TIME-SYMMETRIC SHAPED PULSES FOR SPIN-1 EXCITATION

By

©Simon Habot

A thesis submitted to the School of Graduate
Studies in partial fulfillment of the
requirements for the degree of
Master of Science

Department of Physics
The University Of Lethbridge
May 1998

Lethbridge Alberta Canada

Table of Contents

List of Figures

List of Tables

To my parents

Acknowledgements

It is my great pleasure to thank all those who have contributed to this thesis.

I am very grateful to my supervisor Dr.D.J. Siminovitch for introducing me to the stimulating field of Solid State NMR and for supervising my MSc program.

I also wish to thank Dr. M.K. Ali, who introduced me to the science of Physics as an undergraduate student, for serving on my thesis committee and for providing me with additional insight.

To Dr. D.D. Frantz, who served on my thesis committee during the earlier stages of my graduate program, I am grateful for providing me with practical advice about software and hardware. To Dr. R. Boeré, who joined my thesis committee at a later stage and augmenting with his experience.

To Dr. D. Lu, I thank for providing expressions: (6.1,6.2, 6.3), tables: 6.1, 6.2, figures: 6.1, 6.1, and for information on useful references.

I thank N. J. Tagg for § 4.2 and figures 4.3, 4.4, 6.4 6.5.

I gratefully acknowledge Greg Tompkins and Dave Daisley, for continued technical support.

Chapter 1

Introduction

Shaped pulses can be used for uniform spin-1 excitation. The effects of the pulses on spin-1 excitation is seen as distortion of two types: phase distortions and amplitude distortions. By reducing the distortions a spin-1 excitation becomes more uniform. In the case of time-symmetric shaped pulses, spin-1 excitation is free of phase distortions. The spin-1 excitation in that case can be made uniform over a larger frequency bandwidth.

The number of possible shaped pulses is so large that a computer-aided search is needed to find the desirable shaped pulses.

A theoretical analysis is used to find the connection between a shaped pulse and the corresponding spin-1 excitation. The theoretical analysis in density matrix formalism gives the spin-1 excitation in closed-form expressions that are too complicated. In such a case the connection between a shaped pulse and spin-1 excitation is not straightforward. A brute-force search for a desirable shaped pulse can consume too much computer time and thus limit the scope of the search.

By using the formalism of quaternions in the theoretical analysis, spin-1 excitation is presented in simple closed form expressions. It is then shown that if the choice is limited to time-symmetric shaped pulses then these closed form expressions become much simpler. It is also shown that a spin-1 excitation is free of phase distortions in that case.

These simple closed form expressions can be used as the building blocks of a much more concise program code for the computer aided search. As a result a computer aided search for a desirable shaped pulse becomes much faster in speed and larger in scope.

More shaped pulses for improved spin-1 excitation can be found.

Chapter 2 Introduces the quaternion formalism employed in the theoretical analysis. The advantages of its simplicity are discussed.

Chapter 3 Introduces earlier type of pulses. The excitation profile idea is discussed in the context of each type of pulse.

Chapter 4 The theory of the shaped pulse is introduced. Time symmetry is used to simplify the theory and improve the extent of uniform spin-1 excitation.

Chapter 5 The laboratory experiment is presented: instrumentation, choice of sample, construction of the apparatus used to produce shaped pulses is discussed in terms of hardware and software, pulse programmer used run each experiment.

Chapter 6 Theoretical results of a variety of shaped pulses, composite pulses, and the normal quadrupole pulse are compared. Experimental results of the best time-symmetric shaped pulse are compared with the results of the normal quadrupole pulse.

Chapter 7 On the basis of theoretical and experimental results it is concluded time-symmetric shaped pulses can be used to achieve more uniform spin-1 excitation. Such time-symmetric shaped pulses may have higher efficiency than composite pulses and square pulses.

Appendix A The quaternion formalism is presented. The advantages of quaternions in the context of quantum mechanics are shown.

Appendix B The density matrix evolution is introduced in terms of a particular basis. It is then shown how quaternion formalism is used to express density matrix evolution in a simple closed form.

Appendix C Computer code developed and run to shape the pulses.

Appendix D Plotting routines developed and used on data.

Chapter 2

Theory

2.1 The Density Matrix formalism

The density matrix formalism is a statistical description of a spin ensemble. This formalism is necessary for the study of the physical properties of the ensemble e.g. expectation values.

The time dependance of such a system can be described completely by a state vector (wave function) as

$$| \psi >= \sum_{i=1}^{n} c_i(t) \mid i > \tag{2.1}$$

where $\{c_i(t)\}$ are the components of that state vector and $\{|i>\}$ form a complete orthonormal basis set in Hilbert Space.

Two cases are distinguished in the description of the spin ensemble:

1) A *pure state* where all spin systems of the ensemble are in the same state 2.1. In such a case state vector 2.1 is the state of the entire ensemble.

2) A *mixed state* where each spin system may be in one of several possible states. One can only indicate a probability for the state of each spin system. The state of the ensemble is the average of its spin systems.

In a mixed state the state vector may vary from system to system according to some statistical distribution. Thus the state vector by itself cannot describe the ensemble: the knowledge of the system is incomplete. For example, one cannot calculate an expectation value of a single system in the ensemble. Instead, the entire ensemble must be considered

where each system is described by a state vector $\mid \psi_r >$ and an associated probability P_r. If we now want to calculate an expectation value of an operator B, we can only get a mean value of this expectation value after averaging over *all* the states in the ensemble:

$$< \bar{B} >= \sum_r P_r < \psi_r \mid B \mid \psi_r > \tag{2.2}$$

where $< \psi_r \mid B \mid \psi_r >$ is the average value of B in state $\mid \psi_r >$.

To further explore the mean value of an expectation value operator B, another operator

$$\sum_i \mid i >< i \mid = 1, \tag{2.3}$$

is introduced in terms of the orthonormal basis set $\{\mid i >\}$. This operator is known as a *projection operator.* It is applied to equation (2.2) to give

$$\begin{aligned} < \bar{B} > &= \sum_i \sum_j \sum_r P_r < \psi_r \mid i >< i \mid B \mid j >< j \mid \psi_r > && (2.4) \\ &= \sum_i \sum_j < i \mid B \mid j > \sum_r P_r < j \mid \psi_r >< \psi_r \mid i > && (2.5) \\ &= \sum_i \sum_j < i \mid B \mid j >< j \mid \sum_r \mid \psi_r > P_r < \psi_r \mid i > && (2.6) \\ &= \sum_i \sum_j < i \mid B \mid j >< j \mid \rho \mid i > && (2.7) \end{aligned}$$

where the density matrix ρ is defined as

$$\rho \equiv \sum_r \mid \psi_r > P_r < \psi_r \mid \tag{2.8}$$

Equation (2.7) is a product of a quantum mechanical average of operator B and a statistical average given by the density matrix ρ. The indices of equation(2.7) imply that the R.H.S is a trace of a product of two matrices:

$$< \bar{B} >= Tr\{\rho B\} \tag{2.9}$$

Thus, the statistical nature of the density matrix made it possible to represent the dynamical state of the system, regardless of the degree of knowledge of that state. The

density matrix then, is a sufficient requirement to measure all physically measurable quantities of the ensemble.

2.2 Spin-1 Evolution By Density Matrix

To express the spin-1 evolution of the ensemble in terms of the density matrix we have to consider the state vector and the Schrödinger equation

$$i\hbar\frac{d \mid \psi_r >}{dt} = \mathcal{H} \mid \psi_r >, \tag{2.10}$$

which governs this evolution with $\mathcal{H}$ as a total energy operator known as the *Hamiltonian*. The density matrix, as we already know, is a projection operator expressed in terms of state vectors $\mid \psi_r >$ of the ensemble. Thus, using $\mid \psi_r >$ and equation(2.10) it obeys, one can construct the equation that governs the spin-1 evolution of the density matrix, ρ.

We start by taking the time derivative of the density matrix equation (2.8) and using equation(2.10):

$$\begin{aligned} i\hbar\frac{d\rho}{dt} &= i\hbar\sum_r\left(\frac{d|\psi_r>}{dt}P_r < \psi_r| - |\psi_r > P_r\frac{d < \psi_r|}{dt}\right) && (2.11)\\ &= \sum_r(\mathcal{H}|\psi_r > P_r < \psi_r| - |\psi_r > P_r < \psi_r|\mathcal{H}) && (2.12)\\ &= \mathcal{H}(\sum_r |\psi_r > P_r < \psi_r|) - (\sum_r |\psi_r > P_r < \psi_r|)\mathcal{H} && (2.13)\\ &= \mathcal{H}\rho - \rho\mathcal{H} && (2.14)\\ &= ([\mathcal{H},\rho]) && (2.15) \end{aligned}$$

we get the equation which governs the density matrix:

$$\frac{d\rho}{dt} = \frac{i}{\hbar}[\rho,\mathcal{H}] \tag{2.16}$$

Equation(2.16) is the Liouville-von Neumann equation (LVN hereafter).

A formal solution for the LVN equation is obtained if its Hamiltonian is at least piecewise independent of time. This is usually the case in NMR. A time dependent

Hamiltonian on the other hand, can be discreted into time independent segments of finite duration.

The formal solution is given by

$$\rho(t) = \exp(-\frac{i}{\hbar}\mathcal{H}t)\rho(0)\exp(\frac{i}{\hbar}\mathcal{H}t) \tag{2.17}$$

$$= U(t)\rho(0)U(t)^{-1} \tag{2.18}$$

where $U(t)$ is a unitary operator which propagates the density matrix in time and $U(t)^{-1}$ as its inverse.

Although equation(2.18) is a formal solution, it may not always be a useful one. For example, the density matrix may undergo a series of unitary transformations that may not commute. In such a case the resultant unitary operator for n unitary transformations, given by

$$U_n(t) = \prod_{j=1}^{n} U_j(t), \tag{2.19}$$

is most likely to have a rather complicated functional form. Furthermore, in most cases simplifying $U_n(t)$ may not be easy since success depends on whether or not the right unitary operators can be found. This problem arises from algebraic properties of operators in the density matrix formalism. Such properties are reflected in the type of commutators that may be available. Understandably, one would rather be able to simplify equation(2.19) without having to depend on a functional form..... or have a better formalism in the first place.

2.3 The Density Matrix In The New Form

2.3.1 The Quadrupolar Hamiltonian

In order to manipulate the time evolution of the density matrix, it is essential to construct the Hamiltonian in detail. The Hamiltonian, which describes the total energy of the system, arises from the interaction between the system and its environment as well as the interaction among the internal parts of the system. The interest is in the electric quadrupolar interaction. However, since the manipulation and measurements of the time evolution are usually done by means of magnetism, the electric quadrupolar interaction could only be observed via its effects on the magnetic behaviour of the system. Hence , the interaction between the magnetic dipole and the external field applied to the system will also be addressed.

The energy of the interaction (coupling) between a classical electric charge distribution $\rho(\mathbf{r})$ and an electrostatic potential $V(\mathbf{r})$ is given by

$$E = \int \rho(\mathbf{r})V(\mathbf{r})d\tau \tag{2.2}$$

The electrostatic potential can be expanded about the center of $\rho(\mathbf{r})$

$$V(\mathbf{r}) = V(0) + \sum_{\alpha} x_{\alpha}\frac{\partial V}{\partial x_{\alpha}} + \frac{1}{2!}\sum_{\alpha\beta} x_{\alpha}x_{\beta}\frac{\partial^2 V}{\partial x_{\alpha}x_{\beta}} + \ldots \tag{2.3}$$

where each term of this expansion is due to a fundamental distribution of electric charge external to $\rho(\mathbf{r})$. Each such fundamental charge distribution has its unique symmetry.

The following definitions of the partial derivatives of equation (2.3)

$$V_{\alpha} \equiv \frac{\partial V}{\partial x_{\alpha}} \tag{2.4}$$

$$V_{\alpha\beta} \equiv \frac{\partial^2 V}{\partial x_{\alpha}x_{\beta}} \tag{2.5}$$

Can be used to rewrite the energy of interaction (2.2) as

$$E = V(0)\int \rho d\tau + \sum_{\alpha} V_{\alpha}\int x_{\alpha}\rho d\tau + \frac{1}{2!}\sum_{\alpha,\beta} V_{\alpha\beta}\int x_{\alpha}x_{\beta}\rho d\tau + ... \tag{2.6}$$

where

$$\textit{electric monopole} = \int \rho d\tau \tag{2.7}$$

$$\textit{electric dipole moment} = \int x_{\alpha}\rho d\tau \tag{2.8}$$

$$\textit{electric quadrupole moment} = \int x_{\alpha}x_{\beta}\rho d\tau \tag{2.9}$$

The quantum mechanical version of the energy expression, equation (2.6), is obtained by replacing the classical version of $\rho(\mathbf{r})$ with an operator

$$\rho^{(op)}(\mathbf{r}) = \sum_{k} q_k \delta(\mathbf{r} - \mathbf{r_k}) \tag{2.10}$$

$$= e\sum_{k} \delta(\mathbf{r} - \mathbf{r_k}) \tag{2.11}$$

This operator is now the charge distribution of a nucleons which is a quantum mechanical entity. The electrostatic potential must also be expressed in quantum mechanical terms. The energy of interaction is then given by a quantum mechanical operator, a Hamiltonian.

The electric monopole, equation (2.7), which is a point charge, is independent of the size, shape, and the orientation of the nucleus. The electric dipole moment, equation (2.8), vanishes since $\rho^{(op)}(\mathbf{r})$ has even symmetry (definite parity) while x_{α} is odd. The electric quadrupole moment, equation (2.9), does not vanish since $x_{\alpha}x_{\beta}$ makes that integrand even. In fact half of the terms of equation (2.6) vanish since their respective integrands are odd. The next even term up, the electric hexadecapole moment, does not vanish but can be safely dropped as it is usually too weak to detect. Thus only the electric quadrupole moment may be considered.

The integral part of equation (2.9) can be rewritten as:

$$\int x_{\alpha}x_{\beta}\rho d\tau = \frac{1}{3}(Q_{\alpha,\beta} + \int \delta_{\alpha,\beta} r^2 \rho d\tau) \tag{2.12}$$

where

$$Q_{\alpha,\beta} = \int (3x_\alpha x_\beta - \delta_{\alpha,\beta} r^2)\rho d\tau \tag{2.13}$$

$$\begin{aligned} E^{(2)} &= \frac{1}{2}\sum_{\alpha,\beta} V_{\alpha\beta}\int x_\alpha x_\beta \rho d\tau && (2.14)\\ &= \frac{1}{6}\sum_{\alpha\beta}\left(V_{\alpha\beta}Q_{\alpha\beta} + V_{\alpha\beta}\delta_{\alpha\beta}\int r^2\rho d\tau\right) && (2.15)\\ &= \frac{1}{6}\sum_{\alpha\beta}\left(V_{\alpha\beta}Q_{\alpha\beta} + V_{\alpha\alpha}\int r^2\rho d\tau\right) && (2.16)\\ &= \frac{1}{6}\sum_{\alpha\beta}\left(V_{\alpha\beta}Q_{\alpha\beta} + V_{\alpha\alpha}\int r^2\rho d\tau\right) && (2.17) \end{aligned}$$

Since $V(\mathbf{r})$ is due to extranuclear charge distribution, it must satisfy Laplace's equation. Thus at the nucleus we have

$$\sum_\alpha V_{\alpha\alpha} = 0 \tag{2.18}$$

so that the last term in equation(2.17) vanishes. We then have

$$E^{(2)} = \frac{1}{6}\sum_{\alpha\beta} V_{\alpha\beta}Q_{\alpha\beta} \tag{2.19}$$

which is just a product of two second-rank tensors.

The quantum mechanical version, operator version, of equation(2.13) is then given by

$$\begin{aligned} Q^{(op)}_{\alpha\beta} &= \int (3x_\alpha x_\beta - \delta_{\alpha\beta} r^2)\rho^{(op)}(\mathbf{r})d\tau && (2.20)\\ &= e\sum_k \int (3x_\alpha x_\beta - \delta_{\alpha\beta} r^2)\delta(\mathbf{r} - \mathbf{r_k})d\tau && (2.21)\\ &= e\sum_k (3x_{\alpha k}x_{\beta k} - \delta_{\alpha\beta} r_k^2) && (2.22) \end{aligned}$$

and where equation (2.11) was used. The quantum mechanical version of the quadrupole terms, the quadrupolar Hamiltonian, is given by

$$\mathcal{H}_Q = -\frac{1}{6}\sum_{\alpha\beta} V_{\alpha\beta}Q^{(op)}_{\alpha\beta} \tag{2.23}$$

To obtain the matrix elements of $Q_{\alpha\beta}^{(\sigma p)}$, the Wigner-Eckart theorem is applied to equation (2.22)

$$< Im\eta|e\sum_k(3x_{\alpha k}x_{\beta k} - \delta_{\alpha\beta}r_k^2)|Im'\eta >=< Im\eta|3\frac{(I_\alpha I_\beta + I_\beta I_\alpha)}{2} - \delta_{\alpha\beta}I^2|Im'\eta > C \quad (2.24)$$

where m (and m')is a quantum number of the nucleus and I is its spin number

$$-I \leq m \leq I \quad (2.25)$$

C is a function of I and η only. The latter is a set of quantum numbers associated with other operators that commute with I and I_z. The simplest way to obtain an expression for C, then, is by setting $m = m' = I$ in which case $\alpha = \beta = z$.

$$e\sum_k < II\eta|(3x_{zk}^2 - r_k^2)|II\eta >=< II\eta|3I_z^2 - I^2|II\eta > C = I(2I-1)C \quad (2.26)$$

where

$$Q = \sum_k < II\eta|(3z_k^2 - r_k^2)|II\eta > \quad (2.27)$$

we then have

$$C = \frac{eQ}{I(2I-1)} \quad (2.28)$$

The Hamiltonian in its quantum mechanical form is then given by

$$\mathcal{H}_Q = -\frac{eQ}{6I(2I-1)}\sum_{\alpha\beta}V_{\alpha\beta}\left(3\frac{(I_\alpha I_\beta + I_\beta I_\alpha)}{2} - \delta_{\alpha\beta}I^2\right) \quad (2.29)$$

in an arbitrarily oriented axes system $Oxyz$.

Since, as is already known, $V_{\alpha\beta}$ of equation (2.29) are components of a second-rank tensor, the Wigner-Eckart theorem can also be applied to this tensor to find its quantum mechanical equivalent. It is simpler, however, to use the fact that the quantum mechanical operators representing $V_{\alpha\beta}$ can be replaced by their respective expectation values.

The potential, $V_{\alpha\beta}$, can be expressed in an axes system, ***principal axes system*** (PAS) $OXYZ$, such that

$$V_{\alpha\beta} = 0 \quad if \quad \alpha \neq \beta \tag{2.30}$$

in which case the Hamiltonian looks like

$$\mathcal{H}_Q = -\frac{eQ}{6I(2I-1)}\left(V_{XX}(3I_X^2 - I^2) + V_{YY}(3I_Y^2 - I^2) + V_{ZZ}(3I_Z^2 - I^2)\right) \tag{2.31}$$

$V_{\alpha\beta}$ must also satisfy Laplace's equation in the principal axes system, $OXYZ$, which is just rotated with respect to the arbitrary axes system, $Oxyz$, of equation (2.18). $\mathcal{H}_Q$ can then be rewritten as

$$\mathcal{H}_Q = -\frac{eQ}{4I(2I-1)}\left(V_{ZZ}(3I_Z^2 - I^2) + (V_{XX} - V_{YY})(I_X^2 - I_Y^2)\right) \tag{2.32}$$

If we label the principal axes system such that

$$|V_{ZZ}| \geq |V_{XX}| \geq |V_{YY}| \tag{2.33}$$

and define the following variables

$$eq = V_{ZZ} \tag{2.34}$$

$$\eta = \frac{V_{XX} - V_{YY}}{V_{ZZ}} \tag{2.35}$$

$$0 \leq \eta \leq 1 \tag{2.36}$$

then

$$\mathcal{H}_Q = -\frac{e^2qQ}{4I(2I-1)}\left((3I_Z^2 - I^2) + \eta(I_X^2 - I_Y^2)\right) \tag{2.37}$$

where η is an index of asymmetry. The larger η the larger the deviation of the electrostatic potential from symmetry. In case of at least cubic symmetry $V_{XX} = V_{YY} = V_{ZZ} = 0$ and $\eta = 0$, the quadrupole interaction vanishes. In case of cylindrical symmetry $V_{XX} = V_{YY}$ and $\eta = 0$, only V_{ZZ} of the quadrupolar interaction survives.

Since spin-1, $I = 1$, is of interest in this case, the nuclear quantum number assumes $2I + 1$ possible values according to equation (2.25)

$$m = +1,\ 0,\ -1 \tag{2.38}$$

The respective quadrupolar interaction energy levels are given by

$$E_{+1} = \frac{e^2qQ}{4} \tag{2.39}$$

$$E_0 = -\frac{e^2qQ}{2} \tag{2.40}$$

$$E_{-1} = \frac{e^2qQ}{4} \tag{2.41}$$

The allowed transitions are given by selection rule

$$\Delta m = \pm 1 \tag{2.42}$$

leading to contributions to resonance energies

$$\hbar\omega_{Q+} = E_{+1} - E_0 = \frac{3e^2qQ}{4} \tag{2.43}$$

$$\hbar\omega_{Q-} = E_0 - E_{-1} = -\frac{3e^2qQ}{4} \tag{2.44}$$

each of which can be observed as a line on each side of a reference frequency and equally spaced. If we define a variable

$$\omega_Q = \frac{3}{4}\frac{e^2qQ}{\hbar}$$

then the spacing between the two lines, the *quadrupolar splitting* is given by

$$\frac{2}{3}\omega_Q$$

The term

$$\frac{e^2qQ}{\hbar} \tag{2.45}$$

is *the static quadrupolar coupling constant* between an electric quadrupole moment and an electric field gradient.

The Hamiltonian in the principal axes system looks like

$$\mathcal{H}_Q = -\frac{\omega_Q}{3}\left(3I_Z^2 - \mathbf{I}\bullet\mathbf{I}\right) \tag{2.46}$$

$V_{\alpha\beta}$ and $Q_{\alpha\beta}^{(op)}$ are second-rank tensors in cartesian representation and thus they are reducible - that is each such tensor decomposes into objects that transform differently under rotations. Rotation operations then, become rather complicated. It is more convenient to reduce a cartesian tensor in such a case into irreducible *spherical tensors* which transform like spherical harmonics. In fact, $Q_{\alpha\beta}$ has already been reduced in such a manner in equation (2.29) due to equation (2.13). All five independent components of such a spherical tensor transform the same way under rotation. Rotation operations then become much simplified.

The components of a second-rank spherical tensor in an arbitrary axes system $Oxyz$ are given by

$$V^{(2,\pm2)} = \sqrt{\frac{1}{6}}(V_{xx} - V_{yy} \pm i2V_{xy}) \tag{2.47}$$

$$V^{(2,\pm 1)} = \pm\sqrt{\frac{2}{3}}(V_{xx} \pm iV_{xy}) \tag{2.48}$$

$$V^{(2,0)} = V_{zz} \tag{2.49}$$

Equation (2.23) in terms of spherical tensors then, looks like

$$\mathcal{H}_Q = \sum_{p=-2}^{2} V^{2,p} Q^{2,-p} \tag{2.50}$$

and in the principal axes system we have

$$V_{PAS}^{(2,\pm 2)} = \sqrt{\frac{1}{6}}(V_{XX} - V_{YY}) = \sqrt{\frac{1}{6}}\,\eta V_{ZZ} \tag{2.51}$$

$$V_{PAS}^{(2,\pm 1)} = 0 \tag{2.52}$$

$$V_{PAS}^{(2,0)} = V_{ZZ} \tag{2.53}$$

$$V^{(2,0)} = \sum_{p=-2}^{2} D_{p,0}^{(2)} V_{PAS}^{(2,p)} \tag{2.54}$$

where $D_{p,0}^{(2)}$ is a Wigner rotation matrix that has the simplest functional form in the principal axes system.

$$V^{(2,0)} = \frac{1}{2}\left(3\cos^2\theta - 1\right) V_{ZZ} + \frac{1}{2}\sin^2\theta\cos 2\phi(V_{XX} - V_{YY}) \tag{2.55}$$

$$V^{(2,0)} = \frac{1}{2}V_{ZZ}\left((3\cos^2\theta - 1) + \eta\sin^2\theta\cos 2\phi\right) \tag{2.56}$$

To first-order we have

$$\mathcal{H}_Q = \frac{eQ}{4I(2I-1)} V^{(2,0)}\left(3I_z^2 - I(I+1)\right) \tag{2.57}$$

$$= \frac{eQ}{4I(2I-1)} V_{ZZ}\left(\frac{(3\cos^2\theta - 1)}{2} + \frac{\eta\sin^2\theta\cos 2\phi}{2}\right)\left(3I_z^2 - I(I+1)\right) \tag{2.58}$$

$$= \frac{\omega_Q}{3}\left(\frac{(3\cos^2\theta - 1)}{2} + \frac{\eta\sin^2\theta\cos 2\phi}{2}\right)\left(3I_z^2 - I(I+1)\right) \tag{2.59}$$

if η vanishes, as in the case of a cylindrical electrostatic potential, then we get

$$\mathcal{H}_Q = -\frac{\omega_Q}{3}\left(\frac{3\cos^2\theta - 1}{2}\right)\left(3I_z^2 - I(I+1)\right) \tag{2.60}$$

where now energy levels and lines splitting depend on the orientation of each nucleus with respect to the choice of a nonprincipal axes system given by polar angles (θ, ϕ). The choice of the z-axis is usually along an external field (static magnetic field). If the nuclei are randomly oriented as in a crystalline then a *powder pattern* emerges, as we shall see later.

2.3.2 The Equation Of Motion

At this point we have the LVN equation of motion (2.16) and the quadrupolar Hamiltonian equation (2.60) ($\mathcal{H}_Q$ hereafter) at our disposal. To solve this equation of motion we must express its Hamiltonian and density matrix operator in terms of a convenient orthonormal basis set. Such a set has $(2I+1)^2 - 1$ combinations of spin operators. Where I is a spin number. In our case $I = 1$ and so the basis set has $(3^2 - 1)$ eight combinations of orthonormal spin operators. Such a basis set spans an eight dimensional spin operator space. For our purpose, as we shall see shortly, a convenient choice of cartesian basis operators is given by

$$\begin{gathered} I_x \; ; \; I_y \; ; \; I_z \\ Q_x = I_z I_x + I_x I_z \; ; \; Q_y = I_z I_y + I_y I_z \; ; \; Q_z = 3I_z^2 - \mathbf{I} \bullet \mathbf{I} \\ D_x = I_x^2 - I_y^2 \; ; \; D_y = I_x I_y + I_y I_x \end{gathered} \tag{2.61}$$

and the Hamiltonian of the noninteracting spin-1 ensemble is given by

$$\mathcal{H} = -\frac{\omega_Q}{3}(3I_z^2 - \mathbf{I} \bullet \mathbf{I}) - \omega_1 I_x \tag{2.62}$$

$$= -\frac{\omega_Q}{3}Q_z - \omega_1 I_x \tag{2.63}$$

In this case a set of eight cartesian spin operators is divided into two subsets [1] according to their parity with respect to spin inversion operation

$$\hat{a}_1 = I_z \tfrac{1}{\sqrt{2}} \; ; \; \hat{a}_2 = I_y \tfrac{1}{\sqrt{2}} \; ; \; \hat{a}_3 = Q_x \tfrac{1}{\sqrt{2}} \; ; \; \hat{a}_4 = D_y \tfrac{1}{\sqrt{2}}$$

(2.64)

$$\hat{b}_1 = D_x \tfrac{1}{\sqrt{2}} \; ; \; \hat{b}_2 = Q_y \tfrac{1}{\sqrt{2}} \; ; \; \hat{b}_3 = I_x \tfrac{1}{\sqrt{2}} \; ; \; \hat{b}_4 = Q_z \tfrac{1}{\sqrt{6}}$$

(2.65)

and the Hamiltonian (2.63) is even since it is given in terms of the even basis subset $\{\hat{b}_i\}$ only:

$$\mathcal{H} = -\omega_Q \sqrt{\frac{3}{2}} \hat{b}_4 - w_1 \sqrt{2} \hat{b}_3 \tag{2.66}$$

The density matrix on the hand is expanded in terms of subset (2.64) and subset(2.65) as follows:

$$\rho(t) = a_0 1 + \sum_{i=1}^{4} a_i(t)\hat{a}_i + \sum_{j=1}^{4} b_j(t)\hat{b}_j \tag{2.67}$$

where a_i and b_j are the coefficients of the density matrix. In the course of solving the LVN equation the even Hamiltonian (2.63) through its commutation relations with the density matrix decomposes the eight dimensional spin operator space into two subspaces of four dimensions. The LVN equation in its matrix form appears block diagonalized where each block is a four by four submatrix.

$$\begin{pmatrix} \dot{\mathbf{a}} \\ \dot{\mathbf{b}} \end{pmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{bmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} \tag{2.68}$$

a and **b** are column vectors of four elements each; **A** and **B** are the four by four matrices.

2.3.3 A Solution In A New Matrix Form

The block diagonalized equation of motion (2.68)can be simplified by rearranging it into two separate equations of motion - one equation in each four by four subspace.

The first equation of motion is given by

$$\dot{\mathbf{a}} = \mathbf{A}\mathbf{a} \tag{2.69}$$

with its solution is given by

$$\mathbf{a}(t) = \exp(At)\mathbf{a}(0) \tag{2.70}$$

and the second equation of motion which is given by

$$\dot{\mathbf{b}} = \mathbf{B}\mathbf{b} \tag{2.71}$$

with its solution given by

$$\mathbf{b}(t) = \exp(Bt)\mathbf{b}(0) \tag{2.72}$$

The solutions of the eight dimensional equation of motion (2.16) is a sum of solution(2.70) and solution(2.72) which in matrix form is given by

$$\rho(t) = \begin{bmatrix} \exp(\mathbf{A}t) & \mathbf{0} \\ \mathbf{0} & \exp(\mathbf{B}t) \end{bmatrix} \begin{pmatrix} \mathbf{a}(0) \\ \mathbf{b}(0) \end{pmatrix} \tag{2.73}$$

where $\mathbf{a}(0)$ and $\mathbf{b}(0)$ are the initial values of column vectors $\mathbf{a}$ and $\mathbf{b}$ with four elements each.

The evolution of the density matrix is now easy to manipulate. For example if the initial density matrix is such that

$$a_1 = 1 \tag{2.74}$$

and the Hamiltonian given by equation(2.66), then the entire density matrix evolves in the subspace which is spanned by the basis subset of the odd operators $\{\hat{a}\}$. One could be content with equation (2.73) as the solution and would consider to stop here. However, equation(2.73) can be simplified further by manipulating matrices **A** and **B** (see appendix B).

2.4 Spin-1 Evolution In Successive Stages

The problem of simplifying the spin-1 evolution of successive stages is not unique to a spin-1 system. Historically however, this problem was solved for a spin-$\frac{1}{2}$ system which is simpler[2]. In the case of a spin-1 system a different strategy is required to recast spin-1 evolution in a form analogous to that of spin-$\frac{1}{2}$. The benefits of the new form of the spin-1 evolution are shown in this chapter.

2.4.1 Problems of Successive Rotations

Before discussing the solution to successive rotations, it is perhaps better to prepare the background by discussing the problems first. In NMR Spectroscopy, the density matrix evolves in successive stages. The calculation of the spin-1 evolution of the density matrix by the usual spin operators is quite complicated. Expressions for the spin-1 evolution are often too convoluted. New insights into the nature of the spin-1 evolution may not be possible. In some cases, computer based numerical techniques may be used to average out unwanted components of spin-1 evolution. Although such methods offer new possibilities, the time they consume limits their use far too much. Analytical methods, whenever possible, prove to be much more advantageous in simplifying spin-1 evolution. It is shown that in the case of a spin-1 system expressing the spin-1 evolution in simple closed-form expressions leads to a new insight. Furthermore, these expressions can be

used in constructing simpler computer simulations.

2.4.2 Spin-1 Evolution In Terms of Quaternions

The fact that matrices **A** and **B** of equation(2.73) are antisymmetric associates them with rotation operations [3]. It was shown [1] that spin-1 evolution of a spin-1 system in terms of the rotation generators (J_i and K_i) of these matrices, can be given by simple closed-form expressions (see appendix B).

The exponential form of a rotation operator is now in four dimensions [1] whether $\exp(\Phi\hat{n} \bullet J)$ in J, or $\exp(\Psi\hat{n} \bullet K)$ in K. Any rotation operator can be simplified significantly by using product by exploiting the properties of the rotation generators. A simplified rotation operator in J is given by equation (2.75); a simplified rotation operator in K is given by equation (2.76). These last two equations have the same functional form. So rotation operations in J and in K yield the same result. Discussing one type of rotation entails no loss of generality.

$$\exp(\Phi\hat{n} \bullet J) = \cos(\Phi/2)1 + 2(\hat{n} \bullet J)\sin(\Phi/2) \tag{2.75}$$

$$\exp(\Psi\hat{m} \bullet K) = \cos(\Psi/2)1 + 2(\hat{m} \bullet K)\sin(\Psi/2) \tag{2.76}$$

Consider for example, the following two general successive rotations in four dimensions, their product is given by

$$\begin{aligned}\exp(\Phi_2\hat{n}_2 \bullet J)\exp(\Phi_1\hat{n}_1 \bullet J) &= (\cos(\Phi_2/2)1 + 2(\hat{n}_2 \bullet J)\sin(\Phi_2/2)) \\ &\times (\cos(\Phi_1/2)1 + 2(\hat{n_1} \bullet J)\sin(\Phi_1/2))\end{aligned} \tag{2.77}$$

by using equation(2.75). However, by comparing equation(2.77) and equation(2.75) it is noticed that already for two successive rotations the resultant has a functional form which looks complicated. This functional form becomes even more complicated as the number of successive rotations increases. This problem is significant in NMR Spectroscopy where

spin-1 evolution may be described by successive rotations which can number in the tens or even in the hundreds. Equation(2.77) is a product of *quaternions*. If parameterized in terms of the ER-parameter set $\{\lambda, \Lambda\}$, this product can be reduced into a single resultant quaternion by using two composition rules (2.78) and (2.79) of appendix A.
where Φ is an angle of rotation and $\hat{\mathbf{n}}$ is its axis.

$$\lambda = \lambda_2\lambda_1 - \Lambda_2 \bullet \Lambda_1 \tag{2.78}$$

$$\Lambda = \lambda_2\Lambda_1 + \lambda_1\Lambda_2 + \Lambda_2 \times \Lambda_1 \tag{2.79}$$

It is possible to apply equations(2.78) and (2.79) to *any* number of successive rotations.

$$\prod_{i=1}^{n} \exp(\Phi_i \hat{n}_i \bullet J) = \lambda_{1..nJ} + 2\Lambda_{1..nJ} \bullet J \tag{2.80}$$

Procedure for Simplifying n Successive Rotations

- (i)Given n successive rotations
- (ii)Take the product of the first two rotation operators
- (iii)Transform product of step(ii) according to equation(2.75)
- (iv)Apply compositions rules (2.78) and (2.79) to the result of step (iii). (The result is a single rotation operator) If you are left with a single rotation operator then you have the final result.
- (v)Take the next rotation and transform it according to equation (2.75)
- (vi)Take the product of the result of step (iv) and the result of step(v) and proceed to step (iv).

Note that since each iteration reduces the number of rotations by one, the procedure runs its course when a single quaternion is left.

2.5 The Density Matrix In Quaternion Formalism

Quaternions were first applied in quantum mechanics to a noninteracting spin-$\frac{1}{2}$ system [2]. The Hamiltonian of such a system is linear in terms of spin operators I_x, I_y, and I_z

$$\mathcal{H} = -\omega_z I_z - \omega_1 I_i \tag{2.81}$$

where $i = x, y$.

The propagator of the density matrix as a result of a single pulse of duration τ, for example $S_x(\tau)$, is given by

$$\exp(-i\tau(\omega_z I_z + \omega_1 I_x)) = \exp(-i\omega\tau\hat{n} \bullet I)\ (-i\tau(\omega_z I_z + \omega_1 I_x)) = \exp(-i\omega\tau\hat{n} \bullet I) \text{ where,}$$
$$\omega = \sqrt{(\omega_z^2 + \omega_1^2)},$$

is a quaternion (see appendix A). In fact, *any* spin system with Hamiltonian(2.81) has a propagator which is a quaternion [4]. Thus quaternions can be used to express spin-1 evolution in three dimensions for arbitrary spin number.

Expressing the spin-1 evolution of a noninteracting spin-1 system in terms of quaternions is more involved. A nonspherical symmetry of the electric charge distribution of the nuclei causes an interaction between an electric quadrupole moment and gradient of a surrounding electric field. The Hamiltonian of such a system is quadratic in terms of the spin operators I_x, I_y, and I_z

$$\mathcal{H} = -\frac{\omega_Q}{3}(3I_z^2 - \mathbf{I} \bullet \mathbf{I}) - \omega_1 I_i \tag{2.82}$$

the corresponding propagator is

$$\exp(-\tau(\frac{\omega_Q}{3}(3I_z^2 - \mathbf{I} \bullet \mathbf{I}) - \omega_1 I_i)) \tag{2.83}$$

where $i = x$, y, is *not* a quaternion. The basis set spans(2.61), as we already know, an eight-dimensional operator space. It is possible, however, to decompose this eight-dimensional operator space into two separate subspaces of four dimensions. The corresponding propagator, in terms of four dimensional rotation generators J_i and K_i, is a quaternion (see appendix B).

The spin-1 evolution, following a quadrupole pulse sequence, takes place in a form of spin packets that precess in a plane. Such a plane is defined by two orthogonal operators: one that corresponds to an observable single-quantum coherence, another that corresponds to an unobservable single quantum coherence. The Hamiltonian during that time is quadrupolar with axial symmetry. The symmetry is about the z-axis of a principal axis system.

It is desirable to maximize the in-phase. In such a case the observable single-quantum coherence gives a more complete description of the spin-1 evolution. In fact, if the spin-1 evolution is entirely in terms of the in-phase then the description of the spin-1 evolution as an observable is complete.

By using quaternions the in-phase and the anti-phase can each be expressed as a sum of two closed-form expressions: one that results from spin packets that refocus by the quadrupole pulse sequence (*echo* part) and another that results from spin packets that do not refocus (*feed − through* part). Each echo part originates halfway through the refocusing pulse ($S_y(T)$) while each feed-through part originates halfway through the first pulse ($S_x(T)$). If the spin-1 evolution is completely observable then the echo part of the unobservable anti-phase is identically zero (as we shall see in chapter 3)

The in-phase is given by (appendix B)

$$\begin{aligned} < I_y > &= (\Lambda_2^2 + \Lambda_3^2)\{\Lambda_2 \cos\omega_Q\delta - \Lambda_3 \sin\omega_Q\delta\} \\ &+ ((\lambda^2 - \Lambda^2)\{\Lambda_2 \cos\omega_Q\sigma - \Lambda_3 \sin\omega_Q\sigma\} - 2\lambda\Lambda_1\{\Lambda_3 \cos\omega_Q\sigma + \Lambda_2 \sin\omega_Q\sigma\}) \end{aligned} \tag{2.84}$$

The application of equation(2.84) can be extended to a time-varying pulse. By approximating this pulse as a train of many narrow pulses of fixed amplitude and duration, one can apply this equation to a narrow pulse and a cumulative resultant, one at a time. Each time calculating the corresponding ER-parameter set, $\{\lambda, \Lambda\}$, by using the two composition rules of appendix A. The number of $\{\lambda, \Lambda\}$ sets being the number of narrow pulses less one. The total solution then would look like a product of successive operations in time, where each operator stands for the action of one narrow pulse. The number of the operations is equal to the number of the narrow pulses used to approximate the time-varying pulse less one. The time varying pulse is the main theme as we shall see in Chapter 4.

