School of Computer Science Senior Thesis

QCNMR: Simulation of a Nuclear Magnetic Resonance Quantum Computer

Matthew W. Anderson
ma@andrew.cmu.edu
Carnegie Mellon University

30/4/04

Abstract

We present the implementation of a nuclear magnetic resonance (NMR) universal quantum computer (QC) simulator we have named QCNMR. Our quantum computer simulator uses a pre-existing open source package, GAMMA, for performing the underlying NMR simulation. QCNMR reads in a quantum circuit converts it into a sequences of pulses that can be simulated on an NMR system, our system then returns the output of the simulated computation to the user. Our simulator is universal over the space of quantum computation, analogous to the way a classical computer may be called "universal".

I. Introduction

A. Overview

Classical computers have been around for a large part of a century and the computations that they can perform are entirely deterministic. There exist classes of problems that are not known to be efficiently solvable in polynomial time ($O(n^k)$) by classical computers; these types of problems are known as non-polynomial (NP) time problems.

In 1996, Grover designed an algorithm to set for a marked item on a quantum computer that provided a square root improvement over the best time a classical computer is thought to take to be able to solve the problem [Grov96]. Grover's search algorithm provided a more efficient way to solve a NP-complete problem; "complete" refers to the fact that problem is canonical to the set of NP problems therefore an efficient solution for this problem could be translated into an efficient solution for any other NP problem. Grover algorithm, however, did not provide the crucial speedup, as his algorithm still took NP time in the worst case.

In the later in 1997, Peter Shor devised an algorithm for finding the prime factors of a number in polynomial (P) time on a quantum computer. This represents a drastic speedup over the widely held belief that the problem takes NP time on a classical computer [Shor97]. Factoring, however, is not a NP-complete so these results could not be used to translate the speedup to all NP problems. There are relatively few other examples of algorithms where there is an evident speedup when the computer is allowed to take advantage of quantum mechanical effects. It is not known whether allowing use of quantum effects provides in inherent increase in the available computational power. It is also unknown as to whether there exist other algorithms that may be more efficiently performed using a quantum computer. In either case, it is important that while one is attempting to devise new algorithms for quantum computers that one has ready access to build and test their algorithm to see how and if it performs as expected.

B. Goals

To that end, the implementation of a framework to simulate quantum algorithm on readily available classical hardware would be useful to help researchers study quantum algorithms. There are several possible approaches, each with its own level of accuracy and practicality. In the first and simplest approach you assume that you have an ideal quantum computer and that you can perform more or less arbitrary operations on the qubits in your computer. The implementation itself is just a series of straightforward matrix operations. It should be noted that the time to perform the calculations to simulate the ideal QC is not the same asymptotic time that a real quantum computer would take, as each matrix operation take O(2^n) time in the number of qubits. However, to our knowledge such an ideal physical system for performing quantum computation does not even exist, it can only be approximated by other less accurate and more limited physical systems. So, while it is important that the algorithm function on an ideal quantum computer, it must also function on a realistic physical one, otherwise the algorithm has limited applicability. The next level of increased realism is the simulation of a physically based quantum computation system. This means that the simulator must calculate the computation based on some physical quantum mechanical effects and build a framework for quantum computation limited by these effects. This is the method that we have chosen to implement in this paper. This method discards the assumption that our system is ideal, and requires that the computation have basis in a physically realizable system. The third and most telling method is to actually implement the computation on a physical quantum system. This method is by far the most complicated and expensive of the three. Implementing algorithms correctly and effectively on real systems can take a long amount of time and planning; weeks or months. In addition, these systems are by no means widely available, require a great deal of technical expertise and must be custom built and tailored to the specific task. There exist a number of possible physical realizations for a physical quantum computer: nuclear magnetic resonance, quantum dots, ion traps and optical photon quantum computers and others [Cira95, Cory96, DiVi95, Yann99]. However, it remains to be seen which physical realization, if any, will be viable in the end.

The purpose of our simulator, QCNMR, is two-fold. The first purpose is to allow NMR quantum computer researchers to simulate algorithms before they attempt to perform them on a physical NMR system. This will hopefully decrease the amount of turnaround time to study an algorithm for NMR QCs. It allows basic implementation without the investment of much time. The second purpose is purely educational; it will allow students to quickly experiment with quantum algorithms, even with very little knowledge of NMR itself; hopefully, gain a more intuitive understanding of the algorithm and quantum computation as a whole.

C. Specific work

Our main contribution in this paper was to the implement framework for quantum computation on top of an existing open-source NMR simulator, GAMMA [Smit94]. QCNMR exists as a C++ library that has the ability to create qubits, a quantum circuit and to run the circuit and return the final state of the system after the circuit has run its course. QCNMR first parses the input circuit and then converts into a sequence of NMR pulses, which represents the computation we wish to perform. Pulses are essentially NMR instructions. We pass these pulses on to GAMMA to simulate the NMR system and then read back information and interpret it for the user. The main work in this project was to implement the generation of pulse sequences on the NMR system so that we could selectively address individual qubits and leave the rest of the system invariant, using a technique know as refocusing. Time was also taken to make the graphical interface for the QCNMR library that allows the user to quickly construct and run quantum circuits.

D. Organization of Paper

Section two presents some background information on NMR, quantum computation and performing quantum computation operations through NMR methods. Section three explains the structure and implementation of QCNMR. Section four describes using QCNMR to implement a quantum teleportation algorithm as well as some general comments on our assumptions in building QCNMR. Section five concludes this paper and discusses future work, related work and summarizes the results of this project.

II. Background

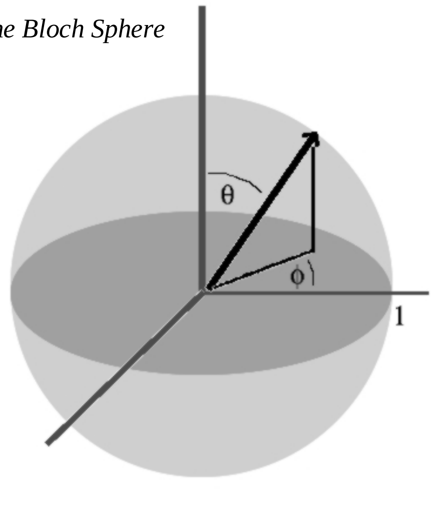
A. Quantum Computation

Quantum computation is perhaps best introduced in as an analog to classical computation. In classical computation, the object that stores or represents information is known as the "bit"; it can take only two discrete values "zero"(0) or "one"(1). The analog for quantum computation is known as the "qubit"; the information the qubit stores, however, is not discrete valued. The qubit is composed of a linear combination of vectors in complex space. Normally, the two-dimensional Hilbert space is describe using the two orthonormal basis states $|0>$ and $|1>$. As a side note, there exist n-iary qubits that are the linear combination of n orthonormal basis states. The value of a general 2-qubit can be written as:

$$|q\rangle = a|0\rangle + b|1\rangle. \qquad (1)$$

Where a and b are complex constants and the norm of $|q>$ is 1. The values of a single qubit can be thought of as a vector from the origin Euclidean three-space to the surface of a unit sphere; this is known as the "Bloch's sphere" representation (Figure 1). It is clear that the orientation of a qubit can be parameterized by two real values, giving it the capacity to store real-valued information opposed to the discrete-valued information that is stored in the classical bit. However, the information that is stored in that real-valued state of the qubit cannot be directly read. The most telling measurements that can be performed are ones that measure in orthogonal basis states, returning the basis state that the qubit's state projected into. This measurement provides a single bit of classical information. To get a more accurate determination of the actual qubit state more measurements are required, each providing a little more information about the qubit. However, this is a problem in many physical realizations as their measurement operations tend to be destructive; meaning that the measurement of a qubit disturbs it from its original state creating history-dependent effects due to measurement; this often makes is difficult to make more than one measurement on a given physical qubit in practice.

