to 0.9425 ". The saddle coils were wound with 120 ° angular extent along grooves wound at the top and bottom edges of the coil substrate. The radius depends somewhat on the way the coils are wound, but is ≈ 0.9375 ", with heights of 1.875 ", for an $\frac{h}{r}$ ratio of 2, far from the optimal value of 4 described in *Ginsberg et al., 1970* [17].

Another factor leading to significant inhomogeneity is that the original coil designs did not take into account the fact that the pulses are to be applied to the *sample*, rather than the cell. In the most recent design of the magnetometer, the bottom of the sample tube is approximately 0.1 " (2.5 mm) from the top of the cell; when filled with a volume of 100 μL, the height of the fluid in the tube is ≈ 0.31 " (8 mm) from the bottom of the tube. The homogeneity of a Helmholtz coil is greatest at the center, with the field dropping off by

around 150 ppm ± 10 % of the radius away from the center, and by 2300 ppm at ±20 %. Since the radius of the original pulse coils was 1 ", the sample being centered ≈ 0.25 " above the coil center, and thus would be in the inhomogeneous fringe field. When the new, larger coils were designed, the pulse coils were thus centered 0.25 " above the center of the optical access coils. Combined with the increase in the coil radius and the optimization of the h/r ratio for the saddle coils, the pulse homogeneity across the sample increased dramatically, as seen in the simulations plotted in Figure 3.9 (which are consistent with observations).

The blueprints for both sets of coils can be found in Appendix C; the new pulse coils are Figures C.3 and C.4, the smaller pulse coils are Figure C.5,

3.3.4 Substrates and Materials

These coils are all wound on cylindrical substrates with grooves to accommodate magnet wire. The choice of material for the substrate is strongly constrained by the fact that these coils are located in the sensitive detection region. Magnetic materials cannot be used at all inside the shields, as they generally become magnetized, inducing large (relative to the sensitivity of the magnetometer) bias fields and gradients. Non-magnetic metals like copper and aluminum can be used if absolutely necessary, but the presence of large conductive bodies tends to induce large amounts of Johnson noise and limit the sensitivity of the device. Additionally, depending on the optimal cell operating temperature and thermal equilibrium characteristics of the system, the temperature in the sensitive region will often be very elevated, particularly near the cell itself - as such, all the system components must be designed to operate at high temperatures.

In our systems, most components are made of a combination of ceramics, high-temperature plastics and fiberglass, each of which has its own advantages. Ceramics are particularly attractive for their physical properties - they can be made as thermal insulators or conductors as needed, they are often quite rigid and have low thermal coefficients of expansion. However, there very few ceramics which can be easily machined, which is a significant constraint in custom applications such as this (though may not be so in bulk industrial production where they can be cast). Fiberglasses such as Garolite can be quite strong and rated to high temperatures. These have many favorable properties, but tend to be avoided when possible because although they are easier to machine than ceramics, doing so can represent a health hazard if the fibers enter the lungs.

In most cases, the best material to use is a high-temperature plastic. In our magnetometers, we generally use either PTFE (Teflon, polytetrafluoroethylene) or PEEK (polyetheretherketone), though these are by no means the only high temperature plastics available. PTFE is extremely common, and so it is relatively inexpensive and frequently available in many forms, making it easy to find appropriate starting materials for a given part. PTFE is generally rated for use up to ≈ 300 °C, and is a good thermal insulator (with a thermal conductivity of 0.25-0.35 $^{W}/_{m\cdot K}$), but it tend to have a very high coefficient of thermal expansion ($135 \cdot 10^{-6} K^{-1}$), and is very slippery - both of which make it very difficult to machine with great precision. Acetal (Delrin) tends to have similar physical properties to PTFE, but with

Figure 3.10: Photographs of the new pulse coil assembly (left), old pulse coil assembly (top right) and a comparison between the new and old substrates (bottom right).

somewhat better friction, and so may be a good replacement if available. PEEK has similar thermal properties - 0.25 $^{W}/_{m\cdot K}$ thermal conductivity and a slightly better $60 \cdot 10^{-6} K^{-1}$ coefficient of thermal conductivity - but has much better friction, and can be much more easily machined. Additionally, PEEK is much less pliable than PTFE, giving much greater tolerances on objects with thin feature sizes, as well as making such objects more robust.

In the latest version of the pulse coil, the substrate was 3D printed using a 3D Systems Z-Printer 150, which uses ink jet technology to selectively bind successive layers in a powder bath, and when complete, the part is infused with a hardening epoxy. The physical properties of the object are primarily determined by the epoxy, and so can be varied depending on what is desired. Although there are a set of well-tested epoxies frequently used with Z-Printer models, any epoxy or resin can theoretically work. Generally a primary concern is the fluid viscosity, as this will determine the degree to which the epoxy can penetrate the material. Most commonly-used epoxies as hardeners are not rated for high-temperature use, and so a special high-temperature epoxy was used.[3] These epoxies generally have a certain temperature hysteresis response, and so they are hardened by sequentially baking and cooling the part progressively until the desired temperature rating was achieved. In our tests of the epoxy some outgassing and scorching were observed during the baking procedure, but in nearly a year of constant use at $\approx$ 90-150 °C, no distortions have been observed.

The primary advantages of 3D printing of parts are time and cost, as the marginal cost of production of a given part is quite low, and printing is often quick - this is particularly advantageous in technology development applications, where prototype production may be a limiting step. However, currently the library of materials available for commercial 3D printing applications is very limited, which is a significant constraint on its potential use.

3.3.5 Pulse Generation Circuit

Pulse control circuitry can be one of the most difficult aspects of magnetometer design as it has many significant challenges. Pulses generally need to be fast, precisely timed, strong, stable, extremely reproducible and not distorted - and produce very little noise. Depending on the purpose of the pulses, one may also be concerned with linear input response, AC performance and output symmetry.

When pulse length may present a problem for a given set of experiments, it is preferable to use extremely strong pulses. This is desirable for example in situations where large numbers of pulses are used, as longer pulses will take up a comparatively greater fraction of the experiment time and may impose limits on the experiment. An example of such problems is provided in section 6.3.2.2, wherein pulsing-time fraction can effect measured relaxation properties. Strong pulses are also valuable for several J-spectroscopy applications wherein spins are addressed by their differential gyromagnetic ratio ($\gamma_I - \gamma_S$) rather than

[3] At the time, no high temperature epoxies were available from 3D Systems, and so I obtained from the local representative a sample of epoxy intended to be tested for high-temperature use. Other than some scorching during the higher-temperature baking process, the epoxy performed well, and there were no structural problems with the resulting parts.

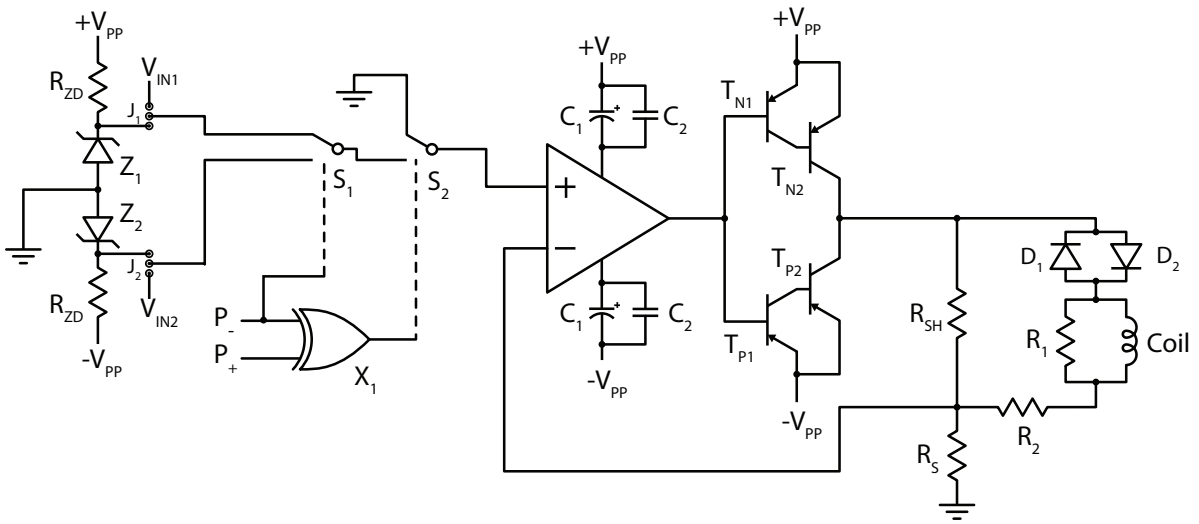


Figure 3.11: The most recent version of the Pulse Coil Circuit

the gyromagnetic ratio of either spin - the closer the two values, the longer the pulse needs to be. One significant downside of using strong pulses is that timing errors produce larger pulse angle errors, as they make up a larger fraction of the pulse. High-powered pulsing applications also require much more care in their design, as the thermal properties of both the components and the coil become important as large currents are passed through them. In general, however, the main limit on our pulsing power has not been the circuit design, but rather the fact that strong pulses tend to magnetize the shields, changing the magnetic environment and interfering with our signal.

The circuit we use is a 4-channel bi-directional TTL-controlled pulse generator which was designed primarily for stability and versatility, as it is used in many experiments on multiple instruments, and often for multiple purposes within a single experiment. It can be used to deliver either DC or low-frequency RF pulses (the frequency cutoff is determined by the gain bandwidth of the op-amp), with maximal symmetry between positive and negative output. The circuit for each channel (which is shown in Figure 3.11) is divided (roughly) into three stages: an input stage, an amplification stage and an output compensation stage.

3.3.5.1 Input Stage

This circuit is a bi-directional TTL-controlled DC pulse generator with the option to configure each pulse "direction" using either a fixed DC level or an analog voltage input. In the physical realization of the circuit, the mode of a given channel is chosen by jumpers J_1 and J_2, as the decision to switch a channel (or half a channel) DC and voltage-controlled currents tends to be semi-permanent. These could also be replaced with physical switches without much issue.

The pulsing is controlled by a pair of TTL-controlled double-pole single throw break-before-make analog switches (MAX319). S_1 controls the input voltage channel and S_2 enables

a pulse. The TTL-level voltage inputs (TTL_1 and TTL_2) feed into an exclusive-OR switch, which controls the analog switch (S_2). The S_1 switch is directly controlled by the channel corresponding to the "normally closed" input channel (in this case TTL_2 - this ensures that when it needs to be activated the S_1 switch will always switch *before* the S_2 switch, as the XOR introduces a short delay into the logic propagation.

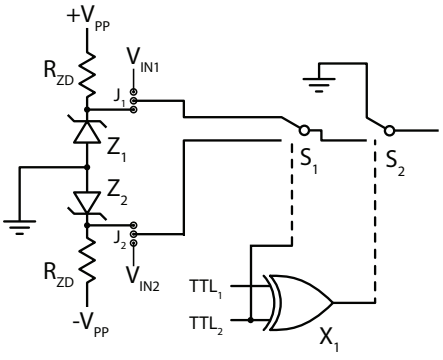


Figure 3.12: Input stage of the pulse coil circuit

The fixed-voltage inputs are provided by a pair of reference Zener diodes, in our application these had a voltage drop of 5 V. The current drawn from these inputs is small, and so the current-limiting resistors R_{ZD} can be large. Depending on the application, it would likely be appropriate to add current-limiting resistors and/or protective optocouplers to the arbitrary voltage inputs.

3.3.5.2 Amplification Stage

The selected input channel is then fed into an operational amplifier configured as a voltage-controlled current source. To supply the often significant current required for strong, square pulses, a push-pull output circuit is wired within the feedback of the operational amplifier. In previous circuits, the current amplification took place outside the feedback stage, as the base turn-on junction of the transistors serves to filter out noise on the inputs of the operational amplifier. However, there are several distinct disadvantages as well — one primary drawback is that when used outside the feedback loop, the actual current output during the pulses will depend on the specific conduction and amplification characteristics of the transistor - which can introduce short-term drift into the pulse calibrations as the transistors heat up, as well as long-term drift in the calibrations as the transistors age.

When included in the feedback loop, the op-amp output compensates for the specific characteristics of the transistors, providing consistent output. An additional advantage to this configuration is that it allows an AC signal to be passed as a voltage input without much output distortion, as the op-amp output effectively eliminates the diode drops in the push-pull stage which normally severely distort AC output. Such AC signals would, however, be limited to fairly low frequencies, however, as the op-amp is required to slew very quickly when the output signal is in the transitory region between $\pm 2 \cdot V_D$ (or $\pm V_D$ when using Sziklai rather than Darlington pairs in the push-pull circuit).

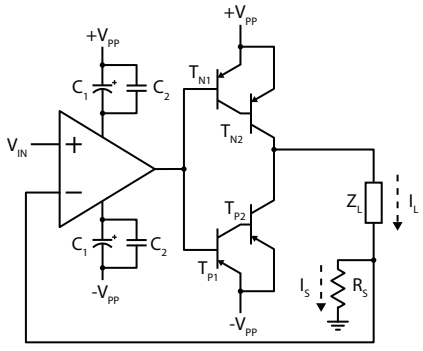


Figure 3.13: Equivalent circuit of the amplification stage of the pulse coil circuit.

3.3.5.3 Output Stage

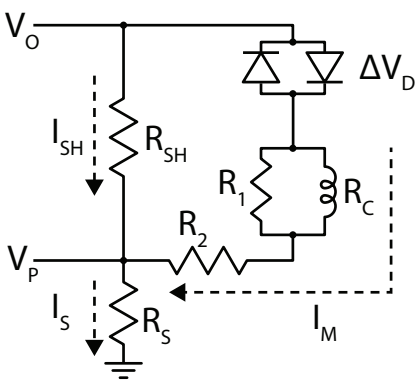


Figure 3.14: The output stage of the pulse coil circuit.

To eliminate noise from the logic and push pull stages of the circuit, anti-parallel diodes are placed in series with the coil, introducing a 0.5 V voltage drop between the output of the push-pull stage and the input to the coil. A shunt resistor (R_{SH}) is in parallel with both the coil and the diodes, and for any current such that $I \cdot R_{SH} < \Delta V_D$, all current flows only through the shunt resistor and not through the coils. Above the diode drop voltage, the fraction of the current passing through the shunt resistor is given by

$$\frac{I_{SH}}{I_{SH} + I_M} = \frac{V_P R_M + \Delta V_D R_S}{V_P (R_M + R_{SH})}, \tag{3.8}$$

which quickly dies off with the pulse strength. Ideally R_{SH} is set such that the the maximum values of the noise current generate voltages on the order of ΔV_D, as this minimizes current passing through the shunt current and reduces distortion in AC cases and when applying low-intensity pulses.

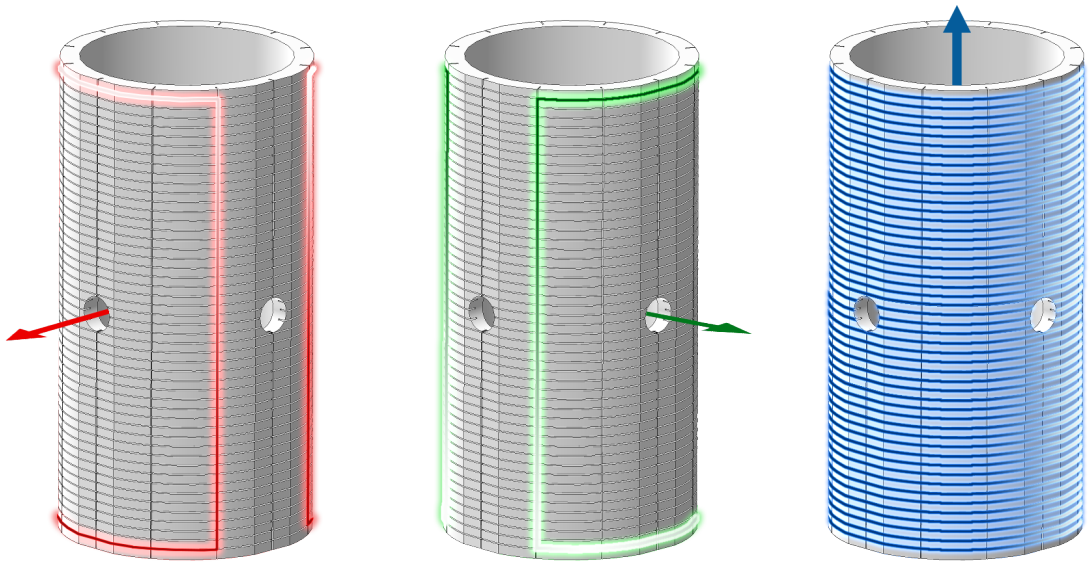
3.4 Field Shimming

3.4.1 Shim Coils

In order to operate at zero field (or at an arbitrary but chosen field), it is necessary to shim out any remaining fields or field gradients resulting either from residual external magnetic fields or the slight magnetization of the shields. In this magnetometer, we use a tri-axial set of coils wound on a PTFE substrate to shim the field to the desired levels in each direction. No gradient shims are wound because no ill effects of residual gradients have been observed in these experiments.

3.4.1.1 Design

The shim coils are wound from 30 AWG copper magnet wire on a PTFE substrate which has a height of 12 " (30.48 cm), an inner diameter of 5.5 " (13.97 cm) and an outer diameter of 6 " (15.24 cm). Radial grooves $\frac{1}{16}$" (1.5875 mm) wide and $\frac{1}{4}$ " (6.35 mm) deep are cut at $\frac{1}{4}$ " intervals, starting 1/8 " (3.175 mm) from the bottom of the substrate, with 12 additional longitudinal grooves arrayed at 30 ° intervals. This substrate is intended to fit snugly inside the innermost shield, allowing the coils to be as big as possible to maximize homogeneity at the point of the cell. Optical access is granted via 2 sets of orthogonal 0.93 "-diameter holes at the center (vertically) of the structure, and in order to accommodate weld seams at the point of the optical access holes in the shields, the 4 sides hosting optical access ports are milled down to a flat surface. Because the optical access holes are larger than the spacing between radial grooves, it is necessary for several of the "z" coil windings to avoid the optical



(a) (b) (c)

Figure 3.15: The coil windings are shown for the x (a), y (b) and z (c) coils.

access holes. In this design, a 1 " ring groove surrounds each optical access hole, allowing the magnet wire to wind around the holes.

Three coils are wound on this structure (Fig. 3.15): the transverse coils use a "saddle" configuration to generate fields along x and y, while the the longitudinal coil uses a solenoid-like configuration to generate fields along z. Each lobe of the transverse coils consists of 2 windings with 120 ° extent, with radius of 2.25 " (13.97 cm) and a height of 11.75 " (29.845 cm), a ratio of 5.22:1, which deviates from the optimal value of 4:1. As a result, the homogeneity of the field created across the cell is 109 ppm, as compared to 66 ppm for an optimal 4:1 coil. However, since there are radial grooves every 0.25 ", it is possible to re-wind the coils with a 9 " height to get an optimal 4:1 ratio without creating a new structure. In any case

The spacing of the longitudinal coils is designed keeping in mind that these coils are sitting inside a shielded environment with flat end-caps. These flat end-caps act as magnetic "mirrors", and since the distance from the end-caps is $\frac{1}{2}$ the distance between windings, this geometry is equivalent to an infinitely long solenoid with 0.25 " inter-wire spacing. This configuration provides a very homogeneous magnetic field, with a calculated homogeneity of 1.8 ppm across the volume of the cell.[4]

[4]This is across the entire volume of the TwinLeaf cell, 5 mm×5 mm×8 mm, not the intersection of the beams, which is a smaller region within the cell.

	Res. (Ω)	Ind. (μH)	Response (G/A)
X	2.5	12.7	0.101
Y	2.4	12.7	0.101
Z	18.1	709.8	3.92

Table 3.1: Physical properties of the shim coils

3.4.1.2 Control Circuit

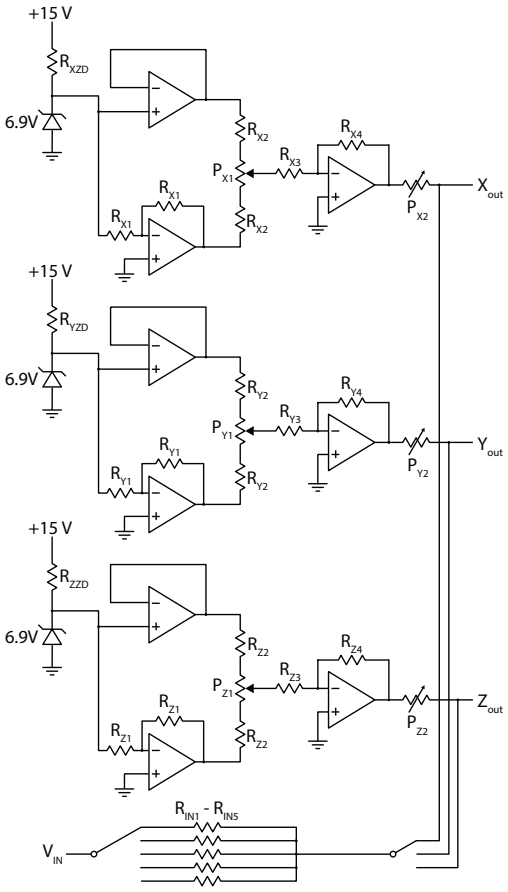


Figure 3.16: The field coil circuit.

The shim coils are controlled with a homebuilt three-channel bi-directional op-amp current source (see 3.16), with the output voltage on each channel controlled by two independent potentiometers a coarse control (P_{1i}) and a fine control (P_{2i}). The output range of the circuit is limited by the choice of resistors, which is adjusted as-needed using a current-limiting potentiometer on the output (P_{3i}). This is generally set such that the span of the output is limited to the voltage which produces a magnetic field spanning the sensitive region of the magnetometer, to give the finest control.

3.4.1.3 Noise Analysis

There are two major sources of noise relating to the field coils, noise from the circuit and Johnson noise induced across the coils themselves. Depending on the specific configuration, either could be significant, and both can be reduced if they become limiting sources of noise. The Johnson noise current induced across a resisting wire is given by Eqn. 3.9:

$$I_{JN} = \sqrt{\frac{4k_B T \Delta f}{R}} \tag{3.9}$$

Where Δf is the bandwidth, k_B is Boltzmann's constant, T is temperature in Kelvin and R is resistance. In the case of this design, this is a much more significant for the z coil, which has a much greater magnetic response than the transverse (x and y) coils. In the specific case detailed in Table 3.1 at a temperature of 65°C, the Johnson noise along the z axis is $13\,^{\mathrm{fT}}/\sqrt{Hz}$, and $0.87\,^{\mathrm{fT}}/\sqrt{Hz}$ along each transverse direction. This can be easily alleviated by choice of wire gauge, as resistance is inversely proportional to the cross-sectional area of the conductor and as such resistance decreases as the square of the wire radius. Thus switching from 30 AWG (as was used here) to 36 AWG would increase the resistance of the coil by a factor

of ≈ 4, and reduce induced Johnson currents by a factor of 2 without affecting coil response or geometry - and as such would be advisable for situations where current is low enough that an increase in resistance would not cause damage to the coil. For significant increases in resistance, it might be necessary to adjust the limiting output resistor (P_{Z2} in Figure 3.16) to compensate.

The second source of noise from the shim coils is noise from the circuit components - primarily the voltage reference and op-amps, as these are much more significant than noise from the resistors used in these circuits. The output current of the z channel of the circuit detailed in Figure 3.16 is given by the following equation:

$$I_Z = \frac{-R_{Z4}(P_{X2} + R_{Coil})}{R_1R_2 - R_{Z3}(R_1 - R_2)} \left[V_{ZD}(R_2 - R_1) + V_{O1}R_2 + V_{O2}\right] + V_{O3}\left(1 - \frac{R_{Z4}}{R_{Z3}}\right) \tag{3.10}$$

Where $R_1 = R_{Z2} + P_{Z1}p$, $R_2 = R_{Z2} + P_{Z1}(1-p)$ and $0 \leq p \leq 1$ (representing the position of the potentiometer). The terms V_{O1}, V_{O2} and V_{O3} refer to the voltage offset input bias, and for the purposes of this error analysis are all set to $0 \pm \sigma_{OP}$. Using this, we can determine the overall noise σ_{IZ} by propagating the errors from the op-amps and Zener diode references:

$$\sigma_{IZ} = \sqrt{\left(\frac{\delta I_Z}{\delta V_{ZD}}\right)^2 \sigma_{ZD}^2 + \left(\frac{\delta I_Z}{\delta V_{O1}}\right)^2 \sigma_{OP}^2 + \left(\frac{\delta I_Z}{\delta V_{O2}}\right)^2 \sigma_{OP}^2 + \left(\frac{\delta I_Z}{\delta V_{O3}}\right)^2 \sigma_{OP}^2} \tag{3.11}$$

The Zener diode term comes out as:

$$\sigma_{ZD}^2 \left(\frac{\delta I_z}{\delta V_{ZD}}\right)^2 = \frac{R_{Z4}^2(P_{X2} + R_{Coil})(R_2 - R_1)}{[R_1R_2 - R_{Z3}(R_1 + R_2)]^2} \tag{3.12}$$

The terms from the errors on the three op-amps are:

$$\sigma_{amp} = \sigma_{op}\sqrt{\left(\frac{\mathrm{d}I_z}{\mathrm{d}V_{o1}}\right)^2 + \left(\frac{\mathrm{d}I_z}{\mathrm{d}V_{o2}}\right)^2 + \left(\frac{\mathrm{d}I_z}{\mathrm{d}V_{o3}}\right)^2} \tag{3.13}$$

$$= \sigma_{OP}\sqrt{(P_{X2} + R_{Coil})\left[\frac{R_{Z4}^2(R_1^2 + R_2^2)}{[R_1R_2 - R_{Z3}(R_1 + R_2)]^2} + \left(1 - \frac{R_{Z4}}{R_{Z3}}\right)^2\right]} \tag{3.14}$$

One important thing to note is that due to the symmetry of the circuit, the noise from each component varies quadratically with p, but the error response from the op-amps is concave while the error response from the voltage reference is convex (see Fig. 3.17). The op-amp noise is minimized at the extremes, because at those points there is only one source of noise, while at $p = 0.5$ the noise is maximized because there are equal contributions from two independent noise sources. Because the voltage reference provides identical noise to each side of the potentiometer, it is minimized at $p = 0.5$, where it is perfectly canceled.

Depending on the noise characteristics of available components, this analysis can argue for different configurations. The follower circuit branch of the current source is in place for added symmetry and to provide high output impedance for the next output stage. If these op-amps are the limiting noise source, however, modest gains may be achieved by removing the follower circuit, provided that sufficient current is supplied to the voltage reference (i.e. R_{ZD} is sufficiently high).

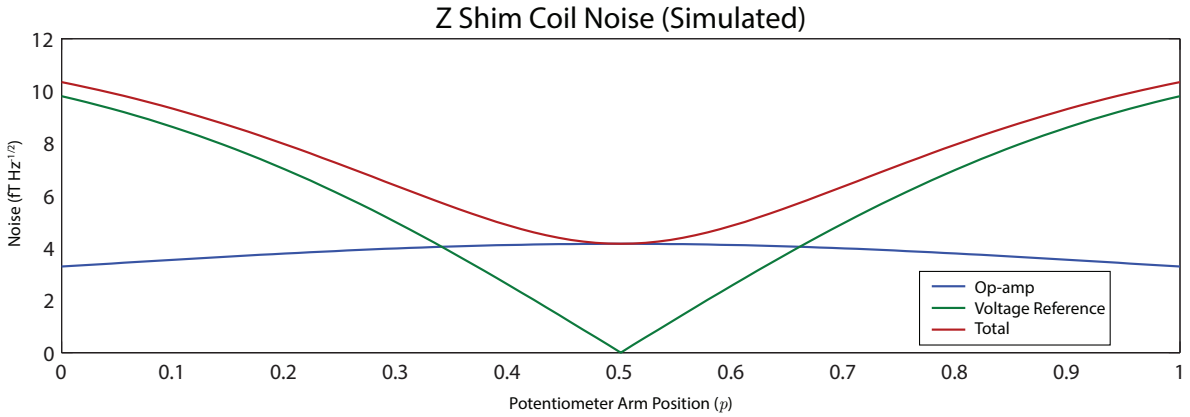


Figure 3.17: The expected circuit noise of the z shim coil based on the current configuration with $\sigma_{ZD} = 100\,\mathrm{nV}/\sqrt{\mathrm{Hz}}$, $\sigma_{OP} = 15\,\mathrm{nV}/\sqrt{\mathrm{Hz}}$.

3.4.2 Field zeroing

3.4.2.1 Applied Oscillation Method

Thanks to the vector nature of the SERF magnetometer, there is a simple way to make a preliminary zeroing of the transverse (x and y - see: Choice of Coordinates) fields using the magnetometer signal alone. The magnetometer signal is given by the spin projection along the probe beam direction[18], which is

$$S_x = S_0 \frac{\beta_z + \beta_x \beta_y}{1 + \left(\beta_x^2 + \beta_y^2 + \beta_z^2\right)}, \tag{3.15}$$

where

$$\boldsymbol{\beta} = \left(\frac{\gamma^e}{R_{OP} + R_{rel}}\right) \mathbf{B}. \tag{3.16}$$

As such, if an oscillating magnetic field is applied along the x direction, to first order the amplitude of the magnetometer response will be modulated by the residual field in the y direction, and if an oscillating magnetic field is applied along the y direction, to first order the amplitude of the magnetometer response will be modulated by the residual field in the x direction. These fields can then be zeroed by minimizing the amplitude of the observed oscillation, while the field in the z direction can be zeroed using the DC level.

Note that this method should be applied iteratively, cycling through the x, y and z zeroing procedures until no further oscillations are observed. This method works best when the magnetometer is operating at closest to zero in all three directions, and as such with each iteration the accuracy of the method is increased.

3.4.2.2 Sample NMR Precession Method

Given accurately determined pulse times, one way to zero the fields transverse to the polarization direction is by observing the low-frequency NMR signal of a long-T_2 sample in the presence of a train of π pulses spaced in increments of τ. For $\tau \ll \gamma B_x$ a π_x train acts as a spin lock along the x direction, and spin precession is entirely attributable to a residual field along B_y. The $\vec{y}$ shim can then be adjusted until no precession is observed.

This approach is potentially most strongly limited by the error on the π pulses, as error in the $\vec{x}$ pulse will translate into error in the B_x shim. For

$$\tilde{B_x} = -\frac{\epsilon_x}{\tau\gamma} \tag{3.17}$$

Where $\tilde{B_x}$ is the offset bias field along x which accounts for a pulse error of angle ϵ_x along $\vec{x}$. Another potentially limiting factor in the accuracy of the calibration is the decay from the T_1 of the sample, as no NMR precession can be observed when $\omega \ll T_1$. For $\omega \ll {}^1/_{T_1}$, signal decay occurs faster than NMR precession, and differences in precession frequency cannot be easily observed. Generally, it is possible for pulse errors to be reduced such that the limiting factor on NMR precession observations is the low field T_2 of the sample.

That said, this is likely the most sensitive measurement of the absolute field at the sample, as it is not subject to light shifts or birefringence, and is an effect based purely on the local magnetic field. Using the proper pulse calibration sequences[5], the pulse error can be reduced to the digitization error on pulse lengths - and the effective pulse error can be reduced even further by application of error-correcting pulse sequences, when pulse length calibration is extremely important. Even greater calibration sensitivity can be achieved by using long-T_1 materials - the simplest thing to do is often to detect water at an elevated temperature, as its low field T_1 and T_2 are a strong function of temperature the samples, and so by either actively heating or reducing cooling mechanisms, the T_1 can be increased as needed.

Even the need for for accurate pulses can essentially be obviated at zero-field by using J-coupling spectra as the field probe; the theory of zero-field J-coupling spectroscopy is described in more detail in Section 5.3. Because there is no preferred axis for J-coupling spectra, the signal relies on the relative orientations of heteronuclei; as a result, spins undergoing incomplete excitation should simply not contribute to the J-coupling signals. When sufficiently small, deviations from zero field show up in J-spectra as perturbations, with the peaks split by the presence of the field.[19] The spectrum of ^{13}C-labeled formic acid, for example, has a single peak at 222 Hz, which is split into a doublet with peaks separated by $\delta\omega = (\gamma_H + \gamma_C)\mathbf{B_0} = 5328.1\,{}^{\text{Hz}}/_{\text{G}}\cdot B_0$. Since the dependence on the field is linear for sufficiently

[5] See Section 5.6.3.1 for more details on pulse calibration.

small fields, the precise location of the required shim strength can be determined by making plots of the splitting frequency as a function of the field applied in each direction.

3.5 Cell Heating

	A_{mTorr}	B_K	C_K
K_{liq}	14.114	-4693	-1.2403
Rb_{liq}	14.197	-4275	-1.3102
Cs_{liq}	14.113	-4062	-1.3359

Table 3.2: The coefficients which determine the alkali vapor pressure, based on Alcock et al.'s data.[20] Applying these coefficients to Eqn. 3.18 for T given in K returns $\ln\left(\frac{p}{mTorr}\right)$.

Alkali vapor-cell magnetometers generally operate at an elevated temperature, as the temperature determines the vapor pressure of gaseous metal atoms through which the beams will pass. This is particularly important for spin-exchange-relaxation-free magnetometers, which operate in cells with a high density of buffer gas used to suppress spin-exchange relaxation[15,21]. Experiments carried out on this magnetometer were performed with cells operating at temperatures ranging from 155-195 °C.

There are several ways that changing temperatures can affect signal and signal strength. Likely the most significant effect is the change in alkali vapor density, which depends strongly on the temperature. The alkali vapor density is given by:[20,22]

$$\log(p) = A + \frac{B}{T} + C \cdot \log(T) \tag{3.18}$$

Where A, B, and C are the 1st-3rd Virial coefficients. From literature values of these coefficients, it can be seen that the partial pressure (which is fairly directly related to the optical density of the cell) is a non-linear exponential function of the temperature.

Creating such high temperatures introduces a number of challenges to the instrument design. Generally it is not desirable for samples or many common materials to be held at such high temperatures, but the fact that sample signal is inversely proportional to the cube of the

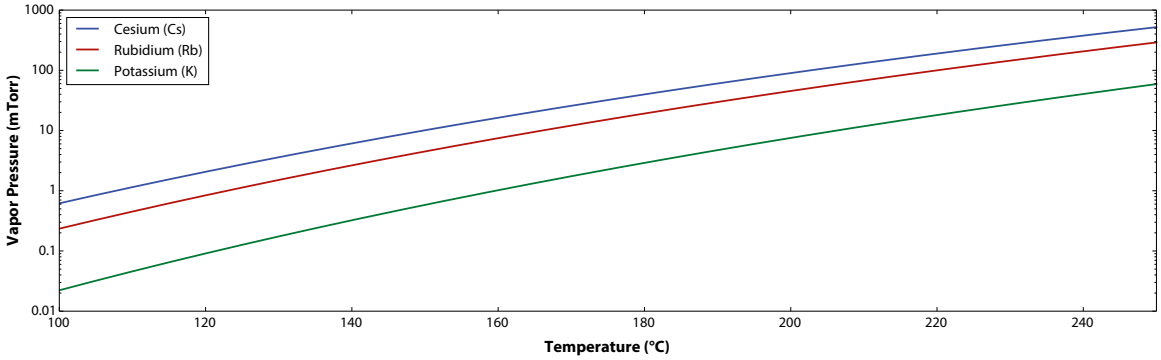


Figure 3.18: The vapor pressure curves for the three alkali metals most commonly used in magnetometry. For low-temperature operations, Cesium is a natural choice, as the required vapor pressure can be created at much lower temperature.

distance from the cell precludes the use of significant insulation. Thus a situation wherein large temperature gradients are present is necessitated by the geometry of the instrument, and this can cause significant convection currents and induce birefringence in the cell's glass windows.[23]

In addition to the problems caused by the elevated temperature, there are also challenges in producing the heating without introducing significant noise. There are two common approaches to this problem: resistive (coil) heating and laser heating. Resistive heating has two major significant problems - the current used to produce the heat will also produce a stray magnetic field, and the presence of enough metal to heat the cell will also likely introduce significant Johnson noise. Laser heating is highly preferable from a noise perspective, but can be very impractical for large cells and open geometries which require significant amount of power to heat. The heater designs used herein have required 10–20 W of power, and it is difficult to find a material capable of efficiently transforming that amount of laser power into heat without ablating or otherwise becoming damaged. For smaller, microfabricated magnetometers, the power requirements can be on the order of 100-200 mW,[13,24] and in those applications, laser heating is practical and advantageous.

3.5.1 Heater Design

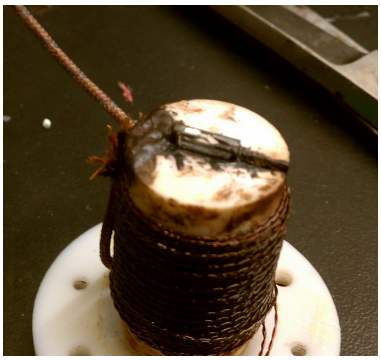


Figure 3.19: The original heater coil, with one of many shattered cells glued in place.

The earliest design for the heater was a simple Macor rod with high-temperature-enamel-coated 32 AWG twisted pair wire wound around it and a cell and thermocouple glued into a groove glued to the top, shown in Fig. 3.19. The circuit was driven by the pulsed DC heater described in Sec. 3.5.2. Among other reasons, this was found to offer inadequate protection to the magnetometer cell, resulting in the shattering of at least one cell during sample shuttling.

In order to minimize the possibility of heater shattering, a cap was added to the heater with offset stands upon which the probe head could sit, leaving $\approx$ 100–200 μm space between the top of the cell and the bottom of the probe head, in an attempt to mechanically isolate the two. The cap was made of PEEK both for ease of machining and to thermally isolate the sample region from the heated cell. To cut down on thermal gradients, power consumption and thermally-driven air convection on the optical beam path, the rest of the heater was redesigned with thermal insulation in mind, given the significant size constraint that the entire heater apparatus needed to fit inside the small pulsing coil structure described in Sec. 3.3.3 and shown in the top right panel of Fig. 3.10.

The first design change made was that the slightly thermally insulating Macor core (with a thermal conductivity 1.46 $^{W}/_{m\cdot K}$ and a specific heat of 0.8 $^{kJ}/_{kg\cdot K}$) with a rod of thermally conductive aluminum nitride (which has a has a thermal conductivity of $\approx$180–200 $^{W}/_{m\cdot K}$ and a specific heat of 0.75 $^{kJ}/_{kg\cdot K}$). Because aluminum nitride is extremely hard and cannot

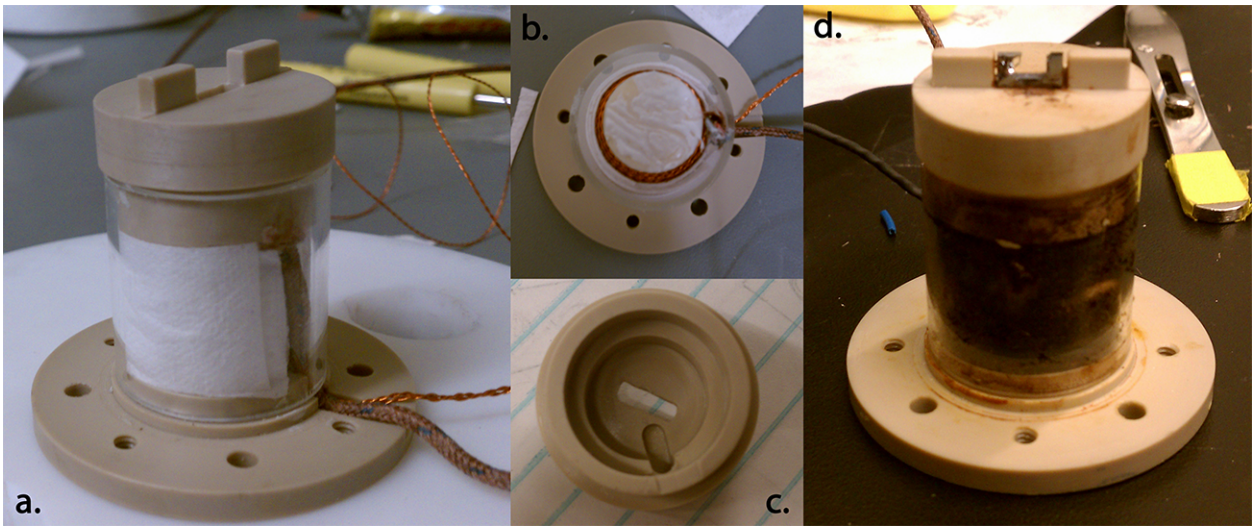


Figure 3.20: The second revision of the heater coil. a.) The assembled apparatus, with no cell in place b.) An overhead view of the wound coils, before the cap is added, but after high-temperature epoxy was applied to the heater coils c.) A view of the underside of the cap - a small notch is milled to accommodate the thermocouple d.) A view of a heater after use, with yet another shattered cell in place; the the extreme temperatures caused significant scorching of the insulating materials.

be readily machined, it was not possible to mill a groove into the heater core that would accommodate a cell, and so the cell was primarily held in place by the cap itself. Rather than exposing the rod to air, in the first version of this design, the coil was wrapped in high-temperature fiberglass insulation, then a single layer of glass. The Teflon base was also replaced with a slightly thicker PEEK base.

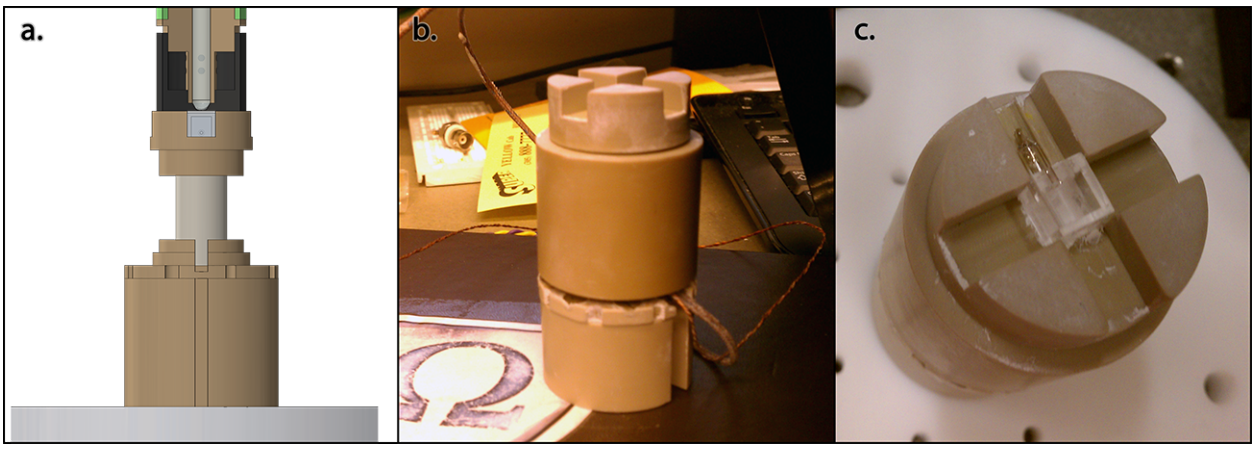


Figure 3.21: The version of the heater coil optimized for the larger "cuvette"-style cells. a.) A render of the heater *in-situ*, without coils or insulation. b.) A photograph of the wound and assembled heater c.) The cell in place on the heater; a small groove was added to one side of the heater cap to accomodate the cell's "stem".

When the microfabricated cell ($\approx 7\,\text{mm}\times4\,\text{mm}\times2\,\text{mm}$) was replaced with a TwinLeaf "cuvette"-style cell ($7\,\text{mm}\times7\,\text{mm}\times12\,\text{mm}$), the heater needed to be redesigned again to ac-

commodate the larger cell; the result is shown in 3.21. At this point, the smaller pulse coils were also replaced with a set of larger, more-homogeneous coils, relaxing the size constraint somewhat, allowing for more insulation around the heater. Because the microfabricated cell had only one optically-transparent surface, the original cap used small "stands" protruding from a mostly-flat cap to allow a wider array of input beam angles. The TwinLeaf cell was optically-transparent along all axes, and so the pump and probe beams passed through the cell normal to the glass surface, rather than at a 45 ° angle; as such, only enough material was removed from the cap to allow the orthogonal beams to pass through, to maximize the volume of insulating material between the heater and the environment and sample. Additionally, a PEEK base was added below the heater, to accommodate for the increased height of the pulse coils. To improve the insulation, the fiberglass-and-glass casing was replaced with a thicker, PEEK case. This version was less fragile (the glass tubes would often break, either from differential thermal expansion or from mishandling) and added thicker thermal insulation. Additionally, since it is opaque, it helped to minimize the escape of heat in the form of infrared radiation.

When the final version of the heater coil was first wound, the coil covered nearly the full length of the coil with 4 layers of wiring,[6] but this led to considerable noise on the magnetometer signal while heating, and so the coil was re-wound to cover only the bottom half of the core. This change did not seem to affect the apparatus's effectiveness. In earlier versions, the wires were initially held in place by superglue during winding,[7] but in the final windings, this was replaced with high-temperature silicone heat sink compound, to increase the thermal conductivity between the layers and hopefully increase the effectiveness of the outer layers.

3.5.2 Pulsed DC Heating

The initial design of heater circuit used a TTL-controlled pulsed DC heater. Leaving aside Johnson noise (which was not likely to be the limiting source of noise in these experiments), this would allow the magnetometer to operate without any potential stray fields from the heater. Since nearly all experiments done to date have had significant dead times for prepolarization and sample transfer, the heating pulses would not necessarily induce abnormally long experiment times.

The major downside, which ultimately made this approach impractical for these experiments, is that the cell cools over the course of the experiment, which serves to modulate many factors in the magnetometer's operation. This is particularly problematic for making measurements which require extremely high resolution where any single acquisition can go on for over a minute. This is somewhat problematic even in the case of short acquisitions, as temperature equilibration generally follows an exponential decay, and so much of the heat

[6] Only 2 layers of twisted pair wiring could fit between the coil and the inner diameter of the cap, so 4 layers were wound up the coil up to the point of the cap, then only two layers under the cap.

[7] The superglue would burn off at high temperatures, but by that point, the coils were tightly wound and/or held in place by a layer of fiberglass insulation, preventing it from spontaneously unwinding.

that will eventually be lost will be lost immediately after the heating is switched off.

Another issue with pulsed DC heating is that the mean steady-state temperature will depend on the duty cycle of the pulses - although generally all our heater circuits are configured as thermostats, equilibration after a change of duty cycle is not immediate, and this may introduce additional error into the measurements. With a sufficiently large heat reservoir and a long inter-pulse spacing, this is not an issue, but when sufficient time is not given for the temperature to stabilize after an experiment has finished, there will be some inevitable drift as the system finds a new equilibrium. Another strategy for dealing with this, however, is to maintain a constant duty cycle even when not conducting experiments, which allows the system to remain in equilibrium even when experiments start. This benefit, however, is tempered by the fact that this does not allow flexibility in the scheduling of experiments as the experiment timing is dictated by the heater's "off" period, rather than the heater's "off" period being dictated by the timing of the experiment.

3.5.2.1 Pulsed DC Heating Circuit

The circuit used for DC heating is relatively straightforward TTL-controlled DC pulse generator with output amplitude modulated by a temperature controller. The temperature controller's output is 1-10 V, and V_{PP} is 24 V, so to take advantage of the full range of the power supply, the gain in the voltage amplification stage (an op-amp configured as a non-inverting amplifier) is set to 2.4. The amplified voltage is fed into a Darlington pair of transistors for current amplification, which is directly connected to the heater coil. These transistors generally dissipate most of the waste energy, and so need adequate heat sinks.

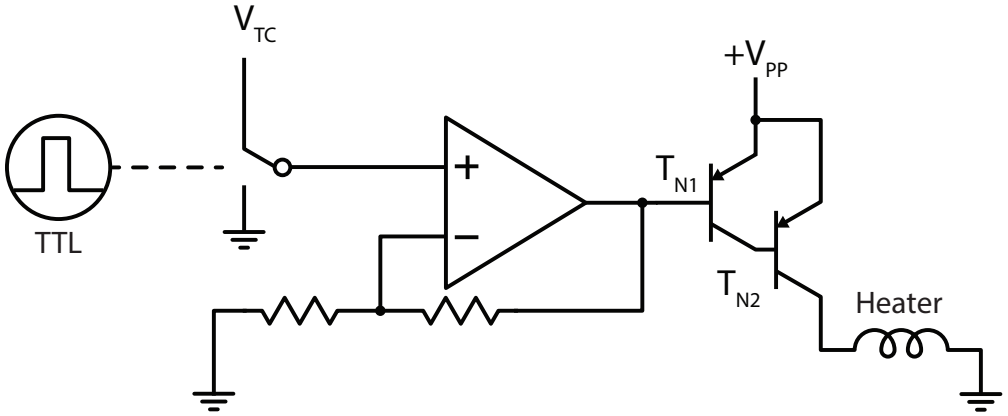


Figure 3.22: The TTL-controlled DC version of the cell heating circuitry.

The input to the op-amp is modulated by a TTL, so that the current can be interrupted during acquisition. This is achieved using a TTL-controlled single-pole double-throw (SPDT) switch, with the "normally closed" connection connected to the temperature controller output (V_{TC}), and the "normally open" connection connected to ground. Because the temperature controller modulates its output to maintain the temperature of the cell at the set temperature,

it is preferable for the "no signal" mode to revert the circuit to the temperature controller, rather than simply turning the current off. Configurations using a single-pole single-throw (SPST) configuration were insufficient because they left the input to the amplification stage floating, making it subject to various sources of noise. What leakage current and noise remains in the "grounded" state typically is filtered out by the Darlington Pair's diode drop, which imposes a voltage threshold condition on amplified current.

3.5.2.2 Intermittent cooling

The fact that the cell cools over the course of an experiment is a major hindrance to the use of pulsed DC heating, as the temperature of the cell has many direct effects on the signal. According to Newton's law of cooling, the rate of cooling is proportional to the difference in temperature. More precisely[25]

$$\frac{\mathrm{d}Q}{\mathrm{d}t} = hA\Delta T(t), \tag{3.19}$$

where Q (W) is the thermal energy, h ($^{\mathrm{W}}/_{\mathrm{m^2K}}$) is the heat transfer coefficient, A (m^2) is the surface contact area and ΔT (K) is the difference in temperature between the cooling object and the environment. The identity $C = {}^{\mathrm{d}Q}/_{\mathrm{d}t}$ can be substituted into Eqn. 3.19 to give an equation for the temperature change as a function of time:

$$\frac{\mathrm{d}T(t)}{\mathrm{d}t} = -\frac{hA}{C}\Delta T(t); \tag{3.20}$$

and the solution to the differential equation is

$$\Delta T(t) = \Delta T_0 e^{-hAt/C}. \tag{3.21}$$

As this is an exponential decay, most of the heat will be lost in the first part of the decay and so particular care would need to be taken with insulation to avoid any significant temperature drop at the cell. Unfortunately, because of the need for optical access, the cells are likely to be among the least insulated part of the heating apparatus, and because the sample needs to be kept relatively cool but also very close to the cell, it is likely to be near an actively cooled region - leading to a significant temperature gradient (high ΔT). It is also the only part of the magnetometer which needs to be kept warm, as the main downside of cooling is its effect on signal strength.

3.5.3 AC Heating

To avoid some of the problems described in the previous section, the heater was switched from DC to AC - this allows the heater to be operated continuously, so long as the circuit is driven at a frequency far outside the signal bandwidth. This removes the concern about intermittent heating, but leaves open the possibility of parasitic signals.

The AC signal was generated using a Wein bridge oscillator set to 15 kHz, output to a 40 W audio amplifier; the second amplification step was necessary because AC power is proportional the root-mean-squared value of the voltage and such requires a higher peak-to-peak voltage to apply the same power. Because the magnetometer was configured primarily for low-frequency ($<$1 kHz) signals, the frequency was chosen to be on the high end of the amplifier bandwidth, around $\approx$ 14 kHz.

Chapter 4

Experimental Control

4.1 Overview

Among the significant challenges presented by the design of scientific instruments, experiment control and data acquisition can often be among the most subtle. Consoles and console software serve as the interface between scientists and the experiments they're conducting, and if poorly designed can serve as a significant barrier to usability or even limit the types of experiments that can be done where no limits might exist in the instrumental hardware. In general, well-designed systems should feel natural, and it is often the case that if users are thinking about the system at all, it's because something has gone wrong. As such these challenges are both extremely important and non-obvious.

One important factor to consider in experimental control systems is general applicability. For many systems, potential use cases will change and expand as time goes on, and so while building an ad hoc system with limited utility may be cheaper and quicker in the short run, it may cause more work in the long run. In our lab, several magnetometers were in use at the time, each using non-ideal ad hoc systems for experimental control and data acquisition, and so this console and console software was designed to be duplicated for general use on all similar systems. Additionally, because these instruments are each unique, much effort was made to make the underlying back-end as modular as possible, so that similar software could be use to control different hardware.

Our experimental control system consists of two parts - a hardware console which controls the timing of the experiment, the data acquisition and has some analog output capabilities, and software to control the console - called (imaginatively) Magnetometer Controller. Our hardware controller uses a 100 MHz 24-channel USB PulseBlaster TTL generator for timing control (described in Sec. 4.2.1) and a USB-6229 BNC 16-bit, 250 $^{kS}/_{s}$ Multifunction DAQ for analog input and output (described in Sec. 4.2.2). The console software was written in ANSI C using LabWindows/CVI and is described in detail in Sec. 4.4.

4.2 Timing control and analog output

Timing is particularly important in NMR experiments, wherein the precision with which one is able to manipulate the spins is related to the precision in the length of an excitation pulse or delay. For experiments with more than one dimension, pulse and acquisition timing becomes a critical factor in the data themselves, and in RF experiments, receiver and pulse phase are determined by how well synchronized the two timing systems are.