Chapter 3

Pulse Sequences

The frequency range of the *electric quadrupolar interaction* for spin $I = 1$ can be as large as 250 kHz [5, 6]. The uniform excitation of such a broadband spectrum by an on-resonance pulse sequence is a challenging task. Such an excitation is given by the response of each spin packet to that pulse sequence. A spin packet is a group of spins of nuclei which have interactions very similar in value (see equation (2.63)). The angular frequency of such an interaction is labeled ω_Q. A plot of the response as a function of a reduced frequency $\omega_Q/2\omega_1$, where ω_1 is the peak angular frequency of the pulse sequence, reveals the spin-1 response (hereafter *excitation profile*) over a frequency domain of interest. An excitation profile is a theoretical method of simulating the effects (distortions) of a pulse sequence on a spin-1 spectrum. An ideal uniform excitation would then look like a rectangle of unit height.

When applied in an experiment, a pulse sequence could cause a sample to produce a time domain signal known as a *free induction decay* or *fid* in short. A Fourier Transform of the *fid* would produce an ideal spectrum. In such a case, the spectrum would appear with all its features undistorted. By calculating the spin-1 response to a pulse sequence, it is possible to determine how uniformly such a pulse sequence can excite a spin-1 spectrum. The measure of a uniformity in such a case is the bandwidth of the excitation - the larger the bandwidth of the excitaion the more uniform the spin-1 response.

By systematically varying the parameters of the pulse (number of pulses, flip angle of individual pulses in case of a square pulse and a composite pulses or coefficients of

functional form in the case of a shaped pulse) the bandwidth of a uniform excitation a pulse produces in spin-1, can be used to broaden such a bandwidth.

3.1 Square Pulses

3.1.1 The Single Square Pulse

A single 90° square pulse is the simplest way to produce a spin-1 excitation. Such a pulse is symbolized by $S_x(T)$ for a 90° rotation about the x-axis ($S_y(T)$ for a 90° rotation about the y-axis) where T is the duration. An angle of rotation (flip angle) is given by

$$\theta = \omega_1 T \tag{3.85}$$

Due to the finite width of this pulse, spin packets with quadrupole interaction such that ω_Qs comparable to ω_1 or larger, do not experience just a rotation when under the action of the pulse. The quadrupole interaction of such spin packets *during* the application of a pulse is not negligible. Such spin packets have the ability to *precess* (rotate about the z-axis) in the rotating frame even when under the action of a pulse. An excitation profile in this case would be distorted; uniform excitation would be limited to a narrow region around the center [6, 7].

In an experiment, precession during the pulse means that the *echo* too originates during that pulse. The *echo* signal then is not accessible for collection (hereafter aquisition), from its starting point, for the following reasons:

1) during the pulse: the time origin of the *fid*, the peak time, is halfway through the pulse. During that time, the high power output of the amplifier suppresses the *fid*. The receiver, used for the acquisition of the *fid*, is blocked as a protection against that out-put. Thus the detection of the *fid* cannot be done.

2) after the pulse: it takes the output of the amplifier time, to switch off. This time, *falltime*, depends on the technical specifications of the amplifier. It also takes the receiver

time, *receiver recovery time* to recover from the high power output of the amplifier. The probe and tank circuit also take time, *ringdown* time, until remnants of the pulse sequence decay.

All these times add up to a total time, *deadtime*, when the *fid* is occurring but cannot be accessed for acquisition. The Fourier Transform of such an incomplete *fid* would result in a distorted spectrum.

Distortions due to *deadtime* can be reduced by increasing the pulse's power , such that $\omega_1 \gg \omega_Q$, while reducing the duration T (to maintain the 90° rotation condition $\omega_1 T = \pi/2$). In such a case, precession of the spin packets during the pulse decreases. In fact, by keeping this trend of the pulse's power and duration, the spin packets' precession during the pulse can be further reduced. More spin packets are then rotated onto the y-axis by $S_x(T)$. Thus distortions due to *deadtime* are further reduced. The amplitude of the corresponding excitation profile then appears flatter over a wider domain of reduced frequency. That is, the spin-1 excitation becomes more uniform. This way of improving uniform excitation is limited by the maximum peak power of the amplifier.

3.1.2 The Two-Pulse Square Pulse

The two-pulse sequence, the *normal quadrupole pulse sequence* can be used to further improve uniform excitation by avoiding distortions due to *deadtime*. This pulse sequence is represented symbolically by:

$$S_x(T) - t_1 - S_y(T) - t_2 - acquire$$

and is made up of several events as described by figure 3.1 and a list. The *normal quadrupole pulse sequence* differs from the single pulse by a second pulse, the $S_y(T)$ pulse.

The first pulse, $S_x(T)$, causes the spin packets a 90° rotation about $\hat{b}_3$. Following

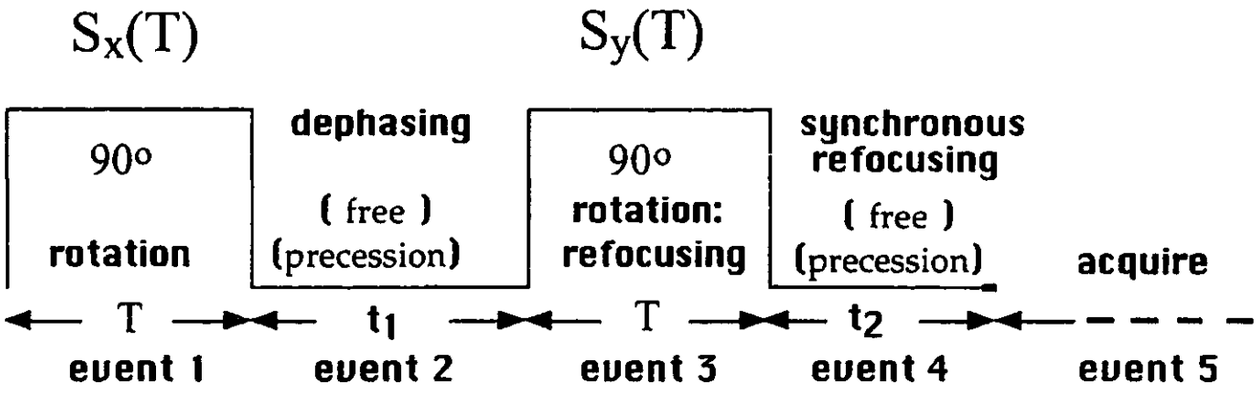


Figure 3.1: This is a pictorial representation of the *normal quadrupole pulse sequence.* Each event describes a stage in the behaviour of the spin packets with respect to time. Event 3 is a refocusing pulse, $S_y(T)$ which by reflecting the spin packets with respect to $\hat{a}_2$ i.e y causes them to precess towards $\hat{a}_2$ in a plane defined by $\hat{a}_2$ and $\hat{a}_3$ during event 4. Then at t_2+T relative to $S_y(T)$, the spin packets align along $\hat{a}_2$. This situation is referred to as *synchronous refocusing.* $\hat{a}_2$ is an observable single-quantum coherence while $\hat{a}_3$ is an unobservable single-quantum coherence since we are dealing with spin $I = 1$.

$S_x(T)$, the spin packets begin to precess away from $\hat{a}_2$ during t_1 - the spin packets *dephase* in a plane defined by $\hat{a}_2$ and $\hat{a}_3$ (see figure 3.2) The second pulse, $S_y(T)$, causes the spin packets a 180° rotation about the y-axis. Following $S_y(T)$, the spin packets begin to precess (in the x-y plane) towards the y-axis in the same sense as during t_1 (see figure 3.3). Then at a later time, $t_2 + T/2$ relative to $S_y(T)$, the spin packets align along the $\hat{a}_2$. They *refocus synchronously.* In an experiment, following $S_y(T)$, an *echo* builds up and then peaks at $t_2 + T/2$ relative to $S_y(T)$. After that, the spin packets cross $\hat{a}_2$ as their precession continues. Maintaining their sense of precession, they now precess

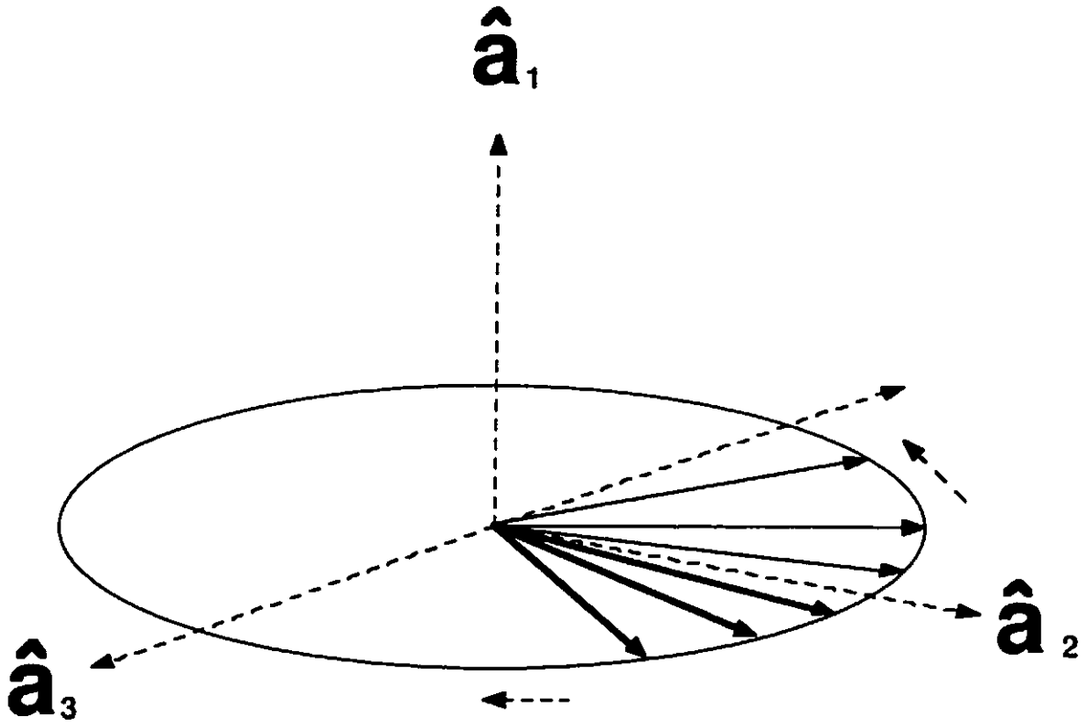


Figure 3.2: This is a snap shot of the spin packets in their free precession under the quadrupole Hamiltonian during t_1. They precess away from the $\hat{a}_2$. This situation is referred to as dephasing.

away from $\hat{a}_2$ and eventually scatter in the plane completely - the spin packets begin to *dephase* again. In an experiment, this corresponds to the decaying of the *echo*. $S_y(T)$ is referred to as *refocusing pulse.*

The Events of The Normal Quadrupole Pulse Sequence

EVENT 1) $S_x(T)$. This pulse causes the equilibrium spin state ($\hat{a}_1$) a 90° rotation about the $\hat{a}_1$. It is identical in function to the single pulse.

EVENT 2) t_1 is a time when these spin packets scatter radially in a common plane. This is *free precession* time since the spin packets evolve under no external forces. This situation is also referred to as *dephasing*. The plane is defined by

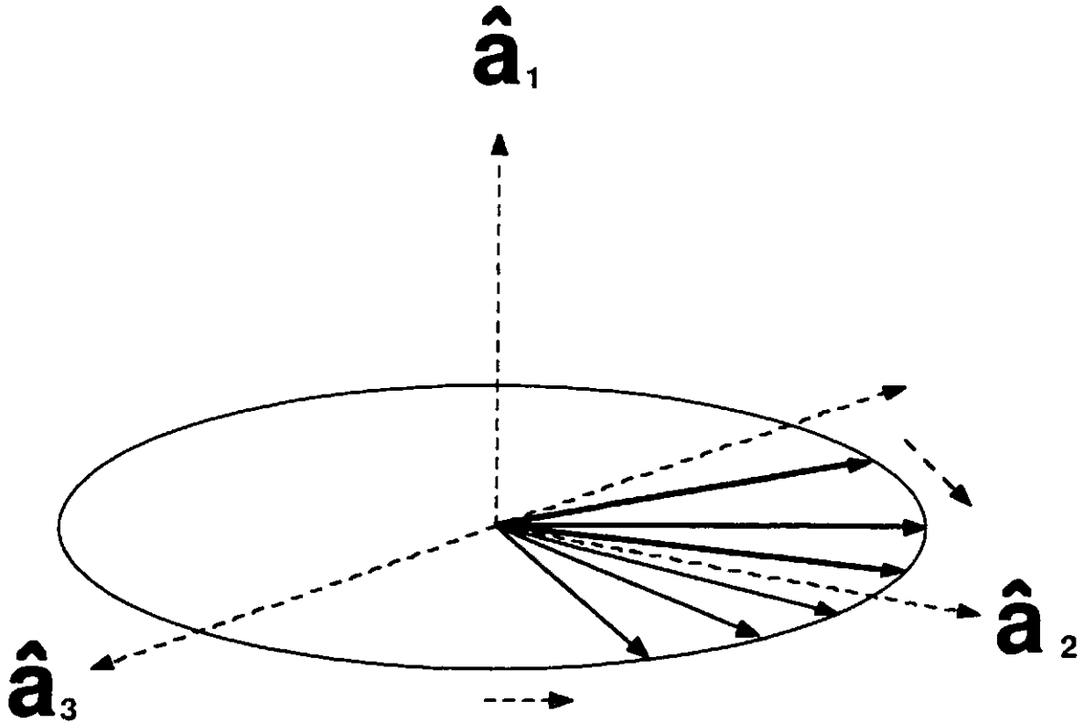


Figure 3.3: This is a snap shot of the spin packets in their free precession under the quadrupole Hamiltonian. They precess, with their sense unchanged, towards $\hat{a}_2$. At time $t_2 + T/2$ relative to $S_y(T)$ they will all align along the $\hat{a}_2$. This situation is referred to as synchronous refocusing

$\hat{a}_3$ and $\hat{a}_2$ which are unobservable and observable single-quantum coherences respectively.

EVENT 3) $S_y(T)$. This pulse reflects the spin packets with respect to $\hat{a}_2$. As a result, the spin packets switch quadrants but remain in the same plane as before. This pulse is referred to as a *refocusing* pulse since it causes the spin packets to reverse as described next.

EVENT 4) t_2 is the time following the application of the *refocusing* pulse. As a result of switching their quadrants but not their sense of precession, the spin packets begin to precess towards $\hat{a}_2$. Then, at $t_2 + T/2$ relative to $S_y(T)$, the

spin packets are oriented along $\hat{a}_2$. This situation is referred to as *synchronous refocusing*. This is when the *echo* peaks.

EVENT 5) *acquire* is when the data of the signal generated by the spin packets is collected. This signal is the echo. A *complete echo* can be acquired well after *deadtime* and be Fourier Transformed. The resulting spectrum is free of *deadtime* distortions.

The peak time, of the echo occurs well after *deadtime*, at $t_2 + T/2$ with respect to $S_y(T)$. A *complete echo* can then be acquired from the peak time onward. A Fourier Transform of such an echo would yield a spectrum free of distortions due to *deadtime*.

While avoiding distortions due to *deadtime*, the *normal quadrupole* pulse sequence introduces more distortions due to finite pulse width. The precession of spin packets, with quadrupole interaction such that ω_Q is comparable to or greater than ω_1, during the rotation pulse $S_x(T)$ and the refocusing pulse $S_y(T)$, is not negligible. As a result, not all the spin packets are rotated and refocused by the normal quadrupole pulse sequence. Distortions due to finite width of $S_x(T)$ symbolized by $D_1(\omega)$ and distortions due to finite width of $S_y(T)$ symbolized by $D_2(\omega)$ must be considered [6]. The combined finite pulse-width distortion of the normal quadrupole pulse sequence is $D_1(\omega)D_2^2(\omega)$. Thus distortions due to finite pulse width are *compounded*. As in the case of the single pulse, distortions due to finite pulse width can be reduced by increasing the power of each pulse while reducing the duration of each pulse.

3.2 Composite Pulses

The *composite pulse sequence* can be used to obtain a uniform excitation which may be superior to that of the normal quadrupole pulse sequence. The sequence of events that makes up a composite pulse sequence is the same as in the case of the normal quadrupole

pulse sequence. The 90° pulse and the refocusing pulse are made up of contiguous rectangular pulses whose individual phase and angle of rotation may differ (see figure 3.4). Viewed this way, a rectangular pulse is made up of contiguous pulses of identical phases. The composite pulse sequence offers three new degrees of freedom: the number of contiguous pulses, the phase of each such pulse, and the angle of rotation of each such pulse. An excitation profile now depends not only on the pulse power but also on these three new degrees of freedom. The sum total of the action of the contiguous pulses which make up each composite pulse in the sequence is referred to as *selfcompensation*. Each of the contiguous pulses by itself would not be a good choice for the normal pulse sequence. This very fact, not being a good performer by itself, is what makes each of these contiguous pulses a member of a well performing composite pulse sequence.

After [8] showing promise in spin-$\frac{1}{2}$, inversion composite pulse sequences were modified for spin-1 excitation by halving the rotation angles of a three-part spin-$\frac{1}{2}$ inversion composite pulse sequence. This choice of composite pulse sequence for spin-1 did not turn out to be a good one. The substantial anti-phase produced by this pulse sequence was eventually found to be associated with phase distortions in the excitation profile. A composite pulse sequence can give rise to *asynchronous refocusing* since refocusing time may vary from one spin packet to another. As a result the phase may vary from one spin packet to another - phase distortions. In such a case it would be hard to determine the echo peak-time. An excitation profile may be misleading or impossible to obtain in terms of the in-phase and and the anti-phase. In an experiment the *fid* would not have a unique peak-time. The *fid* would not be symmetric. A correct spectrum may not be possible to obtain via a Fourier Transform.

The *Transfer Function* method [9], which relies on powder averaging prior to Fourier transformation, incorporates information on phase and amplitude distortions in a *single* spectral profile without depending on echo peak time. Furthermore, this method is not

limited to a particular type of pulse sequence. Its versatility was demonstrated on several composite pulse sequences [7, 10].

Backrotation Operators developed on the assumptions that any pulse sequence can have a unique echo peak time [11]. This assumption proved to be wrong in the case of pulses that were not time-symmetric. Spin packets under the action of such pulses do not focus synchronously.

Clearly, the composite pulse sequence offers an overwhelming selection. A systematic search is needed since uniform excitation may vary with the choice of a composite pulse sequence. A numerical sequence design based on iterations [12] yielded composite pulse sequences which produced improved excitation profiles. Another technique based on Lagrange multipliers [10] and quaternions also yielded composite pulse sequences which produced improved excitation profiles.

There are two types of composite pulse sequences: time-symmetric and nontime-symmetric. Time-symmetric composite pulse sequences are more convenient. The spin packets under the action of a time-symmetric composite pulse sequence refocus synchronously. The unobservable anti-phase is then identically zero - an excitation profile is then free of phase distortions. The observable in-phase gives a complete description of the spin-1 evolution - only amplitude distortions may be present in the excitation profile.

The form of the in-phase, however, confuses the evaluation of the excitation profile. The oscillation of the in-phase, due to unrefocused spin-packets, makes it hard to evaluate the bandwidth for its flat part - its uniform excitation.

The calculation of an excitation profile is further simplified when time-symmetric composite pulse sequences with alternating phases are used (see figure 3.4). For such pulse sequences it is possible to derive closed-form expression for the remaining in-phase by the usual fictitious spin-$\frac{1}{2}$ operators. Furthermore, the closed-form expression can be separated into an echo and a feed-through [7]. Thus the evaluation of an excitation

profile can be made simpler. In the case of other types of composite pulse sequences of interest numerical methods such as density matrix calculations, are the only way to obtain the in-phase and anti-phase since closed-form expressions are not obtainable.

As for the feed-through, it is not only redundant, it is misleading. The amplitude of the feed through is given by

$$A_F(\omega) = 1 - A_E(\omega) \tag{3.86}$$

([7]). where $A_E(\omega)$ and $A_E(\omega)$ are the amplitudes of the echo and the feed-through respectively. So knowing the amplitude of the echo renders that of the feed-through redundant. The oscillatory behavior of the feed-through, its periodicity, stems from its different origin time – effectively twice the interpulse spacing before the echo peak time. So the oscillations are actually irrelevant to the performance of the pulse sequence. By obscuring the echo these oscillations can result in a confusing excitation profile. So it is better to suppress the feed-through and rely on the echo, for the in-phase, as much as possible [13]

Although, as stated above, it is possible to separate the echo from the feed-through in closed-form expressions, the functional form of these expression, for more intricate pulse sequences, is still too complex. Such complexity may obscure further insight to the relationship between the structure of a pulse sequence and its corresponding excitation profile.

It is possible to derive closed-form expressions for the echo and the feed-through that are separate as well as simple. The method which makes such simple expressions possible is based on classical geometry instead of the conventional spin-operator algebra with its complementary matrix representation.

To be more specific, the quantum mechanical unitary operator for rotations is replaced by *quaternion* [2]. By exploiting two simple composition rules for quaternions it is possible

to express a product of quaternions as a single quaternion (see appendix A). These composition rules express the product of any number of successive rotations of any kind as an equivalent single rotation through some angle about some axis - $(\Phi, \hat{n})$. Furthermore, any computation of spin-1 evolution will not be nearly as time consuming as the matrix-based density matrix computation since the new method does not depend on repetitive averaging to eliminate the feed-through (see § 4.2) in order to obtain the echo of the in-phase. Instead, the composition rules offer a much faster and simpler alternative by obtaining each , the echo and the feed-through, *directly* (see appendix B). Once obtained , these expressions can be readily applied to a pulse sequence of any type.

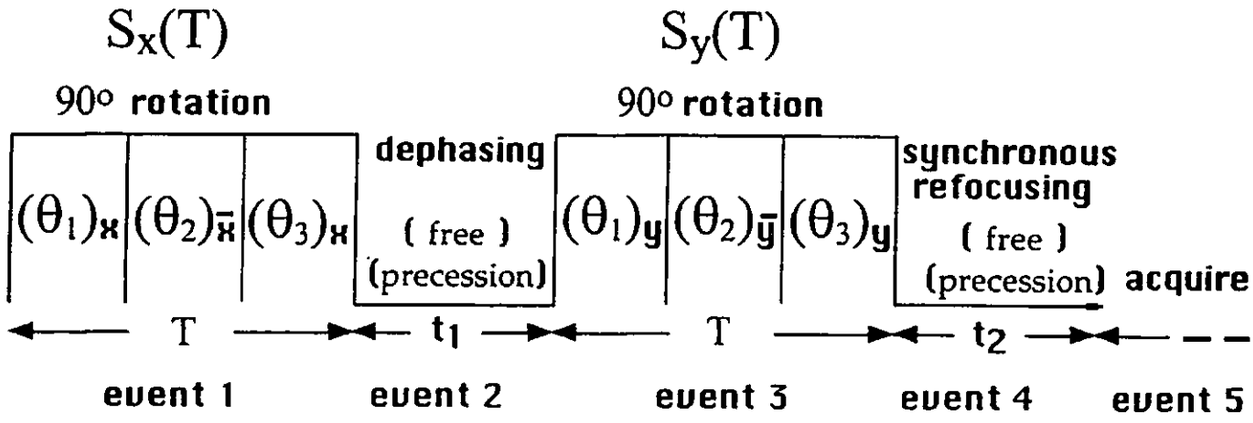


Figure 3.4: This is a pictorial representation of a composite pulse. Notice that the sequence of events is the same as that of the normal quadrupole pulse sequence. Notice the contiguous pulses that make up $S_x(T)$ and $S_y(T)$. These pulses are time-symmetric since the excitation profile is free of phase distortions, the in-phase gives a complete description of spin-1 evolution via the excitation profile. These pulses also have relative phase changes of 180° (hereafter alternating phases). For $S_x(T)$, a phase of 0° is labeled x; a phase of 180° is labeled $\bar{x}$. For $S_y(T)$, a phase of 90° is labeled y; a phase of 270° is labeled $\bar{y}$. The angle of rotation of each of the contiguous pulses is given by θ_i where i is a serial number. This composite pulse is time-symmetric with alternating phases. The echo and the feed-through that make up the in-phase can be disentangled and expressed in closed-form. By using quaternions that are ER parameterized these closed-form expressions can be simplified to great extent (see appendices A and B).

Chapter 4

Shaped Pulses: Theory

4.1 Introduction

A shaped pulse is a pulse with an amplitude which varies with time. Such a pulse is also known as an *amplitude modulated pulse*. Unlike a rectangular or a composite pulse, the envelope of a shaped pulse is smooth and the amplitude may assume negative values. The rotation angle of a shaped pulse is determined by the pulse's area i.e. its integral with respect to time. The fact that the amplitude of a shaped pulse is a new degree of freedom opens new venues for exploration. As in the case of the composite pulse, uniform broadband excitation by a shaped pulse is not limited by maximum peak power a spectrometer has to offer, but by the way the pulse's amplitude varies with time i.e. the shape of the pulse's envelope.

From a computational standpoint, a shaped pulse must be discreted. For that purpose the shaped pulse is divided into N pulse-strips, each with the same duration in time(see figure 4.1). In the limit that the time duration of these pulse-strips approaches zero ($N \rightarrow \infty$), the variation of the amplitudes from one pulse-strip to another will trace out the envelope of the amplitude-variation.

If the amplitude of each pulse-strip (f_i) is normalized to a certain frequency ω_1,

$$f_i = \frac{f(t_i)'}{f'_{max}}\omega_1$$

then the total rotation angle ($\pi/2$) equals the net area, S, of the shaped pulse

$$S = \sum_{i=1}^{N} \delta t \cdot f_i = \delta t \sum_{i=1}^{N} f_i = \delta t \cdot f = \theta = \pi/2 \tag{4.1}$$

where

$$f_i \equiv f(t_i) \tag{4.2}$$

$$t_i = i \cdot \delta t \tag{4.3}$$

and

$$\delta t = T/N \tag{4.4}$$

is the duration of each pulse-strip. The total rotation angle due to the shaped pulse is assumed nominal. This is to say, an angle of rotation imparted on the spin packets while neglecting rotation effects due to other interactions such as the *electric quadrupole interaction*. The total duration of a shaped pulse is then given by

$$T = N \cdot \delta t \tag{4.5}$$

and the rotation angle for each pulse strip is

$$\delta\theta = \delta t \cdot \omega_i \tag{4.6}$$

Negative amplitude modulations in the shape of the pulse can be treated from a computational point of view as 180° phase shifts, since a rotation about an axis $\hat{n}$ through a negative angle Φ (CW) can be expressed as a rotation about an anti parallel axis $\hat{n}$', through a positive angle Φ (CCW) as follows:

$$\exp((-\Phi)\hat{n} \bullet I) = \exp(\Phi(-\hat{n}) \bullet I) \tag{4.7}$$

$$= \exp(\Phi\hat{n}' \bullet I) \tag{4.8}$$

where I is a rotation generator in terms of J_i or K_i (see appendix B) and the axis of rotation lies in the x-y plane: alternating between the x-axis and (-x)-axis for $S_x(T)$ on the one hand, and between the y-axis and (-y)-axis for $S_y(T)$ on the other hand. An axis of rotations is then given by

$$\begin{aligned} -\hat{n} &= -(\sin(\theta)\cos(\phi), \sin(\theta)\sin(\phi), 0) && (4.9) \\ &= (\sin(\theta)\cos(\phi + 180°), \sin(\theta)\sin(\phi + 180°), 0) && (4.10) \\ &= \hat{n}' && (4.11) \end{aligned}$$

where ϕ is the (nominal) axis of rotation (phase) and θ is the (nominal) angle of rotation (flip angle). In that sense a shaped pulse may be regarded as phase alternating.

A quadrupole pulse sequence,

$$S_x(T) - t_1 - S_y(T) - t_2 - acquire$$

can be constructed with S_i ($i = \{x, y\}$) as a 90° (or $\pi/2$) shaped pulse and T the duration of that pulse (see figure 4.2).

The sequence of events that makes up the shaped pulse is the same as that of composite and the normal quadrupole pulse sequences. The list of events given for the normal quadrupole sequence can be followed to the letter. As in the case of the normal quadrupole pulse sequence and the composite pulse sequence, the evolution of the spin packets following a shaped pulse sequence is expressed in terms of an observable in-phase and an unobservable anti-phase components [7, 1, 13]

$$< I_y(t, \omega_Q) >_{in-phase} = E + F \qquad (4.12)$$

and

$$< I_y(t, \omega_Q) >_{anti-phase} = E' + F' \qquad (4.13)$$

In these expressions, (4.12) and (4.13) , each component is conveniently separated into an echo signal E or E' which originate halfway through the refocusing pulse $S_y(T)$, and a transient feed-through signal F or F' apparently originate halfway through the first pulse $S_x(T)$. The latter, feed-through, signals are due to the fact that off-resonance spin packets do not experience full rotation angles for the refocusing pulse $S_y(T)$. Thus feed-through signals keep precessing. In the in-phase/anti-phase excitation profiles, the feed-through contributions are manifested as superimposed off-resonance oscillations, whose periodicity varies with the interpulse delay t_1 [7, 13]. Since these feed-through contributions are neither informative nor useful from the point of view of pulse design [13], it is of some advantage to focus on the echo contribution alone if possible.

At least for phase-alternating composite spin-1 pulses, fictitious spin-1/2 formalism [7] or quaternion formalism [1] have been used to derive explicit closed-form expressions for the echo (E,E') and feed-through (F,F') terms in equations (4.12) and (4.13). The advantage of these expressions is that they disentangle the feed-through contributions from the echo (see appendix B), a direct disentanglement not possible if density matrix evolution techniques [14, 11, 8] are used to calculate the in-phase or anti-phase. Once the feed-through signals are disentangled, it is then possible to carry out numerical optimization using in-phase/anti-phase excitation profiles unobscured by feed-through oscillations. The clear advantages of such a disentanglement have already been well-established in the analysis [7, 1] and design [10] of composite pulse sequences.

It is possible to take advantage of a similar disentanglement for the design of shaped pulses [15], since either of the fictitious spin-1/2 or quaternion formalism previously used for the analysis of phase-alternating composite pulses could be adapted for the analysis and design of shaped pulses. (see figure 4.7). It is done so in this study, choosing an ER spin-1 pulse parameterization [1] used previously for the analysis and design of composite pulses [10].

4.2 Disentangling the Echo and the Feed-through by Averaging

However, in order to clarify the significance of the feed-through disentanglement discussed above, and to point out the particular advantages of the ER-parameterization, we consider first a indirect numerical procedure for using density matrix evolution techniques to effect the same feed-through disentanglement.

The in-phase and anti-phase plots of figure 4.3 and figure 4.4 (both courtesy of N. J. Tagg) show how an averaging procedure may be used to suppress the feed-through oscillations. For any pulse sequence, we calculate the following averages for the in-phase ($< I_{y1}(t_1, \omega_Q) >$) and anti-phase ($< I_{y2}(t_1, \omega_Q) >$) profiles at the time of the echo:

$$\overline{< I_{y1}(\omega_Q) >} = \sum_{t_1} < I_{y1}(t_1, \omega_Q) > \tag{4.14}$$

$$\overline{< I_{y2}(\omega_Q) >} = \sum_{t_1} < I_{y2}(t_1, \omega_Q) > \tag{4.15}$$

As described previously [7, 13], a fictitious spin-1/2 formalism was used to compute each of the in-phase and anti-phase profiles required for these averages, using density matrix evolution techniques. For a given pulse spacing t_1, the echo time was determined from the time-domain signal response of a quadrupole echo sequence to a uniform distribution of frequencies, as described previously [13]. Since the echo signals (E, E') depend only on the parameters of the pulses used, and not on the pulse spacing, they are invariant to a change in pulse spacing, and they accumulate in the averaging. On the other hand, the feed-through signals (F, F') clearly do depend on the pulse spacings, as illustrated in figures 4.3 and 4.4, they destructively interfere in the averaging. We have found that 100 different pulse spacings evenly distributed between 5 μs and 95 μs are sufficient to completely average out the feed-through oscillations, leaving the echo signals (E, E') as the only contribution which survive the averaging (see figure 4.3(E) for E and figure 4.4(E) for E'). It should be noted that a similar averaging procedure was used to

suppress the feed-through signal in quadrupole echo time-domain data [17].

For example, a comparison between the averaged profile in figure 6.4(C) and that calculated from closed-form expressions [7] in figure 4.6(B) for the echo signal of the same QUASH pulse illustrates the reliability of the averaging procedure. Although the efficacy of this averaging procedure suffice to illustrate the nature of the feed-through signal, it is clearly an impractical scheme for shaped pulse design.

4.3 Disentangling the Echo and Feed-Through Analytically

For this purpose, we turn instead to closed-form expressions [1] for the echo and feed-through signals which rely on a ER-parameterization of shaped pulses. If the first and second rotations are given by $R_1(\lambda_1, \Lambda_1)$ and $R_2(\lambda_2, \Lambda_2)$ respectively, the quaternion expression for the compounding formula is [1, 2]

$$R_2(\lambda_2, \Lambda_2)R_1(\lambda_1, \Lambda_1) = R(\lambda, \Lambda) \tag{4.16}$$

where

$$\lambda = \lambda_1\lambda_2 - \Lambda_1 \bullet \Lambda_2, \tag{4.17}$$

$$\Lambda = \lambda_1\Lambda_2 + \lambda_2\Lambda_1 + \Lambda_2 \times \Lambda_1, \tag{4.18}$$

$$\lambda_i = \cos(\Phi_i/2), \qquad i = 1, 2. \tag{4.19}$$

$$\Lambda_i = \hat{\mathbf{n}}_i \sin(\Phi_i/2), \qquad i = 1, 2. \tag{4.20}$$

Rotations are about the axes of $\hat{\mathbf{n}}_i$ through angles of $\Phi_i = \omega_e t_i$, where the effective angular frequency $\omega_e = \sqrt{\omega_Q^2 + 4\omega_1^2}$ during a rf pulse of duration t_i is determined by the interaction Hamiltonian

$$\mathcal{H} = -\frac{\omega_Q}{3}(3I_z^2 - \mathbf{I}^2) \mp \omega_1 I_x \tag{4.21}$$

The uppercase (lowercase) signs in the last term of equation(4.21) correspond to a pulse along the $x(-x)$-axis ($\phi = 0°(180°)$). The compounding formula of equation(4.16), applied previously to phase-alternating -composite pulses [1, 10], is easily adapted to handle the analysis and design of SQUASH pulses.

For a phase-alternating composite quadrupole pulse sequence of the form $S_x(T) - t_1 - S_y(T) - t_2$, where S denotes $(\theta_1)_x - (\theta_2)_{\overline{x}} - (\theta_3)_x - \cdots$, [1] has derived a compact expression for the observable excitation during t_2 in terms of the resultant ER-parameters $\{\lambda, \Lambda\}$:

$$< I_y(t_1, t_2) > = E + F, \tag{4.22}$$

$$E = (\Lambda_2^2 + \Lambda_3^2)[\Lambda_2 \cos(\omega_Q \delta) - \Lambda_3 \sin(\omega_Q \delta)] \tag{4.23}$$

$$F = +(\lambda^2 - \Lambda_1^2)[\Lambda_2 \cos(\omega_Q \sigma) - \Lambda_3 \sin(\omega_Q \sigma)]$$
$$-2\lambda\Lambda_1[\Lambda_3 \cos(\omega_Q \sigma) - \Lambda_2 \sin(\omega_Q \sigma)] \tag{4.24}$$

The E term in these expressions is the quadrupole echo signal with $\delta = t_2 - t_1 - T/2$, while the F term is the feed-through signal with $\sigma = t_2 + t_1 + T/2$. Because the feed-through signal effectively originates with the first pulse, we can choose σ large enough to ensure that F is negligible compared to the noise. Under this assumption, we need only to consider the echo term (E), so that the observable excitation simplifies to

$$< I_y > = A \cos(\omega_Q \delta + \phi) \tag{4.25}$$

$$A = (\Lambda_2^2 + \Lambda_3^2)^{3/2}, \tag{4.26}$$

$$\phi = \tan^{-1} \Lambda_3/\Lambda_2. \tag{4.27}$$

In the case of time-symmetric pulse sequences [like $(\theta_1)_x - (\theta_2)_{\overline{x}} - (\theta_1)_x$], one of the overall ER-parameters, Λ_3, is zero [10], resulting in a very simple expression for the

amplitude of the echo signal at time $t_2 = t_1 + T/2$ (*i.e.* $\delta = 0$):

$$< I_y >_{echo} = \Lambda_2^3 \tag{4.28}$$

A pulse is time-symmetric if there exists a time $t = s$ such that

$$f(t + s) = f(t - s) \tag{4.29}$$

(see figure 4.5).

Provided only that we confine ourselves to time-symmetric phase alternating composite pulses, all quadrupole spin packets refocus simultaneously, and so there are no phase distortions in the quadrupole echo spectrum. Synchronous refocusing obviates the echo peak time calculation, a significant simplification for pulse design by computer search. Moreover, equation (4.28) shows that the entire performance of the sequence is characterized by a single ER-parameter Λ_2, reducing pulse design to maximizing the bandwidth of the in-phase excitation profile Λ_2^3. The fact that the in-phase magnetization profile for symmetric composite pulses depends exclusively on the ER-parameter Λ_2, which can easily be determined via the compounding formulae of equations(4.17) and (4.18), has previously made it possible to expedite a search for optimal composite pulse sequences [10]. We now consider how time-symmetry and an ER-parameterization can be used to some advantage in the design of SQUASH pulses.

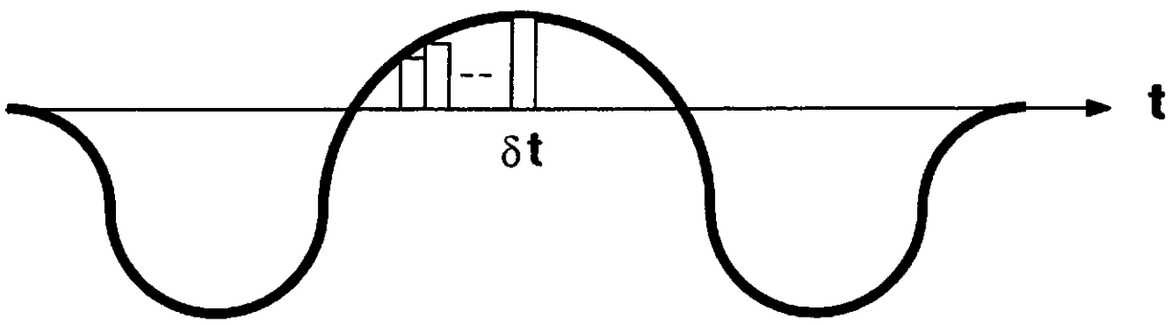


Figure 4.1: This figure shows how a shaped pulse, which is a continuous function of time, is divided into pulse-strips . Each such pulse-strip is treated as a pulse of fixed amplitude, fixed phase, and duration of δt, for the purpose of computation. As the number of the pulse-strips (N) increases, the duration (δt) of each pulse-strip decreases. Ideally, as $N \rightarrow \infty$ the duration $\delta t \rightarrow 0$. In that limit the shaped pulse is a continuous function of time. In reality N of several hundreds is a good approximation of a continuous shaped pulse.

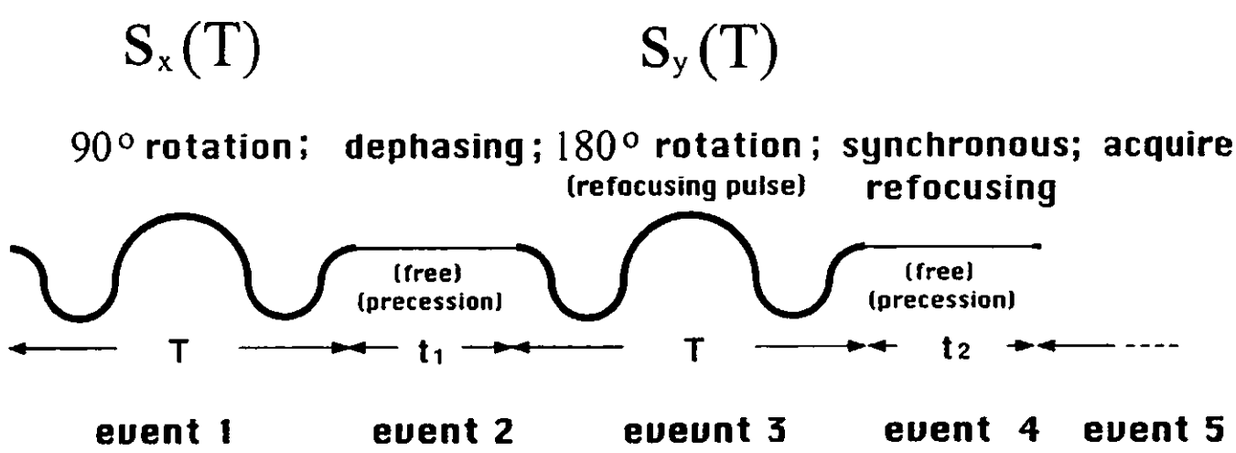


Figure 4.2: This is a quadrupole shaped pulse sequence. Its sequence of events is the same as that of the composite and the normal quadrupole pulse sequences. The negative areas of the shaped pulse sequence are regarded as pulses that are 180° phase shifted relative to the positive areas. $S_x(T)$ is a 90° pulse about the x-axis. $S_y(T)$ is a 90° pulse about the y-axis i.e. the refocusing pulse. In the case of time-symmetric shaped pulses with alternating phases, simple closed-form expressions for excitation profiles can be derived.

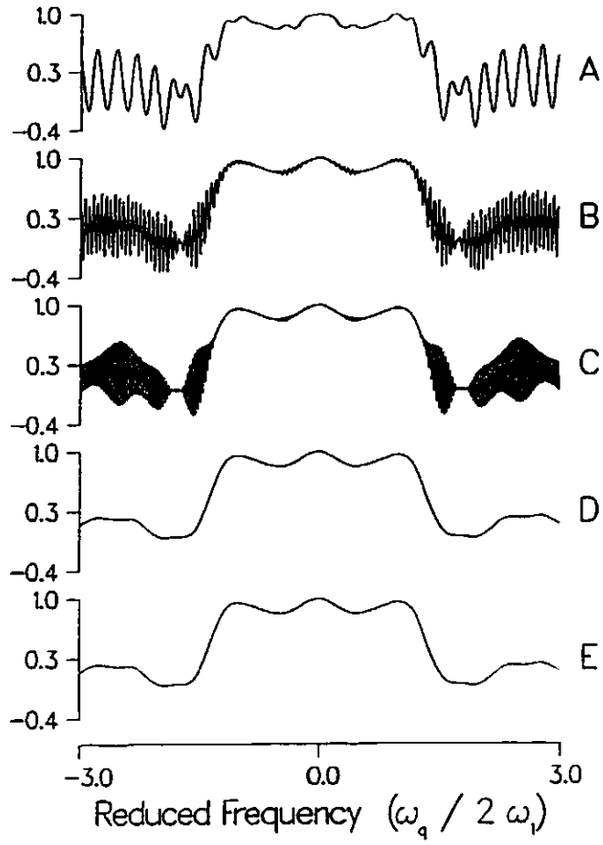


Figure 4.3: Top (A-C): representative quadrupole echo in-phase excitation profiles $< I_{y1}(t_1, \omega_Q) >$ evaluated at time of the echo for interpulse spacings t_1 of (A) 5 μs, (B) 50 μs, and (C) 95 μs. Echo sequences consisted of LSE [8] composite $135^\circ_x - 90^\circ_{-x} - 45^\circ_x$ pulses assuming an rf field strength of 80 kHz. Density matrix evolution techniques based on the fictitious spin-1/2 operators [16] were used to calculate each profile. Bottom (D,E): a comparison of (D), the averaged profile $\overline{< I_{y1}(\omega_Q) >} = \sum_{t_1} < I_{y1}(t_1, \omega_Q) >$ with (E), the profile evaluated using a closed-form expression [7] for the in-phase echo signal E. The efficacy of the averaging procedure is demonstrated by the exact agreement between profiles (D) and (E). Courtesy of N. J. Tagg .

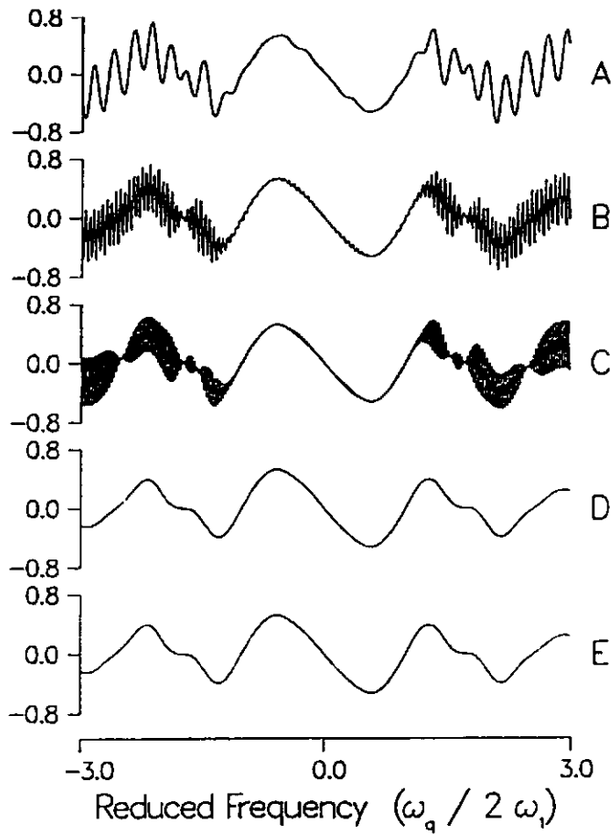


Figure 4.4: Top (A-C): representative quadrupole echo anti-phase excitation profiles $< I_{y2}(t_1, \omega_Q) >$ corresponding to the same interpulse spacings t_1 used for the in-phase profiles of figure 4.3, namely (A) 5 μs, (B) 50 μs, and (C) 95 μs. Bottom (D,E): a comparison of (D), the averaged profile $< I_{y2}(\omega_Q) > = \sum_{t_1} < I_{y2}(t_1, \omega_Q) >$ with (E), the profile evaluated using a closed-form expression for the anti-phase echo signal E'. As in figure 4.3, the efficacy of the averaging procedure is demonstrated by the exact agreement between profiles (D) and (E). Courtesy of N. J. Tagg.

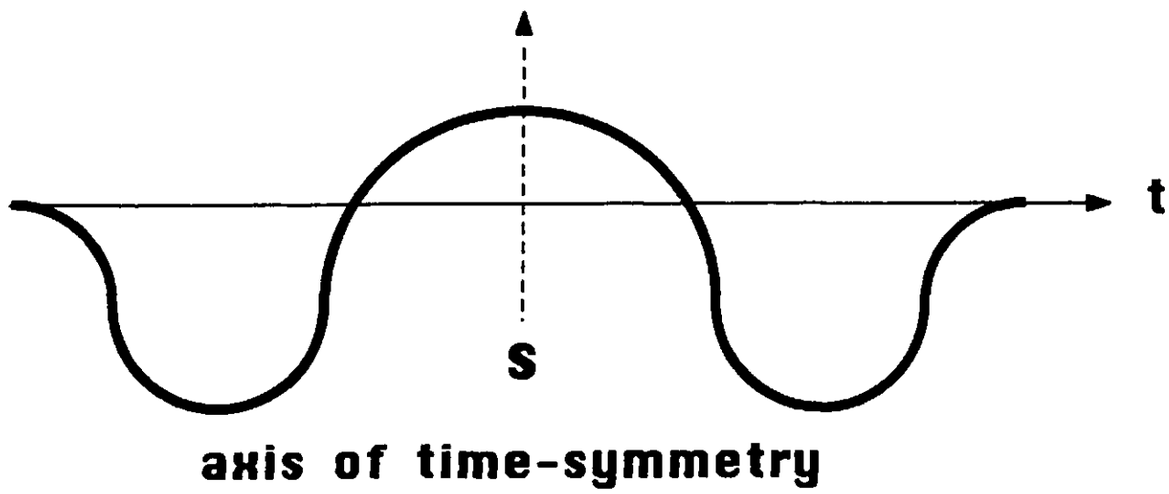


Figure 4.5: This is a time-symmetric shaped pulse since it has a time where a vertical axis of time-symmetry can be placed. If such a time is $t = s$ then $f(t + s) = f(t - s)$. This shape is chosen for the $S_x(T)$ and the $S_y(T)$ shaped pulses.

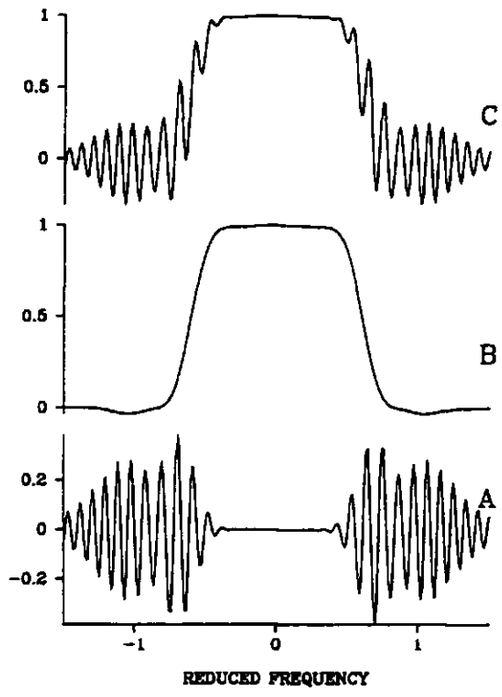


Figure 4.6: This is the analytical disentanglement of a QUASH sequence [11] in-phase shown in (C). The echo signal is shown in (B) while the feed-through signal is shown in (A). The reduced frequency is given by $\omega_Q/2\omega_1$

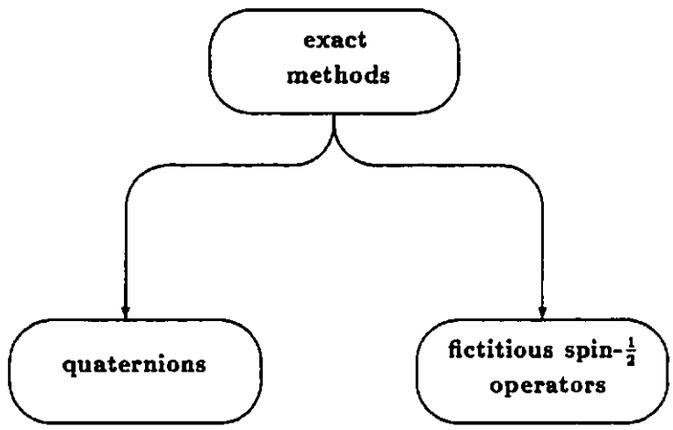


Figure 4.7: A flow chart showing the two exact methods for calculating spin-1 excitation profiles. The method of fictitious spin$\frac{1}{2}$ operators [16] yields closed-form expressions that are too complicated to disentangle analytically; disentanglement of echo and feed-through is done by computer aided averaging which is time consuming. The method of quaternions [1] yields simple closed-form expressions. In this case the echo and feed-through are separated analytically in a simple manner.

Chapter 5

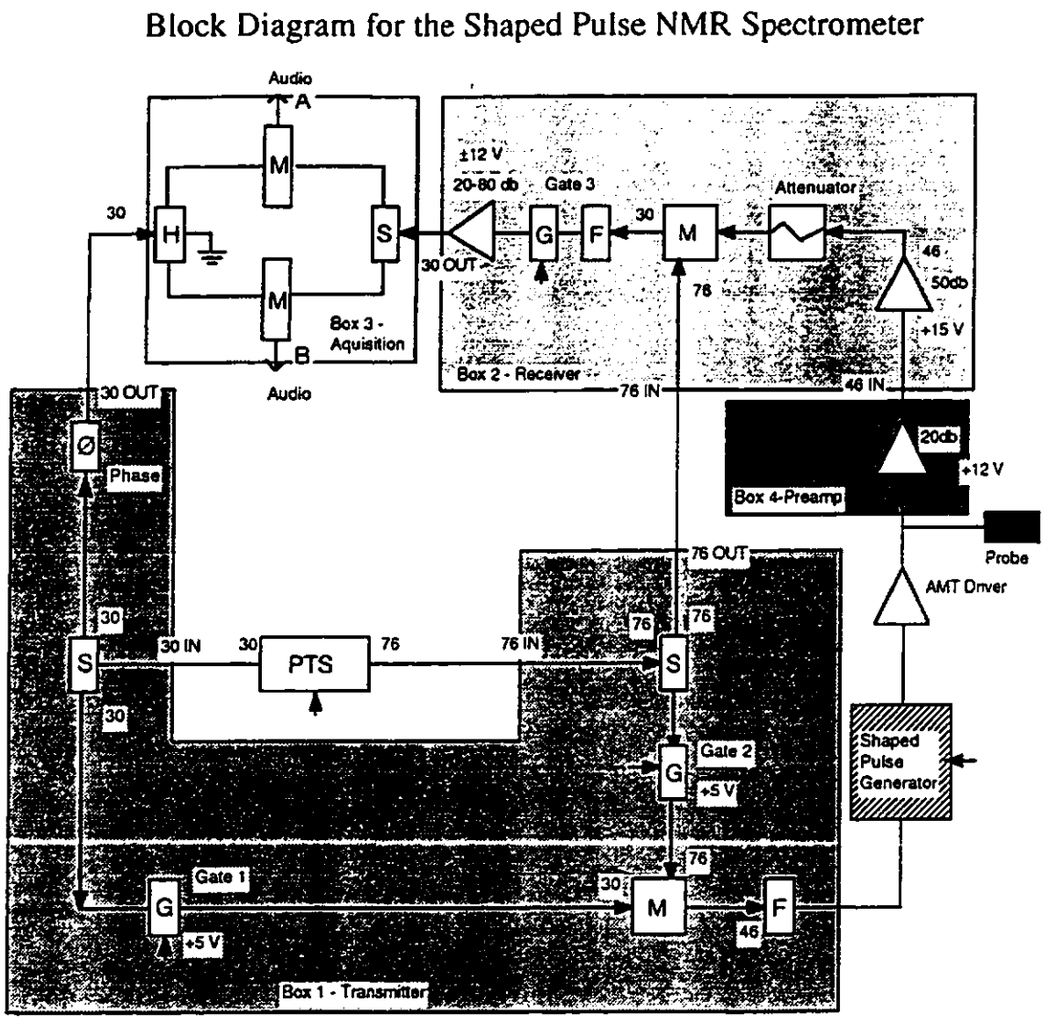
Experimental

5.1 Instrumentation

All ^{2}H NMR experiments were performed at a frequency of 46.06 MHz on a home-built spectrometer (see figure 5.1), using a Cryomagnet Systems (Indianapolis, IN) superconducting solenoid (7.0 T, 70 mm bore) and probehead, a Tecmag (Huston, TX) Libra pulse programmer and data aquisition system controlled by a Apple Quadra 700, and a AMT3206B (American Microwave Technology, Brea, CA) broadband FET power amplifier of nominal 60 db gain(AMT amplifier hereafter). Measurements of waveforms were performed with the aid of a 250 MHz dual-channel oscilloscope (Tektronix 475A, Beaverton, OR) in time domain and a spectrum analyzer (Hewlett-Packard 8590A) in frequency domain.

5.2 Choice of Sample

The samples used in these experiments were perdeuterated palmitic $-d_{31}$ acid $\mathbf{CD_3(CD_2)_{14}COOH}$ (Cambridge Isotopes Laboratories, Woburn, MA) and a perdeuterated plexiglass (poly methylmethacrylate-d_9). Although the spectra of these samples consist of a superposition of two powder patterns, one originating from the deuterated methyl groups, with a splitting of ca. 40 kHz, and the other from the more or less rigid methine (plexiglass) or chain (palmitic acid) deuterons, with a splitting of ca. 120 kHz,



M - Mixer
H - Quadrature Hybrid
S - Splitter
F - Filter
G - Gate
▲ - Under control of Pulse Programmer

Figure 5.1: The diagram of the home-built spectrometer

the relative intensities of these two powder patterns are very different in the two samples. On one hand, in the case of perdeuterated plexiglass, the superposition is dominated by the methyl group Pake doublet, while the rigid methine deuterons give rise to a much broader but very much weaker Pake doublet. On the other hand, the relative intensities in the palmitic-d_{31} acid superposition are reversed, so that the superposition is dominated by the broad Pake doublet originating from the rigid chain deuterons. Although methyl group powder patterns have often been used to experimentally verify the excitation performance of composite and shaped pulses, it is more useful to document the performance of shaped pulses using the more stringent test provided by the larger spectral width of the palmitic-d_{31} acid sample.

5.3 Pulse Shaper

To perform an experiment using a shaped pulse sequence, a shaped pulse generator (see figure 5.2) was interfaced to a spectrometer. The shaped pulse generator receives two inputs from the spectrometer: an unmodulated radio frequency of 46.06 MHz, and a timing signal. The shaped pulse from the output of the shaped pulse generator was amplified by the transmitter.

The shaped pulse generator was made up of a WSB-A12M Quatech waveform synthesizer, its customized 16-bit WSB-100 object file, a MiniCircuits ZAD-1W double balanced mixer as an amplitude modulator, and two Avantek GPD-404 cascadable amplifiers.

The waveform synthesizer produced a shape which described the envelope of the shaped pulse. The waveform synthesizer is a card installed in a 486 IBM PC chassis and operated by a FORTRAN program.

To generate a shape for an envelope of a pulse, a FORTRAN source program had to be

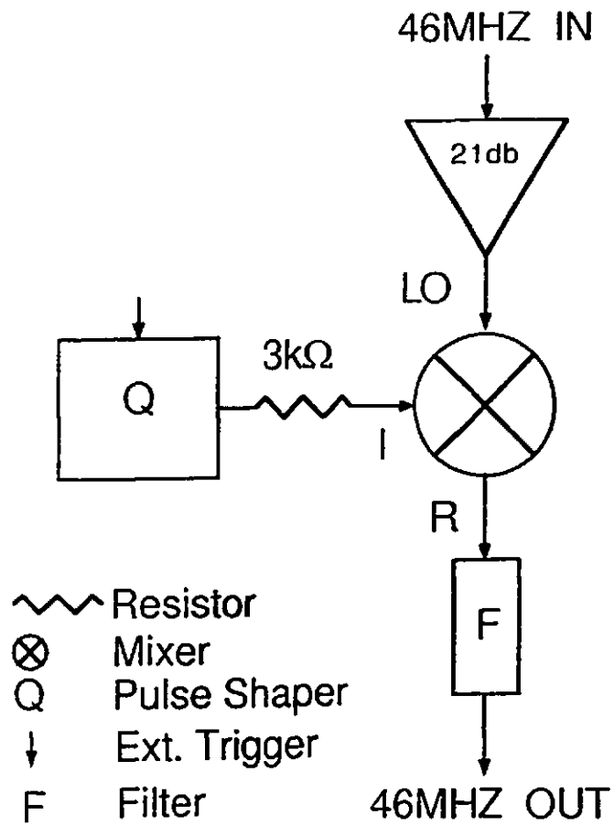


Figure 5.2: This is the diagram of the Shaped Pulse Generator that was constructed and interfaced with the laboratory's spectrometer

written, compiled, and then executed. A typical program was made up of the waveform synthesizer's instructions and parameters, in addition to other FORTRAN instructions.

The 46 MHz radio frequency originated from the laboratory's spectrometer. The power of that frequency was boosted from -12 dBm to 5 dBm for the mixer. For that purpose the two GPD404 Avantek amplifiers, in series, were used to boost the rf power by, 21db, from -12 dBm to 9 dBm. Each such amplifier had a tested 10.5db gain. A 4db attenuator, at the output of that amplifier, reduced the power down to 5 dBm to fulfil the Level-7 input power requirement of the mixer which came next.

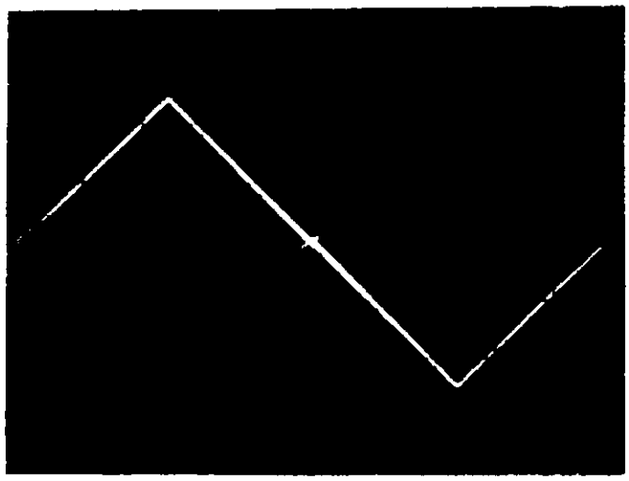


Figure 5.3: This is a snap shot of output R of the mixer used. The input LO received an input of +5.5 dBm and frequency of 46 MHz. The control input I received a current signal traced by the bright line. The faded waveform at output R looked like a dumbbell of two squares. The waveforms were overlapped successfully as a check for linearity of the mixer

The mixer modulated the amplitude of the 46.06 MHz rf in order to produce the actual shaped pulse. The Level-7 designation indicates that the maximum power allowed at the LO input could not exceed 7 dBm for proper operation; the minimum power was 4 dBm. For practical reasons the power of the 46.06 MHz rf at the input LO was set to 5 dBm. To modulate the amplitude of the LO input the mixer had to operate in a linear variable attenuator mode. The output of the waveform synthesizer was fed into the I input of the mixer. The level of the current through the mixer's I input determines much the signal of the LO input was attenuated. In the linear regime of the mixer, the

attenuation is directly proportional to the current through the I input. The linearity of the attenuation of the mixer was verified by the following procedure: the 46.06 MHz rf at input LO was kept at a power of 5 dBm, while the current at the I input was varied linearly by using the waveform synthesizer to generate a triangle waveform. The result at the R output was a 46.06 MHz rf with a diamond shaped envelop (see figure 5.3). A dual

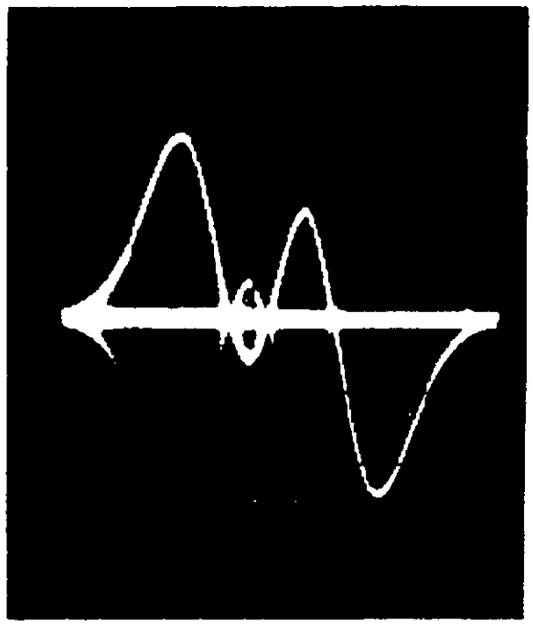


Figure 5.4: This is a snapshot of output R of the same mixer used in obtaining photograph 5.3 before. Input LO received a +5.5 dBm signal at a frequency of 46 MHz. This time, the control input I received a current of a different waveform. The brighter line traces that current's waveform. The faded waveform is the amplitude modulated waveform at output R. Note that the accurate overlap of the forms indicates the linearity of the mixer.

channel oscilloscope was used to superpose the modulating waveform of input I on the waveform of output R. The sought after linear attenuation mode was then verified when

the envelope of the triangle waveform at the I input and the envelope of the R output were to scale. The amplitude of the current was kept at 2.67m. The same procedure was repeated using different waveforms at the I input. The result was the same for each waveform (see example figure 5.4).

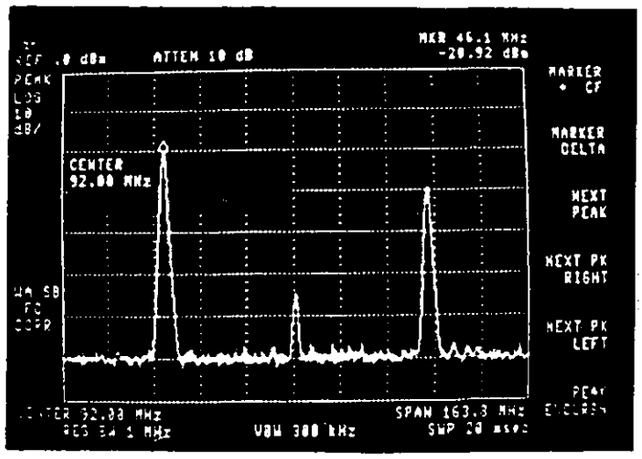


Figure 5.5: This is a snapshot of the R output of the mixer used. In that case, input LO received the same signal i.e. +5.5 dBm at a frequency of 46 MHz. The control input I received a direct current at about 2.6 ma. A spectrum analyzer showed peaks at three frequencies: 46 MHz, 92 MHz, and 138 MHz respectively. This is a typical feature of a mixer: producing multiples of sum and differences of two given frequencies.

A low pass filter was added to the output of the mixer to filter out multiples, 92 MHz and 138 MHz, of the 46.06 MHz frequency. This measure was taken to ensure that the transmitter's power would be devoted to the 46.06 MHz frequency only(see figure 5.5).

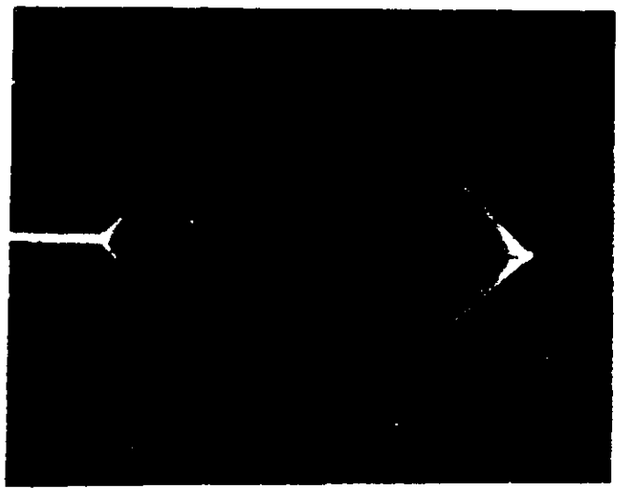


Figure 5.6: This is a snapshot of the output of the AMT amplifier, which is a linear amplifier, for a diamond shaped input. A bidirectional coupler was used to sample the output of that linear amplifier. The sides of this amplitude modulated waveform are slightly twisted. This nonlinear distortion was within the AMT's specifications.

frequencies;

The linearity of the transmitter was examined at the same time. A bidirectional coupler, (Narda 3020A, Hauppauge, NY)at the output of the transmitter, was used to sample the output signal. The observed deviation from linearity was within the transmitter's specifications. (see figure 5.6).

Finally, to construct a working shaped pulse sequence, the timing of the Quatech waveform synthesizer was synchronized with the spectrometer. This was done by sending an external trigger from the spectrometer to the Quatech waveform synthesizer's

external trigger input. To produce a Quadrupole Shaped Pulse (QUASH or SQUASH), a triggering signal had to be sent each time a 90° shaped pulse was required. Each shaped pulse was then transmitted to the sample.

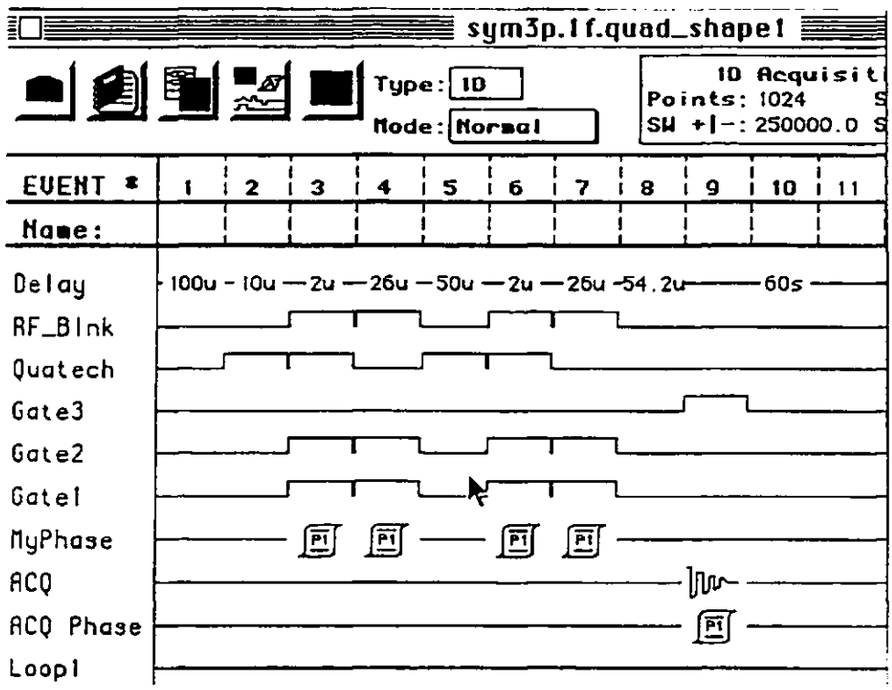


Figure 5.7: This is the pulse programmer Diagram. A pulse sequence is made up of a series of columns. A column is an EVENT

5.4 The Pulse Programmer

The Tecmag (Huston, TX) Libra pulse programmer (pulse programmer hereafter)is used in the composition and the execution of a pulse sequence. The Apple Quadra 700

(MACINTOSH hereafter) controls this pulse programmer through a customized MAC-NMR 5.3 resident software package. Each pulse sequence can be saved as a separate program for future use.

The pulse programmer is connected to those parts of the spectrometer which play an active role in an experiment. The software allows the experimenter to select the state of each of those parts whenever necessary. The control panel of the pulse programmer appears, on the screen of the MACINTOSH, in the form of an array of columns and rows. (see figure 5.7) . A column is a snap shot of the state of the spectrometer at a given time. It is referred to as EVENT. The top row of the array contains the serial number of each EVENT. The leftmost column lists the name of each part of the spectrometer which is under the control of the pulse programmer. A pulse sequence composed of a series of EVENTs.

List Of Pulse Programmer's Controls

CONTROL1) **Delay**: sets the time duration of each EVENT.

CONTROL2) **RF_Blnk**: controls the noise blanking of the AMT amplifier.

CONTROL3) **Quatech**: external trigger pulse to the Shaped Pulse Generator.

CONTROL4) **Gate 3**: TTL pulse receiver gate.

CONTROL5) **Gate2**: TTL pulse for the 76 MHz input to mixer *M*.

CONTROL6) **Gate 1**: TTL pulse for 30 MHz input to mixer *M*.

CONTROL7) **MyPhase**: controls the phase of each pulse and that of the acquisition. It can be set directly to any multiple of 90° by clicking on the icon. For

phase cycling mode the phases can be stored in a data file. The phase icons in figure 5.7 are of phase data files.

CONTROL8) **ACQ**: controls the acquisition unit which samples the fid.

CONTROL9) **ACQ Phase**: sets the phase of the acquisition unit.

The following is a description of a single run of a QUASH or a SQUASH pulse sequence. A run hereafter is referred to as *scan*. The number of scans required for phase cycling is determined by the number of lines in the phase data files. The phases for the first scan are in the first line of each phase data file and so forth.

The SQUASH pulse sequence used in the experiments is shown in figure 5.4. This sequence is composed of ten EVENTs.

To compose a SQUASH pulse sequence or any other shaped pulse sequence, the Quatech control of the pulse programmer must be used. EVENT 3 and EVENT 6 serve as preparation and transmission of the first pulse and the second pulse respectively.

To properly trigger the Quatech pulse shaper it was necessary to take into account that pulse programmer's controls operate in negative logic mode. When Quatech control is on as in EVENT 2 and EVENT 3 in figure 5.7, a 0v is output to the external trigger of the pulse shaper; when Quatech control is off, as in EVENT 4, the external trigger input of the pulse shaper receives 5v. When the external trigger input of Quatech pulse shaper receives a transition from 0v to 5v, a waveform is generated at the output of the Quatech pulse shaper. The first 90° shaped pulse is triggered as the pulse programmer moves from EVENT 3 to EVENT 4. The second 90° shaped pulse is triggered when the pulse programmer moves from EVENT 6 to EVENT 7. The total interpulse spacing is 53μs since the duration of each 90° shaped pulse is 24μs. A delay time of 2μs is added by EVENT 4 to the interpulse time spacing as adjustment. The time of the acquisition, EVENT 9, was adjusted by setting the duration of EVENT 8 to 54.2μs.

Steps of A Shaped Pulse Experiment

EVENT 1) All the controls of the pulse programmer are set to off. The corresponding parts of the spectrometer are off. The duration of this event is 100μs.

EVENT 2) Quatech control outputs a 0v to the external trigger of the Pulse Shaper. Its duration is 10μsec

EVENT 3) RF_Blnk switches off the blanking of the AMT amplifier. The spectrometer is in its transmission state. Nothing is transmitted yet as it receives no input. The output circuits of the AMT amplifier are given time to stabilize prior to transmission. A minimal duration of 2μs is recommended by the manufacturer. Gate 2 and Gate 3 are open. A phase data file is being put on standby.

EVENT 4) Quatech control switches from 0v to 5v thus producing an up transition which prompts the wave generator to send the first 90° shaped pulse . It is of 26μs duration and at a 46 MHz frequency. Gate2 and Gate1 switch on rf signals of 76 MHz and 30 MHz to mixer *M*. Mixer *M* then produced at its output a difference frequency of 46 MHz which is fed into the waveform generator. By varying the amplitude of the 46 MHz frequency the waveform generator creates the first pulse of the ***shaped pulse*** sequence. The phase of the shaped pulse is given by the phase data file whose icon appears on the MyPhase row. The AMT amplifier now has an input to amplify and transmit to the sample. The RF_Blnk control indicates that the state of the blanking is unchanged. The sample is now subject to the transmitted pulse.

EVENT 5) The 90° shaped pulse decayed to zero followed by the switching off of Gate2 and Gate1 RF_Blnk control switches on the blanking of the AMT

amplifier. Transmission to the sample is inhibited. This corresponds to the time of *free precession.* As Delay indicates the duration is 50μs.

EVENT 6) Preparation for transmission is taking place just as in EVENT 4.

EVENT 7) The second 90° shaped pulse is created just as in EVENT 4. Its phase data file contains phases which are 90° phase shifted relative to the pulse in EVENT 3. This is the refocusing pulse.

EVENT 8) All controls are in the same state as in EVENT 1. An echo is forming. . The duration, as indicated by Delay, is 41.2μs

EVENT 9) Gate3 is open. The fid is switched to the acquisition unit. The icon on the line of ACQ indicates that the fid is being sampled by the A to D converter and digitizer. The icon on the line of ACQ Phase is the phase data file of the acquisition. It contains the phases of the sampling.

EVENT 10) All controls are in the same state as in EVENT 1. Its duration, as indicated by Delay is 60μs. Its purpose is to allow complete relaxation of the spin packets of the sample.

The above list can be adapted very easily to the *normal quadrupole pulse sequence* by reconnecting the spectrometer's 46 MHz rf to the AMT amplifier instead of the output from the waveform generator.