Figure 1: *The Bloch Sphere*



Similar to the theory of universal classical computation by logical circuit, we can describe an analog for quantum computation: universal quantum circuits. The wires of our circuit are qubits and the two directions along the wire represent the time evolution of the computation. We can place gates on the wires that transform qubits analogous to the transforms on classical bits by classical gates. The gates in both cases can be thought of as transformation matrices, in classical case the only values that the transformation matrices can contain are zeros and ones; in the quantum case the transformation matrices are arbitrary unitaries. Classically there is a notion of universal computation, which is being able to perform to construct arbitrary function from n to m bits, with some subset of all valid gates. One example is XOR and AND, which are sufficient for universal classical computation [Niel00]. There is the same notion in quantum computation, the controlled-not operator, U_{CNOT}, the Hadamard operator, U_H, and the phase shift operator, U_P, represents a basis for universal quantum computation.

$$U_H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix},\quad U_{CNOT} = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix},$$

$$U_S = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}. \qquad (2)$$

Other useful operators are can be thought of as rotations about the x-, y-, and z-axis, respectively U_X, U_Y, U_Z:

$$U_X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix},\quad U_Y = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix},\quad U_X = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}. \qquad (3)$$

An additional feature of quantum computation that does not have an analog in classical computation is the notation of entanglement. Entanglement is essentially the correlation in the states of two or more qubits. To make this more tangible we will present a brief example with one of the simplest entangled states, a Bell pair:

$$|B_{00}\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle|0\rangle + |1\rangle|1\rangle\right). \qquad (4)$$

This represents a state in the tensor product space of two qubits. We can make a measure of the state in the first or second qubit's space that will return a |0> or a |1>, in either of these cases the measurement of the second qubit in the same basis is fully determined:

$$M_{A|0\rangle}|B_{00}\rangle = \left(\langle 0|_a \otimes I_b\right)|B_{00}\rangle$$
$$= \frac{1}{\sqrt{2}}\left(\langle 0|_a|0\rangle_a \otimes I_b|0\rangle_b + \langle 0|_a|1\rangle_a \otimes I_b|1\rangle_b\right) \qquad (5)$$
$$= \frac{1}{\sqrt{2}}|0\rangle_b.$$

$$M_{A|1\rangle}|B_{00}\rangle = \left(\langle 1|_a \otimes I_b\right)|B_{00}\rangle$$
$$= \frac{1}{\sqrt{2}}\left(\langle 1|_a|0\rangle_a \otimes I_b|0\rangle_b + \langle 1|_a|1\rangle_a \otimes I_b|1\rangle_b\right) \qquad (6)$$
$$= \frac{1}{\sqrt{2}}|1\rangle_b.$$

We see if we measure a |0> for the first qubit then the second qubit must also measure a |0>, a similar thing happens for measuring |1>. Most of the power of quantum computation seems to come from the ability to entangle qubits, allowing information to be correlated across time and distance. It allows the state to exist in the superposition as the linear combination of multiple states. This allows for the parallel processing of many states through the evolution of a couple of qubits. In the end, this parallel state processing provides the speedup in most cases of quantum improvement over classical algorithms. Constructing entanglement requires at least two-qubit operations, such as the controlled-not gate, as seen above.

One note on terminology, a "density matrix" is one representation of a system's state. Suppose we have some state |ψ> the density matrix representation of this state is

$$\rho = |\psi\rangle\langle\psi|. \qquad (7)$$

Where ρ is the outer product of |ψ> with its conjugate, <ψ|. For example:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right), \qquad (8)$$

$$\rho = |\psi\rangle\langle\psi| = \frac{1}{2}\left(|0\rangle\langle 0| - |0\rangle\langle 0| - |0\rangle\langle 1| + |1\rangle\langle 1|\right), \qquad (9)$$

$$\rho = \frac{1}{2}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}. \qquad (10)$$

Notice that the diagonal terms of ρ sum to one; all density matrices of normalized pure states have this property.

There is one restriction on qubit operations. It is the so-called "no cloning theorem", it simply states that we cannot clone the state of one qubit to another qubit, unless we know the input comes from a known set of orthogonal states [Niel00]. This theorem prevents us from allowing fan-out gates which clones the probability distributions of qubits, though we are free to distribute information between the qubits by entangling them. However, doing so does not increase the amount of available information, it only spreads it out. Thus it is not possible to copy a state many times in order to extract more information from a destructive measurement process.

B. NMR

Nuclear magnetic resonance is a technique used in a number of fields. It can be used in analytic chemistry for chemical analysis and is the technical basis for magnetic resonance imaging (MRI) procedures for medical purposes. The isotopes of some elements have non-zero nuclear magnetic moments. These moments can be thought of as a unit vector centered at the origin; not surprisingly, it is very similar to the concept of the Bloch sphere as shown above in Figure 1. If the atom is placed in an external magnetic field, $B_0\mathbf{z}$; canonically the field is aligned pointing up the z-axis. The nuclei's magnetic moment will begin to rotate about the z-axis at a rate known as the Larmor frequency:

$$\omega = -\gamma B_0 \hat{z}. \qquad (11)$$

Where γ is the gyromagnetic ratio of the nucleus; this idea is shown diagrammatically in Figure 2 below. The frequency of the rotation is linearly proportional to the strength of the external field. In the absence of other atoms and other effects the motion of the single atom is constant in time.

When we add other atoms in a molecule with the original atom other effects begin to appear. One effect is chemical shift, $\Delta\omega$, which represents the relative shift from the Larmor frequency of the atom in isolation; this is due to shielding effects based on the arrangement of electrons mediating the bonds between atoms [Keel04]. The chemical shift allows us to differentiate between atoms two atoms of the same isotope in the same molecule. Another effect of the bonds between atoms is the J-coupling effect, ω_J; it is simply an interaction that encourages the moments of the two nuclei to point in the same or opposite directions. The J-coupling interaction can be used to mediate information exchange between two nuclei. The J-coupling interaction is anisotropic, however, its effective strength in the direction of the external magnetic field, far outweighs its strength in the two other orthogonal directions [Ladd03, Keel04].

Now if we allow multiple copies of the same molecule to reside in the same area, usually in a liquid, there can be effects caused by interaction between molecules. These effects are collectively known as decoherence. The first decoherence effect is known as thermal relaxation, which is caused by interaction between the molecules as a whole. It is like randomly applying rotations to the molecule [Grze03]. It has the effect of bringing the system of molecule to thermal equilibrium; an essentially random configuration based on the temperature of the system. It has a characteristic onset time known as T_1. The second decoherence effect is caused by J-coupling interactions taking place between atoms on adjacent molecules this also disturbs the state of the system. The characteristic onset time for the second decoherence effect is known as T_2 [Keel04, Ladd03]. The minimum of T_1 and T_2 represents the maximum time before the system decoheres and loses a substantial amount of information.