Likely the most important type of uncertainty present in digital timing control consoles is the digitization limit - digital devices can only change state in response to a clock pulse, and so timing is quantized into units of clock cycles. For a 1 MHz CPU, timing cannot be controlled at a level finer than 1 μs, whereas a 100 MHz CPU potentially has 10 ns resolution. For a 1 G DC pulse, this is a difference between 0.12 and 0.0012 ° pulse resolution, which can certainly add up when applying large numbers of pulses. Generally uncertainty in the absolute timing is small relative to the digitization error, and are likely to be swamped by timing errors propagating in the controlled devices.

4.2.1 Spincore Pulseblaster

Experimental timing is all provided by a single 24-channel 200 MHz USB PulseBlaster TTL generator from SpinCore, controlled by the SpinAPI C interface. Pulse sequences consist of an array of pulse instructions, which specify the state of each TTL at each clock cycle. Each instruction consists of a 24-bit word describing the on-off state of each TTL, a 32-bit unsigned integer specifying the number of clock cycles before the next change in state, an 8-bit integer specifying the pulse instruction.

4.2.2 NI-DAQ Analog Output

The data acquisition system is based on a USB-6229 M-series multifunction DAQ system from National Instruments. In the current configuration of the instrument, its 4 analog outputs are used to set the DC levels of various pulse programming channels - the pulse amplifier is generally configured with one “fixed” DC level and one floating - with the floating channel controlled by the DAQ, though this is primarily because the USB-6229 has only 4 outputs - there would not be an appreciable increase in the noise or stability were other outputs to be used. The analog output is capable of being programmed for shaped pulses, but at the time of this writing this functionality has not yet been implemented in the software, as it was not necessary for the experiments we performed.

4.3 Data Acquisition

4.4 Software

The main software used for experimental control was the Magnetometer Controller program, written in C using LabWindows/CVI. LabWindows was chosen as it has a more versatile development backend than LabView, but similar support for control of scientific instrumentation and simple interface for GUI building. In earlier versions of the program, a single event triggered the full data acquisition, preventing the use of any windowed acquisition sequences; in the latest version of the program, the acquisition of a single data point is triggered by a retriggerable internal clock waveform, allowing for arbitrary acquisition sequences.

4.4.1 Pulse Programming

4.4.1.1 Overview

One of the core essential functions of the console software is to serve as a simple graphical user interface for pulse programming. To define terminology, in this text, the pulse program refers to the entire set of instructions required to run an experiment - this includes both the timing and pulse controls as well as the analog output and acquisition settings. In this case this must be distinguished from the set of timing instructions executed during each acquisition[1]. The term "experiment" in this context refers to the entire set of acquisitions described by the pulse program - which is not to be confused with a single acquisition (which, again, also has a plausible claim to the term "experiment").

Each experiment steps through a series of acquisitions as dictated by the pulse program. At the beginning of each acquisition, the program loads a series of timing instructions into the memory of the pulse controller, initializes the values for the DAQ analog outputs and sets up an acquisition in the analog input channel of the DAQ. Generally there is much greater uncertainty associated with the timing of software triggers sent via USB than with either the DAQ or controller's internal clocks, and so it is preferable to set the DAQ waiting for a hardware trigger, which is then provided by the pulse controller - thus synchronizing the pulse timing with the acquisition. As such, the DAQ is always initialized to wait for a trigger[2] and the DAQ acquisition task can be started immediately. The acquisition task is always started *before* the pulse controller, as it ensures that the DAQ will be waiting for the trigger before it's issued by the pulse controller.

[1] Occasionally pulse programs do not make use of a set of pulse timings - this is frequently the case when acquiring data from a continuous source like a test signal. It is presumably a semantic question as to whether a null set of instructions can be considered a set of instructions - for simplicity's sake, we shall assume that in this context it is.

[2] This is still done even when no pulse controller is used by configuring the DAQ to trigger from its own internal clock

```
double ** RunAcquisition() {
    if(requires_scan) {
        SetupDAQTask();                                    // Configure the DAQ
        StartDAQTask();                                    // Start the DAQ waiting for a
            trigger.
    }

    if(pulse_controller_used) {
        pulse_controller_status = RUNNING;

        StartPulseController();
        while(pulse_controller_status != STOPPED) {        // Wait for the pulse program
            to finish executing.
            Wait(10 ms);
            pulse_controller_status = GetPulseControllerStatus();   // Query the board
                for its status.
        }
    }

    if(requires_scan) {
        data[current_step] = ReadDoubleData();             // Read data as an array of
            doubles.
        StopDAQTask();                                     // Stop the task so that it can
             be re-created
                        // in the next loop.
    }

    return data;
}
```

Figure 4.1: C-style pseudocode algorithm for each acquisition.

The general algorithm seen in 4.1 is executed in an asynchronous `while` loop which checks both whether the experiment has finished its execution and whether the user has issued a stop command, prematurely terminating the operation. The parameters for each acquisition are stored in a `cexp` (current experiment) structure, and are updated with each experiment.

Listing 4.1: The cexp or *current experiment* structure, which contains the parameters of a given experiment.

```
typedef struct CEXP {
    TaskHandle aTask;       // Signal acquisition task
    TaskHandle cTask;       // Counter task.

    int update_thread;      // Update thread id.

    PPROGRAM *p;            // Current program

    int nchan;              // Number of channels.
    int t_first;            // Transients first or indirect aq first?
                            // Three options:   0 -> ID first, then advance transients.
                            //                  1 -> All transients first, then ID
                            //                  2 -> Phase Cycles first, then IDs, then
                                 repeat as necessary

    char *path;             // Full pathname of where to save the data
    char *fname;            // Experiment filename.
    char *bfname;           // Base filename
```

```
    unsigned int num;          // Experiment number.
    char *desc;                // Description of the experiment.

    dstor data;                // Cached data
    dstor adata;               // Cached data average

    char *ctname;              // Counter task name
    char *atname;              // Acquisition task name

    char *ccname;              // Counter channel name
    char **icnames;            // Input channel names

    int ctset;                 // Whether or not the counter task has been set
    int atset;                 // Whether or not the acquisition task has been set.

    int cind;                  // Current index
    unsigned int steps_size;   // Number of values in *steps
    unsigned int *cstep;       // Same size as steps
    unsigned int *steps;       // Maxsteps -> It will have one of three forms:
                               // 1: !p->varied -> steps = [nt]
                               // 2: MC_TMODE_ID [{dim1, ..., dimn}, nt]
                               // 3: MC_TMODE_TF [nt, {dim1, ... dimn}, nt]
                               // 4: MC_TMODE_PC [{pc1, ... , pcn}, {dim1, ..., dimn}, nt/
                                   npc]


    int ninst;                 // Number of instructions in this run
    PINSTR *ilist;             // List of instructions for this run.

    time_t tstart;             // Time started
    time_t tdone;              // Time that the most recent part was completed.

    double t_el;               // Elapsed time
    double t_prog;             // Amount of time taken by just the pulse program
    double t_rem;              // Remaining time on the pulse program
    double t_tot;              // Total time the program should take

    // Analog output info
    TaskHandle oTask;          // Analog output task

    char *otname;              // Analog output task name

    int otset;                 // Whether or not analog output task is set
    int nochans;               // Number of analog output channels.
    int *ochanson;             // Analog output channels on. Size = nochans, Refers to
        indices in p->ao_chans
    char **ocnames;            // Analog output channel names. Size = nochans.
    float64 *ao_vals;          // Analog output values for each channel. Size = nochans;

    int64 hash;                // Current experiment hash - generated from the experiment
        name.
} CEXP;
```

Experiments can be multi-dimensional, include multiple transients and use multiple phase cycles, and so there are multiple approaches to the order of sampling (i.e. the conversion between our linearly-stepped index and the position in the multi-dimensional sampling space). Different experiments will call for a different order - for example, if drift over the course of an experiment is important, it may be preferable to acquire all points along a given dimension first, then repeat the entire sequence multiple times rather than acquiring all transients before moving on to the next step in the indirect dimension; on the other hand, if the noise

characteristics tend to change but the signal is fairly constant, it may be preferable to sample the full phase cycle immediately, so that the noise will optimally cancel out. Although it makes things slightly more complicated, it was sufficiently important to address these different cases that the software is written such that three sampling options are available: indirect dimensions first [default], transients first, phase cycles first. In the first option, all points along indirect dimensions are acquired first before moving on to the next transient; in the second option, all transients are acquired before moving on to the next step in the sampling space; in the third option, the minimum number of transients required to execute a full phase cycle are performed, then the experiment is moved to the next step in the sampling space and that entire block is repeated to satisfy the specified number of transients.

4.4.1.2 Simple Pulse Programs

The core element of the experiment is a set of pulse instructions. The simplest pulse programs consist of a single set of instructions executed with or without an acquisition. These instructions are specified using a graphical interface to the Spincore PulseBlaster. Each instruction consists of 24 flags specifying the on/off state of each TTL (only 23 are user-accessible and 1 is reserved for acquisition triggering), the duration of the instruction (maximum value 21 seconds), the instruction type and a data parameter used for some instructions.

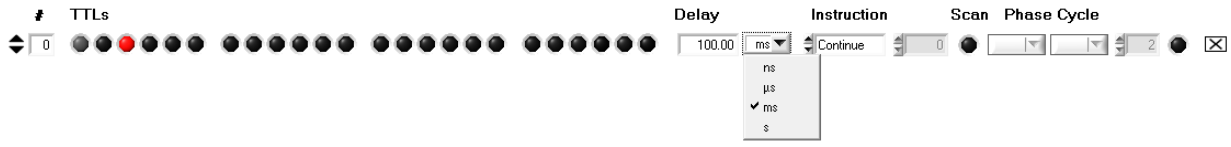


Figure 4.2: A single pulse instruction.

There are 8 instruction types which can be programmed into the board, which are detailed in Table 4.1 - each instruction can be no shorter than 100 ns and no longer than 21 s[3], with 10 ns resolution. For delays longer than 21 s, the *Long Delay* instruction type can be used, which repeats a single instruction n times, where n is specified using the instruction data parameter. While there was some early work done on establishing a mechanism for specifying custom instructions, which would be useful for updating things like pulse length calibrations and the like, no such system has been implemented at the time of this writing.

The basic set of pulse instructions is specified on the *Pulse Program* tab. A basic example program is shown in Figure 4.3 - this is a pulse program in 8 steps. The first step is a 10 s pre-polarization time, implemented as a *Long Delay* which repeats 1 s units 10 times. I have generally found it preferable to use durations under 5 s when possible. The second instruction turns on the leaving solenoid (TTL 1[4]) while leaving the sample in the magnet

[3]Specifically, the instructions can be no longer than $10 \cdot 2^{31}$ ns. Though it is not specified in the promotional materials, this is likely because the clock cycles are counted using a signed 32-bit integer, which has a range of $\pm 2^{31} \approx 2.1 \cdot 10^9$; as clock cycles are 10 ns long, this gives a limit of ≈ 21 s.

[4]TTLs are numbered using a zero-based index.

Instruction	Data	Description
Continue	Unused	Set TTLs then wait specified duration.
Stop	Unused	Stop the pulse program - does not update TTLs before instruction, and TTLs remain at previoiusly-specified values
Loop	Number of iterations	Start a loop - execute this instruction and all instructions up to and including the instruction specified *End Loop* whose instruction data parameter matches
End Loop	Loop instruction	The final instruction to include in a loop started by the *Loop* instruction at the position specified by the instruction data parameter.
JSR	Instruction	Jump to a subroutine which starts at the instruction specified int he data parameter.
RTS	JSR Instruction	Return from a subroutine to the execution of the program.
Branch	Instruction	Sets TTLs to specified levels, waits duration, then jump to the instruction specified by the instruction data parameter. This is generally used to loop indefinitely.
Long Delay	Number of repetitions	This is equivalent to a *Continue* instruction, except that it is repeated a number of times specified in the instruction data parameter. It is used for single-instruction delays longer than 21 seconds, as single instructions cannot be longer than 21 seconds.
Wait	Unused	Sets TTLs to specified levels, waits the duration specified, and then wait for a hardware or software trigger. This cannot be used as the first instruction, and must occur at least 100 ns after the program begins.

Table 4.1: Spincore Pulseblaster instruction types.

for an additional 10 ms (vacuum is controlled with TTL 2). The next instruction is the shuttling time - during this instruction a back-pressure of air is turned on along with the pulse coil along the **z** direction(TTL 10) - these are left on for a transit time of 700 ms. The fourth instruction is a 50 ms delay allowing the magnetometer spins to re-polarize before any measurements are taken, this is followed by a scan trigger (TTL 0), which is configured at 20 μs. The sixth and seventh instructions are the measurement loop - 16 180 $^\circ$ pulses are applied at 100 ms intervals. The penultimate instruction is used to shuttle the sample back into the pre-polarizing magnet, and the final instruction stops the pulse sequence; because *Stop* instructions do not change the TTL state before stopping the program, the configuration specified in the previous instruction will persist after the instructions have been executed.

Beyond the triggering, the acquisition parameters are configured in the *Prog. Config* tab - shown in Figure 4.4 for our example pulse program. The sampling rate is set to 1 kHz as that is the value of the low-pass filter on the preamp so that no aliasing occurs. The acquisition

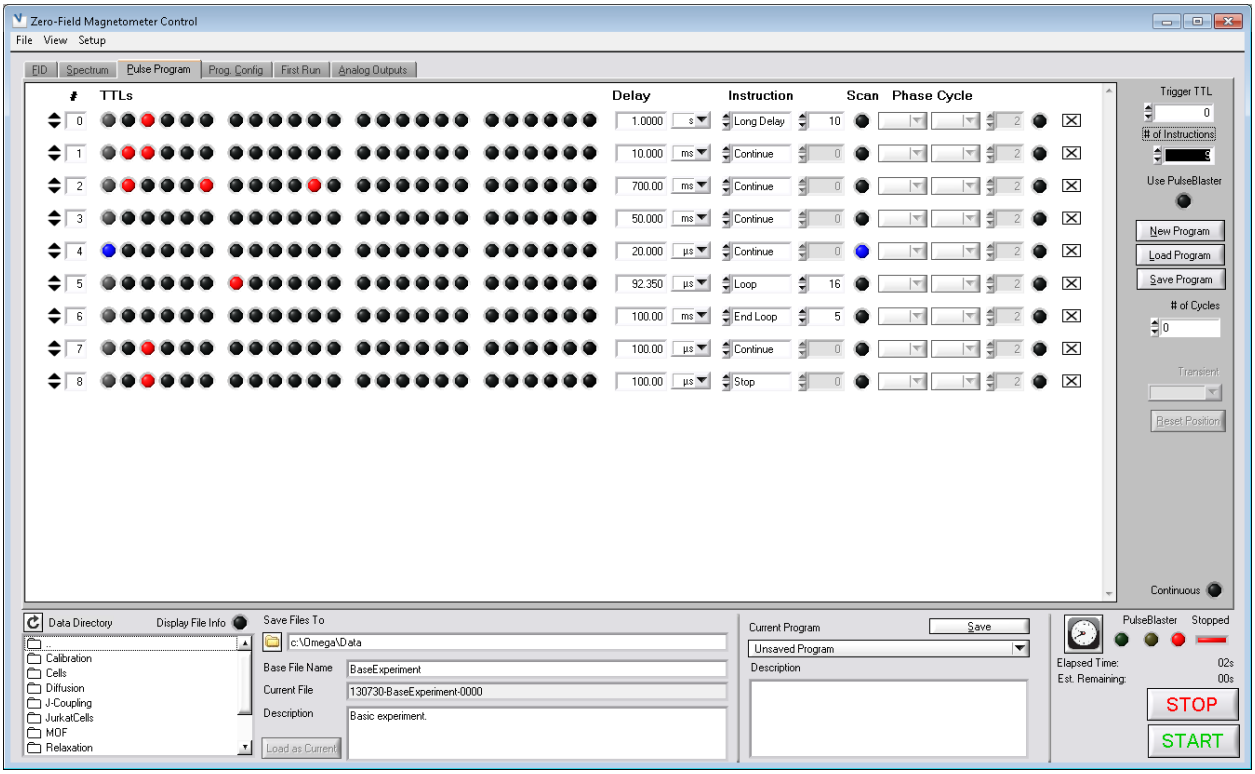


Figure 4.3: The *Pulse Program* tab for a basic experiment.

time, number of points and sample rate are related by the equation $NP = AT/SR$, and so setting any two uniquely determines the third, in the interface, right-clicking on one of these three boxes pulls up a context menu allowing the user to specify which of the other two quantities to hold constant when updating the selected box; since the sampling rate is generally related to the signal bandwidth and analog filtration, it's usually best to configure number of points and acquisition time to hold the sampling rate constant. The number of transients box determines the number of times each step in the acquisition (in any number of dimensions) will be repeated. This value is constrained by the phase cycling, which is discussed in Section 4.4.1.3.

The other boxes in the acquisition configuration refer to the parameters of the actual devices, and are specific to Spincore and National Instruments devices[5]. The *Device* and *PB Dev.* pulldowns are used for selecting between multiple devices plugged into the same computer and generally will only have one option. Up to 8 input channels[6] can be selected via the *Input Channels* pulldown - channels that are turned on will be marked with a dot, and selecting these will toggle them off. Each channel can be configured to a different sensitivity;

[5]the program may work for other devices as generic functions are used, but they have only been tested with Spincore's 24-TTL USB Pulseblaster and the National Instruments USB-6229M devices

[6]This is a software limit imposed to limit the number of LED controls on the data display, and could easily be expanded if necessary. I have never used more than 2 channels at once.

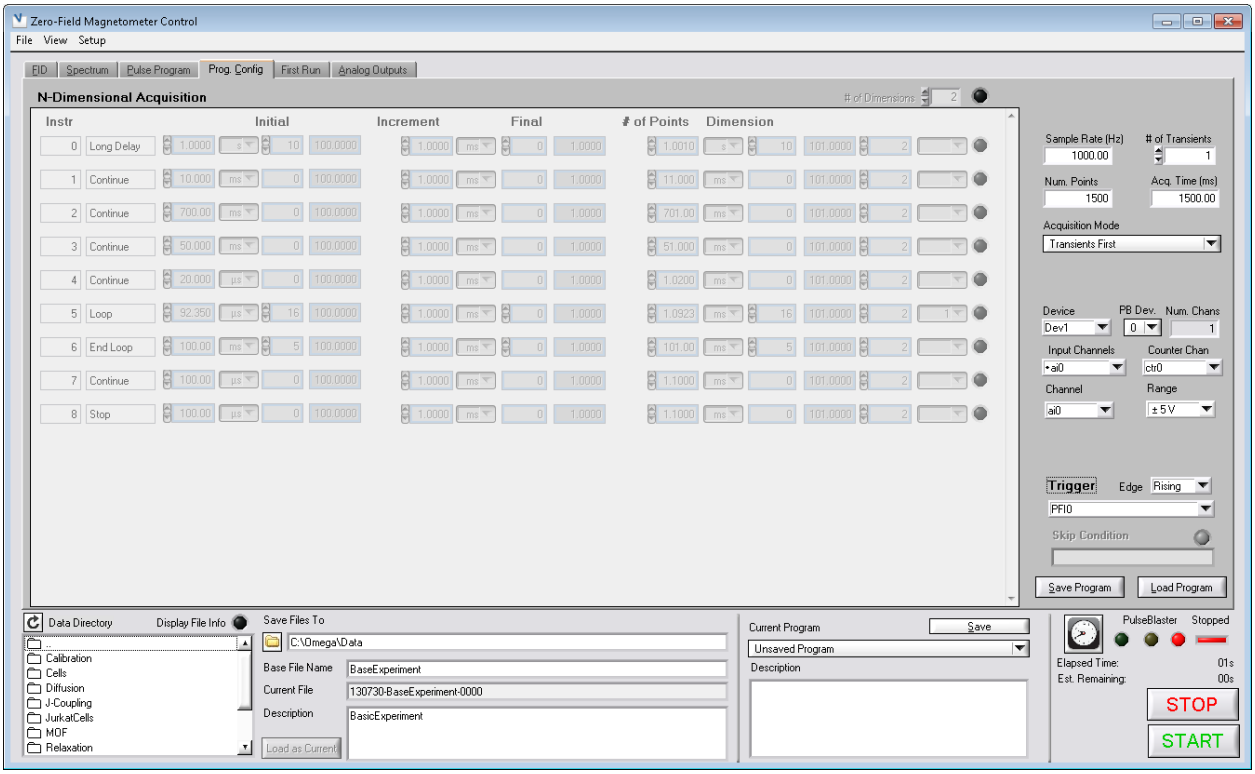


Figure 4.4: The *Prog. Config* tab for a simple 1D program.

USB-6229M DAQs support only 4 possible ranges: ±0.1 V, ±1 V, ±5 V and ±10 V. The *Channel* control selects which channel's sensitivity is being configured.

As described in Section 4.4.1, the acquisitions are initialized before the pulse programmer is activated and waits for any number of hardware triggers. There are a number of potential channels for triggering, but may of these are virtual devices. When triggering using a PulseBlaster, it is best to use one of the 4 digital input channels (PFI0-PFI3); when using an internal trigger, the 20MHzTimeBase or 80MHzTimeBase virtual channels tend to work well. To allow the task to be retriggerable, a virtual counter channel is used to trigger individual the acquisitions of actual points; the NI USB-6229M DAQ supports 2 of these virtual channels, which are named ctr0 and ctr1, and which one of these is used can be selected using the *Counter Chan* control.

To avoid the possibility that data would be lost, data are automatically saved to a new file after each acquisition. The location of these files can be specified easily using the embedded text boxes at the bottom of the magnetometer controller window. The filenames are built automatically using the current date and the specified base experiment name, currently this is set with a hard-coded definition in the library `DataLibPriv.h`, and is set to `MCD_FNAME_FORMAT = %s-%s-%04d` with the first string specified by the date format `MCD_FNAME_DATE_FORMAT = %y%m%d`, the second string is the base filename and the integer is the first number which would not overwrite an existing file; this could easily be moved into a preferences dialog and changed to a user-

controlled setting.

4.4.1.3 Phase Cycling

Phase cycling is a technique which in general is used to control the phase coherences producing signal in an experiment. The most common application of phase cycling is to correct for artifacts in the system which are independent of the relationship between the phase of the transmitter and receiver - the quintessential phase cycling sequence is the CYCLOPS[26] method (shown in Fig. 4.5), where the phase of the transmitter phase over a multiple of 4 transients follows the sequence x, -x, y, -y, while the receiver follows 90° behind. When the transients are averaged, signals whose phase relative to the receiver add, while all others average out, leaving things like NMR precession (whose phase is always 90° away from the transmitter phase) and eliminating signals such as detector asymmetries and signal offsets.

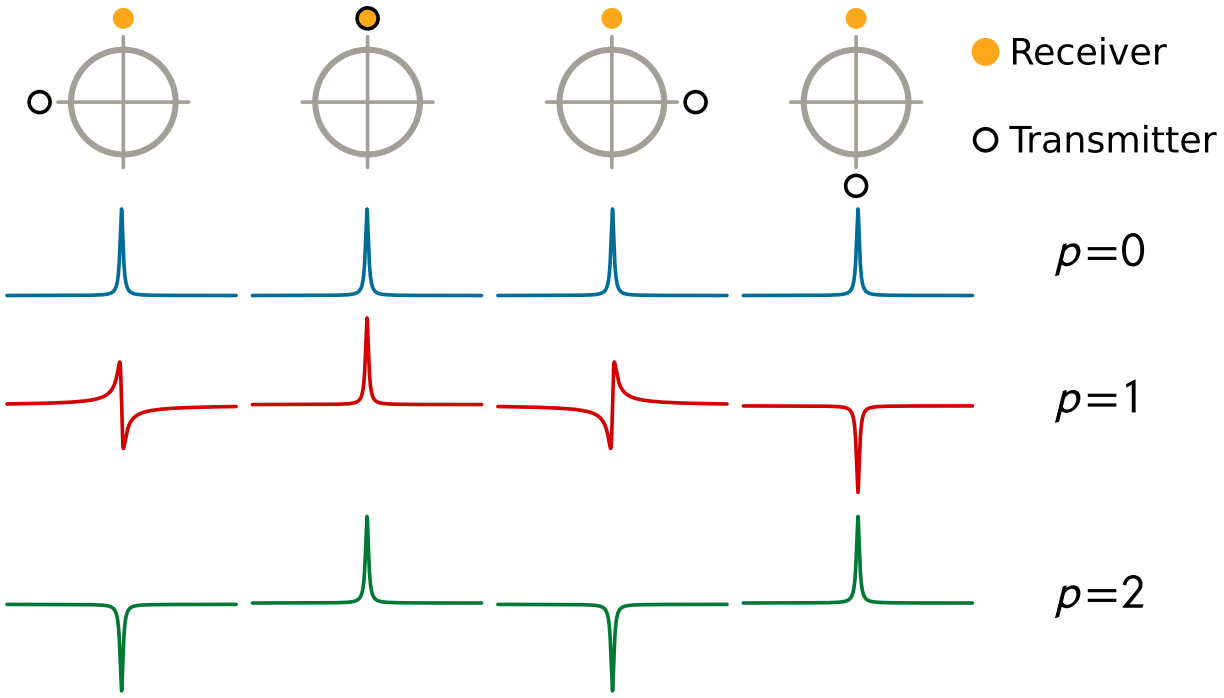


Figure 4.5: A demonstration of the CYCLOPS phase cycling sequence. Over the course of four transient acquisitions, the transmitter phase is varied while the receiver phase is held constant. In this scheme, coherences of order $p \neq 0 \mod 4$ average out, while coherences with order $p = 0 \mod 4$ are unaffected.

This same concept still has significant applicability at zero field, where phase corresponds to a direction in the lab frame, but phase encoding can be somewhat more complicated in a regime with a carrier frequency of zero and two distinct methods of phase cycling need to be implemented. One use for phase cycling for low-frequency acquisitions is to remove discrete noise, such as the ever-present 60 Hz noise and its many harmonics. When the acquisition is started at a random time with respect to a source of discrete noise, the noise

averages out over multiple transients due to changes in the phase of the acquired waveform, which averages away slightly slower than $1/\sqrt{N}$ for a random distribution of phases. If the experiment is instead triggered using a source synced to the problematic discrete noise, the phase of the noise can be deliberately chosen such that it will average out over the course of the acquisitions. Choosing n phases evenly sampled from the interval $[0, 2\pi)$, the signal contribution from the m^{th} harmonic is given by:

$$\sum_{k}^{n} \cos\left(2\pi \cdot k \cdot m/n\right) \tag{4.1}$$

Which sums to 0 for all harmonics $m \not\equiv 0 \bmod n$. In the console software, this is generally accomplished by using a `Wait` instruction followed by a `Continue` instruction with a duration which is varied to sample all multiples of $1/nt \cdot f$ where f is the frequency of the noise source and nt is the number of transients acquired in the cycle; the hardware is then triggered on a function generator synced to the noise source. In general, it is preferable to sample in phase-alternated pairs so that drift or other changes over time do not interfere with the phase averaging (e.g. for a four-phase cycle it is preferable to sample 0,π, $\pi/2$, $3\pi/2$ as the maximum cancellation happens between pairs separated by a phase of exactly π). This helps to minimize the effect of a trigger source which is imperfectly synchronized with the noise source; as the time between experiments is often several orders of magnitude longer than the period of the noise source, even small deviations from the ideal frequency can grow quickly and reduce the effectiveness of this technique. By choosing phase cycles in pairs, the phase drift between experiments is limited to the length of one experiment.

The other primary use of phase cycling in the context of zero-field NMR would be to eliminate potential artifacts due to pulse direction. This may be useful in a number of contexts where the absolute phase of the experiment is chosen arbitrarily, such as the quadrature and vector detection schemes described in Sections 5.4.2 and 5.4.3 respectively. In the vector detection scheme, for example, measurements can be made $x - x - y - y$ or $y - y - x - x$, with some signal decay in whichever is detected second; by phase cycling between the two possibilities, the decay artifact can be reduced or eliminated.

The interface to specify phase cycling in the console is built into the basic pulse program instruction GUI and is shown in Figure 4.6. A *phase cycle* here refers to a full repetition of all the chosen phases; most experiments with phase cycling will have exactly one phase cycles, but it is possible to compound phase cycles together as is necessary for some coherence pathway selection sequences, or to eliminate discrete noise *while* adjusting for directional artifacts. Phase cycling can be enabled using the LED control to the far right of each instruction, doing so enables a pulldown control and automatically increments the number of cycles to 1 if it is set to zero.

Phase cycling on an instruction is enabled using an LED control (labeled **D** in Fig. 4.6), which allows a different instruction to be set for each step in the phase cycle. This can be done either by selecting and modifying the instructions one at a time by cycling through the pulldown labeled **A**, or by expanding the instruction to see all steps by using the expander

Figure 4.6: The initial state of a phase cycled instruction when phase cycling is turned on. The instruction to display is selected using the pulldown labeled **A**. **B** is used to select the phase cycle along which this instruction should vary. **C** sets the number of steps in the phase cycle (and updates all other instructions varying along the same cycle). **D** toggles phase cycling on and off, and **E** is used to collapse or expand the full phase cycle view.

control labeled **E** - bringing up the interface shown in Fig. 4.7. The number of steps in a cycle can be set using the control labeled **C**, which changes the number of steps for all instructions with the same cycle. To add additional cycles, the "number of cycles" control on the *Pulse Program* tab must be incremented. The phase cycle along which a given instruction is varied is set by the control labeled *B*.

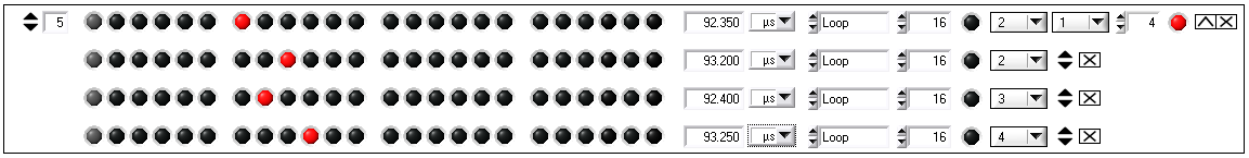


Figure 4.7: The expanded view for a 4-step phase cycle, which cycles a π pulse between x, y, $-x$ and $-y$.

Because of the difficulty of implementation and the low need for such a thing, it is currently not possible to vary a single instruction both along an indirect dimension and as part of a phase cycle. In most cases it is possible to replicate this functionality using two separate instructions, however. Because the actual type of instruction can be varied as part of a phase cycle, it is possible to use a JSR instruction to specify a different subroutine for each cycle, and the instructions in that subroutine can be varied along an indirect dimension if need be. This functionality has not been tested, however, and so it is not clear what its versatility is. Because it is not possible to simultaneously view the *Pulse Program* and *Prog. Config.* tabs, instruction numbers for phase cycled instructions are highlighted in red on the *Prog. Config.* tab as a reminder.

As phase cycling is intended to be averaged out, the phase cycle step is encoded in the transient step; as such, the number of transients in an experiment must be an even multiple of $\prod_i^{nc} ns_i$ where nc is the number of phase cycles and ns_i is the number of steps in cycle i. When the number of phase cycle steps is changed the number of transients is automatically changed to be the closest multiple to the previous value of the transient and the control is changed such that the up and down arrow keys increment the number of transients by the minimum number of transient steps.

4.4.1.4 Multidimensional Acquisition

Many experiments we've been interested in performing are multi-dimensional experiments, such as the relaxometry and diffusometry experiments described in Chapter 6, which are performed in a second, indirect dimension. These acquisitions are again specified using a

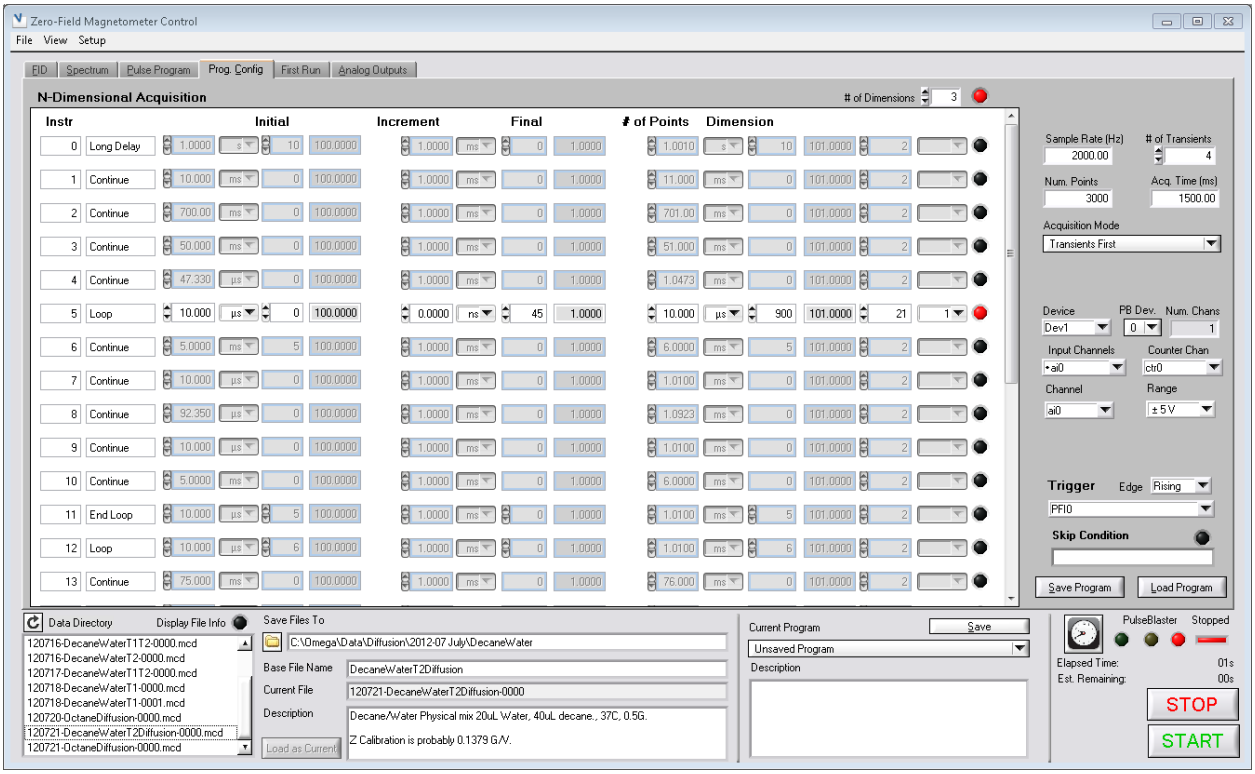


Figure 4.8: The *Prog. Config.* tab for a 3-dimensional experiment for running a T_2-diffusion experiment in an indirect dimension. The mis-positioning of the labels is due to a cosmetic bug that had not been fixed at the time of this writing.

graphical user interface which allows for a number of types of variation. Multi-dimensional acquisitions are mostly configured on the *Prog. Config.* tab, and can be enabled with the *# of Dimensions* control and associated LED. When the LED is enabled, the ND control panel is enabled, and the ND-state of the instructions can be toggled (see Figure 4.8). In this control, the direct dimension is considered one of the dimensions.

The *N-Dimensional Acquisition* panel contains a list of the instructions each with an associated LED control (**K** in Fig. 4.9) for toggling the variation mode. There are three possible modes: no variation [0], linear variation [1] and equation-based variation [2]. Left-clicking toggles in a positive cycle ($0 \rightarrow 1 \rightarrow 2 \rightarrow 0$), and right clicking toggles in the opposite direction ($0 \rightarrow 2 \rightarrow 1 \rightarrow 0$). These LEDs are color-coded, with a red control indicating linear variation and blue indicating equation-based variation.

Each instruction can be varied either in the delay, data or both. An example of both a linear and equation-based instruction is shown in Figure 4.9. In some cases, multiple instructions need to vary along the same direction, so the dimension of variation can be specified using a pulldown control (**J**); the number of steps in the selected dimension can be updated from control **I** on any instruction varying along the specified dimension.

Each instruction is divided into three components, the initial value (**A**), the increment value(**B**) and the final value (**C**). Since there are three controls and only 2 degrees of freedom,

Figure 4.9: A pulse instruction for linear variation (top) and equation-based variation (bottom). Labels are described in the text.

when one is changed, at least one of the other two is recalculated. In the case of linear variation, the update is hard-code such that changing the initial value or increment updates the final value, and changing the final value updates the increment. This is hard-coded into the callbacks `ChangeInitOrFinal` and `ChangeInc` in the primary function library `MC10.c` and can be easily changed in the code. In equation-based variation, both the initial and final values are calculated from the equation, and so both are updated - these values can also be used in the equations, and so caution must be taken to ensure that any variables used are not changed upon evaluation[7]. While in the *Pulse Program* instructions an instruction can be no shorter than 100 ns, this restriction is lifted in the multi-dimensional case to allow for instructions to be omitted entirely - any instruction length shorter than 100 ns will be skipped, as will `Loop` and `Long Delay` instructions of length 0. The upper limit of 21 s is still enforced. Increments can also take on negative values as they represent a change in time rather than an absolute time.

Each value specification in Figure 4.9 is broken into three components, the duration (**D**), data parameter (**E**) and instruction length (**F**). The duration can be specified in any units from nanoseconds to seconds and changing the unit specifier will change the value in the numeric control such that the total duration does not change - which is in contrast to the unit specifier on the *Pulse Program* tab, which in fact keeps the numeric control constant and changes the real duration; while it is often not ideal GUI design to have identical-seeming controls with different behavior, these choices were made based on the most common use case in each context. The **F** control is not user-specified, but rather is an experimental control which attempts to calculate the total duration of a given instruction. This was implemented to make more obvious the duration of a loop instruction whose data parameter is varying.

Figure 4.10: A pulse instruction where the data parameter is varied according to an equation. In this case, stepping through powers of two.

For equation-based variation, I wrote a math parsing library which reads in a small number of variables. When the equation has been entered into controls **G** and/or **H**, it is

[7]One such example would be the equation `init+3`, which sets all values - including the initial value - to be 3 units greater than the initial value. This will result in the initial value being incremented by 3 upon each evaluation

evaluated for all the real values in the experiment[8] and checked for errors. Functions which have been accepted with no errors will display a green background (as seen in Fig. 4.10), while functions with errors will display a red background. The variables available for use in the equations are detailed in Table 4.2, and should be displayed as a tooltip by stalling the mouse over an expression control.

Variable	Description
`nd`	The total number of dimensions (including the direct dimension)
`nc`	The number of phase cycles
`x` / `step`	Two aliases for the current step in the dimension [unused in skip equations]
`ccs[n]`	Current step for phase cycle `n`. Again, a zero-based index is used, so `n` spans from 0 to `nc-1`.
`mds[n]`	Maximum number of steps in dimension `n`.
`mcs[n]`	Maximum number of steps in phase cycle `n`.
`init`	The initial value specified in the instruction [not used in skip equations]. Since this control is updated from the equation, care must be taken to ensure that at step 0, this evaluates to exactly `init`.
`final`	The final value specified in the instruction [not used in skip equations]. As with `init`, care must be taken to ensure that the equation evaluates to `final` at the last step.
`us`	Durations are specified in nanoseconds, so this is an alias for 10^3, to specify μs.
`ms`	Alias for 10^6, for specifying milliseconds.
`s`	Alias for 10^9, for specifying seconds.

Table 4.2: Variables available for math parsing applications during pulse programming.

Most arithmetic and boolean operations are supported, with the order of operations detailed in Table 4.3. Trigonometric functions such as sin() and cos() are not implemented, nor are complex functions. Equations always return doubles which are rounded down by conversion to `int` when used in the data parameter; boolean true returns 1.0 and boolean false returns 0.0, while all non-zero values evaluate to boolean true. Using boolean operations and comparators, it is possible to create rudimentary conditional statements; for example, to implement a delay which increases by 300 ms each step until it reaches 1.5 s with the algorithm described in this pseudocode:

```
if(cds0*300*ms <= 1.5*s) {
    return cds0*300*ms;
} else {
    return 1.5*s;
}
```

[8]This can take some time, as the math parser has not been optimized, and relies heavily on redundant and costly string-processing operations. I have written a new algorithm for much faster math parsing, but it has yet to be implemented.

,

the equation would be set as:

```
(cds0*300*ms <= 1.5*s)*(cds0*300*ms) + (cds0*300*ms > 1.5*s)*(1.5*s)
```

.

Since these are mutually exclusive and span all cases, this is equivalent to the above pseudocode, as either the comparison on the left evaluates to 1.0 and the comparison on the right evaluates to 0.0 or the opposite is true.

Operation	Order	Description
`()`	1	Parentheses for equation grouping.
`exp()`, `log()`	2	Exponential and natural logarithm functions
`^`	3	Exponentiation
`*`,`/`, `%`	4	Multiplication, division and modulo
`+`,`-`	5	Addition and subtraction
`&`,`|`	6	Bitwise AND and OR
`=`,`!=`	7	Comparators = and ≠
`>=`, `>`, `<=`, `<`	7	Comparators ≥, >, ≤ and <
`!`, `&&`, `||`, `<`	8	Boolean operators: Negation, AND and OR.

Table 4.3: The order of operations and supported syntax. Operations are executed according to the *Order* column, generally top-to-bottom.

4.4.1.5 Subsampling

One problem with the multi-dimensional acquisition setup is that it is designed to only acquire along a grid (whether linear or non-linear), when often it would be preferable to sample only a subset of such a grid. For example, in T_1/T_2 acquisitions, values are generally sampled along each of the two dimensions from 0 s to $3{\cdot}T_1$ or $3{\cdot}T_2$ and signal varies according to $e^{-t_1/T_1}e^{-t_2/T_2}$ where t_1 is the time step in the T_1 dimension and t_2 is the time step in the T_2 dimension. At the point where $^{t_1T_2+t_2T_1}/_{T_1T_2} \geq 3T_1T_2$, there is nearly no signal - which in the case of the standard sampling pattern is fully half of the points in the sampling space. What's worse, it is the *longer* half; for $T_1 = T_2 = 1.5$[s], sampling 25 points in each direction from 0 to 4.5 s with a 10 s re-polarization time and a 2 s measurement time takes 2 h51 m53 s, but when all values such that $t_1T_1 + t_2T_2 \geq 3 \cdot T_1T_2$ are skipped, the same experiment can be acquired in only 1 h21 m15 s, saving an hour and a half while discarding only transients which are contributing very little to the signal.

This functionality is implemented in the GUI in the form of *Skip Expressions*, which is an optional box on the *Prog. Config.* tab in which an expression can be entered. The same parser described in Section 4.4.1.4 is used to process these equations, and the syntax and relevant variables can be found in Tables 4.3 and 4.2 respectively. The equation is evaluated as a boolean value for each point in the acquisition with all non-zero values evaluating to

boolean true; if the skip expression evaluates to true for a given point in the acquisition space, this point is skipped. These boolean values are stored in an array and saved with the data to allow the reconstruction of the original acquisition grid during data processing.

4.4.1.6 Analog Output Controls

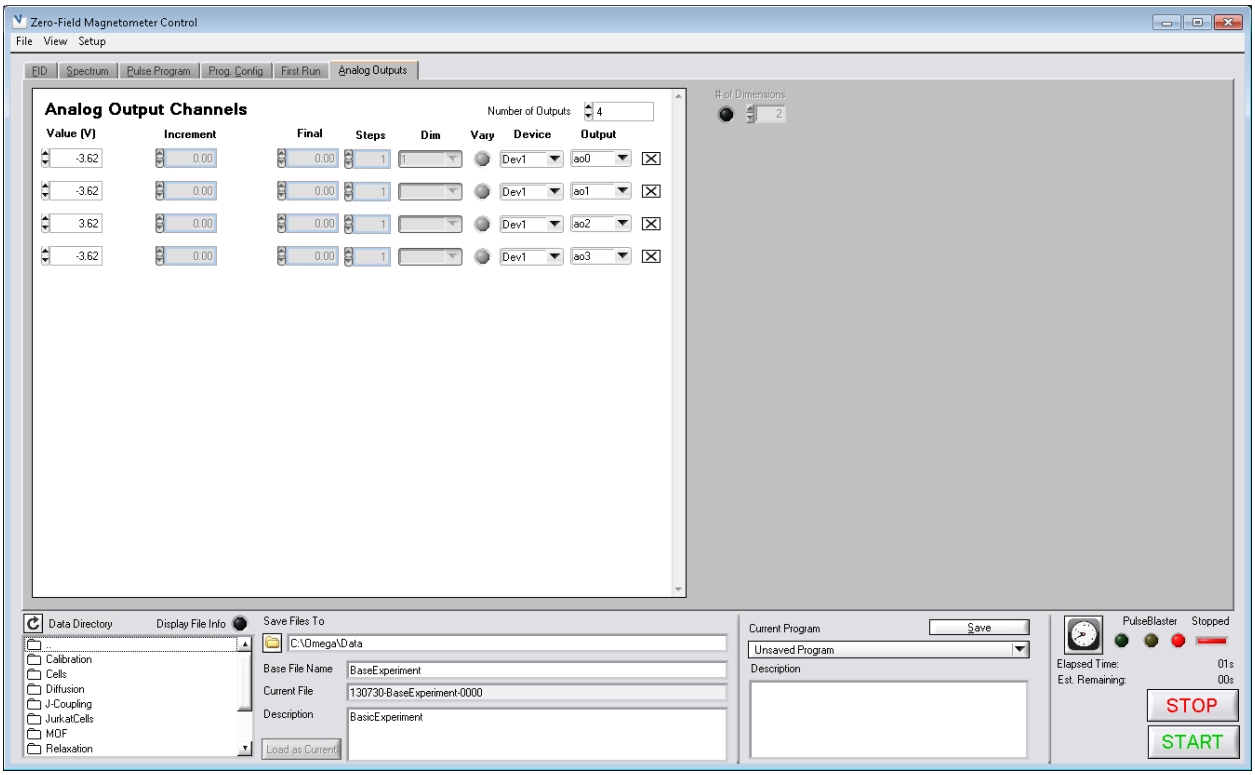


Figure 4.11: The *Analog Output* tab, used to set analog output levels and variation.

The National Instruments USB-6229M DAQ has 4 analog outputs, which are currently used in our console to control the DC pulse levels for the pulser output channels. It is also possible to use these to apply shaped pulses or low-frequency waveforms, but this functionality is not currently supported in the console control program. The analog output channels on any number of devices can be set using the *Analog Output* tab, shown in Figure 4.11. Each analog output can take values from -10[V] to 10 V. To add an analog output, increment the *Number of Outputs* control and then select the Device and Output Channel from the respective pulldowns. Selecting an output from a given device removes it from the other pulldowns so that there will be no conflicts when attempting to set the output levels.

These analog outputs can also be varied along an indirect dimension which might be used to set gradient strengths or sweep a bias offset field, for example. When multi-dimensional acquisitions have been activated using the LED and numeric control on either the *Prog Config.* or *Analog Outputs* tabs, the *Vary* LED control will be activated on each analog

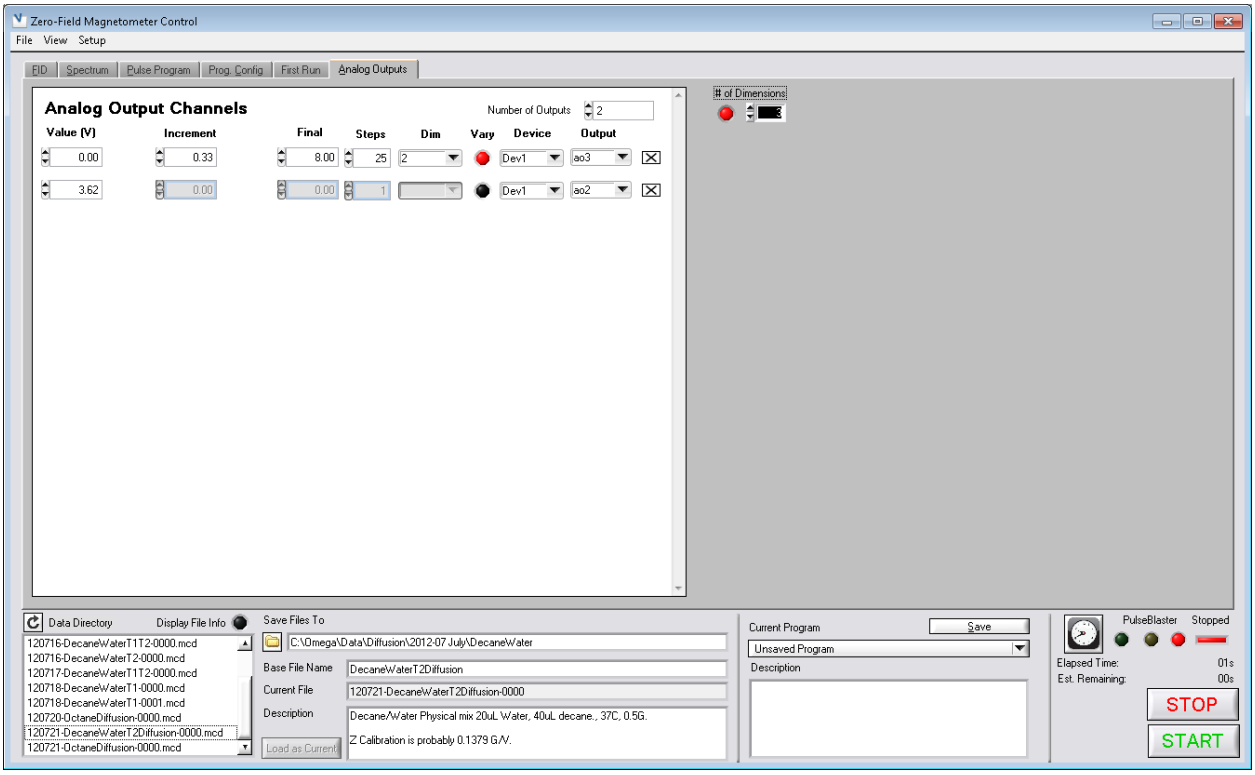


Figure 4.12: A multi-dimensional experiment which varies an analog output channel as a function of one of the indirect dimensions.

output channel. At the moment, only linear variation of analog outputs is supported. The dimensions and dimension number specified in the *Prog. Config* tab are equivalent to those specified in the *Analog Output* tab, and as such it is possible to vary both an instruction and an analog output along the same or different indirect dimensions; this means that changing the number of steps for a shared dimension on either tab will update the number of steps and increments on the other, and so care should be taken that this does not cause any issues when switching between tabs.

4.4.1.7 First and Last Runs

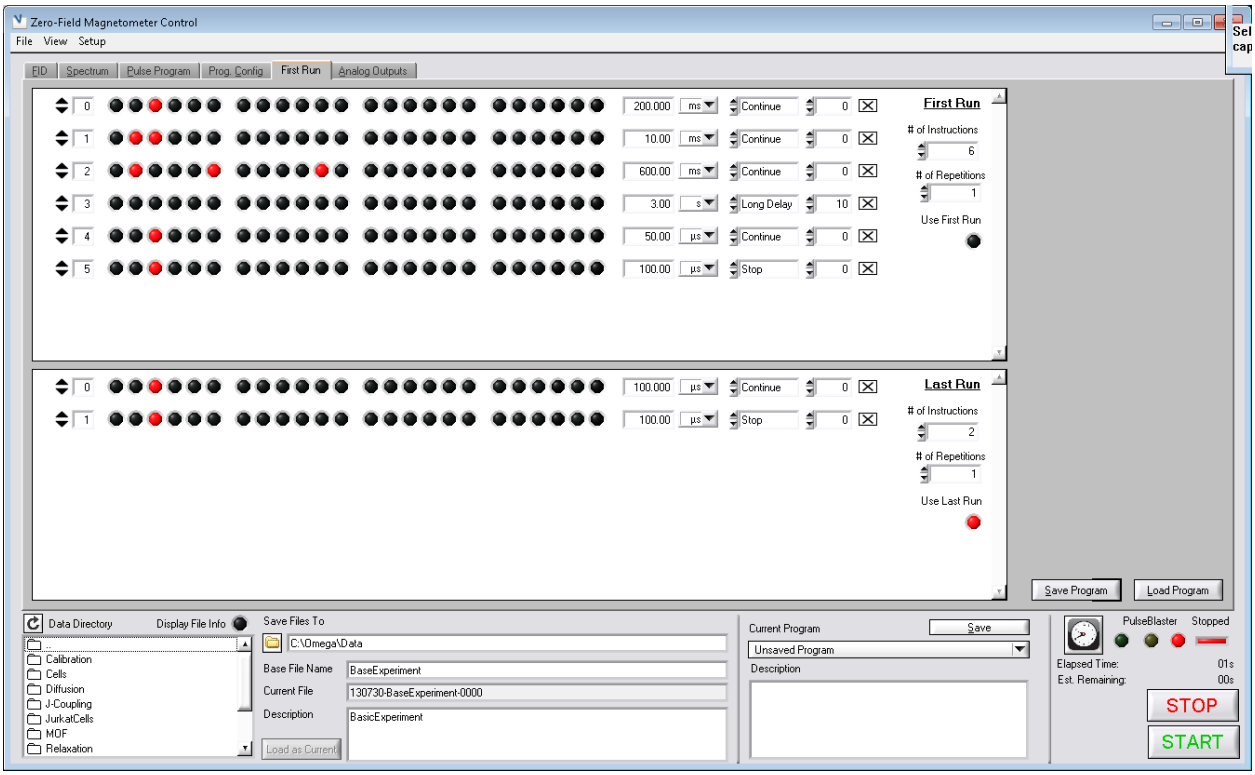


Figure 4.13: The *First Run* tab, used to set the first and last runs in an acquisition

Occasionally during the course of an experiment, the sample or system will undergo an equilibration process, usually due to differences in temperature between the prepolarization region and the detection region. This can cause problems with the first few data points before the system reaches a steady state. To avoid these equilibration problems, it is sometimes best to run a few "dummy" experiments without taking any data. This functionality is implemented in the *First Run* tab, shown in Figure 4.13, which contains a first and last run instruction set.