Chapter 6

Results and Discussion

A computer search for optimal SQUASH pulses was conducted using the following three classes of shaped pulse parameterizations:

$$f = \cos\{a_1 \ln[a_2/(e^t + e^{-t})]\}/(e^t + e^{-t}) \tag{6.1}$$

$$f = \{b_0 + b_2 t^2 + b_4 t^4 + b_6 t^6 + b_8 t^8\} e^{-t^6} \tag{6.2}$$

$$f = \left\{ c_1 \cos(g_1 t) - \frac{c_2 \sin(g_2 t)}{t} e^{-t^2} \right\} e^{-t^6} \tag{6.3}$$

((6.1), (6.2), and (6.3) courtesy of D. Lu)

The following three classes of SQUASH pulses identified by this search are given in table 6.1, while figure 6.1 displays the optimal class III SQUASH pulses, along with the QUASH pulse[14, 11] for comparison.

figure 6.2 compares the in-phase excitation profiles for the QUASH pulse with those of several SQUASH pulses. The fact that simultaneous refocusing occurs in the case of SQUASH pulse sequences but not in the case of QUASH pulse sequences, means that only in the case of the former can these profiles be an accurate representation of the excitation bandwidth BW. These bandwidths, as measured from their respective in-phase excitation profiles, are tabulated in table 6.1. For each pulse of total duration T there is a measure of pulse efficiency, given by the ratio BW/T, and a parameter introduced by [14, 11], the measure of relative pulse energy RE , defined as the ratio of the shaped pulse energy E_{sh} to that of a square pulse E_{sq}:

$$RE = \frac{E_{sh}}{E_{sq}} \tag{6.4}$$

Table 6.1: Shaped Pulse Functions and Performance Parameters at $(\nu_1)_{max} = 50$ kHz

	Functions[a]	T[b] (μs)	BW[c⋆]	BW[†]	$\frac{BW^d}{T}$ ($\times 10^4$ s)	RE[e]
S1	$(1 - 7.05t - 0.225t^2 + 11.0t^3)e^{-t^2}$ $\lvert t\rvert \leq 1.5$	59.5	0.368	0.43‡	0.72	3.4
S2	$(0.5 - 0.6t^2 - 2.75t^4 + 0.06t^6 + 2.49t^8)e^{-t^8}$ $\lvert t\rvert \leq 1.5$	25.63	0.42	0.49	1.91	2.01
S3	$(0.5 - 0.59t^2 - 2.71t^4 + 0.12t^6 + 2.44t^8)e^{-t^8}$ $\lvert t\rvert \leq 1.2$	23.63	0.08	0.60	2.54	1.8
S4	$[1.2\cos(8.7t)\text{-}0.1\sin(11.7t)e^{-t^2}/t]e^{-t^8}$ $\lvert t\rvert \leq 1.3$	49.42	0.54	0.58	1.17	3.36
S5	$[1.2\cos(8.9t)\text{-}0.1\sin(11.7t)e^{-t^2}/t]e^{-t^8}$ $\lvert t\rvert \leq 1.3$	50.78	0.07	0.67	1.32	3.17
S6	$[2.7\cos(9.5t)\text{-}0.4\sin(12.1t)e^{-t^2}/t]e^{-t^8}$ $\lvert t\rvert \leq 1.3$	22.77	0.1	0.60	2.63	2.02
S7	$\cos\{5\ln[2/(e^t + e^{-t})]\}/(e^t + e^{-t})$ $\lvert t\rvert \leq 5$	59.90	0.23	0.34	0.57	1.56

[a] Function $S1$ is the QUASH pulse [14, 11] ; functions $S2 - S7$ are SQUASH pulses.
[b] 90° pulse duration.
[c] Half-bandwidth of in-phase profile in reduced frequency ($\omega_Q/2\omega_1$) units, to an accuracy of at least 99% ($\star$) or 95% (†).
[d] Excitation efficiency as quotient of half-bandwidth ($BW^\dagger$) to pulse duration T.
[e] Relative pulse energy as defined in Eq.(26).
‡ Only an approximation due to presence of anti-phase component. Courtesy of D. Lu

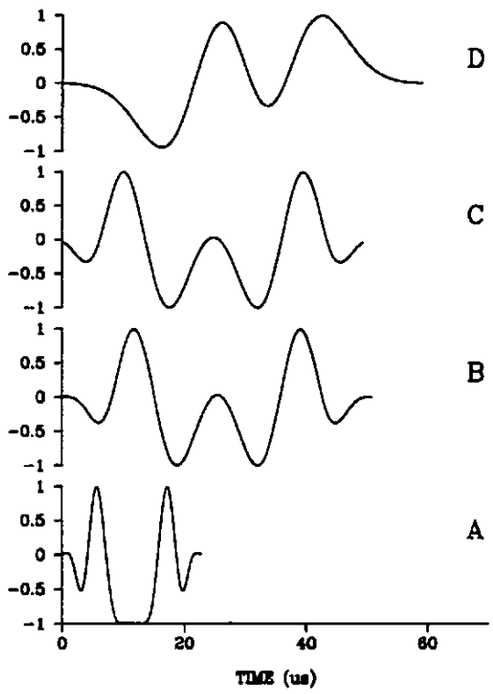


Figure 6.1: A comparison of several class *III* SQUASH pulse shapes (A-C) with (D), the QUASH pulse shape ($S1$ of table 6.1). The functional forms of the SQUASH pulse shapes, as given in table 6.1, are those of (A) $S6$, (B) $S5$, and (C) $S4$. A comparison of the SQUASH pulse shapes $S5$ and $S4$ shown in (B) and (C), respectively, illustrates the sensitivity of the shape to the defining coefficients $\{c_i\}$.
Courtesy of D. Lu

If $\nu_1(t)$ denoted the variable rf field strength of a shaped pulse of duration T, then

$$E_{sh} \propto \int_0^T \nu_1^2(t)dt \tag{6.5}$$

From the excitation profiles figure 6.2 and the performance parameters tabulated in table 6.1, it is evident that the SQUASH pulse out performs the QUASH pulse in terms of excitation bandwidth and efficiency.

Furthermore, keeping in mind that the SQUASH pulse performs so well while introducing no phase-distortions [15], as time-symmetry of that type of a pulse results in

simultaneous refocusing, emphasizes the superiority of this pulse sequence.

Table 6.2 compares the performance of single, composite, and shaped pulses. Note that only for the time-symmetric pulses of this table (i.e. single rectangular, time symmetric composite pulses, and SQUASH pulses), can the excitation bandwidth be well-defined since the anti-phase of each of this type pulses is identically zero [15]. Table 6.2 may be used to compare the performance of SQUASH pulse $S6$ with composite pulses of similar duration or bandwidth.

For example the shortest time-symmetric composite pulse, $C1$ of table 6.2 has roughly the same pulse efficiency and relative pulse energy as those of SQUASH pulse $S6$, but a much reduced excitation bandwidth. The time-symmetric composite pulse which has a reasonable bandwidth and a duration close to that of SQUASH pulse $S6$, suffers from a reduction of 25% in pulse efficiency and even more so in relative pulse energy which is more than twice the relative energy of SQUASH pulse $S6$. Table 6.2 indicates that the only phase-distortion free composite pulses which have comparable or greater excitation bandwidths, $C4$ and $C5$ have at least triple the relative pulse energy of the SQUASH pulse $S6$.

The plexiglass spectra of figure 6.3 compare the experimental performance of the QUASH pulse (B) with that of a single rectangular pulse (A). Unfortunately, the weakest spectral feature in these lineshapes, the broad methine deuteron Pake doublet, is also precisely the feature which could best reveal the breadth of QUASH pulse excitation. Although we may note that peaks of this doublet are very poorly defined in the QUASH spectrum (B) as compared to even the single pulse spectrum (A), it would be premature to conclude on the basis of this very weak spectral feature that the QUASH pulse echo has too narrow an excitation bandwidth. The *strongest* spectral feature is the narrower methyl group Pake doublet, which in each spectrum of figure 6.3 ($\nu_1 = 47$ kHz), is a facsimile of the corresponding methyl group Pake doublets in the L-alanine- d_3 spectra

Table 6.2: Performance Comparison Between Rectangular, Composite and Shaped Pulses at $(\nu_1)_{max}$ = 50 kHz.

	Sequence[a]	T[b] (μs)	BW[c]	RE[d]	$\frac{BW^e}{T}$ ($\times 10^{-1}$ s)
R1	Rectangular	5.0	0.18	1.0	3.60
C1	29° $\overline{148}$° 29°	11.44	0.33	2.29	2.88
C2	135° $\overline{90}$° 45°	15.0	†	3.0	
C3	32.9° $\overline{75.9}$° 176° $\overline{75.9}$° 32.9°	21.86	0.46	4.37	2.10
C4	145.6° $\overline{111.8}$° 22.4° $\overline{111.8}$° 145.6°	29.84	1.01	5.97	3.38
C5	134° $\overline{358}$° 134°	34.77	0.54	6.95	1.55
S1	QUASH	59.5	0.43‡	3.4	0.72
S6	SQUASH	22.78	0.6	2.02	2.63

[a] Overbar in composite pulses indicates 180° phase shift.
[b] 90° pulse duration.
[c] Half-bandwidth of in-phase profile in reduced frequency ($\omega_Q/2\omega_1$) units, to an accuracy of at least 95% (†).
[d] Relative pulse energy as defined in Eq.(26).
[e] Excitation efficiency as quotient of half-bandwidth (BW) to pulse duration T.
† Bandwidth could not be defined due to the large magnitude of the anti-phase component.
‡ Only an approximation due to presence of anti-phase component.
Courtesy of D. Lu

obtained by [14, 11](ν_1 =50 kHz) using the same pulses. At these rf field strengths, they have used a comparison of the QUASH spectrum of L-alanine-d_3, with those obtained using either single rectangular pulses or composite pulses, to conclude that the QUASH pulse is far superior. Certainly there is no doubt that in both the L-alanine-d_3 spectra obtained by [14, 11] and the plexiglass spectra in figure 6.3, the shoulders of the methyl group Pake doublet are better defined in the QUASH spectra, suggesting an improved excitation bandwidth of the QUASH pulse. However, the improved performance of the QUASH pulse is somewhat misleading, since the enhancement of the shoulder intensity in

these experimental spectra is partially the result of phase distortions caused by asynchronous refocusing. The existence of such asynchronous refocusing is indicated by simulated plots of the anti-phase excitation profiles at the time of the QUASH echo,[11] or figure 6.5 C regardless of superimposed feed-through oscillations. Effects of asynchronous refocusing on the NMR lineshape is most easily visualized in the simulated transfer functions [7, 13]. Such functions contain a central region for pulses which give rise to asynchronous refocusing. This dip is particularly pronounced in the LSE [8] 135°_{x}-90°_{-x}-45°_{x} composite pulse [7, 13], and to a lesser extent in that for the QUASH [13]. Experimental tests which exploit relatively narrow methyl group lineshapes as in the case of plexiglass or L-alanine-d_3, these dips are held accountable for enhancing shoulder intensity.

To avoid the pitfall in relying only a narrow spectrum for testing excitation performance, a great deal of care should be taken in assessing the performance of the QUASH pulse, as revealed in the plexiglass spectra of figure 6.3. A more stringent performance test of a pulse sequence would be required. Since insufficient spectral breadth is the reason why pulse sequence performance information was missed, it is obvious that a sample with spectral breadth comparable to that of the pulse's excitation profile, may offer a solution. For that purpose, the much broader spectrum of palmitic-d_{31} acid figure 6.6, is, used to compare experimental performance of a single rectangular pulse, with those of QUASH and SQUASH pulses. The palmitic-d_{31} is the better choice since unlike the spectrum of plexiglass, the spectrum of palmitic-d_{31} has its stronger spectral feature in the broad Pake doublet originating from the rigid chain deuterons i.e. the methine.

A case in point as to how such enhancement can misrepresent a true excitation performance is provided by, again, the tests of the LSE [8] composite pulse 135°_{x}-90°_{-x}-45°_{x}. The relatively narrow spectra of plexiglass acquired with a LSE [8]sequence are classic Pake doublets with particularly well defined shoulders. However, as impressive looking such spectra may be, they obscure the true excitation performance which is revealed in

the severe distortions of the much broader spectra of palmitic-d_{31} acid acquired in this case with this pulse [7].

By comparing the in-phase/anti-phase profile [11, 13] figures 6.4 and 6.5 or transfer functions [13], effects of phase distortion in QUASH pulse spectra are expected to be much less severe and subtle in the narrow spectrum of plexiglass or L-alanine-d_3. In that case, regarding (at the expense of neglecting the anti-phase) the in-phase component only, the conclusion would seem to be that a spectrum is less distorted i.e. more uniformly excited by the QUASH pulse.

However, applying the same QUASH pulse to the broad spectrum of palmitic-d_{31} acid will result in more distinct distortions. Thus it would be more convenient to draw conclusions from the results of the palmitic-d_{31} acid case than from the results of the plexiglass and L-alanine case.

Figure 6.6 shows the results of a more stringent test where the much broader spectra of palmitic-d_{31} acid was used to compare the experimental performance of a single rectangular pulse, with those of the QUASH and SQUASH pulses. It is evident (figure6.6B) from these results that the QUASH pulse sequence is not able to excite as broad a spectrum as that of the palmitic-d_{31} acid. The absence of the shoulders only emphasize this lack of ability. That the breadth of excitation is the narrowest of all the three spectra can be verified by the abrupt cutoff in the excitation profile of the QUASH pulse sequence in figure 6.2D. In addition to amplitude distortions, phase distortions due to asynchronous refocusing also take their toll on the palmitic-d_{31} spectrum rendering the QUASH pulse sequence ineffective in exciting a broadband. It should also be noted that although the square pulse is able to excite a broad a band as the SQUASH pulse, that performance was not nearly as uniform as that of the SQUASH pulse. The SQUASH pulse uniformly excited palmitic-d_{31} spectrum over the broadest bandwidth. These improvements were achieved while attaining a reduction of 41% reduction in relative pulse

energy ,and without *any* phase distortion as the SQUASH pulse is time-symmetric: a three fold achievement.

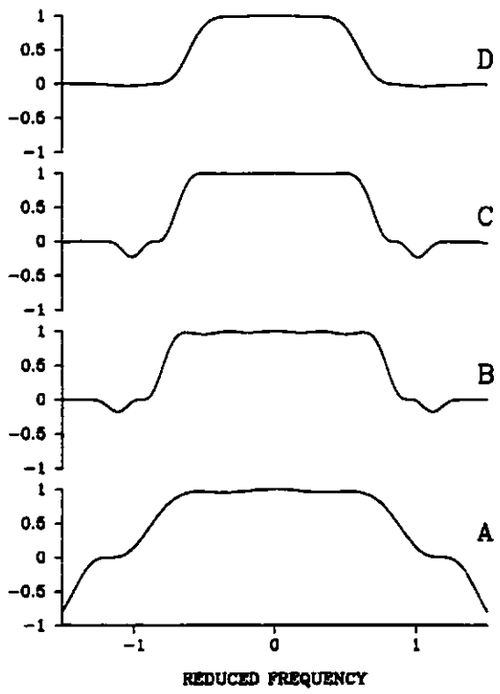


Figure 6.2: A comparison of in-phase excitation profiles $< I_y(\omega_Q) >$ for echo sequences consisting of the class *III* SQUASH pulse shapes (A-C) of figure 6.1, and the QUASH pulse shape (D). A comparison of the excitation profiles shown in (B) and (C), respectively, illustrates the sensitivity of these profiles to the coefficients $\{c_i\}$ defining these class *III* SQUASH pulses. This comparison shows that as much as a 10% variation in coefficients may be tolerated without significantly affecting the excitation profile. Courtesy of D. Lu

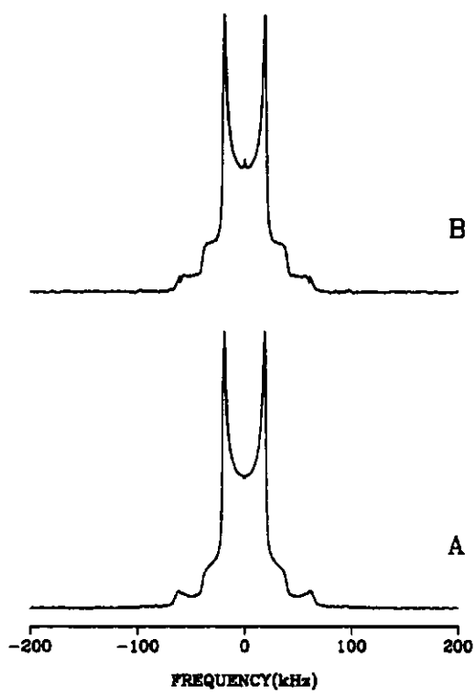


Figure 6.3: A comparison of ^{2}H NMR quadrupole echo spectra of perdeuterated plexiglass (polymethylmethacrylate-d_9) acquired with (A) single, rectangular pulses and (B) QUASH pulses. For the single pulse quadrupole echo experiments, the 90° pulses were 5.3 μs in duration, and separated by an interpulse delay t_1 of 60 μs. For the QUASH pulse echo experiments, the 90° QUASH pulses were 62 μs in duration, and separated by an interpulse delay t_1 of 23 μs. A total of 8 time-domain signals were averaged, each acquired with 1 K data points, using a digitization rate of 2 μs, corresponding to a spectral width of 500 kHz. The recycle delay was 60 s, in order to allow sufficient time for the plexiglass methine deuterons to recover. Both spectra have 500 Hz of line-broadening introduced by exponential multiplication of the echo signal prior to Fourier transformation.

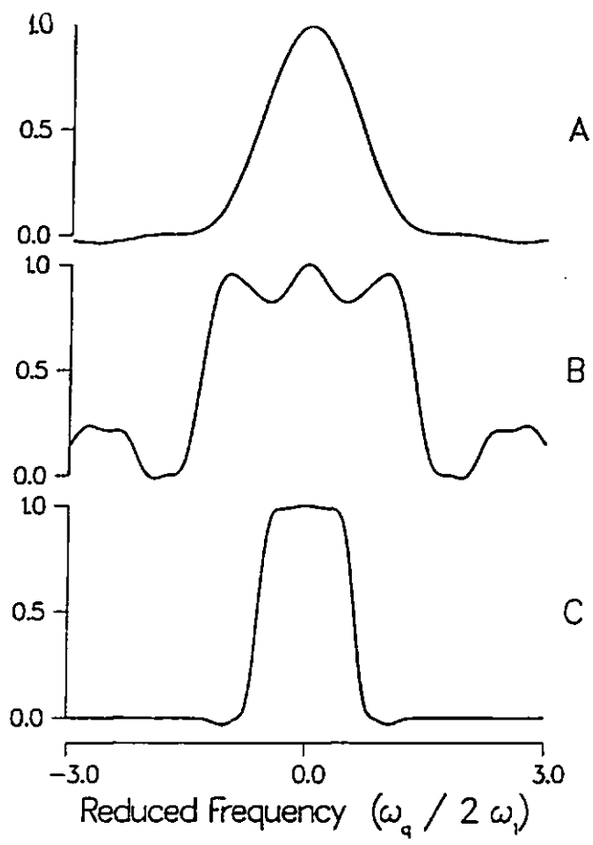


Figure 6.4: A comparison of the *averaged* in-phase excitation profiles $\overline{< I_{y1}(\omega_Q) >}$ for echo sequences consisting of (A) single, rectangular 90° pulses, (B) LSE[8] composite $135^\circ_x - 90^\circ_{-x} - 45^\circ_x$ and (C) QUASH pulses[14, 11]. Several hours of CPU time on a DEC VAX 6300 were required to calculate the averaged QUASH profile; three orders of magnitude less time were required to calculate the corresponding echo profile by the quaternion techniques discussed in the text. Courtesy of N. J. Tagg

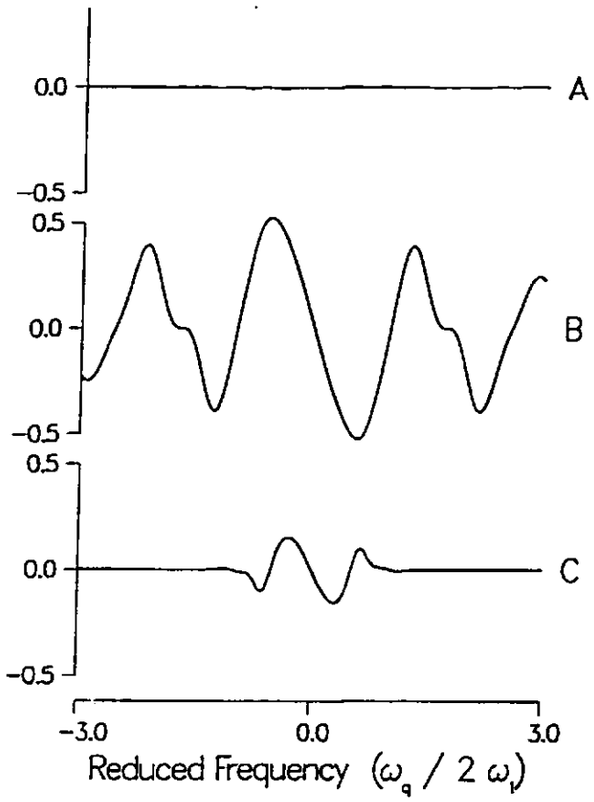


Figure 6.5: A comparison of the *averaged* anti-phase excitation profiles $\overline{< I_{y2}(\omega_Q) >}$ corresponding to the in-phase profiles shown in figure 6.4. The single pulse profile (A) should vanish identically as a consequence of simultaneous refocusing; the small amplitude oscillations in this profile are therefore a remnant of the averaging procedure. In this case, a less than optimum number of averaged profiles was deliberately chosen so as to emphasize the averaged nature of these profiles. Note that even though the magnitude of the QUASH[11] anti-phase excitation is significantly smaller than that of the LSE [8] composite pulse,it is still large enough to contribute to phase distortions. Courtesy of N. J. Tagg

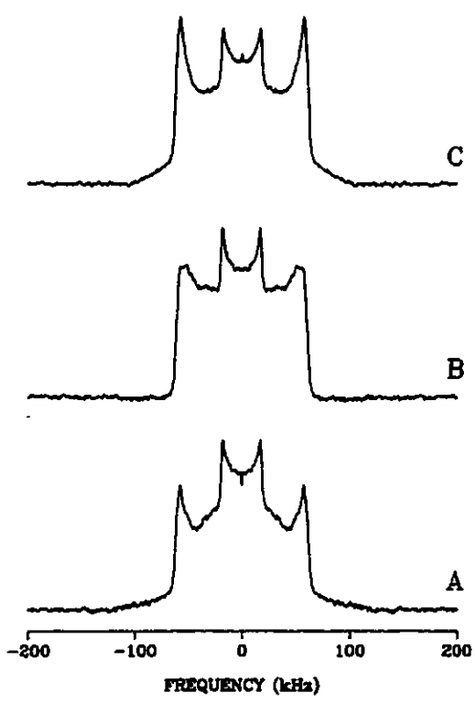


Figure 6.6: A comparison of ^{2}H NMR quadrupole echo spectra of perdeuterated palmitic-d_{31} acid acquired with (A) single, rectangular pulses, (B) QUASH pulses and (C) SQUASH pulses (*S*6 of table 6.1). For the SQUASH pulse echo experiments, the 90° SQUASH pulses were 24 μs in duration, and separated by an interpulse delay t_1 of 55 μs. For the single pulse and QUASH echo experiments, the 90° pulse durations and interpulse delays t_1 were the same as those used to acquire the plexiglass spectra of figure 6.3. A total of 64 time-domain signals were averaged, each acquired with 1 K data points, using a digitization rate of 2 μs, corresponding to a spectral width of 500 kHz. The recycle delay was 60 s, in order to allow sufficient time for the palmitic-d_{31} acid chain deuterons to recover. Both spectra have 500 Hz of line-broadening introduced by exponential multiplication of the echo signal prior to Fourier transformation.

Chapter 7

Conclusion

This work has shown how quaternion formalism can be used to derive a *very simple* relationship between the structure of a pulse sequence and an excitation it produces in a spin-1 ensemble. This relationship was then used to design a type of pulse sequence that would produce a desired excitation.

By rearranging the basis operators of the LVN equation(2.16) it was possible to express the solution, time evolution, in quaternion formalism. Time evolution in that formalism was expressed as four dimensional rotations which were parameterized by the ER-set $\{\lambda, \Lambda\}$. Parameterized this way, a product of any number of quaternions was reduced to a single resultant which was also a quaternion. This reduction was made so simple since a product of such parameterized quaternions obeys two simple composition rules in ER-parameter set. This approach is the simplest and the most direct method of calculating the response to a pulse sequence.

By using ER-parameters, the observable magnetization was disentangled *directly* into two simple closed-form expressions. The functional form of these expressions is a product of an amplitude factor which describes distortions in amplitude of the observable magnetization, and another factor which describes distortions in phase of quadrupole spin packets of the observable magnetization. One expression, echo, is a result of spin packets that refocus by the pulse; the other expression, feed-through, is due to the fact that off-resonance spin packets do not see full flip angles for the refocusing pulse. Pulse optimization was straight forward since the observable echo, unobscured by feed-through

oscillations, permitted direct access to information on pulse performance. Density matrix evolution, by contrast, required a repetitive and elaborate averaging procedure only to obtain the observable echo numerically.

The class of time symmetric pulses, SQUASH pulses, is even more advantageous, analytically as well as numerically. Under the action of a SQUASH pulse, all quadrupole spin packets refocus at the same time i.e. synchronous refocusing. The vanishing of one of the ER-parameters, Λ_3, permits further simplification of the closed-form expression for the observable echo. The phase factor of this echo then becomes identically 0 while the amplitude factor reduces to a function of one ER-parameter, Λ_2^3. The *entire* performance of the SQUASH pulse, as given by the observable echo, can then be expressed by a single ER-parameter, the Λ_2. Since the anti-phase in the case of time-symmetry is identically zero, the task of numerical optimization reduces to maximizing the excitation bandwidth of the in-phase signal, the observable echo.

The existence of exact echo-peak time due to synchronous refocusing means that there is no need for approximate techniques previously used to determine a refocusing time for the peak of the echo [11]. A Fourier transform performed on the time domain signal, *fid*, at such a time, would produce a true picture of time evolution in frequency domain i.e. a correct spectrum. Thus misinterpretations of pulse performance may be avoided.

The pulse performance of the SQUASH pulse is a function of a single resultant ER-parameter, Λ_2, which is calculated by two simple composition rules [1, 8], the typical time required for calculating the echo response via density matrix evolution may be reduced by at least two orders of magnitude. This saving in time considerably expedites the search procedure, and improves pulse design by allowing a search to be conducted over a larger class of shaped pulse parameterization.

On the basis of simulated and experimental performance of SQUASH pulses identified in this study, it is possible to list the following advantages these pulses over the QUASH

pulse found by [11, 14]:

SQUASH pulses, by construction, produce an observable in-phase echo that is completely free of phase-distortion effects as the refocusing of quadrupole spin packets is synchronous. In this case, the observable in-phase echo gives a complete description of the pulse's performance. Also experimental spectra excited by such pulses may be simpler to interpret. Pulses that are not symmetric in time, however, produce a nonvanishing anti-phase echo in addition to an in-phase echo. In this case, the observable in-phase does not give a complete description of the pulse's performance since part of that description is contained in the unobservable anti-phase echo. Anti-phases have been documented for composite pulses [7, 1], and for the QUASH pulse [13], and should be minimized, or even avoided whenever possible. Otherwise, pulse performance might be interpreted incorrectly if only the in-phase is considered[11] . By restricting our choice to a class of time-symmetric pulses that are amplitude modulated and are fixed in phase, SQUASH pulses, it is possible to make big improvements in uniform excitation bandwidth and pulse efficiency, while *avoiding* phase-distortions completely. Thus by some sacrifice in generality pulse performance may be improved significantly.

As a case in point, a comparison between the QUASH pulse [11, 14, 15], and several this class of SQUASH pulses shows dramatic improvement in both excitation bandwidth, and efficiency through pulse energy relative to square pulse energy. In particular, as shown in table 6.1, the most promising member of this class, $S6$, provides a 50% improvement in excitation bandwidth, *while* reducing the relative pulse energy by 41%.

Thus based upon performance, the class of time-symmetric quadrupole shaped pulses (SQUASH) has helped improve the study of the nature of spin-1 in theory and in experiment.

Appendix A

Disentangling Rotation Operators By Quaternions

A disentanglement of the rotation operator introduced by Santiago and Vaidya is used to prove the composition rule for the Euler-Rodrigues parameters in a representation-independent manner[4]. Expressed in terms of the rotation axis $\hat{\mathbf{n}}$ and the rotation angle Φ, the ER parameters $\{\lambda, \mathbf{\Lambda}\} \equiv \left\{\cos\frac{\Phi}{2}, \hat{\mathbf{n}}\sin\frac{\Phi}{2}\right\}$ are the *only* parameters for which the group multiplication rule can be given in closed form [3] as a simple bilinear composition (*vide infra*). They are also the representation of smallest dimension which has such a bilinear composition rule, and consequently a *linear* kinematic equation [18, 19].

Some years ago, [20] showed how the matrix elements of the rotation operator could be evaluated in an extremely simple manner by using Baker-Campbell-Hausdorff (BCH) formulae for the exponentials of the generators of the $SU(2)$ group. For the case of a rotation axis/rotation angle $\{\Phi, \hat{\mathbf{n}}\}$ parameterization of the rotation operator $\mathcal{D} = \exp[-i\Phi\hat{\mathbf{n}} \bullet \mathbf{I}]$, their disentanglement of $\mathcal{D}$ as

$$\mathcal{D} = \exp[A_+ I_+]\exp[\ln(A_0) I_0]\exp[A_- I_-] \tag{A.1}$$

introduced c-number disentangling coefficients $\{A_+(\Phi, \hat{\mathbf{n}}), A_0(\Phi, \hat{\mathbf{n}}), A_-(\Phi, \hat{\mathbf{n}})\}$ which considerably simplified the evaluation of the required matrix elements $\mathcal{D}^{(I)}_{mn}(\Phi, \hat{\mathbf{n}})$. However, it appears to have been overlooked that if these c-number disentangling coefficients are expressed in terms of the ER parameters $\{\lambda, \mathbf{\Lambda}\}$ as $\{A_+(\lambda, \mathbf{\Lambda}), A_0(\lambda, \mathbf{\Lambda}), A_-(\lambda, \mathbf{\Lambda})\}$, then the particular disentanglement technique employed by [20] also provides a direct and simple route for establishing several important properties of the ER parameters. These

properties include their composition rule, as we show in this comment, and their kinematic relations, as shown previously [21]. In the context of NMR, the present comment has two purposes. First, we take advantage of the Santiago and [20] disentanglement procedure to establish the ER parameter composition rule in a representation-independent manner. We believe this provides the first quantum mechanical argument to establish the validity of the ER parameter composition rule for spins of arbitrary magnitude I without choosing a representation. Second, we note that the expression for the c-number coefficient $A_+ \equiv A_+(\lambda, \mathbf{\Lambda})$ in the Santiago and [20] disentanglement of the quantum mechanical rotation operator embodies transformations of two sets of coupled equations of motion. Both sets, one the Euler kinematical relations [22, 23], and the other the Bloch equations, provide a *classical* description of spin kinematics. Using the ER parameters, a composition rule for the $SO(3)$ rotation matrices $\mathbf{R}(\lambda, \mathbf{\Lambda})$ can be defined. The compounding formula of [2]for the product of two such consecutive three-dimensional rotations applied to a spin system takes the following standard form (rotation through an angle Φ_1 about $\hat{\mathbf{n}}_1$ followed by rotation through an angle Φ_2 about $\hat{\mathbf{n}}_2$) :

$$\mathbf{R}_2(\lambda_2, \mathbf{\Lambda}_2)\mathbf{R}_1(\lambda_1, \mathbf{\Lambda}_1) = \mathbf{R}(\lambda, \mathbf{\Lambda}) \tag{A.2}$$

$$\text{where } \lambda = \lambda_1\lambda_2 - \mathbf{\Lambda}_1 \bullet \mathbf{\Lambda}_2 \tag{A.3}$$

$$\lambda_i = \cos\left(\frac{\Phi_i}{2}\right) \quad (i = 1,2) \tag{A.4}$$

$$\mathbf{\Lambda}_i = \hat{\mathbf{n}}_i \sin\left(\frac{\Phi_i}{2}\right) \quad (i = 1,2) \tag{A.5}$$

$$\mathbf{\Lambda} = \lambda_2\mathbf{\Lambda}_1 + \lambda_1\mathbf{\Lambda}_2 + \mathbf{\Lambda}_2 \times \mathbf{\Lambda}_1 \tag{A.6}$$

This formula, originating in rigid body kinematics [24, 25, 3], can be used for calculating the response to spin-1/2 composite pulses [2, 26] as well as phase-alternating spin-1 composite pulses (Barbara 1986). From a classical point of view, the first purely *geometrical* approach to obtaining the resultant ER parameters $\{\lambda, \Lambda\}$ for two consecutive rotations was taken by [24], using the Euler-Rodrigues construction, and spherical trigonometry. An alternative geometrical derivation has recently been given by [27], based on the decomposition of a rotation into two reflections [28, 29, 3]. Such a decomposition can be used to account for the presence of *half*-angles in the composition rule. An entirely different classical approach relies on the relationship between rotations and homographies [30, 31]. Under stereographic projection of the Bloch sphere onto the complex plane, rotations of points on the sphere correspond to homographic transformations $z \rightarrow z'$ of the complex plane. By parameterizing each homographic transformation via the ER parameters $\{\lambda, \Lambda\}$ as

$$z' = \frac{(\lambda + i\Lambda_z)z - (\Lambda_y - i\Lambda_x)}{(\Lambda_y + i\Lambda_x)z + (\lambda - i\Lambda_z)} \tag{A.7}$$

a composition of two such homographies may also be used to obtain the ER parameter composition rule of equations (3) and (6) [30, 32].

Quantum mechanical proofs of this composition rule using the rotation axis/rotation angle $\{\Phi, \hat{\mathbf{n}}\}$ parameterization of the rotation operator have always relied on the spin-1/2 representation of the rotation group [33], [2]. Since the composition rule relates only the angles and axes of the rotations, both of which are independent of the representation, a choice of the spin-1/2 representation is clearly sufficient [2]. By the same token, it should not be necessary to choose *any* representation, which we now demonstrate by establishing the ER parameter composition rule of equations (3) and (6) in a representation-independent manner.

For this purpose, consider a set of operators $K_\pm \equiv K_x \pm iK_y$ and $K_0 \equiv K_z$ which

satisfy the usual commutation relations for the generators of the $SU(2)$ Lie algebra:

$$[K_-, K_+] = -2K_0 \tag{A.8}$$

$$[K_0, K_\pm] = \pm K_\pm \tag{A.9}$$

Then the following normal-order decomposition formula for an exponential function of these generators can be derived [20, 34, 35, 36]:

$$\exp[a_+K_+ + a_0K_0 + a_-K_-] = \exp[A_+K_+]\exp[\ln(A_0)K_0]\exp[A_-K_-] \tag{A.10}$$

$$\text{where } A_\pm = \frac{(a_\pm/f)\sinh f}{\cosh f - (a_0/2f)\sinh f} \tag{A.11}$$

$$A_0 = [\cosh f - (a_0/2f)\sinh f]^{-2} \tag{A.12}$$

$$f = [(a_0/2)^2 + a_-a_+]^{1/2} \tag{A.13}$$

In order to generalize the decomposition formula of equation (9), the ordered product $G(n)$ of the exponential functions of the generators of the Lie algebra can be defined as follows [36]:

$$G(n) = \prod_{k=1}^{n} \exp[a_+(k)K_+ + a_0(k)K_0 + a_-(k)K_-] \tag{A.14}$$

$$\text{where } \prod_{k=1}^{n} g(k) = g(n)g(n-1)\ldots g(2)g(1) \tag{A.15}$$

Using this notation, for arbitrary c-number functions $\{a_\pm\}$ and $\{a_0\}$, [36] has derived the following normal-order BCH formula for the ordered product $G(n)$:

$$G(n) = \prod_{k=1}^{n} \exp[a_+(k)K_+ + a_0(k)K_0 + a_-(k)K_-] = \exp[A_+(n)K_+]\exp[\ln(A_0(n))K_0]\exp[A_-(n)K_-] \tag{A.16}$$

The c-number functions $\{A_\pm\}$ and $\{A_0\}$ are defined by [36] in terms of recursion formulae, of which the following formula for $A_+(n)$ will suffice for the purpose of illustration:

$$A_+(n) = \frac{(a_+/f(n))\sinh f(n) + \{\cosh f(n) + (a_0/2f(n))\sinh f(n)\}A_+(n-1)}{\cosh f(n) - (a_0/2f(n))\sinh f(n) + (a_-/f(n))\sinh f(n)A_+(n-1)} \tag{A.17}$$

$$\text{where } A_+(0) = 0 \tag{A.18}$$

$$f(n) = [(a_0(n)/2)^2 + a_-(n)a_+(n)]^{1/2} \tag{A.19}$$

The above formulae (equations (15-18)) for the ordered product $G(n)$ are not based on geometrical considerations, but rather were obtained by [36] using induction and parameter differentiation [37], and the commutation relations of equations (7) and (8). For a rotation first performed about an axis $\hat{\mathbf{n}}_1$ through an angle Φ_1, followed by a second rotation about an axis $\hat{\mathbf{n}}_2$ through an angle Φ_2, the operator $\mathcal{D}$ of the resultant rotation can be expressed in terms of an ordered product $\mathcal{P}$ of rotation operators as

$$\mathcal{D} \equiv \exp[-i\Phi\hat{\mathbf{n}} \bullet \mathbf{I}] = \exp[-i\Phi_2\hat{\mathbf{n}}_2 \bullet \mathbf{I}] \exp[-i\Phi_1\hat{\mathbf{n}}_1 \bullet \mathbf{I}] \equiv \mathcal{P} \tag{A.20}$$

We now use the generalized BCH formula of equation (15) to show that the ER parameters $\{\lambda, \Lambda\}$ of the resultant rotation can be expressed in terms of the ER parameters $\{\lambda_i, \Lambda_i\}$ of the two consecutive rotations by the composition rule of equations (2-5). For this purpose, using the fact that the axis of rotation $\hat{\mathbf{n}}$ is

$$\hat{\mathbf{n}} = (\sin\ \theta \cos\ \phi, \sin\ \theta \sin\ \phi, \cos\ \theta) \tag{A.21}$$

the operator argument $-i\Phi\hat{\mathbf{n}} \bullet \mathbf{I}$ of the rotation operator can be expressed in the notation of [36, 20] as

$$-i\Phi\hat{\mathbf{n}} \bullet \mathbf{I} \equiv -i\Phi(n_x I_x + n_y I_y + n_z I_z) = a_+ I_+ + a_0 I_0 + a_- I_- \tag{A.22}$$

$$\text{where } a_\pm = -i\Phi \sin\theta \frac{e^{\mp i\phi}}{2} \tag{A.23}$$

$$a_0 = -i\Phi \cos\theta \tag{A.24}$$

Using these expressions for the c-number functions $\{a_\pm\}$ and $\{a_0\}$, f of equation (12) is evaluated as [20]

$$f = \pm i\frac{\Phi}{2} \tag{A.25}$$

so that the c-number functions $\{A_\pm\}$ and $\{A_0\}$ of equations (10) and (11) are defined in terms of the ER parameters $\{\lambda, \Lambda\} \equiv \left\{\cos\frac{\Phi}{2}, \hat{\mathbf{n}}\sin\frac{\Phi}{2}\right\}$ as

$$A_\pm = -i\frac{n_\mp \sin\frac{\Phi}{2}}{\left[\cos\frac{\Phi}{2} + n_z \sin\frac{\Phi}{2}\right]} \equiv -i\frac{(\Lambda_x \mp \Lambda_y)}{\lambda + i\Lambda_z} \tag{A.26}$$

$$A_0 = \left[\cos\frac{\Phi}{2} + n_z \sin\frac{\Phi}{2}\right]^{-2} \equiv [\lambda + i\Lambda_z]^{-2} \tag{A.27}$$

$$\text{where} \quad n_\mp = n_x \mp i n_y \tag{A.28}$$

Using the notation of the generalized decomposition formula of equation (15), the ordered product $\mathcal{P}$ of rotation operators on the RHS of equation (19) can be written as

$$\mathcal{P} = \exp[a_+(2)I_+ + a_0(2)I_0 + a_-(2)I_-]\exp[a_+(1)I_+ + a_0(1)I_0 + a_-(1)I_-] \tag{A.29}$$

$$= \exp[A_+(2)I_+]\exp[\ln(A_0(2))I_0]\exp[A_-(2)I_-] \tag{A.30}$$

Using the recursion formula for $A_+(n)$ given in equation (16), we find

$$A_+(2) = \frac{-i(\Lambda_{2x} - i\Lambda_{2y}) + (\lambda_2 - i\Lambda_{2z})A_+(1)}{(\lambda_2 + i\Lambda_{2z}) - i(\Lambda_{2x} + i\Lambda_{2y})A_+(1)} \tag{A.31}$$

$$\text{where} \quad A_+(1) = -i\frac{(\Lambda_{1x} - i\Lambda_{1y})}{(\lambda_1 + i\Lambda_{1z})} \tag{A.32}$$

From equation (19), we must have $A_+ = A_+(2)$, so that from equations (25) and (30)

$$-i\frac{(\Lambda_x - i\Lambda_y)}{\lambda + i\Lambda_z} = \frac{-i(\Lambda_{2x} - i\Lambda_{2y})(\lambda_1 + i\Lambda_{1z}) + (\lambda_2 - i\Lambda_{2z})(-i)(\Lambda_{1x} - i\Lambda_{1y})}{(\lambda_2 + i\Lambda_{2z})(\lambda_1 + i\Lambda_{1z}) + (-i)(\Lambda_{2x} + i\Lambda_{2y})(-i)(\Lambda_{1x} - i\Lambda_{1y})} \tag{A.33}$$

or equivalently

$$\frac{\Lambda_y + i\Lambda_x}{\lambda + i\Lambda_z} = \frac{[\lambda_2\Lambda_{1y} + \lambda_1\Lambda_{2y} + \Lambda_{1x}\Lambda_{2z} - \Lambda_{1z}\Lambda_{2x}] + i[\lambda_2\Lambda_{1x} + \lambda_1\Lambda_{2x} + \Lambda_{1z}\Lambda_{2y} - \Lambda_{1y}\Lambda_{2z}]}{[\lambda_1\lambda_2 - \Lambda_{1x}\Lambda_{2x} - \Lambda_{1y}\Lambda_{2y} - \Lambda_{1z}\Lambda_{2z}] + i[\lambda_2\Lambda_{1z} + \lambda_1\Lambda_{2z} + \Lambda_{1y}\Lambda_{2x} - \Lambda_{1x}\Lambda_{2y}]} \tag{A.34}$$

From a classical point of view, the composition of the two homographies corresponding to the rotations in the rotation operator product of equation (19) yields the following

two equations [30, 32]:

$$\begin{aligned}\Lambda_y + i\Lambda_x &= [\lambda_2\Lambda_{1y} + \lambda_1\Lambda_{2y} + \Lambda_{1x}\Lambda_{2x} - \Lambda_{1z}\Lambda_{2x}] \\ &+ i[\lambda_2\Lambda_{1x} + \lambda_1\Lambda_{2x} + \Lambda_{1z}\Lambda_{2y} - \Lambda_{1y}\Lambda_{2z}] \qquad \text{(A.35)} \\ \lambda + i\Lambda_z &= [\lambda_1\lambda_2 - \Lambda_{1x}\Lambda_{2x} - \Lambda_{1y}\Lambda_{2y} - \Lambda_{1z}\Lambda_{2z}] \\ &+ i[\lambda_2\Lambda_{1z} + \lambda_1\Lambda_{2z} + \Lambda_{1y}\Lambda_{2x} - \Lambda_{1x}\Lambda_{2y}] \qquad \text{(A.36)}\end{aligned}$$

The quotient of these two equations is equation (33), and so we have recovered via the Santiago and [20] disentanglement procedure a relation which could have been obtained classically via stereographic projection. We return below to another connection between this disentanglement procedure and stereographic projection. Using the relations of equation (33) and those obtained via the recursion relations for $A_-(n)$ and $A_0(n)$, as well as the fact that all the ER parameter sets $\{\lambda, \Lambda\}$ define quaternions of unit norm (see equation (58) below), we find

$$\begin{aligned}\lambda &= \pm\{\lambda_1\lambda_2 - \Lambda_{1x}\Lambda_{2x} - \Lambda_{1y}\Lambda_{2y} - \Lambda_{1z}\Lambda_{2z}\} \qquad \text{(A.37)} \\ &= \pm\{\lambda_1\lambda_2 - \Lambda_1 \bullet \Lambda_2\} \qquad \text{(A.38)} \\ \Lambda_x &= \pm\{\lambda_2\Lambda_{1x} + \lambda_1\Lambda_{2x} + \Lambda_{1z}\Lambda_{2y} - \Lambda_{1y}\Lambda_{2z}\} \qquad \text{(A.39)} \\ &= \pm\{\lambda_2\Lambda_1 + \lambda_1\Lambda_2 - (\Lambda_1 \times \Lambda_2)\}_x \qquad \text{(A.40)} \\ \Lambda_y &= \pm\{\lambda_2\Lambda_{1y} + \lambda_1\Lambda_{2y} + \Lambda_{1x}\Lambda_{2z} - \Lambda_{1z}\Lambda_{2x}\} \qquad \text{(A.41)} \\ &= \pm\{\lambda_2\Lambda_1 + \lambda_1\Lambda_2 - (\Lambda_1 \times \Lambda_2)\}_y \qquad \text{(A.42)} \\ \Lambda_z &= \pm\{\lambda_2\Lambda_{1z} + \lambda_1\Lambda_{2z} + \Lambda_{1y}\Lambda_{2x} - \Lambda_{1x}\Lambda_{2y}\} \qquad \text{(A.43)} \\ &= \pm\{\lambda_2\Lambda_1 + \lambda_1\Lambda_2 - (\Lambda_1 \times \Lambda_2)\}_z \qquad \text{(A.44)}\end{aligned}$$

In terms of the resultant ER parameters $\{\lambda, \Lambda\}$, equation (A.37) corresponds to that part of the ER parameter composition rule which defines λ (Eq. A.(3)), while equations (A.39), (A.41) and (A.43) are the respective x,y and z components of the*vector* equation

defining Λ in the composition rule (equation (A.6)). In commenting on this result, it should be noted that the proof itself depends *only* on the fact that the spin angular momentum operators I_i satisfy the commutation relations of equations (A.8) and (A.9) for the generators of the SU(2) Lie algebra, and *not* on the spin magnitude I, which is arbitrary. The geometrical reason why *half*-angles appear in the composition rule is that a rotation through an angle θ about a given axis $\hat{\mathbf{n}}$ is equivalent to successive *reflections* in two planes that meet along this axis at the half-angle $\theta/2$ [28, 29, 3].

Appendix B

Evolution of The Density Matrix Coefficients

The evolution of the coefficients of the density matrix as a result of a quadrupole pulse sequence $S_x(T) - t_1 - S_y(T) - t_2 acquire$, where S_x is a 90° rotation about the x-axis, S_y is a refocusing pulse which is a 180° rotation about the y-axis.

Given an inversion operation about the x-axis $\Pi_x = e^{-i\pi I_x}$ and its inverse $\Pi_x^\dagger = e^{i\pi I_x}$, the parity of each basis operator is calculated. Each basis operator is then placed either in Table B.1 or in Table B.2 according to its parity: Table B.1 for basis operators with odd parity, table B.2 for basis operators with even parity.

Table B.1: This table shows the calculation of the odd parity for a spin inversion with respect to x

operator	odd parity
$\hat{a}_1 = \frac{1}{\sqrt{2}} I_x$	$\Pi_x \hat{a}_1 \Pi_x^\dagger = -\hat{a}_1$
$\hat{a}_2 = \frac{1}{\sqrt{2}} I_y$	$\Pi_x \hat{a}_2 \Pi_x^\dagger = -\hat{a}_2$
$\hat{a}_3 = \frac{1}{\sqrt{2}} Q_x$	$\Pi_x \hat{a}_3 \Pi_x^\dagger = -\hat{a}_3$
$\hat{a}_4 = \frac{1}{\sqrt{2}} D_y$	$\Pi_x \hat{a}_4 \Pi_x^\dagger = -\hat{a}_4$

$$\mathcal{H} = -\frac{\omega_Q}{3} Q_z - \omega_1 I_x \tag{B.45}$$

and in terms of the notation for the basis operators

$$\mathcal{H} = -\frac{\omega_Q}{3} \sqrt{6} \hat{b}_4 - \omega_1 \sqrt{2} \hat{b}_3 \tag{B.46}$$

Table B.2: This table shows the calculation of the even parity for a spin inversion with respect to x

operator	even parity
$\hat{b}_1 = \frac{1}{\sqrt{2}} D_x$	$\Pi_x \hat{b}_1 \Pi_x^\dagger = +\hat{b}_1$
$\hat{b}_2 = \frac{1}{\sqrt{2}} Q_y$	$\Pi_x \hat{b}_2 \Pi_x^\dagger = +\hat{b}_2$
$\hat{b}_3 = \frac{1}{\sqrt{2}} I_x$	$\Pi_x \hat{b}_3 \Pi_x^\dagger = +\hat{b}_3$
$\hat{b}_4 = \frac{1}{\sqrt{2}} Q_z$	$\Pi_x \hat{b}_4 \Pi_x^\dagger = +\hat{b}_4$

The Hamiltonian(B.46) is of even parity since it is expressed only in terms of the even basis operators, namely, b_4 and b_3. If the density matrix is also expressed in the same set of basis operators, then the LVN equation (2.16) in its matrix form will decompose into two submatrices along its diagonal. Each submatrix is in four dimensions

$$\begin{pmatrix} \dot{a_1} \\ \dot{a_2} \\ \dot{a_3} \\ \dot{a_4} \\ \dot{b_1} \\ \dot{b_2} \\ \dot{b_3} \\ \dot{b_4} \end{pmatrix} = \begin{bmatrix} 0 & -\omega_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \omega_1 & 0 & -\omega_Q & 0 & 0 & 0 & 0 & 0 \\ 0 & \omega_Q & 0 & -\omega_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \omega_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\omega_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \omega_1 & 0 & -\omega_Q & \sqrt{3}\omega_1 \\ 0 & 0 & 0 & 0 & 0 & \omega_Q & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\sqrt{3}\omega_1 & 0 & 0 \end{bmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} \tag{B.47}$$

where the upper submatrix on the diagonal is **A** and thc lower submatrix on the diagonal is **B**.

If the density matrix prior to the S_x pulse is along the z-axis, i.e. $a_1 = 1$, then evolution will occur only in the subspace spanned by the odd basis operators, $\{\hat{a}_i\}$. So

only the upper submatrix, **A**, of equation(B.47) is considered. After the end of the S_x pulse free precession takes place in a plane which is defined by an observable in-phase (I_y) $\hat{a}_2$ and an unobservable anti-phase (Q_x) $\hat{a}_3$. The Hamiltonian for the free precession is also of even parity.

$$\mathcal{H} = -\frac{\omega_Q}{3}\sqrt{6}\hat{b}_4 \tag{B.48}$$

Hence the evolution of the density matrix coefficients also takes place in the subspace which is spanned by the odd basis operators, $\{\hat{a}_i\}$. So again, only the upper submatrix, say $\tilde{\mathbf{A}}$, of equation (B.49) is considered

$$\begin{pmatrix} \dot{a}_1 \\ \dot{a}_2 \\ \dot{a}_3 \\ \dot{a}_4 \\ \dot{b}_1 \\ \dot{b}_2 \\ \dot{b}_3 \\ \dot{b}_4 \end{pmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\omega_Q & 0 & 0 & 0 & 0 & 0 \\ 0 & \omega_Q & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\omega_Q & 0 \\ 0 & 0 & 0 & 0 & 0 & \omega_Q & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} \tag{B.49}$$

As for the refocusing pulse, S_y, the situation changes since now the Hamiltonian contains an operator of odd parity (I_y)$\hat{a}_2$ to account for for this refocusing pulse. S_y is also referred to as a 90° pulse along the y-axis. The effect of the refocusing pulse is to flip (reflect or rotate by 180°) about the y-axis

$$\mathcal{H} = -\frac{\omega_Q}{3}Q_z - \omega_1 I_y \tag{B.50}$$

and in terms of the notation of the basis operators, Table B.3 and Table B.4

$$\mathcal{H} = -\frac{\omega_Q}{3}\sqrt{6}\hat{b}_4 - \omega_1\sqrt{2}\hat{a}_2 \tag{B.51}$$

The matrix form of the LVN equation(2.16) will now be different as commutation relations of Hamiltonian(B.51) are not the same as in case of, S_x. The parity of some of the basis operators may change with respect to an inversion about the y-axis, $\Pi_y = e^{-i\pi I_y}$. To keep operators of different parity in separate subsets these operators are rearranged as shown below in Tables B.3 and B.4

Table B.3: This table shows the calculation of the odd parity for a spin inversion with respect to y

operator	odd parity
$\hat{a}_1 = \frac{1}{\sqrt{2}}I_z$	$\Pi_y\hat{a}_1\Pi_y^\dagger = -\hat{a}_1$
$\hat{b}_2 = \frac{1}{\sqrt{2}}Q_y$	$\Pi_y\hat{b}_2\Pi_y^\dagger = -\hat{b}_2$
$\hat{b}_3 = \frac{1}{\sqrt{2}}I_x$	$\Pi_y\hat{b}_3\Pi_y^\dagger = -\hat{b}_3$
$\hat{a}_4 = \frac{1}{\sqrt{2}}D_y$	$\Pi_y\hat{a}_4\Pi_y^\dagger = -\hat{a}_4$

Table B.4: This table shows the calaculation of the even parity for a spin inversion with respect to y

operator	even parity
$\hat{b}_1 = \frac{1}{\sqrt{2}}D_y$	$\Pi_y\hat{b}_1\Pi_y^\dagger = +\hat{b}_1$
$\hat{a}_2 = \frac{1}{\sqrt{2}}I_y$	$\Pi_y\hat{a}_2\Pi_y^\dagger = +\hat{a}_2$
$\hat{a}_3 = \frac{1}{\sqrt{2}}Q_x$	$\Pi_y\hat{a}_3\Pi_y^\dagger = +\hat{a}_3$
$\hat{b}_4 = \frac{1}{\sqrt{2}}Q_z$	$\Pi_y\hat{b}_4\Pi_y^\dagger = +\hat{b}_4$

$$\begin{pmatrix} \dot{a_1} \\ \dot{b_3} \\ \dot{b_2} \\ \dot{a_4} \\ \dot{b_1} \\ \dot{a_3} \\ \dot{a_2} \\ \dot{b_4} \end{pmatrix} = \begin{bmatrix} 0 & \omega_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\omega_1 & 0 & \omega_Q & 0 & 0 & 0 & 0 & 0 \\ 0 & -\omega_Q & 0 & \omega_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\omega_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\omega_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \omega_1 & 0 & \omega_Q & -\sqrt{3}\omega_1 \\ 0 & 0 & 0 & 0 & 0 & -\omega_Q & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sqrt{3}\omega_1 & 0 & 0 \end{bmatrix} \begin{pmatrix} a_1 \\ b_3 \\ b_2 \\ a_4 \\ b_1 \\ a_3 \\ a_2 \\ b_4 \end{pmatrix} \tag{B.52}$$

where the upper submatrix on the diagonal is just $-\mathbf{A}$ of equation (B.47). The lower submatrix on the diagonal, however, is not related to $\mathbf{B}$ in a straight forward way. The evolution now switches to the subspace which is spanned by the even basis operators, Table B.4. In this case only the lower submatrix of equation(B.52) is considered. After the application of the refocusing pulse (S_y) free precession takes place, again, according to equation(B.49).

At this stage we have a evolution equation in matrix form for each part of the quadrupole echo. The solution of each such evolution equation has the exponential of its corresponding matrix as a propagator.

For the purpose of quaternion calculations each such matrix has to be expressed in terms of its rotation generators J_i and K_i. Submatrices $\mathbf{A}$ and $\mathbf{B}$ have already been found to be [1]

$$\mathbf{A} = \omega_Q K_1 - (\omega_Q J_1 + 2\omega_1 J_2) \tag{B.53}$$

The evolution of the density matrix coefficients, initially at $a_1 = 1$, as a result of a single

pulse of a duration τ [1] is given by

$$\mathbf{a}(\tau) = \exp(\omega_Q \tau K_1) \exp(-\tau(\omega_Q J_1 + 2\omega_1 J_2))\mathbf{a}(0) \tag{B.54}$$

The upper submatrix, say $\tilde{\mathbf{A}}$, of equation(B.49) for free precession is given by

$$\tilde{\mathbf{A}} = \omega_Q(K_1 - J_1) \tag{B.55}$$

The lower submatrix, say $\tilde{\mathbf{B}}$, of equation(B.52) is found to be by inspection

$$\begin{aligned} \tilde{\mathbf{B}} &= \omega_Q(J_1 - K_1) - \omega_1(J_2 + K_2) + \sqrt{3}\omega_1(J_3 - K_3) && \text{(B.56)} \\ &= \omega_Q J_1 - \omega_1 J_2 + \sqrt{3}\omega_1 J_3 - (\omega_Q K_1 + \omega_1 K_2 + \sqrt{3}\omega_1 K_3) && \text{(B.57)} \end{aligned}$$

which can be rearranged to look like

$$\tilde{\mathbf{B}} = \exp(\pi/6 J_1)(\omega_Q J_1 + 2\omega_1 J_3) \exp(-\pi/6 J_1) + \exp(-\pi/6 K_1)(\omega_Q K_1 + 2\omega_1 K_3) \exp(\pi/6 K_1) \tag{B.58}$$

The submatrix for free precession after the refocusing pulse is just equation(B.55)

$$\tilde{\mathbf{A}} = \omega_Q(K_1 - J_1) \tag{B.59}$$

since the evolution of the density matrix coefficients obeys equation (B.49). The Hamiltonian regains even parity. So now evolution occurs in the subspace spanned by the odd basis operators of Table B.1.

Now that we have submatrices $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, and $\mathbf{A}$, in terms of the rotation generators J_i and K_i the evolution of the density matrix coefficients can be given in a quaternion form

$$\mathbf{a}(t) = \exp(\tilde{\mathbf{A}} t_2) \exp(\tilde{\mathbf{B}} T) \exp(\tilde{\mathbf{A}} t_1) \exp(\mathbf{A} T)\mathbf{a}(0) \tag{B.60}$$

By using the two composition rules for quaternions it is possible to simplify the product of the four quaternions (in ER-parameter set $\{\lambda, \Lambda\}$), that make up equation(B.60).

Thus the expressions of the density matrix coefficients are much simpler. Given in their simplified form these density matrix coefficients can be used to express the in-phase (B.61).

where $\delta = t_2 - (t_1 + T/2)$ and $\sigma = t_2 + t_1 + T/2$,and where T is the duration of each pulse (S_x and S_y), with t_1 as the time spacing between the above pulses, and $t_2 + T/2$ as the echo peak-time relative to S_y.

Equation(B.61) describes the evolution of the coefficients of the observable in-phase in terms of the interpulse time spacing, t_1, and the echo-peak time t_2 after the second pulse (S_y) i.e.

$$\begin{aligned} < I_y(t_1,t_2) > &= (\Lambda_2^2 + \Lambda_3^2)\{\Lambda_2 \cos\omega_Q\delta - \Lambda_3 \sin\omega_Q\delta\} \\ &+ (\lambda^2 - \Lambda^2)\{\Lambda_2 \cos\omega_Q\sigma - \Lambda_3 \sin\omega_Q\sigma\} - 2\lambda\Lambda_1\{\Lambda_3 \cos\omega_Q\sigma + \Lambda_2 \sin\omega_Q\sigma\} \end{aligned} \tag{B.61}$$

The first term on the RHS of equation(B.61) can be rearranged as follows:

$$< I_y(t_1,t_2) >_{in-phase} = (\Lambda_2^2 + \Lambda_3^2)^{3/2}\{\cos(\phi)\cos(\omega_Q\delta) - \sin(\phi)\sin(\omega_Q\delta)\} \tag{B.62}$$

$$< I_y(t_1,t_2) >_{in-phase} = (\Lambda_2^2 + \Lambda_3^2)^{3/2}\cos(\omega_Q\delta + \phi) \tag{B.63}$$

the corresponding component of the anti-phase is 90° phase shifted with respect to the in-phase signal (B.62)

$$< I_y(t_1,t_2) >_{anti-phase} = (\Lambda_2^2 + \Lambda_3^2)^{3/2}\{\sin(\phi)\cos(\omega_Q\delta) + \cos(\phi)\sin(\omega_Q\delta)\} \tag{B.64}$$

$$< I_y(t_1,t_2) >_{anti-phase} = (\Lambda_2^2 + \Lambda_3^2)^{3/2}\sin(\omega_Q\delta + \phi) \tag{B.65}$$

where we have used the following equations:

$$\omega_e = (\Lambda_2^2 + \Lambda_3^2)^{1/2} \tag{B.66}$$

$$\cos(\phi) = \Lambda_2/\omega_e \tag{B.67}$$

$$\sin(\phi) = \Lambda_3/\omega_e \tag{B.68}$$

Appendix C

Quatech Computer Programs

```
c ************************************************************
c *                  MCYDUR5a.FOR                            *
c *                  SHAPED PULSE WAVE                       *
c *                                                          *
c *     This Quatech Program controls the Quatech
c *        Wave generator as in a QUASH experiment. It
c *        contains the QUASH pulse in an array.
c *
c *
c *
c *      This example programs the WSB-100 to produce a      *
c *   Shaped Pulse. Each call is followed by a status check  *
c *   to test for errors encountered during the routine.     *
c *   Program execution is as follows:                       *
c *                                                          *
c *      1. Call INIT to initialize parameters               *
c *         a) Board being initialized = 1                   *
c *         b) Base address = 300H                           *
c *         c) Assumed type of module being used is the      *
c *            WSB-A12M analog module                        *
c *         d) DMA channel not utilized                      *
c *         e) Interrupt level not utilized                  *
c *                                                          *
c *      2. Call WSTOP to perform a synchronous stop         *
c *         a) Do not enable hold mode                       *
c *         b) Wait for WSB-100 to enter program mode        *
c *                                                          *
c *      3. Call DEFWAV to define waveform                   *
c *         a) Address in WSB-100 memory = 0H                *
c *         b) Data is stored in sequentially higher         *
c *            address locations                             *
c *         c) Number of points to be stored = 360           *
c *         d) Both positive and negative data provided      *
c *         e) Waveform array = WAVE(NPTS)                   *
c *                                                          *
c *      4. Call SETCLK to specify source and frequency      *
c *         of the data output clock                         *
c *         a) Internal clock source selected (CLKSRC=0,1)   *
c *         b) Low bit used for frequency divider = 3        *
c *         c) High bit used for frequency divider = 100     *
c *         d) The data is clocked out at a frequency        *
c *            given by f=40*10^6/[(RATELO+1)*(RATEHI)]      *
c *                                                          *
c *      5. Call SETRPT to specify number of repitions       *
c *         a) Waveform set to repeat as indicated by user   *
c *                                                          *
c *      6. Call WSCALE to set scale                         *
c *         a) Full scale setting selected                   *
c *                                                          *
c *      7. Call SETDLY to set delay                         *
c *         a) Disable delay between cycles                  *
c *                                                          *
c *      8. Call SETRIG to set trigger                       *
c *         a) Internal trigger source selected              *
c *         b) Returns to program mode after task            *
c *                                                          *
c *      9. Call WSTART to begin waveform                    *
c *         a) Auto-reverse disabled                         *
c *                                                          *
c ************************************************************

      INTEGER*2 ACTBRD, BASEADDR, MODULE, DMACH, IRQ, STATUS
```

```
      INTEGER*2 WHOLD, WWAIT, WBASE, ADIR, NPTS, NBITS
      INTEGER*2 CLKSRC, RATELO, RATEHI, REPT, MULT
      INTEGER*2 DLAY, TSRC, TLAY, RETRIG, AREV, I
      INTEGER*2 WAVE(400),xt,FREQ,NDR,NPTSDF
      REAL*8 P,T,W(400),MX,CH0,CH1,CH2,CH3,B,PI,WS,AMP,RF,DUR
      REAL*8 DUR_RSL,DUR1,DFDUR,TPOINT
      CHARACTER ANS
c *                                                              *
c *                    Initialize WSB-100                        *
c *                                                              *
c *    This sets the active board, the base address, the         *
c *    module used, the DMA channel, the interrupt level,        *
c *    and the operating status.                                 *
c *                                                              *
      ACTBRD = 1
      BASEADDR = 768
      MODULE = 2
      DMACH = 0
      IRQ = 0
      STATUS = 65535
      CALL INIT(ACTBRD,BASEADDR,MODULE,DMACH,IRQ,STATUS)
      IF (STATUS .NE. 0) THEN
         WRITE(*,100) STATUS
 100     FORMAT(3X,'Status code = ',I3)
      ENDIF
c *                                                              *
c *                    Set synchronous stop                      *
c *                                                              *
c *    This disables the hold mode and waits for the             *
c *    program mode.                                             *
c *                                                              *
      WHOLD = 0
      WWAIT = 1
      CALL WSTOP(WHOLD,WWAIT)
      IF (STATUS .NE. 0) THEN
         WRITE(*,200) STATUS
 200     FORMAT(3X,'Status code = ',I3)
      ENDIF
c *                                                              *
c *                      Define waveform                         *
c *                                                              *
c *    This sets the memory address, the memory direction,       *
c *    the number of points, the number of bits, and the         *
c *    sine wave.                                                *
c *                                                              *
      WBASE = 0
      ADIR = 1
C      NPTS = 360

c                    CONTROL LOOP OF QUATECH
c                    =======================
 999     NPTS=100
C       WRITE(*,*) 'CHANGE NUMBER OF POINTS OF SHPED PULSE? CURRENTLY 100 '
      WRITE(*,*) 'NEW NUMBER OF POINTS? Y->YES'
      READ(*,'(A)') ANS
      IF(ANS.EQ.'Y') THEN
      WRITE(*,*)'NEW NUMBER OF POINTS ?'
      READ(*,*) NPTS
      ENDIF
      NBITS = 12
      P = 1.0
      MX = 0.0
```

```
         CH0=1.0
         CH1=-7.05
         CH2=-.225
         CH3=11.0
C         B=SQRT(40.0)
         B=6.45
        DO 21 I = 1,NPTS
           T = B*(REAL(I-1)/REAL(NPTS-1)-.5)
           W(I) =(CH0+CH1*T+CH2*T*T+CH3*T*T*T)*EXP(-P*T*T)
           IF (MX.LT.ABS(W(I))) MX = ABS(W(I))
c           print*, 'w = ',W(I),'  T= ',T
c           READ(*,*)
 21    CONTINUE
           W(1)=MX/4.0
           W(NPTS)=MX/4.0
           print*, 'MAX OF SHAPE= ',MX
           WRITE(*,*)'AMPLITUDE OF WAVEFOR ? (2047 is maximum)'
           READ(*,*) AMP
        DO 210 I = 1,NPTS
c           WAVE(I) = 2047*(W(I)/MX)

C            W(I) = (W(I)/MX)
C            WAVE(I)=AMP*W(I)/NPTS
             WAVE(I) =AMP*(W(I)/MX)
c           print*, 'wave = ',WAVE(I),'T= ',T
c           READ(*,*)
 210    CONTINUE
C NORMALIZE SHAPED PULSE
        DO I=1,NPTS
        W(I)=W(I)/MX
        ENDDO

        PI=4.0*ATAN(1.0)
        WS=0.0
        DO 220 I=1,NPTS
        WS=WS+W(I)
 220    CONTINUE

        WRITE(*,*)'(AREA OF SHAPE)/DT)',WS

        WRITE(*,*)'WHAT RF (IN KHZ)'
        READ(5,*) RF
        RF=RF*1000.0
C ADJUST OUTPUT RATE OF QUATECH TO MATCH TIME-STEP, DLT, OF SAHPED PULSE
        DLT=(PI/2.0)/(ABS(WS)*RF*2.0*PI)
        WRITE(*,*)'CALCULATED TIME STEP: ',DLT
        WRITE(*,*)'CALCULATED DURATION: ' ,NPTS*DLT
        WRITE(*,*)'************************'
        DUR=NPTS*DLT
        RATEHI=1 ! THIS PARAMETER DOES NOT MAKE A DIFFERENCE
        RATELO=DLT/((25.0E-9)*RATEHI)-1.0
        WRITE(*,*) 'DLT=',DLT,'  RATEHI=',RATEHI,'  RATELO=',RATELO
            DUR_RSL=25.0E-9*NPTS
            WRITE(*,*)'RESOLUTION OF DURATION IS ',DUR_RSL
            NDR=DUR/DUR_RSL
            DUR1=DUR_RSL*(RATELO+1)*1    ! NEAREST VALUE TO DESIRED DURATION
            WRITE(*,*)'NDR= ',NDR
            WRITE(*,*)'ACTUAL DURATION OF SHAPED PULSE IS ',DUR1
            DFDUR=DUR-DUR1
            WRITE(*,*)'DEVIATION FROM DESIRED DURATION ',DFDUR,DUR_RSL
            TPOINT=25.0E-9*(RATELO+1)*RATEHI
```

```
            WRITE(*,*)'TIME PER POINT IS ',TPOINT,' DLT=',DLT
            NPTSDF=DFDUR/TPOINT
            WRITE(*,*)'CHANGE NUMBER OF POINTS BY ',NPTSDF
            NPTS=NPTS+NPTSDF
            WRITE(*,*)'ADJUSTED NUMBER OF POINTS IS ',NPTS
C            DUR=TPOINT*NPTS
C  ******UPDATE SHAPED PULSE******
          MX=0.0
          DO I = 1,NPTS
           T = B*(REAL(I-1)/REAL(NPTS-1)-.5)
           W(I) =(CH0+CH1*T+CH2*T*T+CH3*T*T*T)*EXP(-P*T*T)
           IF (MX.LT.ABS(W(I))) MX = ABS(W(I))
c           print*, 'w = ',W(I),'  T= ',T
c           READ(*,*)
          ENDDO
           W(1)=MX/4.0
           W(NPTS)=MX/4.0
           print*, 'MAX OF SHAPE= ',MX
           WRITE(*,*)'AMPLITUDE OF WAVEFOR ? (2047 is maximum)'
           READ(*,*) AMP
          DO I = 1,NPTS
c           WAVE(I) = 2047*(W(I)/MX)

C            W(I) = (W(I)/MX)
C            WAVE(I)=AMP*W(I)/NPTS
             WAVE(I) =AMP*(W(I)/MX)
c           print*, 'wave = ',WAVE(I),'T= ',T
c           READ(*,*)
          ENDDO

        print*, 'Change duration of shaped pulse?'
        print*, '(Yes 1    No 0)'
        read(5,*) xt
        IF (xt.eq.1) THEN    !*****************
            WRITE(*,*)'WHAT DURATION? (IN Microseconds)'
            READ(5,*) DUR
            DUR=DUR*1.0E-6

            RATEHI=1 ! THIS PARAMETER DOES NOT MAKE A DIFFERENCE
            DLT=DUR/NPTS
            RATELO=DLT/((25.0E-9)*RATEHI)-1.0
            WRITE(*,*) 'DLT=',DLT,'  RATEHI=',RATEHI,'  RATELO=',RATELO
            DUR_RSL=25.0E-9*NPTS
            WRITE(*,*)'RESOLUTION OF DURATION IS ',DUR_RSL
            NDR=DUR/DUR_RSL
            DUR1=DUR_RSL*(RATELO+1)*1    ! NEAREST VALUE TO DESIRED DURATION
            WRITE(*,*)'NDR= ',NDR
            WRITE(*,*)'ACTUAL DURATION OF SHAPED PULSE IS ',DUR1
            DFDUR=DUR-DUR1
            WRITE(*,*)'DEVIATION FROM DESIRED DURATION ',DFDUR,DUR_RSL
            TPOINT=25.0E-9*(RATELO+1)*RATEHI
            WRITE(*,*)'TIME PER POINT IS ',TPOINT
            NPTSDF=DFDUR/TPOINT +1
            WRITE(*,*)'CHANGE NUMBER OF POINTS BY ',NPTSDF
            NPTS=NPTS+NPTSDF
            WRITE(*,*)'ADJUSTED NUMBER OF POINTS IS ',NPTS
            RATELO=TPOINT/(25.0E-9*RATEHI)-1
            WRITE(*,*)'ADJUSTED RATELO ',RATELO
C  ******UPDATE SHAPED PULSE******
          MX=0.0
          DO  I = 1,NPTS
           T = B*(REAL(I-1)/REAL(NPTS-1)-.5)
```

```
          W(I) =(CH0+CH1*T+CH2*T*T+CH3*T*T*T)*EXP(-P*T*T)
          IF (MX.LT.ABS(W(I))) MX = ABS(W(I))
c          print*, 'w = ',W(I),'  T= ',T
c          READ(*,*)
         ENDDO
          W(1)=MX/4.0
          W(NPTS)=MX/4.0
          print*, 'MAX OF SHAPE= ',MX
          WRITE(*,*)'AMPLITUDE OF WAVEFOR ? (2047 is maximum)'
          READ(*,*) AMP
         DO  I = 1,NPTS
c          WAVE(I) = 2047*(W(I)/MX)

C           W(I) = (W(I)/MX)
C           WAVE(I)=AMP*W(I)/NPTS
            WAVE(I) =AMP*(W(I)/MX)
c          print*, 'wave = ',WAVE(I),'T= ',T
c          READ(*,*)
         ENDDO

       ENDIF    !*****END OF CONTROL LOOP***************

       CALL DEFWAV(WBASE,ADIR,NPTS,NBITS,WAVE(1))
       IF (STATUS .NE. 0) THEN
          WRITE(*,300) STATUS
 300      FORMAT(3X,'Status code = ',I3)
       ENDIF
c *                                                              *
c *                          Set clock                           *
c *                                                              *
c *    This sets the internal clock, the low bit divider,        *
c *    and the high bit divider.                                 *
c *                                                              *
        CLKSRC = 0

       CALL SETCLK(CLKSRC,RATELO,RATEHI)
       IF (STATUS .NE. 0) THEN
          WRITE(*,400) STATUS
 400      FORMAT(3X,'Status code = ',I3)
       ENDIF
c *                                                              *
c *                        Set repitions                         *
c *                                                              *
c *    NOTE: REPT=0 sets a continuous wavefrom.                  *
c *                                                              *

       print*,'Number of repetitons?'
       READ(5,*) REPT

       CALL SETRPT(REPT)
       IF (STATUS .NE. 0) THEN
          WRITE(*,500) STATUS
 500      FORMAT(3X,'Status code = ',I3)
       ENDIF
c *                                                              *
c *                          Set scale                           *
c *                                                              *
c *    This sets a full scale.                                   *
c *                                                              *
       MULT = 127
       CALL WSCALE(MULT)
       IF (STATUS .NE. 0) THEN
```

```
         WRITE(*,600) STATUS
 600     FORMAT(3X,'Status code = ',I3)
      ENDIF
c *                                                              *
c *                          Set delay                           *
c *                                                              *
c *    This sets a delay of zero.                                *
c *                                                              *
      DLAY = 0
      CALL SETDLY(DLAY)
      IF (STATUS .NE. 0) THEN
         WRITE(*,700) STATUS
 700     FORMAT(3X,'Status code = ',I3)
      ENDIF
c *                                                              *
c *                         Set trigger                          *
c *                                                              *
c *    This sets an internal trigger and then returns to         *
c *    program mode.                                             *
c *                                                              *
c *    TSRC:  0 for internal trigger mode
c *           otherwise for external trigger mode
c *
c *    RETRIG: 0 return to 'program' mode after task
c *             otherwise remain in 'run' mode after task


      TSRC = 0
      RETRIG = 0
      print*,'External trigger mode ?'
      print*,'(Y=1,N=0, any other value end program)'
      read(5,*) xt
      IF (xt.eq.1) then
          print*,'TSRC ?'
          read(5,*) TSRC
          print*,'RETRIG ?'
          read(5,*) RETRIG
      ELSE
          goto 1000

      ENDIF

      CALL SETRIG(TSRC,TLAY,RETRIG)
      IF (STATUS .NE. 0) THEN
         WRITE(*,800) STATUS
 800     FORMAT(3X,'Status code = ',I3)
      ENDIF
c *                                                              *
c *                       Start waveform                         *
c *                                                              *
c *    This disables the auto-reverse and begins waveform.       *
c *                                                              *
      AREV = 0
      CALL WSTART(WBASE,ADIR,NPTS,AREV)
      IF (STATUS .NE. 0) THEN
         WRITE(*,900) STATUS
 900     FORMAT(3X,'Status code = ',I3)
      ENDIF
      goto 999
 1000 END••••••••••••••••••••••
```

```
C ************************************************************
C *                SYMTRIC1.FOR   (SHP2OP1.FOR)              *
C *                  SHAPED PULSE WAVE                       *
C *                                                          *
C *     This program controls the QUATECH Wave Generator
C *         In a SQUASH experiment. It contains the SUQUASH
C *         pulse in an array.
C *
C *      This example programs the WSB-100 to produce a      *
C *  Shaped Pulse. Each call is followed by a status check   *
C *  to test for errors encountered during the routine.      *
C *  Program execution is as follows:                        *
C *                                                          *
C *      1. Call INIT to initialize parameters               *
C *         a) Board being initialized = 1                   *
C *         b) Base address = 300H                           *
C *         c) Assumed type of module being used is the      *
C *            WSB-A12M analog module                        *
C *         d) DMA channel not utilized                      *
C *         e) Interrupt level not utilized                  *
C *                                                          *
C *      2. Call WSTOP to perform a synchronous stop         *
C *         a) Do not enable hold mode                       *
C *         b) Wait for WSB-100 to enter program mode        *
C *                                                          *
C *      3. Call DEFWAV to define waveform                   *
C *         a) Address in WSB-100 memory = 0H                *
C *         b) Data is stored in sequentially higher         *
C *            address locations                             *
C *         c) Number of points to be stored = 360           *
C *         d) Both positive and negative data provided      *
C *         e) Waveform array = WAVE(NPTS)                   *
C *                                                          *
C *      4. Call SETCLK to specify source and frequency      *
C *         of the data output clock                         *
C *         a) Internal clock source selected (CLKSRC=0,1)   *
C *         b) Low bit used for frequency divider = 3        *
C *         c) High bit used for frequency divider = 100     *
C *         d) The data is clocked out at a frequency        *
C *            given by f=40*10^6/[(RATELO+1)*(RATEHI)]      *
C *                                                          *
C *      5. Call SETRPT to specify number of repitions       *
C *         a) Waveform set to repeat as indicated by user   *
C *                                                          *
C *      6. Call WSCALE to set scale                         *
C *         a) Full scale setting selected                   *
C *                                                          *
C *      7. Call SETDLY to set delay                         *
C *         a) Disable delay between cycles                  *
C *                                                          *
C *      8. Call SETRIG to set trigger                       *
C *         a) Internal trigger source selected              *
C *         b) Returns to program mode after task            *
C *                                                          *
C *      9. Call WSTART to begin waveform                    *
C *         a) Auto-reverse disabled                         *
C *                                                          *
C ************************************************************

      INTEGER*2 ACTBRD, BASEADDR, MODULE, DMACH, IRQ, STATUS
      INTEGER*2 WHOLD, WWAIT, WBASE, ADIR, NPTS, NBITS
      INTEGER*2 CLKSRC, RATELO, RATEHI, REPT, MULT
```

```
      INTEGER*2 DLAY, TSRC, TLAY, RETRIG, AREV, I
      INTEGER*2 WAVE(400),xt,FREQ,NDR,NPTSDF,J,NPTS1
      REAL*8 P,T,W(400),MX,CH0,CH1,CH2,CH3,B,PI,WS,AMP,RF,DUR
      REAL*8 DUR_RSL,DUR1,DFDUR,TPOINT,FLIP
      CHARACTER ANS
C *                                                          *
C *                    Initialize WSB-100                    *
C *                                                          *
C *    This sets the active board, the base address, the     *
C *    module used, the DMA channel, the interrupt level,    *
C *    and the operating status.                             *
C *                                                          *
      ACTBRD = 1
      BASEADDR = 768
      MODULE = 2
      DMACH = 0
      IRQ = 0
      STATUS = 65535
      CALL INIT(ACTBRD,BASEADDR,MODULE,DMACH,IRQ,STATUS)
      IF (STATUS .NE. 0) THEN
         WRITE(*,100) STATUS
 100     FORMAT(3X,'Status code = ',I3)
      ENDIF
C *                                                          *
C *                  Set synchronous stop                    *
C *                                                          *
C *    This disables the hold mode and waits for the         *
C *    program mode.                                         *
C *                                                          *
      WHOLD = 0
      WWAIT = 1
      CALL WSTOP(WHOLD,WWAIT)
      IF (STATUS .NE. 0) THEN
         WRITE(*,200) STATUS
 200     FORMAT(3X,'Status code = ',I3)
      ENDIF
C *                                                          *
C *                     Define waveform                      *
C *                                                          *
C *    This sets the memory address, the memory direction,   *
C *    the number of points, the number of bits, and the     *
C *    sine wave.  QUATECH PREPARATION FOR SQUASH STRAT HERE
                    *
C *                                                          *
      WBASE = 0
      ADIR = 1
C      NPTS = 360

C                      CONTROL LOOP OF QUATECH
C                      =======================
 999     NPTS=100
C       WRITE(*,*) 'CHANGE NUMBER OF POINTS OF SHPED PULSE? CURRENTLY 100 '
      WRITE(*,*) 'NEW NUMBER OF POINTS? Y->YES'
      READ(*,'(A)') ANS
      IF(ANS.EQ.'Y') THEN
      WRITE(*,*)'NEW NUMBER OF POINTS ?'
      READ(*,*) NPTS1
      ENDIF
      NPTS=NPTS1
      NBITS = 12
      P = 1.0
      MX = 0.0
```

```
        CH0=1.2
        CH1=8.7
        CH2=0.1
        CH3=11.7
C        B=SQRT(40.0)
        B=2.5
C       WRITE(*,*)'DOMAIN OF SHAPE ?'
C       READ(5,*) B
       DO 21 I = 1,NPTS
C          T = B*(REAL(I-1)/REAL(NPTS-1)-.5)
         T = B*(FLOAT(I-1)/FLOAT(NPTS-1)-.5)
         IF(ABS(T).LT.1E-16) T=1.0E-8
          W(I) = (CH0*COS(CH1*T)-EXP(-T*T)*CH2*SIN(CH3*T)/T)*EXP(-T**6)
          IF (MX.LT.ABS(W(I))) MX = ABS(W(I))
 21   CONTINUE

C  BEGINNIG AND END MARKERS
C          W(1)=MX/4.0
C          W(NPTS)=MX/4.0
          print*, 'MAX OF SHAPE= ',MX
C          WRITE(*,*)'AMPLITUDE OF WAVEFOR ? (2047 is maximum)'
C          READ(*,*) AMP
           AMP=2047
       DO 210 I = 1,NPTS
C           WAVE(I)=AMP*W(I)/NPTS
            WAVE(I) =AMP*(W(I)/MX)
c          print*, 'wave = ',WAVE(I),'T= ',T
c          READ(*,*)
 210   CONTINUE
C NORMALIZE SHAPED PULSE
       DO I=1,NPTS
       W(I)=W(I)/MX
       ENDDO

       WRITE(*,*)'WHAT RF? (IN KHZ) 47'
       READ(5,*) RF
       PI=4.0*ATAN(1.0)
       RF=2.0*PI*RF*1000.0
       DO I=1,NPTS
        W(I)=RF*W(I)
       ENDDO
       WS=0.0
       DO I=1,NPTS
        WS=WS+W(I)
       ENDDO

C ADJUST OUTPUT RATE TO MATCH TIME-STEP, DLT, OF SAHPED PULSE
       DLT=(PI/2.0)/(ABS(WS))
       WRITE(*,*)'CALCULATED TIME STEP: ',DLT
       WRITE(*,*)'CALCULATED DURATION: ' ,NPTS*DLT
       WRITE(*,*)'************************'
       DUR=NPTS*DLT
       RATEHI=1 ! THIS PARAMETER DOES NOT MAKE A DIFFERENCE
       RATELO=DLT/((25.0E-9)*RATEHI)-1.0
       WRITE(*,*) 'DLT=',DLT,'  RATEHI=',RATEHI,'  RATELO=',RATELO
           DUR_RSL=25.0E-9*NPTS
           WRITE(*,*)'RESOLUTION OF DURATION IS ',DUR_RSL
           NDR=DUR/DUR_RSL
           DUR1=DUR_RSL*(RATELO+1)*1    ! NEAREST VALUE TO DESIRED DURATION
           WRITE(*,*)'NDR= ',NDR
           WRITE(*,*)'ACTUAL DURATION OF SHAPED PULSE IS ',DUR1
```

```
            DFDUR=DUR-DUR1
            WRITE(*,*)'DEVIATION FROM DESIRED DURATION ',DFDUR,DUR_RSL
            TPOINT=25.0E-9*(RATELO+1)*RATEHI
            WRITE(*,*)'TIME PER POINT IS ',TPOINT,' DLT=',DLT
            NPTSDF=DFDUR/TPOINT
            WRITE(*,*)'CHANGE NUMBER OF POINTS BY ',NPTSDF
            NPTS=NPTS+NPTSDF
            WRITE(*,*)'ADJUSTED NUMBER OF POINTS IS ',NPTS
C            DUR=TPOINT*NPTS
C  ******UPDATE SHAPED PULSE******
          MX=0.0
          DO I = 1,NPTS
C           T = B*(REAL(I-1)/REAL(NPTS-1)-.5)
           T = B*(FLOAT(I-1)/FLOAT(NPTS-1)-.5)
          IF(ABS(T).LT.1E-16) T=1.0E-8

           W(I) = (CH0*COS(CH1*T)-EXP(-T*T)*CH2*SIN(CH3*T)/T)*EXP(-T**6)

C           W(I) =(CH0+CH1*T+CH2*T*T+CH3*T*T*T)*EXP(-P*T*T)
           IF (MX.LT.ABS(W(I))) MX = ABS(W(I))
c           print*, 'w = ',W(I),'  T= ',T
c           READ(*,*)
          ENDDO
C           W(1)=MX/4.0
C           W(NPTS)=MX/4.0
           print*, 'MAX OF SHAPE= ',MX
C           WRITE(*,*)'AMPLITUDE OF WAVEFOR ? (2047 is maximum)'
C           READ(*,*) AMP
            AMP=2047
          DO I = 1,NPTS
c           WAVE(I) = 2047*(W(I)/MX)

C            W(I) = (W(I)/MX)
C            WAVE(I)=AMP*W(I)/NPTS
             WAVE(I) =AMP*(W(I)/MX)
c           print*, 'wave = ',WAVE(I),'T= ',T
c           READ(*,*)
          ENDDO
C
        DO I=1,NPTS
         W(I)=W(I)/MX
        ENDDO
        DO I=1,NPTS
         W(I)=RF*W(I)
        ENDDO
        WS=0.0
        DO I=1,NPTS
         WS=WS+W(I)
        ENDDO
        WRITE(*,*)'DURATION OF ADJUSTED SHAPE ',NPTS*TPOINT
        FLIP=WS*TPOINT
         WRITE(*,*)'FLIP ANGLE OF ADJUSTED SHAPED PULSE ',FLIP
        print*, 'Change duration of shaped pulse?'
        print*, '(Yes 1   No 0)'
        read(5,*) xt
        IF (xt.eq.1) THEN   !*****************
            WRITE(*,*)'WHAT DURATION? (IN Microseconds)'
            READ(5,*) DUR
            DUR=DUR*1.0E-6

            RATEHI=1 ! THIS PARAMETER DOES NOT MAKE A DIFFERENCE
            DLT=DUR/NPTS
```

```
            RATELO=DLT/((25.0E-9)*RATEHI)-1.0
            WRITE(*,*) 'DLT=',DLT,'  RATEHI=',RATEHI,'  RATELO=',RATELO
            DUR_RSL=25.0E-9*NPTS
            WRITE(*,*)'RESOLUTION OF DURATION IS ',DUR_RSL
            NDR=DUR/DUR_RSL
            DUR1=DUR_RSL*(RATELO+1)*1    ! NEAREST VALUE TO DESIRED DURATION
            WRITE(*,*)'NDR= ',NDR
            WRITE(*,*)'ACTUAL DURATION OF SHAPED PULSE IS ',DUR1
            DFDUR=DUR-DUR1
            WRITE(*,*)'DEVIATION FROM DESIRED DURATION ',DFDUR,DUR_RSL
            TPOINT=25.0E-9*(RATELO+1)*RATEHI
            WRITE(*,*)'TIME PER POINT IS ',TPOINT
            NPTSDF=DFDUR/TPOINT +1
            WRITE(*,*)'CHANGE NUMBER OF POINTS BY ',NPTSDF
            NPTS=NPTS+NPTSDF
            WRITE(*,*)'ADJUSTED NUMBER OF POINTS IS ',NPTS
            RATELO=TPOINT/(25.0E-9*RATEHI)-1
            WRITE(*,*)'ADJUSTED RATELO ',RATELO
C   ******UPDATE SHAPED PULSE******
          MX=0.0
          DO  I = 1,NPTS
C           T = B*(REAL(I-1)/REAL(NPTS-1)-.5)
           T = B*(FLOAT(I-1)/FLOAT(NPTS-1)-.5)
          IF(ABS(T).LT.1E-16) T=1.0E-8

           W(I) = (CH0*COS(CH1*T)-EXP(-T*T)*CH2*SIN(CH3*T)/T)*EXP(-T**6)
C           W(I) =(CH0+CH1*T+CH2*T*T+CH3*T*T*T)*EXP(-P*T*T)
           IF (MX.LT.ABS(W(I))) MX = ABS(W(I))
C            print*, 'w = ',W(I),'  T= ',T ,I,NPTS
c           READ(*,*)
          ENDDO
C           W(1)=MX/4.0
C           W(NPTS)=MX/4.0
           print*, 'MAX OF SHAPE= ',MX
C           WRITE(*,*)'AMPLITUDE OF WAVEFOR ? (2047 is maximum)'
C           READ(*,*) AMP
            AMP=2047
          DO  I = 1,NPTS
c           WAVE(I) = 2047*(W(I)/MX)

C            W(I) = (W(I)/MX)
C            WAVE(I)=AMP*W(I)/NPTS
             WAVE(I) =AMP*(W(I)/MX)
c           print*, 'wave = ',WAVE(I),'T= ',T
c           READ(*,*)
          ENDDO

        ENDIF    !****** END OF CONTROL LOOP ***************

        WRITE(*,*)'DURATION OF ADJUSTED SHAPE ',NPTS*TPOINT
        FLIP=WS*TPOINT
        WRITE(*,*)'FLIP ANGLE OF ADJUSTED SHAPED PULSE ',FLIP

        CALL DEFWAV(WBASE,ADIR,NPTS,NBITS,WAVE(1))
        IF (STATUS .NE. 0) THEN
           WRITE(*,300) STATUS
 300       FORMAT(3X,'Status code = ',I3)
        ENDIF
c *                                                          *
c *                          Set clock                       *
c *                                                          *
c *    This sets the internal clock, the low bit divider,    *
```

```
c *     and the high bit divider.                                      *
c *                                                                    *
        CLKSRC = 0

        CALL SETCLK(CLKSRC,RATELO,RATEHI)
        IF (STATUS .NE. 0) THEN
           WRITE(*,400) STATUS
 400       FORMAT(3X,'Status code = ',I3)
        ENDIF
c *                                                                    *
c *                          Set repitions                             *
c *                                                                    *
c *     NOTE: REPT=0 sets a continuous wavefrom.                       *
c *                                                                    *

        print*,'Number of repetitons?'
        READ(5,*) REPT

        CALL SETRPT(REPT)
        IF (STATUS .NE. 0) THEN
           WRITE(*,500) STATUS
 500       FORMAT(3X,'Status code = ',I3)
        ENDIF
c *                                                                    *
c *                            Set scale                               *
c *                                                                    *
c *     This sets a full scale.                                        *
c *                                                                    *
        MULT = 127
        CALL WSCALE(MULT)
        IF (STATUS .NE. 0) THEN
           WRITE(*,600) STATUS
 600       FORMAT(3X,'Status code = ',I3)
        ENDIF
c *                                                                    *
c *                            Set delay                               *
c *                                                                    *
c *     This sets a delay of zero.                                     *
c *                                                                    *
        DLAY = 0
        CALL SETDLY(DLAY)
        IF (STATUS .NE. 0) THEN
           WRITE(*,700) STATUS
 700       FORMAT(3X,'Status code = ',I3)
        ENDIF
c *                                                                    *
c *                          Set trigger                               *
c *                                                                    *
c *     This sets an internal trigger and then returns to              *
c *     program mode.                                                  *
c *                                                                    *
c *     TSRC:  0 for internal trigger mode
c *            otherwise for external trigger mode
c *
c *     RETRIG: 0 return to 'program' mode after task
c *             otherwise remain in 'run' mode after task


        TSRC = 0
        RETRIG = 0
        print*,'External trigger mode ?'
        print*,'(Y=1,N=0, any other value end program)'
```

```
      read(5,*) xt
      IF (xt.eq.1) then
          print*,'TSRC ?'
          read(5,*) TSRC
          print*,'RETRIG ?'
          read(5,*) RETRIG
      ELSE
          goto 1000

      ENDIF

      CALL SETRIG(TSRC,TLAY,RETRIG)
      IF (STATUS .NE. 0) THEN
         WRITE(*,800) STATUS
 800     FORMAT(3X,'Status code = ',I3)
      ENDIF
c *                                                              *
c *                       Start waveform                         *
c *                                                              *
c *    This disables the auto-reverse and begins waveform.       *
c *                                                              *
      AREV = 0
      CALL WSTART(WBASE,ADIR,NPTS,AREV)
      IF (STATUS .NE. 0) THEN
         WRITE(*,900) STATUS
 900     FORMAT(3X,'Status code = ',I3)
      ENDIF
      goto 999
 1000 END**********************
```

Appendix D

Plotting Routines

```
      PROGRAM MACNMR
C     PROGRAM PG_ZOOM4
C
C THIS PROGRAM PLOTS UP TO FOUR-COLOMN INPUT DATA FILE.
C THE FIRST PLOT IS REAL COMPONENT OF FID.
C THE SECOND PLOT IS IMAGINARY COMPONENT OF FID.
C THE THIRD PLOT IS MAGNITUDE OF FID.
C LABLES FOR AXES AND TITLE ARE ALSO INPUT BY USER.
C ANY PART OF PLOT CAN BE ZOOMED AND PRINTED.
C HAVE THE OPTION TO VIEW/PRINT PLOTS INDIVIDUALLY OR AS GROUP.
C MACNMR DATA FILE HAVE THEIR COLOMNS ARRANGED DIFFERENTLY.
C THIS PROGRAM IS SUITED FOR THAT SPECIFIC ORDER. ANOTHER APPROACH
C IS TO WRITE A PROGRAM WHICH REARRANGES THE COLOMNS TO OUR USUAL
C ORDER AND THEN USE ANY OF OUR EXISTING PGPLOT PLOTTING PROGRAMS.

      CHARACTER*50 NAME, LBX, LBY, LBTITLE, ZOOM,VW
      INTEGER I, J, NUMBER, NBIN, pgbeg, NK, IL, IR, JP,LW,IP
c                                ^^^^^^--need to include this as an integer
c     INCLUDE 'AF.I'
      PARAMETER (N=10000)
      double precision X(N),V(N)
      REAL W, Z(N), T(N), Y(N), XAX(N), A, B, XTMP(N), YTMP(N),Z1(N)
      DIMENSION F1(n), f2(n), f3(n), ran(3)
      xmax=-1e+9
      xmin=-xmax
      YMAX=-200
      YMIN=200
c     NBIN=101
      s=10
      B=0.0
      NK=1000
      NBIN=NK

      WRITE(*,*)'INPUT NAME OF FILE (e.g filename.dat)'
      READ(*,'(A)') NAME

      LBX='TIME'
      WRITE(*,*)'INPUT X-AXIS TITLE'
      READ(*,'(A)') LBX

      LBY='AMPLITUDE'
      WRITE(*,*)'INPUT Y-AXIS TITLE'
      READ(*,'(A)') LBY

      LBTITLE='REAL PART'
      WRITE(*,*)'INPUT TITLE'
      READ(*,'(A)') LBTITLE

      WRITE(*,*)'WIDTH OF LINE? 1  2  3  4...'
      READ(*,*) LW

      WRITE(*,*)'ZOOM? (Y,N)'
      READ(*,'(A)') ZOOM

      IF(ZOOM.EQ.'Y') THEN
         WRITE(*,*)'ZOOM STARTING POINT'
         READ(*,*) IL
         WRITE(*,*)'ZOOM ENDING POINT'
         READ(*,*) IR
         NBIN=IR-IL
         WRITE(*,*) NBIN
      ELSE
```

```
            IL=1
            IR=NK
         END IF

         WRITE(*,*) 'VIEW ALL PLOTS TOGETHER? ( Y->YES  OTHER->NO )'
         READ(*,'(A)') VW
         IF(VW.EQ.'Y') THEN
            JP=-3
         ELSE
            JP=1
         END IF

         OPEN(UNIT=1,FILE=NAME,STATUS='OLD')
         DO 900 I=1,NK
c         READ(1,*,END=900) XAX(I),Y(I),Z(I),Z1(I)
C         READ(1,*,END=900) Y(I),Z(I),XAX(I),Z(1)
          IR=I
         READ(1,*,END=900) Y(I),Z(I),XAX(I)

c         READ(1,100,END=900) XAX(I),Y(I),Z(I),Z1(I)
C          READ(1,100,END=900) XAX(I),Y(I),Z(I)

900      CONTINUE
         CLOSE(1)
c100     format(1f5.3,1f5.3,1f7.1,1f6.2)
c100     FORMAT(4F20.8)
C100      FORMAT(4E13.6)

         DO 950 I=IL,IR
         XTMP(I)=XAX(I)
         YTMP(I)=Y(I)
950      CONTINUE

         DO 800 I=IL,IR
         IF(XMAX.LT.XTMP(I)) XMAX=XTMP(I)
         IF(YMAX.LT.YTMP(I)) YMAX=YTMP(I)
         IF(XMIN.GT.XTMP(I)) XMIN=XTMP(I)
         IF(YMIN.GT.YTMP(I)) YMIN=YTMP(I)
C         WRITE(*,*)YTMP(I),IR
800      CONTINUE
C         READ(*,*)

C   GRAPH DATA
C
         call pgslw(LW)
         IF ((PGBEG(0,'?',1,JP)).NE.1) STOP                !1
C        CALL PGPANL(1,1)
         CALL PGENV (xmin, xmax, ymin, ymax,0,1)           !2
c        CALL PGpt(Nbin,XAX,Y,-1)                          !3
c        CALL PGLAB('TIME','AMPLITUDE','REAL')   !4
         CALL PGLAB(LBX,LBY,LBTITLE)   !4
c        call pgslw(LW)
C        call pgline(nbin,xax,y)                           !5
         call pgline(nbin,XTMP,YTMP)                                !5

         xmax=-1e+9
         xmin=-xmax
         YMAX=-1E+9
         YMIN=1E+9
c  IMAGINARY COMPONENT
         DO 955 I=IL,IR
         XTMP(I)=XAX(I)
```

```
        YTMP(I)=Z(I)
955     CONTINUE
        IP=1
        DO 805 I=IL,IR
        IF(XMAX.LT.XTMP(I)) XMAX=XTMP(I)
        IF(YMAX.LT.YTMP(I)) YMAX=YTMP(I)
        IF(XMIN.GT.XTMP(I)) XMIN=XTMP(I)
        IF(YMIN.GT.YTMP(I)) YMIN=YTMP(I)
        IF(YMIN.GT.YTMP(I)) IP=I
805     CONTINUE
        YTMP(IR)=0.0
        WRITE(*,*)'XMN= ',XMAX,XMIN,'YMN= ',YMAX,YMIN,IP

C       LBTITLE='IMAGINERY PART'
C        WRITE(*,*)'INPUT TITLE (IMAGINARY)'
C       READ(*,'(A)') LBTITLE

C   GRAPH DATA
C
C       IF ((PGBEG(0,'?',1,JP)).NE.1) STOP              !1
C        YMIN=-200.0
        CALL PGENV (xmin, xmax, ymin, ymax,0,1)         !2
c       CALL PGpt(Nbin,XAX,Y,-1)                        !3
c       CALL PGLAB('TIME','AMPLITUDE','IMAGINARY')   !4
        CALL PGLAB(LBX,LBY,LBTITLE)   !4
        call pgslw(LW)
C       call pgline(nbin,xax,y)                         !5
        call pgline(nbin,XTMP,YTMP)                     !5

c  MAGNITUDE
        xmax=-1e+9
        xmin=-xmax
        YMAX=-1.0E+9
        YMIN=-YMAX

        DO 985 I=IL,IR
        XTMP(I)=XAX(I)
        YTMP(I)=Y(I)**2+Z(I)**2
985     CONTINUE

        DO 885 I=IL,IR
        IF(XMAX.LT.XTMP(I)) XMAX=XTMP(I)
        IF(YMAX.LT.YTMP(I)) YMAX=YTMP(I)
        IF(XMIN.GT.XTMP(I)) XMIN=XTMP(I)
        IF(YMIN.GT.YTMP(I)) YMIN=YTMP(I)
885     CONTINUE

C       LBTITLE='MAGNITUDE'
C        WRITE(*,*)'INPUT TITLE (MAGNITUDE)'
C       READ(*,'(A)') LBTITLE

C   GRAPH DATA
C
C       IF ((PGBEG(0,'?',1,JP)).NE.1) STOP              !1
        CALL PGENV (xmin, xmax, ymin, ymax,0,1)         !2
c       CALL PGpt(Nbin,XAX,Y,-1)                        !3
c       CALL PGLAB('TIME','AMPLITUDE','IMAGINARY')   !4
        CALL PGLAB(LBX,LBY,LBTITLE)   !4
        call pgslw(LW)
C       call pgline(nbin,xax,y)                         !5
        call pgline(nbin,XTMP,YTMP)                                 !5
```

```
        call pgend                                              !6
c-------the above 6 lines are part of the plplot routines. If you just wanted
c*******a line then you would only require lines:1,2,5 and 6. Keramat has a
c*******book which has what different subroutines do.
        END
```

```
      PROGRAM PGS2ZNR
C     THIS PROGRAM STACKS PLOTS.
C     THE USER DECIDES HOW MANY.
C     WIDTH (X) OF PHYSICAL PAGE IS 8.8 .
C     HEIGHT (Y) OF PHYSICAL PAGE IS 11 .
C     EACH PLOT CAN BE ZOOMED HORIZONTALLY.
C     BY ENTERING AN UPPER BOUND FOR THE
C     FREQUENCY WHICH SETS THE LOWER BOUND
C     (SYMMETRIC PLOTS SUCH AS SPECTRAS).
C     THE UNITS OF THE X DATA CAN BE CHANGED
C     I.E X DATA CAN BE DIVIDED BY A FACTOR.
C     SO FOR EXAMPLE FRRQUENCY CAN BE SCALED
C     TO KHZ, OR TIME CAN BE IN MICROSECONDS.
C     THIS PROGRAM USES A "WHILE" LOOP !!!
      PARAMETER (N=20000)
      REAL XL,XR,YB,XAX(N),XTMP(N),DY
      REAL XTK,YTK,Y(N),YTMP(N),Z(N)
      REAL XMIN,YMIN,YMAX,R,DX,DLX,XMAX
      INTEGER NX,NY,NBIN,LW,NK
      INTEGER PGBEG,JP,JZ,L
      CHARACTER*26 NAME,XOPT,YOPT
      CHARACTER*26 LBX,LBY,LBT,SZ

      COMMON/S1/LBX,LBY,LBT,XOPT,YOPT   !STRING VARIABLES
      COMMON/S2/JP,LW,NX,NY             !INTEGER VARIABLES
      COMMON/S3/XL,XR,YB,YT             !REAL VARIABLES
      COMMON/S4/PBEG

      L=0
111   WRITE(*,*)'HOW MANY PLOTS?'
      READ(5,*) L
      IF(L.EQ.0) GOTO 111

      R=1.0E+9
      XL=0.0
      XR=4.0
      YFST=1.2
      YB=0.0+YFST
C     YT=2.0+YFST
      YT=11/L+YFST
C     DY=yt-yb
C     DY=(11-3*YFST)/L
      DY=(11-2.0)/L
      YT=DY+YFST
      DX=XR/4.0
      XTK=0.0
      YTK=0.0
      NX=XR-XL+1
      NY=NX
C     LW=1
      WRITE(*,*)'LINE THICKNESS? (INCREASES WITH NO. 1 2 3 4)'
      READ(5,*)LW
      XL=XL+DX*0.0
      XR=XR+DX*2.0
      DLX=0.9
      XL=XL+DLX
      XR=XR+DLX
      CALL BEGIN_GRAPHICS
      RXSCL=1.0
C     **********************
      DO IX=1,L
      XMIN=R
```

```
        YMIN=R
        XMAX=-R
        YMAX=-R

c       NAME='AMPLITUDE.DAT'
222     WRITE(*,*)'NAME OF FILE'
        READ(5,'(A)') NAME
        IF(NAME.EQ.' ') GOTO 222
        WRITE(*,*)'RESCALE X-AXIS? 1->YES OTHER NO'
        READ(5,*) IXANSW
        IF(IXANSW.EQ.1) THEN
           WRITE(*,*)'ENTER SCALING FACTOR?'
           READ(5,*) RXSCL
        ENDIF
        OPEN(UNIT=1,FILE=NAME,STATUS='OLD')
        DO 900 I=1,N
C         READ(1,100,END=900) XAX(I),Y(I)
c         READ(1,*,END=901) XAX(I),Y(I)
c         READ(1,*,END=901) Y(I),Z(I),XAX(I)
         READ(1,*,END=901) XAX(I),Y(I)
         XAX(I)=XAX(I)/RXSCL
         JZ=I
C        WRITE(*,*) XAX(I),Y(I),Z(I),I
C        WRITE(*,100) XAX(I),Y(I),Z(I)
900     CONTINUE
901        CLOSE(1)
C100    FORMAT(3F13.6)
        write(*,*) JZ
        NK=JZ
        nbin= NK
        ICX=NK
        YMAX=-10.0
        YMIN=10.0
        DO I=1,NK
        IF(XMAX.LT.XAX(I)) XMAX=XAX(I)
        IF(YMAX.LT.Y(I)) YMAX=Y(I)
        IF(XMIN.GT.XAX(I)) XMIN=XAX(I)
        IF(YMIN.GT.Y(I)) YMIN=Y(I)
C        WRITE(*,*)YTMP(I),IR
        ENDDO
C        READ(*,*)
        WRITE(*,*)'YMAX=',YMAX,YMIN
        WRITE(*,*)'XMAN=',XMAX,XMIN
C************************
        WRITE(*,*)'ZOOM? (Y,N)'
        READ(*,'(A)') ZOOM
        ICX=JZ
        IF(ZOOM.EQ.'Y') THEN
           WRITE(*,*)'FREQUENCY ENDPOINT (XMAX)'
           READ(5,*) XMAX
           XMIN=-XMAX
           I=1
           DO WHILE(XAX(I).GE.XMAX)
              I=I+1
           ENDDO
           NBIN=NK-2*I
           WRITE(*,*)'NUMBER OF POINTS ',NBIN
           IL=ICX/2-NBIN/2
           IR=ICX/2+NBIN/2
           WRITE(*,*) NBIN
        ELSE
           IL=1
```

```
            IR=ICX
            NBIN=ICX
            WRITE(*,*)'NBIN=',NBIN
         END IF
         DO 950 I=IL,IR
         II=I-IL+1
         XTMP(II)=XAX(I)
         YTMP(II)=Y(I)
950      CONTINUE

C        DO 800 I=1,IR-IL+1
C        IF(XMAX.LT.XTMP(I)) XMAX=XTMP(I)
C         IF(YMAX.LT.YTMP(I)) YMAX=YTMP(I)
C         IF(XMIN.GT.XTMP(I)) XMIN=XTMP(I)
C        IF(YMIN.GT.YTMP(I)) YMIN=YTMP(I)
C         WRITE(*,*)YTMP(I),IR
C800     CONTINUE
C         READ(*,*)

C*************************
C         XMAX=INT(XMAX/10+1)*10
         WRITE(*,*)'YMAX=',YMAX,YMIN
         WRITE(*,*)'XMAN=',XMAX,XMIN

         WRITE(*,*)'NEW END POINTS FOR X-AXIS? 1->YES OTHER NO'
         READ(5,*) IANSW
         IF(IANSW.EQ.1) THEN
            WRITE(*,*)'RIGHT X-ENDPOINT?'
C           XSCALE=1.0
            READ(5,*) XMAX
C           XMAX=XMIN+XSCALE
            WRITE(*,*)'LEFT  X-ENDPOINT?'
            READ(5,*) XMIN
         ENDIF
         WRITE(*,*)'NEW END POINTS FOR Y-AXIS? 1->YES OTHER NO'
         READ(5,*) IANSW
         IF(IANSW.EQ.1) THEN
C           YSCALE=1.0
C           WRITE(*,*)'Y-SCALE?'
C           READ(5,*) YSCALE
            WRITE(*,*)'TOP    Y-ENDPOINT?'
            READ(5,*) YMAX
            WRITE(*,*)'BOTTOM Y-ENDPOINT?'
            READ(5,*) YMIN
C           YMAX=YMIN+YSCALE
         ENDIF

         LBX=' '
         LBY=' '
         LBT=' '
         YOPT=' '
         XOPT=' '
         IF(IX.EQ.1) THEN
            WRITE(*,*)'LABEL FOR X-AXIS'
            READ(5,'(A)') LBX
C           LBX='FREQUENCY'
            XOPT='BINT1'
         ELSE
            IF(IX.EQ.(L/2+1)) THEN
               WRITE(*,*)'LABEL FOR Y-AXIS'
               READ(5,'(A)') LBY
C              LBY='INTENSITY'
```

```
              YOPT='BNVT1'
C              YOPT='BNT1'

            ELSE
               IF(IX.EQ.L) THEN
                  WRITE(*,*)'TITLE'
                  READ(5,'(A)') LBT
               ENDIF
            ENDIF
         ENDIF
         JP=IX
         yopt='BNIVT1'
         CALL DOPLOT(XMIN,XMAX,YMIN,YMAX,NBIN,XTMP,YTMP)
         YB=YB+DY
         YT=YT+DY
         ENDDO
c     *********************
         CALL END_GRAPHICS

         END
c*****************************************
         SUBROUTINE BEGIN_GRAPHICS
         INTEGER PGBEG
         COMMON/S4/PBEG
         IF ((PGBEG(0,'?',1,JP)).NE.1) STOP              !1
         RETURN
         END

         SUBROUTINE DOPLOT(XMIN,XMAX,YMIN,YMAX,NBIN,XTMP,YTMP)
         REAL XTMP(*),YTMP(*),XMIN,XMAX,YMIN,YMAX
         REAL XL,XR,YB,YT,XTK,YTK,CHT
         INTEGER NBIN,JP,LW,NX,NY
         CHARACTER*26 LBX,LBY,LBT,YOPT,XOPT,LBNM
         COMMON/S1/LBX,LBY,LBT,XOPT,YOPT   !STRING VARIABLES
         COMMON/S2/JP,LW,NX,NY             !INTEGER VARIABLES
         COMMON/S3/XL,XR,YB,YT             !REAL VARIABLES
         COMMON/S4/PBEG

         XTK=0.0
         YTK=0.0
c        JP=1
         LBNM=' '
         IF(JP.EQ.1) LBNM='A'
         IF(JP.EQ.2) LBNM='B'
         IF(JP.EQ.3) LBNM='C'
         IF(JP.EQ.4) LBNM='D'
         IF(JP.EQ.5) LBNM='E'

c        IF ((PGBEG(0,'?',1,JP)).NE.1) STOP              !1
         CALL PGSCF(2)
         CHT=1.2
         CALL PGSCH(CHT)
         CALL PGSLW(LW)
         CALL PGVSIZ(XL,XR,YB,YT-.3)
C         CALL PGSVP(XL,XR,YB,YT)      !SET VIEWPORT IN NORMALIZED COORDINATES.
C         CALL PGWNAD(XL,XR,YB,YT)       !ADJUST WINDOW IN WORLD COORDINATES.
         CALL PGSWIN(XMIN,XMAX,YMIN,YMAX)!ADJUST WINDOW IN WORLD COORDINATES
                                         !TO BE MAPPED ON TO VIEWPORT.
         CALL PGBOX(XOPT,XTK,NX,YOPT,YTK,NY) !DRAW LABELED FRAME AROUND VIEWPORT
         CALL PGSLW(LW)
         CALL PGLINE(NBIN,XTMP,YTMP)
C        CALL PGSCF(2)
```

```
C        CALL PGSCH(1.2)
c        CALL PGLAB(LBX,LBY,LBT)
c        CALL PGMTXT('B',-2.0,.0,0.5,LBX)
         CALL PGPTXT(xmin*1.2,ymax*.1,90.0,0.0,LBY)
              DXX=(XMAX-XMIN)/2.0
              XADJ=xmin+DXX
              DYY=(YMAX-YMIN)*.1
              YADJ= YMIN-DYY*2


         CALL PGMTXT('B',3*CHT,0.5,0.5,LBX)

C        CALL PGPTXT(XADJ,YADJ,0.0,0.5,LBX)
              YADJ=YMAX+DYY
         CALL PGMTXT('t',1*CHT,0.5,0.5,LBT)
c        CALL PGPTXT(XADJ,YADJ,0.0,0.5,LBT)
         CALL PGSCH(2.0)
         CALL PGPTXT(xmax*.95,ymax*.25,0.0,0.0,LBNM)

         RETURN
         END
c*************************************
         SUBROUTINE END_GRAPHICS
         CALL PGEND
         RETURN
         END
```

Bibliography

[1] T.M. Barbara. *J. Magn. Reson.*, **67**:491–500, 1986.

[2] C. Counsell, M.H. Levitt, and R.R. Ernst. *J.Magn.Res.*, **63**:133, (1985).

[3] Altmann S. *Rotations, Quaternions and Double Groups.* Clarendon Press, Oxford, 1986.

[4] D. J. Siminovitch and S. Habot. *J. Phys. A: Math. Gen.*, **30**:2577, (1997).

[5] J. Seelig. *Quaterly Review of Biophysics*, **10** (3):353, (1977).

[6] M. Bloom, J.H. Davis, and M.I. Valic. *Can.J.Phys.*, **58**:1510, (1980).

[7] D.J. Siminovitch, D.P. Raleigh E.T. Olejniczak, and R.G. Griffin. *J.Chem.Phys*, **84**(5):2556, (1986).

[8] M.H. Levitt, D. Suter, and R.R. Ernst. *J.Chem.Phys.*, **80** (7):3064, (1984).

[9] J. B. Murdoch. *Ph.D. Thesis.* University of California, Berkeley, 1981.

[10] D. Lu S. Ali and D.J. Siminovitch. *J.Magn.Res. A*, 122:192, 1996.

[11] W.S. Warren and M.A. McCoy. *J.Magn.Res.*, **98**:24, (1992).

[12] T. Fujiwara and K. Nagayama. *J.Magn.Res.*, **93**:563, (1991).

[13] D.J. Siminovitch and N.J. Tagg. *J.Magn.Res.*, **A 108**:82, (1994).

[14] M.A. McCoy. *Ph.D. Thesis.* Princeton University, 1988.

[15] S. Habot, D. Lu, N. J. Tagg, G. R. Gall, and D. J. Siminovitch. *J. Solid State NMR*, **10**:137, (1998).

[16] S. Vega and A. Pines. *J.Chem.Phys.*, **no.12 66**:5624, (1977).

[17] R.L. Vold A.D. Ronemus and R.R. Vold. *J.Magn.Res*, 70:416, 1986.

[18] M. D. Shuster. *J. Astronaut.*, **41**:439–517, 1993.

[19] J. Stuelpnagel. *SIAM Rev.*, **6**:422–30, 1964.

[20] M. A. M. Santiago and A. N. Vaidya. *J. Phys. A: Math. Gen.*, **9**:897–904, 1976.

[21] D. J. Siminovitch. *J. Magn. Reson. A*, **117**:235–45, 1995.

[22] H. Gao J. Zhou and B. C. Sanctuary. *J. Magn. Reson. A*, **101**:119–21, 1993.

[23] C. Ye J. Zhou and B. C. Sanctuary. *J. Chem. Phys.*, **101**:6424–9, 1994.

[24] O. Rodrigues. *J. de Mathématiques Pures et Appliquées*, **5**:380–440, 1840.

[25] L. D. Favro. *Phys. Rev.*, **119** :53–62, 1960.

[26] T. M. Barbara. *J. Magn. Reson. A*, **6**:188–94, 1994.

[27] J. Sivardière. *Am. J. Phys.*, **62**:737–43, 1994.

[28] Coxeter. *A.Math.Monthly*, **53**:136–46, 1946.

[29] C. W. Misner, K. S. Thorne, and J. A. Wheeler. *Gravitation.* W.H.Freeman, San Francisco, 1973.

[30] A. Cayley. *Math. Ann.*, **15**:238–40, 1879.

[31] L. C. Biedenharn and J. D. Louck. *Angular Momentum in Quantum Physics. Theory and Application.* Addison-Wesley, Reading, 1981.

[32] F. Klein. The Icosahedron, 1884.

[33] W. G. Harter and N. dos Santos. *Am. J. Phys.*, **46** :251–63, 1978.

[34] D. R. Truax. *Phys. Rev. D*, **31**:1988–91, 1985.

[35] C. M. Cheng and P.C.W. Fung. *J. Phys. A: Math. Gen.*, **21**:4115–31, 1988.

[36] Ban M. *J. Opt. Soc. Am. B*, **10**:1347–59, 1993.

[37] R. M. Wilcox. *J. Math. Phys.*, **8**:962–82, 1967.

```
c ************************************************************
c *                  MCYDUR5a.FOR  (part of my thesis        *
c *                  SHAPED PULSE WAVE                       *
c *                                                          *
c *     This Quatech Program controls the Quatech
c *        Wave generator as in a QUASH experiment. It
c *        contains the QUASH pulse in an array.
c *
c *
c *
c *      This example programs the WSB-100 to produce a      *
c *   Shaped Pulse. Each call is followed by a status check  *
c *   to test for errors encountered during the routine.     *
c *   Program execution is as follows:                       *
c *                                                          *
c *      1. Call INIT to initialize parameters               *
c *         a) Board being initialized = 1                   *
c *         b) Base address = 300H                           *
c *         c) Assumed type of module being used is the      *
c *            WSB-A12M analog module                        *
c *         d) DMA channel not utilized                      *
c *         e) Interrupt level not utilized                  *
c *                                                          *
c *      2. Call WSTOP to perform a synchronous stop         *
c *         a) Do not enable hold mode                       *
c *         b) Wait for WSB-100 to enter program mode        *
c *                                                          *
c *      3. Call DEFWAV to define waveform                   *
c *         a) Address in WSB-100 memory = 0H                *
c *         b) Data is stored in sequentially higher         *
c *            address locations                             *
c *         c) Number of points to be stored = 360           *
c *         d) Both positive and negative data provided      *
c *         e) Waveform array = WAVE(NPTS)                   *
c *                                                          *
c *      4. Call SETCLK to specify source and frequency      *
c *         of the data output clock                         *
c *         a) Internal clock source selected (CLKSRC=0,1)   *
c *         b) Low bit used for frequency divider = 3        *
c *         c) High bit used for frequency divider = 100     *
c *         d) The data is clocked out at a frequency        *
c *            given by f=40*10^6/[(RATELO+1)*(RATEHI)]      *
c *                                                          *
c *      5. Call SETRPT to specify number of repitions       *
c *         a) Waveform set to repeat as indicated by user   *
c *                                                          *
c *      6. Call WSCALE to set scale                         *
c *         a) Full scale setting selected                   *
c *                                                          *
c *      7. Call SETDLY to set delay                         *
c *         a) Disable delay between cycles                  *
c *                                                          *
c *      8. Call SETRIG to set trigger                       *
c *         a) Internal trigger source selected              *
c *         b) Returns to program mode after task            *
c *                                                          *
c *      9. Call WSTART to begin waveform                    *
c *         a) Auto-reverse disabled                         *
c *                                                          *
c ************************************************************

       INTEGER*2 ACTBRD, BASEADDR, MODULE, DMACH, IRQ, STATUS
```

```
      INTEGER*2 WHOLD, WWAIT, WBASE, ADIR, NPTS, NBITS
      INTEGER*2 CLKSRC, RATELO, RATEHI, REPT, MULT
      INTEGER*2 DLAY, TSRC, TLAY, RETRIG, AREV, I
      INTEGER*2 WAVE(400),xt,FREQ,NDR,NPTSDF
      REAL*8 P,T,W(400),MX,CH0,CH1,CH2,CH3,B,PI,WS,AMP,RF,DUR
      REAL*8 DUR_RSL,DUR1,DFDUR,TPOINT
      CHARACTER ANS
C *                                                           *
C *                    Initialize WSB-100                     *
C *                                                           *
C *    This sets the active board, the base address, the      *
C *    module used, the DMA channel, the interrupt level,     *
C *    and the operating status.                              *
C *                                                           *
      ACTBRD = 1
      BASEADDR = 768
      MODULE = 2
      DMACH = 0
      IRQ = 0
      STATUS = 65535
      CALL INIT(ACTBRD,BASEADDR,MODULE,DMACH,IRQ,STATUS)
      IF (STATUS .NE. 0) THEN
         WRITE(*,100) STATUS
 100     FORMAT(3X,'Status code = ',I3)
      ENDIF
C *                                                           *
C *                  Set synchronous stop                     *
C *                                                           *
C *    This disables the hold mode and waits for the          *
C *    program mode.                                          *
C *                                                           *
      WHOLD = 0
      WWAIT = 1
      CALL WSTOP(WHOLD,WWAIT)
      IF (STATUS .NE. 0) THEN
         WRITE(*,200) STATUS
 200     FORMAT(3X,'Status code = ',I3)
      ENDIF
C *                                                           *
C *                     Define waveform                       *
C *                                                           *
C *    This sets the memory address, the memory direction,    *
C *    the number of points, the number of bits, and the      *
C *    sine wave.                                             *
C *                                                           *
      WBASE = 0
      ADIR = 1
C      NPTS = 360

C                   CONTROL LOOP OF QUATECH
C                   =======================
 999     NPTS=100
C       WRITE(*,*) 'CHANGE NUMBER OF POINTS OF SHPED PULSE? CURRENTLY 100 '
      WRITE(*,*) 'NEW NUMBER OF POINTS? Y->YES'
      READ(*,'(A)') ANS
      IF(ANS.EQ.'Y') THEN
      WRITE(*,*)'NEW NUMBER OF POINTS ?'
      READ(*,*) NPTS
      ENDIF
      NBITS = 12
      P = 1.0
      MX = 0.0
```

```
        CH0=1.0
        CH1=-7.05
        CH2=-.225
        CH3=11.0
C        B=SQRT(40.0)
        B=6.45
       DO 21 I = 1,NPTS
          T = B*(REAL(I-1)/REAL(NPTS-1)-.5)
          W(I) =(CH0+CH1*T+CH2*T*T+CH3*T*T*T)*EXP(-P*T*T)
          IF (MX.LT.ABS(W(I))) MX = ABS(W(I))
c          print*, 'w = ',W(I),'  T= ',T
c          READ(*,*)
 21    CONTINUE
          W(1)=MX/4.0
          W(NPTS)=MX/4.0
          print*, 'MAX OF SHAPE= ',MX
          WRITE(*,*)'AMPLITUDE OF WAVEFOR ? (2047 is maximum)'
          READ(*,*) AMP
       DO 210 I = 1,NPTS
c          WAVE(I) = 2047*(W(I)/MX)

C           W(I) = (W(I)/MX)
C           WAVE(I)=AMP*W(I)/NPTS
            WAVE(I) =AMP*(W(I)/MX)
c          print*, 'wave = ',WAVE(I),'T= ',T
c          READ(*,*)
 210   CONTINUE
C NORMALIZE SHAPED PULSE
       DO I=1,NPTS
       W(I)=W(I)/MX
       ENDDO

       PI=4.0*ATAN(1.0)
       WS=0.0
       DO 220 I=1,NPTS
       WS=WS+W(I)
 220   CONTINUE

       WRITE(*,*)'(AREA OF SHAPE)/DT)',WS

       WRITE(*,*)'WHAT RF (IN KHZ)'
       READ(5,*) RF
       RF=RF*1000.0
C ADJUST OUTPUT RATE OF QUATECH TO MATCH TIME-STEP, DLT, OF SAHPED PULSE
       DLT=(PI/2.0)/(ABS(WS)*RF*2.0*PI)
       WRITE(*,*)'CALCULATED TIME STEP: ',DLT
       WRITE(*,*)'CALCULATED DURATION: ' ,NPTS*DLT
       WRITE(*,*)'************************'
       DUR=NPTS*DLT
       RATEHI=1 ! THIS PARAMETER DOES NOT MAKE A DIFFERENCE
       RATELO=DLT/((25.0E-9)*RATEHI)-1.0
       WRITE(*,*) 'DLT=',DLT,'  RATEHI=',RATEHI,'  RATELO=',RATELO
           DUR_RSL=25.0E-9*NPTS
           WRITE(*,*)'RESOLUTION OF DURATION IS ',DUR_RSL
           NDR=DUR/DUR_RSL
           DUR1=DUR_RSL*(RATELO+1)*1    ! NEAREST VALUE TO DESIRED DURATION
           WRITE(*,*)'NDR= ',NDR
           WRITE(*,*)'ACTUAL DURATION OF SHAPED PULSE IS ',DUR1
           DFDUR=DUR-DUR1
           WRITE(*,*)'DEVIATION FROM DESIRED DURATION ',DFDUR,DUR_RSL
           TPOINT=25.0E-9*(RATELO+1)*RATEHI
```

```
            WRITE(*,*)'TIME PER POINT IS ',TPOINT,' DLT=',DLT
            NPTSDF=DFDUR/TPOINT
            WRITE(*,*)'CHANGE NUMBER OF POINTS BY ',NPTSDF
            NPTS=NPTS+NPTSDF
            WRITE(*,*)'ADJUSTED NUMBER OF POINTS IS ',NPTS
C            DUR=TPOINT*NPTS
C  ******UPDATE SHAPED PULSE******
          MX=0.0
          DO I = 1,NPTS
           T = B*(REAL(I-1)/REAL(NPTS-1)-.5)
           W(I) =(CH0+CH1*T+CH2*T*T+CH3*T*T*T)*EXP(-P*T*T)
           IF (MX.LT.ABS(W(I))) MX = ABS(W(I))
c           print*, 'w = ',W(I),'  T= ',T
c           READ(*,*)
          ENDDO
           W(1)=MX/4.0
           W(NPTS)=MX/4.0
           print*, 'MAX OF SHAPE= ',MX
           WRITE(*,*)'AMPLITUDE OF WAVEFOR ? (2047 is maximum)'
           READ(*,*) AMP
          DO I = 1,NPTS
c           WAVE(I) = 2047*(W(I)/MX)

C            W(I) = (W(I)/MX)
C            WAVE(I)=AMP*W(I)/NPTS
             WAVE(I) =AMP*(W(I)/MX)
c           print*, 'wave = ',WAVE(I),'T= ',T
c           READ(*,*)
          ENDDO

        print*, 'Change duration of shaped pulse?'
        print*, '(Yes 1    No 0)'
        read(5,*) xt
        IF (xt.eq.1) THEN    !*****************
            WRITE(*,*)'WHAT DURATION? (IN Microseconds)'
            READ(5,*) DUR
            DUR=DUR*1.0E-6

            RATEHI=1 ! THIS PARAMETER DOES NOT MAKE A DIFFERENCE
            DLT=DUR/NPTS
            RATELO=DLT/((25.0E-9)*RATEHI)-1.0
            WRITE(*,*) 'DLT=',DLT,'  RATEHI=',RATEHI,'  RATELO=',RATELO
            DUR_RSL=25.0E-9*NPTS
            WRITE(*,*)'RESOLUTION OF DURATION IS ',DUR_RSL
            NDR=DUR/DUR_RSL
            DUR1=DUR_RSL*(RATELO+1)*1    ! NEAREST VALUE TO DESIRED DURATION
            WRITE(*,*)'NDR= ',NDR
            WRITE(*,*)'ACTUAL DURATION OF SHAPED PULSE IS ',DUR1
            DFDUR=DUR-DUR1
            WRITE(*,*)'DEVIATION FROM DESIRED DURATION ',DFDUR,DUR_RSL
            TPOINT=25.0E-9*(RATELO+1)*RATEHI
            WRITE(*,*)'TIME PER POINT IS ',TPOINT
            NPTSDF=DFDUR/TPOINT +1
            WRITE(*,*)'CHANGE NUMBER OF POINTS BY ',NPTSDF
            NPTS=NPTS+NPTSDF
            WRITE(*,*)'ADJUSTED NUMBER OF POINTS IS ',NPTS
            RATELO=TPOINT/(25.0E-9*RATEHI)-1
            WRITE(*,*)'ADJUSTED RATELO ',RATELO
C  ******UPDATE SHAPED PULSE******
          MX=0.0
          DO  I = 1,NPTS
           T = B*(REAL(I-1)/REAL(NPTS-1)-.5)
```

```
          W(I) =(CH0+CH1*T+CH2*T*T+CH3*T*T*T)*EXP(-P*T*T)
          IF (MX.LT.ABS(W(I))) MX = ABS(W(I))
c          print*, 'w = ',W(I),'  T= ',T
c          READ(*,*)
         ENDDO
          W(1)=MX/4.0
          W(NPTS)=MX/4.0
          print*, 'MAX OF SHAPE= ',MX
          WRITE(*,*)'AMPLITUDE OF WAVEFOR ? (2047 is maximum)'
          READ(*,*) AMP
         DO  I = 1,NPTS
c          WAVE(I) = 2047*(W(I)/MX)

C           W(I) = (W(I)/MX)
C           WAVE(I)=AMP*W(I)/NPTS
            WAVE(I) =AMP*(W(I)/MX)
c          print*, 'wave = ',WAVE(I),'T= ',T
c          READ(*,*)
         ENDDO

       ENDIF    !*****END OF CONTROL LOOP***************

       CALL DEFWAV(WBASE,ADIR,NPTS,NBITS,WAVE(1))
       IF (STATUS .NE. 0) THEN
          WRITE(*,300) STATUS
 300      FORMAT(3X,'Status code = ',I3)
       ENDIF
c *                                                              *
c *                         Set clock                            *
c *                                                              *
c *    This sets the internal clock, the low bit divider,        *
c *    and the high bit divider.                                 *
c *                                                              *
        CLKSRC = 0

       CALL SETCLK(CLKSRC,RATELO,RATEHI)
       IF (STATUS .NE. 0) THEN
          WRITE(*,400) STATUS
 400      FORMAT(3X,'Status code = ',I3)
       ENDIF
c *                                                              *
c *                       Set repitions                          *
c *                                                              *
c *    NOTE: REPT=0 sets a continuous wavefrom.                  *
c *                                                              *

       print*,'Number of repetitons?'
       READ(5,*) REPT

       CALL SETRPT(REPT)
       IF (STATUS .NE. 0) THEN
          WRITE(*,500) STATUS
 500      FORMAT(3X,'Status code = ',I3)
       ENDIF
c *                                                              *
c *                         Set scale                            *
c *                                                              *
c *    This sets a full scale.                                   *
c *                                                              *
       MULT = 127
       CALL WSCALE(MULT)
       IF (STATUS .NE. 0) THEN
```

```
         WRITE(*,600) STATUS
 600     FORMAT(3X,'Status code = ',I3)
      ENDIF
c *                                                              *
c *                          Set delay                           *
c *                                                              *
c *    This sets a delay of zero.                                *
c *                                                              *
      DLAY = 0
      CALL SETDLY(DLAY)
      IF (STATUS .NE. 0) THEN
         WRITE(*,700) STATUS
 700     FORMAT(3X,'Status code = ',I3)
      ENDIF
c *                                                              *
c *                        Set trigger                           *
c *                                                              *
c *    This sets an internal trigger and then returns to         *
c *    program mode.                                             *
c *                                                              *
c *    TSRC:  0 for internal trigger mode
c *           otherwise for external trigger mode
c *
c *    RETRIG: 0 return to 'program' mode after task
c *             otherwise remain in 'run' mode after task


      TSRC = 0
      RETRIG = 0
      print*,'External trigger mode ?'
      print*,'(Y=1,N=0, any other value end program)'
      read(5,*) xt
      IF (xt.eq.1) then
          print*,'TSRC ?'
          read(5,*) TSRC
          print*,'RETRIG ?'
          read(5,*) RETRIG
      ELSE
          goto 1000

      ENDIF

      CALL SETRIG(TSRC,TLAY,RETRIG)
      IF (STATUS .NE. 0) THEN
         WRITE(*,800) STATUS
 800     FORMAT(3X,'Status code = ',I3)
      ENDIF
c *                                                              *
c *                      Start waveform                          *
c *                                                              *
c *    This disables the auto-reverse and begins waveform.       *
c *                                                              *
      AREV = 0
      CALL WSTART(WBASE,ADIR,NPTS,AREV)
      IF (STATUS .NE. 0) THEN
         WRITE(*,900) STATUS
 900     FORMAT(3X,'Status code = ',I3)
      ENDIF
      goto 999
 1000 END••••••••••••••••••••••
```

```
c ************************************************************
c *                SYMTRIC1.FOR  (SHP2OP1.FOR)               *
c *                SYMTRIC1.FOR  is part of my thesis        *
c *                  SHAPED PULSE WAVE                       *
c *                                                          *
c *     This program controls the QUATECH Wave Generator
c *         In a SQUASH experiment. It contains the SUQUASH
c *         pulse in an array.
C *
c *      This example programs the WSB-100 to produce a      *
c *  Shaped Pulse. Each call is followed by a status check   *
c *  to test for errors encountered during the routine.      *
c *  Program execution is as follows:                        *
c *                                                          *
c *      1. Call INIT to initialize parameters               *
c *         a) Board being initialized = 1                   *
c *         b) Base address = 300H                           *
c *         c) Assumed type of module being used is the      *
c *            WSB-A12M analog module                        *
c *         d) DMA channel not utilized                      *
c *         e) Interrupt level not utilized                  *
c *                                                          *
c *      2. Call WSTOP to perform a synchronous stop         *
c *         a) Do not enable hold mode                       *
c *         b) Wait for WSB-100 to enter program mode        *
c *                                                          *
c *      3. Call DEFWAV to define waveform                   *
c *         a) Address in WSB-100 memory = 0H                *
c *         b) Data is stored in sequentially higher         *
c *            address locations                             *
c *         c) Number of points to be stored = 360           *
c *         d) Both positive and negative data provided      *
c *         e) Waveform array = WAVE(NPTS)                   *
c *                                                          *
c *      4. Call SETCLK to specify source and frequency      *
c *         of the data output clock                         *
c *         a) Internal clock source selected (CLKSRC=0,1)   *
c *         b) Low bit used for frequency divider = 3        *
c *         c) High bit used for frequency divider = 100     *
c *         d) The data is clocked out at a frequency        *
c *            given by f=40*10^6/[(RATELO+1)*(RATEHI)]      *
c *                                                          *
c *      5. Call SETRPT to specify number of repitions       *
c *         a) Waveform set to repeat as indicated by user   *
c *                                                          *
c *      6. Call WSCALE to set scale                         *
c *         a) Full scale setting selected                   *
c *                                                          *
c *      7. Call SETDLY to set delay                         *
c *         a) Disable delay between cycles                  *
c *                                                          *
c *      8. Call SETRIG to set trigger                       *
c *         a) Internal trigger source selected              *
c *         b) Returns to program mode after task            *
c *                                                          *
c *      9. Call WSTART to begin waveform                    *
c *         a) Auto-reverse disabled                         *
c *                                                          *
c ************************************************************

       INTEGER*2 ACTBRD, BASEADDR, MODULE, DMACH, IRQ, STATUS
       INTEGER*2 WHOLD, WWAIT, WBASE, ADIR, NPTS, NBITS
```

```
      INTEGER*2 CLKSRC, RATELO, RATEHI, REPT, MULT
      INTEGER*2 DLAY, TSRC, TLAY, RETRIG, AREV, I
      INTEGER*2 WAVE(400),xt,FREQ,NDR,NPTSDF,J,NPTS1
      REAL*8 P,T,W(400),MX,CH0,CH1,CH2,CH3,B,PI,WS,AMP,RF,DUR
      REAL*8 DUR_RSL,DUR1,DFDUR,TPOINT,FLIP
      CHARACTER ANS
c *                                                              *
c *                    Initialize WSB-100                        *
c *                                                              *
c *    This sets the active board, the base address, the         *
c *    module used, the DMA channel, the interrupt level,        *
c *    and the operating status.                                 *
c *                                                              *
      ACTBRD = 1
      BASEADDR = 768
      MODULE = 2
      DMACH = 0
      IRQ = 0
      STATUS = 65535
      CALL INIT(ACTBRD,BASEADDR,MODULE,DMACH,IRQ,STATUS)
      IF (STATUS .NE. 0) THEN
         WRITE(*,100) STATUS
 100     FORMAT(3X,'Status code = ',I3)
      ENDIF
c *                                                              *
c *                    Set synchronous stop                      *
c *                                                              *
c *    This disables the hold mode and waits for the             *
c *    program mode.                                             *
c *                                                              *
      WHOLD = 0
      WWAIT = 1
      CALL WSTOP(WHOLD,WWAIT)
      IF (STATUS .NE. 0) THEN
         WRITE(*,200) STATUS
 200     FORMAT(3X,'Status code = ',I3)
      ENDIF
c *                                                              *
c *                       Define waveform                        *
c *                                                              *
c *    This sets the memory address, the memory direction,       *
c *    the number of points, the number of bits, and the         *
c *    sine wave.  QUATECH PREPARATION FOR SQUASH STRAT HERE
                        *
c *                                                              *
      WBASE = 0
      ADIR = 1
C      NPTS = 360

c                       CONTROL LOOP OF QUATECH
c                       =======================
 999     NPTS=100
C      WRITE(*,*) 'CHANGE NUMBER OF POINTS OF SHPED PULSE? CURRENTLY 100 '
      WRITE(*,*) 'NEW NUMBER OF POINTS? Y->YES'
      READ(*,'(A)') ANS
      IF(ANS.EQ.'Y') THEN
      WRITE(*,*)'NEW NUMBER OF POINTS ?'
      READ(*,*) NPTS1
      ENDIF
      NPTS=NPTS1
      NBITS = 12
      P = 1.0
```

```
       MX = 0.0

        CH0=1.2
        CH1=8.7
        CH2=0.1
        CH3=11.7
C        B=SQRT(40.0)
        B=2.5
C       WRITE(*,*)'DOMAIN OF SHAPE ?'
C       READ(5,*) B
       DO 21 I = 1,NPTS
C          T = B*(REAL(I-1)/REAL(NPTS-1)-.5)
         T = B*(FLOAT(I-1)/FLOAT(NPTS-1)-.5)
         IF(ABS(T).LT.1E-16) T=1.0E-8
          W(I) = (CH0*COS(CH1*T)-EXP(-T*T)*CH2*SIN(CH3*T)/T)*EXP(-T**6)
          IF (MX.LT.ABS(W(I))) MX = ABS(W(I))
 21   CONTINUE

C  BEGINNIG AND END MARKERS
C          W(1)=MX/4.0
C          W(NPTS)=MX/4.0
          print*, 'MAX OF SHAPE= ',MX
C          WRITE(*,*)'AMPLITUDE OF WAVEFOR ? (2047 is maximum)'
C          READ(*,*) AMP
           AMP=2047
       DO 210 I = 1,NPTS
C           WAVE(I)=AMP*W(I)/NPTS
            WAVE(I) =AMP*(W(I)/MX)
c          print*, 'wave = ',WAVE(I),'T= ',T
c          READ(*,*)
 210   CONTINUE
C NORMALIZE SHAPED PULSE
       DO I=1,NPTS
       W(I)=W(I)/MX
       ENDDO

       WRITE(*,*)'WHAT RF? (IN KHZ) 47'
       READ(5,*) RF
       PI=4.0*ATAN(1.0)
       RF=2.0*PI*RF*1000.0
       DO I=1,NPTS
        W(I)=RF*W(I)
       ENDDO
       WS=0.0
       DO I=1,NPTS
        WS=WS+W(I)
       ENDDO

C ADJUST OUTPUT RATE TO MATCH TIME-STEP, DLT, OF SAHPED PULSE
       DLT=(PI/2.0)/(ABS(WS))
       WRITE(*,*)'CALCULATED TIME STEP: ',DLT
       WRITE(*,*)'CALCULATED DURATION: ' ,NPTS*DLT
       WRITE(*,*)'************************'
       DUR=NPTS*DLT
       RATEHI=1 ! THIS PARAMETER DOES NOT MAKE A DIFFERENCE
       RATELO=DLT/((25.0E-9)*RATEHI)-1.0
       WRITE(*,*) 'DLT=',DLT,'  RATEHI=',RATEHI,'  RATELO=',RATELO
           DUR_RSL=25.0E-9*NPTS
           WRITE(*,*)'RESOLUTION OF DURATION IS ',DUR_RSL
           NDR=DUR/DUR_RSL
           DUR1=DUR_RSL*(RATELO+1)*1    ! NEAREST VALUE TO DESIRED DURATION
           WRITE(*,*)'NDR= ',NDR
```

```
            WRITE(*,*)'ACTUAL DURATION OF SHAPED PULSE IS ',DUR1
            DFDUR=DUR-DUR1
            WRITE(*,*)'DEVIATION FROM DESIRED DURATION ',DFDUR,DUR_RSL
            TPOINT=25.0E-9*(RATELO+1)*RATEHI
            WRITE(*,*)'TIME PER POINT IS ',TPOINT,' DLT=',DLT
            NPTSDF=DFDUR/TPOINT
            WRITE(*,*)'CHANGE NUMBER OF POINTS BY ',NPTSDF
            NPTS=NPTS+NPTSDF
            WRITE(*,*)'ADJUSTED NUMBER OF POINTS IS ',NPTS
C            DUR=TPOINT*NPTS
C  ******UPDATE SHAPED PULSE******
          MX=0.0
          DO I = 1,NPTS
C           T = B*(REAL(I-1)/REAL(NPTS-1)-.5)
           T = B*(FLOAT(I-1)/FLOAT(NPTS-1)-.5)
          IF(ABS(T).LT.1E-16) T=1.0E-8

           W(I) = (CH0*COS(CH1*T)-EXP(-T*T)*CH2*SIN(CH3*T)/T)*EXP(-T**6)

C           W(I) =(CH0+CH1*T+CH2*T*T+CH3*T*T*T)*EXP(-P*T*T)
           IF (MX.LT.ABS(W(I))) MX = ABS(W(I))
c           print*, 'w = ',W(I),'  T= ',T
c           READ(*,*)
          ENDDO
C           W(1)=MX/4.0
C           W(NPTS)=MX/4.0
           print*, 'MAX OF SHAPE= ',MX
C           WRITE(*,*)'AMPLITUDE OF WAVEFOR ? (2047 is maximum)'
C           READ(*,*) AMP
            AMP=2047
          DO I = 1,NPTS
c           WAVE(I) = 2047*(W(I)/MX)

C            W(I) = (W(I)/MX)
C            WAVE(I)=AMP*W(I)/NPTS
             WAVE(I) =AMP*(W(I)/MX)
c           print*, 'wave = ',WAVE(I),'T= ',T
c           READ(*,*)
          ENDDO
C
        DO I=1,NPTS
         W(I)=W(I)/MX
        ENDDO
        DO I=1,NPTS
         W(I)=RF*W(I)
        ENDDO
        WS=0.0
        DO I=1,NPTS
         WS=WS+W(I)
        ENDDO
        WRITE(*,*)'DURATION OF ADJUSTED SHAPE ',NPTS*TPOINT
        FLIP=WS*TPOINT
         WRITE(*,*)'FLIP ANGLE OF ADJUSTED SHAPED PULSE ',FLIP
        print*, 'Change duration of shaped pulse?'
        print*, '(Yes 1   No 0)'
        read(5,*) xt
        IF (xt.eq.1) THEN   !*****************
            WRITE(*,*)'WHAT DURATION? (IN Microseconds)'
            READ(5,*) DUR
            DUR=DUR*1.0E-6

            RATEHI=1 ! THIS PARAMETER DOES NOT MAKE A DIFFERENCE
```

```
            DLT=DUR/NPTS
            RATELO=DLT/((25.0E-9)*RATEHI)-1.0
            WRITE(*,*) 'DLT=',DLT,'  RATEHI=',RATEHI,'  RATELO=',RATELO
            DUR_RSL=25.0E-9*NPTS
            WRITE(*,*)'RESOLUTION OF DURATION IS ',DUR_RSL
            NDR=DUR/DUR_RSL
            DUR1=DUR_RSL*(RATELO+1)*1    ! NEAREST VALUE TO DESIRED DURATION
            WRITE(*,*)'NDR= ',NDR
            WRITE(*,*)'ACTUAL DURATION OF SHAPED PULSE IS ',DUR1
            DFDUR=DUR-DUR1
            WRITE(*,*)'DEVIATION FROM DESIRED DURATION ',DFDUR,DUR_RSL
            TPOINT=25.0E-9*(RATELO+1)*RATEHI
            WRITE(*,*)'TIME PER POINT IS ',TPOINT
            NPTSDF=DFDUR/TPOINT +1
            WRITE(*,*)'CHANGE NUMBER OF POINTS BY ',NPTSDF
            NPTS=NPTS+NPTSDF
            WRITE(*,*)'ADJUSTED NUMBER OF POINTS IS ',NPTS
            RATELO=TPOINT/(25.0E-9*RATEHI)-1
            WRITE(*,*)'ADJUSTED RATELO ',RATELO
C  ******UPDATE SHAPED PULSE******
           MX=0.0
           DO  I = 1,NPTS
C            T = B*(REAL(I-1)/REAL(NPTS-1)-.5)
            T = B*(FLOAT(I-1)/FLOAT(NPTS-1)-.5)
           IF(ABS(T).LT.1E-16) T=1.0E-8

            W(I) = (CH0*COS(CH1*T)-EXP(-T*T)*CH2*SIN(CH3*T)/T)*EXP(-T**6)
C            W(I) =(CH0+CH1*T+CH2*T*T+CH3*T*T*T)*EXP(-P*T*T)
            IF (MX.LT.ABS(W(I))) MX = ABS(W(I))
C             print*, 'w = ',W(I),'  T= ',T ,I,NPTS
c            READ(*,*)
           ENDDO
C            W(1)=MX/4.0
C            W(NPTS)=MX/4.0
            print*, 'MAX OF SHAPE= ',MX
C            WRITE(*,*)'AMPLITUDE OF WAVEFOR ? (2047 is maximum)'
C            READ(*,*) AMP
            AMP=2047
           DO  I = 1,NPTS
c            WAVE(I) = 2047*(W(I)/MX)

C             W(I) = (W(I)/MX)
C             WAVE(I)=AMP*W(I)/NPTS
              WAVE(I) =AMP*(W(I)/MX)
c            print*, 'wave = ',WAVE(I),'T= ',T
c            READ(*,*)
           ENDDO

         ENDIF     !****** END OF CONTROL LOOP **************

         WRITE(*,*)'DURATION OF ADJUSTED SHAPE ',NPTS*TPOINT
         FLIP=WS*TPOINT
         WRITE(*,*)'FLIP ANGLE OF ADJUSTED SHAPED PULSE ',FLIP

         CALL DEFWAV(WBASE,ADIR,NPTS,NBITS,WAVE(1))
         IF (STATUS .NE. 0) THEN
            WRITE(*,300) STATUS
 300        FORMAT(3X,'Status code = ',I3)
         ENDIF
c *                                                              *
c *                          Set clock                           *
c *                                                              *
```

```
c *     This sets the internal clock, the low bit divider,    *
c *     and the high bit divider.                             *
c *                                                           *
        CLKSRC = 0

        CALL SETCLK(CLKSRC,RATELO,RATEHI)
        IF (STATUS .NE. 0) THEN
           WRITE(*,400) STATUS
 400       FORMAT(3X,'Status code = ',I3)
        ENDIF
c *                                                           *
c *                        Set repitions                      *
c *                                                           *
c *     NOTE: REPT=0 sets a continuous wavefrom.              *
c *                                                           *

        print*,'Number of repetitons?'
        READ(5,*) REPT

        CALL SETRPT(REPT)
        IF (STATUS .NE. 0) THEN
           WRITE(*,500) STATUS
 500       FORMAT(3X,'Status code = ',I3)
        ENDIF
c *                                                           *
c *                          Set scale                        *
c *                                                           *
c *     This sets a full scale.                               *
c *                                                           *
        MULT = 127
        CALL WSCALE(MULT)
        IF (STATUS .NE. 0) THEN
           WRITE(*,600) STATUS
 600       FORMAT(3X,'Status code = ',I3)
        ENDIF
c *                                                           *
c *                          Set delay                        *
c *                                                           *
c *     This sets a delay of zero.                            *
c *                                                           *
        DLAY = 0
        CALL SETDLY(DLAY)
        IF (STATUS .NE. 0) THEN
           WRITE(*,700) STATUS
 700       FORMAT(3X,'Status code = ',I3)
        ENDIF
c *                                                           *
c *                        Set trigger                        *
c *                                                           *
c *     This sets an internal trigger and then returns to     *
c *     program mode.                                         *
c *                                                           *
c *     TSRC:  0 for internal trigger mode
c *            otherwise for external trigger mode
c *
c *     RETRIG: 0 return to 'program' mode after task
c *             otherwise remain in 'run' mode after task


        TSRC = 0
        RETRIG = 0
        print*,'External trigger mode ?'
```

```
      print*,'(Y=1,N=0, any other value end program)'
      read(5,*) xt
      IF (xt.eq.1) then
          print*,'TSRC ?'
          read(5,*) TSRC
          print*,'RETRIG ?'
          read(5,*) RETRIG
      ELSE
          goto 1000

      ENDIF

      CALL SETRIG(TSRC,TLAY,RETRIG)
      IF (STATUS .NE. 0) THEN
         WRITE(*,800) STATUS
 800     FORMAT(3X,'Status code = ',I3)
      ENDIF
c *                                                              *
c *                       Start waveform                         *
c *                                                              *
c *    This disables the auto-reverse and begins waveform.       *
c *                                                              *
      AREV = 0
      CALL WSTART(WBASE,ADIR,NPTS,AREV)
      IF (STATUS .NE. 0) THEN
         WRITE(*,900) STATUS
 900     FORMAT(3X,'Status code = ',I3)
      ENDIF
      goto 999
 1000 END.....................
```

```
      PROGRAM MACNMR
C     PROGRAM PG_ZOOM4
C
C THIS PROGRAM PLOTS UP TO FOUR-COLOMN INPUT DATA FILE and is part of my thesis
.
C THE FIRST PLOT IS REAL COMPONENT OF FID.
C THE SECOND PLOT IS IMAGINARY COMPONENT OF FID.
C THE THIRD PLOT IS MAGNITUDE OF FID.
C LABLES FOR AXES AND TITLE ARE ALSO INPUT BY USER.
C ANY PART OF PLOT CAN BE ZOOMED AND PRINTED.
C HAVE THE OPTION TO VIEW/PRINT PLOTS INDIVIDUALLY OR AS GROUP.
C MACNMR DATA FILE HAVE THEIR COLOMNS ARRANGED DIFFERENTLY.
C THIS PROGRAM IS SUITED FOR THAT SPECIFIC ORDER. ANOTHER APPROACH
C IS TO WRITE A PROGRAM WHICH REARRANGES THE COLOMNS TO OUR USUAL
C ORDER AND THEN USE ANY OF OUR EXISTING PGPLOT PLOTTING PROGRAMS.

      CHARACTER*50 NAME, LBX, LBY, LBTITLE, ZOOM,VW
      INTEGER I, J, NUMBER, NBIN, pgbeg, NK, IL, IR, JP,LW,IP
c                                 ^^^^^^--need to include this as an integer
c     INCLUDE 'AF.I'
      PARAMETER (N=10000)
      double precision X(N),V(N)
      REAL W, Z(N), T(N), Y(N), XAX(N), A, B, XTMP(N), YTMP(N),Z1(N)
      DIMENSION F1(n), f2(n), f3(n), ran(3)
      xmax=-1e+9
      xmin=-xmax
      YMAX=-200
      YMIN=200
c     NBIN=101
      s=10
      B=0.0
      NK=1000
      NBIN=NK

      WRITE(*,*)'INPUT NAME OF FILE (e.g filename.dat)'
      READ(*,'(A)') NAME

      LBX='TIME'
      WRITE(*,*)'INPUT X-AXIS TITLE'
      READ(*,'(A)') LBX

      LBY='AMPLITUDE'
      WRITE(*,*)'INPUT Y-AXIS TITLE'
      READ(*,'(A)') LBY

      LBTITLE='REAL PART'
      WRITE(*,*)'INPUT TITLE'
      READ(*,'(A)') LBTITLE

      WRITE(*,*)'WIDTH OF LINE? 1  2  3  4...'
      READ(*,*) LW

      WRITE(*,*)'ZOOM? (Y,N)'
      READ(*,'(A)') ZOOM

      IF(ZOOM.EQ.'Y') THEN
         WRITE(*,*)'ZOOM STARTING POINT'
         READ(*,*) IL
         WRITE(*,*)'ZOOM ENDING POINT'
         READ(*,*) IR
         NBIN=IR-IL
         WRITE(*,*) NBIN
```

```
        ELSE
           IL=1
           IR=NK
        END IF

        WRITE(*,*) 'VIEW ALL PLOTS TOGETHER? ( Y->YES  OTHER->NO )'
        READ(*,'(A)') VW
        IF(VW.EQ.'Y') THEN
           JP=-3
        ELSE
           JP=1
        END IF

        OPEN(UNIT=1,FILE=NAME,STATUS='OLD')
        DO 900 I=1,NK
c        READ(1,*,END=900) XAX(I),Y(I),Z(I),Z1(I)
C        READ(1,*,END=900) Y(I),Z(I),XAX(I),Z(1)
         IR=I
        READ(1,*,END=900) Y(I),Z(I),XAX(I)

c        READ(1,100,END=900) XAX(I),Y(I),Z(I),Z1(I)
C         READ(1,100,END=900) XAX(I),Y(I),Z(I)

900     CONTINUE
        CLOSE(1)
c100    format(1f5.3,1f5.3,1f7.1,1f6.2)
c100    FORMAT(4F20.8)
C100     FORMAT(4E13.6)

        DO 950 I=IL,IR
        XTMP(I)=XAX(I)
        YTMP(I)=Y(I)
950     CONTINUE

        DO 800 I=IL,IR
        IF(XMAX.LT.XTMP(I)) XMAX=XTMP(I)
        IF(YMAX.LT.YTMP(I)) YMAX=YTMP(I)
        IF(XMIN.GT.XTMP(I)) XMIN=XTMP(I)
        IF(YMIN.GT.YTMP(I)) YMIN=YTMP(I)
C        WRITE(*,*)YTMP(I),IR
800     CONTINUE
C        READ(*,*)

C   GRAPH DATA
C
        call pgslw(LW)
        IF ((PGBEG(0,'?',1,JP)).NE.1) STOP              !1
C       CALL PGPANL(1,1)
        CALL PGENV (xmin, xmax, ymin, ymax,0,1)         !2
c       CALL PGpt(Nbin,XAX,Y,-1)                        !3
c       CALL PGLAB('TIME','AMPLITUDE','REAL')  !4
        CALL PGLAB(LBX,LBY,LBTITLE)  !4
c       call pgslw(LW)
C       call pgline(nbin,xax,y)                         !5
        call pgline(nbin,XTMP,YTMP)                             !5

        xmax=-1e+9
        xmin=-xmax
        YMAX=-1E+9
        YMIN=1E+9
c  IMAGINARY COMPONENT
        DO 955 I=IL,IR
```

```
        XTMP(I)=XAX(I)
        YTMP(I)=Z(I)
955     CONTINUE
        IP=1
        DO 805 I=IL,IR
        IF(XMAX.LT.XTMP(I)) XMAX=XTMP(I)
        IF(YMAX.LT.YTMP(I)) YMAX=YTMP(I)
        IF(XMIN.GT.XTMP(I)) XMIN=XTMP(I)
        IF(YMIN.GT.YTMP(I)) YMIN=YTMP(I)
        IF(YMIN.GT.YTMP(I)) IP=I
805     CONTINUE
        YTMP(IR)=0.0
        WRITE(*,*)'XMN= ',XMAX,XMIN,'YMN= ',YMAX,YMIN,IP

C       LBTITLE='IMAGINERY PART'
C        WRITE(*,*)'INPUT TITLE (IMAGINARY)'
C       READ(*,'(A)') LBTITLE

C   GRAPH DATA
C
C       IF ((PGBEG(0,'?',1,JP)).NE.1) STOP              !1
C        YMIN=-200.0
        CALL PGENV (xmin, xmax, ymin, ymax,0,1)         !2
c       CALL PGpt(Nbin,XAX,Y,-1)                        !3
c       CALL PGLAB('TIME','AMPLITUDE','IMAGINARY')  !4
        CALL PGLAB(LBX,LBY,LBTITLE)  !4
        call pgslw(LW)
C       call pgline(nbin,xax,y)                         !5
        call pgline(nbin,XTMP,YTMP)                     !5

c  MAGNITUDE
        xmax=-1e+9
        xmin=-xmax
        YMAX=-1.0E+9
        YMIN=-YMAX

        DO 985 I=IL,IR
        XTMP(I)=XAX(I)
        YTMP(I)=Y(I)**2+Z(I)**2
985     CONTINUE

        DO 885 I=IL,IR
        IF(XMAX.LT.XTMP(I)) XMAX=XTMP(I)
        IF(YMAX.LT.YTMP(I)) YMAX=YTMP(I)
        IF(XMIN.GT.XTMP(I)) XMIN=XTMP(I)
        IF(YMIN.GT.YTMP(I)) YMIN=YTMP(I)
885     CONTINUE

C       LBTITLE='MAGNITUDE'
C        WRITE(*,*)'INPUT TITLE (MAGNITUDE)'
C       READ(*,'(A)') LBTITLE

C   GRAPH DATA
C
C       IF ((PGBEG(0,'?',1,JP)).NE.1) STOP              !1
        CALL PGENV (xmin, xmax, ymin, ymax,0,1)         !2
c       CALL PGpt(Nbin,XAX,Y,-1)                        !3
c       CALL PGLAB('TIME','AMPLITUDE','IMAGINARY')  !4
        CALL PGLAB(LBX,LBY,LBTITLE)  !4
        call pgslw(LW)
C       call pgline(nbin,xax,y)                         !5
        call pgline(nbin,XTMP,YTMP)                             !5
```

```
        call pgend                                              !6
c-------the above 6 lines are part of the plplot routines. If you just wanted
c*******a line then you would only require lines:1,2,5 and 6. Keramat has a
c*******book which has what different subroutines do.
        END
```

```
      PROGRAM PGS2ZNR
c     THIS PROGRAM STACKS PLOTS and is part of my thesis.
C     THE USER DECIDES HOW MANY.
C     WIDTH (X) OF PHYSICAL PAGE IS 8.8 .
C     HEIGHT (Y) OF PHYSICAL PAGE IS 11 .
C     EACH PLOT CAN BE ZOOMED HORIZONTALLY.
C     BY ENTERING AN UPPER BOUND FOR THE
C     FREQUENCY WHICH SETS THE LOWER BOUND
C     (SYMMETRIC PLOTS SUCH AS SPECTRAS).
C     THE UNITS OF THE X DATA CAN BE CHANGED
C     I.E X DATA CAN BE DIVIDED BY A FACTOR.
C     SO FOR EXAMPLE FRRQUENCY CAN BE SCALED
C     TO KHZ, OR TIME CAN BE IN MICROSECONDS.
C     THIS PROGRAM USES A "WHILE" LOOP !!!
      PARAMETER (N=20000)
      REAL XL,XR,YB,XAX(N),XTMP(N),DY
      REAL XTK,YTK,Y(N),YTMP(N),Z(N)
      REAL XMIN,YMIN,YMAX,R,DX,DLX,XMAX
      INTEGER NX,NY,NBIN,LW,NK
      INTEGER PGBEG,JP,JZ,L
      CHARACTER*26 NAME,XOPT,YOPT
      CHARACTER*26 LBX,LBY,LBT,SZ

      COMMON/S1/LBX,LBY,LBT,XOPT,YOPT   !STRING VARIABLES
      COMMON/S2/JP,LW,NX,NY             !INTEGER VARIABLES
      COMMON/S3/XL,XR,YB,YT             !REAL VARIABLES
      COMMON/S4/PBEG

      L=0
111   WRITE(*,*)'HOW MANY PLOTS?'
      READ(5,*) L
      IF(L.EQ.0) GOTO 111

      R=1.0E+9
      XL=0.0
      XR=4.0
      YFST=1.2
      YB=0.0+YFST
C     YT=2.0+YFST
      YT=11/L+YFST
C     DY=yt-yb
c     DY=(11-3*YFST)/L
      DY=(11-2.0)/L
      YT=DY+YFST
      DX=XR/4.0
      XTK=0.0
      YTK=0.0
      NX=XR-XL+1
      NY=NX
c     LW=1
      WRITE(*,*)'LINE THICKNESS? (INCREASES WITH NO. 1 2 3 4)'
      READ(5,*)LW
      XL=XL+DX*0.0
      XR=XR+DX*2.0
      DLX=0.9
      XL=XL+DLX
      XR=XR+DLX
      CALL BEGIN_GRAPHICS
      RXSCL=1.0
c     ************************
      DO IX=1,L
      XMIN=R
```

```
        YMIN=R
        XMAX=-R
        YMAX=-R

c       NAME='AMPLITUDE.DAT'
222     WRITE(*,*)'NAME OF FILE'
        READ(5,'(A)') NAME
        IF(NAME.EQ.' ') GOTO 222
        WRITE(*,*)'RESCALE X-AXIS? 1->YES OTHER NO'
        READ(5,*) IXANSW
        IF(IXANSW.EQ.1) THEN
           WRITE(*,*)'ENTER SCALING FACTOR?'
           READ(5,*) RXSCL
        ENDIF
        OPEN(UNIT=1,FILE=NAME,STATUS='OLD')
        DO 900 I=1,N
C         READ(1,100,END=900) XAX(I),Y(I)
c         READ(1,*,END=901) XAX(I),Y(I)
c         READ(1,*,END=901) Y(I),Z(I),XAX(I)
         READ(1,*,END=901) XAX(I),Y(I)
         XAX(I)=XAX(I)/RXSCL
         JZ=I
C        WRITE(*,*) XAX(I),Y(I),Z(I),I
C        WRITE(*,100) XAX(I),Y(I),Z(I)
900     CONTINUE
901        CLOSE(1)
C100    FORMAT(3F13.6)
        write(*,*) JZ
        NK=JZ
        nbin= NK
        ICX=NK
        YMAX=-10.0
        YMIN=10.0
        DO I=1,NK
        IF(XMAX.LT.XAX(I)) XMAX=XAX(I)
        IF(YMAX.LT.Y(I)) YMAX=Y(I)
        IF(XMIN.GT.XAX(I)) XMIN=XAX(I)
        IF(YMIN.GT.Y(I)) YMIN=Y(I)
C        WRITE(*,*)YTMP(I),IR
        ENDDO
C        READ(*,*)
        WRITE(*,*)'YMAX=',YMAX,YMIN
        WRITE(*,*)'XMAN=',XMAX,XMIN
C************************
        WRITE(*,*)'ZOOM? (Y,N)'
        READ(*,'(A)') ZOOM
        ICX=JZ
        IF(ZOOM.EQ.'Y') THEN
           WRITE(*,*)'FREQUENCY ENDPOINT (XMAX)'
           READ(5,*) XMAX
           XMIN=-XMAX
           I=1
           DO WHILE(XAX(I).GE.XMAX)
              I=I+1
           ENDDO
           NBIN=NK-2*I
           WRITE(*,*)'NUMBER OF POINTS ',NBIN
           IL=ICX/2-NBIN/2
           IR=ICX/2+NBIN/2
           WRITE(*,*) NBIN
        ELSE
           IL=1
```

```
            IR=ICX
            NBIN=ICX
            WRITE(*,*)'NBIN=',NBIN
         END IF
         DO 950 I=IL,IR
         II=I-IL+1
         XTMP(II)=XAX(I)
         YTMP(II)=Y(I)
950      CONTINUE

C        DO 800 I=1,IR-IL+1
C        IF(XMAX.LT.XTMP(I)) XMAX=XTMP(I)
C         IF(YMAX.LT.YTMP(I)) YMAX=YTMP(I)
C         IF(XMIN.GT.XTMP(I)) XMIN=XTMP(I)
C        IF(YMIN.GT.YTMP(I)) YMIN=YTMP(I)
C         WRITE(*,*)YTMP(I),IR
C800     CONTINUE
C         READ(*,*)

C*************************
C         XMAX=INT(XMAX/10+1)*10
         WRITE(*,*)'YMAX=',YMAX,YMIN
         WRITE(*,*)'XMAN=',XMAX,XMIN

         WRITE(*,*)'NEW END POINTS FOR X-AXIS? 1->YES OTHER NO'
         READ(5,*) IANSW
         IF(IANSW.EQ.1) THEN
            WRITE(*,*)'RIGHT X-ENDPOINT?'
C           XSCALE=1.0
            READ(5,*) XMAX
C           XMAX=XMIN+XSCALE
            WRITE(*,*)'LEFT  X-ENDPOINT?'
            READ(5,*) XMIN
         ENDIF
         WRITE(*,*)'NEW END POINTS FOR Y-AXIS? 1->YES OTHER NO'
         READ(5,*) IANSW
         IF(IANSW.EQ.1) THEN
C           YSCALE=1.0
C           WRITE(*,*)'Y-SCALE?'
C           READ(5,*) YSCALE
            WRITE(*,*)'TOP    Y-ENDPOINT?'
            READ(5,*) YMAX
            WRITE(*,*)'BOTTOM Y-ENDPOINT?'
            READ(5,*) YMIN
C           YMAX=YMIN+YSCALE
         ENDIF

         LBX=' '
         LBY=' '
         LBT=' '
         YOPT=' '
         XOPT=' '
         IF(IX.EQ.1) THEN
            WRITE(*,*)'LABEL FOR X-AXIS'
            READ(5,'(A)') LBX
C           LBX='FREQUENCY'
            XOPT='BINT1'
         ELSE
            IF(IX.EQ.(L/2+1)) THEN
               WRITE(*,*)'LABEL FOR Y-AXIS'
               READ(5,'(A)') LBY
C              LBY='INTENSITY'
```

```
              YOPT='BNVT1'
C              YOPT='BNT1'

            ELSE
               IF(IX.EQ.L) THEN
                  WRITE(*,*)'TITLE'
                  READ(5,'(A)') LBT
               ENDIF
            ENDIF
         ENDIF
         JP=IX
         yopt='BNIVT1'
         CALL DOPLOT(XMIN,XMAX,YMIN,YMAX,NBIN,XTMP,YTMP)
         YB=YB+DY
         YT=YT+DY
         ENDDO
c      *********************
         CALL END_GRAPHICS

         END
c*****************************************
         SUBROUTINE BEGIN_GRAPHICS
         INTEGER PGBEG
         COMMON/S4/PBEG
         IF ((PGBEG(0,'?',1,JP)).NE.1) STOP              !1
         RETURN
         END

         SUBROUTINE DOPLOT(XMIN,XMAX,YMIN,YMAX,NBIN,XTMP,YTMP)
         REAL XTMP(*),YTMP(*),XMIN,XMAX,YMIN,YMAX
         REAL XL,XR,YB,YT,XTK,YTK,CHT
         INTEGER NBIN,JP,LW,NX,NY
         CHARACTER*26 LBX,LBY,LBT,YOPT,XOPT,LBNM
         COMMON/S1/LBX,LBY,LBT,XOPT,YOPT    !STRING VARIABLES
         COMMON/S2/JP,LW,NX,NY              !INTEGER VARIABLES
         COMMON/S3/XL,XR,YB,YT              !REAL VARIABLES
         COMMON/S4/PBEG

         XTK=0.0
         YTK=0.0
c        JP=1
         LBNM=' '
         IF(JP.EQ.1) LBNM='A'
         IF(JP.EQ.2) LBNM='B'
         IF(JP.EQ.3) LBNM='C'
         IF(JP.EQ.4) LBNM='D'
         IF(JP.EQ.5) LBNM='E'

c        IF ((PGBEG(0,'?',1,JP)).NE.1) STOP              !1
         CALL PGSCF(2)
         CHT=1.2
         CALL PGSCH(CHT)
         CALL PGSLW(LW)
         CALL PGVSIZ(XL,XR,YB,YT-.3)
C         CALL PGSVP(XL,XR,YB,YT)      !SET VIEWPORT IN NORMALIZED COORDINATES.
C         CALL PGWNAD(XL,XR,YB,YT)       !ADJUST WINDOW IN WORLD COORDINATES.
         CALL PGSWIN(XMIN,XMAX,YMIN,YMAX)!ADJUST WINDOW IN WORLD COORDINATES
                                         !TO BE MAPPED ON TO VIEWPORT.
         CALL PGBOX(XOPT,XTK,NX,YOPT,YTK,NY) !DRAW LABELED FRAME AROUND VIEWPORT
         CALL PGSLW(LW)
         CALL PGLINE(NBIN,XTMP,YTMP)
C        CALL PGSCF(2)
```

```
C        CALL PGSCH(1.2)
c        CALL PGLAB(LBX,LBY,LBT)
c        CALL PGMTXT('B',-2.0,.0,0.5,LBX)
         CALL PGPTXT(xmin*1.2,ymax*.1,90.0,0.0,LBY)
              DXX=(XMAX-XMIN)/2.0
              XADJ=xmin+DXX
              DYY=(YMAX-YMIN)*.1
              YADJ= YMIN-DYY*2


         CALL PGMTXT('B',3*CHT,0.5,0.5,LBX)

C        CALL PGPTXT(XADJ,YADJ,0.0,0.5,LBX)
              YADJ=YMAX+DYY
         CALL PGMTXT('t',1*CHT,0.5,0.5,LBT)
c        CALL PGPTXT(XADJ,YADJ,0.0,0.5,LBT)
         CALL PGSCH(2.0)
         CALL PGPTXT(xmax*.95,ymax*.25,0.0,0.0,LBNM)

         RETURN
         END
c*************************************
         SUBROUTINE END_GRAPHICS
         CALL PGEND
         RETURN
         END
```

```
      PROGRAM PGS2ZNR
c     THIS PROGRAM STACKS PLOTS and is part of my thesis.
C     THE USER DECIDES HOW MANY.
C     WIDTH (X) OF PHYSICAL PAGE IS 8.8 .
C     HEIGHT (Y) OF PHYSICAL PAGE IS 11 .
C     EACH PLOT CAN BE ZOOMED HORIZONTALLY.
C     BY ENTERING AN UPPER BOUND FOR THE
C     FREQUENCY WHICH SETS THE LOWER BOUND
C     (SYMMETRIC PLOTS SUCH AS SPECTRAS).
C     THE UNITS OF THE X DATA CAN BE CHANGED
C     I.E X DATA CAN BE DIVIDED BY A FACTOR.
C     SO FOR EXAMPLE FRRQUENCY CAN BE SCALED
C     TO KHZ, OR TIME CAN BE IN MICROSECONDS.
C     THIS PROGRAM USES A "WHILE" LOOP !!!
      PARAMETER (N=20000)
      REAL XL,XR,YB,XAX(N),XTMP(N),DY
      REAL XTK,YTK,Y(N),YTMP(N),Z(N)
      REAL XMIN,YMIN,YMAX,R,DX,DLX,XMAX
      INTEGER NX,NY,NBIN,LW,NK
      INTEGER PGBEG,JP,JZ,L
      CHARACTER*26 NAME,XOPT,YOPT
      CHARACTER*26 LBX,LBY,LBT,SZ

      COMMON/S1/LBX,LBY,LBT,XOPT,YOPT    !STRING VARIABLES
      COMMON/S2/JP,LW,NX,NY              !INTEGER VARIABLES
      COMMON/S3/XL,XR,YB,YT              !REAL VARIABLES
      COMMON/S4/PBEG

      L=0
111   WRITE(*,*)'HOW MANY PLOTS?'
      READ(5,*) L
      IF(L.EQ.0) GOTO 111

      R=1.0E+9
      XL=0.0
      XR=4.0
      YFST=1.2
      YB=0.0+YFST
C     YT=2.0+YFST
      YT=11/L+YFST
C     DY=yt-yb
c     DY=(11-3*YFST)/L
      DY=(11-2.0)/L
      YT=DY+YFST
      DX=XR/4.0
      XTK=0.0
      YTK=0.0
      NX=XR-XL+1
      NY=NX
c     LW=1
      WRITE(*,*)'LINE THICKNESS? (INCREASES WITH NO. 1 2 3 4)'
      READ(5,*)LW
      XL=XL+DX*0.0
      XR=XR+DX*2.0
      DLX=0.9
      XL=XL+DLX
      XR=XR+DLX
      CALL BEGIN_GRAPHICS
      RXSCL=1.0
c     ************************
      DO IX=1,L
      XMIN=R
```

```
        YMIN=R
        XMAX=-R
        YMAX=-R

c       NAME='AMPLITUDE.DAT'
222     WRITE(*,*)'NAME OF FILE'
        READ(5,'(A)') NAME
        IF(NAME.EQ.' ') GOTO 222
        WRITE(*,*)'RESCALE X-AXIS? 1->YES OTHER NO'
        READ(5,*) IXANSW
        IF(IXANSW.EQ.1) THEN
           WRITE(*,*)'ENTER SCALING FACTOR?'
           READ(5,*) RXSCL
        ENDIF
        OPEN(UNIT=1,FILE=NAME,STATUS='OLD')
        DO 900 I=1,N
C         READ(1,100,END=900) XAX(I),Y(I)
c         READ(1,*,END=901) XAX(I),Y(I)
c         READ(1,*,END=901) Y(I),Z(I),XAX(I)
         READ(1,*,END=901) XAX(I),Y(I)
         XAX(I)=XAX(I)/RXSCL
         JZ=I
C        WRITE(*,*) XAX(I),Y(I),Z(I),I
C        WRITE(*,100) XAX(I),Y(I),Z(I)
900     CONTINUE
901        CLOSE(1)
C100    FORMAT(3F13.6)
        write(*,*) JZ
        NK=JZ
        nbin= NK
        ICX=NK
        YMAX=-10.0
        YMIN=10.0
        DO I=1,NK
        IF(XMAX.LT.XAX(I)) XMAX=XAX(I)
        IF(YMAX.LT.Y(I)) YMAX=Y(I)
        IF(XMIN.GT.XAX(I)) XMIN=XAX(I)
        IF(YMIN.GT.Y(I)) YMIN=Y(I)
C        WRITE(*,*)YTMP(I),IR
        ENDDO
C        READ(*,*)
        WRITE(*,*)'YMAX=',YMAX,YMIN
        WRITE(*,*)'XMAN=',XMAX,XMIN
C************************
        WRITE(*,*)'ZOOM? (Y,N)'
        READ(*,'(A)') ZOOM
        ICX=JZ
        IF(ZOOM.EQ.'Y') THEN
           WRITE(*,*)'FREQUENCY ENDPOINT (XMAX)'
           READ(5,*) XMAX
           XMIN=-XMAX
           I=1
           DO WHILE(XAX(I).GE.XMAX)
              I=I+1
           ENDDO
           NBIN=NK-2*I
           WRITE(*,*)'NUMBER OF POINTS ',NBIN
           IL=ICX/2-NBIN/2
           IR=ICX/2+NBIN/2
           WRITE(*,*) NBIN
        ELSE
           IL=1
```

```
            IR=ICX
            NBIN=ICX
            WRITE(*,*)'NBIN=',NBIN
         END IF
         DO 950 I=IL,IR
         II=I-IL+1
         XTMP(II)=XAX(I)
         YTMP(II)=Y(I)
950      CONTINUE

C        DO 800 I=1,IR-IL+1
C        IF(XMAX.LT.XTMP(I)) XMAX=XTMP(I)
C         IF(YMAX.LT.YTMP(I)) YMAX=YTMP(I)
C         IF(XMIN.GT.XTMP(I)) XMIN=XTMP(I)
C        IF(YMIN.GT.YTMP(I)) YMIN=YTMP(I)
C         WRITE(*,*)YTMP(I),IR
C800     CONTINUE
C         READ(*,*)

C*************************
C         XMAX=INT(XMAX/10+1)*10
         WRITE(*,*)'YMAX=',YMAX,YMIN
         WRITE(*,*)'XMAN=',XMAX,XMIN

         WRITE(*,*)'NEW END POINTS FOR X-AXIS? 1->YES OTHER NO'
         READ(5,*) IANSW
         IF(IANSW.EQ.1) THEN
            WRITE(*,*)'RIGHT X-ENDPOINT?'
C           XSCALE=1.0
            READ(5,*) XMAX
C           XMAX=XMIN+XSCALE
            WRITE(*,*)'LEFT  X-ENDPOINT?'
            READ(5,*) XMIN
         ENDIF
         WRITE(*,*)'NEW END POINTS FOR Y-AXIS? 1->YES OTHER NO'
         READ(5,*) IANSW
         IF(IANSW.EQ.1) THEN
C           YSCALE=1.0
C           WRITE(*,*)'Y-SCALE?'
C           READ(5,*) YSCALE
            WRITE(*,*)'TOP    Y-ENDPOINT?'
            READ(5,*) YMAX
            WRITE(*,*)'BOTTOM Y-ENDPOINT?'
            READ(5,*) YMIN
C           YMAX=YMIN+YSCALE
         ENDIF

         LBX=' '
         LBY=' '
         LBT=' '
         YOPT=' '
         XOPT=' '
         IF(IX.EQ.1) THEN
            WRITE(*,*)'LABEL FOR X-AXIS'
            READ(5,'(A)') LBX
C           LBX='FREQUENCY'
            XOPT='BINT1'
         ELSE
            IF(IX.EQ.(L/2+1)) THEN
               WRITE(*,*)'LABEL FOR Y-AXIS'
               READ(5,'(A)') LBY
C              LBY='INTENSITY'
```

```
              YOPT='BNVT1'
C              YOPT='BNT1'

            ELSE
               IF(IX.EQ.L) THEN
                  WRITE(*,*)'TITLE'
                  READ(5,'(A)') LBT
               ENDIF
            ENDIF
         ENDIF
         JP=IX
         yopt='BNIVT1'
         CALL DOPLOT(XMIN,XMAX,YMIN,YMAX,NBIN,XTMP,YTMP)
         YB=YB+DY
         YT=YT+DY
         ENDDO
c     **********************
         CALL END_GRAPHICS

         END
c*****************************************
         SUBROUTINE BEGIN_GRAPHICS
         INTEGER PGBEG
         COMMON/S4/PBEG
         IF ((PGBEG(0,'?',1,JP)).NE.1) STOP              !1
         RETURN
         END

         SUBROUTINE DOPLOT(XMIN,XMAX,YMIN,YMAX,NBIN,XTMP,YTMP)
         REAL XTMP(*),YTMP(*),XMIN,XMAX,YMIN,YMAX
         REAL XL,XR,YB,YT,XTK,YTK,CHT
         INTEGER NBIN,JP,LW,NX,NY
         CHARACTER*26 LBX,LBY,LBT,YOPT,XOPT,LBNM
         COMMON/S1/LBX,LBY,LBT,XOPT,YOPT    !STRING VARIABLES
         COMMON/S2/JP,LW,NX,NY              !INTEGER VARIABLES
         COMMON/S3/XL,XR,YB,YT              !REAL VARIABLES
         COMMON/S4/PBEG

         XTK=0.0
         YTK=0.0
c        JP=1
         LBNM=' '
         IF(JP.EQ.1) LBNM='A'
         IF(JP.EQ.2) LBNM='B'
         IF(JP.EQ.3) LBNM='C'
         IF(JP.EQ.4) LBNM='D'
         IF(JP.EQ.5) LBNM='E'

c        IF ((PGBEG(0,'?',1,JP)).NE.1) STOP              !1
         CALL PGSCF(2)
         CHT=1.2
         CALL PGSCH(CHT)
         CALL PGSLW(LW)
         CALL PGVSIZ(XL,XR,YB,YT-.3)
C         CALL PGSVP(XL,XR,YB,YT)      !SET VIEWPORT IN NORMALIZED COORDINATES.
C         CALL PGWNAD(XL,XR,YB,YT)        !ADJUST WINDOW IN WORLD COORDINATES.
         CALL PGSWIN(XMIN,XMAX,YMIN,YMAX)!ADJUST WINDOW IN WORLD COORDINATES
                                         !TO BE MAPPED ON TO VIEWPORT.
         CALL PGBOX(XOPT,XTK,NX,YOPT,YTK,NY) !DRAW LABELED FRAME AROUND VIEWPORT
         CALL PGSLW(LW)
         CALL PGLINE(NBIN,XTMP,YTMP)
C        CALL PGSCF(2)
```

```
C        CALL PGSCH(1.2)
c        CALL PGLAB(LBX,LBY,LBT)
c        CALL PGMTXT('B',-2.0,.0,0.5,LBX)
         CALL PGPTXT(xmin*1.2,ymax*.1,90.0,0.0,LBY)
              DXX=(XMAX-XMIN)/2.0
              XADJ=xmin+DXX
              DYY=(YMAX-YMIN)*.1
              YADJ= YMIN-DYY*2


         CALL PGMTXT('B',3*CHT,0.5,0.5,LBX)

C        CALL PGPTXT(XADJ,YADJ,0.0,0.5,LBX)
              YADJ=YMAX+DYY
         CALL PGMTXT('t',1*CHT,0.5,0.5,LBT)
c        CALL PGPTXT(XADJ,YADJ,0.0,0.5,LBT)
         CALL PGSCH(2.0)
         CALL PGPTXT(xmax*.95,ymax*.25,0.0,0.0,LBNM)

         RETURN
         END
c************************************
         SUBROUTINE END_GRAPHICS
         CALL PGEND
         RETURN
         END
```

```
      PROGRAM PGS2ZNR
c     THIS PROGRAM STACKS PLOTS and is part of my thesis.
C     THE USER DECIDES HOW MANY.
C     WIDTH (X) OF PHYSICAL PAGE IS 8.8 .
C     HEIGHT (Y) OF PHYSICAL PAGE IS 11 .
C     EACH PLOT CAN BE ZOOMED HORIZONTALLY.
C     BY ENTERING AN UPPER BOUND FOR THE
C     FREQUENCY WHICH SETS THE LOWER BOUND
C     (SYMMETRIC PLOTS SUCH AS SPECTRAS).
C     THE UNITS OF THE X DATA CAN BE CHANGED
C     I.E X DATA CAN BE DIVIDED BY A FACTOR.
C     SO FOR EXAMPLE FRRQUENCY CAN BE SCALED
C     TO KHZ, OR TIME CAN BE IN MICROSECONDS.
C     THIS PROGRAM USES A "WHILE" LOOP !!!
      PARAMETER (N=20000)
      REAL XL,XR,YB,XAX(N),XTMP(N),DY
      REAL XTK,YTK,Y(N),YTMP(N),Z(N)
      REAL XMIN,YMIN,YMAX,R,DX,DLX,XMAX
      INTEGER NX,NY,NBIN,LW,NK
      INTEGER PGBEG,JP,JZ,L
      CHARACTER*26 NAME,XOPT,YOPT
      CHARACTER*26 LBX,LBY,LBT,SZ

      COMMON/S1/LBX,LBY,LBT,XOPT,YOPT   !STRING VARIABLES
      COMMON/S2/JP,LW,NX,NY             !INTEGER VARIABLES
      COMMON/S3/XL,XR,YB,YT             !REAL VARIABLES
      COMMON/S4/PBEG

      L=0
111   WRITE(*,*)'HOW MANY PLOTS?'
      READ(5,*) L
      IF(L.EQ.0) GOTO 111

      R=1.0E+9
      XL=0.0
      XR=4.0
      YFST=1.2
      YB=0.0+YFST
C     YT=2.0+YFST
      YT=11/L+YFST
C     DY=yt-yb
c     DY=(11-3*YFST)/L
      DY=(11-2.0)/L
      YT=DY+YFST
      DX=XR/4.0
      XTK=0.0
      YTK=0.0
      NX=XR-XL+1
      NY=NX
c     LW=1
      WRITE(*,*)'LINE THICKNESS? (INCREASES WITH NO. 1 2 3 4)'
      READ(5,*)LW
      XL=XL+DX*0.0
      XR=XR+DX*2.0
      DLX=0.9
      XL=XL+DLX
      XR=XR+DLX
      CALL BEGIN_GRAPHICS
      RXSCL=1.0
c     ************************
      DO IX=1,L
      XMIN=R
```

```
        YMIN=R
        XMAX=-R
        YMAX=-R

c       NAME='AMPLITUDE.DAT'
222     WRITE(*,*)'NAME OF FILE'
        READ(5,'(A)') NAME
        IF(NAME.EQ.' ') GOTO 222
        WRITE(*,*)'RESCALE X-AXIS? 1->YES OTHER NO'
        READ(5,*) IXANSW
        IF(IXANSW.EQ.1) THEN
           WRITE(*,*)'ENTER SCALING FACTOR?'
           READ(5,*) RXSCL
        ENDIF
        OPEN(UNIT=1,FILE=NAME,STATUS='OLD')
        DO 900 I=1,N
C         READ(1,100,END=900) XAX(I),Y(I)
c         READ(1,*,END=901) XAX(I),Y(I)
c         READ(1,*,END=901) Y(I),Z(I),XAX(I)
         READ(1,*,END=901) XAX(I),Y(I)
         XAX(I)=XAX(I)/RXSCL
         JZ=I
C        WRITE(*,*) XAX(I),Y(I),Z(I),I
C        WRITE(*,100) XAX(I),Y(I),Z(I)
900     CONTINUE
901        CLOSE(1)
C100    FORMAT(3F13.6)
        write(*,*) JZ
        NK=JZ
        nbin= NK
        ICX=NK
        YMAX=-10.0
        YMIN=10.0
        DO I=1,NK
        IF(XMAX.LT.XAX(I)) XMAX=XAX(I)
        IF(YMAX.LT.Y(I)) YMAX=Y(I)
        IF(XMIN.GT.XAX(I)) XMIN=XAX(I)
        IF(YMIN.GT.Y(I)) YMIN=Y(I)
C        WRITE(*,*)YTMP(I),IR
        ENDDO
C        READ(*,*)
        WRITE(*,*)'YMAX=',YMAX,YMIN
        WRITE(*,*)'XMAN=',XMAX,XMIN
C************************
        WRITE(*,*)'ZOOM? (Y,N)'
        READ(*,'(A)') ZOOM
        ICX=JZ
        IF(ZOOM.EQ.'Y') THEN
           WRITE(*,*)'FREQUENCY ENDPOINT (XMAX)'
           READ(5,*) XMAX
           XMIN=-XMAX
           I=1
           DO WHILE(XAX(I).GE.XMAX)
              I=I+1
           ENDDO
           NBIN=NK-2*I
           WRITE(*,*)'NUMBER OF POINTS ',NBIN
           IL=ICX/2-NBIN/2
           IR=ICX/2+NBIN/2
           WRITE(*,*) NBIN
        ELSE
           IL=1
```

```
            IR=ICX
            NBIN=ICX
            WRITE(*,*)'NBIN=',NBIN
         END IF
         DO 950 I=IL,IR
         II=I-IL+1
         XTMP(II)=XAX(I)
         YTMP(II)=Y(I)
950      CONTINUE

C        DO 800 I=1,IR-IL+1
C        IF(XMAX.LT.XTMP(I)) XMAX=XTMP(I)
C         IF(YMAX.LT.YTMP(I)) YMAX=YTMP(I)
C         IF(XMIN.GT.XTMP(I)) XMIN=XTMP(I)
C        IF(YMIN.GT.YTMP(I)) YMIN=YTMP(I)
C         WRITE(*,*)YTMP(I),IR
C800     CONTINUE
C         READ(*,*)

C************************
C         XMAX=INT(XMAX/10+1)*10
         WRITE(*,*)'YMAX=',YMAX,YMIN
         WRITE(*,*)'XMAN=',XMAX,XMIN

         WRITE(*,*)'NEW END POINTS FOR X-AXIS? 1->YES OTHER NO'
         READ(5,*) IANSW
         IF(IANSW.EQ.1) THEN
            WRITE(*,*)'RIGHT X-ENDPOINT?'
C           XSCALE=1.0
            READ(5,*) XMAX
C           XMAX=XMIN+XSCALE
            WRITE(*,*)'LEFT  X-ENDPOINT?'
            READ(5,*) XMIN
         ENDIF
         WRITE(*,*)'NEW END POINTS FOR Y-AXIS? 1->YES OTHER NO'
         READ(5,*) IANSW
         IF(IANSW.EQ.1) THEN
C           YSCALE=1.0
C           WRITE(*,*)'Y-SCALE?'
C           READ(5,*) YSCALE
            WRITE(*,*)'TOP    Y-ENDPOINT?'
            READ(5,*) YMAX
            WRITE(*,*)'BOTTOM Y-ENDPOINT?'
            READ(5,*) YMIN
C           YMAX=YMIN+YSCALE
         ENDIF

         LBX=' '
         LBY=' '
         LBT=' '
         YOPT=' '
         XOPT=' '
         IF(IX.EQ.1) THEN
            WRITE(*,*)'LABEL FOR X-AXIS'
            READ(5,'(A)') LBX
C           LBX='FREQUENCY'
            XOPT='BINT1'
         ELSE
            IF(IX.EQ.(L/2+1)) THEN
               WRITE(*,*)'LABEL FOR Y-AXIS'
               READ(5,'(A)') LBY
C              LBY='INTENSITY'
```

```
              YOPT='BNVT1'
C              YOPT='BNT1'

            ELSE
               IF(IX.EQ.L) THEN
                  WRITE(*,*)'TITLE'
                  READ(5,'(A)') LBT
               ENDIF
            ENDIF
         ENDIF
         JP=IX
         yopt='BNIVT1'
         CALL DOPLOT(XMIN,XMAX,YMIN,YMAX,NBIN,XTMP,YTMP)
         YB=YB+DY
         YT=YT+DY
         ENDDO
c     **********************
         CALL END_GRAPHICS

         END
c*****************************************
         SUBROUTINE BEGIN_GRAPHICS
         INTEGER PGBEG
         COMMON/S4/PBEG
         IF ((PGBEG(0,'?',1,JP)).NE.1) STOP              !1
         RETURN
         END

         SUBROUTINE DOPLOT(XMIN,XMAX,YMIN,YMAX,NBIN,XTMP,YTMP)
         REAL XTMP(*),YTMP(*),XMIN,XMAX,YMIN,YMAX
         REAL XL,XR,YB,YT,XTK,YTK,CHT
         INTEGER NBIN,JP,LW,NX,NY
         CHARACTER*26 LBX,LBY,LBT,YOPT,XOPT,LBNM
         COMMON/S1/LBX,LBY,LBT,XOPT,YOPT   !STRING VARIABLES
         COMMON/S2/JP,LW,NX,NY             !INTEGER VARIABLES
         COMMON/S3/XL,XR,YB,YT             !REAL VARIABLES
         COMMON/S4/PBEG

         XTK=0.0
         YTK=0.0
c        JP=1
         LBNM=' '
         IF(JP.EQ.1) LBNM='A'
         IF(JP.EQ.2) LBNM='B'
         IF(JP.EQ.3) LBNM='C'
         IF(JP.EQ.4) LBNM='D'
         IF(JP.EQ.5) LBNM='E'

c        IF ((PGBEG(0,'?',1,JP)).NE.1) STOP              !1
         CALL PGSCF(2)
         CHT=1.2
         CALL PGSCH(CHT)
         CALL PGSLW(LW)
         CALL PGVSIZ(XL,XR,YB,YT-.3)
C         CALL PGSVP(XL,XR,YB,YT)      !SET VIEWPORT IN NORMALIZED COORDINATES.
C         CALL PGWNAD(XL,XR,YB,YT)        !ADJUST WINDOW IN WORLD COORDINATES.
         CALL PGSWIN(XMIN,XMAX,YMIN,YMAX)!ADJUST WINDOW IN WORLD COORDINATES
                                         !TO BE MAPPED ON TO VIEWPORT.
         CALL PGBOX(XOPT,XTK,NX,YOPT,YTK,NY) !DRAW LABELED FRAME AROUND VIEWPORT
         CALL PGSLW(LW)
         CALL PGLINE(NBIN,XTMP,YTMP)
C        CALL PGSCF(2)
```

```
C        CALL PGSCH(1.2)
c        CALL PGLAB(LBX,LBY,LBT)
c        CALL PGMTXT('B',-2.0,.0,0.5,LBX)
         CALL PGPTXT(xmin*1.2,ymax*.1,90.0,0.0,LBY)
              DXX=(XMAX-XMIN)/2.0
              XADJ=xmin+DXX
              DYY=(YMAX-YMIN)*.1
              YADJ= YMIN-DYY*2


         CALL PGMTXT('B',3*CHT,0.5,0.5,LBX)

C        CALL PGPTXT(XADJ,YADJ,0.0,0.5,LBX)
              YADJ=YMAX+DYY
         CALL PGMTXT('t',1*CHT,0.5,0.5,LBT)
c        CALL PGPTXT(XADJ,YADJ,0.0,0.5,LBT)
         CALL PGSCH(2.0)
         CALL PGPTXT(xmax*.95,ymax*.25,0.0,0.0,LBNM)

         RETURN
         END
c*************************************
         SUBROUTINE END_GRAPHICS
         CALL PGEND
         RETURN
         END
```

Simon Habot
Department of Physics
University of Lethbridge
4401 University Drive
Lethbridge
Alberta T1K 3M4

COLIN BATES
2222 - 20th Avenue N.W.
Calgary
Alberta T2M 1J2

Bibliography

[1] M. Bloom, J.H. Davis, and M.I. Valic. *Can.J.Phys.*, **58**:1510, (1980).

[2] R.A. Byrd M. Rance. *J.Magn.Res.*, **52**:221, (1983).

[3] A.J. Shaka, J. Keeler, and R. Freeman. *J.Magn.Res.*, **53**:313, (1983).

[4] C. Counsell, M.H. Levitt, and R.R. Ernst. *J.Magn.Res.*, **63**:133, (1985).

[5] F. Loaiza, M.A. McCoy, and M.H. Levitt. *J.Magn.Res.*, **76**:504–527, (1988).

[6] T. Fujiwara and K. Nagayama. *J.Magn.Res.*, **93**:563, (1991).

[7] W.S. Warren and M.A. McCoy. *J.Magn.Res.*, **98**:24, (1992).

[8] D.J. Siminovitch and N.J. Tagg. *J.Magn.Res.*, **A 108**:82, (1994).

[9] S. Vega and A. Pines. *J.Chem.Phys.*, **no.12 66**:5624, (1977).

[10] D.J. Siminovitch, D.P. Raleigh E.T. Olejniczak, and R.G. Griffin. *J.Chem.Phys*, **84**(5):2556, (1986).

[11] Z. Luz and A.J Vega. *J.Chem.Phys*, **86**(4):1803, (1986).

[12] W. S. Warren. *J.Chem.Phys.*, **81**:5437–5448, (1984).

[13] M.H. Levitt, D. Suter, and R.R. Ernst. *J.Chem.Phys.*, **80** (7):3064, (1984).

[14] M.H. Levitt. *J.Chem.Phys.*, **81**:680, (1984).

[15] I.Solomon. *Phys.Rev.*, **110**no.1:61, (1958).

[16] D. J. Siminovitch and S. Habot. *J. Phys. A: Math. Gen.*, **30**:2577, (1997).

[17] R. Freeman. *Chem.Rev*, **91**:1397, (1991).

[18] S. Habot, D. Lu, N. J. Tagg, G. R. Gall, and D. J. Siminovitch. *J. Solid State NMR*, **10**:137, (1998).

[19] P.T. Narasimhan A. Ramamoorthy. *Paramana J. Phys*, **36**:399, (1991).

[20] P.T. Narasimhan A. Ramamoorthy. *Mole.Phys*, **73**:207, (1991).

[21] W.S. Warren S. McDonald. *Cncpt.Magn.Res*, **3**:55, (1991).

[22] J. Seelig. *Quaterly Review of Biophysics*, **10** (3):353, (1977).

[23] M. Bloom M.I. Valic J.H.Davis, K.R.Jeffrey and T.P.Higgs. *Chem.Phys.Ltr*, **42**:390, (1976).

[24] B. Ewing, S.J. Glaser, and G.P. Drobny. *Chem.Phys.Ltr*, **147**:121, (1990).

[25] L. C. Biedenharn and J. D. Louck. *Angular Momentum in Quantum Physics. Theory and Application.* Addison-Wesley, Reading, 1981.

[26] M. Bouten. *Physica*, **42** :572–80, 1969.

[27] G. Campolieti and B. C. Sanctuary. *J. Chem. Phys.*, **91**:2108–23, 1989.

[28] A. Cayley. *Math. Ann.*, **15**:238–40, 1879.

[29] C. M. Cheng and P.C.W. Fung. *J. Phys. A: Math. Gen.*, **21**:4115–31, 1988.

[30] P. L. Corio. *Structure of High-Resolution NMR Spectra.* Academic Press, New York, 1966.

[31] G. Darboux. *Leçons sur la Théorie Général des Surfaces Vol.I.* Chelsea, Bronx, 1887.

[32] L. P. Eisenhart. *A Treatise on the Differential Geometry of Curves and Surfaces.* Dover, New York, 1960.

[33] D. J. Evans. *Molec. Phys.*, **34** :317–25, 1977.

[34] L. D. Favro. *Phys. Rev.*, **119** :53–62, 1960.

[35] W. G. Harter and N. dos Santos. *Am. J. Phys.*, **46** :251–63, 1978.

[36] P. C. Hughes. *Spacecraft Attitude Dynamics.* Wiley & Sons, New York, 1986.

[37] M. A. Keniry and B. C. Sanctuary. *Chem. Phys. Lett.*, **172** :295–8, 1990.

[38] F. Klein. The Icosahedron, 1884.

[39] G. L. Lamb. *Rev. Mod. Phys.*, **43** :99–124, 1971.

[40] M. H. Levitt. *Prog. NMR Spectrosc.*, **18**:61–122, 1986.

[41] Z. A. Melzak. *Mathematical Ideas, Modeling and Applications.* Wiley & Sons, New York, 1976.

[42] C. W. Misner, K. S. Thorne, and J. A. Wheeler. *Gravitation.* W.H.Freeman, San Francisco, 1973.

[43] V. S. Popov. *Sov. