# 8.13-14 Experimental Physics I & II "Junior Lab"

Fall 2007 - Spring 2008

# Quantum Information Processing with NMR

MIT Department of Physics
(Dated: April 10, 2008)

This experiment will let you perform a series of simple quantum computations on a two spin system, demonstrating one and two quantum-bit quantum logic gates, and a circuit implementing the Deutsch-Jozsa quantum algorithm. You will use NMR techniques and manipulate the state of a proton and a carbon nucleus in a chloroform molecule, measuring ensemble nuclear magnetization. *WARNING: you should know Matlab well to successfully do this experiment!* You will measure: (1) the coupling constant describing the electron-mediated interaction between the proton and carbon nuclear spins of chloroform, (2) the classical input-output truth table for a controlled-NOT gate, (3) the numerical output of the Deutsch-Jozsa quantum algorithm, and (4) optionally, the output and oscillatory behavior of the Grover quantum search algorithm.

## 1. PREPARATORY QUESTIONS

1. The NMR spectrum of $^{13}$C chloroform has four peaks: two proton peaks (centered around $\sim$ 200 MHz in this experiment) and two carbon peaks (around $\sim$ 50 MHz). If the initial state of the system is described by a diagonal density matrix,

$$\rho = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{bmatrix} \tag{1}$$

(where the states are 00, 01, 10, and 11, with proton on left and carbon on right) then after a $R_x(\pi/2)$ readout pulse, the integrals of the two proton peaks (in the proton frequency spectrum) are given by $a-c$ and $b-d$, and the integrals of the two carbon peaks are given by $a-b$ and $c-d$. The spectrum is the fourier transform of the free induction decay signal, Eq.(22), for a given spin state $\rho$; see the Appendix for a detailed derivation. (a) Sketch the spectra you expect for the four cases when only one of $a$, $b$, $c$, and $d$ is nonzero. (b) What peak integrals do you expect for $\rho = \rho_{therm}$, the thermal state of Eq.(26)? (c) What are the spectra if $\rho$ is a CNOT gate applied to the thermal state, that is $\rho = U_{cn}\rho_{therm}U_{cn}^{\dagger}$? How about the near-CNOT?

2. Explicitly compute

$$U = \exp\left[i(n_x\sigma_x + n_y\sigma_y + n_z\sigma_z)\right]$$

and show this gives

$$U = \cos(|n|) + i\frac{n_x\sigma_x + n_y\sigma_y + n_z\sigma_z}{|n|}\sin(|n|)\,.$$

3. Give a sequence of pulses to implement a proper controlled-NOT, which has matrix elements of only one and zero, that is:

$$U_{cn} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

(up to an irrelevant overall phase). You may find it helpful to start with the near controlled-NOT operation of Eq.(8). You should need no more than four additional rotations about $\hat{x}$ and $\hat{y}$ axes, or just two rotations about $\hat{z}$ axes to transform the near-CNOT to a real CNOT. The Matlab script `qipgates.m` in the Junior Lab locker on the MIT Server, under `matlab/qip`, is very helpful for this problem.

4. Consider the four possible (classical) functions which have one bit as input and one bit as output.

   - State the function $f_k(x)$ for each of these functions (k from 1 to 4). Sketch a classical circuit which implements each of these functions. Sketch a quantum circuit which implements each of these functions using rotation operations and CNOT gates (the quantum circuit will consist of two qubits, an input 'x' and an output 'y', such that 'x' remains unchanged by the circuit and 'y' encodes the function $f_k(x) \oplus y$). Particularly useful references are [1, 2]
   - State the four pulse sequences which implement the above quantum circuits (denote them as $U_{f1}$, $U_{f2}$, $U_{f3}$, and $U_{f4}$) comprised of rotations about $\hat{x}$ and $\hat{y}$ axes, and delay times $\tau$ during which free two-qubit evolution occurs according to Eq.(4).
   - Compute the 4×4 unitary matrices $U_k$ corresponding to the four pulse sequences $U_k = R_{\bar{y}2}R_{y1}\,U_{fk}\,R_{y2}R_{\bar{y}1}$, for $k$ from 1 to 4.

5. Compute the one-sided fourier transform (integral from $t = 0$ to $t = +\infty$) of $e^{i(\omega_0 t+\phi)}e^{-t/T_2}$ and relate your result to the parameters for the Lorentzian in Eq.(23).

### Suggested schedule

Day 1: Measure $\omega_P$, $\omega_C$, and $J$, pulse widths for 90 degree rotations; estimate $T_1$, and $T_2$. Day 2: Implement near controlled-NOT gate and full controlled-NOT gate on

the thermal spectrum, and characterize its peak amplitudes. Day 3: Implement the Deutsch-Jozsa algorithm, and characterize sources of error. Day 4: (if you have time) Implement the Grover algorithm.

## 2. INTRODUCTION

Information always has a physical representation, and physical systems are governed by Laws of Nature. The earliest mechanical computers represented their state using the positions of gears and cogs; these computed by using Newton's laws of classical mechanics. Modern electronic computers store information using the presence or absence of electronic charge on small semiconductor capacitors, and use laws of electricity and magnetism to process information. *Quantum* computers represent information using states of quantum systems, and perform computation by exploiting the laws of quantum physics. Since quantum mechanics is capable of transformations which are impossible classically, new feats of computation are enabled which can speed the solution of some interesting mathematical problems; this potential was first foreseen by Richard Feynman, who pointed out in 1982[3] that quantum systems are difficult to simulate using classical computers. This is because the amount of information necessary to completely describe a quantum state grows exponentially in its size; the wavefunction of an $n$ spin-1/2 particle system is given by about $4^n$ real numbers. Feynman posed the question: how much computational power could be obtained by using the dynamics of quantum systems to solve classical problems?

Many early questions had to be answered before the potential for quantum information processing was to become evident. First was the thermodynamic cost of computation: for example, von Neumann believed it was necessary to dissipate $\approx k_B T$ of energy per elementary computational step[4]; however, it was then realized by Landauer[5] that energy dissipation is necessary only to *erase* information; in a closed (frictionless) computational system, there is no energy cost to computation! In part, von Neumann's belief came from the fact that gates such as the logical AND are irreversible; Bennett[6] then showed that all Boolean circuits can be made reversible with overhead (number of additional gates) only polynomial in the size of the original circuit. A whole field of reversible computation blossomed from this[7].

These insights brought quantum dynamics into compatibility with the idea of computation, since it was well known that closed quantum systems evolve unitarily, and the microscopic evolution of physical systems is reversible. Benioff[8] began by expressing classical computation as the dynamical behavior of a quantum system, and Feynman introduced various elementary quantum gates[9], but it was not until Deutsch's article in 1989[10] that a problem was found which could provably be solved faster than is possible with a classical computer. Essentially, this was the computation of $f(0) + f(1)$ in one evaluation of a function $f$ which has a domain and range of one bit: $\{0, 1\}$.

**Factoring & Search.** Deutsch's result languished in relative obscurity until Peter Shor surprised the world in 1994, with his discovery of a quantum algorithm for factoring integers[11]. This result extended a quantum period-finding algorithm by Simon[12] and utilized a number-theoretic result to turn a period of a certain function into factors of $L$-digit integers, using $O(L^3)$ quantum gates, in contrast to the $O(2^{L^{1/3}})$ operations required by the best known classical algorithm. Subsequently, in 1996 Grover[13] discovered a quantum search algorithm which can be used to speed up a wide range of problems from requiring $O(N)$ steps to only $O(\sqrt{N})$.

**Implementation.** Large-scale implementation of these algorithms remains a significant experimental challenge, but their basic working principles are now readily demonstrated using nuclear magnetic resonance (NMR) techniques applied to small multi-spin molecules. This methodology[1, 2] involves some new averaging techniques to extract a signal as if the spins were initialized to a zero-temperature state (in reality, they are in a high-temperature Boltzmann distribution). However, implementation of single quantum-bit (qubit) and multiple-qubit logic gates is straightforward, as well as is measurement of the final result. NMR has been successfully used to demonstrate the Deutsch-Jozsa, Grover, and Shor quantum algorithms, as well as quantum error correction and simulation of quantum systems[14, 15].

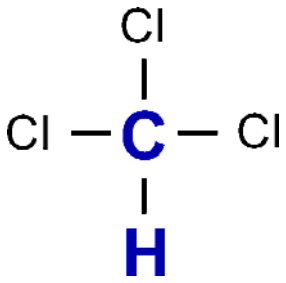


FIG. 1: Chloroform molecule.

In this lab, you will use a simple two-spin system comprised of the hydrogen (proton) and carbon nuclei of $^{13}$C labeled chloroform (Fig. 1) to implement a series of quantum logic gates, beginning with single qubit rotation operations, and a simple quantum addition gate. You will then use these building blocks to implement the Deutsch-Jozsa algorithm. Optionally, you may also implement Grover's quantum search algorithm and observe its expected oscillatory behavior.

## 3. THEORY: QUANTUM GATES, CIRCUITS, AND ALGORITHMS

Digital electronic computers are constructed from elementary building blocks known as *logic gates*, and the

digital quantum computers we study here are similarly constructed from *quantum logic gates*. These are nothing more than unitary transforms which act on finite-dimensional Hilbert spaces. The physical systems are comprised of numbers of *quantum bits*, two-level quantum systems, whose states are acted upon by a sequence of unitary transforms describing a *quantum circuit*. Finally, a projective measurement collapses the state, giving a probabilistic sequence of classical bits as output. The meaning of this output is dependent on the *quantum algorithm* which is instantiated by the circuit. These algorithms solve certain problems faster than is known to be possible with classical algorithms, because quantum logic gates provide additional new transforms, impossible classically.

We quickly tour this theoretical arena beginning with a description of the basic quantum logic gates, how they are composed to form circuits which implement two basic quantum algorithms, and finally how the circuits are implemented with NMR. More information can be found in the literature[16].

### 3.1. Quantum gates and circuits

**qubits**. A single qubit is a vector $|\psi_1\rangle = c_0|0\rangle + c_1|1\rangle$ parameterized by two complex numbers satisfying $|c_0|^2 + |c_1|^2 = 1$. Operations on a qubit must preserve this norm, and thus are described by 2×2 unitary matrices. Similarly, two-qubit states are described by four-dimensional vectors $|\psi_2\rangle = c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle$, satisfying $\sum_k |c_k|^2 = 1$, and transforms are 4×4 unitary matrices. In vector form, $|\psi_1\rangle = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}$, and similarly $|\psi_2\rangle = [c_{00}c_{01}c_{10}c_{11}]^T$.

**Measurement**. When measured in the *computational basis* of $|0\rangle$ and $|1\rangle$, the single qubit state $|\psi_1\rangle$ produces 0 and 1 with probability $|c_0|^2$ and $|c_1|^2$, respectively. The two-qubit state behaves similarly. Note that the overall phase of a wavefunction is unmeasurable and has no physical meaning, so that $e^{i\theta}|\psi_1\rangle$ is indistinguishable from $|\psi_1\rangle$ for any $\theta$. Thus, a single qubit is often visualized as a unit vector on a sphere (see the Appendix on Bloch sphere representations).

**Single qubit gates**. The Pauli matrices,

$$\sigma_x \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \sigma_y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad \sigma_z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2}$$

are important as *generators* of unitary transforms, and a good starting point for describing how a physical system performs quantum computation is to write down its Hamiltonian in terms of such matrices. A system with Hamiltonian $H$ evolves under the Schrödinger equation $i\hbar\partial_t|\psi\rangle = H|\psi\rangle$, which has the solution $|\psi(t)\rangle = U|\psi(t=0)\rangle$, where $U = e^{-iHt/\hbar}$ is unitary.