The top panel is used to build the instruction set for the dummy runs, which can be repeated an arbitrary number of times using the *# of Repetitions* control. The instruction

set will be saved regardless of whether or not it is used, and so its use can be toggled using the *Use First Run* LED control.

This tab also contains a last run panel - this was implemented primarily because of a bug in the PulseBlaster which tends to trigger whatever experiment is loaded into memory occasionally when the chassis voltage changes suddenly (which tended to happen due to static buildup on a person's fingers when the optical table was touched). To prevent a long sequence with many pulses from activating on its own, the Last Run was generally set to a simple pulse program which kept the shuttle from sampling. This was loaded into memory and run 1 time after the pulse sequence. It is not possible to acquire data during either the first or last run.

4.4.2 Data Display

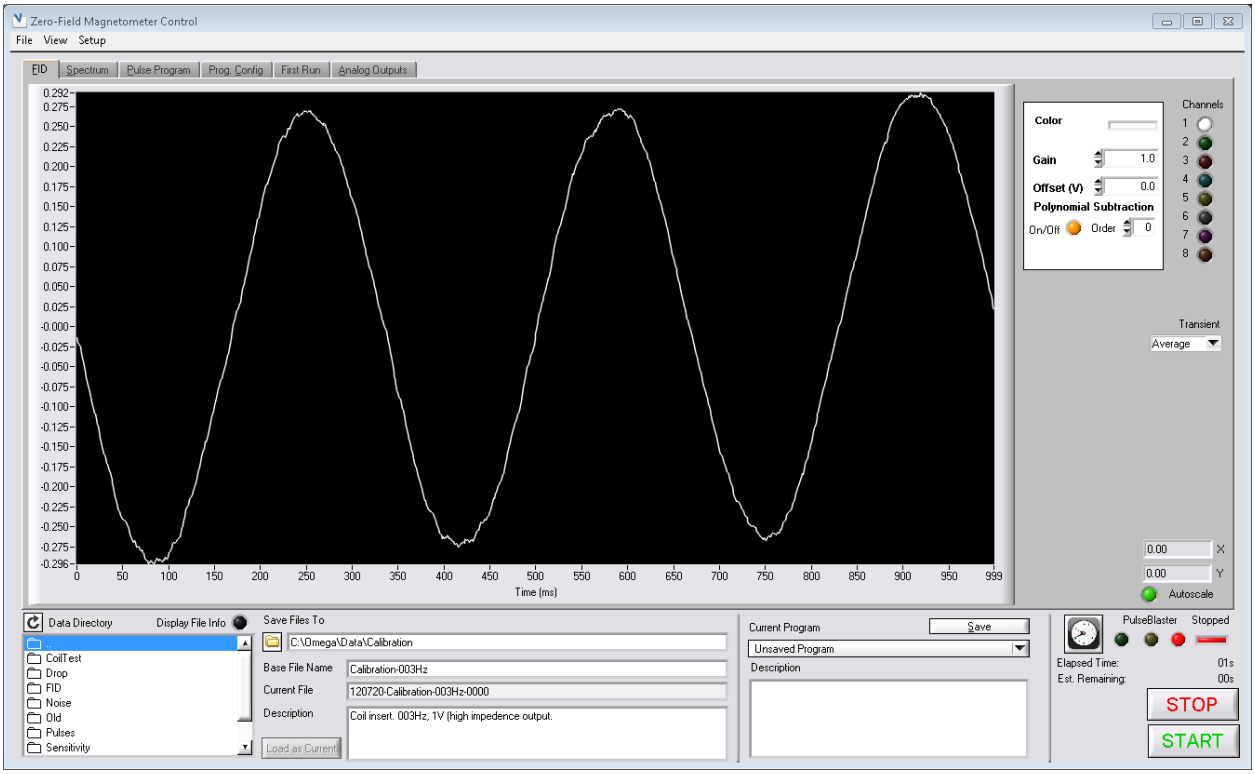


Figure 4.14: The *FID* tab, which displays data in the time domain.

While most data is likely to be analyzed in MATLAB or an equivalent platform, it can also be read and manipulated to some extent directly in the console program. Data is displayed automatically upon acquisition using the most recent settings, or can be loaded from the File>Load menu or using the navigation box in the bottom left hand corner of the program.

The data can be displayed only in one dimension, either in the time (FID) domain (Figure 4.14) or in the frequency domain (Figure 4.15). Up to 8 channels can be independently displayed, each with its own user-configurable line color, gain, offset and polynomial subtraction settings. Setting the order of the polynomial subtraction to 0 subtracts off the data mean. In the FID domain, polynomial subtraction is applied, followed by gain, then offsets. In the frequency domain, polynomial subtraction is applied to the time domain data before Fourier transformation, then gain, then offsets. When polynomial subtraction is turned on in a given channel, it is turned on in both the time domain and the frequency domain, while gain and offsets are set independently. Data are assumed to be in units of volts.

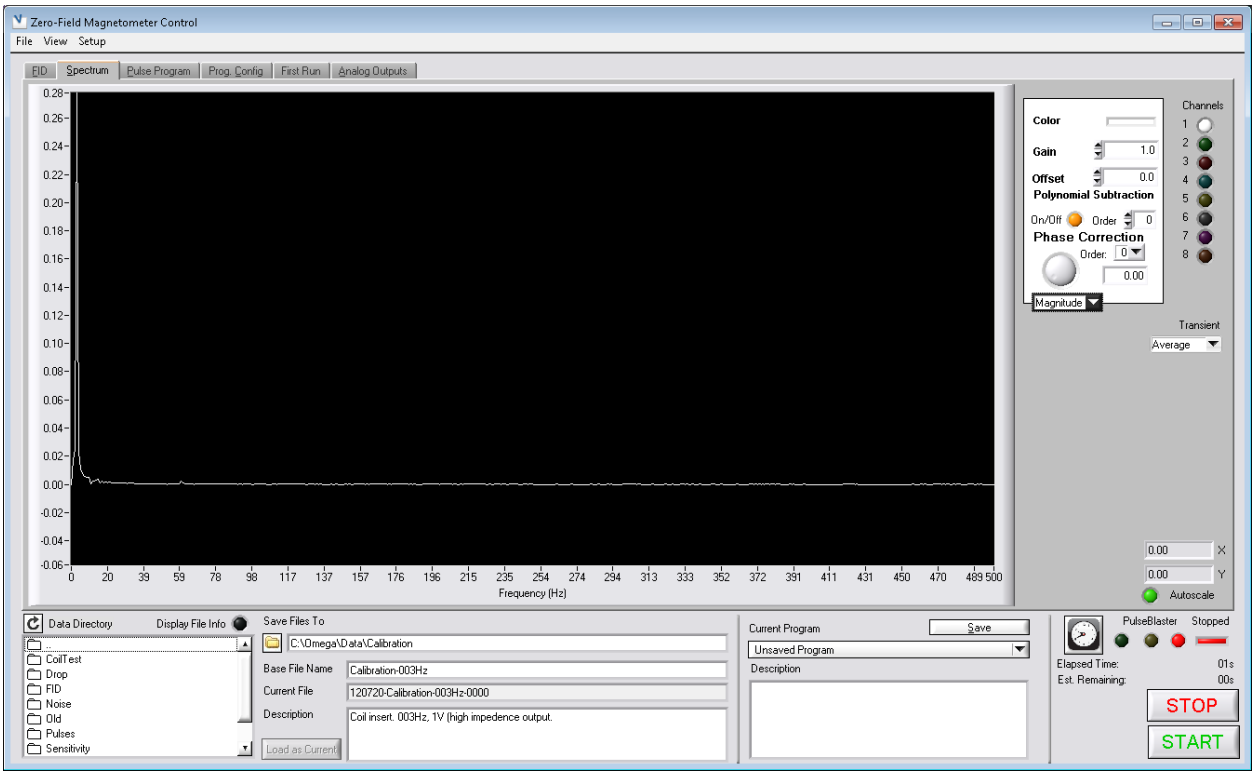


Figure 4.15: The *FFT* tab, which displays data in the frequency domain.

Data in the frequency domain can be displayed either as the real, imaginary or magnitude signal. Since quadrature detection is not used, the negative frequency domain information is not relevant and is discarded. The Fourier transform is figured to be power-conserving and its height is independent of the number of points acquired. Data are zero-padded to the nearest power of 2, but currently cannot be apodized. Up to 2^{nd} order phase corrections can be applied using the *Phase Correction* controls on the *FFT* tab. A knob indicator with a numeric control is used to set each order, and the display is updated in real-time as changes are made. Finer adjustments can be made by clicking on the knob indicator and holding down the left mouse button while dragging the mouse far away to a larger radius. For corrections ϕ_0, ϕ_1 and ϕ_2, the phase correction is applied as an element-wise multiplication

by Eqn. 4.2.

$$e^{\phi_0+\phi_1\cdot f+\phi_2\cdot f^2} \tag{4.2}$$

Data can be zoomed to a box by holding the `Ctrl` button and dragging a relevant box. There are a number of other view controls which can be accessed on the context menu brought up by right-clicking on the data display box. The *Autoscale* LED control in the bottom right hand corner of both the *FID* and *FFT* tabs determines whether the view is scaled automatically when new data are displayed or if the current view is used. The X and Y numeric indicators will display the coordinates of a point selected by left-clicking on the data display window.

As all data are saved both as individual transients and an average file, the data can be navigated using the *Transient* pulldown control on either the *FID* or *FFT* tabs. The default display upon acquiring a new transient can be set to display either the updated average, the newly acquired transient or set to make no changes in the View > Transient View menu. All three modes are common use cases for monitoring of different activities.

4.5 Data Storage and Processing

The data need to be stored for easy retrieval later and with as much metadata as it is convenient to attach - this allows the user to more easily determine experimental conditions which they may not have, at the time, thought it necessary to record. At the very least, the name of the experiment, time it was recorded and the pulse program used to record it are all stored in the data file. In some later versions of the program, some additional metadata such as a description and some calibration data are also stored automatically. During the development of the magnetometer, three different methods for saving the data were used, the last being the binary serialization format which is the current standard. MATLAB-based scripts were also developed for easy export of all three formats, so that data could be manipulated directly and if necessary exported in various common standard formats.

4.5.1 Data Formats

4.5.1.1 ASCII

The first approach we took in storing data was fairly inefficient, both from a computational and data storage point of view, though it was sufficient for our needs. Data from each acquisition were stored in a separate ASCII-formatted text file, containing a header and a set of newline-delimited points, formatted as long floats (6 digits after the decimal point). For multi-channel acquisitions, each channel was stored as a separate file. Multi-dimensional acquisitions are stored in a set of nested folders named for their position in the sampling space (e.g. in a 3-dimensional experiment, the point at [20, 40] would be stored in the folder

D01-0020/D02-0040). For each folder, the average value of all transients is pre-calculated and stored in a separate file for each channel.

As with all formats described herein, the header of each file contains the pulse program. In this case, the set of instructions used to perform each particular experiment is stored in the individual transient files, while the multi-dimensional variation (as well as the position in sampling space that has been completed) is stored is a file in the experiment's root directory.

Listing 4.2: A sample ASCII file with most of the data cut out for brevity.

```
Transient Data - Experiment: 111011-MYCalibration0002
Channel: 0 of 1
Completed: Tue Oct 11 19:17:56 2011

RawTime: 3527374676

Transient 1 of 1

#ValidPulseProgram#
NInstructions= 10
NTransients= 1
DelayTime= 0.000000
TriggerTTL= 0

NPoints= 8000
SamplingRate= 2000.000000

PhaseCycle= 0
NumCycles= 1
CycleInstr= 2
CycleFreq= 60.000000

Dimensions= 1

[Instructions]
Instruction 0 0 4 7 2 5.000000 1000000000.000000
Instruction 1 0 6 0 0 10.000000 1000000.000000
Instruction 2 0 1058 0 0 650.000000 1000000.000000
Instruction 3 0 1024 0 0 20.000000 1000000.000000
Instruction 4 0 0 0 0 20.000000 1000000.000000
Instruction 5 1 1 0 0 10.000000 1000.000000
Instruction 6 0 512 2 100 10.000000 1000.000000
Instruction 7 0 0 3 6 50.000000 1000000.000000
Instruction 8 0 4 0 0 100.000000 1000000.000000
Instruction 9 0 4 1 0 100.000000 1000000.000000
[EndInstructions]
```

```
[TransientData]
-0.336406
-0.358480
-0.302645
-0.297127
...
-0.339652
-0.340301
-0.342249
[EndTransientData]
```

Despite its inefficiency, ASCII files do have the significant advantage of being fairly easy for humans to read and interpret. The core elements of a sample ASCII file are shown in Figure 4.2, which is the data from a single transient. The time-domain data are delimited by the lines `[TransientData]` and `[EndTransientData]`. Header metadata is stored in the first few lines, and the pulse program directly precedes the data and always starts with the line `#ValidPulseProgram#`.

In the pulse program the parameter names are straightforward, with the exception of the instruction table, which has unlabeled columns. Instructions are always delimited by `[Instructions]` and `[EndInstructions]` tabs, and always start with the word "Instruction". Following that, the columns are described in more detail in Table 4.4. In forms of the program which used ASCII data storage, phase cycling was only supported relative to a trigger frequency, with a specified instruction (`CycleInstr`) incremented through a number of phases (`NumCycles`) of the carrier wave at the specified frequency (`CycleFreq`); the `PhaseCycle` parameter was a boolean value determining whether or not the phase cycling parameter should be on.

Col.	Type	Description
1	`int32`	Instruction number: The position in the instruction sequence - this should be linearly increasing
2	`bool`	Scan: Whether or not to initiate a scan on this instruction. Stored as an integer value 0 or 1.
3	`int32`	Flags: The first 24 bits encode the TTL on/off flag values of the 24 TTLs. The remaining 8 bits are unused.
4	`int32`	Instruction: The numerical value for the instruction (e.g. *Continue*, *Loop*, etc) to be used.
5	`int32`	Instruction data: The *data* parameter that is passed with some instructions.
6	`double`	Delay time: Duration of the instruction in the units specified by column 7. Units are specified separately so that the durations can be displayed in the units specified by the user in different instances of the program.
7	`double`	Units: Instructions are passed to the board in nanoseconds. This column should always be of the form 10^n with 1.0 corresponding to nanoseconds, 1000.0 corresponding to microseconds, etc.

Table 4.4: Columns of a pulse program instruction in ASCII.

4.5.1.2 TDM

As part of a general reworking of the program to eliminate some inefficiencies, the data and program storage system was updated, with a particular eye towards reducing the number of files, which can be quite unwieldy in large, multi-dimensional acquisitions. However, storing all data as one large file, possibly unstructured file can be problematic in reading out specific data subsets, and so I sought an indexed file solution. The first pass at this was the use of the National Instruments technical data management (TDM) system, which stores data in two files - a .tdm file and a .tdx file.

These files were used only very briefly as there were some issues with readout compatibility with MATLAB, and so it is not within the scope of this document to give a detailed examination of their file system. The specification allows for named fields to be created in a file with specified data types; these fields can be arrays or scalar values. A MATLAB script for reading out old TDM files is provided in the appendix and the field values used can be found there if necessary.

4.5.2 Binary Serialization Format

In order to most efficiently store and retrieve data, a binary serialization format was developed in which data are stored as named variables of one of several pre-specified types, and each variable is preceded by a header containing information about its type, name and

size. Assuming the files do not become corrupted, this information is sufficient to navigate through the data without reading out the variable values, making for quicker indexing in large data sets. The primary motivations in the development of this specification was finding an efficient mechanism for the storage of large data sets in a single file, rather than a set of smaller files.

Two different file types were used: PP (.pp) or Pulse Program files and MCD (.mcd) or Magnetometer Controller Data files. Both organized their data into top-level "container" variables, intended to separate out the domains of the varied data each contains.

4.5.2.1 Specification

The data are structured as binary files with no indexing header. Endianness is not specified in the files and is assumed to be little-endian as the programs which generate these files are written exclusively for use in Windows (x86/x64) architecture. As such, any future cross-platform file generators should be careful to generate little-endian data, or add a flag specifying the endianness[9].

The files consist of a linear sequence of entries (variables), each containing a header containing metadata with the following structure (in order):

Size (Bytes)	Type	Name	Description
4	`uint32`	`ns`	The length of the name in the entry (including null-termination)
1 · `ns`	`char*`	`name`	The name of the entry
1	`uint8`	`type`	The data type (see 4.5 for possible values)
4	`uint32`	`size`	The size of the entry's value (not including header size). The number of values in the array is `size/sizeof(type)`

The data types available are detailed in Table 4.5:

The `FS_CUSTOM` data type is used for storing an array with a custom data type, such as a `struct`. It is stored as an array of `uint8` and has a special header containing the names, sizes and types of each element. It is similar to `FS_CONTAINER` in that it is used for arrays of named elements, but it is only used for arrays with constant-sized elements. The header contains an additional `uint32` value named `nfields`, which gives the number of fields present. The names of the fields are stored in couplets arranged as `[uint32[1](ns), char[ns](name), uint8[1](type)]`. Note that to remain consistent with the other data types, this special header is considered part of the data for purposes of the `size` parameter.

[9] In all currently used file types, a version variable has been stored, allowing for the future addition of flags such as one specifying endianness, allowing for a smooth transition to more extensible file type specifications.

Value	Name	Size (Bytes)	Type	Description
0	FS_NULL	-	-	Invalid/unspecified type
1	FS_CHAR	1	char	A single signed byte (usually interpreted as a character)
2	FS_UCHAR	1	uint8	A single unsigned byte
3	FS_INT	4	int32	A 32-bit signed integer
4	FS_UINT	4	uint32	A 32-bit unsigned integer
5	FS_FLOAT	4	float	A single-precision floating-point integer
6	FS_DOUBLE	8	double	A double-precision floating-point integer
7	FS_INT64	8	int64	A 64-bit signed integer
8	FS_UINT64	8	uint64	A 64-bit unsigned integer
32	FS_CONTAINER	1	(char)	A container for binary serialization format entries. The contents are stored as a byte array and should be processed as new entries in the file. This arrangement can be nested indefinitely.
64	FS_CUSTOM	1	(uint8)	See the section on FS_CUSTOM

Table 4.5: Binary serialization format data types

By way of example, to store an array of 100 of the following structure:

```
struct example {
    uint32 an_int;
    uint32 another_int;
    uint32 some_int;

    uint8 bin;
    double time;
}
```

The header of a (parsed) `FS_CUSTOM` would read:

```
ns = 8;
name = "example\0";
```

```
type = 64;
size = 2163;
data = [5, R
        7, "an_int\0", FS_INT,
       12, "another_int\0", FS_INT,
        9, "some_int\0", FS_INT,
        4, "bin\0", FS_UCHAR,
        5, "time\0", FS_DOUBLE,
        {21 x 100 array}]
```

Note that the size of each element is `uint32·3 + uint`*cdot*`8 + double` $= 12 + 1 + 8 = 21$, so the size of the data array is 2100 bytes. 4 bytes are used for the number of fields (5), 5 bytes for the field types, 20 bytes for the 5 field name lengths (`uint32` $\cdot$ `5`), and then in this case the field names are 7, 12, 9, 4 and 5 characters long (including null-termination), for a total of 37. So the total size of the data is $2100 + 37 + 20 + 1 + 5 = 2163$, which is reflected in the "size" parameter.

4.5.2.2 Pulse Program (.pp) Files

Pulse Program (.pp) files utilize the binary serialization format to contain information about NMR experiments and pulse programs. The data are all stored in a top-level `FS_CONTAINER` named `PulseProgram`. This is done because it allows pulse programs to be read from MCD files with no modification to the readout routine, as the readout routine always starts by searching for the container named `PulseProgram`. This also serves as a check on files which may have become corrupted.

In the LabWindows/CVI program, pulse programs are stored in their entirety in a `PPROGRAM` structure, which serves as a pseudo-class for conveniently passing pulse program information between various functions.[10] The structure definition can be found in Fig. 4.3. As individual instructions are used in various different contexts, a separate compound type (`PINSTR`, defined in Fig. 4.4) is defined to contain a single instruction, and arrays of these structures form the backbone of the pulse programs.

In the .pp files, the contents of these structs are broken up into seven sub-containers: `[Properties]`, `[Instructions]`, `[ND/PC]`, `[AnalogOutput]`, `[Skip]`, `[FRInstructions]` and `[LRInstructions]`. `[Properties]` and `[Instructions]` are included in all programs, while the others are included only as-needed. They are broken up this way for easy categorization and to allow for modular pulse programming files which omit entire sections rather than having sections filled with erroneous or unnecessary information.

Listing 4.3: The `PPROGRAM` structure used to store pulse programs.

```
typedef struct PPROGRAM // This is a structure for containing information about pulse
    programs.
{
    int np;             // Points to sample in direct dimension
    double sr;          // Sampling rate in direct dimension
    int nt;             // Number of transients
    int trigger_ttl;    // Which TTL flag corresponds to the trigger?
```

[10] As LabWindows/CVI uses an ANSI-C backend, true object-oriented programming is not supported.

```
                    // [For multi-device compatibility]

int tmode;          // Three Options (Data Acquisition)
                    // MC_TMODE_ID [0]: ID first, then advance transients.
                    // MC_TMODE_TF [1]: All transients first, then ID
                    // MC_TMODE_PC [2]: Phase Cycles first, then IDs


int scan;           // Boolean indicating whether a scan is taken
int use_pb;         // Boolean indicating whether the PulseBlaster is used
int varied;         // Boolean indicating if the program uses indirect
                    // dimensions or phase cycles.

int n_inst;         // Number of instructions

double total_time;  // Total calculated duration of the pulse program

PINSTR **instrs;    // Pointer to an array of instructions
int nUniqueInstrs;  // Number of unique instructions (size of *instrs)


PINSTR *frins;      // Array of first-run instructions
int frnInstrs;      // Number of first-run instructions
int frnReps;        // Number of first-run repetitions
int fr;             // Whether or not to use the first run.

PINSTR *lrins;      // List of the last run instructions
int lrnInstrs;      // Number of last-run instructions
int lrnReps;        // Number of last-run repetitions.
int lr;             // Whether or not to use the last run.

// Variation (indirect dimensions and phase cycling)
int nDims;          // Number of dimensions [1+#indirect dimensions]
int nCycles;        // Number of phase cycles
int nVaried;        // Number of instructions which vary
int real_n_steps;   // Actual number of steps after skips.
int max_n_steps;    // Maximum number of steps, which is:
                    // maxsteps[0]*maxsteps[1]*...*maxsteps[end]

int skip;           // Boolean indicating whether the skip condition is used
char *skip_expr;    // The expression used to generate the skips

char **delay_exprs; // Array containing the delay expressions.
char **data_exprs;  // Array containing the data expressions.
                    // Both of these string arrays are "" if unused.

// The following two indices are little-endian with size nDims+1
// They take the form :
// [{position in transient space} {position in indirect sampling space}]
int steps_size;     // Size of steps
int *maxsteps;      // Maximum position in the sampling space
int *steps;         // Linear indexing dimension sizes based on tmode.
                    // MC_TMODE_ID: [{dim1,...,dimn},nt]
                    // MC_TMODE_TF: [nt, {dim1,...,dimn}]
                    // MC_TMODE_PC: [{pc1,...,pcn},{dim1,..., dimn},nr_pc]
                    // Where nr_pc = nt/prod(pc[:])

int *v_ins;         // Index of which instructions are varied.
int *v_ins_dim;     // Dimension/cycle along which each varies -
                    // Encoded bitwise, cycles first
int *v_ins_mode;    // The mode in which they are varied.
                    // PP_V_ID [1]:  indirect
                    // PP_V_PC [2]:  phase cycling
```

```
                        // PP_V_BOTH[3]: both.

    int **v_ins_locs;   // A multidimensional array indicating, for each varied
                        // instruction, which instruction in the instruction
                        // array its spot in instr_locs should be pointing at.
                        // The locations are stored as the values in a linearly
                        // indexed two-dimensional array.
                        // The array will be of size [nVaried][total_num_steps]
                        // For c_step = [a b c] of maxsteps = [d e f], for
                        // the n-th varied instruction, use instruction
                        // v_ins_locs[n][a+b*d+c*(d*e)] for v_ins[n].
                        //
                        // This array has total size nVaried*max_n_steps

    int *skip_locs;     // A linear-indexed array of size max_n_steps saying
                        // whether to skip that point in sampling space

    // Analog outputs
    int nAout;          // Number of analog output channels
    int n_ao_var;       // Number of varied channels
    int *ao_varied;     // Bool array of size nAout indicating channel variation
    int *ao_dim;        // Dimension each channel varies (size nAout)
    double **ao_vals;   // Analog output values
                        // Size = [nAout][(ao_varied)?dim_steps[ao_dim[i]]:1]
    char **ao_chans;    // Full name of each channel used (size nAout)
    char **ao_exprs;    // The expression generating the thing
                        // Size nAout, null if ao_varied[i] != 2;


    int valid;          // Is this PPROGRAM valid - only used internally.
} PPROGRAM;
```

Listing 4.4: The `PINSTR` structure used to store individual pulse instructions.

```
typedef struct // This is a structure for containing individual instructions.
{
    int flags;          // TTL flags, encoded bitwise in a 32 bit integer.
                        // For the n-th TTL, the on/off state is given by:
                        // TTL[n] = (flags & 2^(n) > 0)
    int instr;          // The instruction index (WAIT, LOOP, STOP, etc)
    int instr_data;     // The instruction data parameter
    int trigger_scan;   // Whether this instruction triggers a scan

    double instr_time;  // The duration of the instruction
    int time_units;     // Instruction units (by SI prefix) - multiply instr_time
                        // by (1000)^(time_units) to get the value in nanoseconds.
                        // Thus, ns = 0, us = 1, ms = 2, etc.
} PINSTR;
```

4.5.2.2.1 [Properties]

The `[Properties]` section contains the basic properties of the pulse program, such as the number of instructions, number of transients, number of dimensions, number of phase cycles, etc. This is mostly metadata *about* the program itself, and is primarily useful in operations like readout and saving - where these properties determine things such as what sections are included, what controls are active, the size of arrays to allocate and the type of experiment that is being performed. The items and format in the properties header can be found in Table

4.6, in the order the are found. Many boolean values are stored as FS_UCHAR (unsigned single byte) to be conservative with space - these are stored as 32-bit int values in the PPROGRAM struct for ease of conversion and readout.

Name	Type	Description
Version	FS_DOUBLE	The version of the .pp file.
np	FS_INT	Number of points in the direct dimension.
sr	FS_DOUBLE	The sampling rate in Hz
nt	FS_INT	The number of transients
trigger_ttl	FS_UCHAR	The TTL index used for triggering; this allows programs to easily be transferred between different setups by rearranging the TTLs to adjust to different triggering configurations.
tmode	FS_UCHAR	Data acquisition mode 0 (MC_TMODE_ID): Indirect dimensions first 1 (MC_TMODE_TF): Transients first 2 (MC_TMODE_PC): Phase cycles first, then indirect dimensions, repeat as necessary.
scan	FS_UCHAR	If the program records a scan
use_pb	FS_UCHAR	If the program uses the PulseBlaster
varied	FS_UCHAR	If the program varies along either an indirect dimension or with a phase cycled instruction
n_inst	FS_INT	The number of instructions in the main pulse program instruction list.
nUniqueInstrs	FS_INT	As many instructions are repeated in multi-dimensional experiments, the instructions array is stored as the set of unique instructions plus an instruction-indexing array (v_ins_loc). This is the number of unique instructions in the array.
frnInstrs	FS_INT	The number of instructions in the first-run program.
frnReps	FS_INT	The number of times to repeat the first-run instruction set.

fron	FS_INT	Whether or not the first run is used.
lrnInstrs	FS_INT	The number of instructions in the last-run program.
lrnReps	FS_INT	The number of times to repeat the last-run instruction set.
lron	FS_INT	Whether or not the last run is used.
total_time	FS_DOUBLE	The total time that the pulse program should take - calculated in nanoseconds (older versions of the program do not generate reliable values for this entry).
nDims	FS_INT	Number of dimensions, including the direct dimension
nCycles	FS_INT	Number of phase cycles present
nVaried	FS_INT	Number of instructions which vary, either along an indirect dimension or by phase cycling.
max_n_steps	FS_INT	Maximum number of indirect steps in the program, including all indirect dimensions and phase cycles based on the dimensionality of the experiment.
real_n_steps	FS_INT	Number of indirect steps that are actually performed after the skip condition is taken into account.
skip	FS_UCHAR	Boolean indicating whether the skip condition is used
nAout	FS_INT	The number of analog outputs used in the experiment.
n_ao_var	FS_INT	The number of analog output channels which vary along an indirect dimension.

Table 4.6: The items in the "Properties" header for .pp files with Version 0.1

4.5.2.2.2 [Instructions]

The [Instructions] container consists of a single array of instructions, using the custom-array data type FS_CUSTOM - which has a header defining a compound data type and the number of elements of this type in the array. In this case, each instruction consists of 6 elements, as shown in Table 4.7.

Name	Type	Description
flags	FS_INT	The instruction flags, encoded bitwise in the first 24 bits of a 32-bit integer.
instr	FS_INT	The instruction type index (CONTINUE, WAIT, STOP, etc)
instr_data	FS_INT	The instruction data parameter
trigger_scan	FS_UCHAR	Whether or not this instruction should trigger a scan
instr_time	FS_DOUBLE	The instruction delay in the units specified by time_units.
time_units	FS_UCHAR	The units of the delay in SI prefixes. To get the instruction time in nanoseconds, multiply instr_time by $10^{3 \cdot \texttt{time\_units}}$.

Table 4.7: The items in the instruction array compound type.

Variation either along indirect dimensions or phase cycles is encoded in the same way, where the indices of the varied instructions are stored in an array (more details on this in Section 4.5.2.2.4), and as the experiment is stepped through indirect dimensions, these instructions are replaced with a version of the instruction modified appropriately. To allow for arbitrary instruction variation, these modified instructions are generated once and stored in an array of length nUniqueInstrs - this is the array stored in the [Instructions] container.

4.5.2.2.3 [AnalogOutput]

The [AnalogOutput] header stores information about the use of analog output channels and devices in the pulse program. The general parameters regarding the use of analog outputs (number of channels, whether or not outputs are used etc) are stored in [Properties], while this header specifically contains the configuration for each of the individual channels; as such, each parameter is an array with at least one dimension having length nAout.

Name	Type	Description
ao_varied	FS_UCHAR	Boolean array of length nAout indicating whether each of the active analog outputs varies along an indirect dimension.
ao_dim	FS_UCHAR	Array of length nAout indicating what dimension each output varies along - the value of dimension n should not be used if ao_varied does not evaluate to True. Dimension is a 1-based index starting with the first *indirect* dimension.
ao_vals	FS_DOUBLE	The analog output level for each of the channels. This is an array of arrays - the main dimension has length nAout, for non-varied channels the second dimension has length 1, otherwise the length is the number of points along the appropriate dimension.
ao_chans	FS_CHAR	A string array of length nAout containing the name of the physical channel (and device) corresponding to each analog output - this helps to match equivalent channels when device configurations change. These generally have the form Dev[#]/ao[#], e.g. Dev1/ao2.
ao_exprs	FS_CHAR	A string array of length nAout containing expressions used to generate the analog output values. As of this writing, non-linear analog output variation is not supported, and so this parameter is a placeholder for future functionality.

Table 4.8: The items in the [AnalogOutput] header for .pp files with Version 0.1

4.5.2.2.4 [ND/PC]

This section concerns the variation components of the pulse program and is only present if at least one instruction is varied either along an indirect dimension or a phase cycle. These are a set of various arrays used to determine the parameters of the program and how indexing proceeds. Phase cycling and multidimensional experiments both represent modifications to a base program which replace one or more of the instructions from the original set; as such,

the two features are implemented using a common core set of functions.

The "variation space" has dimensionality $nd + nc$ where nd is the number of dimensions and nc is the number of phase cycles. The size of the sampling grid (which can be subsampled using the Skip functionality described in Section 4.4.1.5) is defined as [sd_1, sd_2, ... sd_{nd}, sc_1, ... , sc_{nc}] where sd_n is the number of points in the n^{th} indirect dimension, and sc_n is the number of points in the n^{th} phase cycle. The size of the *entire* grid as stored is [sd_1, ... , sd_{nd}, sc_1, ..., sc_{nc}, nr] where nr is the number of times the full phase set of phase cycles is repeated, given by:

$$nr = \frac{nt}{\prod_{i=0}^{nc} sd_i},$$

where nt is the total number of transients in the experiment. While phase cycles are generally indexed within transient acquisitions primarily because the relevant signal is most often the average over all positions in phase space, it is conceptually valuable to separate transients with identical pulse programs from transients with varied pulse programs as the distinction can occasionally be useful. The *total space* sampling grid is simply the *variation space* sampling grid, tiled along the "repetitions" axis nr times, and so it is often easier to simply store or access the grid using the *variation space* subset and a simple index. This conceptual separation also enables to choice of sampling-order described in Section 4.4.1.1, wherein all unique instructions are sampled before repetitions.

As described in the 4.5.2.2.2 section, instruction variation is stored as the subset of unique instructions and an indexing matrix (`v_ins_locs`) which stores the index of the unique instructions array to use for a given varied instruction at a given point in the sampling space. When actually performing the experiment, a method for linearly indexing the `steps` (and, consequently, the `maxsteps`) array is chosen based on the parameter `tmode` in the `[Properties]` header; however, although the indexing matrix is stored as an array of linearly-indexed arrays, the indexing method used is *not* dependent upon the transient acquisition mode. Although storing the `v_ins_locs` array using its own linear indexing scheme necessitates the conversion from a linear index (acquisition step) to a position in the multi-dimensional variation space and then back to a linear index (position in the indexing matrix), there is very little overhead associated with these conversions, and the alternative would mean that the entire `v_ins_locs` array could effectively be corrupted just by incorrectly setting the `tmode` flag. As such, the inner array of `v_ins_locs` is always linearly indexed based on the scheme s

Name	Type	Description
`maxsteps`	`FS_INT`	The size of the variation space subset of the sampling grid. 1D array of length `nDims`+`nCycles`
`steps`	`FS_INT`	The size of the total sampling grid. 1D array of length `nDims`+`nCycles`+1

v_ins	FS_INT	The instructions (as indexed in the base pulse program) which vary during the experiment. 1D Array of length nVaried
v_ins_dim	FS_INT	The dimension or cycle along which each of the varied instructions varies. 1D array of size nVaried
v_ins_mode	FS_INT	A flag indicating how the varied instruction should be treated 0 (PP_V_ID): Varies along an indirect dimension 1 (PP_V_PC): Varies along a phase cycle 2 (PP_V_BOTH): Varies along both an indirect dimension and a phase cycle 4 (PP_V_ID_EXPR): Uses an expression to vary the indirect dimension. 8 (PP_V_ID_ARB): Uses an arbitrary table to vary the indirect dimension. (not currently supported)
v_ins_dim	FS_INT	The dimension or cycle along which each of the varied instructions varies. 1D array of size nVaried
v_ins_locs	FS_INT	The indexing matrix which maps a position in sampling space onto a set of pulse instructions. Size of the array is [nVaried][$\left(\prod_{i=0}^{\texttt{nDims}} sd_i\right)\left(\prod_{i=0}^{\texttt{nDims}} sd_i\right) \cdot \left(\prod_{i=0}^{\texttt{nCycles}} sc_i\right)$]. The inner array is always indexed with dimensions first, followed by cycles.
delay_exprs	FS_CHAR	Newline-delimited string array containing expressions used for the delays. Size is nVaried - unused expressions will be empty strings.
data_exprs	FS_CHAR	Newline-delimited string array containing expressions used for varying the data parameters. Size is nVaried - unused expressions will be empty strings.

Table 4.9: The items in the [ND/PC] header for .pp files with Version 0.1

4.5.2.2.5 [Skip]

The [Skip] header is used only if a skip condition is found (regardless of whether or not the condition is used[11]). It consists of only two parameters, skip_expr and skip_locs. The skip_expr is stored as a single FS_CHAR array (string), and contains the expression used to generate the skip condition. The skip_locs array is a linearly-indexed array containing one boolean value (saved as FS_UCHAR) for each acquisition in the experiment, indicating whether the point should be skipped.

4.5.2.2.6 [FRInstructions] and [LRInstructions]

The [FRInstructions] and [LRInstructions] containers are instruction arrays constructed using the same method described in Section 4.5.2.2.2 - though since last and first run instruction sets cannot vary parameters, a straightforward instruction array is used. This functionality was unfortunately added without an accompanying increase in the version number, and so while all pulse programs saved in the current version contain both of these instruction sets (whether or not they are used they are saved for convenience), their presence in all version 0.1 .pp files cannot be assumed.

4.5.2.3 Magnetometer Controller Data (.mcd) Files

Magnetometer Controller Data (.mcd) files contain all the information that is stored about a single NMR experiment, divided into four top-level containers: DataHeader, which contains information about the the data set and how it was collected, DisplayHeader, which contains information about how the data was displayed - including information about stored calibration values, PulseProgram, which is the pulse program used in the experiment and DataGroup, which contains the collected data.

4.5.2.3.1 DataHeader: Experiment Metadata

The DataHeader is a relatively small container which is used to save metadata about the data file and the conditions of the experiment such as time of day, name of the experiment and a short description. The details for each parameter are given in Table 4.10. It is worth noting that the parameters HashCode, TimeDone and TimeDoneEpoch have been improperly implemented in the most recent version of the software, and may not necessarily store meaningful information.

Name	Type	Description

[11] This information is stored even when unused because the .pp files are also used to allow the UI state to persist between instances of the program, and so it is used whenever the UI state differs from the default value.

filename	FS_CHAR	The filename that the data were originally saved under
ExperimentName	FS_CHAR	The base experiment name which is used to generate the filename
ExperimentNum	FS_UINT	The repetition number of the experiment - a number added to the filename to prevent duplicate filenames.
Description	FS_CHAR	A description of the experiment entered in the software by the user
HashCode	FS_CHAR	A currently unused parameter. It was originally intended for use in identifying files when restarting experiments.
NumChans	FS_CHAR	The number of channels in the acquisition.
DateStamp	FS_CHAR	The date of the experiment. The string is formatted with `strftime()` with the format `MCD_DATE_FORMAT`, which is currently defined as `%y%m%d`.
TimeStarted	FS_CHAR	The time that the experiment was started, as a human-readable timestamp. The format is given by `MCD_TIME_FORMAT`, which is currently `%H:%m:%S, %a, %b %d, %Y`
TimeDone	FS_CHAR	The time that the experiment finished, in the same format as `TimeStarted`.
TimeStartedEpoch	FS_UINT	The time that the experiment was started, in seconds since the start of the Unix Epoch. This information is redundant with `TimeStarted`, but is included to make the time stamps more machine-readable.
TimeDoneEpoch	FS_UINT	The time that the experiment finished, in seconds since the start of the Unix Epoch.
CurrentIndex	FS_UINT	The (zero-based) index that the experiment reached before this file was written. For experiments that fully finished, this should be equal to `MaxSteps-1`.

MaxSteps	FS_INT	The total number of steps in the experiments, including indirect dimensions and transient averaging.

Table 4.10: The entries in the [DataHeader] group of a .mcd file, containing the experiment metadata.

4.5.2.3.2 DisplayHeader

As the magnetometer controller program is designed to both operate the device and display the data, a certain amount of data processing is necessary to display meaningful information to the users, e.g. calibration and spectrum phasing. While it would be imprudent to permanently modify data before it is saved, it is also preferable to have some record of these parameters, so that later users can recreate the data as they were displayed while it was acquired. This can be useful when multiple types of experiments tend to be displayed in significantly different ways - for example, relaxometry and diffusometry experiments tend to have significant artifacts in the FFT domain, and the spectrum display is turned off, while J-coupling experiments are *primarily* displayed in the FFT domain; as such, it is preferable to load the display parameters when the experiment is loaded into the program, to save the user the trouble of changing parameters when changing experiment type.

Folded into this header are also some parameters about the actual experimental setup, such as the gain on the preamplifier and the most recent calibration values, as these are used in the display calculations and no separate "parameter" group is included in the MCD files.

Name	Type	Description
poly_on	FS_UCHAR	Whether or not polynomial baseline subtraction is turned on.
poly_ord	FS_UINT	The order of the polynomial to fit to the baseline.
phase	FS_FLOAT	The phase correction vector for each of the nc active channels. This is a linear-indexed matrix of size 3*nc, grouped by channel.
fft_channel	FS_INT	The currently selected FFT channel (for purposes of adjusting the channel settings such as color, gain, offset, etc.)
spec_gains	FS_FLOAT	A vector of size nc containing the display gain for the spectrum of each channel.

spec_offsets	FS_FLOAT	A vector of size nc containing the display offset for the spectrum of each channel.
fid_gains	FS_FLOAT	A vector of size nc containing the display gain for the FID of each channel.
fid_offsets	FS_FLOAT	A vector of size nc containing the display offsets for the FID of each channel.
spec_chans_on	FS_UCHAR	A vector of boolean values indicating whether to display the spectrum for each of the nc channels.
fid_chans_on	FS_FLOAT	A vector of boolean values indicating whether to display the FID for each of the nc channels.
mag_cal	FS_DOUBLE	The calibration factor which converts between volts and the magnetization unit.
mag_cal_units	FS_CHAR	The unit associated with the magnetization. This is a human-readable string, such as pT or nG.
x_units	FS_CHAR	The units for the x (time) axis, as displayed. This is a human-readable string such as ms| or s.
y_units	FS_CHAR	The units for the y (amplitude) axis, as displayed. This is a human-readable string such as mV or V.
amp_gain	FS_DOUBLE	The gain on the amplifier input. This can be used along with res_val to calculate the current across the photodiode.
res_val	FS_DOUBLE	The value of the shunt resistor on the acquisition photodiode circuit.

Table 4.11: The entries in the [DisplayHeader] group of a .mcd file, containing the experiment display parameters.

4.5.2.3.3 DataGroup

The data themselves are stored in a separate group, which stores the data from each step in the experiment as a series of numbered vectors. Rather than pre-allocating the entire data file, the main data group is simply the last group in the file, and new data is appended to

the end of the file. Since the headers are of fixed-size for a given experiment, the information in the headers can be updated without restructuring the entire file, allowing the timestamps, current index, etc to be updated with each new transient.

The numbering scheme of the data vectors is used to represent their position in the full experimental space (including transients, indirect dimensions and phase cycles). In the most recent version of the program, the points in this space are not delimited in any way (though this is something that should likely be remedied in future versions), and the values are instead encoded as fixed-size blocks. The block size is always chosen to be the smallest block size which accommodates the dimension with the greatest number of steps, so for a 25×150 experiment with 4 transients, each of the 3 blocks would be of size 3, with format `"[000000000]"`; for a 5×26 experiment with 1 transient, each block would only have size 2 and would be formated as `"[000000]"`.

4.5.3 MATLAB Readout

To make the data more accessible for data processing, a small library was developed for reading out the binary serialization format into generic MATLAB struct, independent of its contents. This allows for exploration of unspecified serialized formats - which is useful for debugging and future-proofing against undocumented changes possibly made on-the-fly. The full source of these scripts can be found in Section D.1.

4.5.3.1 General File Readout: `mc_read_bin`

The `mc_read_bin` script is a format-agnostic binary serialization format readout script. It converts from the binary serialization format into a MATLAB struct. Each item in a container or file is converted into a field on the output struct with the relevant data type (numeric, string or, in the case of containers, struct). Because all entries in the binary files have internal names, the fields in the struct are given the same names as their corresponding file entries, but with invalid special characters stripped out; the only exception to this naming convention is when a field name contains only numbers, as numbers are not valid MATLAB field names. In in the case of numeric names, `ind_` is prepended to the number, to make a valid field.

4.5.3.2 PP File Readout: `mc_read_prog`

The `mc_read_prog` script looks for a `[PulseProgram]` group in a binary file and processes the information into a struct containing a mixture of human-readable and machine-readable information. The results are used extensively in processing and interpreting the data in the `mc_read_data` script. While most of the program metadata fields are the same as those detailed in 4.5.2.2, the instructions are parsed into both an easy-to-read table in `.instrs`, and into a machine-readable structure, `.ps.instrs`. While `.instrs` displays the data as a cell with one column per parameter, the `.ps.instrs` structure breaks the parameters into vectors of size `ps.instrs.ni`.

4.5.3.3 MCD File Readout: `mc_read_data`

The `mc_read_bin` script reads magnetometer controller data from a binary file and performs various relevant types of data processing on the data. Because many of the experiments performed with the magnetometer are "magnetization measurement" type experiments, which apply a train of π pulses and observe magnetization flipping, the script reads the pulse program and searches for a loop containing an acquisition, and, upon finding one, extracts the data between pulses (this is necessary because acquisition is "always on" in these experiments, rather than being switched on and off when the required data become available). If the script fails to find such a loop, data is processed as an FID to be Fourier transformed, and apodization and zero-packing are applied. In all cases, the original data are stored in the `.odata` array, while the processed data are stored in the `.mdata` array.

Chapter 5

Low Field NMR

5.1 Overview

Nuclear magnetic resonance is a tool with wide applicability across many disciplines in both industrial and research contexts. Traditional NMR experiments are performed using superconducting coils to generate magnetic fields which are both strong and homogeneous, which are generally used to perform either some chemical analysis, for tomographic imaging (generally of optically opaque media)[27–29], or both.[30]

The magnetic field plays two roles in an NMR experiment, it polarizes the spins (see Section 5.2 for more details), and it induces precession in spins aligned transverse to the field, at frequency $\omega_0 = \gamma \mathbf{B_0}$, where γ is the gyromagnetic ratio of the spin and B_0 is the bias field strength.[31,32] Most NMR experiments are detected using inductive coils, which detect the first time derivative of the magnetic flux through the coil, and as such the sensitivity of the detector is an increasing function of the frequency of the signal. As such, strong fields both increase the signal generated by the sample and increase the sensitivity of the detector.[1] As a consequence of the dependence of spin frequency on magnetic field strength, signal coherence - and as a consequence resolution - depends upon the homogeneity of the field.[33,34]

Much work has been done on applications of NMR in situations wherein one or more of these constraints either must or can be relaxed. Generally speaking, the discipline of low-field NMR is concerned with NMR performed without employing superconducting magnets; this includes small-molecule NMR spectroscopy[35] performed using specially-constructed permanent magnets as well as *ex-situ* NMR measurements such as those used in oil-well logging,[36,37] wherein measurements of relaxation and diffusion are performed in the gradient field generated by a single-sided NMR instrument[38–40] (see Chapter 6 for further discussion of these techniques).

The use of various hyperpolarization techniques like spin-exchange optical pumping of noble gases,[41–43] para-hydrogen induced polarization[44,45] and dynamic nuclear polarization[46,47] allow the creation of highly polarized samples, generally far in excess of what can be achieved using the bias field for sample polarization; meanwhile the development of low-

frequency magnetic field detectors such as Superconducting QUantum Interference Devices (SQUIDs)[48,49] or vapor cell magnetometers has dramatically increased detector sensitivity in low magnetic fields. The latter improvement has led to the development of ultra-low-field NMR and MRI, wherein detection takes place in Earth's magnetic field[50] or at even lower fields in magnetically shielded environments.[2,3,18,19,29]

5.2 Sample Prepolarization

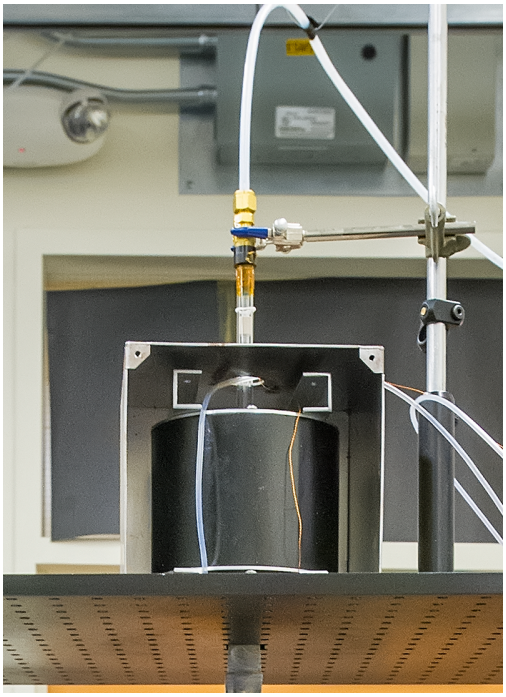


Figure 5.1: A photograph of the polarization region. The sample is shuttled up and down using the central vacuum and air lines, respectively.

In NMR experiments, the quantity measured is the evolution of the net magnetization - which is a bulk property of an ensemble of spins. The net magnetization is directly related to the spin polarization, as each spin has a small magnetic dipole, which adds for coherent spins polarized in the same direction but subtracts for spins polarized in the opposite direction. The equilibrium spin polarization is given by Eqn. 5.1:

$$\frac{P_+ - P_-}{P_+ + P_-} = \frac{1 - e^{-\hbar\gamma B_0/k_B T}}{1 + e^{-\hbar\gamma B_0/k_B T}} \tag{5.1}$$

$$= \frac{e^{\hbar\gamma B_0/2k_B T} - e^{-\hbar\gamma B_0/2k_B T}}{e^{\hbar\gamma B_0/2k_B T} + e^{-\hbar\gamma B_0/2k_B T}} \tag{5.2}$$

$$= \tanh\left(\frac{\hbar\gamma B_0}{2k_B T}\right). \tag{5.3}$$

For $\gamma B_0 \ll k_B T$, it is appropriate to use the first-order Taylor expansion of the *tanh* function, giving an approximate bulk polarization of:

$$\frac{P_+ - P_-}{P_+ + P_-} \approx \frac{\hbar\gamma B_0}{2k_B T}. \tag{5.4}$$

At high field, this produces small but noticeable polarization, so for protons at room temperature (taken here as 298 K), where $\gamma_H = 267.513\,^{\text{rad}\cdot\,\text{MHz}}/_{\text{T}}$, spin polarization is $\approx 3.43\,^{\text{ppm}}/_{\text{T}}$. At zero field, however, the spin polarization is also 0, and even if a several Gauss prepolarizing pulse were used (roughly the maximum value without magnetizing the shields), the polarization level would be $0.343\,^{\text{ppb}}/_{\text{G}}$, a loss in 4 orders of magnitude of an already small signal.

The signals generated herein are generated using a prepolarizing permanent magnet of either 1 or 2 T, which sits outside the outermost layer of magnetic shielding. Spins rest in the field for at least a few T_1 periods before being shuttled to the lower-field detection region. Under otherwise adiabatic sample transfer (see Sec. 3.2.1 for more details), spin polarization dies off roughly as the T_1 of the sample during the transfer. For $T_1 \leq t_{shuttle}$, the loss in

signal between the two regions is negligible. This does, however, add a T_1 filter into the output experiment, rendering short-T_1 components of the sample relaxation impossible (or at least difficult) to measure by this method.

This sample prepolarization method is useful because it allows each characteristic experimental field to be individually optimized: the field in which spins are detected needs to be extremely homogeneous, but not necessarily extremely strong when using a magnetometer, the field in which spins are polarized needs to be extremely strong, but not necessarily extremely homogeneous. This allows us to use a not-necessarily homogeneous field for polarization and a low field for detection - whereas when these are the same field, it must be both large and homogeneous (such as fields generated by certain superconducting magnet designs).

While prepolarizing with an inhomogeneous field can lead to inhomogeneous polarization, once polarized, the spins will still evolve coherently in a homogeneous magnetic field. For bulk samples the signal will be proportional to the integral of the field over the entire sample - and thus a small relative drop in the field in parts of the gradient will lead to a small relative drop in overall signal but no distortion. For imaging, in the absence significant spin diffusion during the transfer period, inhomogeneous prepolarization can lead to some small signal strength distortions - which can likely be corrected for by mapping the polarization gradient using a known standard.

5.3 J-coupling Spectroscopy

One of the most compelling applications of magnetometry to date is J-coupling spectroscopy, which is a technique for measuring the magnetic resonance spectrum of spins as they evolve in response to indirect spin-spin dipolar couplings (called *J-couplings*).[50–52] These couplings are mediated by hyperfine interactions between the nuclear spins and their local electronic environment and thus depend on many factors of the molecular configuration - bond angles and distances, connectivity and electronic structure. However, since they are a function only of spin-spin interactions, they are largely independent of the local magnetic field and are present even at zero field. This insight has formed the basis of a significant branch of our research agenda exploring the use of atomic magnetometers as detectors for J-spectroscopy at zero field.[53–56]

5.3.1 Theory

The standard liquid-state NMR Hamiltonian for a multi-spins system is given in Eqn 5.5:[57]

$$H = \sum_i \sum_{j \neq i} J_{ij} \mathbf{I}_i \cdot \mathbf{I}_j + \sum_i \gamma_i \mathbf{B} \cdot \mathbf{I}_i \tag{5.5}$$

When the local magnetic field $\mathbf{B} = 0$, only the J-coupling Hamiltonian remains:

$$H_J = \sum_i \sum_j \mathbf{I}_j \cdot \mathbf{I}_j. \tag{5.6}$$

This interaction is thus available even in the presence of zero magnetic field. Still, the form of the Hamiltonian presents potential problems for NMR signal detection, as the scalar operator $\mathbf{I}_i \cdot \mathbf{I}_j = I_{ix}I_{jx} + I_{iy}I_{jy} + I_{iz}I_{jz}$ is completely spatially isotropic and invariant to any spatial transformation such as rotations or mirror inversions.

$$\rho_0 = \frac{\hbar\gamma B_0}{2k_B T}\left(\sum_i \mathbf{I}_i\right) \tag{5.7}$$

And the time-evolution of the density matrix is given by the Louisville Equation (Eqn. 5.8):

$$\frac{\partial \rho}{\partial t} = -\frac{i}{\hbar}\left[H, \rho\right]. \tag{5.8}$$

Unfortunately, $[\rho_0, H_J] = 0$ (see Sec. B.2.1.1 for a detailed proof), and so thermal polarization is a stationary state of the Hamiltonian. This is true at high field as well, but at high field the B_0 field provides an a magnetically anisotropic environment, and changing the orientations of the spins relative to that environment are sufficient to see precession. The J-coupling Hamiltonian, on the other hand, is spatially isotropic, and depends only on the *relative* orientations of the interacting spins - and as such, changing their direction has no effect on the evolution of the state.

For two coupled spins I_1 and I_2 rotated relative to one another by angle θ in the $x - z$ plane defined as the plane normal to $I_1 \times I_2$:

$$\rho_0 = \frac{\hbar\gamma B_0}{2k_B T}\left[\frac{1+\cos(\theta)}{2}(I_{1z} + I_{2z}) + \frac{1-\cos(\theta)}{2}(I_{1z} - I_{2z}) + \sin(\theta)I_{2x}\right] \tag{5.9}$$

5.3.2 Application

The detectable evolution under the J-coupling comes from the $I_z - S_z$ terms in the density matrix. These terms show up naturally in the initial density matrix of an ensemble of heteronuclear spins. For the two spin case, the initial polarization is given by:

$$\rho_0 = \frac{\hbar\gamma_I B_0}{2k_B T} I_z + \frac{\hbar\gamma_S B_0}{2k_B T} S_z \tag{5.10}$$

Which can be rearranged into the relevant parallel and antiparallel components of the density matrix:

$$\rho_0 = \frac{\hbar B_0\left(\gamma_I + \gamma_S\right)}{4k_B T}\left(I_z + S_z\right) + \frac{\hbar B_0\left(\gamma_I - \gamma_S\right)}{4k_B T}\left(I_z - S_z\right) \tag{5.11}$$

As has already been shown from theory section (Sec. 5.3.1), only terms corresponding to $I_z - S_z$ will evolve under the J-coupling Hamiltonian. As can be seen from Eqn 5.11, heteronuclei will have a natural polarization in both the parallel and antiparallel configurations, corresponding to Eqn 5.12:

$$P_{\uparrow\downarrow} = \frac{\gamma_I - \gamma_S}{2\gamma_I} \qquad P_{\uparrow\uparrow} = \frac{\gamma_I + \gamma_S}{2\gamma_I} \tag{5.12}$$

For the case of carbon-hydrogen pairs, $\gamma_H \approx 4\gamma_C$, so $P_{\uparrow\downarrow} = {}^3\!/\!_8$, while $P_{\uparrow\uparrow} = {}^5\!/\!_8$, and so the signal can be increased if the spins can be pulsed out of phase of one another. This has the additional advantage of allowing the "start point" of the experiment to be determined by the pulse, ensuring a consistent phase between transients. In the case of carbon and hydrogen, the simple 4:1 ratio allows $I_z + S_z$ terms to be easily transformed into $I_z - S_z$ terms (and vice-versa) by applying a $4\pi_x$ pulse to the hydrogens, which corresponds to a π pulse to the carbons. This is particularly convenient because it starts the spins aligned along the z axis, which is the sensitive direction of the magnetometer.

In the more general case, for a pulse about the y axis of angle $\theta_I = \theta + \frac{\Delta}{2}$ on spins I_z and angle $\theta_S = \theta + \frac{\Delta}{2}$ [1] on the S_z spins applied to spins initially aligned along $I_z + S_z$, the density matrix evolves in Eqn 5.13[2]:

$$\rho(\theta) = \cos\left({}^{\Delta}\!/\!_2\right)\cos(\theta)\left(I_z + S_z\right) + \cos\left({}^{\Delta}\!/\!_2\right)\sin\left(\theta\right)\left(I_x + S_x\right) \tag{5.13}$$

$$+ \sin\left({}^{\Delta}\!/\!_2\right)\sin\left(\theta\right)\left(I_z - S_z\right) - \sin\left({}^{\Delta}\!/\!_2\right)\left(I_x - S_x\right) \tag{5.14}$$

Both $I_z - S_z$ and $I_x - S_x$ represent evolving, singlet polarization, along different spatial axes under the zero-quantum coherences evolving under the J-coupling Hamiltonian, and they are maximized when $\Delta = n\pi$ for $n \in \mathbb{Z}$, and so the optimal pulse length on the **I** spins is:

$$\begin{aligned} \theta_{I,op} &= \theta_{S,op} - n\pi \\ &= \theta_{I,op}\left(\frac{\gamma_S}{\gamma_I}\right) - n\pi \\ &= \frac{n\pi}{1 - \frac{\gamma_S}{\gamma_I}} \end{aligned} \tag{5.15}$$

When using scalar magnetometers, only the $\vec{z}$ component of the magnetic field is measured, and so only the $I_z - S_z$ terms are detectable (see Sec. 5.3.4), and so the signal is maximized when n is chosen such that ${}^{n\pi}\!/\!_{1\,-\,\frac{\gamma_S}{\gamma_I}}$ is as close to an integer value as possible. Alternatively, a composite pulse sequence can be used to guarantee both that $\Delta = n\pi$ and

[1] From this definition $\Delta = \frac{\theta_I \gamma_S}{2\gamma_I}$.

[2] A full proof for this assertion is found in Sec. B.2.1.3

that all the spins are aligned along $\vec{z}$, so long as pulses can be applied along arbitrary directions. The sequence consists of applying a pulse of angle ${}^{\theta_{S,OP}}/_{2}$ along $\vec{x}$, followed by a pulse of $\theta_I = \pi$ along the direction ${}^{\theta_{S,OP}}/_{2}$ of the $\vec{z}$ axis (i.e. $[0, -\sin({}^{\theta_{S,OP}}/_{2}), \cos({}^{\theta_{S,OP}}/_{2})]$), and finally a pulse along $\vec{x}$ of angle $-{}^{\theta_{S,OP}}/_{2}$.

The primary downside of this pulse sequence is the difficulty of applying pulses along an arbitrary direction. At high field, this is not as much of a problem, because in the rotating frame, phase shifting an RF signal corresponds to changing the direction of the pulse. At zero field, however, spins are not precessing about a large bias field, and so the pulse direction corresponds to a physical direction - in practice, this means that a linear combination of DC pulse coils must be used to apply pulses along an arbitrary direction. This can be complicated by differences in the response of each coil and potential digitization of potential output values. The pulses may need to be re-calibrated as a function of direction, and so it is likely less useful to use such a pulse sequence except in circumstances where pulse error due to gyromagnetic ratio mismatch are very large. For the most common case - hydrogen and carbon, ${}^{\gamma_{1H}}/_{\gamma_{13C}} = 3.9760836$. If a simple 4π pulse is applied to the hydrogen spins, the resulting pulse error is 6 mrad. Each pulse of $\Delta = \pi$ corresponds to $\theta_I = {}^{1}/_{1 - \frac{\gamma_S}{\gamma_I}}$, which is 1.336π rad, representing a pulse error of 336 mrad away from ideal. However, on pulses of $\Delta = 3n\pi$, the pulse error is $25.04n$ mrad. For arbitrary γ_S and γ_I, the optimal pulse time can be found iteratively:

$$n_i = n_{i-1}\text{round}\left(\left|\text{round}\left(\frac{n_{i-1}}{1-\frac{\gamma_S}{\gamma_I}}\right) - \frac{n_{i-1}}{1-\frac{\gamma_S}{\gamma_I}}\right|^{-1}\right) \tag{5.16}$$

Starting with $n_0 = 1$ and $i > 0$. As this is an iterative product of increasing size (as the residuals get smaller n_i gets larger), the optimal choice of i is likely going to be fairly low, as errors in the pulse calibration are likely to swamp any gains from increasing optimality of pulses. For the ${}^{13}C{-}^{1}H$, $n_{i\in[0,4]} = (1, 3, 372, 107136, 98672256, 189944092800)$, corresponding to pulse errors of (1056, 25.2, 10.9, 3.41, 1.63, 0.575) mrad. For a ${}^{15}N$-${}^{1}H$ system, the first 3 terms are $n_{i\in[0,2]} = (1, 9, 579)$ corresponding to residuals of (354.5, 48.9, 13.8) mrad. For these systems, using $n_i = 1$, as it represents at most a 2.5% and 4.9% drop in overall signal (to some extent this is mitigated by the fact that anti-parallel terms occur in the thermal distribution of heteronuclei, and so some signal is present even with no pulse). For a ${}^{13}C$ $-{}^{15}N$ system, however, the first 4 terms are: $n_{i\in[0,3]} = (1, 3, 21, 630)$, which correspond to residuals of (903, 434, 106, 48.6) mrad. As such, even the n_2 term represents a 10% drop in signal. In a case such as this, it may be preferable to use a composite pulse sequence to prepare the initial state.

5.3.3 Zero-Quantum Subspace

Evolution under the J-coupling Hamiltonian takes place in the zero-quantum subspace - which consists of terms corresponding to terms with no change in total spin angular momentum. This can be easily demonstrated in the two-spin case, where the zero-quantum

subspace terms correspond to I_+S_- and I_-S_+, which represent coherences between states of the form $|\uparrow\downarrow\rangle$ and $|\downarrow\uparrow\rangle$; in this special case, the 2×2 zero-quantum subspace can be treated as a single pseudo-spin, with axes ZQ_x, ZQ_y and ZQ_z, which can be expressed as linear combinations of the appropriate single-spin operators.

5.3.4 Detectable Coherences

While precession under the J-coupling Hamiltonian is necessary to observe J-spectroscopy, it is not alone sufficient, as coherences must also produce a net magnetic dipole. The general quantum mechanical observable associated with the magnetic dipoles generated by an ensemble of polarized spins is given by Eqn. 5.17:

$$\mu_z = \hbar\gamma_I \sum_i I_{iz} \tag{5.17}$$

Taking as an example the two-spin case, the spins are initialized along ZQ_z ($ZQ_z = I_z - S_z$), and the Hamiltonian lies along the ZQ_x axis, inducing the spins to precess in the $ZQ_y - ZQ_z$ plane:

$$\begin{aligned}\rho(t) &= e^{-iJ_{IS}(I_xS_x+I_yS_y)t}(I_z - S_z)e^{iJ_{IS}(I_xS_x+I_yS_y)t} \\ &= \cos(J_{IS}t)(I_z - S_z) - \sin(J_{IS}t)(I_xS_y - I_yS_x)\end{aligned} \tag{5.18}$$

The magnetometer signal will then be given by the expectation value of the dipole operator, $\langle\mu_z\rangle$. The expectation value of an operator is equivalent to the trace of that operator operated on the density matrix (Eqn. 5.19):

$$\langle A\rangle(t) = \mathrm{Tr}\left[\rho(t)A\right] \tag{5.19}$$

This can be written as:

$$\begin{aligned}\langle\mu_z\rangle =&\hbar \cdot \mathrm{Tr}\left[cos(J_{IS}t)(\gamma_I I_z + \gamma_S S_z)(I_z - S_z)\right. \\ &\left. - \sin(J_{IS}t)(\gamma_I I_z + \gamma_S S_z)(I_xS_y - I_yS_x)\right]\end{aligned} \tag{5.20}$$

And because traces are linear functionals (i.e. $Tr(aA + bB) = aTr(A) + bTr(B)$), this can be split up into its constituent parts:

$$\langle\mu_z\rangle(t) = \hbar\cos(J_{IS}t)\,\mathrm{Tr}\left[\mu_z(I_z - S_z)\right] - \hbar\sin(J_{IS}t)\,\mathrm{Tr}\left[\mu_z(I_xS_y - I_yS_x)\right] \tag{5.21}$$

The magnetization from the ZQ_z term is then:

$$\langle\mu_{z,ZQ_z}\rangle(t) = \cos(J_{IS}t)\mathrm{Tr}\left[(\gamma_I I_z + \gamma_S S_z)(I_z - S_z)\right] \tag{5.22}$$

And the magnetization from the ZQ_y term is:

$$\left\langle \mu_{z,ZQ_y} \right\rangle (t) = -\sin(J_{IS}t)\mathrm{Tr}\left[(\gamma_I I_z + \gamma_S S_z)(I_x S_y - I_y S_x)\right] \tag{5.23}$$

The value of the trace can be calculated from the following properties of the spin operators:

$$I_x^2 = I_y^2 = I_z^2 = \mathbb{1} \tag{5.24}$$

$$Tr(I_x) = Tr(I_y) = Tr(I_z) = 0 \tag{5.25}$$

$$I_i S_n = \sigma_i \otimes \sigma_n \tag{5.26}$$

$$Tr(A \otimes B) = Tr(A)Tr(B) \tag{5.27}$$

From Eqns 5.26 and 5.27, we can separate the traces of heteronuclear products to the products of the traces of their homonuclear terms (e.g. $\mathrm{Tr}(I_x S_x) = \mathrm{Tr}(I_x)\mathrm{Tr}(S_x)$):

$$\langle \mu_{z,ZQ_z} \rangle (t) = \hbar\cos(J_{IS}t)\left[\mathrm{Tr}(\gamma_I I_z^2) - \mathrm{Tr}(\gamma_S S_z^2) - \mathrm{Tr}(I_z)\mathrm{Tr}(S_z) + \mathrm{Tr}(I_z)\mathrm{Tr}(S_z)\right] \tag{5.28}$$

$$\begin{aligned} \left\langle \mu_{z,ZQ_y} \right\rangle (t) = \quad & -\hbar\sin(J_{IS}t)\left[\gamma_I \mathrm{Tr}(I_z I_x)\mathrm{Tr}(S_y) - \quad \gamma_I \mathrm{Tr}(I_z I_y)\mathrm{Tr}(S_x)\right. & (5.29)\\ & \left. +\gamma_S \mathrm{Tr}(I_x)\mathrm{Tr}(S_z S_y) - \quad \gamma_S \mathrm{Tr}(I_y)\mathrm{Tr}(S_z S_x)\right] & (5.30) \end{aligned}$$

And from Eqn. 5.25, all terms corresponding to the trace of a single spin operator (e.g. S_z, I_x) are 0, while squared spin operators are 1, per Eqn. 5.24. The trace of two-spin mixed operators (e.g. $I_x I_y$) need not be calculated, as in these cases they are always paired with a a lone heteronuclear term. Plugging this in to Eqn. 5.30, we find that the ZQ_y term gives no detectable magnetization:

$$\left\langle \mu_{z,ZQ_y} \right\rangle (t) = 0, \tag{5.31}$$

and so only magnetization from the ZQ_z term are detectable:

$$\langle \mu_z \rangle (t) = \cos(J_{IS}t)(\gamma_I - \gamma_S). \tag{5.32}$$

5.4 Magnetization Detection

In many instances, rather than directly observe the spectral characteristics of the samples, it is preferable to look at the level of bulk magnetization of the sample - this is particularly true in experiments performed in an indirect dimension (see Sec. 5.5). The magnetization is simply a function of the spin polarization and gyromagnetic ratio and shows up on the a simple DC level on the magnetometer. However, the presence of *any* magnetic field can cause a shift in the magnetometer signal, as can drift and noise - which is particularly high close to DC due to the presence of $^1/_f$ noise, and discrete noise sources from physical jostling of the instrument. An additional problem is that the sample magnetization decays over time as the spins relax to their non-polarized state, and as such have some low-frequency characteristics themselves beyond a simple DC offset. A proper measurement of the magnetization can be made by taking advantage of the fact that ensembles of spins react to magnetic pulses, while bias offsets and noise are invariant under such transforms. This allows one to design a pulse sequence which measures only sample magnetization and filter out other sources of magnetic field.

5.4.1 π Train

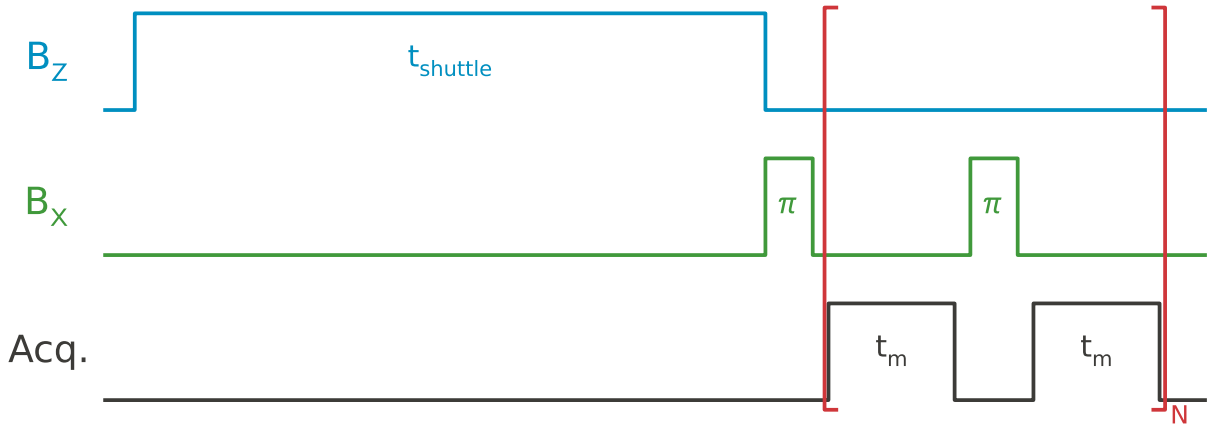


Figure 5.2: A magnetization detection sequence using a π train along a single direction. A strong bias offset field is applied during the shuttling time to ensure an adiabatic transfer of magnetization.

The simplest and most common pulse sequence for the measurement of bulk sample magnetization is the application of a train of π pulses transverse to the direction of initial polarization. This should invert the signal magnetization, and the change in DC level in response to a pulse corresponds to the signal magnetization - as other sources of magnetic field should be invariant under pulses, the difference in signal is

$$\Delta B = (B_s + B_{off}) - (-B_s + B_{off}) = 2B_s, \tag{5.33}$$

where B_s is the field generated by the sample and B_{off} is the field represented by any local fields which vary slowly with respect to the pulse time. Repeating this in a train of π pulses separated by time τ should create signal which looks like a decaying square wave (as can be seen in Fig. 5.2). In a sense, this is similar to maximally-aliased version of the resonance condition present in a normal NMR experiment, where the observed effective frequency of oscillation is reduced to $\omega_0 {}^{t_p}/_{t_p+\tau}$. One advantage of this method is that the inhomogeneous broadening induced by the pulses scales with t_p, while the relaxation will mostly scale with τ, so a typical case of $t_p = 100\,\mu$s, $\tau = 50$ ms and $T_2 = 3$ s, using the very inhomogeneous small pulsing coils (see Sec. 3.3.3), with a linewidth of 2 Hz corresponding to a T_2^* of 159 ms. The effective T_2^* from inhomogeneous broadening from the pulses is 79 s, and so the pulse homogeneity does not play much role in the relaxation.

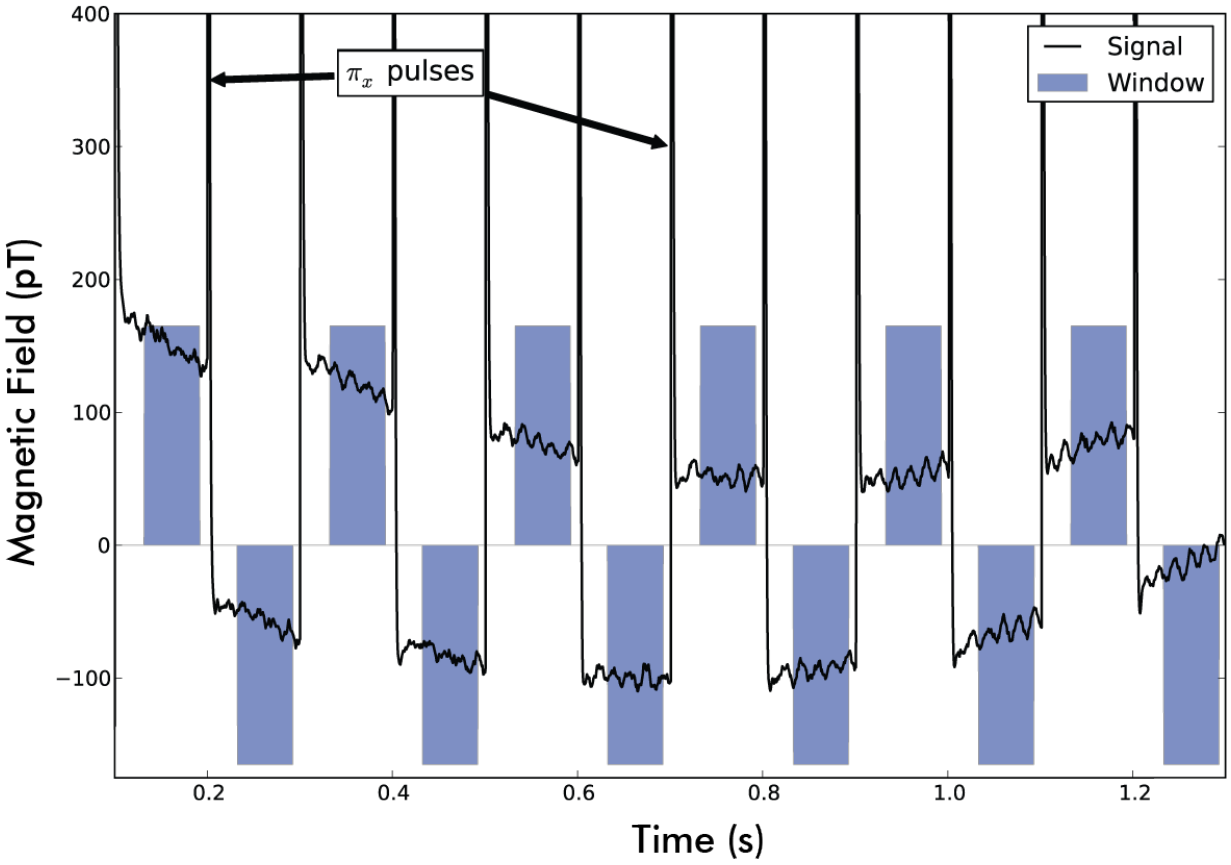


Figure 5.3: Direct dimension signal of water acquired using a π train. The sample magnetization is determined from the change in the magnetic field between positive and negative "window" periods.

5.4.1.1 Multiphase π Trains

Despite the simplicity of the 1st order effect - spin flipping - there are many variations available on the pulse sequence presented in Fig. 5.2, all of which can have different effects

on different systems. The simplest variation is the use of pulses along $\vec{y}$, or indeed along any arbitrary phase $\vec{u}$ in the transverse plane. The main effect would be identical, but consider the case of a small, static bias offset magnetic field field along an arbitrary direction $\vec{v}$ that has been inadequately shimmed out. The effect of a train of evenly-spaced π pulses along x is:

$$\begin{aligned} I_x &\rightarrow \quad I_x \\ I_y &\rightarrow -I_y \\ I_z &\rightarrow -I_z \end{aligned} \tag{5.34}$$

This is equivalent to a mirror reflection in both $\vec{y}$ and $\vec{z}$, which induces a series of spin echoes of the component of magnetization along $\vec{y}$ and $\vec{z}$, and the only remaining bias offset fields effectively felt by the spins are the fields along $\vec{x}$. This can be quite useful for removing persistent bias offset fields, as it allows each of the three axes to be addressed individually. Since the detection takes place along the longitudinal axis, applying a "π train" along $\vec{z}$ is still possible, using the sequence $\frac{\pi}{2}_x - \pi_z - \frac{\pi}{2}_x$, which has the effect:

$$\begin{array}{ccccccccc} & & \left(\frac{\pi}{2}\right)_x & & & \pi_z & & \left(\frac{\pi}{2}\right)_x & \\ I_x & \longrightarrow & & I_x & \longrightarrow & & -I_x & \longrightarrow & -I_x \\ I_y & \longrightarrow & & -I_z & \longrightarrow & & -Iz & \longrightarrow & -I_y \\ I_z & \longrightarrow & & -I_y & \longrightarrow & & I_y & \longrightarrow & I_z \end{array}$$

The downside to these pulses, however, is that it takes $\approx 2t_\pi$ rather than t_π, but this is generally not particularly limiting, as the dead time of the magnetometer response is often substantially longer than the pulses themselves. The spin-echo generating properties of π trains can also be used to greater effect by alternating the phases of the pulses at each pulse. For a $\pi_x - \tau - \pi_y - \tau$ sequence, the $\vec{y}$ and $\vec{z}$ components of the magnetization refocus at $2n\tau$. The $\vec{x}$ magnetization, however, refocus on $4n\tau$, because the first π_y pulse does not occur until 2τ after the spins begin to evolve. Additionally, the symmetry of the sequence allows the alternating π pulses to correct the errors in the pulses of their complement as well, as the pulse errors are refocused as well:

$$I_z \xrightarrow{\pi_x+\varepsilon_x} -I_z\cos(\varepsilon_x) - I_x\sin(\varepsilon_x) \xrightarrow{\pi_y} I_z\cos(\varepsilon_x) + I_x\sin(\varepsilon_x) \xrightarrow{\pi_x+\varepsilon_x} -I_z \tag{5.35}$$

As can be seen in Eqn. 5.35, error along the π_x direction is corrected on every other pulse by the spin echo property of the π_y pulse. When there are errors on both the π_x and π_y pulses, alternating the pulses partially corrects the pulses in the perpendicular direction. This follows a four-pulse series, with the residual error for $\varepsilon_x, \varepsilon_y \ll 1$ after the 4^{th} pulse given by:

$$\theta_4 \approx 1 - \frac{\varepsilon_x^4\varepsilon_y^2 + \varepsilon_x^2\varepsilon_y^4}{8} \tag{5.36}$$

And the residual error given by the $4n^{\text{th}}$ pulse is:

$$\theta_{4n} \approx \left(1 - \frac{\varepsilon_x^4 \varepsilon_y^2 - \varepsilon_y^4 \varepsilon_x^2}{8}\right)^{n^2} \tag{5.37}$$

And given that $\varepsilon_x, \varepsilon_y \ll 1$, this can be truncated at the 1st term:

$$\theta_{4n} \approx 1 - n^2 \left(\frac{\varepsilon_x^4 \varepsilon_y^2 - \varepsilon_y^4 \varepsilon_x^2}{8}\right) \tag{5.38}$$

This can be compared to the error from a simple π train, which scales linearly with ε. One thing to note, however, is that since the error correction takes place over the course of 4 pulses, the shot-to-shot error will still be on the order of ε_x and ε_y. Using a fairly consistent pulse circuit, it is likely that the pulse error will be limited by the clock speed of the pulse controller. For a 100 MHz clock, the minimum pulse resolution is 10 ns. For 0.5 G pulses on protons, that corresponds to $\varepsilon = 6.69{\cdot}10^{-5}$ rad, and so generally even a single-phase π-train does not introduce significant angular error due to pulse imperfections - though benefits can still possibly be gained by correcting for a shifting bias offset field.

5.4.1.2 Heteronuclear Magnetization

The simple and even multi-phase π train methods of magnetization detection work only when the spins all have the same gyromagnetic ratio (and thus the same π pulse duration). This is not the case in heteronuclear ensembles. While it may be possible to measure the J-spectrum, it is sometimes preferable to make a simple measurement of the magnetization, at least for testing purposes. For two nuclei, this can be accomplished easily with a simple composite pulse:

$$\tfrac{\pi}{2}_{x,I} - \pi_{y,S} - \tfrac{\pi}{2}_{x,I} \tag{5.39}$$

This works the same way as the error correcting pulses above, wherein the symmetry about the π_y pulse cancels out any evolution about the $\vec{x}$ axis in the **S** spin, and since the **I** spins are along $\vec{y}$ when the pulse is applied, they are unaffected by the π_y pulse. This is effectively the inverse of the composite J pulse described in Sec. 5.3.2, which attempts to invert the spins relative to one another, whereas this attempts to prevent them from dephasing while inverting their direction. This method is not simply generalized to higher numbers of nuclei, which should be taken on a case-by-case basis, taking into account the relative gyromagnetic ratios of the spins.

5.4.2 Quadrature Signal Detection

While a π train is sufficient for most experiments measuring only bulk sample magnetization, it measures only the scalar $\vec{z}$ magnetization of the sample and discards the information along $\vec{x}$ and $\vec{y}$ directions. This can be occasionally be useful information, particularly when

acquiring free-induction decays in an indirect dimension. Take, for example, the signal acquired after the sequence detailed in Fig. 5.4: The spins are tipped along $\vec{y}$ and allowed

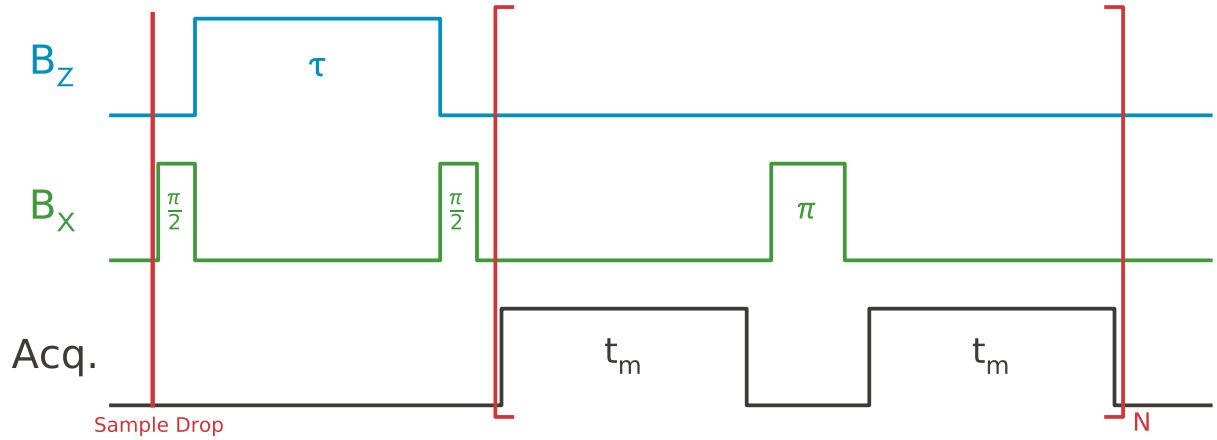


Figure 5.4: A sequence for acquiring FIDs in the indirect dimension.

to precess for time τ, resulting in the state:

$$\rho(\tau) = I_y \cos(\gamma_I B_Z \tau) - I_x \sin(\gamma_I B_z \tau) \tag{5.40}$$

After a second $\frac{\pi}{2}_x$ pulse, the spins are in the state:

$$\rho_D(\tau) = -I_z \cos(\gamma_I B_Z \tau) - I_x \sin(\gamma_I B_z \tau)$$

If the standard pulse sequence is used, the signal will be $S(\tau) = \cos(\gamma_I B_Z \tau)$, which throws away the imaginary channel entirely - roughly half the signal. There are two ways to acquire the imaginary channel - one is to repeat the measurement, but instead replacing the second pulse with a $\frac{\pi}{2}_{-y}$ pulse. However, when the detection region is at zero field, since the remaining transverse component does not evolve except under the pulses, the imaginary channel can be also be acquired in a single acquisition by switching to the pulse sequence shown in Fig 5.5:

This sequence causes the signal along $\vec{z}$ to alternate between representing the $\vec{x}$ and $\vec{y}$ components of the precession in the indirect dimension, such that the real channel ($\vec{x}$) signal is given by $l_{4n+2} - l_{4n+1}$ and the imaginary channel ($\vec{y}$) signal is given by $l_{4n+4} - l_{4n+3}$ where l_n is the average signal level in the n^{th} lobe of the square wave, and $n = [0, {}^{(N-1)}/_4]$ for N lobe. Assuming that you start with your magnetization entirely in the $\vec{x} - \vec{z}$ plane (though this is more likely to be generated by rotating spins by ${}^{\pi}/_2$ along $\vec{y}$ after some evolution in the $\vec{x} - \vec{y}$ plane), a train of $\pi_x - {}^{\pi}/_{2_y}$ pulses is applied at a fixed interval. For a given initial density matrix:

$$\rho_0 = c_z I_z + c_x I_x \tag{5.41}$$

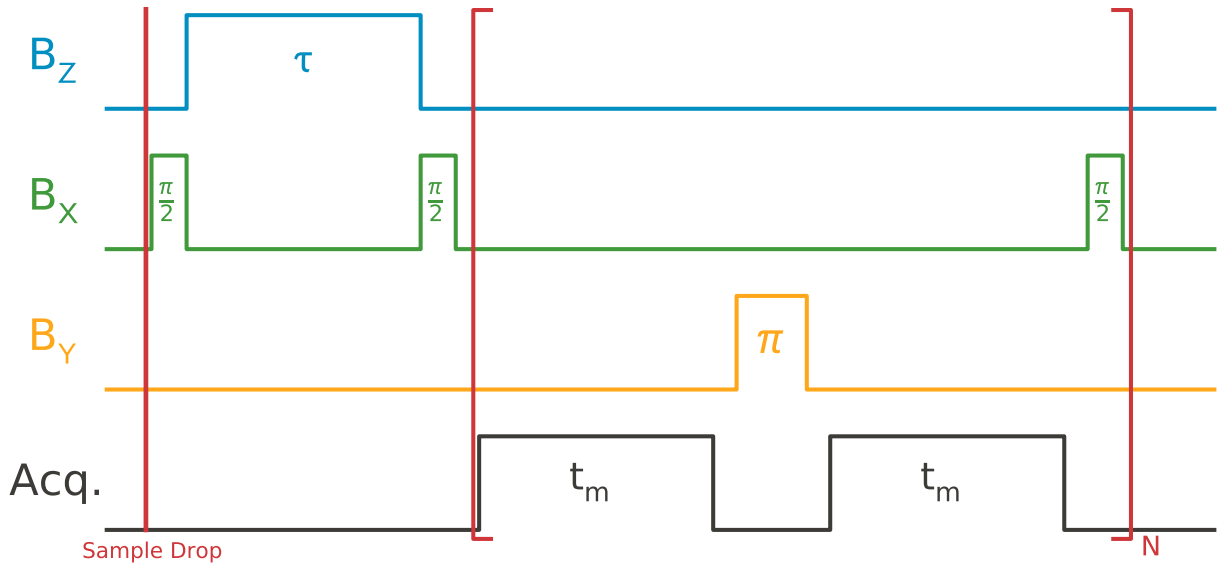


Figure 5.5: A sequence for acquiring quadrature magnetization signals.

The values of c_z and c_x are the real and imaginary components of the signal, respectively. The initial π_x pulse has no effect on the I_x spins, but flips the I_z spins, giving a difference of $2c_z$. After a $\pi/2_y$ pulse, the $\vec{z}$ component is tipped to be along $\vec{x}$, and the $\vec{x}$ component is tipped to be along $\vec{z}$, giving:

$$\rho = c_z I_x - c_x I_z$$

Flipping these spins with a π_x pulse gives a difference of $-2c_x$, and after one more $\pi/2_y$ pulse, the initial density matrix from Eqn. 5.41 is restored and the pulse cycle can be repeated indefinitely. One concern with this sequence is that there's an asymmetry between the real and imaginary channels due to decay in the direct dimension. If distortions arise, it is likely preferable to switch to a 2-transient acquisition mode, where the measurement is repeated twice, once measuring the real channel and the other measuring the imaginary channel. If, for some reason, this is undesirable but 2 transients can still be used (for example, in the case where data is being analyzed initially in real time with transients repeated somewhat infrequently), then it is likely worthwhile to phase cycle the sequence.

This is done by alternating the detected magnetization between being in the $\vec{x} - \vec{z}$ and $\vec{y} - \vec{z}$ plane, where in the first case the pulse sequence is $\left(\pi_x - \pi/2_y\right)_n$ (with initial pulse $\pi/2_x$ if the spins need to be rotated from the $\vec{x} - \vec{y}$ plane), and in the second case the pulse sequence is $(\pi_y - \pi/2_x)$ (with initial pulse $\pi/2_y$ if the spins need to be rotated from the $\vec{x} - \vec{y}$ plane). Alternatively, the sequence can be phase cycled without alternating between detection in the $\vec{x} - \vec{z}$ plane and $\vec{y} - \vec{z}$ plane by simply switching the order of the pulses between $\left(\pi_x - \pi/2_y\right)$ and $\left(\pi/2_y - \pi_x\right)$. When the first pulse is a π pulse, it should come *after* the first lobe of the

square wave, whereas when the first pulse is a $\pi/2$ pulse, it should come *before* the first lobe of the square wave. In both phase cycling methods, there are asymmetries between the two phases - method 1 (alternating between $\vec{x}-\vec{z}$ and $\vec{y}-\vec{z}$) has greater symmetry in the number of pulses applied and the type of pulses (e.g. lobe 2 always follows a π pulse, lobe 3 always follows a $\pi/2$ pulse, etc), whereas method 2 has greater symmetry in the pulse directions (e.g. all $\pi/2$ pulses are all along $\vec{x}$), and greater symmetry in the coordinate system (e.g. the components of the spin polarization are always constrained to the $\vec{x}-\vec{z}$ plane). In most cases, the two methods will have identical outcomes, and they should only be important in systems which have some particular inhomogeneities or pulse imperfections.

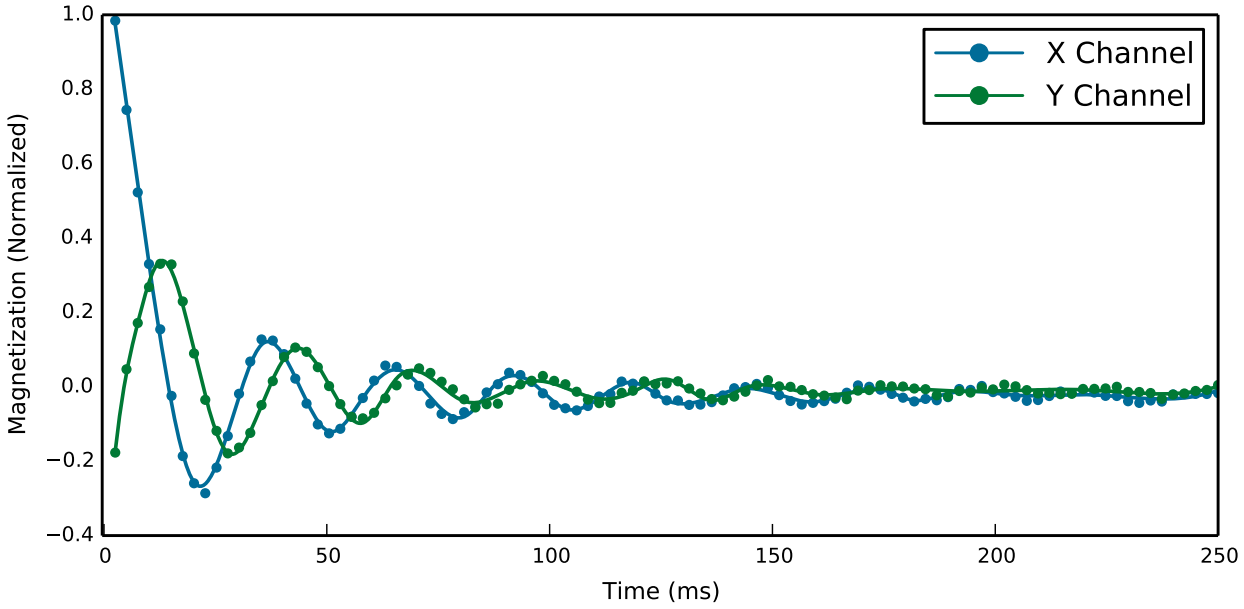


Figure 5.6: An FID acquired with the quadrature detection sequence.

5.4.3 Vector Signal Detection

The more generalized form of the quadrature detection scheme is a method for detecting the full vector spin polarization along all three components, using largely the same principles as those used in the quadrature detection. Because NMR precession usually occurs in a single plane, it is very unlikely that a vector detection sequence of this sort would be needed. It is most likely to be useful in the case of a sample after it has been passed through an unknown field, or when attempting to determine the direction of arbitrary offset fields.

As in the quadrature pulse sequence, the sequence is an alternating train of π and $\pi/2$ pulses along $\vec{x}$ and $\vec{y}$. Using a "pulse-conservative" approach, wherein every pulse applied switches between measurement states, it is not possible to return to the original state after cycling through the pulses once, as the cycle results in a shift of an odd permutation (e.g.

xyz → yzx or zyx), which cannot be reversed with a single $\frac{\pi}{2}$ pulse. However, repeating the same cycle twice (e.g. x→y→z→x→y→z) returns to the original permutation state (up to a phase). The simplest pulse sequence which accomplishes this, in general form, is:

$$\left(\tau_m - \pi\binom{x}{y} - \tau_m - \tfrac{\pi}{2}_{\pm\varphi} - \tau_m - \pi\binom{x}{y} - \tau_m - \tfrac{\pi}{2}_{\pm\bar{\varphi}}\right)_{3n} \tag{5.42}$$

In Eqn. 5.42, φ is either x or y. At each point in this subsequence, a binary choice is presented, either choosing between x or y, or choosing between $+\varphi$ and $-\varphi$ (or $\pm\bar{\varphi}$, as applicable). These choices affect only the sign of the outputs along each vector, while the choice of φ determines the permutation order of the outputs. The pulses transform the inputs (keeping track of phase) as in 5.43.

$$\begin{array}{ccc}
\overset{\pi\binom{x}{y}}{\longrightarrow} & \overset{\frac{\pi}{2}x}{\longrightarrow} & \overset{\frac{\pi}{2}y}{\longrightarrow} \\
I_x \longrightarrow \pm I_x & I_x \longrightarrow I_x & I_x \longrightarrow -I_z \\
I_y \longrightarrow \mp I_y & I_y \longrightarrow I_z & I_y \longrightarrow I_y \\
I_z \longrightarrow -I_z & I_z \longrightarrow -I_y & I_z \longrightarrow I_x
\end{array} \tag{5.43}$$

From this, we can determine the phase of each output (necessary for consistent data processing) and choose a sequence of pulse phases which returns the spins to their original state after 12 pulses. Assigning boolean values **A**-**D** to the binary choices presented in Eqn. 5.42, the resulting states, $\varphi = x$ and $\varphi = y$ are:

$$\begin{array}{cccccc}
 & \mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{D} & \\
I_x: & \binom{x}{+}_{-} & \binom{x}{+}_{+} & \binom{x}{+}_{-} & \binom{z}{-}_{+} & (\varphi = x) \\
I_y: & \binom{y}{-}_{+} & \binom{z}{+}_{-} & \binom{z}{-}_{-} & \binom{x}{+}_{-} & \\
I_z: & \binom{z}{-}_{-} & \binom{y}{-}_{+} & \binom{y}{-}_{+} & \binom{y}{+}_{+} &
\end{array}$$

$$\begin{array}{cccccc}
 & \mathbf{A} & \mathbf{B} & \mathbf{C} & \mathbf{D} & \\
I_x: & \binom{x}{+}_{-} & \binom{z}{-}_{+} & \binom{x}{+}_{-} & \binom{z}{-}_{+} & (\varphi = y) \\
I_y: & \binom{y}{-}_{+} & \binom{z}{+}_{-} & \binom{z}{-}_{-} & \binom{x}{+}_{-} & \\
I_z: & \binom{z}{-}_{-} & \binom{y}{-}_{+} & \binom{y}{-}_{+} & \binom{y}{+}_{+} &
\end{array}$$

Taking "true" to result in a phase of -1 and "false" to result in a phase of 1, and where a boolean "true" chooses the top choice (e.g. if **A** is true, the first pulse is π_x, otherwise π_y, if

B is true, the second pulse is $\frac{\pi}{2}+\phi$, etc) the state after a single application of the minimum subsequence unit is given as:

$$\begin{array}{cccc|cccc} & \varphi = x & & & & \varphi = y & & \\ c_xI_x & \longrightarrow & \neg\mathbf{ACD}\,(c_xI_z) & & c_xI_x & \longrightarrow & \neg\mathbf{ABD}\,(c_xI_y) & \\ c_yI_y & \longrightarrow & \mathbf{ABD}\,(c_yI_x) & & c_yI_y & \longrightarrow & \mathbf{ACD}\,(c_yI_z) & \\ c_zI_z & \longrightarrow & \neg\mathbf{BC}\,(c_zI_y) & & c_zI_z & \longrightarrow & \neg\mathbf{BC}\,(c_zI_x) & \end{array} \tag{5.44}$$

Repeating the sequence 3 times (with φ constant) cycles through all 3 states, and so if **A**, **B**, **C** and **D** are kept constant, all three axes accumulate the same phase, which is given by Eqns. 5.45 and 5.46:

$$\neg\mathbf{ABC} \oplus \mathbf{ABD} \oplus \neg\mathbf{BC} = 0 \tag{5.45}$$

$$\neg\mathbf{ABD} \oplus \mathbf{ACD} \oplus \neg\mathbf{BC} = 0 \tag{5.46}$$

In both cases, the phase cancels out, and the spins return to their original state with no phase, so long as **A**, **B**, **C** and **D** are constant throughout. It then only remains to determine which lobes correspond to which axial measurements - and what phase do these have. In the $\varphi = x$ sequence, these are given by Eqns. 5.47 ($\varphi = x$) and 5.48 ($\varphi = y$):

$$\begin{aligned} 2c_x &= \neg\mathbf{ACD}\,(l_{12n+4} - l_{12n+5})\,, \neg\mathbf{D}\,(l_{12n+10} - l_{12n+11}) \\ 2c_y &= \neg\mathbf{AB}\,(l_{12n+2} - l_{12n+3})\,, \neg\mathbf{BC}\,(l_{12n+8} - l_{12n+9}) \\ 2c_z &= (l_{12n} - l_{12n+1})\,, \mathbf{AC}\,(l_{12n+6} - l_{12n+7}) \end{aligned} \tag{5.47}$$

$$\begin{aligned} 2c_x &= \neg\mathbf{AB}\,(l_{12n+2} - l_{12n+3})\,, \neg\mathbf{BC}\,(l_{12n+8} - l_{12n+9}) \\ 2c_y &= \mathbf{ACD}\,(l_{12n+4} - l_{12n+5})\,, \mathbf{D}\,(l_{12n+10} - l_{12n+11}) \\ 2c_z &= (l_{12n} - l_{12n+1})\,, \mathbf{AC}\,(l_{12n+6} - l_{12n+7}) \end{aligned} \tag{5.48}$$

Here l_i is the magnitude of I_z at lobe i, for $i \in [0, N]$ where N is the total number of pulses applied. These can be grouped more easily if it is assured that the two measurements of each axis always have the same phase, as this reduces the minimum number of lobes in the repeating sequence. Since the phase accumulated between any two measurements is always **AC** in both $\varphi = x$ and $\varphi = y$, this condition is met so long as $\mathbf{A} = \mathbf{C}$. Although we're likely indifferent to the individual phases, for symmetry we can also choose to constrain the phases to always be positive by choosing $\mathbf{B} = \neg\mathbf{A}$ and $\mathbf{D} = 1$ in the case of $\varphi = x$, and $\mathbf{B} = \neg\mathbf{A}$, $\mathbf{D} = 0$ in the case of $\varphi = y$. With these constraints, the vector form is given by:

$$\begin{aligned} 2c_\varphi &= l_{6n+2} - l_{6n+3} \\ 2c_{\bar{\varphi}} &= l_{6n+4} - l_{6n+5} \\ 2c_z &= l_{6n} - l_{6n+1} \end{aligned} \tag{5.49}$$

As was the case in the quadrature detection scheme, since these lobes are separated in time by several τ, τ should likely be chosen such that $\tau \ll T_2$. If asymmetry between the channels is a problem, phase cycling between the $\varphi = x$ and $\varphi = y$ sequences may be somewhat helpful. If a 3-phase cycle is needed (i.e. x-first, y-first then z-first), it is likely better to make three separate scalar measurements of c_x, c_y and c_z, rather than a single vector measurement, as this will give much less distortion.

5.5 Indirect Detection and Field Cycling

Many experiments can be more easily performed in an indirect dimension by measuring changes in the sample magnetization (or in some cases, the spectrum) as a function of changes in some experimental parameter. The use of an indirect dimension is also particularly useful when the experiment is best performed in a magnetic field other than the detection field - such experiments are performed in a field cycled indirect dimension, wherein a field is applied during the experiment period which is removed during detection.

One simple example of a field cycled experiment is the detection of a free-induction decay curve along an indirect dimension. These signals can be time-consuming to acquire, and may as a consequence have poor spectral resolution, but they are often very useful for characterizing something about the sample (see Sec. 5.7) or the coils used to apply the bias field (see Sec. 5.6.2). The pulse sequence for an FID measurement is found in Fig. 5.7.

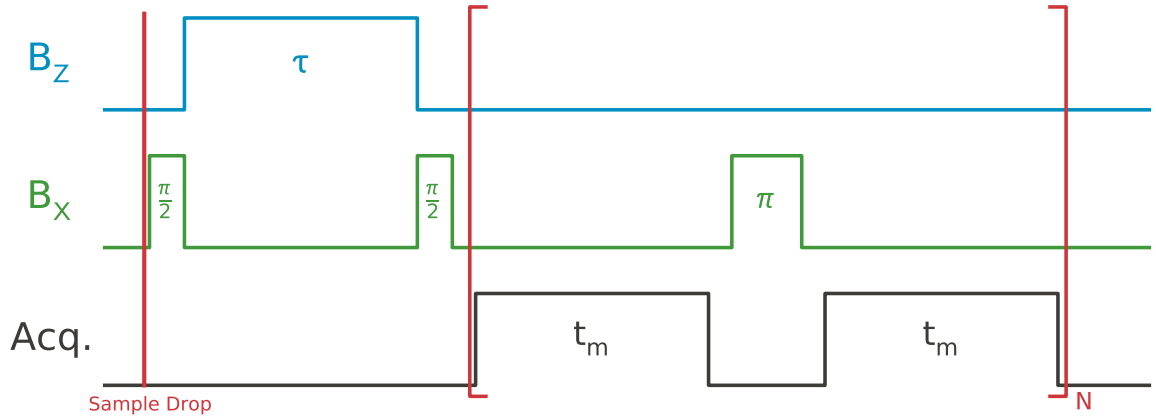


Figure 5.7: The pulse sequence for a free-induction decay measurement along a field-cycled indirect dimension.

The sample is prepolarized, and upon being dropped, the spins are pointing along $\vec{z}$, so a $\frac{\pi}{2}$ pulse is applied to tip them into the transverse plane. After the spins are tipped into the transverse plane, a field along $\vec{z}$ is applied for time τ, after which time a second $\frac{\pi}{2}$ pulse tips the spins along $\vec{z}$, after which one of the magnetization detection pulse sequences is applied (See Sec. 5.4) for more details), and the magnetization is plotted as τ is varied. The result of one such measurement (real channel only) is shown in Fig. 5.8.

These measurements are much more time consuming than detection in a direct dimension, as nearly the entire experiment needs to be repeated for each point in the experiment - which means waiting at least $3T_1$ for the sample to polarize, and in the case of prepolarized samples, adding twice the shuttling / transport time to the experimental time. In the case of magnetometers, it is possible to shift the detection frequency such that the detection is on-resonance at higher fields, allowing the experiments to be performed in a direct dimension - however, this eliminates much of the versatility of a zero-field detection scheme. Any time the magnetometer detection frequency is shifted, it will need to be calibrated at the higher field, and the pulse frequency and power will have to adjusted to match. In the indirect detection scheme, however, the bias field during the experiment can be varied arbitrarily and the magnetometer need not be re-calibrated. Additionally, the bias field is removed while the DC rotation pulses are applied, and so the pulses can be used without re-calibration at any experiment field.

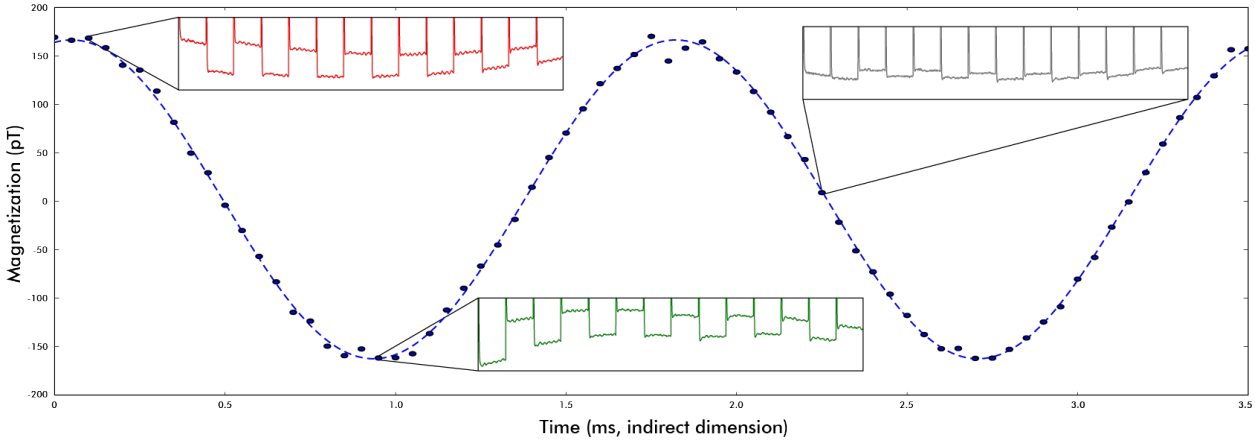


Figure 5.8: A free-induction decay curve acquired in an indirect dimension, field cycled to 139 mG

A sort of middle-ground between pure direct dimension detection and detection in an indirect dimension is the use of a windowed acquisition sequence. In such sequences, the experiment proceeds in the direct dimension, but is interrupted periodically for short periods to apply a brief measurement sequence. This allows the entire experiment to be acquired in a single transient in the direct dimension, but there is some trade-off between spending more time making measurements (increasing the signal-to-noise), making more measurements (increasing the resolution), and allowing the experiment to proceed without distortions from the field cycling. While these are some cause for concern, the greatest challenge in making windowed acquisition work is the dead time of the magnetometer. After a pulse is applied, the magnetometer signal becomes saturated, and so no measurements can be made until the alkali spins have had time to become polarized again (which depends on the pumping rate) - this is often on the order of 10-30 ms, while the pulses are on the order of 100 μs. The loss of 10-30 ms per pulse necessitates measurement times (τ_m) on the order of 50 ms per lobe, and so a single magnetization measurement can take up a significant fraction of the entire

experiment time (which is often on the order of a few seconds). With a faster re-polarization time, it would, however, be possible to do many of these experiments in a windowed direct dimension.

5.6 Pulse Strength Calibration

In order to know how long to apply pulses for, it's important to know the calibration of the pulse coils. This value depends on many factors - the coil configuration, the pulse circuit, the sample position and the nuclear gyromagnetic ratio, to name a few. For some of the more complicated pulse sequences, errors in these calibrations can become very important, as hundreds of pulses may be applied, or the signal may be very sensitive to small errors in the pulse length. As such, it is very important to have robust methods for finding the right pulse calibration.

5.6.1 Calculated

While calculations of the pulse strength will not necessarily provide low-error pulse calibration values, as they only address the idealized form of the coil geometry, they can make an excellent first-order approximation for what length you can expect from a given coil configuration. These sort of electromagnetic simulations can also give some insight into some aspects of the coil which may be difficult to estimate experimentally, such as the pulse inhomogeneity over the sample volume.

Using the same Biot-Savart law simulator used to generate the homogeneity assessments in Fig. 3.9, the strength and homogeneity of pulses in the region of interest (the sample), the expected FID for evolution under a given pulse can be calculated, and from the frequency of oscillation, the pulse lengths can be extracted. Since the input to the pulse coils is a voltage, but the coil response is a function of amperage, it is also necessary to simulate the pulse coil circuit to determine the correspondence between voltage in and current across the coil, this is one of the most likely sources of error in the calculation, as it may significantly depend on potentially hard-to-measure properties of the circuit components.

The calculated π pulse times for the x, y, and z coils using the standard 5 V input pulses are predicted from this approach as $t_{x,s} = 95.8\,\mu$s, $t_{y,s} = 90.4\,\mu$s and $t_{z,s} = 153.3\,\mu$s[3], with the differences arising because of differences in the size, resistances and inductances of the coils. When measured with the multipulse method described in Sec. 5.6.3.1, the π pulse times are determined to be $t_{+x,e} = 92.56\,\mu$s, $t_{-x,e} = 87.15\,\mu$s, $t_{+y,e} = 92.8\,\mu$s and $t_{-y,e} = 87.15\,\mu$; the $\pi/2$ pulse length for the positive 90 channel was similarly measured at $47.3\,\mu$s. Using the sample FID method described in Sec. 5.6.2, the z coil's response was determined to be $0.14\,\mathrm{G/V}$, which corresponds to a 5 V pulse time of $167.5\,\mu$s.

As expected, the calculated pulse times are roughly correct, in all cases within $\pm 10\,\%$ of the correct values. The fact that the negative pulses are stronger than the positive pulses

[3]Here the "s" subscript denotes simulated results, while "e" represents experimental results.

may be a coincidence, or may be due to a systematic difference – it is potentially the result of differences between the NPN and PNP transistors used in the amplification stage of the pulse coil; since the coils themselves are passive components, it seems unlikely that there is any systematic difference between current flowing in different directions across them. There is also a difference between the effective coil response between the π and $\pi/2$ pulses, with the $\pi/2$ pulse taking somewhat more than half as long as the π pulse. This is most likely explained by the effect of rise time in the coil, which is a decreasing fraction of the pulse as a function of time. An alternative explanation for this difference would be some sort of heating effect in components of the circuit, but this seems unlikely, because the experimental investigation reveals that the pulse time is not effected by duty cycle, as long as the time between pulses is longer than the ringdown time. Because the SPICE simulator does simulate the transient response of the circuit and coil, if more accurate calculations are required, it would be possible to incorporate this information into the model.

5.6.2 Sample FID

A fairly accurate, but time-consuming way of measuring pulse length can be found by measuring the sample's free-induction decay (FID) as it evolves under the field produced by the pulse coil. Sample FID offers two primary advantages over other methods of pulse strength calibration. The first advantage is that the shape of an FID can give information about the homogeneity of the pulse and also the effects of the pulses' rise and fall times. Pulse distortions are likely to be greatest for very small t_p, whereas these distortions will average out over the course of a longer pulse. Additionally, by taking longer-term measurements (likely since these detections occur in an indirect dimension, these should be heavily aliased to reduce the number of samples), the FID curves and/or the Fourier transform spectra can be used to determine T_2^* (the inhomogeneous broadening from the pulse coil).

As such, while the pulse angle should be linear in the duration of the pulse (and, as such, fitting to 2-3 points should be enough to calibrate pulses of arbitrary angle), the true signal may be have a non-linear response, particularly for short pulses. Likely the best way to characterize the nature of this non-linearity is by taking a sample FID, which measures the angular frequency response as a function of pulse duration. This also relates to the second advantage, which is that the FID simultaneously measures the pulse time for a wide array of angles, whereas the much more commonly used Pulse Train method (Sec. 5.6.3) is only used to calibrate for a single pulse angle (though, very accurately).

The data are acquired in an indirect dimension, and since they use magnetization detection, require that the pulses already be somewhat calibrated (poorly calibrated pulses will cause aliased precession in the direct dimension, leading to reduced magnetization signal, though the FID can certainly still be detected without issue), as such, it is likely best to use the multiphase π train described in Sec. 5.4.1, as this can reduce pulse-related errors.

5.6.3 Pulse Train

The most common and in many cases the best way to calibrate the strength of the pulse strength is by applying a train of the pulses to a sample and observing the results. Unlike the sample FID method, which can give a decently accurate prediction of the length of a pulse of arbitrary angle, pulse train measurements can be used to determine the pulse length of a pulse of a specific angle to nearly arbitrary precision - in most cases the pulse calibration will be limited by the clock speed of the pulse generator, not by uncertainty in the calibration process itself. That said, depending on the number of pulses to be calibrated (usually it's best to start with at least 4: π_x, π_y, $\frac{\pi}{2}_x$ and $\frac{\pi}{2}_y$) and the degree of accuracy, this can also become quite time consuming.

The pulse train experiment should generally be designed like a magnetization detection experiment where the pulse time is varied over some plausible range. For calibrating pulses $\theta \neq n\pi$, it is preferable to repeat the pulse n times where $n = \frac{m\pi}{\theta}$ where m is an odd number - under this condition, the magnetization should be inverted after each pulse when the appropriate pulse time calibration has been found. Caution should be taken to leave a sufficiently long space between pulses to allow the coil to fully discharge, otherwise the fact that multiple pulses are being applied in a row will affect the calibration, and an error can be introduced. Generally the appropriate inter-pulse spacing can be found by repeating the same experiment a few times with different inter-pulse spacings to gauge the magnitude of the effect.

Such a pulse-acquire-pulse-acquire sequence is equivalent to making a measurement in the direct dimension of the spin precessesion under the pulse - and choosing pulses close to the duration of a π pulse is equivalent to sampling such that the signal precession is the same as the Nyquist frequency - which should alias the signal down to 0 Hz, and any precession that occurs is indicative of either a pulse error or a bias field (see Sec. 5.6.3.1 for more details on how to deal with a bias offset). For a candidate pulse $t_c = t_\pi + t_\varepsilon$, where t_π is the optimal π pulse length and t_ε is the error in the pulse length, the resulting signal should effectively oscillate at frequency $\omega_\varepsilon = \gamma B_0 {}^{t_\varepsilon}/_{\tau_m}$. Because relaxation occurs during the τ_m period as well as during the pulses, the linewidth depends on both choice of t_c and τ_m. The magnetization decay and precession after 1 t_c pulse and 1 τ_m measurement period, assuming that T_2^* is roughly constant, is given by Eqn. 5.50:

$$M_1 = M_0 e^{-\frac{t_c(1+\tau_m/t_c)}{T_2^*}} e^{-i\gamma B_1 t_c} \tag{5.50}$$

As such, the effective T_2^* of the measurement is $T_{2p}^* = {T_2^*}/{(1+\frac{\tau_m}{t_c})}$, and the observed linewidth will be ${(1+\frac{\tau_m}{t_c})}/{\pi T_2^*}$. This essentially limits the accuracy of such measurements to $\omega_\varepsilon < T_{2p}^*$, as slow precession on the timescale of relaxation is very difficult to accurately measure. However, this is really a somewhat "soft" limit on the error, as more pulses can be applied before another measurement is taken - switching from a train of $\pi + \varepsilon$ to $3\pi + 3\varepsilon$ will give a signal precessing at effectively 3 ω_ε, and so the pulse error can be reduced by a factor of just under 3. Longer trains of pulses will, of course, increase t_c relative to τ_m, and a maximum is reached when $t_c = {\tau_m T_{2C}^*}/{T_{2M}^*}$, where T_{2C}^* is the T_2^* during the t_c pulse

and T_{2M}^* is the T_2^* during the measurement. This value can easily be less than the limiting condition where the period of the pulse generator's clock is less than $\gamma t_\varepsilon B_1$, since at that point uncertainty is limited by the ability to pulse accurately. It is not worth increasing the accuracy of t_c beyond this point, as the pulses cannot be applied with sufficient precision anyway.

5.6.3.1 Multiple Pulse Method

The major problem with the pulse train method is that it cannot distinguish between a small error in the pulse time and an uncorrected bias offset field along the direction of the pulse. While some bias offsets can be removed using the applied oscillation method (see. Sec. 3.4.2.1), the vector signal is sensitive to light shifts along the direction of the pump and probe beams, and so the field which "zeros" the magnetometer signal along those directions is in fact applying a real magnetic field that the sample spins will feel to counter an effective magnetic field that they will not. The effective frequency as a function of t_c is then given by Eqn. 5.51:

$$\omega = \gamma B_1 t_\varepsilon + \gamma B_r \left(t_c + \tau_m\right), \tag{5.51}$$

where B_r is the residual bias field. One way to solve this problem is to keep t_c constant and measure ω as a function of τ_m. The slope of this line should be γB_ϵ, which can be zeroed out and the experiment repeated until the slope is constant. This can also be done empirically, by varying τ_m in the direct dimension and varying the bias offset field until there is no precession during any value of τ_m - if this is to be done, it is best to switch τ_m only after an even number of equal-sized lobes, as this largely retains the spin-echo properties of the π train, whereas alternating each lobe between multiple values of τ_m hampers echo formation, and so fields along the two other orthogonal dimensions must be simultaneously corrected as well.

This method may work for zeroing the field, but then the pulse error becomes tied to the precision of the shim coils. This may not be problematic, but a much more accurate pulse calibration method - which works largely independent of the bias offset field - is to perform the pulse train calibration multiple times, varying the number of π pulses applied before each measurement. The signal after N pulses is

$$M_N = (-1)^N e^{-Nt_c/T_{2C}^*} e^{-iN\gamma B_1(t_\varepsilon) - i\gamma B_\epsilon t_c}. \tag{5.52}$$

As can be seen from Eqn. 5.52, the pulse error t_ε is a function of N, but the evolution under the bias offset field is not. As such, in a measurement of ω as a function of t_c and N, the t_c at which all values of ω_N intersect is the point where t_ε. Once this is found, it can be used to zero field as detailed in Sec. 3.4.2.2 on the NMR sample precession field zeroing method.

5.7 Gradient Strength Calibration

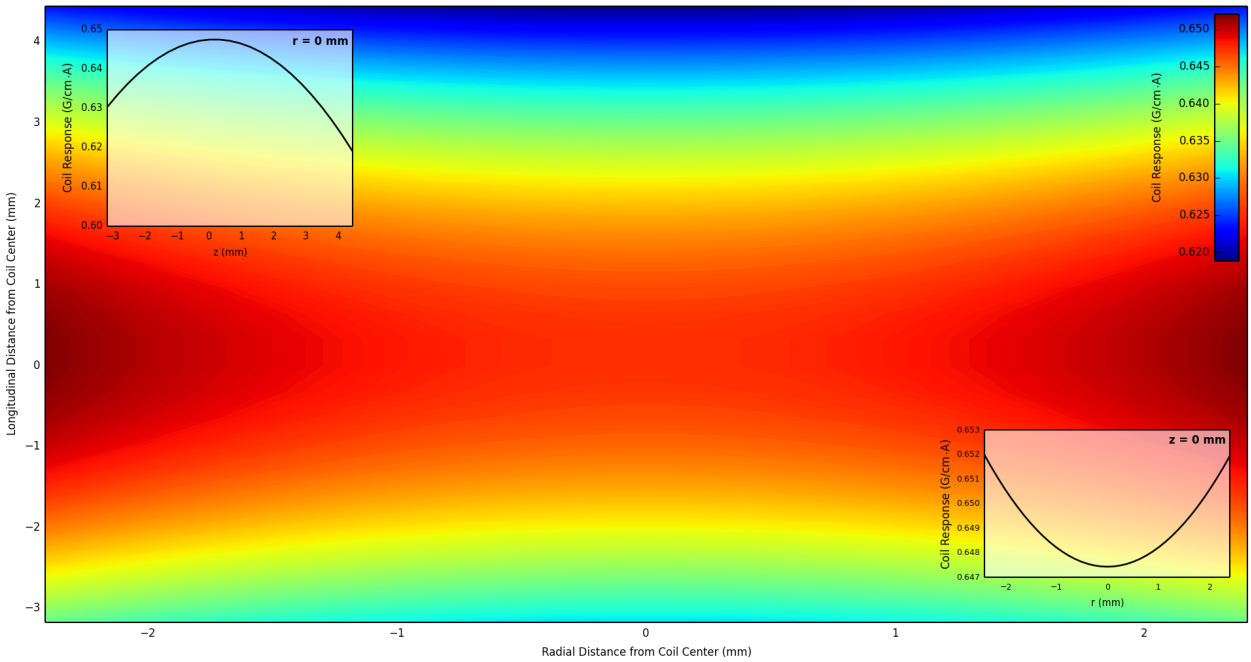


Figure 5.9: A contour map of the calculated gradient field generated in the region of the sample by the anti-Helmholtz pair used in the diffusion experiments. Inset at the top left is a projection of the gradient strength along the longitudinal axis, and inset at the bottom right is a projection along the radial axis.

The accuracy of diffusion (and imaging) experiments depends on having an accurate calibration of the strength of the magnetic field. Since the gradient strength is set by the analog voltage outputs, it depends both on the response of the coils and the coefficient of the voltage-to-current DC pulse circuit, described in Sec. 3.3.5. As with the pulse coils, this can be determined to first order from electromagnetic calculations, but this many not adequately capture deviations from ideality in the pulse coil or circuit.

The coils are centered 0.125 " (0.3175 cm) above the top of the heater; the thickness of the probe head cap at its thinnest is 0.0125 " (0.03175 cm), and the NMR tube glass is 0.38 mm thick, so the lowest sample position is 0.098 " (2.478 mm) below the center of the gradient. The results of Biot-Savart law calculations for the anti-Helmholtz pair in this region are shown in Fig. 5.9. The weighted average value of the gradient in this region is 0.64 $^{\mathrm{G}}/_{\mathrm{cm\cdot A}}$, with no more than 0.4 % deviation within this volume.

Likely the best way to confirm the gradient strength is to take an image (1-dimensional) with known dimensions and determine the gradient strength from the correspondence between frequency and position,

$$\Delta x = \frac{\Delta f}{2\pi\gamma G}, \tag{5.53}$$

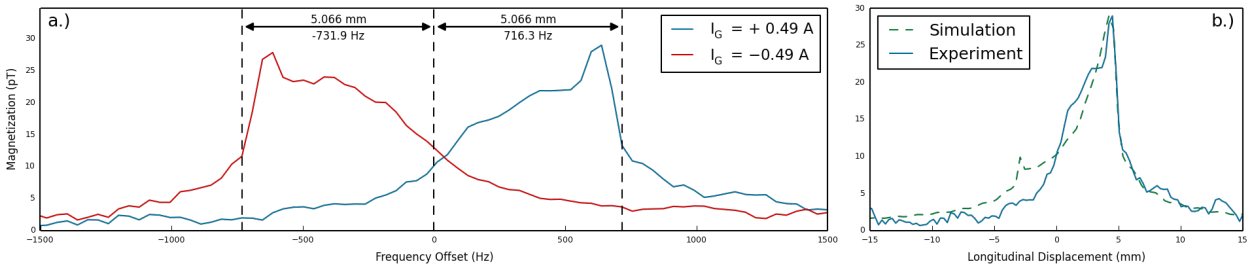


Figure 5.10: a.) An image taken of a sample 5 mm NMR tube filled with 100 μL of water while applying a 0.5 G bias offset field and $\pm$0.49 A across the gradient coil. b.) The lineshape is compared to one calculated from a Biot-Savart simulator taking into account the simulated gradient, sample volume and beam intersection volume.

which can be arranged to solve for G as such:

$$G = \frac{\Delta f}{2\pi\gamma\Delta x}. \tag{5.54}$$

Ideally, a phantom with distinct features of known size varying across its length would be used to displace sample, effectively allowing individual gradient measurements along the entire volume of the phantom. Due to time pressure and the difficulty of constructing a phantom that would stay in place while being shuttled, this approach was not feasible, so instead an image was taken of a sample of known volume (100 μL) using both the positive and negative gradient coils, and a small bias offset (0.5 G) intended to truncate concomitant gradients. Inverting Eqn. 6.30 (introduced in a more thorough discussion of sample volumes in Sec. 6.5.2.2), and confirmed by measurement with a micrometer, this corresponded to a height of 7.88 mm. Because the gradient coils are centered 2.478 mm above the bottom of the sample, there should be a precipitious drop-off in signal 5.07 mm from the center frequency of the FID. The results, based on an FID with 201 points sampled in an indirect dimension at 20 kHz are shown in Fig. 5.10. The result was that the positive gradient calibration was found to be 0.68 $^{\mathrm{G}}/_{\mathrm{cm\cdot A}}$ and the negative gradient calibration was found to be 0.69 $^{\mathrm{G}}/_{\mathrm{cm\cdot A}}$[4]

5.8 Sample Temperature

Because the cell operates at elevated temperature, even with active air cooling the NMR samples are often somewhere above room temperature - and with liquid nitrogen-based cooling they may in fact be below room temperature. Many effects, important among them being relaxation and diffusion, depend strongly on temperature, and this should be accounted for when analyzing data. However, making direct measurements of the sample temperature

[4]The reason these are different is likely because of differences in the circuitry. Since the pulse controller uses a voltage-controlled current source, in reality, the gradient calibrations were calculated to be 0.0443 $^{\mathrm{G}}/_{\mathrm{cm\cdot V}}$ and 0.0452 $^{\mathrm{cm\cdot V}}/$, respectively, but this cannot be compared directly with the Biot-Savart calculations without an analysis of the circuit.

in situ can be quite problematic - the sample needs to be able to move, and so a thermocouple cannot be inserted directly into it for monitoring, many layers of the instrumentation must be opaque, and so optical monitoring of the temperature is problematic, and because the sample is both actively being air-cooled and shuttled, the equilibrium temperature of the sample will depend, to some extent, on the rate of equilibration and the duty cycle (up vs. down).

Generally to combat the effect of sample heating over the course of an experiment, experiments were designed such that either the duty cycle did not change over the course of the experiment, though in some situations this was not necessarily possible and may have introduced small drifts in the signal into the samples. The biggest observed temperature change artifact was generally during the first few transient acquisitions, after the sample had been allowed to reach thermal equilibrium at room temperature, and was thus took a few transients to reach a new thermal equilibrium. To combat this, generally the sample was run through several "dummy" experiments, wherein no data are acquired before starting the real experiment. Generally pulses were still applied in these experiments, in case any of the pulse components were similarly undergoing heating which might change their effective coil response.

Without active liquid N_2 cooling (described in more detail in Section 3.2.3), typical temperatures observed were on the order of 35-37 °C, and with no air-cooling, the system tended to equilibrate closer to 60 °C.

5.8.1 Relaxation-based Measurement Method

The easiest and most robust way of measuring temperature — which we use as a standard in our experiments — is by using a calibrated NMR relaxation signal. Water relaxation, in particular, is extremely sensitive to temperature, and so by measuring a calibration curve of temperature vs. NMR relaxation constants, water from the same source (relaxation is also highly dependent on impurities) can be used as a standard thermometer. Unfortunately, for the same reason that it is not possible to continuously monitor the temperature of the samples, the instrument itself could not be used to make these calibration measurements, and so a Magritek Terranova-MRI Earth's field NMR system was used to make standard calibration curves. Because water's T_1 and T_2 are also quite sensitive to magnetic field at low fields, all relaxation-based temperature measurements made in the magnetometer were field-cycled to 0.435 G, which was the field in which the Terranova experiments were performed.[5] The results are summarized in Table 5.1.

To make these measurements, a special sample bottle was prepared, a glass bottle insulated with several layers of fiberglass insulation. The bottle had two caps, one unmodified cap to be used during the measurements, and a second cap with a thermocouple inserted through a hole and held in place with epoxy. The second cap was used to monitor the temperature between experiments. The water was boiled in a standard electric tea kettle and

[5]The Terranova system has two T_1 measurement sequences, measuring "B_P" and "B_E", B_P measures T_1 recovery in the Terranova's prepolarizing field, while B_E measures T_1 decay in Earth's field.

Temp. (°C)	T_1 (s)	T_2 (s)
27	2.5	2.24
39	3.5	3.19
54	4.5	4.9
64	5.7	6.2
84	7.8	6.7

Table 5.1: The T_1 and T_2 measurements of DI water from UC Berkeley's Stanley Hall, performed at 0.435 G.

poured into the glass bottle, then allowed to cool to room temperature over the course of several hours. As cooling processes follow an exponential decay, there is more uncertainty in the highest temperatures - the reported temperatures are an average of the temperature measured immediately before and after the experiment. This is not an ideal way to perform this experiment, as the temperature uncertainty is compounded by the fact that water's T_1 is inversely proportional to temperature, thus necessitating long experiments at the highest temperatures (which are cooling the fastest). An experiment with active temperature control of the sample would be much more accurate.

The standard model for the temperature dependence is a bi-exponential rate equation of the form:[47,58,59]

$$\frac{1}{T_1} = A_1 e^{-E_1/RT} + A_2 e^{-E_2/RT}. \tag{5.55}$$

From the somewhat sparse data in Table 5.1, fitting the data gives $A_1 = 1.303{\cdot}10^{-4}$ s, $A_2 = 8.49{\cdot}10^{-4}$ s, $E_1 = 44.5\,{}^{\text{kJ}}/_{\text{mol}}$, $E_2 = 14.9$ kJmol, which are generally consistent with literature values measured at higher fields.

5.9 Detectable Sample Region and Sample Geometry

Unlike in a traditional NMR spectrometer or a SQUID magnetometer, where the sample sits entirely within the detector volume, atomic magnetometers are inherently one-sided *ex-situ* devices which measure the magnetic field generated by the region surrounding the detector. Additionally, in a simple one-detector setup, the detection efficiency dies off as ${}^{1}/_{r^3}$ where r is distance from the detector. Generally, the size of the most sensitive region of the magnetometer is proportional to the size of the detector - which in practice means (approximately) the intersection volume of the beams within the cell.

Additionally, the geometry of the NMR sample tubes is such that the most sensitive region (the bottom of the tube) is curved, losing signal where it is most important and somewhat complicating the signal profile. Assuming the curved bottom of the NMR tube is approximately hemispherical, the lost detection volume is $\frac{1}{3}\pi r^3$ where r is the radius of the tube, which for a 5 mm diameter NMR tube with 0.381 mm wall thickness is $\approx 9.96\,\mu L$.

Generally, signal closer to the signal is much stronger than signal far from the detector. The signal at the origin from a dipole $\vec{\mu}$ at $\vec{r}$ is given by

$$B(\vec{r}) = \frac{\mu_0}{4\pi} \frac{(\vec{\mu} \cdot \vec{r})\vec{r} - |\vec{r}|^2 \vec{\mu}}{|\vec{r}|^5}. \tag{5.56}$$

The $\vec{z}$ component of this volume is

$$B_z(\vec{r}) = \frac{\mu_0 \mu_z}{4\pi} \frac{3\cos^2(\varphi) - 1}{|\vec{r}|}. \tag{5.57}$$

Eqn. 5.57 is positive inside a cone with angle θ_m, where θ_m is the magic angle, $\theta_m = \cos^{-1}(1/\sqrt{3}) \approx 54.7356\,^\circ$ and negative outside that cone. In general, picking out the positive lobe of a cone defined by φ_0, the signal from a magnetized volume gives

$$\int_0^R \int_0^\pi \int_{-\varphi_0}^{\varphi_0} \frac{3\cos^2(\varphi) - 1}{r^3} \mathrm{d}\varphi \sin(\theta) \mathrm{d}\theta r^2 \mathrm{d}r = 2\log(R) \left[\varphi_0 + \tfrac{3}{2}\sin(2\varphi_0)\right]. \tag{5.58}$$

The area inside the θ_m cone contributes $2\log(R)\left[2\theta_m + \sqrt{2}\right]$ to the signal, and the remaining area on the positive lobe ($-\frac{\pi}{2} \leq \varphi \leq \frac{\pi}{2}$) contributes $\pi - 2\theta_m - \sqrt{2}$, which is $\approx 32.6\%$ of the maximum signal.

Additionally, curved surfaces are much stronger than flat surfaces in response to impact (for one thing, the surface area of a hemisphere is $4\pi r^2$, whereas for a flat surface, it's $2\pi r^2$, so in a curved surface, the force is distributed about twice as much area), and so the glass can be thinner as a result. It is likely an empirical question what thickness tubes will be usable, and depends on many factors.

Chapter 6

Relaxometry and Diffusometry

6.1 Overview

Relaxometry and diffusometry are important tools for the characterization of heterogeneous materials and porous media[60], with applications including the study of biological systems,[61,62] medical imaging, food characterization[40,63] and oil-well logging.[36,37] These methods can be extremely effective in applications where high-resolution NMR is either unnecessary, impractical or both. The pulse sequences used to make these measurements are often robust even in the presence of sample motion, strong gradients and pulse imperfections,[34,64] allowing for the use of magnetic resonance in extreme environments or under conditions with rigid design constraints. Relaxometry and diffusometry are often used when making *ex-situ* measurements - wherein the geometry of the usual NMR experiment is inverted and a detector is used to either probe the surface of a larger sample or it is entirely surrounded by the sample.[38,39,65] In these cases, the volume and geometry within which it is possible to create a homogeneous field are limited, and in general it is not feasible to generate the field strengths necessary to make high resolution NMR measurements; however, relaxation and diffusion depend on many environmental factors including chemical composition, pressure[66], temperature,[47,58] phase[67,68] and even sample geometry,[69] and as such measurements of these processes can yield considerable information about the sample.

6.2 Spin-Lattice Relaxation (T_1)

Spin-lattice relaxation is the mechanism by which non-equilibrium spin polarization returns to its equilibrium state by energy exchange with the environment (called the lattice). In traditional high-field experiments, this takes the form of spins tipped into a transverse plane re-aligning with the bias offset field, and would be measured using an inversion-recovery or saturation-recovery experiment. In low-field experiments, however, the initial polarization induced by the prepolarizing magnet is much higher than the endpoint polarization in small bias fields (see Sec. 6.2.3.2 for more details), and so spin-lattice relaxation causes

re-equilibration to near-zero polarization and thus represents approximately a pure loss of spin polarization.[30]

6.2.1 Pulse Sequence

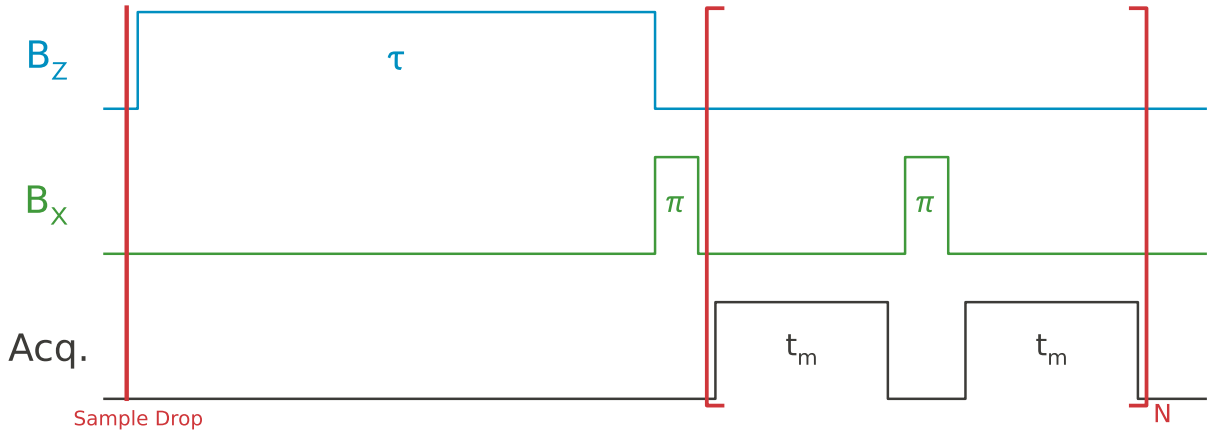


Figure 6.1: Pulse sequence for indirect acquisition of T_1 at low field.

The T_1 measurement consists of prepolarizing a simple, field cycling to the desired field and after a time τ measuring the remaining spin magnetization. This is always done in an indirect dimension to allow for greater flexibility in the experimental design - the bias field (B_z in Fig. 6.1) can be set to an arbitrary value between ≈ 0 and 1 G without the need to recalibrate pulse length or power, or to recalibrate the magnetometer.

For a prepolarizing field B_{PP} and sample transit time t_t, the full equation for signal intensity as a function of τ is given by Eqn 6.1:

$$M(t) = \left[\frac{\gamma B_{PP}}{k_B T} e^{-t_t/T_{1,t}} + \frac{\gamma B_{tt}}{k_B T}\left(1 - e^{-\tau/T_{1,B_Z}}\right)\right] e^{-\tau/T_{1,B_Z}} + \frac{\gamma B_Z}{k_B T}\left(1 - e^{-\tau/T_{1,B_Z}}\right), \tag{6.1}$$

where T_{1,B_Z} is the T_1 in field B_z and $T_{1,t}$ is the effective T_1 during the shuttling period. For $B_z, B_{tt} \ll B_{PP}$, this simplifies to Eqn 6.2:

$$M(\tau) = \frac{\gamma B_{PP}}{k_B T} e^{-t_t/T_{1,t}} e^{-\tau/T_{1,B_Z}}. \tag{6.2}$$

The results are then fit using a standard exponential $Ae^{-\tau/T_1}$, where A should be constant with respect to τ (more on when this assumption breaks down in Sec. 6.2.3).

6.2.2 Temperature Dependence

The mechanisms by which spin-lattice relaxation occurs can depend on temperature, with the temperature effect being particularly pronounced in the case of water.[59] This is particularly

important in these experiments, wherein the magnetometer cell is maintained at an elevated temperature. Small changes in the insulation properties of the heater or probe can lead to significant changes in the temperature of the liquid samples. Moreover, because the samples are pneumatically shuttled between the polarization and detection regions, it is very difficult to undertake in line temperature measurements during experiments.

Assuming the samples reach equilibrium sufficiently quickly (or, once reached, do not cool significantly between measurements), the temperature dependence of T_1 and T_2 can be used as a measure of the sample temperature. This is explored in more detail in Sec. 5.8.1.

6.2.3 Uncertainty

There are several sources of uncertainty in these relaxation measurements of varying importance. The most prominent mechanism for uncertainty is error on the measurements themselves - this is an error on the magnetization, which can be occasionally difficult to definitively characterize. Likely the most accurate way to characterize the error is by taking the standard deviation of repeated measurements, but this raises some question about what is to be considered a measurement. Magnetization measurements consist of a series of direct-dimension measurements of the magnetization, which are aggregated into a sort of average magnetization value over the direct dimension. Most forms of random noise are largely self-similar, and so the two methods of noise characterization should be roughly equivalent. However, one of the greatest difficulties in characterizing the uncertainty comes from the fact that many of the greatest sources of noise in our experiments are discrete noise, which only averages if the phase is shifting over time. For $N \to \infty$, the noise from discrete sources will converge to 0, but in the short run, the noise characteristics can be unpredictable, and there may be significant differences between the noise on each direct-dimension measurement and the noise on the cumulative measurements.

6.2.3.1 Temperature uncertainty

One significant potential source of uncertainty is temperature shifts. The T_1 as a function of temperature is parametrized as:[47,58,59]

$$\frac{1}{T_1} = A_1 e^{E_1/RT} A_2 e^{E_2/RT} \tag{6.3}$$

E_1 and E_2 are activation energies for the two contributions to the intermolecular relaxation, and our measured values of $E_1 = 4.45{\cdot}10^4\,{}^{\mathrm{kJ}}/_{\mathrm{mol}}$ and $E_2 = 1.49{\cdot}10^4\,\mathrm{kJmol}$ are in strong agreement with measurements at 1.4 T[59] and 3.4 T. The pre-exponential factors at 0.435 G were found to be $A_1 = 1.30{\cdot}10^{-9}\,\mathrm{s}$ and $A_2 = 8.49{\cdot}10^{-4}\,\mathrm{s}$, which also has strong agreement with the higher-field experimental data. Propagating the error in T gives:

$$\sigma_{T_1} = \left| \frac{A_1 E_1 e^{E_1/RT} + A_2 E_2 e^{E_2/RT}}{RT^2 \left(A_1 e^{E_1/RT} + A_2 e^{E_2/RT}\right)^2} \sigma_T \right| \tag{6.4}$$

For the same reasons that it is difficult to measure the temperature in the first place, though, it is even more difficult to measure the temperature fluctuations, and so a robust measure of σ_T cannot be made. Assuming a scenario of fluctuations on the order of $\sigma_T = 2\,\mathrm{K}$ (likely the equilibrium fluctuations are smaller than this) at $\approx 38\,\mathrm{K}$, the error σ_{T1} would be $\approx 0.178\,\mathrm{s}$ (4.6%). Even for $\sigma_T = 10\,\mathrm{K}$, the error σ_{T1} would be $0.89\,\mathrm{s}$ (20%). Even here, temperature fluctuations will likely average out over the course of a full experiment.

6.2.3.2 Decay Endpoint

These exponential decays are modeled as a decay to zero signal, because polarization on the order if 1 G is extremely low. Technically, this is a source of error in the system, and so it is necessary to examine the underlying assumptions to ensure that the approximation is justified. The general form of an exponential decay is:

$$f(t) = f_\infty - (f_\infty - f_0)\, e^{-rt} \tag{6.5}$$

Where f_∞ is the steady-state value of the decaying variable, f_0 is the initial value, and r is the decay rate (in our case, $r = {}^1\!/_{T_1}$), so the true equation for a T_1 decay from a state polarized at 1.8 T to the 1 G steady state is:

$$P_{1.8\mathrm{T}} e^{-t/T_1} + P_{1\mathrm{G}} \left(1 - e^{-t/T_1}\right) \tag{6.6}$$

Assuming polarization and measurement at 300 K (in reality it will be somewhat higher, giving a reduced steady state polarization), the initial and steady-state polarizations are:

$$P_{1.8\mathrm{T}} = \tanh\left(\frac{\hbar\gamma_H(1.8\mathrm{T})}{2(300\mathrm{K})k_B}\right) = 6.13\cdot 10^{-6} \quad P_{1\mathrm{G}} = \tanh\left(\frac{\hbar\gamma_H(10^{-4}\mathrm{T})}{2(300\mathrm{K})k_B}\right) = 3.4\cdot 10^{-10} \tag{6.7}$$

Assuming the effective field corresponding to the initial polarization at 1.8 T is 600 pT, the error from polarization of $3.4{\cdot}10^{-10}$ corresponds to 30 fT, which is near to the fundamental limit of sensitivity, and smaller than most other sources of uncertainty on the T_1 measurement. If this were leading to significant errors, however, there are two simple enough methods to counter-act it. The first is to simply fit to a model which includes the offset to account for the error - but this adds some model error into the equation, wherein an additional improper fit parameter might allow for "over-fitting" to the noise. A fit-neutral way to counter-act this effect is to simply use an even number of transients, and on odd-numbered transients, apply a π pulse to the spins before the decay begins and subtract even-transient magnetization from the odd-transient magnetization. This gives:

$$\begin{aligned} S(t) &= P_{1.8\mathrm{T}}\left(1 - e^{-t/T_1}\right) + P_{1\mathrm{G}} e^{-t/T_1} - \left(-P_{1.8\mathrm{T}}\left(1 - e^{-t/T_1}\right) + P_{1\mathrm{G}} e^{-t/T_1}\right) \\ &= 2P_{1.8\mathrm{T}}\left(1 - e^{-t/T_1}\right) \end{aligned} \tag{6.8}$$

This cancels out the effects from the decay endpoint, giving a clear exponential decay.

6.3 Spin-Spin Relaxation (T_2)

Another form of relaxation is the longitudinal relaxation, which results in a coherently precessing pack of spins begins to lose phase coherence. Assuming the direction of measurement is along $\vec{x}$[1], the signal is proportional to the $\vec{x}$ component of the sum of the vectors:

$$S(t) \propto \sum_j \langle I_{j,x}| \rho(t) |I_x\rangle = \sum_j \cos\left[\omega_0 t + \phi_j(t)\right] \tag{6.9}$$

For spins in a perfectly homogeneous field, $\omega_j = \omega_0$ and $\phi_j(t) = 0$ and the spins precess perfectly coherently. Even assuming that the bias offset field B_z were perfectly homogeneous, the spins would still accrue differential phases ϕ_j over time due to dynamic spin-spin interactions, such as transient dipole-dipole interactions. The rate at which spins dephase due to spin-spin interactions is a property of a given sample in a given system (the rate is also dependent upon field and temperature, for example).

6.3.1 Spin Echo Pulse Sequence

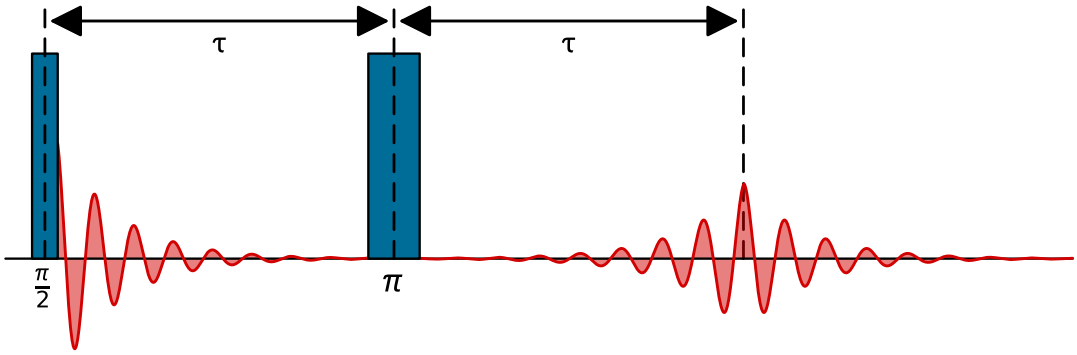


Figure 6.2: The simplest T_2 pulse sequence - a single pulse is applied and the signal at 2τ (at the echo) is measured as a function of τ.

The observed transverse relaxation constant T_2^* is affected by both the dynamic inhomogeneities induced by spin-spin interactions and by static inhomogeneities in the bias offset field. Since the static inhomogeneities are a product of the system and not of the sample, T_2^* is not generally a useful measure, except for when characterizing NMR measurement systems. However, the asymmetries between static and dynamic inhomogeneities allows T_2 to be measured, even in the presence of inhomogeneous broadening, using a spin-echo pulse sequence (see Fig 6.2). When a π_x pulse is applied to N spins initially along I_x after time

[1]In our case we measure in the direct dimension along $\vec{z}$, but T_2 measurements are made in an indirect dimension, and magnetization along $\vec{x}$ is transformed to lie along $\vec{z}$ for the purposes of the measurement.

τ, the $\vec{y}$ and $\vec{z}$ components are inverted, effectively applying mirror reflections about $\vec{x}$. Assuming each spin is precessing in its own local field B_{zj} at its own frequency ω_j, the density matrix after time τ is:

$$\rho(\tau) = \sum_j \cos(\omega_j \tau)\, I_x - \sin(\omega_j \tau)\, I_y. \tag{6.10}$$

The degree of spread in ω_j will determine the coherence of the spin-evolution, and so if ω_j has a wide spread, there will be no signal after a short τ. After applying a π_x pulse, the density matrix is:

$$\rho(\tau + t_p) = \sum_j \cos(\omega_j \tau)\, I_x + \sin(\omega_j \tau)\, I_y \tag{6.11}$$

Assuming that the set $\{\omega_j\}$ does not change after the pulse, the density matrix after a second period of evolution of time τ serves to return the spins to their original state:

$$\begin{aligned}
\rho(2\tau + t_p) &= \sum_j \cos(\omega_j \tau)\left[\cos(\omega_j \tau)\, I_x - \sin(\omega_j \tau)\, I_y\right] + \sin(\omega_j \tau)\left[\cos(\omega \tau)\, I_y\right] \\
&= \sum_j \left[\cos^2(\omega_j \tau) + \sin^2(\omega_j \tau)\right] I_x + \left[\cos(\omega_j \tau)\sin(\omega_j \tau) - \cos(\omega_j \tau)\right] I_y \\
&= \sum_j I_x = N I_x.
\end{aligned} \tag{6.12}$$

Because of the transient nature of the spin-spin interactions, the dynamic field inhomogeneities are not reversible, as the local field experienced by each spin changes before and after the pulse. Calling $\tilde{\omega}_j$ the frequency of spin evolution for spin j, the density matrix after 2τ is given not by Eqn. 6.12, but by Eqn. 6.13:

$$\rho(2\tau + t_p) = \sum_j \cos\left[(\omega_j - \tilde{\omega}_j)\tau\right] I_x + \sin\left[(\omega_j - \tilde{\omega}_j)\tau\right] I_y \tag{6.13}$$

And since this is linear in $\omega_j - \tilde{\omega}_j$, even in the presence of transient effects, all static inhomogeneities cancel out, leaving *only* the dyanmic effects, allowing for measurement of T_2 rather than T_2^*. The simplest T_2 detection pulse sequence (in an indirect dimension), then, is a single-echo T_2 measurement, as seen in Fig. 6.2.[70].

6.3.2 CPMG Echo Train Pulse Sequence

One problem with the single echo pulse sequence used in Sec. 6.3 is that not all irreversible effects depend only on the sample - this happens most commonly when spins are moving with respect to a static inhomogeneity, and as a result there is some asymmetry between the phase accrued before and after the π pulse. Most of these effects can be alleviated using the

Carr-Purcell-Meiboom-Gill (CPMG) sequence, which consists of a train of π pulses spaced at $(2n-1)\tau$ (for $n \in [1, N]$), rather than a single pulse. This serves to refocus the pulses frequently, and prevents much phase from accruing between echoes.[71,72]

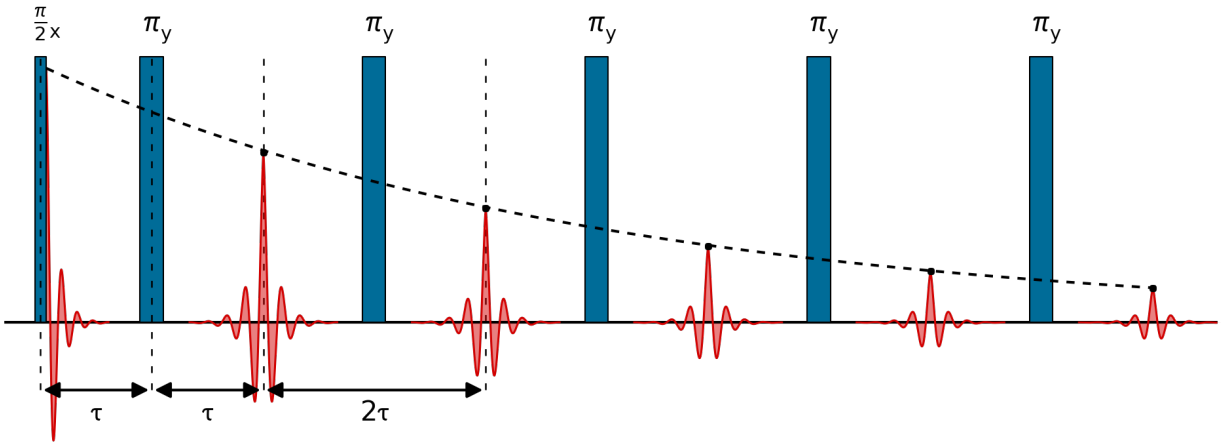
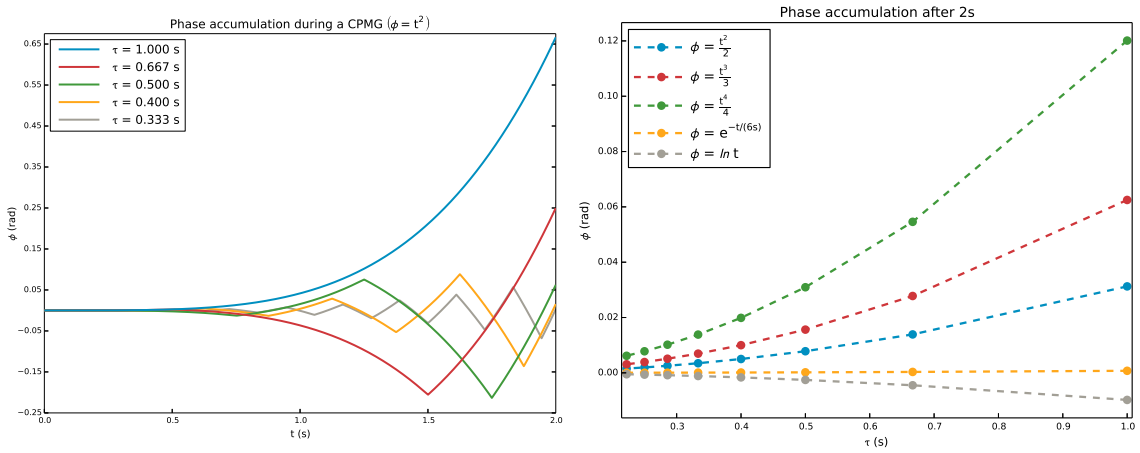


Figure 6.3: The CPMG sequence is used for making T_2 measurements.

In addition to the moment nulling effects described in Sec. 6.3.2.1, the CPMG pulse sequence allows for spins to be refocused before large phase differences accrue. If a sufficiently short τ is chosen such that the phase accrued over a 2τ period is well-approximated by a linear function (i.e. if the phase is locally linear over the interval 2τ), most of the inhomogeneities will refocus under the CPMG pulse, as seen in Fig. 6.4.

There are two ways to use this to more accurately measure T_2 - in one case, the single-echo sequence is modified to have a large number of pulses of duration τ, and τ is varied over the course of the experiment, and in the other, the signal at each echo (or some subset of echoes) is measured over N measurements with a constant τ. While the first method is in a sense an improvement over the single-pulse case, there are still many non-linear effects introduced by this method - the effects from convection and diffusion scale as τ^2, and in a constant-N approach, the fraction of time the spins spend in the pulse field is also changing as a function of τ. In a constant-τ approach, even if practical constraints prevent a sufficiently short τ to fully eliminate these effects they are at least made significantly more linear, reducing their effects on the signal.



(a) (b)

Figure 6.4: The affect of a CPMG sequence on phase accumulation for several non-linear functions of t. (a) Although $\phi = e^{-Rt}$ is a non-linear function of t, for $\tau \ll R$, the change in the function is small enough that very little phase accumulates. (b) The phase accumulation after $N\tau$ =2 s as a function of τ for gradients characterized by several non-linear functions - the degree of phase accumulation is essentially related to the linearity of the function over a given 2τ interval.

6.3.2.1 Moment Nulling

One problem that arises is that the sample is flowing in some way - either due to convection effects, actual fluid flow or sample jostling. In our magnetometry experiments, convection effects and stirring due to sample impact were serious concerns - both causing some mass nuclear movement in the presence of some potential real gradients. For τ sufficiently short that the spins can be thought of as moving in a linear gradient, we can model the phase accrual after τ as:[71]

$$\phi_j = \int_{\tau_1}^{\tau_2} \left[\omega_j + \sum_{m=1} (m+1) \left(\alpha_j^{(m)} t^m \right) \right] \mathrm{d}t \tag{6.14}$$

Where $\alpha_j^{(m)}$ are a set of coefficients representing phase accrual due to the m^{th} derivative of the position as a function of time, i.e. $\alpha_j^{(1)}$ relates to the phase contributed by the velocity, $\frac{\partial x}{\partial t}$, and $\alpha_j^{(2)}$ represents the phase contributed by the acceleration, etc.[2] This evaluates on the interval $[\tau_1, \tau_2]$ as

$$\phi_j = \omega_j \tau + \sum_{m=0} \alpha^{(m)} \left(\tau_2^{(m)} - \tau_1^{(m)} \right). \tag{6.15}$$

For a system limited to only 1st order (velocity) effects, the phase for spin j after τ is $\phi_j = \omega_j \tau + \alpha_j^{(1)} \tau^2$. Applying a π_x pulse, the mirror reflection inverts the phase, so

[2] I have arbitrarily defined the set $\alpha_j^{(m)}$ such that the integer factor is included after integration, which is why the sum is defined with each term multiplied by $m + 1$.

$\phi'_j = -\omega_j\tau - \alpha_j^{(1)}\tau^2$. After a second period of evolution (between time τ and 2τ), the spins pick up an additional phase $\omega_j\tau + \alpha_j^{(1)}\left[(2\tau)^2 - \tau^2\right] = \omega_j\tau + 3\alpha_j^{(1)}\tau^2$. Which gives a total phase of $\omega_j\tau + 3\alpha_j^{(1)}\tau^2 - \omega_j\tau - \alpha_j^{(1)}\tau^2 = 2\alpha_j^{(1)}\tau^2$ - as expected, the asymmetry due to the motion of the spins remains, even when the static inhomogeneities represented by ω_j are canceled out. However, if another π pulse is applied at 3τ, the first order term cancels out at the second echo:

$$\begin{aligned}\phi &= 2\alpha_j\tau^2 \xrightarrow{2\tau \qquad 3\tau} \omega_j\tau + 7\alpha_j\tau^2 \xrightarrow{\pi_x} -\omega_j\tau - 7\alpha_j\tau^2 \\ &= \xrightarrow{3\tau \qquad 4\tau} \omega_j\tau - \omega_j\tau - 7\alpha_j\tau^2 + 7\alpha_j\tau^2 = 0. \end{aligned} \tag{6.16}$$

And when the spins have completely refocused, the same exact analysis applies as if the spins are at $\tau = 0$ at the second echo, and so it is clear that the zeroth order (static) inhomogeneities refocus on each echo, while the second order inhomogeneities refocus every other echo. The higher-order components, however, do not naturally refocus using a simple CPMG pulse. At the n^{th} echo, the phase accrued from the 2^{nd} order effect ($\phi_n^{(2)}$) is given by the sum

$$\begin{aligned}\frac{\phi^{(2)}}{\alpha_j^{(2)}\tau^3} &= (-1)^n \sum_{m=1}^{n} (-1)^m \left[(2m)^3 - 2\,(2m-1)^3 + (2m-2)^3\right] \\ &= (-1)^n \sum_{m}^{n} (-1)^m\,(12m - 6) = 6n. \end{aligned} \tag{6.17}$$

(6.18)

From Eqn. 6.18 it can be seen that the 2^{nd} order effect is reduced from a polynomial effect to a linear effect at each echo. This is simple to deal with, because linear phases cancel out after a single echo - by applying a π pulse *during* the 2^{nd} echo sequence (when both the 0^{th} and 1^{st} order effects have been canceled), at $n = 4$, the 0^{th}, 1^{st} and 2^{nd} order effects are fully refocused at 8τ. If the same method of analysis is applied to the 3^{rd} order effect, it is found that using this pulse sequence results in a linear phase accruing on even echoes - the pulse sequence can be then modified such that the first 4 components cancel after 8 echoes by removing all the π pulses from every other echo (at which point the first 3 components are zero anyway, and so are unaffected by whether or not a pulse occurs). Consider the modified pulse sequence

$$(\tau - \pi_x - \tau - \varphi(i))_n; \tag{6.19}$$

in this sequence, φ_i is the phase applied at the i^{th} repetition of the sequence. For optimal cancellation, φ_i is

$$\varphi_i = \pi\left(\frac{1 + (-1)^{v_2(i)}}{2}\right), \tag{6.20}$$

where $v_2(i)$ (sometimes called $ord_2(i)$) is the *2-adic valuation* of i - the multiplicity of the number 2 in the prime factors of i. Put another way, for $i = 2^k j$ where $i, j, k \in \mathbb{Z}$ and j is an odd number, $v_2(i) = k$. This can easily be calculated by counting how many times $i > 0$ can be divided by 2 without returning an odd number. Using this pulse sequence, at the $n^{\mathbf{th}}$ echo, all components $m \leq v_2(n)$ are 0 (e.g. at $n = 160 = 5 \cdot 2^5$, $\phi^{(0)} = \phi^{(1)} = \phi^{(2)} = \phi^{(3)} = \phi^{(4)} = \phi^{(5)}$).

6.3.2.2 Pulse-time fraction

One last concern to address is the fraction of time that spins feel the pulse field. At high field, this is not a significant effect, as pulses are very small compared to the main bias offset field (B_0). The situation is different, however, at low field, where the strength of the pulses is the same order of magnitude as (or larger than) the bias field. This can may cause significant shifts in the measurement of T_2 if care is not taken. Because the relaxation operations commute with one another (i.e. $M_0 e^{-\left(t^{(2)} - t^{(1)}\right)/T_2^{(1)}} e^{-\left(t^{(1)} - t^{(0)}\right)/T_2^{(0)}} = M_0 e^{-\left(t^{(1)} - t^{(0)}\right)/T_2^{(0)}} e^{-\left(t^{(2)} - t^{(1)}\right)/T_2^{(1)}}$), the overall relaxation constant $\overline{R}_2 = 1/\overline{T}_2$ can be expressed as the integral over the relaxation constants during the period in question - which is the continuous-form equivalent of a time-weighted average of the relaxation times:

$$e^{-\overline{R}_2 t} = \prod_i e^{R_{2,i}\Delta t_i} = e^{-\sum_i \Delta t_i R_{2,i}} \longrightarrow \lim_{n\to\infty} e^{\sum_i \Delta t_i R_{2,i}} = e^{\int_0^t R_2(t)\mathrm{d}t} \tag{6.21}$$

$$\overline{R}_2 = \int_0^t R_1(t)\mathrm{d}t. \tag{6.22}$$

Taking the greatest difference we've observed in water T_2 as a "worst case" estimate of the effect, we will assume that we're interested in measuring the zero field T_2 ($T_{2,zf}$), and pulses are applied at 27 mG where $t_p \approx 27\,\mathrm{ms}$[3]. At $\approx 37\,^\circ\mathrm{C}$, $T_{2,zf} \approx 2.88\,\mathrm{s}$ and $T_{2,27\mathrm{mG}} \approx 2.39\,\mathrm{s}$. Assuming the relaxation is only changing in response to the different field at the pulse, then:

$$\overline{R}_2 = \frac{t_p T_{2,zf} + t_m T_{2,27\mathrm{mG}}}{T_{2,27\mathrm{mG}} T_{2,zf}\,(t_p + t_m)} \tag{6.23}$$

Where t_m is the time between pulses. Given the length of t_p, the most probable value for t_m is $\approx 73\,\mathrm{ms}$ (such that $t_p + t_m = 100\,\mathrm{ms}$), so $\overline{R}_2 = 0.362\,\mathrm{s}^{-1}$ and $\overline{T}_2 = 2.52\,\mathrm{s}$ - an error of $\approx 5\%$. In a typical case, a T_2 measurement would be made at 0.5 G using pulses either at 114 mG ($t_\pi = 1.03\,\mathrm{ms}$) or at 1.1 G ($t_\pi = 107\,\mu\mathrm{s}$). Under the same conditions as in the earlier analysis, at 0.5 G, $T_2 \approx 2.88\,\mathrm{s}$. At 114 mG, $T_2 \approx 2.59\,\mathrm{s}$, and for typical value $\tau = 20\,\mathrm{ms}$, the measured $\overline{T}_2$ would be 2.86 s a bias of $\approx 0.5\%$. At 1.1 G, $T_2 \approx 2.91\,\mathrm{s}$, and for typical value $\tau = 5\,\mathrm{ms}$, the measured $\overline{T}_2$ would be 2.885 s, an error of $\approx 0.2\%$. Although the

[3]This is actually a quite low pulse field. Standard pulses are generally on the order of 100 μs, and "low intensity" pulses are on the order of 1 ms

difference in T_2 can be arranged to be an insignificant source of error, it is still necessary to consider during data processing, as often when applying long CPMG trains the fraction of time spent pulsing is a considerable fraction of the pulse sequence. Additionally, it is worth noting that these calculations make the assumption that the spins are relaxing due to the homogeneously-broadened T_2, not the inhomogeneously-broadened coil T_2^* - which may be *much* shorter. This, however, can be alleviated by alternating the direction of the π pulses between $\vec{x}$ and $\vec{y}$, which act to mutually correct their static inhomogeneities, for reasons detailed in the context of pulse error correction in Sec, 5.4.1.1.

6.4 Diffusometry Measurements - Methods

While it may necessitate error correcting pulse sequences, the fact that spins moving through static inhomogeneities are imperfectly canceled by spin echo pulse sequences is quite useful for measuring fluid flow (as in many velocity and acceleration imaging applications) and diffusion. Like T_1 and T_2, the diffusion properties of spins is an inherent property of a system, and can be used as a proxy for many important measurements like temperature, viscosity, fat content in dairy products, etc.[40,62]

6.4.1 Pulse Sequences

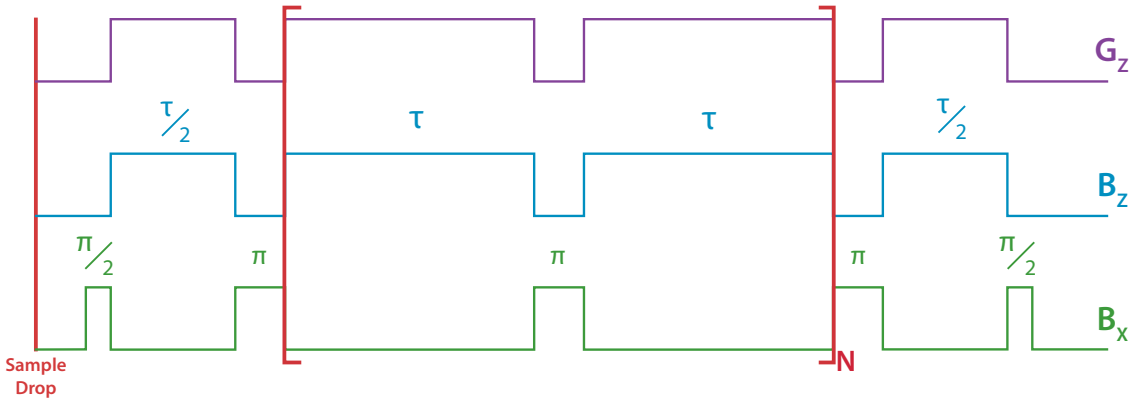


Figure 6.5: The most basic diffusion pulse sequence is a CPMG pulse sequence in the presence of an applied gradient G_z.

Unlike in the examples of fluid flow and convection, wherein a persistent direction of flow causes phase to grow (on most time-scales) fairly linearly - or at least by a function which is well-fit to a low-order polynomial - diffusing spins are effectively executing a random walk, and the position of a given diffusing spin is increasingly uncorrelated with its starting position. The non-refocused phase accumulation for spins diffusing in a linear gradient of strength G after an echo ($t = \tau$) comes from the root-mean-square distribution of the spins[4],

[4] $\sqrt{\langle x^2 \rangle} = \sqrt{\langle y^2 \rangle} = \sqrt{\langle z^2 \rangle} = \sqrt{2Dt}$

and is given in Eqn. 6.24:

$$\phi_D = (\gamma G D)^2 \frac{t^3}{3} \tag{6.24}$$

Thus the signal after time τ is:

$$M(t) = M_0 e^{-t/T_2} e^{-(\gamma G)^2 D\tau^3/3} \tag{6.25}$$

Assuming that the different periods of evolution commute (which should be true for the assumed motion in a linear $\partial B_z/dz$ gradient), repeated applications of the spin echo sequence add linearly, and so after a CPMG pulse train of length N, the signal is:[73]

$$M(t) = M_0 e^{-t/T_2} e^{-N(\gamma G)^2 D\tau^3/3} \tag{6.26}$$

$$= M_0 e^{-t/T_2} e^{-(\gamma G\tau)^2 Dt/3} \tag{6.27}$$

Where Eqn. 6.27 is found by substituting $t = N\tau$ into Eqn. 6.26. When making T_2 measurements, it is necessary to choose a τ sufficiently small that diffusion effects are minor, whereas when measuring diffusion, it's necessary to make measurements which minimize the signal asymmetries due to anything but diffusion. In Eqn. 6.27 the signal is proportional to three different control variables - N, τ and G. Varying either G or τ gives a one-sided Gaussian decay (e^{-x^2}), while varying N gives a simple exponential decay (e^{-x}), however varying N or τ also necessarily varies t, and the exponential in Eqn. 6.27 can be expressed as $-t\left[1/T_2 + (\gamma G\tau)^2 D/3\right]$, and so the output will not be a pure function of diffusion. That said, it may be acceptable to vary t when $(\gamma G\tau)^2 D/3 \gg 1/T_2$. In general, though, the greatest symmetry is achieved by acquiring signal while varying G_z, as the pulse sequence is identical save for the gradient strength - which affects only moving components of the spin echo sequence, and with the right choice of pulse sequence, effects like convection or residual angular momentum in the sample can be easily nulled.

One major concern that can be limiting is the choice of N and τ - which determine the speed of the Gaussian decay, as well as the total time t of each run. Ideally G_z will be swept at least through the linear region of the curve, and preferably until the signal fully decays. For example, to sweep through the curve from 0 decay due to diffusion up to at least 95% decay due to diffusion, N, τ and G_z should be chosen such that $(\gamma G_{max}\tau)^2 Dt/3 \geq -\ln(0.95) \approx -0.0513$. Generally G_{max} will be the limiting factor - either the fundamental limits of the pulsing circuit, the coil or the shields will prevent the gradient from increasing too much and remaining linear. Holding γG_{max} constant, the equation can then be expressed as:

$$N\tau^3 \geq -\frac{3\ln(0.05)}{\gamma^2 G_{max}^2 D} \tag{6.28}$$

Unfortunately, with this approach, a full T_2 curve is not measured, and so all the signal earlier in the sequence is eliminated. As a result, $t = N\tau$ should be minimized with respect to Eqn. 6.28, and N should be constrained to be an even multiple of 4 or, if possible, 8, as

this allows for nulling of the first few orders of motion terms. Minimizing $N\tau$ for a given value of $N\tau^3$ is clearly done by using the smallest possible value of N, as the minimized function grows linearly in N, but as the cube root of τ^3, and so the solution is then to choose $N = 4$ or 8 (depending on the strength of the higher-order corrections), so τ is given by:

$$\tau \geq \sqrt[3]{\frac{-3\ln(0.05)}{N\gamma^2 G_{max}^2 D}} \tag{6.29}$$

For a typical value of G_{max} =0.4 G/cm, the self-diffusion coefficient for water at T=300 K is D =2.35·10^{-5} cm^2/s,[74] $\tau_4 \approx 94$ ms and $\tau_8 \approx 75$ ms with $t_4 = 376.5$ ms and $t_8 = 597.5$ ms. For $T_2 \approx 2.4$ s, this represents a loss of 15% and 22% of signal, respectively. Ideally, the gradient strength would be greatly increased, allowing for much shorter values of τ - this is mainly problematic in some shielded magnetometry applications due to the fringe field from the gradient, which can serve to magnetize the innermost layer of shielding.

6.5 Measurements of Hydrocarbons and Water

Relaxometry and diffusometry are used extensively in portable and *ex-situ* NMR applications, as these techniques can be applied even in grossly inhomogeneous fields for characterization of bulk heterogeneous materials.[60,64] These techniques for the basis for nearly all NMR methods for oil well logging, where the "sample" is fluid in rock formations surrounding a well bore. As the Earth's field is generally homogeneous over very large regions, Earth's field detection has the potential to vastly increase the effective sample volume of NMR measurements performed without a bias offset field.

To this end, we performed a number of experiments using an atomic magnetometer to measure relaxation and diffusion in samples of pure hydrocarbons and water at Earth's field as an initial proof of concept that atomic magnetometers have the chemical sensitivity required to quantitatively and qualitatively distinguish between hydrocarbons and water. The experiments are described in detail in Sections 6.2, 6.3 and 6.4 - all are performed in an indirect dimension at 0.5 G. Although all measurements were made at zero field, T_1 and T_2 measurements are sensitive to the magnetic field and so during the experimental evolution 0.5 G was chosen to best emulate the strength of the Earth's ambient magnetic field; diffusion is not sensitive to magnetic field strength,[5] but the degree to which concomitant gradients are truncated by the presence of a bias offset field will indeed be sensitive to the bias offset field.

6.5.1 Pure solvents

The first test performed was to measure relaxation and diffusion in an array of pure solvents to show that the technique has sufficient chemical resolution to distinguish between an ar-

[5] At least not in solvents. It is conceivable that magnetic field strength could have an effect on diffusion in ferrofluids or other magnetic liquids.

ray of substances. Measurements were performed on a number of solvents including water, methanol, ethanol and an array of alkanes of increasing chain length. Because T_1 and T_2 are functions of both temperature and magnetic field and these experiments were performed at both a lower magnetic field (0.5 G) and a higher temperature (37 °C, due to proximity to the cell) than is common, literature values are not generally available for comparison.

6.5.2 Mixtures

As our experiments are intended as a proof-of-concept of the utility of magnetometry at Earth's field for the measurement of heterogeneous materials, it was desirable to make measurements of mixtures of hydrocarbons and water. However, the relatively small sensitive radius of our particular magnetometer (see Sec. 5.9), and the fact that the magnetometer is located directly *below* the sample is a problem, as any physical mixture of immiscible liquids will separate into a bilayer, with the denser material on bottom (close to the sensor) and the lighter material on top, leading to a bias in any quantitative analysis. As such, it was necessary to develop a strategy wherein there was either no longitudinal bias in the sample distribution, or a known and characterized bias for which we can compensate. The latter approach is undesirable, as it introduces model error into the quantification.

6.5.2.1 Emulsions

The first approach we took in addressing the problem of mixture separation was to use some surfactant in an attempt to create stable emulsions of hydrocarbon solvents and water. The most effective emulsifying agent was a small amount of dish soap added to the water, followed by long periods of sonication. A longer period of sonication leads to the creation of smaller bilayer domains, and significantly increased the stability of the emulsion. After an hour or more of sonication, decane-water emulsions stored at room temperature and at rest were, by visual inspection, stable for over a day, and 1-dimensional relaxation measurements made of the samples were both qualitatively and quantitatively accurate. However, the stable lifetime of the emulsion was found to be significantly reduced at the elevated temperature necessitated by close proximity to the magnetometer cell, combined with the physical disruption provided by the rapid pneumatic shuttling of the sample. Because in our system 2-dimensional relaxation measurements were taken over the course of 12-24 hours, this introduced artifacts into the measurement as the sample components slowly separated out over the course of the measurement.

6.5.2.2 Physical Separation

Physically separating the layers using a longitudinal barrier can cause some loss of overall signal, as the barrier will take up some of the sample volume, but has several distinct advantages over emulsions. In addition to solving the issue of the stability of the separation, it also preserves the physical properties of the separated components, and so the T_1, T_2 and

diffusion measurements are unlikely to be affected by the domain size of the emulsion or interactions with the surfactant.

Ideally, we would have used specialized NMR tubes with a thin longitudinal barrier to evenly separate the two substances, but the much more practical solution was to use nested tubes, such as those used in high-field susceptibility matching experiments. In our experiments, we used 3.3 mm coaxial inserts from Wilmad-Labglass (part number 517-INNER), which was fit into a standard 5 mm NMR tube (part number 517-OUTER), and kept in place with a small spacer (part number 517-SPACER). Using a coaxial insert, there is some asymmetry between the inner and outer sample volumes. Assuming a hemispherical bottom with the same radius r as the radius of the cylindrical portion of the tube, the volume of liquid in an NMR tube filled to height $h > r$ is given by:

$$V(h) = \pi r^2 \left(h - {}^{r}/_{3}\right). \tag{6.30}$$

With a coaxial insert with inner radius r_I and glass thickness t_I in a larger NMR tube with inner radius r_O, the volume of the inner NMR tube to height h is thus

$$V_i(h) = \pi r_I^2 \left(h - t_I - {}^{r_I}/_{3}\right), \tag{6.31}$$

and the volume outside the NMR tube is

$$V_o(h) = \pi h \left[r_O^2 - (r_I + t_I)^2\right] + \frac{\pi}{3}\left[(r_I + t_I)^3 - r_O^3\right]. \tag{6.32}$$

In our experiments, we used 4.97 mm NMR tubes with thickness 0.38 mm, and coaxial inserts with 3.3 mm outer diameter and 0.1 mm thickness, so $r_O = 2.11$ mm, $r_I = 1.55$ mm, $t_I = 0.1$ mm. For all practical purposes, all signal comes from the bottom 2 cm of the tube. In this configuration, $V_I = 146\,\mu$L, $V_O = 102\,\mu$L, and so 41% of the fluid is in the outer volume, with the remaining 59% in the inner volume. The total volume of the first 2 cm of the larger NMR tube is 372 μL, so approximately 33% of signal is lost due to the presence of the coaxial insert.

6.5.3 Results

6.5.3.1 Pure Solvents

Measurements of these samples were necessarily made at elevated temperatures due to the high temperature environment in the vicinity of the cell. Active cooling with dry nitrogen (see 3.2.2) was used to bring the samples down to ≈37 °C. As a proof of concept of the sensitivity of the technique, T_1 and T_2 were measured for a variety of different solvents at 0.5 G. The results are summarized in Table 6.1.

The measurements were limited to water and small-chain alkanes and alcohols because larger molecules tend to have a faster relaxation rate, and the 700 ms effective T_1 filter imposed by the pneumatic shuttling time places a lower limit on the measurable relaxation times. Despite this, our experiments demonstrate a high degree of chemical sensitivity, with

Compound	T_1 (s)	T_2 (s)	D $(\cdot 10^{-5} cm^2 \cdot s^{-1})$
Water	3.27 ± 0.15	2.08 ± 0.13	2.6 ± 0.3
Methanol	2.88 ± 0.08	2.50 ± 0.05	3.7 ± 0.2
Ethanol	2.07 ± 0.06	2.07 ± 0.06	2.6 ± 0.1
Hexadecane	0.75 ± 0.03	0.65 ± 0.02	—
Limonene	1.62 ± 0.04	1.40 ± 0.10	2.5 ± 0.2
Heptane	1.91 ± 0.04	1.70 ± 0.07	7.0 ± 0.7
Octane	1.83 ± 0.08	1.67 ± 0.03	4.6 ± 0.4
Decane	1.56 ± 0.07	1.32 ± 0.08	1.46 ± 0.07

Table 6.1: Relaxation times and diffusion coefficients of some common solvents measured at 0.5 G and approximately 37 °C

signal-to-noise sufficiently low that it is possible to confidently distinguish between almost any two components based only on their measured T_1 and T_2 values. All single component data were fit using a least squares curve fit to a single exponential decay.

Diffusion data are extremely sensitive to temperature, and because the temperature of the sample could not be monitored externally, it is hard to determine the accuracy of these values. These data were gathered on different days, likely using different nitrogen flow conditions and potentially different cell temperatures, and so it is not unlikely that the temperatures at which these data were acquired would be radically different.

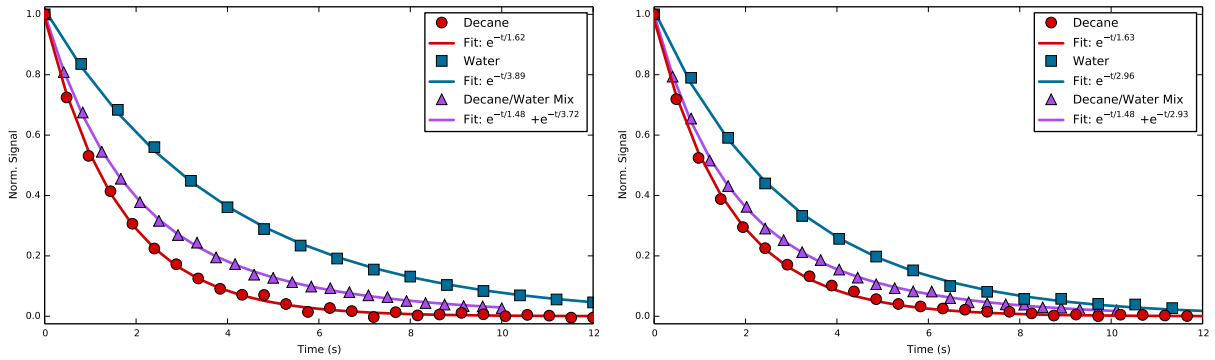
6.5.3.2 Mixtures

More directly relevant to the problem of heterogeneous material characterization is the ability to (hopefully quantitatively) distinguish components from mixtures of fluids. These experiments were performed using decane and water as a test case, using the coaxial-insert physical separation method described in Sec. 6.5.2.2.

The T_1 experiments were performed using the simple shuttle-delay-acquire pulse sequence described in Sec. 6.2.1. As with all the other experiments described in this section, the data were acquired in an indirect dimension, field-cycled to 0.5 G to approximate the Earth's ambient magnetic field.

The T_2 experiments were performed using the CPMG sequence described in Sec. 6.3.2, with the time between pulse and echo (τ) set to 5 ms, sampled in an indirect dimension at echo numbers n where n was constrained to be an even number to minimize any potential convection or motion effects.

The diffusion experiments were performed using the pulse sequence described in Sec. 6.4.1, with the number of echoes before acquisition, N, constrained to be 6 and τ is set at 75 ms and the gradient strength was varied between



(a) (b)

Figure 6.6: T_1 (6.6a) and T_2 (6.6b) relaxation measurement data.

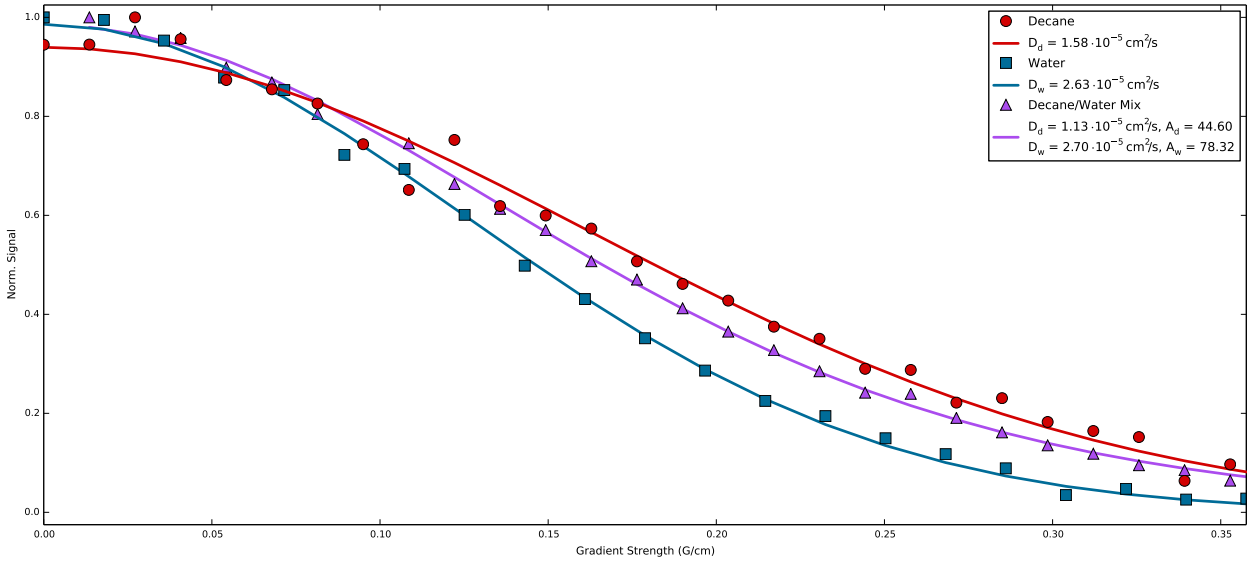


Figure 6.7: Diffusion measurements of water, decane and a decane-water mixture.

The data from each experiment were fit to a least squares curve fit to the appropriate kernel function — the results are shown in Figs 6.6 and 6.7. The relaxation times and diffusion coefficients found in the mixtures can be directly compared to those in the pure solvents to show that this technique is sufficient for separating out the components of hydrocarbon/water mixtures in a qualitative manner. From the amplitudes of the respective signals, we can also extract quantitative information, though it is somewhat less straightforward.

To signal generated from the sample will depend on the prepolarization and shuttling step, which acts as a T_1 filter, and differentially affects the different components. For the relaxation experiments, the fraction of signal from a given component, $f_{c,R}$ is calculated from

Eqn. 6.33

$$f_{c,R} = \frac{\rho_{H,c} \cdot V_c \cdot e^{-t_s/T_{1,c}}}{\sum_i \rho_{H,c_i} \cdot V_{c_i} \cdot e^{-t_s/T_{1,c_i}}}, \tag{6.33}$$

where V_c is the volume of a given component in the set of all components c_i and $\rho_{H,C}$ is the hydrogen density. In diffusion experiments, the signal fraction ($f_{D,c}$) also depends on the T_2 relaxation occurring during the pulse sequence itself:

$$f_{c,D} = \frac{\rho_{H,c} \cdot V_c \cdot e^{-t_s/T_{1,c}} e^{-2n\tau/T_{2,c}}}{\sum_i \rho_{H,c_i} \cdot V_{c_i} \cdot e^{-t_s/T_{1,c_i}} e^{-2n\tau/T_{2,c_i}}}. \tag{6.34}$$

These calculations are complicated somewhat by the fact that the T_1 relaxation during the shuttling time does not necessarily take place in the fixed 0.5 G field induced by the z coil, but rather passes at an unknown speed through a somewhat more complicated field environment generated by, at various times, the prepolarizing magnet, the leading coil, and the B_z coil, making calibration somewhat difficult. At 0.5 G, the T_1 of water does not seem particularly sensitive to field (see Sec. 6.6 for more details), and so this does not seem to be a major source of uncertainty.

Applying Eqns. 6.31 and 6.32, we calculate the sample volumes to be $V_d = 53\pm1\,\mu L$ and $V_w = 59\pm1\,\mu L$. The hydrogen densities for water and decane are $\rho_{H,d} = 112.95$ M and $\rho_{H,w} = 110$ M. Using the measured values of $T_{1,d}$=1.62 s and $T_{1,w} = 3.89$ s, the expected decane sample fraction in the relaxation experiments is $f_{d,R} = 0.42\pm0.01$; the measured values for the curves shown in Fig. 6.6 are $f_{d,T_1} = 0.48\pm0.04$ and $f_{d,T_2} = 0.45\pm0.04$. Using the measured values of $T_{2,d} = 1.63$ s and $T_{2,w} = 2.96$, the decane signal fraction in the diffusion experiment is calculated to be $f_{d,D} = 0.36\pm0.01$, which is a perfect match for the measured value of 0.36.

For samples with a known number of components, the least squares curve fitting method is an appropriate method of data analysis, but in general, these techniques are applied to unknown heterogeneous material samples, which may not have a known (or even well-defined) number of components. For these more complicated samples, the pseudo-spectrum $F(x)$ in the domain of the parameter of interest can be generated from the magnetization signals $M(\tau)$ by solving the relevant Fredholm integral of the first kind for the appropriate kernel function $k(\tau, x)$:[75,76]

$$M(\tau) = \int k(\tau, x) F(x) \mathrm{d}x. \tag{6.35}$$

This also extends to two-dimensional correlation spectra:

$$M(\tau_1, \tau_2) = \int\int k_1(\tau_1, x) k_2(\tau_2, y) F(x, y) \mathrm{d}x\mathrm{d}y. \tag{6.36}$$

These are well-known to be ill-posed problems[77], giving results which are sensitive to error in the data, and so in our work, the inversion was performed using a MATLAB toolkit

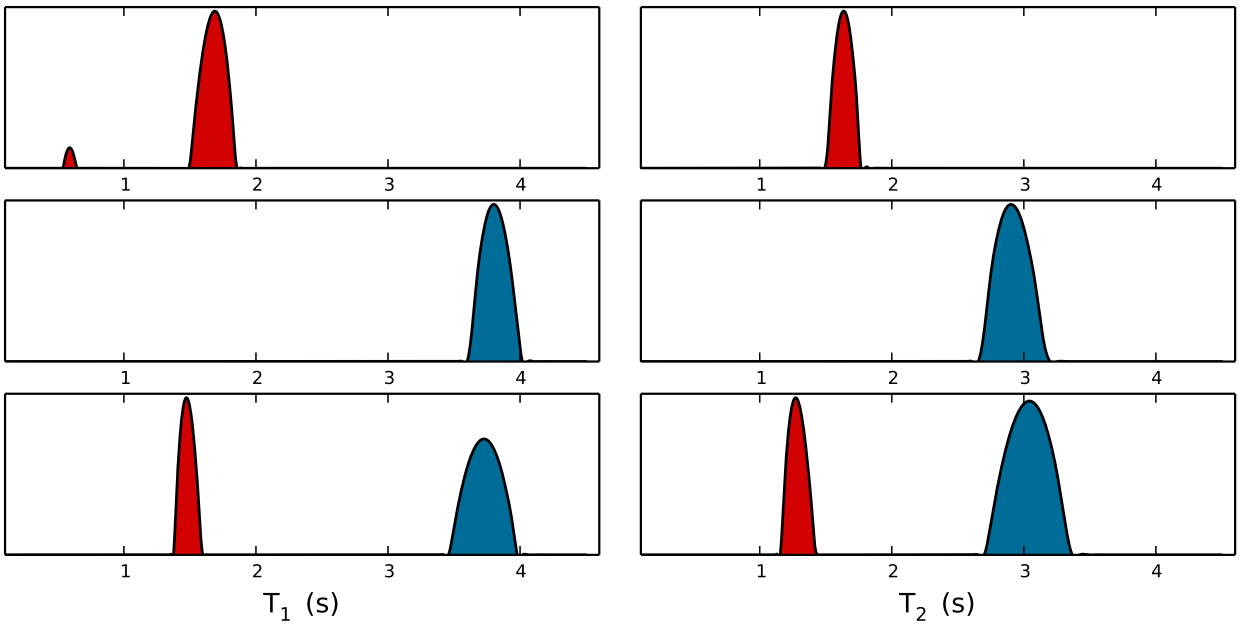


Figure 6.8: The 1-dimensional T_1 and T_2 pseudospectra of pure decane (top), pure water (center) and a decane/water mixture (bottom).

with SVD (singular value decomposition) truncation and Tikhonov regularization based on the procedures detailed in *Venkataramanan et al., 2002* [76] and *Mitchell et al., 2012* [78]. The relaxation experiments were inverted using a Laplacian kernel (Eqn. 6.37), and the diffusion experiments were inverted using a Gaussian kernel (Eqn. 6.38).

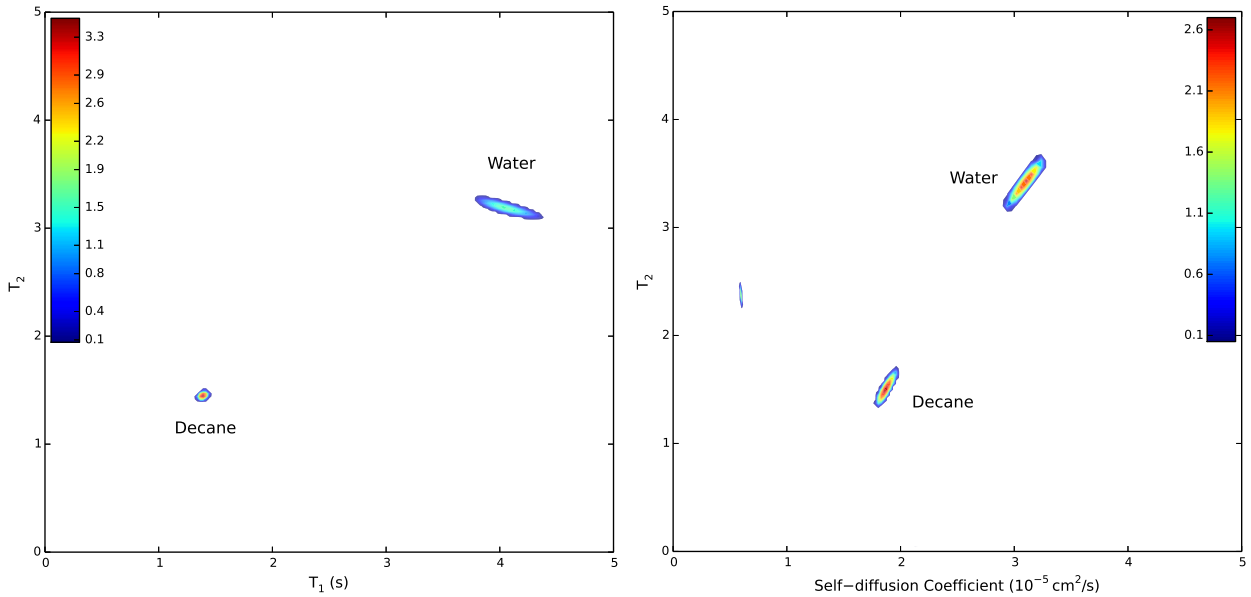
$$k(\tau, T) = e^{-\tau/T} \tag{6.37}$$

$$k(G, D) = e^{-G^2 \cdot D} \tag{6.38}$$

As can be seen in Fig. 6.8, Laplace inversions of the data presented in Fig. 6.6 correctly pick out the number of components[6] without any foreknowledge, and there is a clear separation between the two components. [79]

The same pulse sequences can also be used in two-dimensional correlation measurements to extract further information and help resolve close or broad components. Again using a decane/water mixture as the test sample, we performed T_1-T_2 and T_2-diffusion correlation experiments, the resulting 2D pseudospectra are shown in Fig. 6.9. As these data were acquired over a long time period (12-36 h), drift was a significant concern, and so experiments were acquired without active cooling to avoid potential temperature instabilities due to changes in the N_2 back pressure over the course of the day, and as such the results are a somewhat elevated temperature. For consistency, the data in Figs. 6.6, 6.7 and 6.8 were

[6]The decane T_1spectrum shows a small artifact peak, but this is likely due to an uncompensated offset in the instrument and is not a significant source of errors.



(a) (b)

Figure 6.9: T_1-T_2 (6.9a) and T_2-diffusion (6.9b) Laplace pseudospectra of decane and water. The extra peak in the T_2-diffusion spectrum is an artifact of the inversion procedure used. The colors in the figure correspond to relative intensity, normalized to 100.

similarly performed without N_2 back pressure. Due to uncorrected drift over the course of the experiments, minor fast-relaxing artifacts appear in the T_2-diffusion Laplace psuedospectrum; these artifacts do not significantly affect the results and are likely unique to this experimental configuration.

6.6 Low Field Relaxation Dynamics

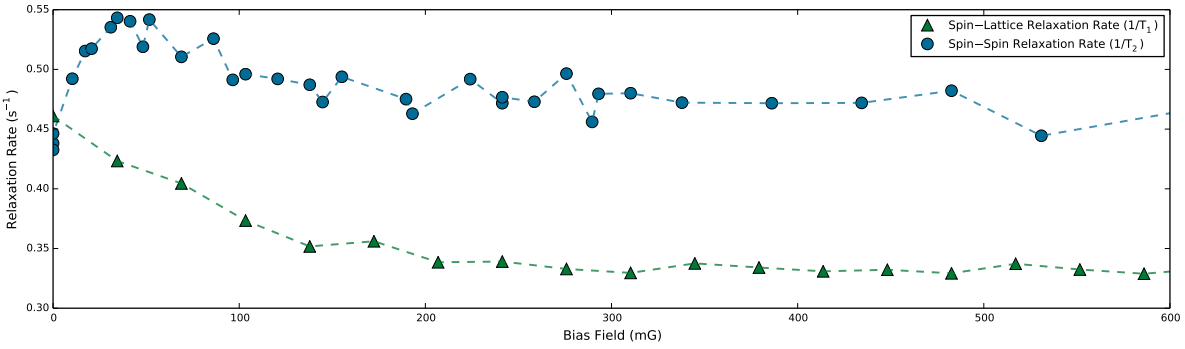


Figure 6.10: Measured relaxation rates ($1/T_1$ and $1/T_2$) as a function of the bias offset field at 36 °C.

One significant advantage of performing our experiments in a field-cycled indirect dimen-

sion (as detailed in Sec. 5.5) is that the detection field can easily be changed without any recalibration. This allows us to use the field in which the experiment is performed as an additional dimension in the experiment. As a test of these capabilities, we measured the T_1 and T_2 of water as a function of field at very low fields (<1 G). We also made a cursory study of the effect of small magnetic fields on T_2 relaxation of various hydrocarbon solvents.

The values measured here, having been acquired at an elevated temperature (36 °C), are not directly comparable to the literature values, which were measured at room temperature, but the features of the data set are largely in agreement. At zero field there is no symmetry-breaking bias field, and as such there should be no distinction between T_1 and T_2, which is confirmed by the data presented in Fig. 6.10. Below 100 mG, we also observe the anomalously high relaxation rates first presented in *Hartwig et al., 2011* [80].

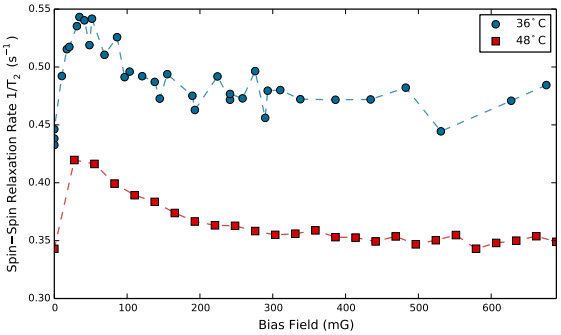


Figure 6.11: The measured spin-spin relaxation rate ($1/T_2$) as a function of field for two different temperatures, 36 °C and 48 °C

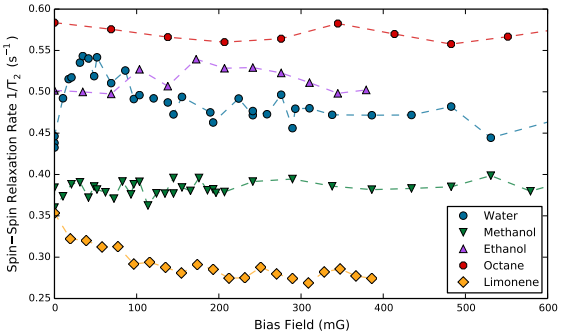


Figure 6.12: The measured spin-spin relaxation rate ($1/T_2$) as a function of field for several hydrocarbon-based solvents at 36 °C

As a general test for possible compounding of the effect of temperature and bias offset field on relaxation rates in water, we repeated the T_2-field measurement without any dry N_2 flow cooling the sample, which raised the temperature to 47 °C. The data, shown in Fig. 6.11, scale roughly uniformly with the change in temperature, which is preliminary evidence against any compound temperature-field effects in low field water relaxation dynamics.

Finally, we performed a small survey of the effect of small magnetic fields on various hydrocarbon solvents, the results of which are presented in 6.12. There was not enough time to fully explore this space, and so only T_2 data were acquired for a small number of solvents. A further exploration of this space could potentially be useful in identifying an ideal field in which to perform field-cycled NMR relaxometry in order to maximize contrast between sample components. In our experiments, T_2 was measured to test whether anomalous low-field relaxation would be observed in other systems, particularly methanol and ethanol, but the T_1 dynamics at low field would likely be a more generally useful data set; this is because T_1 is generally measured in an indirect dimension anyway, using either inversion- or saturation-recovery pulse sequences, and it is easier to field cycle samples in an indirect dimension than to change the field in a direct dimension.

Bibliography

[1] I. M. Savukov, S. J. Seltzer, and M. V. Romalis, "Detection of NMR signals with a radio-frequency atomic magnetometer.," *Journal of Magnetic Resonance*, vol. 185, pp. 214–20, Apr. 2007, doi:10.1016/j.jmr.2006.12.012. 2, 81

[2] M. P. Ledbetter, I. M. Savukov, D. Budker, V. Shah, S. Knappe, J. Kitching, D. J. Michalak, S. Xu, and A. Pines, "Zero-field remote detection of NMR with a microfabricated atomic magnetometer.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, pp. 2286–90, Feb. 2008, doi:10.1073/pnas.0711505105. 2, 82

[3] P. D. D. Schwindt, S. Knappe, V. Shah, L. Hollberg, J. Kitching, L.-A. Liew, and J. Moreland, "Chip-scale atomic magnetometer," *Applied Physics Letters*, vol. 85, no. 26, pp. 6409–6411, 2004, doi:10.1063/1.1839274. 2, 82

[4] H. G. Dehmelt, "Slow spin relaxation of optically polarized sodium atoms," *Physical Review*, vol. 105, pp. 1487–1489, Mar. 1957, doi:10.1103/PhysRev.105.1487. 2

[5] W. E. Bell and A. L. Bloom, "Optical detection of magnetic resonance in alkali metal vapor," *Physical Review*, vol. 107, pp. 1559–1565, Sept. 1957, doi:10.1103/PhysRev.107.1559.

[6] W. E. Bell and A. L. Bloom, "Optically driven spin precession," *Physical Review Letters*, vol. 6, pp. 280–281, Mar. 1961, doi:10.1103/PhysRevLett.6.280. 2

[7] W. Happer and W. a. Wijngaarden, "An optical pumping primer," *Hyperfine Interactions*, vol. 38, pp. 435–470, Dec. 1987, doi:10.1007/BF02394855. 2

[8] W. Happer, "Optical pumping," *Reviews of Modern Physics*, vol. 44, pp. 169–249, Apr. 1972, doi:10.1103/RevModPhys.44.169. 2

[9] A. L. Bloom, "Principles of operation of the rubidium vapor magnetometer," *Applied Optics*, vol. 1, p. 61, Jan. 1962, doi:10.1364/AO.1.000061. 2

[10] S. J. Seltzer, *Developments in Alkali-Metal Atomic Magnetometry*. PhD thesis, Princeton University, 2008, ISBN: 9780549933557. 3, 4, 5

[11] W. Franzen and A. G. Emslie, "Atomic orientation by optical pumping," *Physical Review*, vol. 108, pp. 1453–1458, Dec. 1957, doi:10.1103/PhysRev.108.1453. 3

[12] V. Shah, S. Knappe, P. D. D. Schwindt, and J. Kitching, "Subpicotesla atomic magnetometry with a microfabricated vapour cell," *Nature Photonics*, vol. 1, pp. 649–652, Nov. 2007, doi:10.1038/nphoton.2007.201. 3

[13] P. D. D. Schwindt, B. Lindseth, S. Knappe, V. Shah, J. Kitching, and L.-A. Liew, "Chip-

scale atomic magnetometer with improved sensitivity by use of the M[sub x] technique," *Applied Physics Letters*, vol. 90, no. 8, p. 081102, 2007, doi:10.1063/1.2709532. 3, 33

[14] Z. Wu, M. Kitano, W. Happer, M. Hou, and J. Daniels, "Optical determination of alkali metal vapor number density using Faraday rotation," *Applied Optics*, vol. 25, p. 4483, Dec. 1986, doi:10.1364/AO.25.004483. 8

[15] J. C. Allred, R. N. Lyman, T. W. Kornack, and M. V. Romalis, "High-sensitivity atomic magnetometer unaffected by spin-exchange relaxation," *Physical Review Letters*, vol. 89, p. 130801, Sept. 2002, doi:10.1103/PhysRevLett.89.130801. 11, 32

[16] D. Dubbers, "Simple formula for multiple mu-metal shields," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 243, pp. 511–517, Mar. 1986, doi:10.1016/0168-9002(86)90989-7. 11

[17] D. M. Ginsberg and M. J. Melchner, "Optimum geometry of saddle shaped coils for generating a uniform magnetic field," *Review of Scientific Instruments*, vol. 41, no. 1, pp. 122–123, 1970, doi:10.1063/1.1684235. 20

[18] S. J. Seltzer and M. V. Romalis, "Unshielded three-axis vector operation of a spin-exchange-relaxation-free atomic magnetometer," *Applied Physics Letters*, vol. 85, no. 20, p. 4804, 2004, doi:10.1063/1.1814434. 30, 82

[19] M. Ledbetter, T. Theis, J. Blanchard, H. Ring, P. Ganssle, S. Appelt, B. Bluemich, A. Pines, and D. Budker, "Near-zero-field nuclear magnetic resonance," *Physical Review Letters*, vol. 107, no. 10, p. 5, 2011. 31, 82

[20] C. B. Alcock, V. P. Itkin, and M. K. Horrigan, "Vapour pressure equations for the metallic elements: 2982500K," *Canadian Metallurgical Quarterly*, vol. 23, pp. 309–313, July 1984, doi:10.1179/000844384795483058. 32

[21] I. Savukov and S. J. Seltzer, *Spin-exchange-relaxation-free (SERF) magnetometers*, ch. 5, pp. 85–103. Cambridge University Press, 1 ed., 2013, ISBN: 978-1107010352. 32

[22] C. T. Ewing, D. Chang, J. P. Stone, J. R. Spann, and R. R. Miller, "New method of correlating the vapor pressures of alkali metals over the temperature range 400 to 2500 F," *Journal of Chemical & Engineering Data*, vol. 14, pp. 210–214, Apr. 1969, doi:10.1021/je60041a009. 32

[23] S. Carusotto, E. Iacopini, E. Polacco, F. Scuri, G. Stefanini, and E. Zavattini, "Measurement of the magnetic birefringence of noble gases," *Journal of the Optical Society of America B*, vol. 1, pp. 635–640, Aug. 1984, doi:10.1364/JOSAB.1.000635. 33

[24] R. Mhaskar, S. Knappe, and J. Kitching, "A low-power, high-sensitivity micromachined optical magnetometer," *Applied Physics Letters*, vol. 101, no. 24, p. 241105, 2012, doi:10.1063/1.4770361. 33

[25] L. C. Burmeister, *Convective Heat Transfer*. New York: Wiley, 2nd ed., 1993, ISBN: 9780471577096. 37

[26] J. Keeler, "Coherence selection: phase cycling and field gradient pulses," in *Understanding NMR Spectroscopy*, ch. 11, Hoboken: Wiley, 2nd ed., 2013. 48

[27] A. Kumar, D. Welti, and R. R. Ernst, "NMR Fourier zeugmatography," *Journal of Magnetic Resonance*, vol. 18, pp. 69–83, Apr. 1975, doi:10.1016/0022-2364(75)90224-3.

81

[28] A. Kumar, D. Welti, and R. R. Ernst, "Imaging of macroscopic objects by NMR Fourier zeugmatography," *Die Naturwissenschaften*, vol. 62, pp. 34–34, Jan. 1975, doi:10.1007/BF00594040.

[29] M. Mösle, S.-I. Han, W. R. Myers, S.-K. Lee, N. Kelso, M. Hatridge, A. Pines, and J. Clarke, "SQUID-detected microtesla MRI in the presence of metal," *Journal of Magnetic Resonance*, vol. 179, no. 1, pp. 146 – 151, 2006, doi:10.1016/j.jmr.2005.11.005. 81, 82

[30] M. H. Levitt, *Spin Dynamics*. New York: Wiley, 2nd ed., 2008, ISBN: 978-0470511176. 81, 110

[31] F. Bloch, "Nuclear induction," *Physica*, vol. 17, no. 34, pp. 272 – 281, 1951, doi:10.1016/0031-8914(51)90068-7. 81

[32] E. L. Hahn, "Nuclear induction due to free Larmor precession," *Physical Review*, vol. 77, pp. 297–298, Jan. 1950, doi:10.1103/PhysRev.77.297.2. 81

[33] A. L. Bloom, "Nuclear induction in inhomogeneous fields," *Physical Review*, vol. 98, pp. 1105–1111, May 1955, doi:10.1103/PhysRev.98.1105. 81

[34] M. D. Hürlimann, "Diffusion and relaxation effects in general stray field NMR experiments.," *Journal of Magnetic Resonance*, vol. 148, pp. 367–78, Feb. 2001, doi:10.1006/jmre.2000.2263. 81, 109

[35] G. Moresi and R. Magin, "Miniature permanent magnet for table-top NMR," *Concepts in Magnetic Resonance*, vol. 19B, pp. 35–43, Oct. 2003, doi:10.1002/cmr.b.10082. 81

[36] R. L. Kleinberg and J. A. Jackson, "An introduction to the history of NMR well logging," *Concepts in Magnetic Resonance*, vol. 13, no. 6, pp. 340–342, 2001, doi:10.1002/cmr.1018. 81, 109

[37] M. D. Hürlimann, "Well Logging," in *Encyclopedia of Magnetic Resonance*, John Wiley and Sons, Ltd, 2012, doi:10.1002/9780470034590.emrstm0593.pub2. 81, 109

[38] F. Casanova, J. Perlo, and B. Blümich, eds., *Single-Sided NMR*. Springer, 2011, ISBN: 978-3642163067. 81, 109

[39] B. Blümich, P. Blümler, G. Eidmann, A. Guthausen, R. Haken, U. Schmitz, K. Saito, and G. Zimmer, "The NMR-mouse: construction, excitation, and applications," *Magnetic Resonance Imaging*, vol. 16, pp. 479–484, June 1998, doi:10.1016/S0730-725X(98)00069-1. 109

[40] A. Guthausen, G. Guthausen, A. Kamlowski, H. Todt, W. Burk, and D. Schmalbein, "Measurement of fat content of food with single-sided NMR," *Journal of the American Oil Chemists' Society*, vol. 81, pp. 727–731, Aug. 2004, doi:10.1007/s11746-004-0969-5. 81, 109, 119

[41] T. G. Walker and W. Happer, "Spin-exchange optical pumping of noble-gas nuclei," *Reviews of Modern Physics*, vol. 69, pp. 629–642, Apr. 1997, doi:10.1103/RevModPhys.69.629. 81

[42] B. M. Goodson, "Nuclear magnetic resonance of laser-polarized noble gases in molecules, materials, and organisms.," *Journal of Magnetic Resonance*, vol. 155, pp. 157–216, Apr. 2002, doi:10.1006/jmre.2001.2341.

[43] L. Schröder, T. Meldrum, M. Smith, T. J. Lowery, D. E. Wemmer, and A. Pines, "Temperature response of ^{129}Xe depolarization transfer and its application for ultrasensitive NMR detection," *Physical Review Letters*, vol. 100, p. 257603, June 2008, doi:10.1103/PhysRevLett.100.257603. 81

[44] T. Theis, P. Ganssle, G. Kervern, S. Knappe, J. Kitching, M. P. Ledbetter, D. Budker, and A. Pines, "Parahydrogen-enhanced zero-field nuclear magnetic resonance," *Nature Physics*, vol. 7, pp. 571–575, May 2011, doi:10.1038/nphys1986. 81

[45] L. S. Lloyd, R. W. Adams, M. Bernstein, S. Coombes, S. B. Duckett, G. G. R. Green, R. J. Lewis, R. E. Mewis, and C. J. Sleigh, "Utilization of SABRE-derived hyperpolarization to detect low-concentration analytes via 1D and 2D NMR methods.," *Journal of the American Chemical Society*, vol. 134, pp. 12904–7, Aug. 2012, doi:10.1021/ja3051052. 81

[46] T. Maly, G. T. Debelouchina, V. S. Bajaj, K.-N. Hu, C.-G. Joo, M. L. Mak-Jurkauskas, J. R. Sirigiri, P. C. A. van der Wel, J. Herzfeld, R. J. Temkin, and R. G. Griffin, "Dynamic nuclear polarization at high magnetic fields," *The Journal of Chemical Physics*, vol. 128, p. 052211, Feb. 2008, doi:10.1063/1.2833582. 81

[47] P. J. M. van Bentum, G. H. A. van der Heijden, J. A. Villanueva-Garibay, and A. P. M. Kentgens, "Quantitative analysis of high field liquid state Dynamic Nuclear Polarization.," *Physical Chemistry Chemical Physics*, vol. 13, pp. 17831–40, Oct. 2011, doi:10.1039/c1cp22002k. 81, 107, 109, 111

[48] R. A. Webb, "New technique for improved low-temperature SQUID NMR measurements," *Review of Scientific Instruments*, vol. 48, no. 12, p. 1585, 1977, doi:10.1063/1.1134950. 82

[49] D. J. Meredith, G. R. Pickett, and O. G. Symko, "Application of a SQUID magnetometer to NMR at low temperatures," *Journal of Low Temperature Physics*, vol. 13, pp. 607–615, Dec. 1973, doi:10.1007/BF00656548. 82

[50] S. Appelt, H. Kühn, F. W. Häsing, and B. Blümich, "Chemical analysis by ultrahigh-resolution nuclear magnetic resonance in the Earth's magnetic field," *Nature Physics*, vol. 2, pp. 105–109, Jan. 2006, doi:10.1038/nphys211. 82, 83

[51] D. Zax, A. Bielecki, K. Zilm, and A. Pines, "Heteronuclear zero-field NMR," *Chemical Physics Letters*, vol. 106, pp. 550–553, May 1984, doi:10.1016/0009-2614(84)85381-6.

[52] R. McDermott, A. H. Trabesinger, M. Muck, E. L. Hahn, A. Pines, and J. Clarke, "Liquid-state NMR and scalar couplings in microtesla magnetic fields.," *Science*, vol. 295, pp. 2247–9, Mar. 2002, doi:10.1126/science.1069280. 83

[53] M. P. Ledbetter, C. W. Crawford, A. Pines, D. E. Wemmer, S. Knappe, J. Kitching, and D. Budker, "Optical detection of NMR J-spectra at zero magnetic field.," *Journal of Magnetic Resonance*, vol. 199, pp. 25–9, July 2009, doi:10.1016/j.jmr.2009.03.008. 83

[54] T. Theis, M. P. Ledbetter, G. Kervern, J. W. Blanchard, P. J. Ganssle, M. C. Butler, H. D. Shin, D. Budker, and A. Pines, "Zero-field NMR enhanced by parahydrogen in reversible exchange.," *Journal of the American Chemical Society*, vol. 134, pp. 3987–90, Mar. 2012, doi:10.1021/ja2112405.

[55] M. C. Butler, M. P. Ledbetter, T. Theis, J. W. Blanchard, D. Budker, and A. Pines,

"Multiplets at zero magnetic field: The geometry of zero-field NMR," *The Journal of Chemical Physics*, vol. 138, pp. 184202–15, 05/14/ 2013, doi:10.1063/1.4803144.

[56] T. Theis, J. W. Blanchard, M. C. Butler, M. P. Ledbetter, D. Budker, and A. Pines, "Chemical analysis using J-coupling multiplets in zero-field NMR," *Chemical Physics Letters*, 2013, doi:10.1016/j.cplett.2013.06.042. 83

[57] M. C. Butler, G. Kervern, T. Theis, M. P. Ledbetter, P. J. Ganssle, J. W. Blanchard, D. Budker, and A. Pines, "Parahydrogen-induced polarization at zero magnetic field," *The Journal of Chemical Physics*, vol. 138, p. 234201, 06/2013 2013, doi:doi:http://dx.doi.org/10.1063/1.4805062. 83

[58] J. Simpson and H. Carr, "Diffusion and nuclear spin relaxation in water," *Physical Review*, vol. 630, no. 1954, 1958. 107, 109, 111

[59] J. Hindman, A. Svirmickas, and M. Wood, "Relaxation processes in water. A study of the proton spin lattice relaxation time," *The Journal of Chemical Physics*, vol. 59, no. 3, pp. 1517–1522, 1973, doi:10.1063/1.1680209. 107, 110, 111

[60] Y.-Q. Song, *Novel Two Dimensional NMR of Diffusion and Relaxation for Material Characterization*, ch. 2.7.1, pp. 163–182. Wiley-VCH, 1 ed., 2006, ISBN: 978-3527312344. 109, 121

[61] J. G. Seland, M. Bruvold, H. Anthonsen, H. Brurok, W. Nordhø y, P. Jynge, and J. Krane, "Determination of water compartments in rat myocardium using combined D-T_1 and T_1-T_2 experiments," *Magnetic Resonance Imaging*, vol. 23, pp. 353–4, Feb. 2005, doi:10.1016/j.mri.2004.11.062. 109

[62] M. Köpf, C. Corinth, O. Haferkamp, and T. F. Nonnenmacher, "Anomalous diffusion of water in biological tissues.," *Biophysical journal*, vol. 70, pp. 2950–8, June 1996, doi:10.1016/S0006-3495(96)79865-X. 109, 119

[63] M. D. Hürlimann, L. Burcaw, and Y.-Q. Song, "Quantitative characterization of food products by two-dimensional D-T_2 and T_1-T_2 distribution functions in a static gradient.," *Journal of Colloid and Interface Science*, vol. 297, pp. 303–11, May 2006, doi:10.1016/j.jcis.2005.10.047. 109

[64] M. D. Hürlimann and D. D. Griffin, "Spin dynamics of Carr-Purcell-Meiboom-Gill-like sequences in grossly inhomogeneous B_0 and B_1 fields and application to NMR well logging," *Journal of Magnetic Resonance*, vol. 143, pp. 120–35, Mar. 2000, doi:10.1006/jmre.1999.1967. 109, 121

[65] J. A. Jackson, L. J. Burnett, and J. Harmon, "Remote (inside-out) NMR. III. Detection of nuclear magnetic resonance in a remotely produced region of homogeneous magnetic field," *Journal of Magnetic Resonance*, vol. 41, pp. 411–421, Dec. 1980, doi:10.1016/0022-2364(80)90298-X. 109

[66] T. DeFries and J. Jonas, "Pressure dependence of NMR proton spinlattice relaxation times and shear viscosity in liquid water in the temperature range -15–10 °C," *The Journal of Chemical Physics*, vol. 66, no. 3, p. 896, 1977, doi:10.1063/1.433995. 109

[67] E. Rumeur, J. de Certaines, P. Toulouse, and P. Rochcongar, "Water phases in rat striated muscles as determined by T_2 proton NMR relaxation times," *Magnetic Resonance Imaging*, vol. 5, pp. 267–272, Jan. 1987, doi:10.1016/0730-725X(87)90003-8. 109

[68] H. A. Resing, "Apparent phasetransition effect in the NMR spinspin relaxation time caused by a distribution of correlation times," *The Journal of Chemical Physics*, vol. 43, no. 2, p. 669, 1965, doi:10.1063/1.1696791. 109

[69] L. van der Weerd, S. M. Melnikov, F. J. Vergeldt, E. G. Novikov, and H. Van As, "Modelling of self-diffusion and relaxation time NMR in multicompartment systems with cylindrical geometry," *Journal of Magnetic Resonance*, vol. 156, pp. 213–221, June 2002, doi:10.1006/jmre.2002.2550. 109

[70] E. L. Hahn, "Spin echoes," *Physical Review*, vol. 80, pp. 580–594, Nov. 1950, doi:10.1103/PhysRev.80.580. 114

[71] H. Carr and E. Purcell, "Effects of diffusion on free precession in nuclear magnetic resonance experiments," *Physical Review*, vol. 94, pp. 630–638, May 1954, doi:10.1103/PhysRev.94.630. 115, 116

[72] S. Meiboom and D. Gill, "Modified spin-echo method for measuring nuclear relaxation times," *Review of Scientific Instruments*, vol. 29, no. 8, p. 688, 1958, doi:10.1063/1.1716296. 115

[73] E. Fukushima and S. B. W. Roeder, pp. 197–201. 1993, ISBN: 0201627264. 120

[74] J. H. Wang, "Self-diffusion coefficients of water," *The Journal of Physical Chemistry*, vol. 69, pp. 4412–4412, Dec. 1965, doi:10.1021/j100782a510. 121

[75] Y.-Q. Song, L. Venkataramanan, M. D. Hürlimann, M. Flaum, P. Frulla, and C. Straley, "T_1-T_2 correlation spectra obtained using a fast two-dimensional Laplace inversion.," *Journal of Magnetic Resonance*, vol. 154, pp. 261–8, Feb. 2002, doi:10.1006/jmre.2001.2474. 126

[76] L. Venkataramanan and M. Hurlimann, "Solving Fredholm integrals of the first kind with tensor product structure in 2 and 2.5 dimensions," *IEEE Transactions on Signal Processing*, vol. 50, pp. 1017–1026, May 2002, doi:10.1109/78.995059. 126, 127

[77] J. P. Butler, J. A. Reeds, and S. V. Dawson, "Estimating solutions of first kind integral equations with nonnegative constraints and optimal smoothing," *SIAM Journal on Numerical Analysis*, vol. 18, pp. 381–397, June 1981, doi:10.1137/0718025. 126

[78] J. Mitchell, T. C. Chandrasekera, and L. F. Gladden, "Numerical estimation of relaxation and diffusion distributions in two dimensions.," *Progress in Nuclear Magnetic Resonance Spectroscopy*, vol. 62, pp. 34–50, Apr. 2012, doi:10.1016/j.pnmrs.2011.07.002. 127

[79] Y.-Q. Song, "Resolution and uncertainty of Laplace inversion spectrum," *Magnetic Resonance Imaging*, vol. 25, pp. 445–8, May 2007, doi:10.1016/j.mri.2006.11.023. 127

[80] S. Hartwig, J. Voigt, H.-J. Scheer, H.-H. Albrecht, M. Burghoff, and L. Trahms, "Nuclear magnetic relaxation in water revisited," *The Journal of Chemical Physics*, vol. 135, no. 5, p. 054201, 2011, doi:10.1063/1.3623024. 129

Appendix A

Terminology

In a developing and cross-disciplinary field such as NMR and magnetometry, there are bound to be differences in terminology. In this section, these possible ambiguities are elucidated by providing standardized definitions where they differ from the literature and citations, where the literature is unclear, or where multiple standards are used.

A.1 Choice of Coordinate System

In the field of atomic magnetometry, particularly in the physics literature, the common practice is to call the direction of the probe beam $\vec{x}$, the direction of the measured field $\vec{y}$ and the direction of the pump beam $\vec{z}$. While this is a useful framework when the subjects under study are the rubidium spins, which are polarized along the pump beam direction, it is inconsistent with the NMR convention of defining the $\vec{z}$ axis in terms of the direction along which the spins have their initial polarization (which generally corresponds to the "lab frame" $\vec{z}$ axis - that is, normal to the floor). As the magnetometer described herein is used primarily for NMR studies, a sample-centered approach is used, defining the probe beam axis as $\vec{x}$, the pump beam axis as $\vec{y}$ and the sensitive direction (the direction along which the samples are polarized) as $\vec{z}$.

A.2 Magnetometer Components

Many components in this setup are named by analogy to a high-field instrument, or a name has been created *de novo* for its use in the magnetometer.

Appendix B

Proofs

As it is sometimes useful to see the proofs behind equations, but rarely sufficiently compelling that it justifies a treatment in the main text, the proofs of many analyses have been relegated to this appendix.

B.1 Circuit Analyses

B.1.0.3 Pulse Angle Error from Rise Times

The tip angle (θ_p) from an induced pulse is given by:

$$\theta_p = -\gamma \int B_p(t)\mathrm{d}t \tag{B.1}$$

Where γ is the gyromagnetic ratio of the spin and $B_p(t)$ is the magnetic field used for the pulse along the pulse direction in the spin's frame of reference (generally this is given in the rotating frame, but at zero field the rotating frame coincides with the lab frame). For a DC pulse with a characteristic time constant τ, applying a pulse of duration t_p, the tip angle as a function of time is thus:

$$\theta(t) = \begin{cases} -\gamma B_0 \int_0^t \left(1 - e^{-\hat{t}/\tau}\right) \mathrm{d}\hat{t} & 0 \leq t \leq \hat{t_p}. \\ -\gamma B_0 \int_0^{t_p} \left(1 - e^{-\hat{t}/\tau}\right) \mathrm{d}\hat{t} + \gamma B_0 \left(1 - e^{-t_p/\tau}\right) \int_{t_p}^t e^{-\hat{t}/\tau} \mathrm{d}\hat{t} & t > t_p \end{cases} \tag{B.2}$$

Solving the integral for the first case (during the rise time):

$$\begin{aligned} \int_0^t \left(1 - e^{-\hat{t}/\tau}\right) \mathrm{d}\hat{t} &= \left[\hat{t} - \tau e^{-\hat{t}/\tau}\right]_0^t && \text{(B.3)} \\ &= \left(t - \tau e^{-t/\tau}\right) - \left(0 - e^0\right) && \text{(B.4)} \\ &= t - \tau \left(1 - e^{-t/\tau}\right) && \text{(B.5)} \end{aligned}$$

And for the decay time:

$$\left(1-e^{-t_p/\tau}\right)\int_{t_p}^{t} e^{-\left(\hat{t}\,-\,t_p\right)/\tau}\mathrm{d}\hat{t} = \left(1-e^{-t_p/\tau}\right)\left[\hat{t}-\tau e^{-\hat{t}/\tau}\right]_{t_p}^{t} \tag{B.6}$$

$$= \tau\left(1-e^{-t_p/\tau}\right)\left[1-e^{-\left(t\,-\,t_p\right)/\tau}\right] \tag{B.7}$$

So combining these two (assume $t > t_p$):

$$\int_0^t B(\hat{t})\mathrm{d}\hat{t} = t_p + \tau e^{-\left(t\,-\,t_p\right)/\tau}\left(1-e^{-t_p/\tau}\right) \tag{B.8}$$

And for $t \gg \tau$, $e^{-\left(t\,-\,tp\right)/\tau} \approx 0$ and the second term drops out, indicating that for sufficiently long inter-pulse spacing relative to τ, deviations from perfect square pulses have no effect on the tip angle.

B.1.0.4 Calculation of V_O

For convenience, define the effective resistance of the coil output stage (after the noise-limiting diodes) R_M as:

$$R_M = \left(\frac{1}{R_1}+\frac{1}{R_C}\right)^{-1} + R_2 \tag{B.9}$$

$$R_M = \frac{R_1 R_C}{R_1+R_C} + R_S \tag{B.10}$$

$$\tag{B.11}$$

Then, as nearly all circuit proofs do, start from Kirchoff's rules and Ohm's law:

$$I_S = I_{SH} + I_M \tag{B.12}$$

$$\frac{V_P}{R_S} = \frac{V_o - V_P}{R_{SH}} + \frac{V_C - V_P}{R_M} \tag{B.13}$$

$$\frac{V_P}{R_S} = V_o\left(\frac{R_{SH}+R_M}{R_{SH}R_M}\right) - V_P\left(\frac{R_{SH}+R_M}{R_{SH}R_M}\right) - \frac{\Delta V_D}{R_M} \tag{B.14}$$

$$V_o\left(\frac{R_{SH}+R_M}{R_{SH}R_M}\right) = V_P\left(\frac{R_{SH}+R_M}{R_{SH}R_M}+\frac{1}{R_S}\right) + \frac{\Delta V_D}{R_M} \tag{B.15}$$

$$V_o = V_P\left[1+\frac{R_M R_{SH}}{R_S(R_M+R_{SH})}\right] + \frac{\Delta V_D R_{SH}}{R_M+R_{SH}} \tag{B.16}$$

This applies only when $V_o \geq \Delta V_D$, otherwise treat R_M as effectively open and I_M as 0. When (as should likely be the case), $R_{SH} \gg R_M$, the last equation simplifies to:

$$V_o \approx V_P\left(1+\frac{R_M}{R_S}\right) + \Delta V_D \tag{B.17}$$

B.1.0.5 Calculation of Shunt Resistor Current Loss

Start with the convenient definition:

$$\Delta V = V_o - V_P \tag{B.18}$$

$$\begin{aligned} \frac{I_{SH}}{I_{SH}+I_M} &= \frac{\Delta V}{R_{SH}}\left(\frac{\Delta V}{R_{SH}}+\frac{\Delta V-\Delta V_D}{R_M}\right)^{-1} && \text{(B.19)}\\ &= \left(\frac{R_{SH}+R_M}{R_M}-\frac{\Delta V_D R_{SH}}{R_M \Delta V}\right)^{-1} && \text{(B.20)}\\ & && \text{(B.21)} \end{aligned}$$

Now calculate ΔV:

$$\begin{aligned} \Delta V &= V_P\left[1+\frac{R_M R_{SH}}{R_S(R_M+R_{SH})}\right]-\frac{\Delta V_D}{R_M}-V_P && \text{(B.22)}\\ &= \frac{V_P R_M R_{SH}}{R_S(R_M+R_{SH})}-\frac{\Delta V_D}{R_M} && \text{(B.23)} \end{aligned}$$

Finally substitute and simplify:

$$\frac{I_{SH}}{I_{SH}+I_M} = \left[1+\frac{R_{SH}}{R_M}-\frac{\Delta V_D R_{SH}}{\frac{V_P R_{SH}}{R_S(R_M+R_{SH})}-\Delta V_D}\right]^{-1} \tag{B.24}$$

Again, this holds only when $V_o \geq \Delta V_D$, otherwise the fraction is 1.

B.1.0.6 Calculation of I_C

First, defining the voltage drop across the coil as ΔV_C:

$$\begin{aligned} \frac{I_C}{I_1+I_C} &= \frac{\frac{\Delta V_C}{R_C}}{\frac{\Delta V_C}{R_C}+\frac{\Delta V_C}{R_1}} && \text{(B.25)}\\ &= \frac{1}{R_C\left(\frac{1}{R_C}+\frac{1}{R_1}\right)} && \text{(B.26)}\\ &= \frac{1}{\left(1+\frac{R_C}{R_1}\right)} && \text{(B.27)}\\ &= \frac{R_1}{R_1+R_C} && \text{(B.28)} \end{aligned}$$

Now calculate the current $I_M = I_1 + I_C$:

$$\begin{aligned} I_M &= \frac{\Delta V - \Delta V_D}{R_M} && \text{(B.29)} \\ &= \frac{1}{R_M}\left(\frac{V_P R_M R_{SH}}{R_S(R_M + R_{SH}} - \frac{\Delta V_D}{R_M} - \Delta V_D\right) && \text{(B.30)} \\ &= \frac{V_P R_{SH}}{R_S(R_M + R_{SH})} - \frac{\Delta V_D}{R_M}\left(1 + \frac{1}{R_M}\right) && \text{(B.31)} \end{aligned}$$

Now substitute in R_M from Eqn. B.11:

$$I_M = \frac{V_P R_{SH}}{R_S\left(\frac{R_1 R_C}{R_1+R_C} + R_2 + R_{SH}\right)} - \frac{\Delta V_D}{\frac{R_1 R_C}{R_1+R_C} + R_2}\left[1 + \frac{R_1 + R_C}{R_2(R_1 + R_C) + R_1 R_C}\right] \quad \text{(B.32)}$$

And finally, we can multiply this result by the result from B.28 to obtain I_C:

$$I_C = \frac{V_P R_{SH} R_1}{R_S\left(R_1 + R_C\right)\left(\frac{R_1 R_C}{R_1+R_C} + R_2 + R_{SH}\right)} - \frac{\Delta V_D R_1}{R_1 R_C + R_2}\left[1 + \frac{R_1 + R_C}{R_2(R_1 + R_C) + R_1 R_C}\right] \quad \text{(B.33)}$$

B.1.1 Balanced polarimeter

The voltage signal from channels 1 and 2 are given by Eqns B.34 and B.35, respectively:

$$V_1 = I_1 R_{L1} \sin^2(\theta) \quad \text{(B.34)}$$
$$V_2 = I_2 R_{L2} \cos^2(\theta) \quad \text{(B.35)}$$

The difference voltage signal from the balanced polarimeter is thus:

$$\Delta V = I_1 R_{L1} \sin^2(\theta - \pi/4) - I_2 R_{L2} \cos^2(\theta - \pi/4) \quad \text{(B.36)}$$

B.1.1.1 False balance from imbalanced load resistors.

Even with identical responses in each channel ($I_1 = I_2 = I_0$), a false balance can still be induced from a difference in the load resistors. To determine the extent that imbalanced load resistances affect the output signal, start with Eqn B.36 in the identical response, "balanced" ($\Delta V = 0$) condition, given by Eqn B.37:

$$R_{L1} \sin^2\left(\theta_e - \pi/4\right) - R_{L2} \cos^2\left(\theta_e - \pi/4\right) = 0 \quad \text{(B.37)}$$

Rearranging this we get the following:

$$R_{L1}\left[1-\sin(2\theta_e)\right]-R_{L2}\left[1+\sin(\theta_e)\right]=0 \tag{B.38}$$

$$(R_{L1}-R_{L2})-(R_{L1}+R_{L2})\sin(2\theta_e)=0 \tag{B.39}$$

$$\sin(2\theta_e)=\frac{R_{L1}-R_{L2}}{R_{L1}+R_{L2}} \tag{B.40}$$

Which gives Eqn B.41 for the angle as a function of the load imbalance:

$$\theta_e=\frac{1}{2}\text{asin}\left(\frac{R_{L1}-R_{L2}}{R_{L2}+R_{L1}}\right) \tag{B.41}$$

And taking each resistor as deviations from the mean resistance such that $R_{L1}=R_L+\sigma_R$ + $R_{L2}=R_L-\sigma_R$, this equation simplifies to Eqn B.42:

$$\theta_e=\frac{1}{2}\text{asin}\left(\frac{\sigma_R}{R_L}\right) \tag{B.42}$$

B.2 Quantum Mechanics

B.2.1 J-Couplings

B.2.1.1 Commutation of the J-Coupling Hamiltonian with Scalar Polarization

The initial density matrix for a set of thermally polarized spins (Eqn. 5.7) is:

$$\rho_0=\frac{B_z}{k_BT}\sum_i\gamma_iI_{iz}$$

Where the z axis is defined as the direction of the polarization of the spins. And the Hamiltonian is given by:

$$H_J=\sum_i\sum_{j\neq i}J_{ij}\mathbf{I_i}\cdot\mathbf{I}_j=\sum_{ij}I_{ix}I_{jx}+I_{iy}I_{jy}+I_{iz}I_{jz}$$

The commutator of the density matrix and the Hamiltonian can be calculated from the cyclic commutation relationship:

$$[I_{ia},I_{jb}]=2i\delta_{i,j}\varepsilon_{abc}I_{ic} \tag{B.43}$$

The commutator is thus:

$$[\rho_0, H_J] = \frac{\bar{h}\gamma B_0}{2k_B T} \sum_i \sum_{j \neq i} J_{ij} \left[I_{iz}, I_{ix}I_{jx} + I_{iy}I_{jy} + I_{iz}I_{jz} \right] \tag{B.44}$$

$$= \frac{\bar{h}\gamma B_0}{2k_B T} \sum_i \sum_{j \neq i} J_{ij} \left([I_{iz}, I_{ix}I_{jx}] + [I_{iz}, I_{iy}I_{jy}] \right) \tag{B.45}$$

$$= \frac{\bar{h}\gamma B_0}{2k_B T} \sum_i \sum_{j \neq i} J_{ij} \left(I_{iy}I_{jx} - I_{ix}I_{jy} \right) \tag{B.46}$$

$$\tag{B.47}$$

Taking advantage of the fact that J-couplings are always symmetric (i.e. $J_{ij} = J_{ji}$), we can see that for each term $J_{ij} \left(I_{ix}I_{jy} - I_{iy}I_{jx} \right)$, there is a corresponding term $J_{ji} \left(I_{jx}I_{iy} - I_{jy}I_{ix} \right) = -J_{ij} \left(I_{ix}I_{jy} - I_{iy}I_{ix} \right)$. Using this, Eqn. B.46 can be re-written as Eqn. B.48:

$$[\rho_0, H_J] = \frac{i\Delta E}{k_B T} \left(\sum_i \sum_{j \neq i} I_{ix}I_{jy} - \sum_i \sum_{j \neq i} I_{ix}I_{jy} \right) \tag{B.48}$$

And thus $[\rho_0, H_J] = 0$.

B.2.1.2 Evolution of the initial density matrix for heteronuclei with non-identical γ

For a thermally polarized ensemble of spins $\mathbf{I}_i$ and $\mathbf{S}_n$, the initial density matrix is given by:

$$\rho_0 = \frac{\gamma_I B_0}{2k_B T} \sum_i I_{iz} + \frac{\gamma_S B_0}{2k_B T} \sum_n S_{nz} \tag{B.49}$$

$$\rho_0 = \frac{\hbar B_0}{2k_B T} \left(\gamma_i \sum_i I_{iz} + \gamma_S \sum_n S_{nz} \right) \tag{B.50}$$

$$\rho_0 = \frac{\hbar B_0(\gamma_I + \gamma_S)}{2k_B T} \left(\sum_{in} I_{iz} + S_{nz} \right) + \frac{\hbar B_0(\gamma_I - \gamma_S)}{2k_B T} \left(\sum_{in} I_{iz} - S_{nz} \right) \tag{B.51}$$

It is known from Sec. B.2.1.1 that terms of the form $I_{iz} + S_{nz}$ will commute with the Hamiltonian, but terms of the form $I_{iz} - S_{nz}$ have significant, detectable precession under the J-coupling Hamiltonian.

B.2.1.3 Evolution of a two-spin heteronuclear system under a pulse

For a two-spin system initially polarized in a state $I_z + S_z$, the application of a pulse along the y direction with pulse angle $\theta_I = \theta - \Delta$ for the $\mathbf{I}$ spins will apply a (possibly) different type angle $\theta_S = \theta + \Delta$ to the $\mathbf{S}$ spins, creating zero-quantum and double-quantum coherences

by introducing a relative tip angle 2Δ between the spins. After the pulse, the density matrix is given by:

$$\begin{aligned}
\rho(\theta) =& \cos(\theta-\Delta)I_z - \sin(\theta-\Delta)I_x + \cos(\theta+\Delta)S_z - \sin(\theta+\Delta)S_x && \text{(B.52)}\\
=& \frac{\cos(\theta-\Delta)+\cos(\theta+\Delta)}{2}(I_z+S_z) + \frac{\cos(\theta-\Delta)-\cos(\theta+\Delta)}{2}(I_z-S_z) && \text{(B.53)}\\
&- \frac{\sin(\theta-\Delta)+\sin(\theta+\Delta)}{2}(I_x+S_x) - \frac{\sin(\theta-\Delta)-\sin(\theta+\Delta)}{2}(I_x-S_x) && \text{(B.54)}\\
=& \cos(\theta)\cos\Delta(I_z+S_z) - \sin(\theta)\sin(\Delta)(I_z-S_z) && \text{(B.55)}\\
&- \sin(\theta)\cos(\Delta)(I_x+S_x) + \cos(\theta)\sin(\Delta)(I_x-S_x) && \text{(B.56)}
\end{aligned}$$

And this is probably best represented by breaking it out into antiparallel terms (which evolve under the Hamiltonian), and parallel terms (which commute with the Hamiltonian):

$$\begin{aligned}
\rho(\theta) =& \cos(\Delta)\left[\cos(\theta)(I_z+S_z) - \sin(\theta)(I_x+S_x)\right] && \text{(B.57)}\\
&- \sin(\Delta)\left[\sin(\theta)(I_z-S_z) - \cos(\theta)(I_x-S_x)\right] && \text{(B.58)}
\end{aligned}$$

And of course Δ is a function of θ, γ_I and γ_S:

$$\begin{aligned}
\theta &= \theta_I + \Delta && \text{(B.59)}\\
\theta &= \theta_S - \Delta && \text{(B.60)}\\
\Delta &= \frac{\theta_I - \theta_S}{2} && \text{(B.61)}
\end{aligned}$$

And this is further constrained by the fact that θ_I and θ_S are functions of the same underlying process:

$$\begin{aligned}
\theta_I &= \gamma_I B_1 t && \text{(B.62)}\\
\theta_S &= \gamma_S B_1 t && \text{(B.63)}\\
&= \frac{\gamma_S}{\gamma_I}\gamma_I B_1 t = \frac{\gamma_S}{\gamma_I}\theta_I && \text{(B.64)}
\end{aligned}$$

Now substituting into Δ:

$$\begin{aligned}
\Delta &= \left[\theta_I - \theta_I\left(1 - \frac{\gamma_S}{\gamma_I}\right)\right] && \text{(B.65)}\\
&= \frac{\theta_I\gamma_S}{2\gamma_I} && \text{(B.66)}
\end{aligned}$$

And from this follows θ:

$$\theta = \theta_I - \Delta = \theta_I \left(1 - \frac{\gamma_S}{2\gamma_I}\right) \tag{B.67}$$

B.3 Pulse Sequences

B.3.1 Pulse Error corrections in multiphase π trains.

When applying a π train while alternating phase between $\vec{x}$ and $\vec{y}$ pulses, the π pulses have a multually correcting effect. For errors ϵ_x and ϵ_y on π_x and π_y, respectively, the density matrix after n pulses may have some components along I_x, I_y and I_z, these coefficients are defined as:

$$\rho_n = c_{n,x} I_x + c_{n,y} I_y + c_{n,z} \tag{B.68}$$

The effect of the pulses with error are:

$$\begin{array}{ccc|ccc} & \pi_x + \epsilon_x & & & \pi_y + \epsilon_y & \\ I_x & \longrightarrow & I_x & I_x & \longrightarrow & -I_x\cos(\epsilon y) + I_z \sin(\epsilon y) \\ I_y & \longrightarrow & -I_y\cos(\epsilon_x) - I_z\sin(\epsilon_x) & I_y & \longrightarrow & I_y \\ I_z & \longrightarrow & -I_z\cos(\epsilon_x) + I_y\sin(\epsilon_x) & I_z & \longrightarrow & -I_z\cos(\epsilon_y) - I_x\sin(\epsilon_y) \end{array}$$

And so the terms of the density matrix at $2n$ and $2n{+}1$ (for $n \in [0, \inf]$) can be determined recursively:

$$\vec{c}_{2n+1} = \begin{bmatrix} c_{2n,x} \\ -c_{2n,y}\cos(\epsilon_x) + c_{2n,z}\sin(\epsilon_x) \\ -c_{2n,z}\cos(\epsilon_x) - c_{2n,y}\sin(\epsilon_x) \end{bmatrix}$$

$$\vec{c}_{2n} = \begin{bmatrix} -c_{2n-1,x}\cos(\epsilon_y - c_{2n-1,z}\sin(\epsilon_y) \\ c_{2n-1,y} \\ -c_{2n,z}\cos(\epsilon_y) + c_{2n,x}\sin(\epsilon_y) \end{bmatrix}$$

The pulses follow a 4-pulse cycle, and maximum symmetry is achieved on the $4n$ pulses. Taking $n = 0$, we can derive c_4:

$$\vec{c}_4 = \begin{bmatrix} \sin(\epsilon_y \left[\cos(\epsilon_x)\cos(\epsilon_y) + \cos^2(\epsilon_x)\cos(\epsilon_y) + \sin^2(\epsilon_x)\right] \\ -\sin(\epsilon_x)\cos(\epsilon_x)\left[\cos(\epsilon_y) - 1\right] \\ \cos^2(\epsilon_x)\cos^2(\epsilon_y) + \sin^2(\epsilon_x)\cos(\epsilon_y) + \cos(\epsilon_x)\sin^2(\epsilon_y) \end{bmatrix}$$

The signal should be all in c_z, and so we can take as our measure of the error $1 - c_z$. Expanding the terms of c_{4z} in a MacLaurin series:

$$\cos^2(\epsilon_x) = \left(\sum_{n=0}(-1)^n \frac{\epsilon_x^{2n}}{(2n)!}\right)\left(\sum_{m=0}(-1)^m \frac{\epsilon_x^{2m}}{(2m)!}\right)$$
$$= \sum_{n=0}\sum_{m=0}(-1)^{n+m}\frac{\epsilon_x^{2(n+m)}}{(2n)!(2m)!}$$

$$\sin^2(\epsilon_x) = \left(\sum_{n=0}(-1)^n \frac{\epsilon x^{2n+1}}{(2n+1)!}\right)\left(\sum_{m=0}(-1)^m \frac{\epsilon_x^{2m+1}}{(2m+1)!}\right)$$
$$= \sum_{n=0}\sum_{m=0}(-1)^{n+m}\frac{\epsilon_x^{2(n+m+1)}}{(2n+1)!(2m+1)!}$$

And adding in the cross-terms:

$$\cos^2(\epsilon_x)\cos(\epsilon_y) = \sum_{k,l,m,n=0}(-1)^{k+l+m+n}\frac{\epsilon_x^{2(n+m)}\epsilon_y^{2(k+l)}}{(2n)!(2m)!(2k)!(2l)!} \tag{B.69}$$

$$\sin^2(\epsilon_x)\cos(\epsilon_y) = \ldots \tag{B.70}$$

The first several of these terms cancel out:

$$\cos^2(\epsilon_x)\cos^2(\epsilon_y) + \sin^2(\epsilon_x)\cos(\epsilon_y) + \cos(\epsilon_x)\sin^2(\epsilon_y) = 1 - \epsilon_x^2 - \epsilon_y^2 +$$
$$\epsilon_x^2\epsilon_y^2 + \frac{\epsilon_x^4+\epsilon_y^4}{3} - \frac{\epsilon_x^2\epsilon_y^2}{3}\left(\epsilon_y^2+\epsilon_x^2\right) + \frac{\epsilon_x^4\epsilon_y^4}{9} + \epsilon_x^2 + \frac{\epsilon_x^2\epsilon_y^2}{2} + \frac{x^2y^2}{24}\left(4\epsilon_x^2+\epsilon_y^2\right) -$$
$$\frac{\epsilon_x^4\epsilon_y^4}{72} + \epsilon_y^2 + \frac{\epsilon_x^2\epsilon_y^2}{2} + \frac{x^2y^2}{24}\left(\epsilon_x^2+4\epsilon_y^2\right) - \frac{\epsilon_x^4\epsilon_y^4}{72} + O(\epsilon^{10})$$
$$= 1 + \frac{\epsilon_x^2\epsilon_y^2}{8}\left(\epsilon_x^2+\epsilon_y^2\right) - \frac{\epsilon_x^4\epsilon_y^4}{12} + O\left(\epsilon^{10}\right)$$

And for $\epsilon_x, \epsilon_y \ll 1$, the total error in the angle is:

$$\theta_e \approx \frac{\epsilon_x^2\epsilon_y^2}{8}\left(\epsilon_x^2+\epsilon_y^2\right) \tag{B.71}$$

Appendix C

Blueprints

Many of the components of the magnetometer have been designed and re-designed during the process of development of a practical device for NMR measurements. This appendix collects each of these designs for all components of the magnetometer to document these changes and give a more exact sense of the instrumentation used in these experiments. Strictly speaking, these are technical drawings produced by AutoCAD, and are not produced by blue-printing, a process wherein technical drawings are copied by the application of a light to the originals, which sit on top of a paper coated in a ferro-gallate solution. The ferro-gallate solution is light-sensitive, and where no lines are present exposure causes the ferro-gallate to convert into a stable blue dye. With the advent of digital home printing, this process is no longer necessary, and the word "blueprint" is merely a skeuomorph from a bygone age.

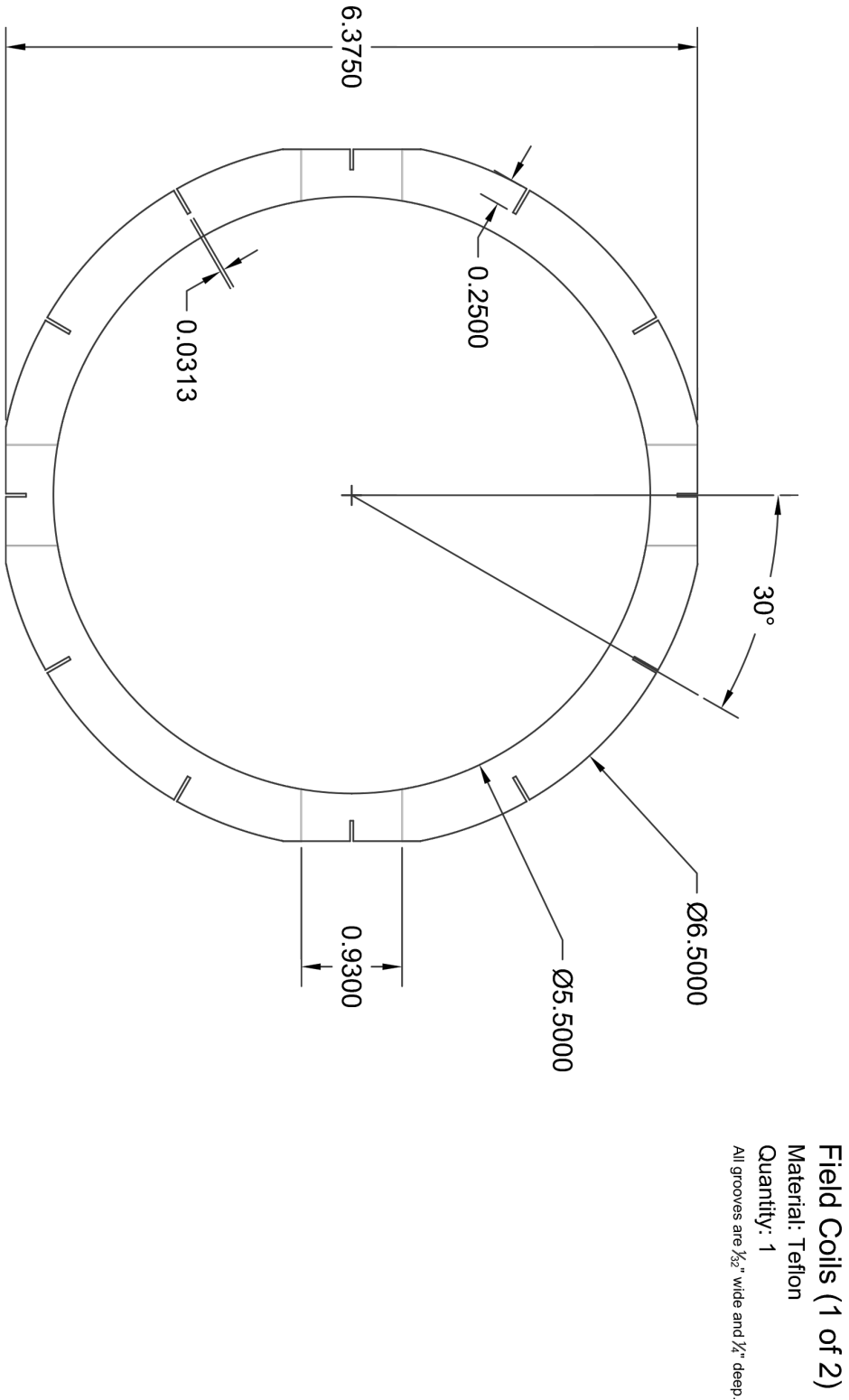


Figure C.1: The shim field coil substrate, radial (top) view.

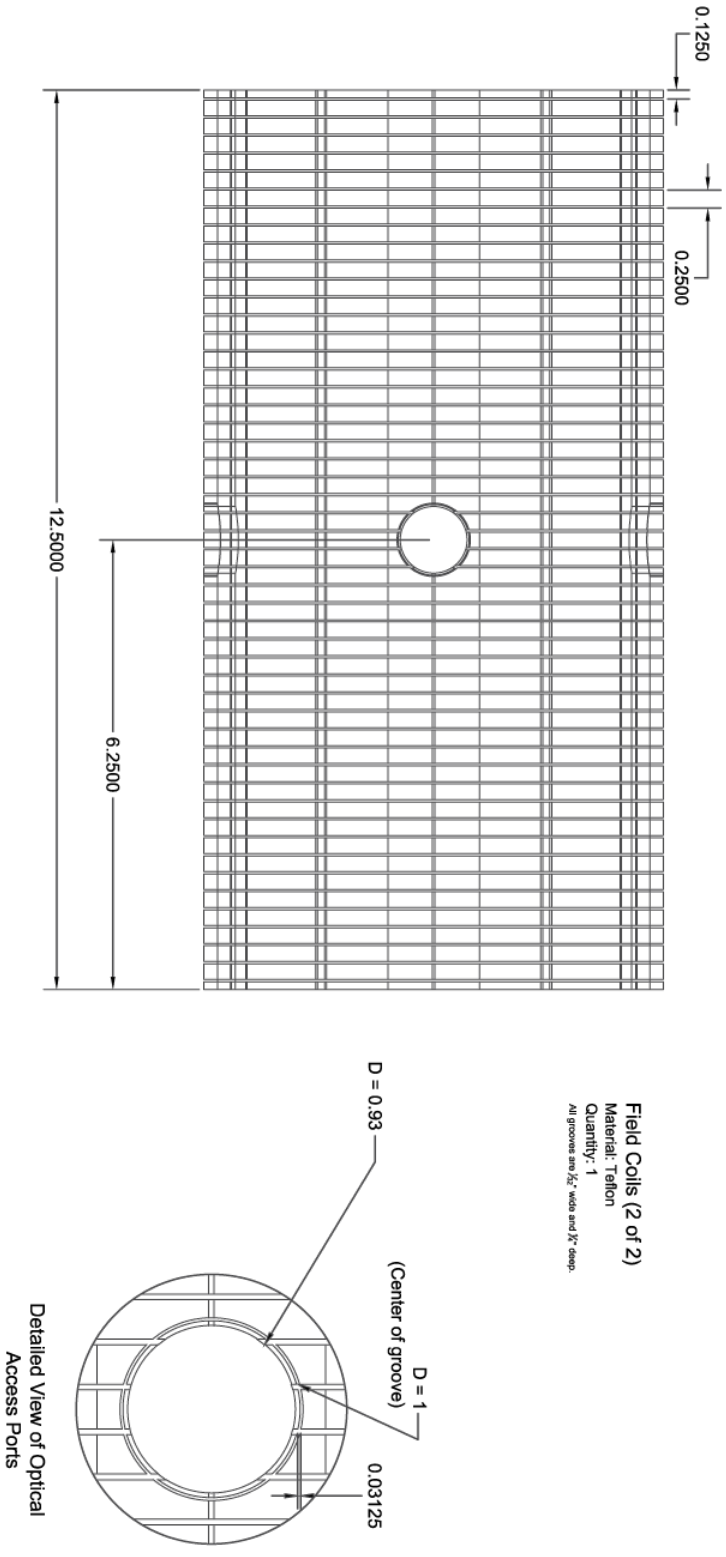


Figure C.2: The shim field coil substrate, side view.

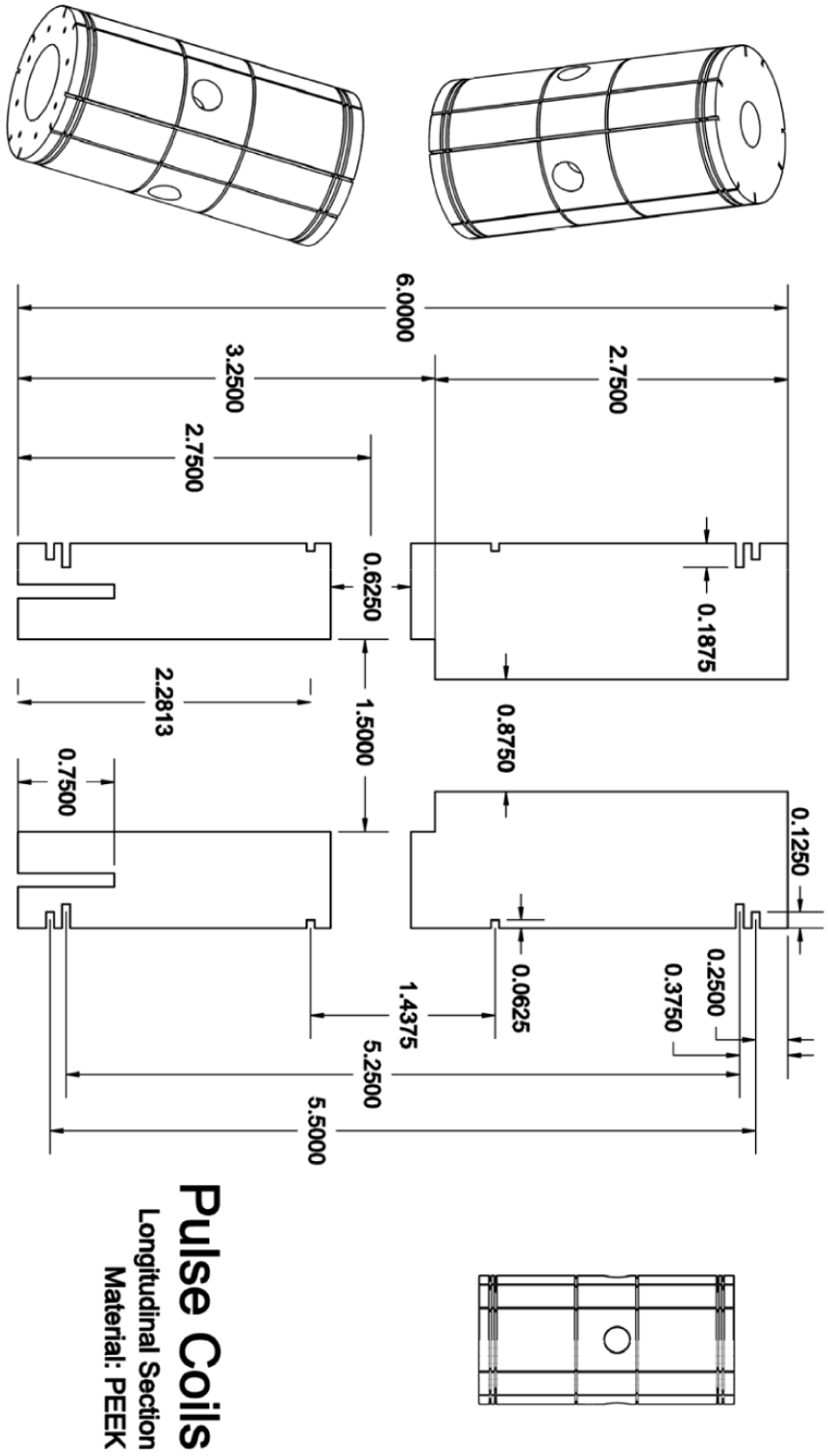


Figure C.3: The new, more homogeneous pulse coils, longitudinal (side) view.

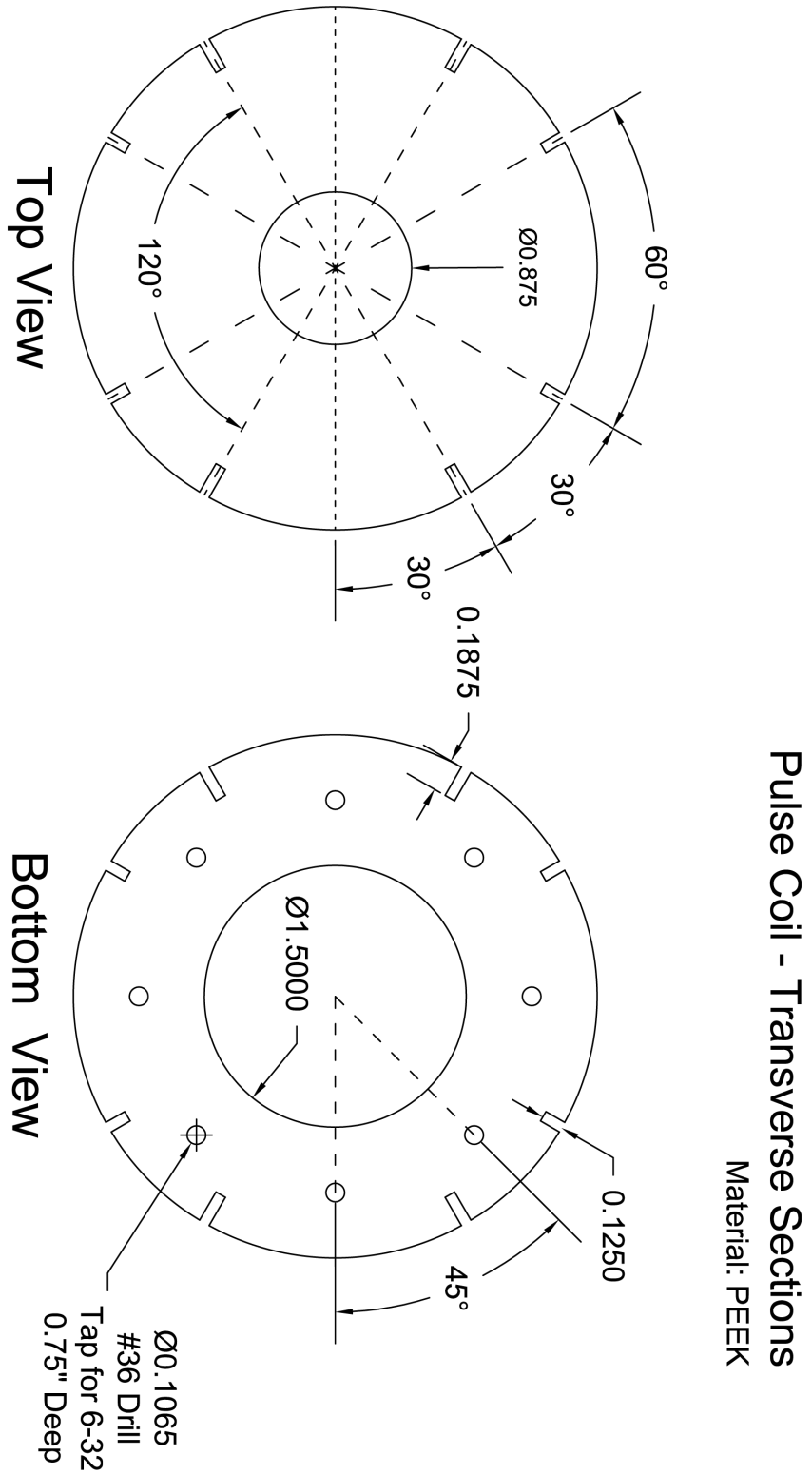


Figure C.4: The new, more homogeneous pulse coils, transverse (top) view.

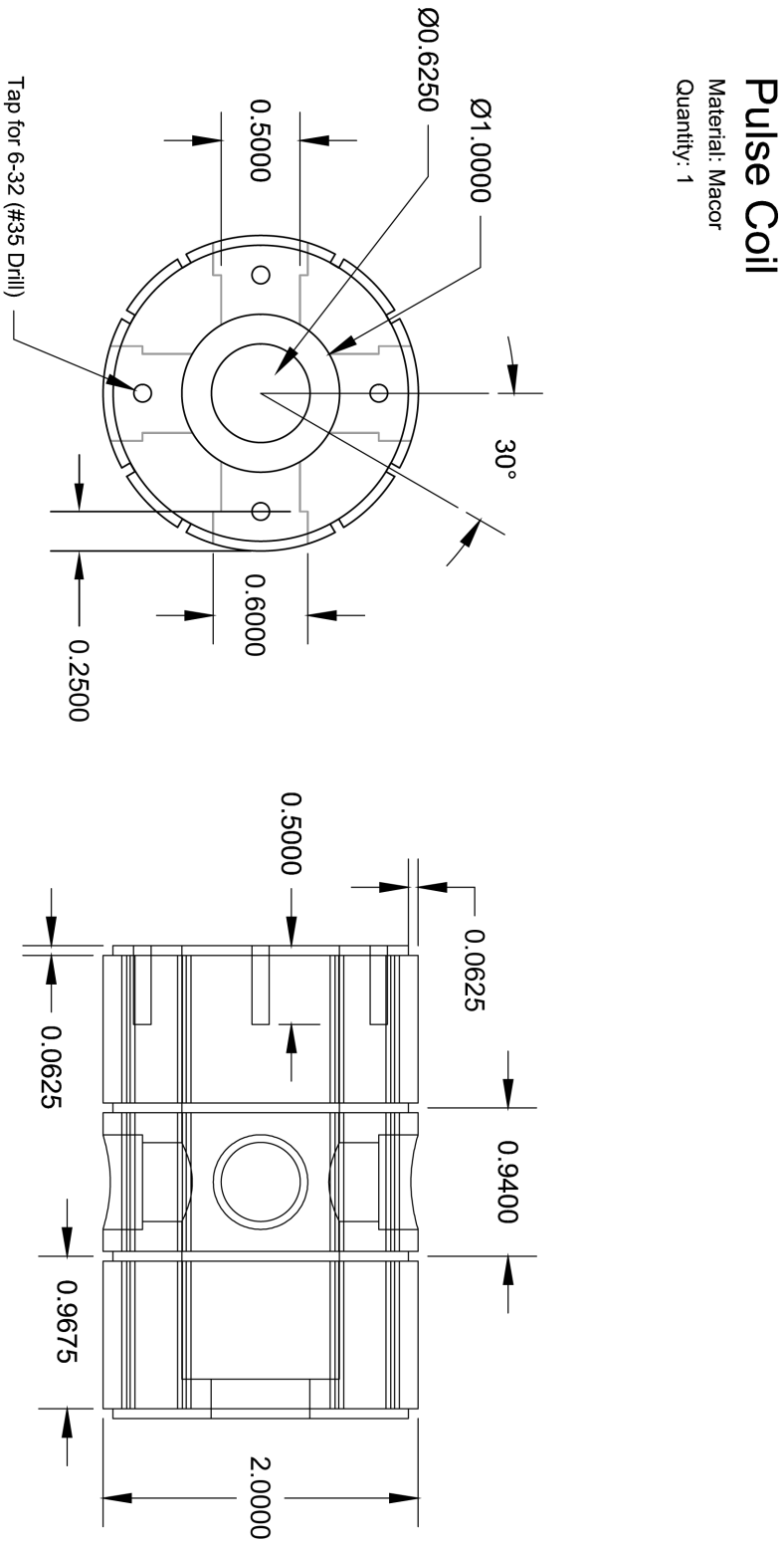


Figure C.5: The older, inhomogeneous pulse coils.

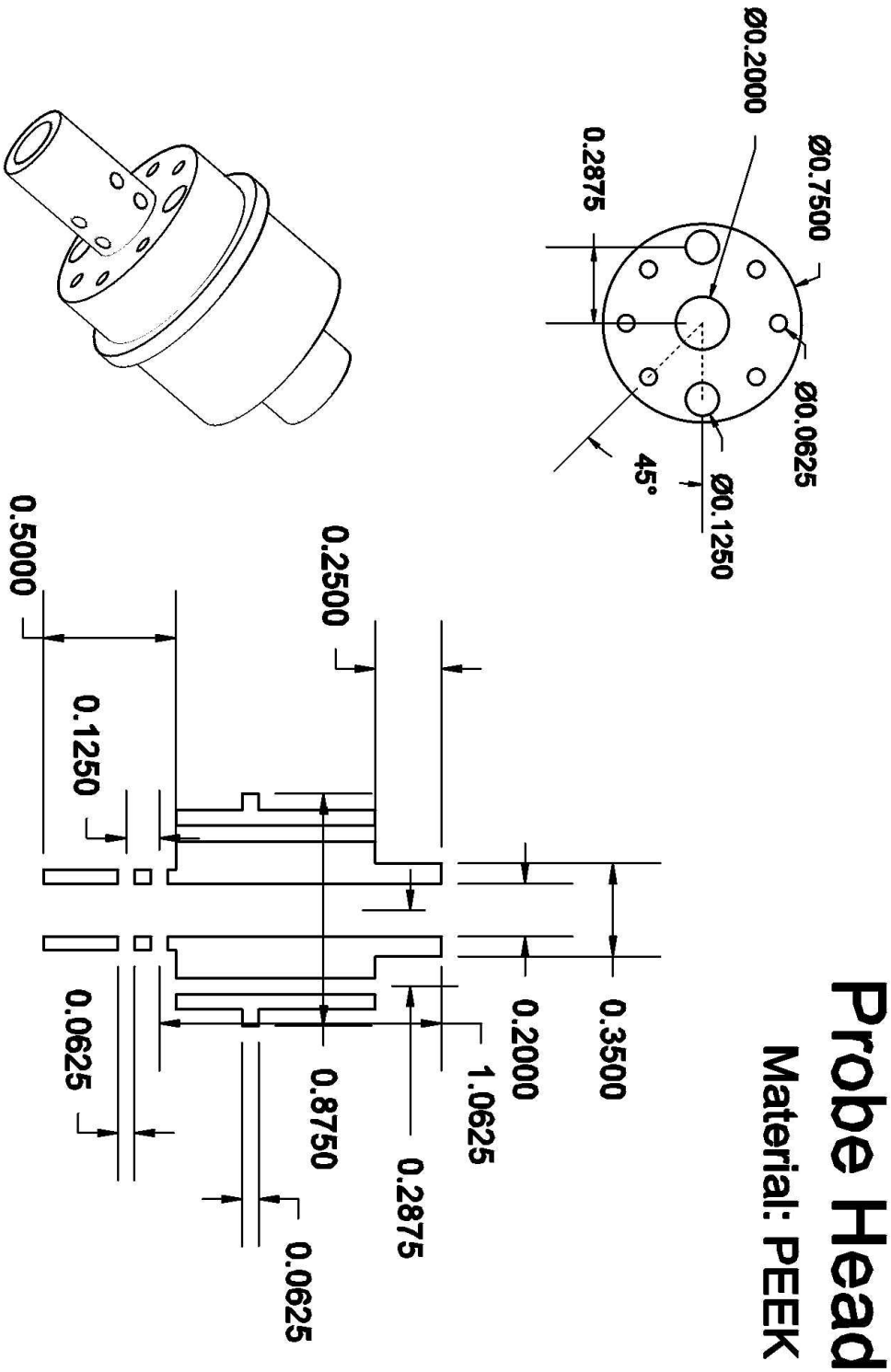


Figure C.6: The probe head body.

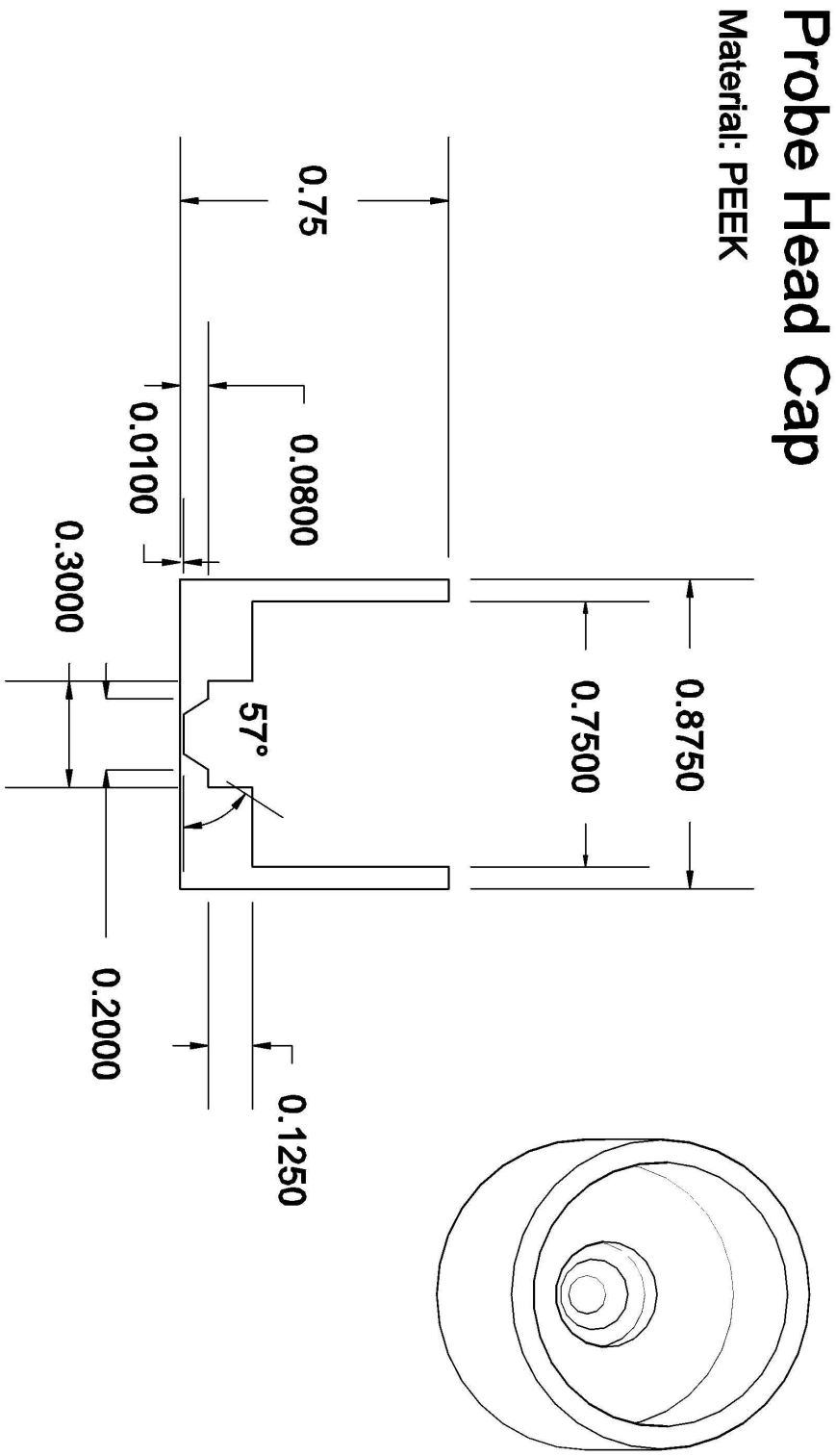


Figure C.7: The probe head cap.

Appendix D

Matlab Scripts

For posterity, a collection of Matlab scripts used in data processing and readout are included herein. These were most recently used with Matlab 7.13.0.564 and no specific effort was made to make them backwards compatible. A more complete collection of MATLAB scripts, C-code and data can currently be found at https://github.com/pganssle-research.

D.1 Binary Serialization Format

There are a suite of scripts for converting files from the binary serialization format into a Matlab-readable format. The most general, `mc_read_bin`, reads data into a `struct` which parses the available metadata in a generic way, making no assumptions about the contents of the files. The scripts `mc_read_data` and `mc_read_prog` are used for parsing .mcd and .pp files specifically, and will throw errors if the results do not conform to the relevant specifications.

D.1.1 Readout

D.1.1.1 `mc_read_bin`

```
function [out, fl] = mc_read_bin(path, histpath)
% Reading data and PPROGRAM from MCD-type files.

% Get the file - has a history to remember recent directories.
history = {};

if(~exist('histpath', 'var'))
    histpath = 'mc_read_bin_hist.mat';
end

if(~exist('path', 'var') || isnumeric(path) || ~exist(path, 'file'))
    if ~exist(histpath, 'file')
        default_dir = pwd; % If the readout_history file is missing,
        if(~exist(default_dir, 'file'))
            default_dir = pwd;
```

```
        end
    else
        load(histpath);

        if (length(history) < 1)
            default_dir = pwd;
        else
            % Now search through the history file to find the most
            % recently used one that exists. This is useful if the same
            % history file is synced across multiple systems. We'll set
            % a variable keepatmost in the readout_history.mat file, so
            % that we can adjust how long the history we want to keep
            % is. Default is keep all., keepatmost == -1 also means
            % keep all.
            default_dir = pwd;
            for j = length(history):-1:1
                if exist(history{j}, 'file')
                    default_dir = history{j};
                    break; % Stop looking once you've found it
                end
            end

            % List of the positions of duplicate entries
            dupes = ismember(history, default_dir);

            % The most recent one is OK to stay, the others shouldn't even be there.
            dupes = dupes(1:end-1);
            history(dupes) = []; %#ok<*NODEF> Delete the relevant entries
        end
    end

    % Generate the user prompt.
    [filepath,filefolder]=uigetfile({'*.mcd;*.pp';'*.pp';'*.mcd'},...
                                    'Select a binary file', ...
                                    default_dir);
    path=fullfile(filefolder,filepath);

    if (isempty(path) || (~exist(path, 'file')))
        warning('Invalid file name.');
        return;
    end

    % Update the history
    history(ismember(history, filefolder)) = [];
    history = [history, {filefolder}];

    if(exist('keepatmost', 'var') && keepatmost >= 1 && length(history) > keepatmost)
       history = history((end-keepatmost):end);
    end

    if(~isempty(history))
        if(exist(histpath, 'file'))
            save(histpath, 'history', '-append');
        else
            save(histpath, 'history');
        end
    end
end

% Create an output struct, containing all the values.
f = fopen(path, 'rb');

if(f == -1)
```

```
    error('Failed to open file.');
end

try
    out = [];
    fl = [];
    while(~feof(f))
        [val, name, fs] = read_fsave_from_file(f, 1);

        if(isempty(val) || isempty(fs))
           continue;
        end

        n2 = name;
        i = 0;
        while(isfield(out, n2))
            n2 = [name i];
            i = i + 1;
        end

        name = n2;

        eval(['out.' name ' = val;']);
        eval(['fl.' name ' = fs;']);
    end
catch msg_id
    warning(msg_id.identifier, msg_id.message);

    fprintf('Stack Trace:\n');
    stack = msg_id.stack;
    for i = 1:length(msg_id.stack)
        fprintf('line %d of %s in %s\n', stack(i).line, stack(i).name, stack(i).file);
    end

    f = fclose(f);
end

if(f)
    fclose(f);
end

function [val, name, fs] = read_fsave_from_file(f, subs)
% Read an entry from file.
fs = read_fsave_header(f);
val = [];
name = [];

if(isempty(fs))
    return;
end

% Whether or not to get sub-structs
if(~exist('subs', 'var'))
    subs = false;
else
    subs = logical(subs);
end

% If it's a container, we should call this recursively and get all the sub-structs.
FS_CONTAINER = 32;
if(~subs || fs.type ~= FS_CONTAINER)
    % Simply read out the file.
    fs.data = fread(f, fs.numel, fs.prec);
```

```
    val = fs.data;
else
    ipos = ftell(f);
    dpos = ipos;
    while(~feof(f) && dpos-ipos < fs.size)
        % Suppress the mlint warnings - they fail to detect that these variables are used
        % in the eval a bit later.
        [data, n, fs_buff] = read_fsave_from_file(f, subs); %#ok<ASGLU,NASGU>;

        i = 0;
        n2 = n;
        while(isfield(fs.data, n2))
            n2 = [n i];
            i = i+1;
        end

        n = n2;

        eval(['fs.data.' n ' =  fs_buff;']);
        eval(['val.' n ' = data;']);

        dpos = ftell(f);
    end
end

% Name output - for field names.
name = regexprep(fs.name, '[^a-zA-Z0-9_]', '');
if(isempty(name))
    name = 'noname';
end

if(isstrprop(name(1), 'digit'))

    name = sprintf(['%s%0', num2str(length(name)), 'd'], 'ind_', str2double(name));
end

function fs = read_fsave_header(f)
% Reads fsave header from file into a struct - doesn't rewind.
fs = [];

% Read the name of the thing
spos = ftell(f);

ns = fread(f, 1, 'uint');
if(isempty(ns))
    return;
end

if(ns >= 1)
    % Remove null terminations.
    name = fread(f, ns, '*char');
    name = deblank(name');
else
    name = '';
end

% Get the type
type = fread(f, 1, 'uchar');

% The size now
s = fread(f, 1, 'uint');

% Finally the array.
```

```
[prec, ts] = get_precision(type);

if(ts <= 0)
    return;
end

numel = s/ts;

if(numel <= 0)
    return;
end

dpos = ftell(f);

fs.name = name;
fs.type = type;
fs.prec = prec;
fs.numel = numel;
fs.size = s;
fs.spos = spos;
fs.dpos = dpos;
fs.data = [];

function [prec, s] = get_precision(type)
% Gets the precision based on the type of an fsave.
FS_NULL = 0;            %#ok<NASGU>; - For reference.
FS_CHAR = 1;
FS_UCHAR = 2;
FS_INT = 3;
FS_UINT = 4;
FS_FLOAT = 5;
FS_DOUBLE = 6;
FS_INT64 = 7;
FS_UINT64 = 8;
FS_CONTAINER = 32;
FS_CUSTOM = 64;

switch(type)
    case FS_CHAR
        prec = '*char';
        s = 1;
    case FS_CONTAINER
        prec = 'char';
        s = 1;
    case FS_CUSTOM
        prec = 'uchar';
        s = 1;
    case FS_UCHAR
        prec = 'uchar';
        s = 1;
    case FS_INT
        prec = 'int';
        s = 4;
    case FS_UINT
        prec = 'uint';
        s = 4;
    case FS_FLOAT
        prec = 'float';
        s = 4;
    case FS_DOUBLE
        prec = 'double';
        s = 8;
    case FS_INT64
```

```
        prec = 'int64';
        s = 8;
    case FS_UINT64
        prec = 'uint64';
        s = 8;
    otherwise
        prec = '';
        s = -1;
end
```

D.1.1.2 mc_read_data

Requires: mc_read_bin, mc_read_prog, find_loop_locs, process_data

```
function [out, fpath] = mc_read_data(fpath, ns)
% Function for reading out data structures from .mcd files.
%
% Inputs:
% fpath:    The path of the file to read out (optional: prompts if missing)
% ns:       The number of "spans" to include in the curve averaging portion
%           of magnetization measurement experiments.
%
% [out, fpath] = mc_read_data([fpath, ns])

if(~exist('fpath', 'var') || isempty(fpath) || ~ischar(fpath))
    fpath = -1;
end

if(~exist('ns', 'var'))
    ns = 1;
end

% Get the raw structure (this will create a history file)
[s, f] = mc_read_bin(fpath, 'mc_read_data_hist.mat');

out = [];
if(isempty(f))
    return;
end

% Separately process the groups
MCD_DATAHEADER = '[Data Header]';
MCD_DISPHEADER = '[Display Header]';
MCD_DATAGROUP = '[DataGroup]';

% Data header should come first - That'll be the main portion of the
% structure - so those are top-level values.
s1 = find_struct_by_name(f, MCD_DATAHEADER);
if(isempty(s1))
    return;
end

% Data Structure Names
MCD_FNAME = 'filename';
MCD_ENAME = 'ExperimentName';
MCD_ENUM = 'ExperimentNum';
MCD_DATADESC = 'Description';
MCD_HASH = 'HashCode';
MCD_NCHANS = 'NumChans';
MCD_TSTART = 'TimeStarted';
MCD_TDONE = 'TimeDone';
MCD_CIND = 'CurrentIndex';
```

```
out.Filename = [];
out.ExperimentName = [];
out.ExperimentNum = [];
out.hash = [];
out.tstart = [];
out.tdone = [];
out.nc = 0;
out.cind = -1;

sb = find_struct_by_name(s1.data, MCD_FNAME);
if(~isempty(sb))
    fname = deblank(sb.data');
    li = find(fname == '\', 1, 'last');
    if(isempty(li) || li == length(fname))
        out.Filename = fname;
    else
        out.Filename = fname((li+1):end);
    end
end

sb = find_struct_by_name(s1.data, MCD_ENAME);
if(~isempty(sb))
    out.ExperimentName = deblank(sb.data');
end

sb = find_struct_by_name(s1.data, MCD_ENUM);
if(~isempty(sb))
    out.ExperimentNum = sb.data;
end

sb = find_struct_by_name(s1.data, MCD_DATADESC);
if(~isempty(sb))
    out.desc = deblank(sb.data');
end

sb = find_struct_by_name(s1.data, MCD_HASH);
if(~isempty(sb))
    out.hash = sb.data;
end

sb = find_struct_by_name(s1.data, MCD_TSTART);
if(~isempty(sb))
    out.tstart = deblank(sb.data');
end

sb = find_struct_by_name(s1.data, MCD_TDONE);
if(~isempty(sb))
    out.tdone = deblank(sb.data');
end

sb = find_struct_by_name(s1.data, MCD_NCHANS);
if(~isempty(sb))
    out.nc = sb.data;
end

sb = find_struct_by_name(s1.data, MCD_CIND);
if(~isempty(sb))
    out.cind = sb.data;
end

% Display header is next - we can just do a direct dump
out.disp = [];
```

```
[~, loc] = find_struct_by_name(f, MCD_DISPHEADER);
if(isfield(s, loc))
    out.disp = eval(['s.' loc]);
end

if(isfield(out, 'disp'))
    % These until these are added to Experimental Parameters

    out.disp.z_cal = 0.1379; % Current z calibration
    out.disp.G_cal = 0.0447;  % Current gradient calibration
end

% Read the program.
out.prog = mc_read_prog(s);

if(~isempty(out.prog))
    np = out.prog.np;
    sr = out.prog.sr;
    out.t = linspace(0, np/sr, np);

end

% Get the data itself
out.mdata = [];
[~, loc] = find_struct_by_name(f, MCD_DATAGROUP);
if(isfield(s, loc))
    dg = s.(loc);
    if(isstruct(dg))
        fn = fieldnames(dg);

        if(length(fn) >= 1)
            if(isfield(out.prog, 'steps'))
                ps = out.prog.steps;
            else
                ps = length(fn);
            end

            ci = num2cell(ps);
            data = zeros(length(dg.(fn{1})), ci{:});

            % Parse the index - first we need to figure out the number of
            % digits in each number.
            ind = fn{1};
            ns2 = length(ps);
            ndig = (length(ind)-4)/ns2;

            for i = 1:length(fn)
                % Now we can parse the actual indexes
                % The +1 is because it's a 1-based index in Matlab, but in C we
                % stored it as a 0-based index.
                ci = num2cell(str2num(reshape(fn{i}(5:end), ndig, ns2)')+1); %#ok<ST2NM>;
                data(:, ci{:}) = dg.(fn{i});
            end

            out.mdata = data;

            if(length(fn) > 1)
                out.adata = mean(out.mdata, 2);
            end
        end
    end
end
```

```
if(isfield(out, 'prog') && isfield(out.prog, 'instrs') && out.prog.use_pb)
    % Find all the loop locations which it could be (starting with the
    % instruction which triggers the scan.

    sn = find(out.prog.ps.instrs.scan == 1, 1, 'first');
    tlspans = find_loop_locs(out.prog.ps, sn, 1);

    % For testing for x-y-x-y trains:
    xpulse = bitor(2^6, 2^7);
    ypulse = bitor(2^8, 2^9);
    xy = 0;

    if(sn > 0 && ~isempty(tlspans))
        instrs = out.prog.ps.instrs;
        ins = 0;
        r_loop = 0;

        for i = 1:size(tlspans, 1)
            % Check for an x-y-x-y train.

            if(tlspans(i, 2)-tlspans(i, 1) == 3)
                % Two possibilities here - wait-pulse-wait-pulse or
                % pulse-wait-pulse-wait

                sp = tlspans(i, 1);

                if((bitand(xpulse, instrs.flags(sp)) && ...
                    bitand(ypulse, instrs.flags(sp+2))) || ...
                    (bitand(ypulse, instrs.flags(sp)) && ...
                    bitand(xpulse, instrs.flags(sp+2))))

                    % Pulse-wait-pulse-wait condition

                    r_loop = 1;
                    c_l = [instrs.ts(sp+1), instrs.ts(sp+3)];

                    cins = [sp+1, sp+3];
                    ins = i;
                    xy = 1;
                    break;
                elseif((bitand(xpulse, instrs.flags(sp+1)) && ...
                    bitand(ypulse, instrs.flags(sp+3))) || ...
                    (bitand(ypulse, instrs.flags(sp+1)) && ...
                    bitand(xpulse, instrs.flags(sp+3))))

                    % Wait-pulse-wait-pulse condition

                    r_loop = 1;
                    c_l = [instrs.ts(sp), instrs.ts(sp+2)];

                    xy = 1;
                    cins = [sp, sp+2];
                    ins = i;
                    break;
                end
            end

            for j = tlspans(i, 1):tlspans(i, 2)
                if(instrs.flags(j) == 0 && instrs.ts(j) > 20e-3)
                    % Loop located.
                    r_loop = 1;
                    c_l = instrs.ts(j);
                    cins = j;
```

```
            ins = i;
            break;
        end
    end
end

if(r_loop)
    ad = zeros(out.prog.nDims, 1);

    if(out.prog.varied && isfield(out.prog, 'vins'))
        vins = out.prog.vins + 1;
        ad1 = arrayfun(@(j)j > sn && j < tlspans(ins,2 ), vins);

        for i = 1:out.prog.nDims
            ad(i) = any(ad1(out.prog.vinsdim == i));
        end


    end

    sd = size(out.mdata);
    cc = num2cell(ones(size(sd)));

    ad2 = [0; 0; ad];
    ad2 = logical(ad2);

    if(sum(ad) == 0)
        ts = 1;
        vinstrs = [];
        ind = [];
    else
        ind = sd(ad2);

        ts = prod(ind);
        vinstrs = out.prog.ps.vinstrs;
    end

    out.win.ad = ad2;
    out.win.ind = ind;

    ad = logical(ad);

    l_l = ones(ts, 1);
    t_t = zeros(ts, 1);
    c_t = zeros(ts, 1);
    e_t = zeros(ts, 1);

    if(isfield(out.prog, 'vins') && ~isempty(intersect(cins, out.prog.vins)))
        clb = c_l;
        c_l = zeros(ts, 1);
        c_l(1) = clb;
    else
        c_l = ones(ts, 1)*c_l;
    end

    for i = 1:ts
        if(~isempty(vinstrs))
            [cc{ad}] = ind2sub(ind, i);
            instrs = out.prog.ps.vinstrs(cc{:});
        end

        l_l(i) = instrs.data(tlspans(ins, 1));
        t_t(i) = calc_span_length(instrs, tlspans(ins, :)); % Get loop length.
```

```
    if(c_l(i) ~= 0)
        c_l(i, :) = instrs.ts(cins);
    end

    c_t(i) = t_t(i)/l_l(i); % Get per-loop length.
    e_t(i) = calc_span_length(instrs, [sn, tlspans(ins,1)-1]);
end

c_t = t_t./l_l;

c_t = c_t * 1000; % In ms;

% Calculate the spans we'd like to skip.
ne = 1;
start = e_t*1000+c_t*ns;
num_win = floor((out.t(end)*1000 - start)./c_t - ne);

if(num_win > l_l - ne)
    num_win = l_l - ne;
end

ecb = 'out.mdata(:, :, ';

cc = num2cell(ones(size(sd)));

out.odata = out.mdata;

for i = 1:ts
    % Generate a command
    if(~isempty(ind))
        [cc{ad2}] = ind2sub(ind, i);
        inds = '';
        % The inds array grows with every iteration. This is probably
        % not the limiting speed factor here, so not bothering to
        % pre-allocate the array.
        for j = 1:length(ad)
            if(~ad(j))
                inds = [inds, ', :'];                    %#ok<AGROW>;
            else
                inds = [inds, ', ', num2str(cc{j+2})];   %#ok<AGROW>;
            end
        end

        inds = inds(3:end);

        ec = [ecb, inds, ');'];
        cdata = eval(ec);
    else
        cdata = out.mdata;
    end

    asym = 0.9;
    frac = 0.75;

    c_l = (c_l*1000)-20; % 20ms of this will be useless.

    if(~xy)
        frac = frac*(c_l./c_t); % Take 75% of remaining fraction.

        [points, out.win.spans{i}, ~, t_c] = get_subset(cdata, ...
            c_t(i), start(i), asym, frac(i), ...
            num_win(i), out.prog.sr);
```

```
else
    % If we're in xy mode, we want to get two subets with
    % different fractions then interpolate them.

    cl1 = c_l(1);
    cl2 = c_l(2);

    frac1 = frac*(cl1./c_t);
    frac2 = frac*(cl2./c_t);

    % This is the tricky part - each one is a fraction of a
    % fraction. We'll assume that the pulse is an insignificant
    % fraction of the total in any given loop.
    %
    % Here's how this works:
    %         --------
    %        |        |       cl1
    %   a1   |        |   a2   |     a3
    % ______|   b1   |________|___________________
    %
    % Normally, asym = a1/(a1+a2), but in this case we need to
    % transform that in such a way that it represents
    % a1/(a1+a2+a3), so as to ignore a3 (the second lobe). We
    % can do this by multiplying by (a1+a2)/(a1+a2+a3).
    %
    % For the second lobe:
    %
    %                                  --------
    %                         cl1      |        |
    %       a3              | a1  |        |   a2
    %___________________| ______|   b1   |________
    %
    % We want to transform it in this diagram such that
    % asym2 = (a1+a3)/(a1+a2+a3), we can do this by multiplying
    % asym by (a1+a2)/(a1+a2+a3) as before, then adding
    % a3/(a1+a2+a3).

    ws1 = frac1*c_t;
    ws2 = frac2*c_t;

    cl1 = cl1+20;
    cl2 = cl2+20;

    at1 = c_t-ws1;
    at2 = c_t-ws2;

    asym1 = asym*(c_t-cl2-ws1)/at1;
    asym2 = asym*(c_t-cl1-ws2)/at2+cl1/at2;

    % Do this twice and interpolate later - it may be preferable
    % to change this function to accept multiple asym/frac
    % values to get subsets of subsets.
    [p1, spans1, ~, t_c1] = get_subset(cdata, c_t(i), ...
        start(i), asym1, frac1, ...
        num_win(i), out.prog.sr);

    [p2, spans2, ~, t_c2] = get_subset(cdata, c_t(i), ...
        start(i), asym2, frac2, ...
        num_win(i), out.prog.sr);

    points = interp_mat(p1, p2, 2);
    out.win.spans{i} = spans1 - spans2;
```

```
    t_c = interp_mat(t_c1, t_c2);
end
points = mean(points, 1);
s2 = size(points);

if(~isvector(points))
    points = reshape(points, s2(2:end));
else
    % For whatever reason the vectors come out as row vectors.
    points = points';
end

points = points * (-2*mod(ns+1, 2)+1);

out.win.it{i} = t_c;

out.win.spans{i} = out.win.spans{i} - mean(out.win.spans{i});

t_c = t_c/1000;

if(out.disp.poly_ord >= 0 && out.disp.poly_ord < 99)
    s2 = size(points);

    % Get the fits
    warning('off','all');
    pfit = zeros(out.disp.poly_ord+1, size(cdata(:, :), 2));
    ocdata = cdata;

    for j = 1:size(cdata(:, :), 2)
        pfit(:, j) = polyfit(t_c, points(:, j), out.disp.poly_ord);
        cdata(:, j) = ocdata(:, j)-polyval(pfit(:, j), out.t');
        points(:, j) = points(:, j)-polyval(pfit(:, j), t_c);
    end

    out.win.polyfit{i} = reshape(pfit, [out.disp.poly_ord+1, s2(2:end)]);
    if(~isempty(ind))
        ec = [ecb, inds, ') = cdata;'];
        eval(ec);
    else
        out.mdata = cdata;
    end
    warning('on','all');
end

if(xy)
    num_win(i) = num_win(i)*2;
end

c = zeros([num_win(i)-1, s2(2:end)]);
ct = zeros(num_win(i)-1, 1);
c(1:2:end, :) = points(1:2:(end-1), :) - points(2:2:end, :);
c(2:2:end, :) = points(3:2:end, :) - points(2:2:(end-1), :);

ct(1:2:end) = (t_c(1:2:(end-1))+t_c(2:2:end));
ct(2:2:end) = (t_c(3:2:end)+t_c(2:2:(end-1)));

ct = ct/2;

out.win.p{i} = points;
out.win.ap{i} = mean(points, 2);
out.win.c{i} = c;
out.win.ac{i} = mean(c, 2);
out.win.ct{i} = ct;
```

```
            end

            if(~isempty(ind) && length(ind) > 1)
                out.win.c = reshape(out.win.c, ind);
                out.win.ac = reshape(out.win.ac, ind);
                out.win.ct = reshape(out.win.ct, ind);
                out.win.it = reshape(out.win.it, ind);
                out.win.ap = reshape(out.win.ap, ind);
                out.win.p = reshape(out.win.p, ind);
                out.win.polyfit = reshape(out.win.polyfit, ind);
                out.win.spans = reshape(out.win.spans, ind);
            end
        end
    end
end

if(~isfield(out, 'win'))
    ord = 2;

    if(isfield(out, 'disp') && isfield(out.disp, 'polyord'))
        ord = out.disp.polyord;
    end

    out = process_data(out, ord);
end
out = add_fft(out);

function [s, loc] = find_struct_by_name(in, name)
% Find a struct from its .name parameter.
s = [];
loc = [];
flist = fieldnames(in);

for i = 1:length(flist)
    b = in.(flist{i});

    if(isfield(b, 'name') && strcmp(b.name, name))
        s = b;
        loc = [flist{i}];
        break;
    end

    if(isstruct(b.data))
        [s, l] = find_struct_by_name(b.data, name);
        if(~isempty(s))
            loc = [flist{i} '.' l];
            break;
        end
    end
end

function [out, spans, subset, tc] = get_subset(data, len, start, asym, ...
                                                frac, num_windows, sr)
% Gets a subset of the data, returns a variable "spans" indicating where
% the subsets were taken from (scaled to the min/max of the data).
%
% All outputs other than data and len are optional. sr is only optional if
% data is a standard struct. Pass 'adata' to 'sr' if you want to use the
% average data from the struct.
%
% start, asym, frac and num_windows will use default values if they are set
% to any negative number.
%
```

```
% Default values:
% start = 0 ms
% asym = 0.5
% frac = 0.5
% num_windows = (all available)
%
% Usage:
% [out, spans, subset, tc] = get_subset(data, len, start, asym, frac, num_windows, sr);

if ~exist('data', 'var') || ...
    (isstruct(data) && ~isfield(data, 'mdata') && ~isfield(data, 'adata'))
    error('Must supply data!');
end

if(~exist('len', 'var'))
    error('Must supply a length of the subset.');
end

adata = 0; % Boolean, for later

if ~exist('sr', 'var') || ischar(sr)
    if(exist('sr', 'var') && strcmp(sr, 'adata'))
        adata = 1;
    end

    if ~isstruct(data) ...
        || ~isfield(data, 'prog') ...
        || ~isstruct(data.prog) ...
        || ~isfield(data.prog, 'sr')
        error('Must provide sampling rate if data is not a well-formed structure.');
    else
       sr = data.prog.sr;
    end
end

sr = sr/1000;  % Convert to kHz.

% We need to read the data we'll be working with for the next part
if isstruct(data)
   if(adata && isfield(data, 'adata'))
       dat = data.adata;
   else
       dat = data.mdata;
   end
else
    dat = data;
end

if(isempty(dat))
    error('Must supply data!');
end

% Data can be any size, but it must be of the form [data {everything else}].
ds = size(dat);
tl = ds(1)/sr;     % Total length, in ms.

if(len > tl)
   error('Subset length cannot be longer than total length.');
end

if(~exist('start', 'var') || start < 0 || start >= tl)
    start = 0;
end
```

```
tl2 = tl-start;     % Total length, minus start offset

start = start*sr; % How many samples is this?

if(~exist('num_windows', 'var') || num_windows > tl2/len || num_windows <= 0)
    % If num_windows is not specified or is too many, get them all.
    num_windows = floor(tl2/len);
end

dlen = len*sr;
if(dlen ~= round(dlen))
 warning(['Length is not an even multiple of the sampling rate -'...
         ' this could cause minor issues.']);
end

dlen = len*sr;
if(dlen <= 0)
    error('Length too short.');
end

if(~exist('frac', 'var') || frac <= 0 || frac > 1)
    frac = 0.5; % Make it about half the length.
end

window = ceil(frac*dlen); % How many points in each window.

if(window == 0)
    error('Fraction of length too short.');
end

if(~exist('asym', 'var') || asym < 0 || asym > 1)
   asym = 0.65;
end

% Now we have the inputs we need, we can get the outputs.
if(length(ds) > 1)
    c = num2cell(ds(2:end));
else
    c = {1};
end

tlen = length(dat(1, :));
spans = zeros(ds(1), 1);

% Where to start within a window
off = (dlen-window)*asym;
if((off+window)>dlen)
    off = dlen-window;
end

% Where to sample from
indices_pos = cell2mat(arrayfun(@(x)(x*dlen+off+1):(x*dlen+off+window), ...
                                0:2:(num_windows-1), ...
                                'UniformOutput', false))' + start;

indices_neg = cell2mat(arrayfun(@(x)(x*dlen+off+1):(x*dlen+off+window), ...
                                1:2:(num_windows-1), ...
                                'UniformOutput', false))' + start;

indices = cell2mat(arrayfun(@(x)(x*dlen+off+1):(x*dlen+off+window), ...
                            0:1:(num_windows-1), ...
                            'UniformOutput', false))' + start;
```

```
% Round at the end to reduce rounding errors.
indices_pos = round(indices_pos);
indices_neg = round(indices_neg);
indices = round(indices);

% Set the span outputs - scaled to the outputs
% Now the actual output vector
out = cell2mat(arrayfun(@(x)dat(indices, x), 1:tlen, 'UniformOutput', false));
out = reshape(out, window, num_windows, c{:});

Min = min(out(:));
Max = max(out(:));
s = (Max-Min)*0.05;
smean = mean(out(:));

spans(:) = smean;
spans(indices_neg) = Min-s;
spans(indices_pos) = Max-s;

subset = indices;
tc = (indices(1:window:(end-1))+indices(window:window:end))/(2*sr);

function len = calc_span_length(instrs, span)
% Calcualtes the length (in seconds) of a span of instructions.
%
% Inputs:
% instrs:   An instrs structure as found in mc_struct.prog.ps.instrs
% span:     A 1x2 array of the form [s, e] where s = start of span and e =
%               end of span. Uses a 1-based index. Default is full span.
%
% Outputs:
% len:      Time in seconds.
%
% Usage:
% len = calc_span_length(instrs[, span]);

% These are the instruction op-codes. Suppressing "unused variables" here
% so that they can serve as documentation in case we want them later.
CONTINUE = 0;       STOP = 1;           LOOP = 2;           %#ok<NASGU>;
END_LOOP = 3;       JSR = 4;            RTS = 5;            %#ok<NASGU>;
BRANCH = 6;         LONG_DELAY = 7;     WAIT = 8;           %#ok<NASGU>;

is_loop = 0;
len = 0;
spans = [];

if(~exist('span', 'var'))
    span = [1, instrs.ni];
end

if (instrs.instr(span(1)) == LOOP ...
   && instrs.instr(span(2)) == END_LOOP ...
   && instrs.data(span(2)) == span(1)-1)
    spans = find_loop_locs(instrs, [span(1)+1, span(2)-1], 1);
    l_dat = instrs.data(span(1));
    is_loop = 1;
end

for i = span(1):span(2)
    if(~isempty(spans) && ...
       ~isempty(find(arrayfun(@(x, y)i >= x && i <= y, spans(:, 1), spans(:, 2)), 1)))
        continue;
```

```
    end

    if(instrs.instr(i) == LONG_DELAY)
        len = len + instrs.ts(i)*instrs.data(i);
    else
        len = len + instrs.ts(i);
    end
end

for i = 1:size(spans, 1)
    len = len + calc_span_length(instrs, spans(i, :));
end

if(is_loop)
    len = len * l_dat;
end
```

D.1.1.3 mc_read_prog

Requires: `find_loop_locs`

```
function prog = mc_read_prog(path)
% Either pass this a path to load or pass it an mc_read_bin struct and it
% will parse out a program from it.
%
% Usage:
% prog = mc_read_prog;
% prog = mc_read_prog(path);
% prog = mc_read_prog(struct);

if(~exist('path', 'var'))
    path = -1;
end

if(~isstruct(path))
    s = mc_read_bin(path, 'mc_read_pp_hist.mat');
else
    s = path;
end

% Field names and such
MCD_PROGHEADER = 'PulseProgram';

MCD_NDPC = 'NDPC';
MCD_ANALOGOUT = 'AnalogOutput';
MCD_PULSEPROPS= 'Properties';
MCD_INSTRUCTIONS = 'Instructions';

MCD_STEPS = 'steps';
MCD_MAXSTEPS = 'maxsteps';
MCD_DATAEXPRS = 'dataexprs';
MCD_DELAYEXPRS = 'delayexprs';
MCD_VINS = 'v_ins';
MCD_VINSDIM = 'v_ins_dim';
MCD_VINSMODE = 'v_ins_mode';
MCD_VINSLOCS = 'v_ins_locs';

MCD_AOVARIED = 'ao_varied';
MCD_AODIM = 'ao_dim';
MCD_AOVALS = 'ao_vals';
```

```
prog = [];
p = find_struct_by_field(s, MCD_PROGHEADER);
if(~isempty(s))
    prog = p.(MCD_PULSEPROPS);

    if(isfield(p, MCD_NDPC))
        s1 = p.(MCD_NDPC);
        if(isfield(s1, MCD_MAXSTEPS))
            prog.maxsteps = s1.(MCD_MAXSTEPS);
        end

        if(isfield(s1, MCD_STEPS))
            prog.steps = s1.(MCD_STEPS);
        end

        if(isfield(s1, MCD_VINS))
            prog.vins = s1.(MCD_VINS);
        end

        if(isfield(s1, MCD_VINSDIM))
            prog.vinsdim = s1.(MCD_VINSDIM);
        end

        if(isfield(s1, MCD_VINSMODE))
            prog.vinsmode = s1.(MCD_VINSMODE);
        end

        if(isfield(s1, MCD_VINSLOCS))
            prog.vinslocs = s1.(MCD_VINSLOCS);
        end

        if(isfield(s1, MCD_DELAYEXPRS))
            prog.delayexprs = s1.(MCD_DELAYEXPRS);
        end

        if(isfield(s1, MCD_DATAEXPRS))
            prog.dataexprs = s1.(MCD_DATAEXPRS);
        end


    end

    if(isfield(p, MCD_ANALOGOUT))
        s1 = p.(MCD_ANALOGOUT);
        if(isfield(s1, MCD_AOVALS))
            prog.aovals = s1.(MCD_AOVALS);
        end

        if(isfield(s1, MCD_AOVARIED))
            prog.aovaried = s1.(MCD_AOVARIED);

            if(any(prog.aovaried) && isfield(s1, MCD_AODIM))
                prog.aodim = s1.(MCD_AODIM);

                prog.aodim(prog.aodim > 8) = -1;
                prog.aodim = prog.aodim+1;
            end
        end
    end

    if(isfield(p, MCD_INSTRUCTIONS))
        s1 = uint8((p.(MCD_INSTRUCTIONS))');
```

```
        nfields = typecast(s1(1:4), 'int32');
        prog.instrs = cell(prog.nUniqueInstrs+1, nfields);
        sizes = zeros(nfields, 1);
        types = cell(nfields, 1);

        j=5;
        for i = 1:nfields
            l = typecast(s1(j:j+3), 'int32'); % Get the length of the field name
            if(l > 10000)
                error('Memory overload.');
            end

            j = j+4;

            % Flags
            prog.instrs{1, i} = deblank(char(s1(j:j+l-2)));
            j = j+l;

            if(strncmp(prog.instrs{1, i}, 'instr_data', length('instr_data')))
                prog.instrs{1, i} = 'data';
            end

            if(strncmp(prog.instrs{1, i}, 'trigger_scan', length('trigger_scan')))
                prog.instrs{1, i} = 'scan';
            end

            if(strncmp(prog.instrs{1, i}, 'instr_time', length('instr_time')))
                prog.instrs{1, i} = 'time';
            end

            if(strncmp(prog.instrs{1, i}, 'time_units', length('time_units')))
                prog.instrs{1, i} = 'units';
            end

            type = typecast(s1(j), 'uint8');
            sizes(i) = fs_size(type);
            types{i} = fs_type(type);

            j = j+1;
        end


        units = {'ns', 'us', 'ms', 's'};
        for i=1:prog.nUniqueInstrs
            for k=1:nfields
                prog.instrs{i+1, k} = typecast(s1(j:(j+sizes(k)-1)), types{k});

                j = j+sizes(k);
            end

            prog.instrs{i+1, 5} = prog.instrs{i+1, 5}*10^(-double(prog.instrs{i+1, 6})*3);
            prog.instrs{i+1, 6} = units{prog.instrs{i+1, 6}+1};
        end
    end
else
    return;
end

if(isfield(prog, 'instrs'))
    prog.ps = parse_instructions(prog);
    p = prog;

    % If it's varied in indirect dimensions, read out the values.
```

```
    if(prog.nDims)
        vtype = zeros(prog.nDims, 1);

        if(isfield(prog, 'vinsdim'))
            ps = prog.ps;

            % Cell array along each dimension for each thing, also creates a
            % bool array determining if each dimension varies delay, data or
            % both.

            dels = {};
            datas = {};

            for d = 1:p.nDims
                ins = p.vins(p.vinsdim == d);
                vdata = zeros(p.maxsteps(d), length(ins));
                vdel = vdata;

                cind = num2cell(ones(size(size(ps.vinstrs))));

                for i = 1:p.maxsteps(d)
                    cind{d} = i;

                    for j = 1:length(ins)
                        k = ins(j);
                        vdata(i, j) = ps.vinstrs(cind{:}).data(k+1);
                        vdel(i, j) = ps.vinstrs(cind{:}).ts(k+1);
                    end
                end

                dels = [dels, {vdel}];
                datas = [datas, {vdata}];

                for i = 1:length(ins)
                    if(~isempty(find(vdel(1, i) ~= vdel(:, i), 1, 'first')))
                        vtype(d) = bitor(vtype(d), 1);
                    end

                    if(~isempty(find(vdata(1, i) ~= vdata(:, i), 1, 'first')))
                        vtype(d) = bitor(vtype(d), 2);
                    end
                end
            end

            prog.vdel = dels;
            prog.vdata = datas;
        end

       prog.vtypes = vtype;
    end
end

function [s, loc] = find_struct_by_field(in, name)
% Find a struct from the name of its field.
% Stops at the first one it finds. Breadth-first.

s = [];
loc = [];

flist = fieldnames(in);
num = find(strcmp(name, flist), 1, 'first');
```

```
if(~isempty(num))
    s = in.(flist{num});
    loc = [flist{num}];

    return;
end

for i = 1:length(flist)
    b = in.(flist{i});
    if(isstruct(b))
        [s, l] = find_struct_by_field(b, name);
        if(~isempty(s))
            loc = [flist{i} '.' l];
            break;
        end
    end
end

function s = parse_instructions(prog)
% Parse instructions into a meaningful structure. Must pass this something
% with a prog.instrs field.
%
% Inputs:
% prog: Pulse program type structure with prog.instrs cell array.
%
% Outputs:
% s:        A parsed structure with the cell array broken into structs and
%           full pulse sequences for each point in multidimensional sequences.
%           More machine-readable, less human-readable.
%
% Usage:
% s = parse_instructions(prog);

% This should define things like CONTINUE, STOP, etc.
CONTINUE = 0;       STOP = 1;           LOOP = 2;
END_LOOP = 3;       JSR = 4;            RTS = 5;
BRANCH = 6;         LONG_DELAY = 7;     WAIT = 8;

instrs = {'CONTINUE', ...
          'STOP', ...
          'LOOP', ...
          'END_LOOP', ...
          'JSR', ...
          'RTS', ...
          'BRANCH', ...
          'LONG_DELAY', ...
          'WAIT'};
u = struct('s', 1, ...
           'ms', 1000, ...
           'us', 1e6, ...
           'ns', 1e9);

p = prog;

s.ni = p.n_inst;

ib = zeros(s.ni, 1);
cb = {cell(s.ni, 1)};
cprog = struct('ni', s.ni, ...
               'tot_time', 0, ...
               'flags', ib, ...
               'instr', ib, ...
               'data', ib, ...
```

```
                'time', ib,...
                'units', cb,...
                'ts', ib, ...
                'un', ib, ...
                'instr_txt', cb);

% Parse the first version of the program
for i = 2:(s.ni+1)
    units = p.instrs{i, 6};
    un = u.(units);
    time = p.instrs{i, 5};
    ts = time/un;

    instr = p.instrs{i, 2};
    data = p.instrs{i, 3};

    j = i-1;

    cprog.flags(j) = p.instrs{i, 1};
    cprog.scan(j) = p.instrs{i, 4};
    cprog.instr(j) = instr;
    cprog.instr_txt{j} = instrs{instr+1};
    cprog.data(j) = data;
    cprog.time(j) = time;
    cprog.units{j} = {units};
    cprog.un(j) = un;
    cprog.ts(j) = ts;
end

% spans = find_loop_locs(cprog);
s.instrs = cprog;

% Generate a set of pulse program instructions for each step in the
% multi-dimensional space.
if(prog.varied && isfield(prog, 'vinslocs'))
    msteps = num2cell(p.maxsteps);
    s.msteps = msteps;
    p.vInstrs = repmat(cprog, msteps{:});
    vil = reshape(p.vinslocs, p.max_n_steps, p.nVaried);
    nv = p.nVaried;
    nis = p.max_n_steps;

    cs = msteps;
    for i = 1:nis
        [cs{:}] = ind2sub(p.maxsteps, i);
        for j = 1:nv
            k = vil(i, j)+2; % +1 for non-zero index, +1 for header.
            l = p.vins(j)+1;
            p.vInstrs(cs{:}).flags(l) = p.instrs{k, 1};
            p.vInstrs(cs{:}).instr(l) = p.instrs{k, 2};
            p.vInstrs(cs{:}).data(l) = p.instrs{k, 3};
            p.vInstrs(cs{:}).scan(l) = p.instrs{k, 4};
            p.vInstrs(cs{:}).time(l) = p.instrs{k, 5};
            p.vInstrs(cs{:}).units{l} = p.instrs{k, 6};
            units = p.instrs{k, 6};
            time = p.instrs{k, 5};
            un = u.(units);

            p.vInstrs(cs{:}).un(l) = un;
            p.vInstrs(cs{:}).ts(l) = time/un;
        end

        s.vinstrs = p.vInstrs;
```

```
    end

end

function o = fs_size(type)
% Gives the sizes of various types as per the FS file spec (MCD, PP), etc.
%
% Usage:
% o = fs_size(type);

% File types
FS_CHAR = 1;
FS_UCHAR = 2;
FS_INT = 3;
FS_UINT = 4;
FS_FLOAT = 5;
FS_DOUBLE = 6;
FS_INT64 = 7;
FS_UINT64 = 8;

if(type < 1 || type > 8)
    o = 1;
elseif(type == FS_CHAR || type == FS_UCHAR)
    o = 1;
elseif(type == FS_INT || type == FS_UINT || type == FS_FLOAT)
    o = 4;
elseif(type == FS_DOUBLE || type == FS_INT64 || type == FS_UINT64)
    o = 8;
end

function o = fs_type(type)
% Returns the type as parsed in FS file types, as a string that can be used
% for the matlab function "typecast"
%
% Usage:
% o = fs_type(type);

% File types
FS_CHAR = 1;
FS_UCHAR = 2;
FS_INT = 3;
FS_UINT = 4;
FS_FLOAT = 5;
FS_DOUBLE = 6;
FS_INT64 = 7;
FS_UINT64 = 8;

if(type < 1 || type > 8 || type == FS_CHAR)
    o = 'char';
elseif(type == FS_UCHAR)
    o = 'int8';
elseif(type == FS_INT)
    o = 'int32';
elseif(type == FS_UINT)
    o = 'uint32';
elseif(type == FS_FLOAT)
    o = 'float';
elseif(type == FS_DOUBLE)
    o = 'double';
elseif(type == FS_INT64)
    o = 'int64';
elseif(type == FS_UINT64)
    o = 'uint64';
```

```
end
```

D.1.1.4 find_loop_locs

```
function spans = find_loop_locs(instrs, span, top_lev)
% Pass this a struct containing instructions and it will find all loops.
%
% Inputs
% instrs:   A struct that contains parsed instructions as those given by
%               parse_instructions. This can be one of three things, a struct
%               with field ".prog.ps", a "ps" (parsed struct), or the field one
%               below (.instrs).
% span:     The span of instructions to search (of the form [ins1, ins2].
%               Uses a 1-based index. Defaults to the full span. Pass -1 for
%               default.
% top_lev:  Boolean, whether or not to search for ONLY top-level loops (no
%               nesting). Defaults to 0.
%
% Outputs:
% spans:        An n x 2 array where n is the number of loops, spans(:, 1) is
%               the start of each loop, spans(:, 2) is the end instr. Uses a
%               one-based index. Empty if there are no spans.
%
% Usage:
% spans = find_loop_locs(instrs, span, top_lev);

if(~isfield(instrs, 'flags'))
    if(isfield(instrs, 'prog') && isfield(instrs.prog, 'ps'))
        instrs = instrs.prog.ps.instrs;
    elseif(isfield(instrs, 'ps') && isfield(instrs.ps, 'instrs'))
        instrs = instrs.ps.instrs;
    elseif(isfield(instrs, 'instrs'))
        instrs = instrs.instrs;
    end
end

if(~exist('span', 'var') || isempty(span))
    if(isfield(instrs, 'ni'))
        span = [1, instrs.ni];
    else
        span = length(instrs.flags);
    end
end

if(isscalar(span))
    span = [span, instrs.ni];
end

if(~exist('top_lev', 'var'))
    top_lev = 0;
end

LOOP = 2;
END_LOOP = 3;

spans = [];

i = span(1);

ni = span(2);
```

```
while i <= span(2)
    if(instrs.instr(i) == LOOP)
        for j = (i+1):ni
            if(instrs.instr(j) == END_LOOP && instrs.data(j) == i-1)
                % Found it
                spans = [spans; i, j]; %#ok<AGROW>; This shouldn't be speed limiting.

                if(top_lev)
                    i = j;
                end
            end
        end
    end

    i = i+1;
end
```

D.1.2 Analysis

These are some scripts used for data analysis which operate the outputs of `mc_read_bin`.

D.1.2.1 `add_fft`

Requires: `process_data`

```
function out = add_fft(in, poly_order, cut, apod_lb)
% Give this a structure like those generated from the pulse controller,
% this will return a structure with ffts added to it.

if(~exist('poly_order', 'var'))
    poly_order = -1;
end

if(~exist('cut', 'var'))
    cut = -1;
end

if(~exist('apod_lb', 'var'))
    apod_lb = -0.5;
end

% Do the basic data processing.
if(poly_order >= 0 || cut > 0 && apod_lb > 0)
    out = process_data(in, poly_order, cut, apod_lb);
else
    out = in;
end

% Apply the fourier transform.
sr = out.prog.sr;
np_fft = 2^(ceil(log2(size(out.mdata, 1)))+1);

f = linspace(0, sr/2, np_fft/2);
s = fft(out.mdata(:, :), np_fft);
s = 2*s/size(out.mdata, 1);

sd = num2cell(size(out.mdata));

out.f = f;
```

```
out.fft = s(1:(np_fft/2), :);

if(length(sd) > 1)
    out.fft = reshape(out.fft, np_fft/2, sd{2:end});
end
```

D.1.2.2 process_data

Requires: sub_poly, apodize

```
function out = process_data(in, poly_order, cut, apod_lb)
% Does the most basic data processing - polynomial subtraction, truncation
% and zero-packing
%
% out = process_data(in[, poly_order, cut, apod_lb])
if(~exist('poly_order', 'var'))
    poly_order = -1;
end

if(~exist('cut', 'var'))
    cut = -1;
end

out = in;

% In case you want to make adjustments.
if(~isfield(out, 'odata'))
    out.odata = out.mdata; % Save the old data
    out.mdata = sub_poly(out.mdata, poly_order, 0, cut+1);
else
    if(cut >= 0 || poly_order >= 0)
        out.mdata = sub_poly(out.odata, poly_order, 0, cut+1);
    end
end

% Cut and zero pack
if(isscalar(cut) && cut > 1)
    out.mdata = circshift(out.mdata, -cut);
    out.mdata((end-cut+1):end, :) = 0;
end

% Apodize
if(exist('apod_lb', 'var') && apod_lb > 0)
    out.mdata = apodize(out.mdata, out.t, apod_lb);
end
```

D.1.2.3 sub_poly

```
function out = sub_poly(in, order, windowed, cut)
% Function that subtracts a polynomial offset from your data.
%
% Feed this a 1D or 2D matrix, tell it how many points you want to cut off, and
% it will fit those points to a polynomial and apply said polynomial to the
% output.
%
% If windowed evaluates to logical-true, then the first two dimensions will
% be reshaped before and after evaluation.
```

```
%
% Usage: out = sub_poly(in, order, windowed, cut);

if(~exist('windowed', 'var'))
    windowed = 0;
end

s_orig = size(in);
s_orig_cell = num2cell(s_orig);

if(windowed)
    if(length(s_orig) > 2)
        in = reshape(in, prod(s_orig(1:2)), s_orig_cell{3:end});
    elseif(length(s_orig)>1)
        in = reshape(in, prod(s_orig(1:2)), 1);
    end
end

s = size(in);

if(~exist('order', 'var') || order < 0)
    order = 1;
end

if(~exist('cut', 'var') || (isscalar(cut) && cut < 1) || isempty(cut))
    cut = 1;
end

if order > 99
    order = 99;
end

points = (1:s(1))'; % Initialize the "t" vector to have all the points

% Get the polynomial fits
if(isscalar(cut))
    p = cell2mat(arrayfun(@(x)polyfit(points(cut:end), ...
                                      in(cut:end, x), ...,
                                      order), ...
                          1:prod(s(2:end)), ...
                 'UniformOutput', false)');
else
    p = cell2mat(arrayfun(@(x)polyfit(points(cut), ...
                                      in(cut, x), ...
                                      order), ...
                          1:prod(s(2:end)), ...
                'UniformOutput', false)');
end

% Get the output array
out = cell2mat(arrayfun(@(x)in(:, x)-polyval(p(x, :), points), ...
                        1:prod(s(2:end)), ...
               'UniformOutput', false));

% Needs to be reshaped
if(windowed)
    if(length(s_orig) > 1)
        out = reshape(out, s_orig_cell{:});
    end
else
    s = num2cell(s);
    out = reshape(out, s{:});
end
```

D.1.2.4 apodize

```
function out = apodize(data, t, lb)
% Function that generates apodized data in 1 dimension
% data = data you want in
% t = time vector in seconds
% lb = line broadening in hz
%
% size(data, 1) must be the same size as t.
% Applies along only the first dimension.
%
% Usage: out = apodize(data, t, lb)

if(size(data, 1) ~= length(t))
   error('Incommensurate size between data and time vectors.');
end

if(size(t, 1) ~= length(t))
   t = t';
end

% Generate the exponentials
e_t = exp(-t/lb);

% Apply it to the data
data(:, :) = cell2mat(arrayfun(@(x)e_t.*data(:, x), ...
                      1:length(data(1, :)), ...
                      'UniformOutput', false));

out = data;
```