Nuclear magnetic resonance systems exist as a conglomeration of molecules. Usually the molecule of interest is put into a liquid solution with some other molecule that is not active in NMR, has only zero nuclear magnetic moments, so that the interesting molecule's properties can be distinguished from the solution's properties. The amalgamation is an ensemble, so that any measurement is represents the average value of the measurement over all molecules in the ensemble. However, since it is an ensemble, measurements that are taken are not necessarily as destructive as in other physical realizations. The measurement action in an NMR system is know as the Free Induction Decay (FID). The measurement is taken by allowing the final state of the system to decay to the thermal state. The change in the orientation of the nuclear magnetic moments is made evident by a current induce in the coils that perform the r.f. pulses. The time-varying induction, rather the Fourier transform of the time-varying induction, is used to determine the final average state of the molecular ensemble before it began to decay [Keel04].

In physics, one way of describing the time evolution of a system is through the Hamiltonian of the system. The Hamiltonian of the NMR system of the ensemble of molecules, in the frame of the laboratory, can be written as:

$$H_{lab} = \sum_{i}^{n} \left(\omega_i J_z^i + \sum_{j=i+1}^{n} \omega_{J_{ij}} J_z^i J_z^j \right). \qquad (12)$$

Where J^i_z is the z spin operator of ith nucleus; the first term represents Zeeman Effect, which is essentially the Larmor precession of the nuclear magnetic moment about the external field axis. The second term, which is the summation, represents the J-coupling interactions between all pairs of nuclei. One should notice that this construction of the Hamiltonian is time independent; therefore the way the system evolves is independent of the amount of time that has elapsed since it started. The time-dependent formulation removes the consideration of relaxation effect because their onset is time-dependent. However, to first-order, the effects of relaxation can be considered as maximum runtime cutoffs.

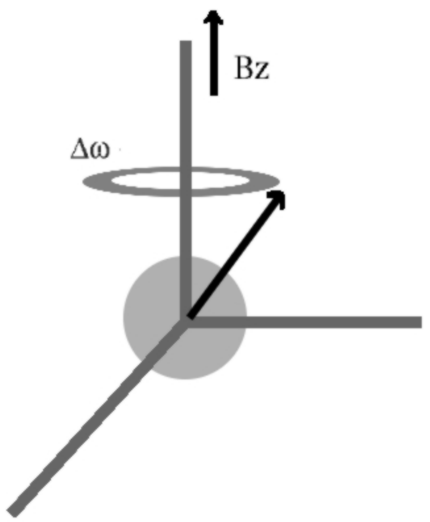
Figure 2: *Larmor Precession in the rotating frame*

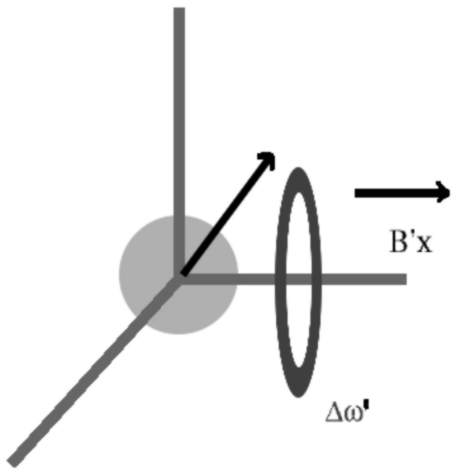
The Hamiltonian in the rotating frame of each isotope is simply the previous equation with $\Delta\omega_i$ substituted for ω_i [Ladd03].

In order to perform any sort of computation we have to be able to manipulate the NMR system. We can manipulate the system through something known in NMR as a “pulse”. A pulse is simply a radio frequency (r.f.) magnetic field applied to the system in a direction perpendicular to the external magnetic field. In order to apply the pulses in a more intuitive fashion we tend to apply them in the rotating frame of the isotope. This means if we have m different isotopes we have m different rotating frames or “channels”; these channels have a specific carrier frequency that represents the frequency of the rotating frame relative the lab frame. Generally the Larmor frequency of different isotopes is sufficiently different as to be able to address each channel separately; to be able to apply pulses to a channel without it affecting the other channels.

The simplest pulse is a one on a single qubit, it uses an r.f. pulse on the carrier frequency at the same frequency as the Larmor frequency in the qubit’s rotating frame (fig 3). In the rotating frame the effective external magnetic is many orders of magnitude smaller because the moments are now only rotating with respect to the already rotating frame, though they may not be rotating at all in the rotating frame if their chemical shift is zero.

Figure 3: *Larmor Precession in the rotating frame of a pulse.*





Therefore the dominant effect is the applied pulse which causes the qubit's magnetic moment to rotate about the axis of the applied r.f. magnetic field (Figure 3 (above)). So in order to get a ninety degree rotation about the x-axis, we must apply a pulse on the rotating x-axis for a time proportional to the Larmor frequency in the rotating frame about the pulsed axis. It should be noted that the only pulses that can be applied are pulses about an axis in the x-y plane. There exists no z-axis pulse because there is already a large constant magnetic field in the z direction. In order for the pulse to be selective the Fourier transform of the pulse must have negligible value at the frequencies of atoms you do not wish to affect; this loosely translates into a constraint on the length of time you may apply a pulse.

Longer pulses are more highly selective and shorter pulse time lengths are more indiscriminate. One important thing to note is that while a pulse is being applied selectively to one qubit all the other qubits are continuing to evolve as they normally would without the presence of the pulse, this will necessitate the use of the technique known as "refocusing", which we will describe later on. Basic parameters for defining a pulse are: start time, pulse time, central resonant frequency and r.f. strength.

C. NMR for QC

In order to perform quantum computation using an NMR system we need to define a mapping from qubits and operators to molecules and pulses. The mapping is mostly obvious. The separate qubits in the NMR system are unique atoms in a given molecule with uniquely addressable spins. If there are two atoms with very similar resonant frequency it will be difficult to use them in any way to store or transfer information. They are still part of the system and must be refocused to prevent their interactions from affecting the rest of the computation state. They are more or less ignored except in refocusing when they are grouped together and refocused. We will assume from now on that the only qubits that are in the system are individually addressable and will be useful at some stage of the computation, although the second assumption is not really necessary. When you wish to perform a QC on physical NMR equipment, much thought must go into the engineering of the specific molecule that you will use. You must have sufficiently different Larmor frequencies, which means that the molecule will likely consist of different isotopes or the small isotopes with large chemical shifts; this is all necessary to be able to reliably address nuclei. You must have the appropriate molecular structure to both mediate a strong J-coupling interaction, for quicker information exchange and a structure that increases relaxation times, T_1 and T_2, to allow more time for computation. This is only a qualitative discussion of the requirements of NMR properties for robust QC; however, a real quantitative discussion is beyond the scope of this paper, in the realm of chemical engineering.

The quantum computation operators are analogous to pulse sequences. Let us consider a simple molecule with only one atom; this negates the need for any refocusing because nothing else is evolving. We can perform a U_Z operation by simple waiting the time required for the magnetic moment to rotate by π around the z-axis (or more likely wait for $2\pi m + \pi$). We can apply U_X and U_Y by applying r.f. pulses for the appropriate time about the x- and y-axis respectively. We can therefore apply any arbitrary rotation of a qubit on the Bloch sphere.