Quantum logic gates are realized by turning on and off terms in a Hamiltonian via an external control mechanism. For example, a single qubit may be realized by a two-level spin sitting in a static $\hat{z}$-oriented magnetic field, described by the Hamiltonian $H_1 = \hbar\omega_1\sigma_z + \hbar P_{x1}(t)\sigma_x/2 + \hbar P_{y1}(t)\sigma_y/2$, where $P_{x1}(t)$ and $P_{x2}(t)$ are classical variables (c-numbers) resulting from externally controlled magnetic fields about the $\hat{x}$ and $\hat{y}$-axes (see the Appendix on magnetic resonance).

When $P_{x1}$ is turned on, while $P_{y1} = 0$ (and let us assume for now we can neglect the $\hbar\omega_1$ term; this is done by moving into the rotating frame of the spin), the spin state transforms under $U = e^{-iP_{x1}t\sigma_x/2}$. For $P_{x1}t = \pi$, this gives $U = e^{-i\pi\sigma_x/2} = i\sigma_x$. It is easy to see this transform, known as a $\pi$ rotation about $\hat{x}$, is analogous to a classical NOT gate, since $\sigma_x|\psi_1\rangle = c_1|0\rangle + c_0|1\rangle$; the probability amplitudes of 0 and 1 are reversed. We denote this transform as $R_x(\pi)$. **You will implement such NOT gates to prepare inputs from the initial state $|00\rangle$.**

What happens when $P_{x1}t = \pi/2$? This is called a $\pi/2$ rotation, and results in the transform

$$R_x(\pi/2) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}, \tag{3}$$

which has no classical analogue! It might be thought of as a "square-root" of NOT, since performing this twice gives $R_x(\pi/2)R_x(\pi/2) = R_x(\pi)$, which we identified as a NOT. Similar rotations occur when $P_{y1}$ is nonzero. In general, we can perform rotations $R_x(\theta)$ and $R_y(\theta)$ for any value of $\theta$, by controlling $P_{x1}$ and $P_{y1}$ appropriately. Furthermore, by performing appropriate $\hat{x}$ and $\hat{y}$ rotations sequentially, we can construct rotations about the $\hat{z}$-axis, $R_z(\theta)$.

**Two qubit gates**. Now consider a two-qubit system described by the Hamiltonian

$$\begin{aligned} H &= \frac{\hbar\pi}{2} J\sigma_z^1\sigma_z^2 + \frac{\hbar P_{x1}(t)}{2}\sigma_x^1 + \frac{\hbar P_{y1}(t)}{2}\sigma_y^1 \\ &\quad + \frac{\hbar P_{x2}(t)}{2}\sigma_x^2 + \frac{\hbar P_{y2}(t)}{2}\sigma_y^2, \end{aligned} \tag{4}$$

where the superscripts describe which qubit the operators act upon, and $J$ is a fixed *coupling constant* describing a first-order spin-spin coupling. This kind of coupling is common in liquid-state NMR systems such as that we will be working with. As before, the $P(t)$ terms describe classical controls used to effect single-qubit operations. We assume that each of the $P(t)$ can be turned on separately, and with a magnitude such that $|P(t)| \gg J$, so that while any single qubit operation is active, the $J$-coupling can be neglected, to very good approximation. This will be the case in our experiment, and the above Hamiltonian will describe our system excellently.

How are single qubit operations on a two-qubit system described mathematically? For example, $R_{x1}$ denotes a $\pi/2$ rotation about $\hat{x}$ on qubit 1 (note that in this labguide, qubit 1 is the carbon nucleus; also we sometimes write $R_x$ as short for $R_x(\pi/2)$). This implies that we wish to do nothing, i.e. the *identity* operation, to qubit 2. To express "do nothing to the second qubit

and a $R_x$ to the first qubit" we write $R_{x1} = I \otimes R_{x1}$, where $\otimes$ is the *tensor product* (or sometimes, *Kronecker product*) operator. More about this, and the mathematics used for composite quantum systems, is described in the Appendix. With matlab, you can compute the matrix $R_{x1} = \texttt{kron}(I, R_x)$. Similarly, $\sigma_z^1\sigma_z^2$ is the matrix $\sigma_z \otimes \sigma_z$.

Consider the sequence of operations $U_{ncn} = R_{x1} - \tau - R_{\bar{y}1}$, where $\tau$ stands for a free evolution period (with all $P(t) = 0$) of time $1/2J$, and $R_{x1}$ and $R_{y1}$ are $\pi/2$ rotations acting on the first qubit. Note that this sequence is written with time going left to right, but when multiplying unitary transforms the first operator goes on the *right*. Using a Dirac ket labeling of $|c_2c_1\rangle$ (that is, the first qubit has its label on the right), and writing the matrix rows and columns in the natural numerical order of 00, 01, 10, 11 (left to right, and top to bottom), we obtain

$$\begin{aligned}
U_{ncn} &= R_{\bar{y}1}\tau R_{x1} && (5)\\
&= \left[I \otimes R_{\bar{y}}\right] e^{-i(\pi/4)\sigma_z\otimes\sigma_z} \left[I \otimes R_x\right] && (6)\\
&= \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 0 & 0\\ -1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & -1 & 1\end{bmatrix} \frac{1}{\sqrt{-i}}\begin{bmatrix} -i & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & -i\end{bmatrix}\\
&\quad \times \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -i & 0 & 0\\ -i & 1 & 0 & 0\\ 0 & 0 & 1 & -i\\ 0 & 0 & -i & 1\end{bmatrix} && (7)\\
&= \frac{1}{\sqrt{-i}}\begin{bmatrix} -i & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & -i\\ 0 & 0 & -1 & 0\end{bmatrix}. && (8)
\end{aligned}$$

This two-qubit gate is interesting: acting on each of the four computational basis input states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$, we obtain $U|00\rangle = -i|00\rangle$, $U|01\rangle = |01\rangle$, $U|10\rangle = -|11\rangle$, $U|11\rangle = -i|10\rangle$. If we denote the input as $|xy\rangle$, we find that the output from measuring $U|xy\rangle$ gives the classical bits $x'$ and $y'$ such that $x' = x$ and $y' = x \oplus y$, where $\oplus$ is binary addition modulo two ($0 \oplus 0 = 0, 0 \oplus 1 = 1 \oplus 0 = 1, 1 \oplus 1 = 0$). This quantum gate is thus an analog of a classical XOR gate. It is possible to insert further rotations about the $\hat{z}$-axes of the two qubits in order to make all the matrix elements equal to 1; the resulting operation is known as a *controlled*-NOT (or CNOT) gate, and the one given above, with non-ideal phases, is a "near" CNOT. **You will implement the quantum controlled-NOT gate and confirm its classical input-output truth table.**

In drawing these circuits graphically, let the top line be qubit 1, and the bottom qubit 2, such that an $\oplus$ on the bottom line represents $R_{x2}^2$, that is a $\pi$ pulse on qubit 2. The vertical line connecting a solid dot with an $\oplus$ is a symbol for a **cnot** gate.

### 3.2. Quantum algorithm: Deutsch-Jozsa

We are now ready to explore the first non-trivial quantum algorithm ever invented, the Deutsch-Jozsa algorithm, and solve the following problem: give a function $f(x)$ which accepts one bit as input and produces one bit as output, what is $f(0) \oplus f(1)$? Equivalently: give a coin, is it a fake (both sides have heads or tails), or fair (heads one one side, tails on the other)? Clearly, any classical computer must evaluate the function (look at both sides of the coin) at least twice to answer this question. On the other hand, a quantum computer need only evaluate $f$ once!

Here is how it works. We use two qubits, one to store the argument $x$, and the other to store the value of $f(x)$. In our basis this looks like $|x\ f(x)\rangle$. The two qubits are initialized in the state $|\psi_0\rangle = |00\rangle$. We then perform $U_1 = R_{y2}(\pi/2)R_{\bar{y}1}(\pi/2)$ to the state (note that $R_{\bar{y}1}(\pi/2) = R_{y1}(-\pi/2)$ is simply a rotation about the $-\hat{y}$ axes), obtaining

$$\begin{aligned}
|\psi_1\rangle &= U_1|\psi_0\rangle && (9)\\
&= \left[R_{y2}(\pi/2)|0\rangle\right] \otimes \left[R_{\bar{y}1}(\pi/2)|0\rangle\right] && (10)\\
&= \left[\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right] \otimes \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] && (11)\\
&= \frac{1}{2}\left[|00\rangle - |01\rangle + |10\rangle - |11\rangle\right]. && (12)
\end{aligned}$$

Now a unitary transform $U_f$ implementing $f$ is applied; this is defined to give $U_f|xy\rangle = |x\rangle|f(x) \oplus y\rangle$. (There are other ways $f$ could be implemented in such a reversible, unitary manner, but they are all equivalent for our purposes.) The result is the state $|\psi_2\rangle$,

$$\begin{aligned}
|\psi_2\rangle &= U_f|\psi_1\rangle && (13)\\
&= \frac{1}{2}\left[|0\rangle|f(0)\rangle - |0\rangle|f(0) \oplus 1\rangle\right.\\
&\qquad \left. + |1\rangle|f(1)\rangle - |1\rangle|f(1) \oplus 1\rangle\right] && (14)\\
&= \frac{1}{2}\left[(-1)^{f(0)}|0\rangle(|0\rangle - |1\rangle)\right.\\
&\qquad \left. + (-1)^{f(1)}|1\rangle(|0\rangle - |1\rangle)\right] && (15)\\
&= \left[\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right] \otimes \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] && (16)
\end{aligned}$$

The simplification from the second to third lines above occurs because $0 \oplus 1 = 1$, but $1 \oplus 1 = 0$. Note that the state of the first qubit (the one on the right) remains unchanged. We now apply two final single-qubit rotations, $U_2 = R_{\bar{y}2}(\pi/2)R_{y1}(\pi/2)$, giving $|\psi_3\rangle$. Recalling that $R_{\bar{y}}(\pi/2)|0\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$, and $R_{\bar{y}}(\pi/2)|1\rangle =$

$(|0\rangle + |1\rangle)/\sqrt{2}$,

$$|\psi_3\rangle = U_2|\psi_2\rangle \quad (17)$$

$$= \left[R_{\bar{y}2}(\pi/2)\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right] \otimes |0\rangle \quad (18)$$

$$= \left[\frac{(-1)^{f(0)}(|0\rangle - |1\rangle) + (-1)^{f(1)}(|0\rangle + |1\rangle)}{2}\right] \otimes |0\rangle \quad (19)$$

$$= \left[\frac{(-1)^{f(0)} + (-1)^{f(1)}}{2}|0\rangle + \frac{-(-1)^{f(0)} + (-1)^{f(1)}}{2}|1\rangle\right] \otimes |0\rangle\,. \quad (20)$$

Note how the first qubit (the one on the right) is left in its original state, $|0\rangle$. As for the second qubit (the one on the left): by inspection, we see that if $f(0) = f(1)$, then $f(0) \oplus f(1) = 0$ and measurement of the second qubit gives 0; otherwise $f(0) \oplus f(1) = 1$, and the measurement gives 1. This is the desired result.

Summarizing, the two-qubit Deutsch-Jozsa algorithm can be expressed as the sequence of operations

$$R_{\bar{y}2}R_{y1}\,U_{fk}\,R_{y2}R_{\bar{y}1} \quad (21)$$

acting on the initial state $|00\rangle$, where $U_{fk}$ implements the $k^{th}$ of the four possible functions $f_k(x)$, and each of the $R$'s denotes a $\pi/2$ rotation. These functions can be described by the truth tables

| $x$ | $f_1(x)$ | $f_2(x)$ | $f_3(x)$ | $f_4(x)$ |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |

In one evaluation of $f$, the quantum algorithm distinguishes between $k = \{1, 2\}$ and $k = \{3, 4\}$, whereas classically this would require two evaluations.

The action of this quantum circuit may be understood as having *interfered* two computational pathways to obtain the final answer. Indeed, the rotation operations are like beamsplitters, and the structure of the algorithm is that of a complex, four-path interferometer. A generalization of this structure is used in the quantum factoring algorithm. **You will construct quantum circuits for the four possible functions $U_f$, implement the Deutsch-Jozsa algorithm, and observe the measurement result.**

### 3.3. Quantum algorithm: Grover (optional)

In this lab, you may optionally implement the Grover quantum search algorithm on two qubits. The theory and implementation procedure for this optional part are described in the Appendix.

### 3.4. Implementation with NMR

This experiment utilizes an NMR apparatus which allows complex pulse sequences to be applied simultaneously at two different frequencies, to control and observe the proton and carbon spins inside molecules of $^{13}CHCl_3$. You are expected to have already completed Experiment #12, Spin Echos / Pulsed NMR, and know the basic physics of NMR. Below is some additional information specific to this quantum computation experiment.

**Magnetization readout**. The principal output of an experiment is the free induction decay (FID) signal $V(t)$ for spin $k$, mathematically given as

$$V(t) = -V_0 \mathrm{tr}\left[e^{-iHt}\rho e^{iHt}(i\sigma_x^k + \sigma_y^k)\right], \quad (22)$$

where $\sigma_x^k$ and $\sigma_y^k$ operate only on the $k$th spin, and $V_0$ is a constant factor dependent on coil geometry, quality factor, and maximum magnetic flux from the sample volume. This signal originates from the pickup coils detecting the magnetization of the sample in the $\hat{x}-\hat{y}$ plane. In the laboratory frame, this signal will oscillate at a frequency equal to the precession frequency $\omega_0$ of the nuclei; however, $V(t)$ is usually mixed down with an oscillator locked at $\omega_0$, then fourier transformed.

Notice how the voltage is *complex* valued; this is achieved using a superheterodyne receiver (much like that employed in the 21-cm Radio Astrophysics MIT Junior Lab experiment), and allows one to differentiate between a spin circulating clockwise or counterclockwise. The voltage signal also decays exponentially, as $e^{-t/T_2}$, so that the fourier transform signal of each spin resonance line is a complex-valued Lorentzian $g(\omega)$,

$$g(\omega) = \frac{\alpha\Gamma}{i(\omega - \omega_0) + \Gamma}\,, \quad (23)$$

where $\omega_0$ is the center frequency of the line, $\alpha = |\alpha|e^{i\arg\alpha}$ is the (complex-valued) height of the peak, and $2\Gamma$ is the full-width at half-max. You will read out $|0\rangle$ and $|1\rangle$ states (which are along the $\hat{z}$ axes) by tipping them into the $\hat{x} - \hat{y}$ plane, such that they are either $|+y\rangle$ or $|-y\rangle$ states. By convention, the local oscillator reference phase is set such that these states correspond to peaks which are positive, with $\arg\alpha = 0$, or negative, with $\arg\alpha = \pi$.

For a single (uncoupled) spin, the FID has only a single frequency. Two coupled spins will produce *four* frequencies; these occur as two pairs, in this experiment centered around the proton frequency ($\approx 200$ MHz) and the carbon frequency ($\approx 50$ MHz). The reason each spin's line is split into two is because of the coupling; as schematically shown in Fig. 2, we can think of what happens as one spin seeing the magnetic field of the other.

In chloroform, we observe that each line is split by $J \approx 215$ Hz, due to a Fermi-contact interaction mediated by electrons in the chemical bond between the carbon and proton. The spectra will appear much like the data shown in Fig. 3.

**The input state preparation problem**. *(This discussion requires knowledge of basic density matrices and statistical mechanics; it is optional (you can do the lab without fully understanding it) but essential to how the experiment works.)* NMR systems would be ideal for

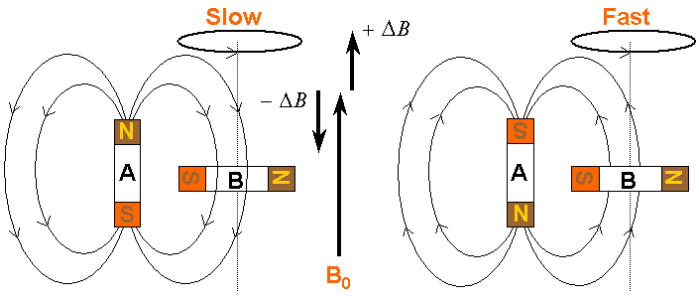


FIG. 2: Semiclassical explanation for spin-spin coupling, showing two spins in a strong magnetic field $B_0$. Each spin produces its own local magnetic field, which is seen by its neighbor. When spin A is spin-up, its field subtracts from that seen by B, thus causing it to precess slower. When A is spin-down, its field adds, causing B to precess faster. A is a quantum spin which only has two states.

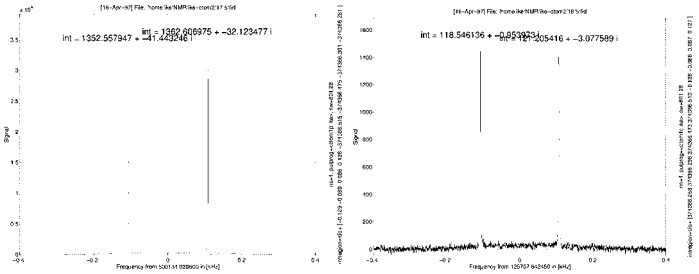

FIG. 3: Proton (left) and carbon (right) spectra of $^{13}CHCl_3$, showing $\approx 215$ Hz $J$ coupling.

quantum computation if not for one problem which particularly concerns this experiment: NMR is typically applied to physical systems in equilibrium at room temperature, where the spin energy $\hbar\omega$ is much less than $k_B T$. This means that the initial state of the spins is nearly completely random. Traditional quantum computation requires the system be prepared in a pure state (i.e. $|00\rangle$, as describe above); how can quantum computation be performed with a system which is in a high entropy mixed state?

Mathematically, we may write the initial state as the thermal equilibrium state (note our sign convention here),

$$\rho = \frac{e^{+\beta H}}{\mathcal{Z}}, \tag{24}$$

where $\beta = 1/k_B T$, and $\mathcal{Z} = \mathrm{tr}\, e^{+\beta H}$ is the usual partition function normalization, which ensures that $\mathrm{tr}(\rho) = 1$. Since $\beta \approx 10^{-4}$ at modest fields for typical nuclei at room temperature, the high temperature approximation

$$\rho \approx 2^{-n}\left[1 + \beta H\right] \tag{25}$$

is appropriate, for a system of $n$ spins. For the two-spin chloroform molecule, the initial state is approximately

$$\rho \approx \frac{I}{4} + 10^{-4}\begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & -3 & 0 \\ 0 & 0 & 0 & -5 \end{bmatrix}, \tag{26}$$

since $\omega_H \approx 4\omega_C$, by virtue of the approximate factor of four difference in the gyromagnetic factor for the proton compared with the carbon.

**Initialization solution: temporal labeling**. This experiment uses the following technique to extract a signal from only the $|00\rangle$ initial state; it is based on two important facts: quantum operations are linear, and the observables measured in NMR are traceless. Suppose a two spin system starts out with the density matrix

$$\rho_1 = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{bmatrix}, \tag{27}$$

where $a$, $b$, $c$, and $d$ are arbitrary positive numbers satisfying $a+b+c+d = 1$. We can use a circuit $P$ constructed from controlled-NOT gates to obtain a state with the permuted populations

$$\rho_2 = P\rho_1 P^\dagger = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & c & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & b \end{bmatrix}, \tag{28}$$

and similarly,

$$\rho_3 = P^\dagger \rho_1 P = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & b & 0 \\ 0 & 0 & 0 & c \end{bmatrix}. \tag{29}$$

A unitary quantum computation $U$ is applied to each of these states, to obtain (in three separate experiments, which may be performed at different times) $C_k = U\rho_k U^\dagger$. By linearity,

$$\sum_{k=1,2,3} C_k = \sum_k U\rho_k U^\dagger \tag{30}$$

$$= U\left[\sum_k \rho_k\right] U^\dagger \tag{31}$$

$$= (4a-1)U\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} U^\dagger + (1-a)\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{32}$$

In NMR, observables $M$ (such as Pauli $\sigma_x$ and $\sigma_y$) for which $\mathrm{tr}(M) = 0$ are the only ones ever measured; thus,

$$\mathrm{tr}\left(\sum_k C_k M\right) = \sum_k \mathrm{tr}\left(L C_k M\right) \tag{33}$$

$$= (4a-1)\,\mathrm{tr}\left(U \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} U^\dagger M\right) \tag{34}$$

$$= (4a-1)\,\mathrm{tr}\left(U|00\rangle\langle 00|U^\dagger\right). \tag{35}$$

We find that the sum of the measured signals from the three experiments gives us a result which is proportional to what we would have obtained had the original system been prepared in a pure state $|00\rangle\langle 00|$ instead of in the arbitrary state of Eq.(27).

States which are of the form $\rho = 2^{-n}(1-\epsilon)I + \epsilon U|00\ldots0\rangle\langle 00\ldots0|U^\dagger$ (where $U$ is any unitary operator), are called 'effective pure states', or 'pseudopure' states[14]. There are many strategies for preparing such states, and in general they all incur some cost. Effective pure states make it possible to observe zero temperature dynamics from a system which equilibrates to a high temperature state, as long as the coupling of the system to its high temperature environment is sufficiently small. This is the way it is used in NMR quantum computation.

## 4. EXPERIMENTAL APPARATUS

The experimental apparatus consists of a specially prepared chemical sample containing $^{13}CHCl_3$, a NMR spectrometer, and a control computer. A schematic of the Bruker acquisition system is shown in Figure 4.

You will be using a special Bruker Avance 200 NMR spectrometer located in Junior Lab for this experiment. It is primarily controlled by a Linux work-station running Bruker's xwin-nmr; you will interact with the spectrometer through the MIT Server workstation next to it, which speaks to xwin-nmr over the network, using matlab as your interface. Your main task is to design and implement sequences of pulses (a "pulse program") for the quantum circuits and algorithms described above. You will acquire data, compare with theoretical expectations, and explain likely sources of error and how they could be dealt with.

### 4.1. Sample

The 7% by weight $^{13}CHCl_3$ sample is prepared in a flame-sealed 5mm glass tube (Fig. 5) and mounted in a special "spinner" which allows the tube to be rapidly spun to average away transverse inhomogeneities in the $B_0$ axial field. The solvent is d6-acetone; the deuterium in the solvent is used to produce a lock signal which is picked up by the NMR probe and used in a active feedback loop to stabilize the axial magnetic field from drift.

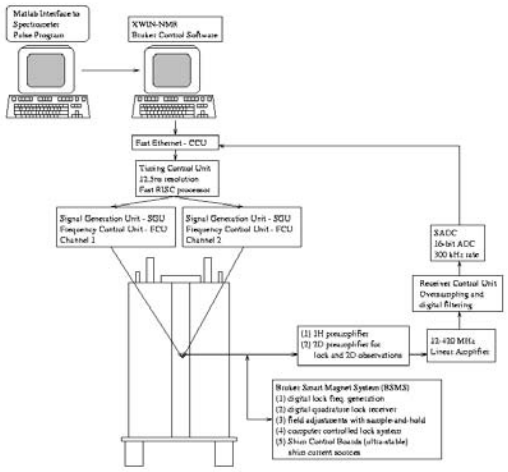


FIG. 4: An overview of the Bruker acquisition system.

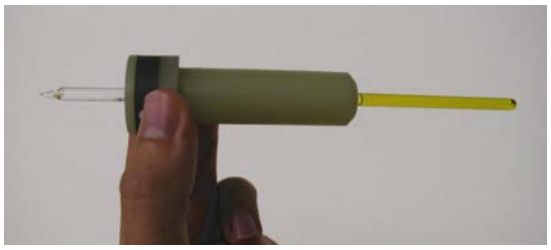

FIG. 5: Picture of a typical NMR sample in a spinner.

### 4.2. NMR Spectrometer hardware

The spectrometer is constructed from radiofrequency (RF) electronics and a large superconducting magnet with a proton NMR frequency at about 200 MHz, within the bore of which is held the sample in a glass tube, as shown in Figure 6. There, the static $\hat{z}$ oriented magnetic field $B_0$ is carefully trimmed to be uniform over approximately 1 cm$^3$ to better than one part in $10^9$. Orthogonal saddle coils lying in the transverse plane allow small, oscillating magnetic fields to be applied along the $\hat{x}$ and $\hat{y}$ directions. These fields can be rapidly pulsed on and off to manipulate nuclear spin states. The same coils are also part of tuned circuits which are used to pick up the RF signal generated by the precessing nuclei (much like how a spinning magnet inductively generates an alternating current in a nearby coil).

A typical experiment begins with a long waiting period in which the nuclei are allowed to thermalize to equilibrium; this is on the order of 60 seconds for our sample.

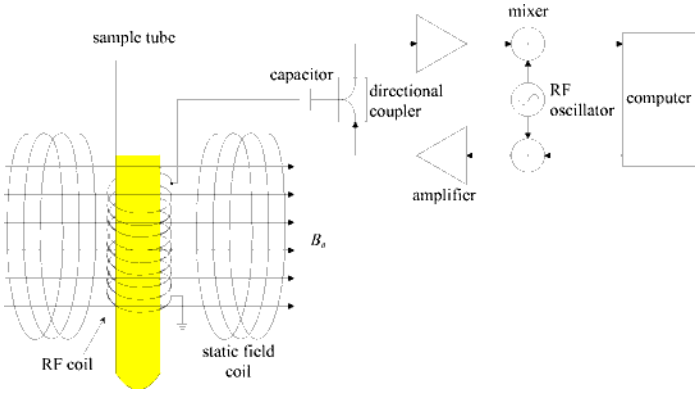


FIG. 6: Schematic diagram of an NMR apparatus.

Under control of the computer, RF pulses are then applied to effect the desired transformation on the state of the nuclei. The high power pulse amplifiers are then quickly switched off and a sensitive pre-amplifier is enabled, to measure the FID, which is then fourier transformed to obtain a frequency spectrum with peaks whose areas are functions of the spin states.

There are many important practical issues which lead to observable imperfections. For example, spatial inhomogeneities in the static magnetic field cause nuclei in different parts of the fields to precess at different frequencies. This broadens lines in the spectrum. An even more challenging problem is the homogeneity of the RF field, which is generated by a coil which must be orthogonal to the $B_0$ magnet; this geometric constraint and the requirement to simultaneously maintain high $B_0$ homogeneity usually forces the RF field to be inhomogeneous and generated by a small coil, leading to imperfect control of the nuclear system. Also, pulse timing, and stability of power, phase, and frequency are important issues.

### 4.3. Control system and software environment

Two linux computers are used in this experiment. The qip workstation is the main controller for the Bruker spectrometer; it will be setup in a server configuration for you, and you should not need to interact with it under normal operation. The main workstation used in this experiment is the MIT Server linux terminal. The MIT Server computer communicates with the other computer over the network to control the spectrometer and perform experiments. The main functions you will use are

- `NMRCalib(pw,phref,d1)` – applies one pulse of width `pw` to the proton and carbon channels, for calibration of the center frequencies, the $J$ coupling, the phase references, and the 90 degree pulse widths of the proton and carbon. A matlab structure containing the proton and carbon spectra, as specified below, is returned. The pulse performs a $R_x$ rotation. `d1` is the delay time to wait before starting the sequence.
- `NMRRunPulseProg(pw90,phref,pulses,phases, delays,tavgflag,nucflag,d1)` – runs a pulse program, as specified by `pulses`, `phases`, `delays` using the 90 degree pulse width `pw90` specified, returning a structure containing the proton and carbon spectra, phased according to `phref`, as described below. If `tavgflag` is 1 (that is the default, if this parameter is left out), then the averaging procedure for temporal labeling is performed; otherwise, no averaging is performed and just a single pair of spectra (one proton, one carbon) are taken. `nucflag` specifies whether both proton and carbon spectra are acquired (the default case), or just one. `d1` is the delay time to wait before starting the sequence.

The function arguments follow the following form:

- `pw` = pulse width (for `NMRCalib`) in [$\mu$s]
- `pw90` = 90-degree pulse width array (proton,carbon) in [$\mu$s]
- `d1` = delay time to wait before pulse sequence in [s]; the default value is 50 seconds, which should be long enough for the sample to re-equilibrate between excitations.
- `phref` = vector of two scalar elements $[\phi_p \phi_c]$ specifying the proton and carbon phase references, in units of [deg]. The spectra are multiplied by $\exp(i\phi_p)$ and $\exp(i\phi_c)$ by the program.
- `pulses` = $2\times N_p$ (two row, and $N_p$ column) matrix specifying what pulse widths to apply for proton and carbon. $N_p$ is the total number of pulses. The first row gives the proton pulse widths, and the second row gives the carbon pulse widths, in units of `pw90`. For example, [1 0; 0 1] describes a sequence of two 90-degree pulses, the first on the proton, and the second on the carbon.
- `phases` = $2\times N_p$ matrix specifying the phases of the proton and carbon pulses. The first row gives the proton phases, and the second row the carbon phases. Phases are 0, 1, 2, 3 for rotations about the $\hat{x}$, $\hat{y}$, $-\hat{x}$, and $-\hat{y}$ axes, respectively.
- `delays` = $1\times N_p$ vector specifying the delays, in [ms], to perform after each pulse. The first element in this vector specifies the delay after the first pulse, and so forth. Note that the delays vector, the pulses matrix, and the phases matrix must all have the same number of columns.
- `nucflag` = 0 is acquire both spectra (the default), 1 is proton only, 2 is carbon only. The spect structure that is returned will have only the corresponding elements set.

- `tavgflag` = 1 (the default) is to perform temporal labeling, 0 is no temporal labeling.

The proton and carbon spectra must be taken sequentially, one at a time, because the machine only has a single receiver. And if the averaging procedure for temperal labeling is performed, then this requires three experiments (with two spectra each) to be taken, so you have to be patient while the data is taken.

Each of these procedures returns (and saves) a data structure containing the proton and carbon spectra. Specifically, the returned or saved structure `spect` has fields:

- **spect.hfreq**: proton frequency scale data [Hz] - relative to hsfo
- **spect.cfreq**: carbon frequency scale data [Hz] - relative to csfo
- **spect.hspect**: proton spectral data phased according to hphase
- **spect.cspect**: carbon spectral data phased according to cphase
- **spect.cfid**: carbon free induction decay data
- **spect.hfid**: proton free induction decay data
- **spect.tacq** acquisition time [sec]
- **spect.hsfo**: proton transmitter frequency [MHz]
- **spect.csfo**: carbon transmitter frequency [MHz]
- **spect.dt**: date-time marker [string]
- **spect.hpeaks**: integrals of proton peaks phased according to hphase
- **spect.cpeaks**: integrals of carbon peaks phased according to cphase
- **spect.hphase**: phase value for proton spectra [radians]
- **spect.cphase**: phase value for carbon spectra [radians]
- **spect.pp**: pulse program structure containing `spect.pp.pulses`, `spect.pp.phases`, `spect.pp.delays`
- **spect.tavgflag**: flag indicating if temporal labeling was done [binary]
- **spect.nucflag**: flag indicating if one or both of the proton and carbon spectra were acquired [0/1/2]. Normally set to 1 or 2 but not 0.
- **spect.craw**: cell array of the three carbon spectra added together when temperal labeling is done
- **spect.hraw**: cell array of the three proton spectra added together when temporal labeling is done

These two procedures will also save the entire spect structure in a date-time stamped file (e.g., `spect-07Mar02-182304.mat`, i.e. `spect-date-time.mat`). You can load in this structure for later analysis.

## 5. OBSERVATIONS AND MEASUREMENTS

### 5.1. OVERVIEW

A typical experiment will involve the following steps:

1. Measure phase references, and $J$.
2. Measure the pulse widths required to implement $R_x(\pi/2)$ rotations on the proton and carbon spins.
3. Measure $T_1$ and $T_2$ of the sample.
4. Implement a controlled-NOT gate and measure its input-output truth table.
5. Implement the Deutsch-Jozsa quantum algorithm and test its four cases.
6. (optional) Implement the Grover algorithm, test its four cases, and observe the predicted theoretical oscillation.

### 5.2. DETAILED PROCEDURE & ANALYSIS

You will need an instructor or graduate TA to introduce you to the spectrometer during the first lab session. It is a very sensitive system and there are special safety precautions so please request help before you begin! Start by ensuring the Junior Lab chloroform sample is in the spectrometer, and taking a sample spectrum to make sure things are working normally. The technical staff may also need to *shim* the magnetic field to optimize its homogeneity. Each time you begin, you will want to quickly check your basic measurements (of the proton and carbon frequencies, and 90 degree pulse widths) again. Login to the MIT Server workstation, and then:

- **1.** Start `/usr/local/bin/matlab`
- **2.** `addpath /home/nmrqc/matlab`
- **3.** `NMRStartup`

You should now have all the proper Junior Lab QIP scripts in your path and can control the NMR machine. You may check that the network connection to the spectrometer is up by running the script `testnmrx`. If the connection is down, you will get the error message "do_nmr_command failed". In this case, you will

need to go to the Bruker console and run the command `netnmr2` inside xwin-nmr; a staff member can help with this. Aborting commands while communicating over the network may also require you to restart `netnmr2`.

#### 5.2.1. *Measurement of* `phref`, $\omega_P$, $\omega_C$, *and* $J$

Run `NMRCalib` with `pw`=5, and `phref`=[0 0]. The output is the structure `spect` (type 'help NMRCalib' for more) and 2 plots will show up, one for carbon and one for proton.

The data you obtain at first will have complex-valued peak integrals, because the phase reference of the receiver is not set properly (that is the role of `phref`). By plotting `spect.hspect` versus `spect.hfreq`, determine what value of $\phi$ you need, such that $e^{i\phi\pi/180} \times$ `spect.hspect` is real-valued (and appears as in Fig. 7). Do similarly for the carbon. Then run `NMRCalib` again with these two phase values in `phref`, and it will give you properly phased spectra. Note that you do not need an absolutely perfect phase setting for this experiment to work properly: having the imaginary component be $\langle 10\%$ of the real component is acceptable.

**Example:** The matlab commands

```
spect = NMRCalib(5,[10 34])
figure;
plot(real(spect.hfid));
figure;
plot(spect.hfreq,spect.hspect);
```

apply a 5 $\mu$s pulse to the proton, measure the proton and carbon spectra, and plot these in two figures. `10` and `34` are used as the phase reference angles; they should be chosen (by you) such that the spectra are real. The two plot commands plot the proton FID and the proton spectra. You can use `plot(spect.hfreq,exp(i*x)*spect.hspect)` to change the phase by `x` radians.

**Analysis:** Plot the free induction decay data (`spect.hfid` and `spect.cfid`) as functions of time (`spect.tacq` is the total acquisition time for each FID); this is the data that is fourier transformed to give the spectral data. Plot the absolute value of the spectral data (`spect.{hspect,cspect}`) versus frequency (`spect.{hfreq,cfreq}`), and note the four Lorentzian lines. The RF frequency of the carrier has to be set to exactly the center of the two split lines, for both the proton and carbon (this is how one works in the rotating frame). Be sure to write down the filenames in which your spectral data are saved, as you take them. Carefully measure the splitting between the lines, $J$, by fitting Lorentzians to the spectral data. $J$ should not change between experiments, and is not a function of the magnetic field strength, but the proton and carbon frequencies may drift slightly from session to session, and also depending on how well the sample is shimmed. When fitting Lorentzians to the lines to obtain the best center positions, refer back to Eq.(23).

The ideal thermal spectrum can be computed as the fourier transform of Eq.(22), using the thermal density matrix of Eq.(26) as the initial state $\rho$. It is useful to know that if the initial state is a diagonal density matrix $\rho = \mathrm{diag}(a, b, c, d)$, then after a $R_x(\pi/2)$ readout pulse, the peak integrals for the proton peaks are $a - c$ and $b - d$, and the carbon peaks are $a - b$ and $c - d$. For a voltage scale reference of $V_0 = 1$, the integrals of the two peaks in the proton spectrum should be $+8$ and $+8$, while the carbon spectrum should have integrals $+2$ and $+2$, as shown in Fig. 7. **Write down the four peak integrals from your real-valued spectra as your reference thermal spectrum peak integral values and assign their values to the theoretical ratios (+8 or +2.**

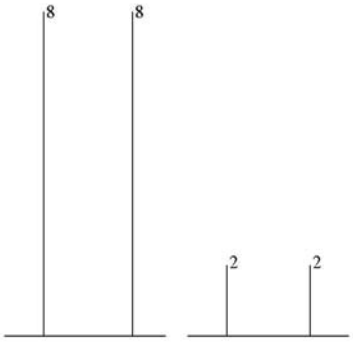


FIG. 7: Thermal spectra for proton (left) and carbon (right): stick figures with nominal peak integrals.

You must be careful of the phase of the signals in this lab, and be aware of how the phase is referenced. Ideally, we wish a 90 degree pulse about $\hat{x}$ to result in a real-valued peak (i.e. a Lorentzian of Eq.(23) with $\arg\alpha = 0$). Always be sure that a single $R_x(\pi/2)$ pulse indeed gives a spectrum with no net imaginary component! From studying Eq.(23), you should note that because of $\Gamma$ being nonzero, you will always see some imaginary spectral component, but the integrated area of the peak will be zero. The phase tells us the relative phase of the $|0\rangle$ and $|1\rangle$ components of the proton and carbon qubits; equivalently, this can be visualized as their point along the equator of the Bloch sphere. The qubits start along $+\hat{z}$, and then get tipped by a $R_x(\pi/2)$ onto $+\hat{y}$, and by convention we choose that to give us a real valued peak. Thus, a $R_y(\pi/2)$ pulse, which tips $+\hat{z}$ onto $+\hat{x}$ should give us a purely imaginary valued peak; you may check that this is true using `NMRRunPulseProg`.

### 5.2.2. *Calibration of* 90 *degree pulse widths*

Run `NMRCalib` with a set of different pulse widths to determine the number of microseconds required for a $\pi/2$ pulse on the carbon and $\pi/2$ on the proton. Note that they are different, not in the least because the two channels use different RF power amplifiers (why else might they be different?). The best estimate comes from taking a regularly spaced set of pulse widths, and fitting an exponentially damped sinusoid to the points given by peak height versus pulse width (why does the sinusoid exponentially decay?). **On a first pass, do not spend too much time getting your calibrations absolutely perfect; it is important that you get through to finishing the lab.**

**Example:** The matlab commands

```
peaks = [];
pwtab = linspace(1,15,8);
for pw = pwtab
  sd = NMRCalib(pw,[10 34]);
  peaks = [peaks ; sd.hpeaks sd.cpeaks]
end
figure;
plot(pwtab,real(peaks(:,1)));
figure;
plot(pwtab,real(peaks(:,3)));
```

will measure the proton and carbon spectra for 8 different peak widths from 1 to 15 microseconds, and plot the real part of the left peak integrals as a function of pulse width. `[10 34]` is a vector which specifies the phase references; be sure to replace this with your own values. Note that each signal acquisition takes $\approx$ 50 seconds (by default), so the script as given above will take $\approx 50 * 8 * 2 = 13$ minutes.

**Analysis:** Take the peak integrals from each of the experiments and plot them as a function of the pulse widths. (You should compare the peak integrals given by the program to what you get from Lorentzian linefits.) Fit an exponentially decaying sinusoid to the proton data and to the carbon data, and from this determine the optimal 90 degree pulse widths for the two nuclei. What sources of error contribute to your pulse width calibrations?

### 5.2.3. *Measurement of* $T_1$ *and* $T_2$

$T_1$ is measured using a 180 – $\Delta t$ – 90 "inversion-recovery" pulse sequence, and fitting an exponential decay to the resulting data of peak height versus $\Delta t$. Use `NMRRunPulseProg` for this; construct a pulse program with appropriate pulse widths, phases, and delays to perform the experiment. **Be aware that each FID acquisition takes 50 seconds or more, because the system must re-equilibrate between excitations. Do not setup an experiment that will take too long to complete**. The purpose of this part is to familiarize yourself with the system, and to put a lower bound on $T_1$ and $T_2$ so that you can convince yourself that the quantum information processing part of the experiment should work.

**Example:** The matlab commands

```
peaks = [];
dtab = linspace(1,8000,5);
pulses=[2 1;0 0]
phases=[0 0 ;0 0]
nucflag = 0
for delay = dtab
  sd = NMRRunPulseProg([7.3 5.5],[10 34],
     pulses,phases,[delay 0],0,nucflag);
  peaks = [peaks ; sd.hpeaks ]
end
figure;
plot(dtab,real(peaks(:,1)));
```

will perform an inversion recovery $T_1$ measurement on the proton, taking 5 datapoints with delays from 1 millisecond to 8 seconds. Note that the units of delay is **milliseconds**. `pulses` specifies that the first pulse should be a $\pi$ pulse on the proton, followed by a $\pi/2$ pulse on the proton. `phases` specifies that all the pulses are along $+\hat{x}$. Note that `[7.3 5.5]` give the $\pi/2$ pulse widths for proton and carbon, in this example, and should be replaced with your actual measured values; similarly for the phase reference values.

**Analysis:** Measure data to calculate $T_1$, for both the proton and carbon. Be warned that $T_1$ is on the order of 1 to 30 seconds, and you must wait $\geq 5T_1$ between spectra for the sample to re-equilibrate. Also estimate $T_2$ from the linewidths, obtained by fitting the Lorentzians (and using your answer to prep question #5). Why does the complex amplitude of the Lorentzians vary as they do when you change $\Delta t$, particularly on the time scale of 2 to 8 milliseconds?

### 5.2.4. *Characterization of controlled-*NOT *and its truth table*

Write a pulse sequence for `NMRRunPulseProg` to implement the "near"controlled-NOT gate (with strange phases), $U_{ncn} = R_{\bar{y}1}\tau R_{x1}$, where $\tau$ is a $1/2J$ delay, and also your "real" controlled-NOT gate (which you obtained as an answer to prep question #3), $U_{cn}$.

1. Apply $U_{ncn}$ and a readout pulse ($R_x(\pi/2)$) to the thermal state, and take both proton and carbon spectra; there should be only one readout pulse and it should correspond to the nucleus you are acquiring data for. (What should these spectra look like?)

2. Setup single and two-pulse sequences to prepare all four possible classical inputs (00, 01, 10, 11), using

temporal labeling to prepare the initial 00. Then apply $U$ and a readout pulse, to confirm the classical truth table of the CNOT.

3. Repeat the two above steps for your own CNOT pulse sequence, $U_{cn}$.

You will need to have a proper $U_{cn}$ to proceed to the next step of this lab!

**Example:** The matlab commands

```
pulses=[0 0 1;1 1 0]
phases=[0 0 0;0 3 0]
delays=[1/2/J 0 0]
tavgflag = 0
nucflag = 1
sd = NMRRunPulseProg([7.3 5.5],[10 34],
       pulses,phases,delays,tavgflag,nucflag);
```

applies a near-CNOT gate with one readout pulse on the proton.

The first row of `pulses` specifies that only one pulse should be applied to the proton, and this is a $\pi/2$ pulse in the third time interval. The second row of `pulses` specifies that $\pi/2$ pulses are applied to the carbon, in the first and second time intervals. The `phases` matrix specifies that the proton pulse should be about $\hat{x}$ (since `phases(1,3)=0`), and the carbon pulses should be about $\hat{x}$ and $-\hat{y}$. And the delays vector specifies that a delay of $1/2J$ (*in milliseconds*!) occurs after the first pulse is applied; this implements the $\tau$ free-evolution unitary operator involved in the construction of the near-CNOT of Eq.(5).

To obtain a similar spectrum, but for the carbon, you will need to set `nucflag=2` and move the readout pulse to the carbon. `tavgflag=0` tells the machine not to do temporal averaging, so that it applies the pulse sequence to a thermal spectrum. To apply the pulse sequence to the state $|00\rangle$, obtained by temporal averaging, simply set `tavgflag=1`. To apply the sequence to other pure inputs, such as $|01\rangle$, you will need to add additional pulses.

**Analysis:** Write down the four peak integrals you obtain for the near-CNOT applied to the thermal spectrum. Refering back to prep question #1, and the four reference peak integrals you obtained in the first part of this lab, **compare your peak integrals to make sure you have the right ratios**. That is, let $P_{H1}$, $P_{H2}$, $P_{C1}$, $P_{C2}$ be the reference peak integrals, and $P^{cn}_{H1}$ etc. be the peak integrals for the CNOT applied to the thermal state. Compute $P^{cn}_{H1}/P_{H1}$, $P^{cn}_{H2}/P_{H2}$, $P^{cn}_{C1}/P_{C1}$, $P^{cn}_{C2}/P_{C2}$, and compare $P_{H1}/P_{C1}$ versus $P^{cn}_{H1}/P^{cn}_{C1}$. If you see the right peak integral ratios, then you have successfully implemented a logic operation on two nuclear spins – your first simple quantum computation!

Be sure to estimate your errors - where do they come from, and are your results within a standard deviation of the ideal theoretical prediction? Also verify that you are able to create the 00, 01, 10, and 11 states by acquiring corresponding spectra; note that a 0 is a positive real peak, and a 1 is a negative real peak. How close is your output to the theoretical ideal?

Show that for these inputs (the thermal state and the computational basis states) the proton and carbon spectra should be the same whether you implement the ideal CNOT, with all the proper phases, or the "near" CNOT, on the thermal state (or indeed, on any classical input state).

If the spins are in a computational basis state (i.e. 00, 01, 10, or 11), then readout of the proton (or carbon) spectrum alone actually gives full information about the state; only one of the two spectra need be acquired. How does one obtain the state from one spectrum, and why is this possible?

### *5.2.5. Implementation of Deutsch-Jozsa quantum algorithm*

The Deutsch-Jozsa algorithm has four possible cases, corresponding to the four possible functions mapping one bit to one bit. Let these be $f_1$, $f_2$, $f_3$, and $f_4$. In prep question #4, you found pulse sequences to implement these functions. As a hint: $U_{f2}$ needs just one pulse, $U_{f3} = U_{cn}$, and $U_{f4}$ is a simple modification of $U_{f3}$. Put these together with the theoretical discussion above sequences implementing the full algorithm for four cases. Specifically, translate Eq.(21) into pulses. You should be able to fit the sequences into at most eight pulses on each spin. For example, for $k = 1$ the pulse sequence is simply $R_{\bar{y}2}R_{y1}R_{y2}R_{\bar{y}1}$, since $U_{f1}$ is the identity operation.

Write pulse programs for all these four cases, adding readout pulse, and run `NMRRunPulseProg` to acquire the data. It is helpful to translate your pulses into matrices and confirm theoretically that their product gives the expected theoretical result, and matches your experimental observations.

**Analysis:** Be sure to set your phase reference `phref` properly (as in the calibration step above) such that plotting the reference thermal spectra gives you a positive set of peak parameters $\alpha$. With the same `phref`, you should find that all the output spectra from the Deutsch-Jozsa experiments give real-valued peak integrals, if all your gates (and in particular, your CNOT gate) are correct.

You should obtain the same spectra for the $k = 1$ and $k = 2$ cases, and a different spectrum for the $k = 3$ and $k = 4$ cases. Verify that this is the case. Note that just as for the controlled-NOT gate experiment, only one of the two (proton and carbon) spectra are needed to fully distinguish the outputs of the two-qubit Deutsch-Jozsa algorithm.

What sources of error contribute to getting non-ideal spectra? **Compute the theoretically expected spectra, and propagate errors from all your measurements and calibrations to obtain an estimate of how far you expect to be off from the ideal; are you within one standard deviation, and if not,**

**why?**

Further optional topics: Can you identify where the quantum behavior of the spin system is uniquely required for this algorithm to work? What choice of assignment of qubits to spins did you use, and what happens if you reverse this?

## 6. SUGGESTED ADVANCED TOPICS

### 6.1. NMR techniques for quantum computation with other physical systems

The techniques you learned in this lab for quantum computation with NMR are actually extremely similar to what is employed to perform quantum computations with other atomic, molecular, and optical implementations. For example, the Deutsch-Jozsa algorithm was recently implemented[17] on a trapped Ca ion, using pulses of laser light similar to the RF pulses used in this lab. Investigate how quantum bits are realized using trapped ions or superconductors and describe how pulsed excitations implement quantum computations in those systems.

### 6.2. Entangled states in NMR

One of the most intriguing aspects of quantum systems is a property known as *entanglement*; it is widely believed that entanglement is necessary to achieve any significant computational speedup (or nontrivial cryptographic protocol with qubits) – although this necessity remains unproven in general. By definition, a pure state of two systems $A$ and $B$ in a state $|\psi_{AB}\rangle$ is entangled if and only if it cannot be written as a tensor product, that is, there does not exist $|\psi_A\rangle$ and $|\psi_B\rangle$ such that $|\psi_{AB}\rangle = |\psi_A\rangle \otimes |\psi_B\rangle$. Of such entangled states, the most well studied are the four two-qubit states

$$|\Psi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad (36)$$

$$|\Psi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}} \quad (37)$$

$$|\Phi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \quad (38)$$

$$|\Phi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}} \quad (39)$$

Give quantum circuits for creating these four states from the initial state $|00\rangle$, describe how to implement this on the two-qubit chloroform system used in this lab, and explain what the measured output signal should be.


[1] D. G. Cory, A. F. Fahmy, and T. F. Havel, Proc. Nat. Acad. Sci. USA **94**, 1634 (1997).
[2] N. Gershenfeld and I. L. Chuang, Science **275**, 350 (1997).
[3] R. P. Feynman, Int. J. Theor. Phys. **21**, 467 (1982).
[4] H. Leff and R. Rex, *Maxwell's Demon: Entropy, Information, Computing* (Princeton University Press, Princeton, NJ, 1990).
[5] R. Landauer, IBM J. Res. Dev. **5**, 183 (1961).
[6] C. H. Bennett, IBM J. Res. Dev. **17**, 525 (1973).
[7] E. Fredkin and T. Toffoli, Int. J. Theor. Phys. **21**, 219 (1982).
[8] P. Benioff, J. Stat. Phys. **22**, 563 (1980).
[9] R. P. Feynman, Optics News p. 11 (1985).
[10] D. Deutsch, Proc. R. Soc. London A **425**, 73 (1989).
[11] P. W. Shor, in *Proceedings, $35^{th}$ Annual Symposium on Foundations of Computer Science* (IEEE Press, Los Alamitos, CA, 1994), pp. 124–134.
[12] D. Simon, in *Proceedings, $35^{th}$ Annual Symposium on Foundations of Computer Science* (IEEE Press, Los Alamitos, CA, 1994), pp. 116–123.
[13] L. K. Grover, in *28th ACM Symposium on Theory of Computation* (Association for Computing Machinery, New York, 1996), p. 212.
[14] D. Cory, R. Laflamme, E. Knill, L. Viola, T. Havel, N. Boulant, G. Boutis, E. Fortunato, S. Lloyd, R. Martinez, et al., Fortschr. Phys. **48**, 875 (2000).
[15] L. Vandersypen, C. Yannoni, and I. Chuang, in *to appear in The encyclopedia of NMR (supplement)*, edited by D. Grant and R. Harris (John Wiley and Sons, West Sussex, England, 2001).
[16] M. Nielsen and I. Chuang, *Quantum computation and quantum information* (Cambridge University Press, Cambridge, England, 2000).
[17] Guilde, Riebe, Lancaster, Becher, Eschner, Haffner, Schmidt-Kaler, Chuang, and Blatt, Nature **421**, 48 (2003).


## APPENDIX A: BLOCH SPHERE REPRESENTATION OF A SINGLE QUBIT

Recall that a single qubit in the state $a|0\rangle + b|1\rangle$ can be visualized as a point $(\theta, \phi)$ on the unit sphere, where $a = \cos(\theta/2)$, $b = e^{i\phi}\sin(\theta/2)$, and $a$ can be taken to be real because the overall phase of the state is unobservable. This is called the Bloch sphere representation, and the vector $(\cos\phi\sin\theta, \sin\phi\sin\theta, \cos\theta)$ is called the Bloch vector.

The Pauli matrices give rise to three useful classes of unitary matrices when they are exponentiated, the *rotation operators* about the $\hat{x}$, $\hat{y}$, and $\hat{z}$ axes, defined by the

equations:

$$R_x(\theta) \equiv e^{-i\theta X/2} = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} X = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \quad \text{(A1)}$$

$$R_y(\theta) \equiv e^{-i\theta Y/2} = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \quad \text{(A2)}$$

$$R_z(\theta) \equiv e^{-i\theta Z/2} = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} Z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \quad \text{(A3)}$$

One reason why the $R_{\hat{n}}(\theta)$ operators are referred to as rotation operators is the following fact. Suppose a single qubit has a state represented by the Bloch vector $\vec{\lambda}$. Then the effect of the rotation $R_{\hat{n}}(\theta)$ on the state is to rotate it by an angle $\theta$ about the $\hat{n}$ axis of the Bloch sphere. This explains the rather mysterious looking factor of two in the definition of the rotation matrices.

An arbitrary unitary operator on a single qubit can be written in many ways as a combination of rotations, together with global phase shifts on the qubit. A useful theorem to remember is the following: Suppose $U$ is a unitary operation on a single qubit. Then there exist real numbers $\alpha, \beta, \gamma$ and $\delta$ such that

$$U = e^{i\alpha} R_x(\beta) R_y(\gamma) R_x(\delta). \quad \text{(A4)}$$

## APPENDIX B: FUNDAMENTAL EQUATIONS OF MAGNETIC RESONANCE

The magnetic interaction of a classical electromagnetic field with a two-state spin is described by the Hamiltonian $H = -\vec{\mu} \cdot \vec{B}$, where $\vec{\mu}$ is the spin, and $B = B_0\hat{z} + B_1(\hat{x}\cos\omega t + \hat{y}\sin\omega t)$ is a typical applied magnetic field. $B_0$ is static and very large, and $B_1$ is usually time varying and several orders of magnitude smaller than $B_0$ in strength, so that perturbation theory is traditionally employed to study this system. However, the Schrödinger equation for this system can be solved straightforwardly without perturbation theory, in terms of which the Hamiltonian can be written as

$$H = \frac{\omega_0}{2} Z + g(X\cos\omega t + Y \sin\omega t), \quad \text{(B1)}$$

where $g$ is related to the strength of the $B_1$ field, and $\omega_0$ to $B_0$, and $X, Y, Z$ are the Pauli matrices as usual. Define $|\phi(t)\rangle = e^{i\omega t Z/2}|\chi(t)\rangle$, such that the Schrödinger equation

$$i\partial_t|\chi(t)\rangle = H|\chi(t)\rangle \quad \text{(B2)}$$

can be re-expressed as

$$i\partial_t|\phi(t)\rangle = \left[e^{i\omega Zt/2} H e^{-i\omega Zt/2} - \frac{\omega}{2} Z\right]|\phi(t)\rangle. \quad \text{(B3)}$$

Since

$$e^{i\omega Zt/2} X e^{-i\omega Zt/2} = (X\cos\omega t - Y\sin\omega t), \quad \text{(B4)}$$

(B3) simplifies to become

$$i\partial_t|\phi(t)\rangle = \left[\frac{\omega_0 - \omega}{2} Z + gX\right]|\phi(t)\rangle, \quad \text{(B5)}$$

where the terms on the right multiplying the state can be identified as the effective 'rotating frame' Hamiltonian. The solution to this equation is

$$|\phi(t)\rangle = e^{i\left[\frac{\omega_0-\omega}{2} Z + gX\right]t}|\phi(0)\rangle. \quad \text{(B6)}$$

The concept of *resonance* arises from the behavior of this solution, which can be understood to be a single qubit rotation about the axis

$$\hat{n} = \frac{\hat{z} + \frac{2g}{\omega_0-\omega}\hat{x}}{\sqrt{1 + \left(\frac{2g}{\omega_0-\omega}\right)^2}} \quad \text{(B7)}$$

by an angle

$$|\vec{n}| = t\sqrt{\left(\frac{\omega_0-\omega}{2}\right)^2 + g^2}. \quad \text{(B8)}$$

When $\omega$ is far from $\omega_0$, the spin is negligibly affected by the $B_1$ field; the axis of its rotation is nearly parallel with $\hat{z}$, and its time evolution is nearly exactly that of the free $B_0$ Hamiltonian. On the other hand, when $\omega_0 \approx \omega$, the $B_0$ contribution becomes negligible, and a small $B_1$ field can cause large changes in the state, corresponding to rotations about the $\hat{x}$ axis. The enormous effect a small perturbation can have on the spin system, when tuned to the appropriate frequency, is responsible for the 'resonance' in nuclear magnetic resonance.

In general, when $\omega = \omega_0$, the single spin rotating frame Hamiltonian can be written as

$$H = g_1(t)X + g_2(t)Y, \quad \text{(B9)}$$

where $g_1$ and $g_2$ are functions of the applied transverse RF fields.

## APPENDIX C: STATE TOMOGRAPHY

How does one debug a quantum computer? A classical computer is analyzed by measuring its internal state at different points in time. Analogously, for a quantum computer, an essential technique is the ability to measure its density matrix – this is called *state tomography*.

Recall that the density matrix of a single qubit can be expressed as

$$\rho = \frac{1}{2}\left[1 + \sum_k r_k \sigma_k\right], \tag{C1}$$

where $\sigma_k$ are the Pauli matrices and $r_b$ is a real, three-component vector. Because of the trace orthogonality property of Pauli matrices,

$$\mathrm{tr}(\sigma_k \sigma_j) = 2\delta_{kj}\,, \tag{C2}$$

it follows that $\rho$ can be reconstructed from the three measurement results

$$r_k = \langle \sigma_k \rangle = \mathrm{tr}(\rho \sigma_k)\,. \tag{C3}$$

Measurement of the usual observable in NMR, (22), preceded by the appropriate single qubit pulses, allows us to determine $\langle \sigma_k \rangle$, and thus obtain $\rho$. Similar results hold for larger numbers of spins. In practice, it is convenient to measure just the traceless deviation of $\rho$ from the identity; this is called the deviation density matrix.

## APPENDIX D: QUANTUM ALGORITHM: GROVER (OPTIONAL)

In this lab, you may optionally implement the Grover quantum search algorithm on two qubits.

### 1. Theory

Another quantum algorithm which solves a problem faster than is possible classically is Grover's quantum search algorithm: for a problem size of four elements ($n = 2$ qubits), we are given the set $x = \{0, 1, 2, 3\}$ for which $f(x) = 0$ except at one value $x_0$, where $f(x_0) = 1$. The goal is to find $x_0$, which can be classically accomplished by evaluating $f(x)$ an average of 2.25 times. In comparison, the quantum algorithm finds $x_0$ by evaluating $f(x)$ only once.

Three operators are required: the oracle operator $O$ (which performs a phase flip based on the function $f(x)$), the Hadamard operator on two qubits $H^{\otimes 2}$, and the conditional phase shift operator $P$. The oracle $O$ flips the sign of the basis element corresponding to $x_0$; for $x_0 = 3$, this is

$$O = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \tag{D1}$$

Denoting free evolution period of time $1/2J$ as $\tau$, we find that $O = R_{y1}R_{x1}R_{\bar{y}1}R_{y2}R_{x2}R_{\bar{y}2}\,\tau$ (up to an irrelevant overall phase factor) for the $x_0 = 3$ case, where each of the rotations is by the angle $\pi/2$. $H^{\otimes 2}$ is the operator

$$H^{\otimes 2} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{D2}$$

which can be implemented as $H^{\otimes 2} = R_{x1}(\pi)R_{y1}(\pi/2)R_{x2}(\pi)R_{y2}(\pi/2)$. And the operator $P$,

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{D3}$$

is simply realized as $P = R_{y1}R_{\bar{x}1}R_{\bar{y}1}R_{y2}R_{\bar{x}2}R_{\bar{y}2}\,\tau$, where again the angles are $\pi/2$. From these, we construct the Grover iteration $G = H^{\otimes 2}PH^{\otimes 2}O$. This operator can be simplified straightforwardly by eliminating unnecessary operations which obviously cancel.

Let $|\psi_k\rangle = G^k H^{\otimes 2}|00\rangle$ be the state after $k$ applications of the Grover iteration to the initial state. We find that the amplitude $\langle x_0|\psi_k\rangle \approx \sin((2k+1)\theta)$, where $\theta = \arcsin(1/\sqrt{2})$; this periodicity is a fundamental property of the quantum search algorithm, and is a natural feature to test in an experiment. For $x_0 = 3$, we find $|\psi_1\rangle = |11\rangle$, which is the desired result. Indeed, for the other three possible values of $x_0$ we also find $|\psi_1\rangle = |x_0\rangle$. Another feature of the algorithm is that $|x_0\rangle = |\psi_1\rangle = -|\psi_4\rangle = |\psi_7\rangle = -|\psi_{10}\rangle$, a period of 3 if the overall sign is disregarded. **You may (optionally) implement Grover's algorithm and observe all four cases of $x_0$. Time permitting, you may also confirm the predicted oscillatory behavior.**

### 2. Implementation of Grover quantum algorithm

The Grover algorithm for two qubits has four possible cases, corresponding to searching one of the four special elements $x_0$, $x_1$, $x_2$ or $x_3$. Construct the individual circuit elements, $P$ and $O$ using just $Z$-rotation, time delays and Hadamard gates. Then simplify your sequences using the tricks below and keeping in mind that $Z$-rotations and time delays commute:

$$H = Z^2\bar{Y} \tag{D4}$$

$$H = X^2Y \tag{D5}$$

$$Z\bar{Y}\bar{Z} = X \tag{D6}$$

Do not simplify the first two Hadamard gates. You will be able to specify the number of grover iterations you wish to apply.

Write pulse programs for all these four cases, and run `NMRRunPulseProg` to acquire the data.

## APPENDIX E: INTERFACE TO BRUKER'S XWIN-NMR

This appendix contains some reference information about the QIP experiment software configuration which you will generally not need to know for this lab, but may be helpful for instructors when debugging problems.

`netnmr2` is a C program (called an "au" script) which runs under xwin-nmr. It is a server which listens on a special socket port for commands sent from Matlab over TCP/IP. Under xwin-nmr, you can start a `netnmr2` process by typing "netnmr2" into the xwin-nmr command prompt line; it will respond with a message saying that the server is running. You can check to see if a `netnmr2` server is already running by typing "show cmd". You can kill a process by typing "kill" – then click on the process to kill.

Matlab speaks to `netnmr2` using a MEX function, `nmrx`. This function can send text commands to xwin-nmr, read parameters, run experiments, and retrieve data. For example, `nmrx('zg')` sends the "zg" command to xwin-nmr, which is "zero-go"; it zeros memory then runs an experiment. This command waits until the experiment is done, then returns control to matlab.

## APPENDIX F: MATHEMATICS OF COMPOSITE QUANTUM SYSTEMS

Given quantum systems $A$ and $B$, how does one properly describe the composite quantum system represented "$A + B$" by the combination of the two subsystems? In quantum mechanics, the answer is the *tensor product*.

Suppose $V$ and $W$ are vector spaces of dimension $m$ and $n$ respectively. For convenience we also suppose that $V$ and $W$ are Hilbert spaces. Then $V \otimes W$ (read '$V$ tensor $W$') is an $mn$ dimensional vector space. The elements of $V \otimes W$ are linear combinations of 'tensor products' $|v\rangle \otimes |w\rangle$ of elements $|v\rangle$ of $V$ and $|w\rangle$ of $W$. In particular, if $|i\rangle$ and $|j\rangle$ are orthonormal bases for the spaces $V$ and $W$ then $|i\rangle \otimes |j\rangle$ is a basis for $V \otimes W$. We often use the abbreviated notations $|v\rangle|w\rangle$, $|v, w\rangle$ or even $|vw\rangle$ for the tensor product $|v\rangle \otimes |w\rangle$. For example, if $V$ is a two-dimensional vector space with basis vectors $|0\rangle$ and $|1\rangle$ then $|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle$ is an element of $V \otimes V$. In terms of vector notation, for example, where $|0\rangle = (10)^T$ and $|1\rangle = (01)^T$, the tensor product $|0\rangle \otimes |1\rangle = (0100)^T$, and $|0\rangle \otimes |0\rangle = (1000)^T$, and so forth.

By definition the tensor product satisfies the following basic properties:

1. For an arbitrary scalar $z$ and elements $|v\rangle$ of $V$ and $|w\rangle$ of $W$,
$$z\left(|v\rangle \otimes |w\rangle\right) = (z|v\rangle) \otimes |w\rangle = |v\rangle \otimes (z|w\rangle)\,. \quad \text{(F1)}$$
2. For arbitrary $|v_1\rangle$ and $|v_2\rangle$ in $V$ and $|w\rangle$ in $W$,
$$(|v_1\rangle + |v_2\rangle) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle. \quad \text{(F2)}$$
3. For arbitrary $|v\rangle$ in $V$ and $|w_1\rangle$ and $|w_2\rangle$ in $W$,
$$|v\rangle \otimes (|w_1\rangle + |w_2\rangle) = |v\rangle \otimes |w_1\rangle + |v\rangle \otimes |w_2\rangle. \quad \text{(F3)}$$

For matrices, the tensor product is also known as the *Kronecker product*. For example, suppose $A$ is an $m$ by $n$ matrix, and $B$ is a $p$ by $q$ matrix. Then we have the matrix representation:

$$A \otimes B \equiv \overbrace{\left.\begin{bmatrix} A_{11}B & A_{12}B & \dots & A_{1n}B \\ A_{21}B & A_{22}B & \dots & A_{2n}B \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}B & A_{m2}B & \dots & A_{mn}B \end{bmatrix}\right\} mp}^{nq}. \quad \text{(F4)}$$

In this representation terms like $A_{11}B$ denote $p$ by $q$ submatrices whose entries are proportional to $B$, with overall proportionality constant $A_{11}$. For example, the tensor product of the vectors $(1, 2)$ and $(2, 3)$ is the vector

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \otimes \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \times 2 \\ 1 \times 3 \\ 2 \times 2 \\ 2 \times 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 6 \end{bmatrix}. \quad \text{(F5)}$$

The tensor product of the Pauli matrices $X$ and $Y$ is

$$X \otimes Y = \begin{bmatrix} 0 \cdot Y & 1 \cdot Y \\ 1 \cdot Y & 0 \cdot Y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \\ 0 & -i & 0 & 0 \\ i & 0 & 0 & 0 \end{bmatrix}. \quad \text{(F6)}$$

In Matlab, you can use the function `kron` for the Kronecker product.

## APPENDIX G: HOW PROTON AND CARBON SPECTRA ARISE FROM THE DENSITY MATRIX

A two-spin density matrix

$$\rho = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{bmatrix} \quad \text{(G1)}$$

produces a proton spectrum with peak areas $a - c$ and $b - d$ for the $\omega_P - J/2$ and $\omega_P + J/2$ peaks, respectively, after a $R_x(\pi/2) \otimes I$ proton readout pulse is applied. The same density matrix also produces a carbon spectrum with peak areas $a - b$ and $c - d$ for the $\omega_C - J/2$ and $\omega_C + J/2$ peaks, respectively, after a $I \otimes R_x(\pi/2)$ carbon readout pulse is applied.

Here, we prove this claim, based on the fact that the voltage in the pick-up coil for spin $k$ is given by

$$V(t) = -V_0 \mathrm{tr}\left[e^{-iHt}\rho e^{iHt}(i\sigma_x^k + \sigma_y^k)\right], \quad \text{(G2)}$$

where $H$ is the Hamiltonian for the two-spin system, $\sigma_x^k$ and $\sigma_y^k$ operate only on the $k$th spin, and $V_0$ is a constant factor dependent on coil geometry, quality factor, and maximum magnetic flux from the sample volume.

### 1. The readout operator

Let $R_{xP} = R_x(\pi/2) \otimes I$ denote a $\pi/2$ readout pulse on the proton, and $R_{xC}$ similarly for the carbon. Our goal is to compute

$$V_P(t) = -V_0 \mathrm{tr}\left[e^{-iHt} R_{xP}\rho R_{xP}^\dagger e^{iHt}[(i\sigma_x + \sigma_y) \otimes I]\right], \quad \text{(G3)}$$

and similarly for the carbon. It is helpful first to move into the rotating frame of the proton and carbon, in which case nothing changes except we utilize the Hamiltonian

$$H = \frac{J}{4}\sigma_z \otimes \sigma_z\,, \quad \text{(G4)}$$

representing the spin-spin coupling. Utilizing the cyclic property of the trace, $V_P(t)$ can be written as

$$V_P(t) = -V_0 \mathrm{tr}\left[\rho R_{xP}^\dagger e^{iHt}[(i\sigma_x + \sigma_y) \otimes I]e^{-iHt} R_{xP}\right], \quad \text{(G5)}$$

at which point it is useful to define

$$\hat{M}_P = -R_{xP}^\dagger e^{iHt}\left[(i\sigma_x+\sigma_y)\otimes I\right]e^{-iHt}R_{xP} \qquad \text{(G6)}$$

as our proton magnetization "readout operator," such that $V_P(t) = V_0\mathrm{tr}(\rho\hat{M}_P)$. Explicitly working this out in terms of matrix products, we obtain:

$$\begin{aligned}
\hat{M}_P &= -R_{xP}^\dagger e^{iHt}\begin{bmatrix}0&0&0&0\\0&0&0&0\\2\,i&0&0&0\\0&2\,i&0&0\end{bmatrix}e^{-iHt}R_{xP} && \text{(G7)}\\
&= -R_{xP}^\dagger\begin{bmatrix}e^{\frac{i}{4}Jt}&0&0&0\\0&e^{\frac{-i}{4}Jt}&0&0\\0&0&e^{\frac{-i}{4}Jt}&0\\0&0&0&e^{\frac{i}{4}Jt}\end{bmatrix}\begin{bmatrix}0&0&0&0\\0&0&0&0\\2\,i&0&0&0\\0&2\,i&0&0\end{bmatrix}\begin{bmatrix}e^{\frac{-i}{4}Jt}&0&0&0\\0&e^{\frac{i}{4}Jt}&0&0\\0&0&e^{\frac{i}{4}Jt}&0\\0&0&0&e^{\frac{-i}{4}Jt}\end{bmatrix}R_{xP} && \text{(G8)}\\
&= -R_{xP}^\dagger\begin{bmatrix}0&0&0&0\\0&0&0&0\\2\,ie^{-\frac{i}{2}Jt}&0&0&0\\0&2\,ie^{\frac{i}{2}Jt}&0&0\end{bmatrix}R_{xP} && \text{(G9)}\\
&= -\begin{bmatrix}\frac{1}{\sqrt{2}}&0&\frac{i}{\sqrt{2}}&0\\0&\frac{1}{\sqrt{2}}&0&\frac{i}{\sqrt{2}}\\\frac{i}{\sqrt{2}}&0&\frac{1}{\sqrt{2}}&0\\0&\frac{i}{\sqrt{2}}&0&\frac{1}{\sqrt{2}}\end{bmatrix}\begin{bmatrix}0&0&0&0\\0&0&0&0\\2\,ie^{-\frac{i}{2}Jt}&0&0&0\\0&2\,ie^{\frac{i}{2}Jt}&0&0\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{2}}&0&\frac{-i}{\sqrt{2}}&0\\0&\frac{1}{\sqrt{2}}&0&\frac{-i}{\sqrt{2}}\\\frac{-i}{\sqrt{2}}&0&\frac{1}{\sqrt{2}}&0\\0&\frac{-i}{\sqrt{2}}&0&\frac{1}{\sqrt{2}}\end{bmatrix} && \text{(G10)}\\
&= \begin{bmatrix}e^{\frac{-i}{2}Jt}&0&-ie^{-\frac{i}{2}Jt}&0\\0&e^{\frac{i}{2}Jt}&0&-i\,e^{\frac{i}{2}Jt}\\-ie^{-\frac{i}{2}Jt}&0&-e^{\frac{-i}{2}Jt}&0\\0&-i\,e^{\frac{i}{2}Jt}&0&-e^{\frac{i}{2}Jt}\end{bmatrix}. && \text{(G11)}
\end{aligned}$$

Similarly, we find that the analogous carbon magnetization "readout operator" $\hat{M}_C$ is

$$\begin{aligned}
\hat{M}_C &= -R_{xC}^\dagger e^{iHt}\left[I\otimes(i\sigma_x+\sigma_y)\right]e^{-iHt}R_{xC} && \text{(G12)}\\
&= -R_{xC}^\dagger e^{iHt}\begin{bmatrix}0&0&0&0\\2\,i&0&0&0\\0&0&0&0\\0&0&2\,i&0\end{bmatrix}e^{-iHt}R_{xC} && \text{(G13)}\\
&= \begin{bmatrix}e^{\frac{-i}{2}Jt}&-ie^{-\frac{i}{2}Jt}&0&0\\-ie^{-\frac{i}{2}Jt}&-e^{\frac{-i}{2}Jt}&0&0\\0&0&e^{\frac{i}{2}Jt}&-i\,e^{\frac{i}{2}Jt}\\0&0&-i\,e^{\frac{i}{2}Jt}&-e^{\frac{i}{2}Jt}\end{bmatrix}. && \text{(G14)}
\end{aligned}$$

### 2. The proton and carbon spectra

$\hat{M}_P$ and $\hat{M}_C$ are very useful, because they now allows us to compute the free induction decay signal for the proton (centered in frequency around $\omega_P$) and carbon (centered about $\omega_C$) for any state $\rho$. For the state in Eq.(G1), we obtain the proton FID

$$V_P(t) = V_0 \mathrm{tr}(\rho \hat{M}_P) \tag{G15}$$

$$= V_0 \mathrm{tr} \left( \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & d \end{bmatrix} \begin{bmatrix} e^{\frac{-i}{2} J t} & 0 & -ie^{-\frac{i}{2} J t} & 0 \\ 0 & e^{\frac{i}{2} J t} & 0 & -i\, e^{\frac{i}{2} J t} \\ -ie^{-\frac{i}{2} J t} & 0 & -e^{\frac{-i}{2} J t} & 0 \\ 0 & -i\, e^{\frac{i}{2} J t} & 0 & -e^{\frac{i}{2} J t} \end{bmatrix} \right) \tag{G16}$$

$$= V_0 \left[ (a-c) e^{-iJt/2} + (b-d) e^{iJt/2} \right] . \tag{G17}$$

And for the carbon FID,

$$V_C(t) = V_0' \mathrm{tr}(\rho \hat{M}_C) = V_0 \left[ (a-b) e^{-iJt/2} + (c-d) e^{iJt/2} \right] . \tag{G18}$$

Note that in general, it is likely that the proton and carbon signals have different voltage scale factors $V_0$, since they are usually detected by different pick-up coils in an experimental apparatus, and this is reflected in the use of $V_0'$ in the last equation above.

These equations may also be used to show why application of two simultaneous readout pulses (to proton and carbon) in one experiment may give different final voltage signals than applying one readout pulse at a time and doing two experiments.

## APPENDIX H: MATLAB SCRIPT `QIPGATES.M`

This is a Matlab script which defines quantities useful for computing what you should obtain theoretically in this experiment. The theory describes the experiment extremely well, so what you obtain with matlab is what you can obtain in the experiment.

```
%
% File:   qipgates.m
% Date:   26-Feb-03
% Author: I. Chuang <ichuang@mit.edu>
%
% Standard QC gates; with proper sign convention to be
% consistent with MIT Junior Lab quantum information processing labguide

global hadamard cnot cphase sx sy sz si rx ry rz N zz xx yy

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% pauli matrices

sx = [0 1; 1 0];
sy = [0 -i; i 0];
sz = [1 0; 0 -1];
si = [1 0; 0 1];

pauli = {sx,sy,sz};
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% two-qubit interaction terms

zz = kron(sz,sz);
xx = kron(sx,sx);
yy = kron(sy,sy);
zi = kron(sz,si);
iz = kron(si,sz);
ii = kron(si,si);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% single qubit rotations (acting on 1 qubit: 2x2 unitaries)

rx = expm(-i*pi/4*sx);
ry = expm(-i*pi/4*sy);
rz = expm(-i*pi/4*sz);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% single qubit rotations (acting on 1 of 2 qubits: 4x4 unitaries)

rx1 = kron(si,rx);
rx2 = kron(rx,si);
ry1 = kron(si,ry);
ry2 = kron(ry,si);
rz1 = kron(si,rz);
rz2 = kron(rz,si);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% one-qubit computational basis states

psi0 = [1 ; 0];
psi1 = [0 ; 1];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% two-qubit computational basis states

psi00 = kron(psi0,psi0);
psi01 = kron(psi0,psi1);
psi10 = kron(psi1,psi0);
psi11 = kron(psi1,psi1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% two-qubit Hamiltonian (for CHCl3) & coupled evolution gate

ham = zz;
tau = expm(-i*pi/4*zz);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% standard ideal quantum logic gates

hadamard = [1 1 ; 1 -1]/sqrt(2);
cnot = [1 0 0 0 ; 0 1 0 0 ; 0 0 0 1 ; 0 0 1 0];
cphase = [1 0 0 0 ; 0 1 0 0 ; 0 0 1 0 ; 0 0 0 -1];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Example: near-controlled-not

Uncnot = ry1' * tau * rx1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Example: effect of Uncnot on thermal state density matrix

rho_therm = [5 0 0 0; 0 3 0 0; 0 0 -3 0; 0 0 0 -5];
rho_out = Uncnot * rho_therm * Uncnot';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Example: Deutsch-Jozsa

Uf1 = [ 1 0 0 0 ; 0 1 0 0 ; 0 0 1 0 ; 0 0 0 1 ];
Uf2 = [ 0 1 0 0 ; 1 0 0 0 ; 0 0 0 1 ; 0 0 1 0 ];
Uf3 = [ 1 0 0 0 ; 0 1 0 0 ; 0 0 0 1 ; 0 0 1 0 ];
Uf4 = [ 0 1 0 0 ; 1 0 0 0 ; 0 0 1 0 ; 0 0 0 1 ];

Udj1 = ry2' * ry1 * Uf1 * ry2 * ry1';
Udj2 = ry2' * ry1 * Uf2 * ry2 * ry1';
Udj3 = ry2' * ry1 * Uf3 * ry2 * ry1';
Udj4 = ry2' * ry1 * Uf4 * ry2 * ry1';

Out1 = Udj1 * psi00;
Out2 = Udj2 * psi00;
Out3 = Udj3 * psi00;
Out4 = Udj4 * psi00;
```