We can generate the phase shift by performing a $Z(\pi/2)$, waiting half the time as for the $Z(\pi)$ [Niel00]. The Hadamard operator can be expressed as the sequence single qubit rotations: $Y(\pi/4)X(\pi)Y(-\pi/4)$. To have the two-qubit U_{CNOT}, we obviously now need two qubits, the U_{CNOT} pulse is constructed using several single qubit rotations and allowing the J-coupling interaction between the qubits to evolve (eqn 13).

$$U_{CNOT_{ij}} => Z_i\left(\frac{\pi}{2}\right)Z_j\left(\frac{3\pi}{2}\right)X_j\left(\frac{\pi}{2}\right)J_{ij}(\pi)Y_j\left(-\frac{\pi}{2}\right) \quad (13)$$

Thus we now have a basis for of a universal set of quantum unitary operators available. This gives us universality of computation in the realm of unitary quantum operators, analogous to the universal set in the classical realm.

The final issue that remains is that measurement of final computational state. Measurements in NMR cannot be made of individual atoms. Measurements made in NMR are ensemble measurements of the average value of some state variable, like the projection of the magnetic moment of the ith qubit onto the z-axis. Measurements are determined by the FID measurement as described before. In general, most of the average final state of the system can be determined, and we can use the FID to generate the final density matrix describing the system, though the density matrix may not be unique.

D. Refocusing

Refocusing is the technique of stopping the time evolution of certain nuclei in a NMR system. During a pulse unless a given nuclei is being addressed by the pulse that nuclei will continue to evolve as it normally would. The evolution of the entire system can be expressed, by the rotational frame, by the Hamiltonian:

$$H_{rot} = \sum_i^n \left(\Delta\omega J_z^i + \sum_{j=i+1}^n \omega_{J_{ij}} J_z^i J_x^j \right). \quad (14)$$

What we would ideally like to be able to do is perform an arbitrary unitary operation on some subset of qubits and have the rest remain unchanged. In order to perform refocusing we must determine a sequence of auxiliary pulses to prevent the unwanted time evolution.

The technique we used for generating appropriate refocusing pulse sequences is from Leung [Leun99]. This technique utilizes the properties of Hadamard matrices. Hadamard matrices are boolean n-dimension square matrices, where n is an even integer or one. Hadamard matrices have the property that all pairs of rows and all pairs of columns differ by n/2 elements. These matrices are isomorphic under swapping of rows and swapping of columns and the logical negation of rows and columns. It is always possible to make one row and one column of a Hadamard matrix all TRUE elements. Note, that in all but the one row or column that has all TRUE elements, the number of TRUE elements in each row or column must be exactly n/2. There are several methods for generating Hadamard matrices, most are fairly complex, however, there is one simple construction method for the some (eqn 16):

$$H_1 = [1], \quad H_2 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \quad (15)$$

$$H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}. \quad (16)$$

Hadamard matrices are not guaranteed to exist for all even numbers nor are all Hadamard matrices of a given dimension necessarily isomorphic under simple swap and negation operations. In order to perform efficient refocusing we need a Hadamard matrix with dimension at least as great as the number of nuclei in our system. The dimension of the matrix is the same as the number of discrete refocusing intervals we must use. The refocusing method is as follows:

1. We transform the Hadamard matrix so that the first row and the last column of the matrix contain only TRUE values.
2. Each nuclei is assigned a row in the matrix.

3. Each column in the matrix represent a T/n step in the time evolution, where T is the total evolution time of spent refocusing.
4. Assume without loss of generality, that all the elements outside the matrix are TRUE.
5. A 180-degree rotation (pulse) about the y-axis is applied to the nuclei at the beginning of each time step where the element in the matrix is FALSE and the previous element was TRUE.
6. A -180-degree rotation (pulse) about the y-axis is applied to the nuclei at the beginning of each time step where the element in the matrix is TRUE and the previous element was FALSE.

Because of (1.) no refocusing pulses are applied to qubits assigned the first row of the matrix, this means that Zeeman evolution is allowed to continue for this nuclei, through the J-coupling interaction is refocused with nuclei assigned to other rows. Also from (1.) there are no pulses applied at the end time of the refocusing sequence, T, which is a practical concern since pulses take a non-zero, finite amount of time to apply. If any qubits share assigned rows the J-coupling interaction between them is not refocused. The reasons that this scheme works are a bit mathematical and would take a bit of time to explain; we direct you to the original paper for a more insightful and thorough explanation [Leun99].

The intuition, however, is to think that the value of the element describes the direction in time the nuclei is evolving. TRUE implies forward time motion and FALSE backward time motion. Because Hadamard matrices have the property that all but the first row differ by exactly n/2 elements all the nuclei assigned other rows than the first evolve half the time going forward and half the time going backward so the net evolution is zero. This effectively refocuses the Zeeman/Larmor evolution. The J-coupling is refocused similarly, as all pairs of unique rows differ by n/2 elements meaning that half the time the two qubits are moving in the same direction and half the time they are moving in opposite directions effectively stopping the time-evolution of the interaction between them.

To make thing more concrete we will describe how we applied this scheme to construct our quantum gates. The $Z_i(\Theta)$ gate pulse sequence with refocusing is simple. Assign the nuclei we wish to evolve the first row of the matrix and all the other nuclei to unique rows in the matrix. The total evolution time is given:

$$T_z(\theta) = \frac{m + \theta/2\pi}{\Delta\omega_i}, \qquad (17)$$

where m is the number of extra full period z-rotations. This depends on the length of the refocusing pulses and the number of qubits being refocused. The constraint on T can be formed as

$$T \geq t_{mpl} \cdot n, \qquad (18)$$

where t_{mpl} is the maximum pulse length over all possibly applied pulses on a single nuclei for any of the refocusing intervals. Given these two constraints we can choose a minimum T subject to both those constraints. One thing to note is that if the chemical shift is zero the evolution time is infinite. It is a removable singularity, however, since the carrier frequency of the rotational frame is arbitrary, so we can change it to keep the refocusing times finite and reasonable.

The constructions for the $X_i(\Theta)$ and $Y_i(\Theta)$ gates are very similar to the $Z_i(\Theta)$ gate. There are two important changes, however. The total evolution time is now:

$$T_x(\theta) = T_y(\theta) = \frac{m}{\Delta\omega_i} \qquad (19).$$

The nuclei are assigned rows the same way as for the $Z_i(\Theta)$ gate, and the second constraint is still in effect. At the beginning of the pulsing sequence we apply the appropriate pulse to generate the rotation we want on the active qubit. The time constraint serves to refocus the Zeeman interaction for the active nuclei, because it was assigned the first row. Strictly speaking, this is not necessary, because we can assign the active qubit a different row in the matrix, thereby refocusing the Zeeman interaction using pulses instead of letting the evolution repeat. This would remove the constraint involving the chemical shift entirely and let the evolution time be only dependent on the maximum pulse length constraint.

The $J_{ij}(\Theta)$ gate is also very similar. The ith and jth nuclei are assigned same row of the matrix, though not the TRUE-valued row, and all other nuclei are assigned unique rows in the matrix, though also not in the TRUE-valued row. The time constraint for this gate is as follows:

$$T_J(\theta) = \frac{m + \theta/2\pi}{\omega_{J_{ij}}}. \tag{20}$$

The standard constraint of maximum pulse length (Eqn. 18) is also considered in the choice of evolution time. Note it is necessary that ω_{Jij} be non-zero for there to exist a finite refocusing time. This is not a removable singularity like the one for the $Z_i(\Theta)$ gate, the interaction must exist for the gate to be perform and the closer ω_{Jij} is to zero it is the longer the it will take to perform this action.

III. Implementation

A. GAMMA capabilities

GAMMA is an open-source library for conducting NMR Simulations created by researchers at ETH in Zurich, Switzerland for the study of nuclear magnetic resonance techniques [Smit94]. For our purposes, the GAMMA library provides a means of constructing spin systems with various interactions and properties of isotopes. It provides a means of constructing and applying pulses to the system. At the end of a run GAMMA can produce an FID (and more importantly its Fourier transform) to read the output of the final system state or can simply just provide the final density matrix.

Constructing our molecule for use in GAMMA is simply a matter of telling GAMMA the isotopes we would like to use, their chemical shift values and the J-coupling interaction strengths that exist between the various nuclei. We can then choose to set the initial state of the nuclei to whatever initial state we wish by inputting the initial density matrix to GAMMA. The capability to automatically generate initial thermal equilibrium states is provided by the library.

GAMMA allows us several different types of pulses to apply to a system of nuclear spins. The first type of pulse is an ideal pulse. The ideal pulse is performed instantaneously in simulation time and acts on a single isotope rotating all nuclei of a given isotope by a set amount. The pulse's time-strength function is a delta function. This is neither practical nor realistic. Pulses must have some finite time duration and some selectivity. The next type of pulse that GAMMA provides is a hard pulse. The hard pulse pulses all nuclei of a certain isotope about a set angle. The hard pulse length is non-zero in time, through it is usually quite short. The time-strength function for this pulse type is a square function. These sorts of pulses are appropriate when there exists only one nucleus in the molecule per isotope, if there exist more than one nuclei in the channel, they will both be excited by the pulse. The final type of pulses GAMMA provides are shaped pulses. Shaped pulses are simply pulses who's time-strength function is inputted by the user. These types of pulses allow the user to tune the selectivity of the pulse and excite only nuclei in certain frequency regions of the channel by varying the pulse strength over time. The frequencies that are excited by the shaped pulse are dependent on the Fourier transform of the time-strength function. One can tune the shaped pulse to selectively address multiple nuclei on a given channel without affecting other nuclei on the same channel. GAMMA provides some capability for the construction of the time-strength function and even has some built-in functions that are usually used in these types of pulses (i.e. Gaussian functions).

One thing that GAMMA does not provide is the ability to apply more than one pulse simultaneously in simulation time. While it represents some of a complicated issue to even physically apply two simultaneous pulses on a single channel; it is not a physical restriction that prevents applying two simultaneous pulses on two different channels. Applying two pulses at the same time on different channels is a common occurrence in everyday NMR, GAMMA's lack of capacity for this task is likely due to the inherent complexity in making an API that provide for that general purpose use. As a result we will have to implement this capacity in QCNMR.

GAMMA can return output in two ways. The first is to simply return the density matrix of the final state. The second way is to simulate the free induction decay of the system and return an array that stores the current induced in the r.f. coils as a function of time. It also provides that capability to take the Fourier transform of the FID. However, GAMMA stops there; there is little built in to the API that allows for analysis of the FID. There is nothing that attempts to estimate the final state by analysis the FID, there is nothing built in to even find peaks on the FID's Fourier transform. With the lack of built in functional for analyzing FIDs, producing FIDs is only for qualitative purposes.

GAMMA has several limitations. The first was already mentioned, the lack of simultaneous pulsing. The second limitation is the lack of FID analysis. The third limitation is that including the T_1 and T_2 relaxation effects in a way that would have been meaningful for QCNMR would have been very difficult. The implementation of that part of the library was not fully featured or complete, though the authors left notes in some of the source files for hinting at possible ways to expand the implementation. However, despite this its limitations, it is very simple to construct and run pulses for simple NMR simulations using GAMMA. GAMMA made the work of this paper possible, even though it was not tailored exactly to our needs.

B. Program Structure

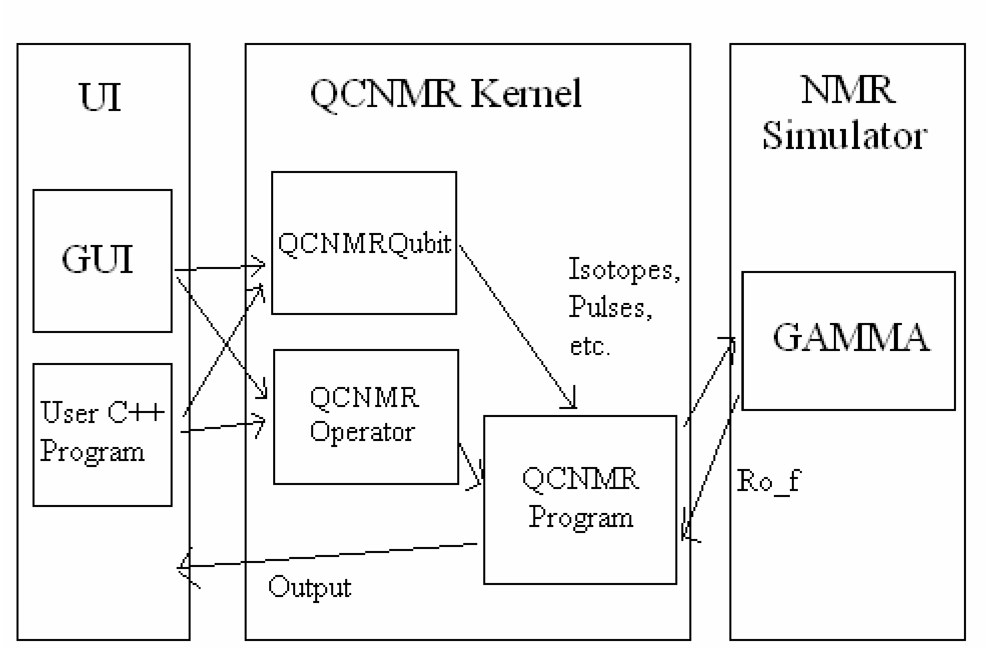
The core of QCNMR is its C++ library kernel. The kernel provides all the capacities for constructing qubits, circuit gate, executing the circuit and marshalling the output. The library is meant to work along side GAMMA providing extra functionality for quantum computation tasks. Some of the datatypes that are returned from QCNMR are quantum computationally useful mathematical types like matrices and operators, which inherently provide for more power computation in QCNMR user's programs. In addition, there is a graphical user interface that allows the quick construction and testing of circuits, without the need to compile a C++ program; its computational power is limited due to the complexity of making an interface that can fully exercise the library under an arbitrary C++ program. However, the graphical interface is more meant as an educational tool than a power computational apparatus

i. The Kernel

The kernel consists of three main classes:

1. *QCNMRQubit*
2. *QCNMROperator*
3. *QCNMRProgram*

The respective purpose of each of these classes is almost obvious, but we will explain it anyway. The instantiation of the *QCNMRQubit* class is the equivalent of a single qubit. It contains all information about the qubit including its isotope, chemical shift, J-coupling interaction strengths, pointers to all the operators that will operate on it, pointers to all the other qubits of the same isotope and information about what types of pulses can be applied to it.



.

Figure 4: *QCNMR Design Layout*

There really is not much work being done within this class during the course of the computation, it is really just a glorified struct.

The instantiation of the *QCNMROperator* class is the equivalent of a single qubit gate. It stores information about which *QCNMRQubits* are being operated on, at what time the gate is applied and the types of pulses need to occur at the top level (ignoring refocusing) to accomplish the requested operation. An auxiliary class *QCNMROperatorGen* is used to actually generate instances of *QCNMROperators*; this is because some global knowledge must be stored to determine whether or not that creation of an operator on certain qubits at a certain time is a valid thing to do.

The final kernel class is *QCNMRProgram*. It is the workhorse of the implementation. It takes in an array of *QCNMRQubits* and then allows the addition of circuit elements in the form of *QCNMROperators*. Most of the work happens in the *QCNMRProgram*::Run() function, which actually interfaces with GAMMA to transform our *QCNMRQubits* into their GAMMA representation. Then *QCNMRProgram* converts the inputted *QCNMROperators* in pulse sequences that it then runs on GAMMA, the *QCNMROperators* are destroyed in this process, though the pulse sequence that they were converted into to can be returned to the user.

The idea is that user is be allowed to perform some arbitrary circuit, look at the output and then proceed from the final state by adding new elements and running the new circuit. *QCNMRProgram* has the power to return pulse sequences to user or to save them to disk as in a human readable format. *QCNMRProgram* can also output a text-based diagram of the circuit that it sends to the disk.

The main computation in *QCNMRProgram* is dedicated to generating and running refocusing pulses. The first part of this was implementing that refocusing pulse sequence generation as was fully described in section two. The second key concern was allowing GAMMA to run simultaneous pulses. To understand what was done we must first give a brief explanation of how pulse sequences are implemented in GAMMA. A pulse on a single channel, iso, on a specific resonant chemical shift frequency, $\Delta\omega$, is implemented in the following fashion (without loss of generality the pulse is for Θ radian about the x-axis):

1. Shifting all nuclei of that isotope from the original rotating frame into the rotating frame of the resonant chemical shift frequency, we get a new Hamiltonian:

$$H'_{rot} = H_{rot} - \Delta\omega \sum_{i \in iso} J_z^i . \qquad (21)$$

2. Next we add the r.f magnetic field in the direction of our axis (the x-axis). The strength of the field is determined by the pulse time length and the angle we wish to rotation through:

$$H_{eff} = H'_{rot} + \frac{\theta}{2\pi t} \sum_{i \in iso} J^i_x. \qquad (22)$$

3. We then leave the system to evolve under H_{eff} for time t.
4. Since we shifted to another rotating frame we need to shift back to the original frame, this can be done by constructing a Hamiltonian:

$$H_{shift} = \Delta\omega \sum_{i \in iso} J^i_z. \qquad (23)$$

5. We evolve the system again for time t under H_{shift}, though this time is not real simulation time, it is just some mathematical bookkeeping to put us back in the original rotating frame.
6. The pulsing is now complete and we return to using our original H_{rot}.

This method performs a hard rectangular pulse or, if the time is long enough a soft rectangular pulse centered on the specified frequency. In order to perform more complex pulses this method is used as an atomic operation. To perform shaped pulse we apply a series of rectangular pulses with varying strengths. The shaped pulse is applied through the discretization of its time-strength function. So in order to allow pulses to fire simultaneously in a general sort of way you have to keep track of all the changes that you have made and all the changes you need to make to the effective Hamiltonian. The discrete time step method for performing shaped pulses adds even more complexity to the problem. However, when you get down to, it is not a technically difficult problem, it is just tedious. We will not go into the implementation details for scheduling and ordering changes to the Hamiltonian and evolving the state appropriately as it is just a programming task and does not add anything interesting to this discussion.

QCNMRProgram's main output is the final density matrix of the system. This is clearly not a realistic feature, an NMR machine will not simply give the final density matrix of the system; you must compute it from the FID. There are basically two reasons we choose not to provide the output via FID. The reason one reads an FID and converts it to a density matrix was simply because the density matrix itself is unavailable, so since GAMMA freely gives us the density matrix it makes little sense to transform the density matrix to an FID for the purposes of transforming it back to an FID. The second reason is that as we mentioned before GAMMA does not have much built in capacity for analyzing FIDs, so it would have required significant extra work to add that on to support an admittedly dubious process. We did add the ability to take the partial trace of a density matrix against some parts of the space to make it easier to examine the states of subspaces of the system.

ii. The GUI

The graphical front-end of QCNMR is almost an afterthought. It has a simply point-click interface that allows the user to quickly construct quantum circuits and view the final output of the system as a graphically expressed density matrix or as a diagram of pulses. At the time of writing this paper, the first version of the GUI is finished, though a few minor bugs still remain. It makes it very easy to just put down a few gates and experiment with some small circuits. The output capabilities are limit to diagrams and graphs, because large amounts of textual data are confusing and often not useful. In the short time we have had to experiment with it, it seems as though it will be useful educationally, even if a person knows little about the nuclear magnetic resonance effects it is based on. It is an interesting thing to play with if nothing else.

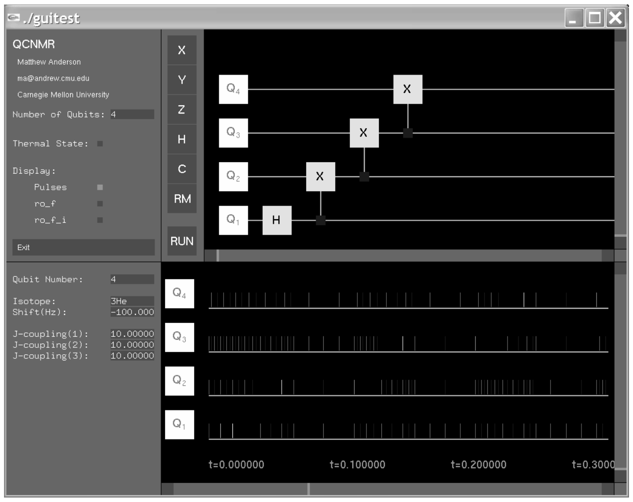


Figure 5: *Graphical User Interface Screenshot*

./guitest

QCNMR
Matthew Anderson
ma@andrew.cmu.edu
Carnegie Mellon University

Number of Qubits: 4
Thermal State:
Display:
Pulses
ro_f
ro_f_i

Qubit Number: 4
Isotope: 3He
Shift(Hz): -100.000
J-coupling(1): 10.00000
J-coupling(2): 10.00000
J-coupling(3): 10.00000

IV. Results - Quantum Teleportation

One interesting quantum algorithm that we had a chance to implement using QCNMR, was the canonical quantum teleportation circuit. The circuit is shown below in Figure 6. All three qubits start out in the state |0>. Operations may be performed on the Q_1, state to produce a different input to the system (step one on the diagram). Next qubit two and three are initialized into an entangled Bell state (step two on the diagram):

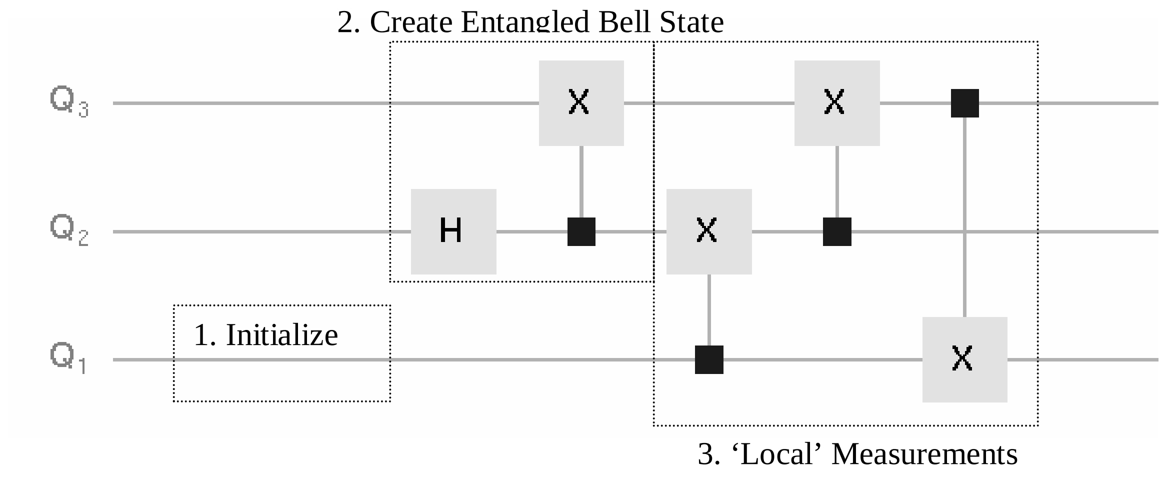
$$\left|B_{oo}\right\rangle = \frac{1}{\sqrt{2}}\left(\left|0\right\rangle\left|0\right\rangle + \left|1\right\rangle\left|1\right\rangle\right) \tag{24}$$

In the original formulation of the problem the qubit two was given to Alice and qubit three was given to Bob after the qubits have been entangled. Alice starts by initializing qubit one into some state. Then Alice can make some local measurements on qubit one and qubit two and send the results classical to Bob, who may be some distance away. Bob can then use the classical information that Alice sent him to decide whether or not to apply certain operations to his qubit three. The end result is that the final state of Bob's qubit three is the same quantum state as initial state that Alice prepared. Because we have not implemented classical channels with QCNMR, we have to use an equivalent circuit with only quantum operations.

We implemented this circuit on a heteronuclear three-qubit system. The spin parameters where more or less arbitrary, the chemical shift for each nuclei was 200Hz, and the J-coupling strength between each pair was set to 10Hz. Since we have a heteronuclear system all pulses are hard pulses on a channel. The nominal pulse length was one microsecond. The pulse sequence for the sections two and three of the teleportation circuit are shown below in Figure 7. As you can see the total execution time for the circuit is on the order of 0.3 seconds. Notice the four symmetric open regions in the pulse diagram. These are the regions where the J-coupling interaction was being allowed to evolve. Since the J-coupling term is an order of magnitude smaller than the chemical shift, the time required to perform the J-coupling over some angle increases by an order of magnitude. Similarly, the tightly packed regions of the diagram correspond to the single-qubit evolution that occur as part of the CNOT gates and the initial Hadamard gate. The numerical results themselves are not too interesting.

The original density matrix after preparation of the qubit one traced down to qubit one is almost identical to the final density matrix traced down to qubit three.

The interesting thing is that when we interesting the length of the hard pulses, the fidelity of the final teleported state to the initial prepared state decreases dramatically. However, this is to be expected, one of the requirements of the refocusing processes was that the pulse lengths must be kept short [Leun99]. The time that the pulses are being applied is larger relative to the total duration of the refocusing, likely due to the fact that not all qubits are being pulsed for the same amount of time during the refocusing sequence; we believe



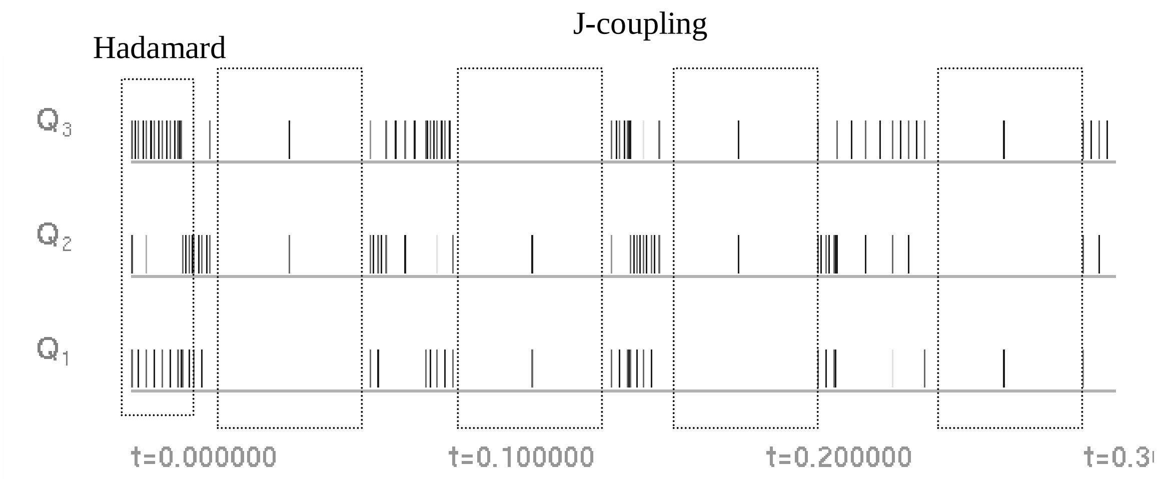


Figure 6 (top): *Quantum Teleportation Circuit*
Figure 7 (bottom): *Quantum Teleportation Pulse Sequence*

that this causes the effectiveness of the J-coupling refocusing between qubits to break down. The Zeeman refocusing should be unaffected. This suggests that smaller J-coupling strengths would improve this problem, however, that would increase the total refocusing time required. In the end it is a balancing effort between pulse length and refocusing time.

We did not have a chance to examine as many quantum circuits as we would like to have. The main stumbling block other than time, was that most of the interesting circuits involve construction of some set of U_f's that act as oracles in the Deutsch-Jozsa problem or the Grover Search problem [Erma03]. It was not obvious to us how to construct these unitary functions using the gates we had available. However, during the early stages of development of QCNMR, we did test out the unitaries provided for the three-qubit Deutsch-Jozsa algorithm [Kim00]. These unitary functions where not constructed using the standard gate set, they we constructed using NMR rotations and interactions. It is in the realm of QCNMR to perform such operations as we described in previous sections; that functionality is privately internal to the kernel and for design choices was not exposed to the user. It would be simple to allow the construction gates that allowed for arbitrary X, Y and Z rotations, but they were hidden in order to make the user's interface cleaner. In future iterations of the QCNMR, this functionality will likely be exposed.

Finally, a note about simulated and experimental accuracy; QCNMR is not intended to exactly simulate a nuclear magnetic resonance computer in complete detail. There are a number of practical assumptions that had to be made in order to make this project tractable. The most important assumption is that we consider the NMR system as a statistical ensemble as opposed to a physical n-body system, which is, itself, is a horribly intractable problem. Along with that assumption was the assumption that the Hamiltonian for our system was time-independent, meaning the bulk relaxation effects were not being considered, though they could be included to zero-th order as a constraint on total computation time. Another assumption that we considered was that the only qubits in the system were nuclei that we involved in the computation, this is not strictly necessary, the user can add whatever extra qubits they wish to add, the only cost is that refocusing is performed uniquely on each one of them. The final, and perhaps somewhat understated assumption, is that we assumed that refocusing was necessary for atomic operation, strictly speaking, refocusing is probably not always necessary, in fact, it is probably wasteful sometimes. The problem is that it is difficult to quantify the effectiveness of using refocusing at some point in the evolution of the circuit. While always refocusing is good from a correctness point of view, it can often be dubious from an efficiency perspective. Again, it is a case where balance and optimization of parameters comes into play. We decide to err on the side of correctness as opposed to the side of efficiency, for it will likely be easy to approach an optimal solution from a solution that is already correct.

V. Conclusion

A. Summary

We presented our construction of a nuclear magnetic resonance quantum computer simulator, QCNMR. We discussed the necessary background for a layman to get a good physical understanding of NMR and quantum computing at a basic level. Our algorithms for constructing refocusing sequences and for performing simultaneous pulses were also discussed. We examined the structure and interface to the open-source NMR simulation library that we used as a backend for QCNMR's nuclear magnetic resonance simulation as well as its technical short-comings and limitations. The basic structure and classes in our implementation were briefly described in their form and function. We have concluded with a short example circuit that we implemented on QCNMR. For the most part this project constructed what it set out to do; to implement an emulator that takes a "classical" quantum circuit and performs it by using NMR pulses. Finally we commented on a number of assumptions that were made in the formulation and implementation of QCNMR. The next major step from QCNMR is to construct a time-correctness optimization solver for applying pulses.

B. Related Work

There are relatively few examples of similar attempts to implement a nuclear magnetic resonance quantum computer simulator. One such example is the Quantum Computer Emulator (QCE) by a group at the University of Groningen, Demark [Mich03]. Their approach is at a much lower level, by allowing the user to choose all the pulses and the parameters of the pulses. They make this slightly more tractable to a novice by constructing instruction sets of basic useful pulses with pre-set parameters. Their implementation, however, seems to focus on being an ideal quantum computer simulator as opposed to a NMR based simulator. They to not seem to provide the automatic construction of the necessary refocusing pulse for the user; it is up to the user to implement them however they choose. This method is allows the user more control over how their abstract quantum circuit is implemented in NMR hardware, though at the cost of considerably more time to implement with the QCE system. QCNMR provides a quick way for user to test and examine the basic properties of circuits that will be on simulated NMR system at the loss of user control of the pulse sequencing.

C. Future Work

We would like to have spent more time testing some example circuits to examine the effectiveness of QCNMR, but backlogs in the development schedule hindered the process. We did not spend much time trying to optimize the pulse sequences beyond applying multiple pulses at the same time. Future work could address optimizing pulse sequences both on a per-operation basis and on a global scale using some linear programming techniques. There was not enough time to compare the results of the simulated computation and the generated pulse sequences to the physical implementations and actual pulse used sequences. There is one thing that remains to be implemented, however, at the time of publishing this paper, simultaneous homonuclear soft pulses are not yet implemented due to complexity issues and time constraints. We also look to further iterations on the development of the GUI, in order to add make it a more powerful tool that allows it to exercise more features of the QCNMR library. The issue of swapping qubits (and optimizing the swapping of qubits) for performing information exchange on qubits that have zero or very small J-coupling interaction strength was not even discussed. There are a number of simple optimizations that come to our minds as this paper was being written; mostly optimizations in the maximum time duration of a refocusing sequences and the organization of pulses within each refocusing pulse interval.

D. Acknowledgements

Firstly, thanks to the research group at the University of Florida and in Switzerland that produced GAMMA and made it openly available to the public, for without which this project would have been impossible. Secondly, thanks to Dr. Arvind, for some useful pointers and discussions about using NMR for quantum computation. Thirdly, we would like to thank Dr. Avrim Blum for signing as co-advisor to this project and supporting it even though it is not his area of expertise. Fourthly, we would like to thank Mark Stehlik for organizing the CS Senior Thesis program, as well as for some good advice concerning this thesis and other aspects of CMU CS undergraduate life. Finally, and most importantly, we would like to Dr. Robert Griffiths for being the main advisor of this research; there were definitely some rough spots during the past two semesters, but it seems we made it through no worse for wear having completed a frustrating, though ultimately worthwhile, project. Thanks everybody.

VI. Bibliography

Cira95 Cirac I J and Zoller P *Quantum computations with cold trapped ions*, Phys. Rev. Let., 74: 4091-4094, 1995.

Cory96 Cory, D. G., Fahmy, A. F. and Havel, T. F., Nuclear magnetic resonance spectroscopy: an experimentally accessible paradigm for quantum computing. In Proceedings of the 4th Workshop on Physics and Computation, Boston: New England Complex Systems Institute, (1996).

DiVi95 DiVincenzo, D. P., Quantum Computation. Science, pages 270:255, 1995.

Erma03 Ermakov, V., B. M. Fung, Nuclear Magnetic Resonance Implemention of the Deutsch-Jozsa Algorithm Using Different Initial States, arXiv:quantph/0304058 (2003).

Grov96 L. K. Grover, A fast quantum mechanical algorithm for database search, Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, 212-219, Philadelphia, PA, (May, 1996).

Grze03 Grzesiek, S., Notes on Relaxation and Dynamics. Course Notes, EMBO Practical Course on NMR, Heidelberg, 2003.

Keel04 Keeler, James. Understanding NMR, http://www-keeler.ch.cam.ac.uk/lectures/ (2004).

Kim00 Kim, J., J. Lee, S. Lee, C. Cheong, Implementation of the Refined Deutsch-Jozsa Algorithm on a Three-Bit NMR Quantum Computer. Physical Review A, 62, 022312 (2000).

Ladd03 Ladd, T., NMR Quantum Computation. Course Notes, Quantum Information Science and Technology. (2003).

Leun99 Leung, D., I. Chuang. F. Yamaguchi, Y. Yamamoto. Efficient Implementation of Selective Recoupling in Heteronuclear Spin Systems Using Hadamard Matrices. arXiv:quant-ph/9904100.

Mich03 Michielsen, K., H. Raedt, QCE: A Simulator for Quantum Computer Hardware, University of Groningen, The Netherlands, (2003).

Niel00 Nielson, M., I. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, (2000).

Shor97 Shor, Peter W., "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", SIAM Journal on Computing,26,5,pages 1484-1509,1997.

Smit94 S.A. Smith, T.O. Levante, B.H. Meier, and R.R. Ernst, Computer Simulations in Magnetic Resonance: An Object Oriented Programming Approach, *J. Magn. Reson.*, 106a,77-104,(1994).

Yann99 Yannoni, C., M. Sherwood, D. Miller, I. Chuang, Nuclear Magnetic Resonance Quantum Computing Using Liquid Crystal Solvents. arXiv:quant-ph/9907063 (1999).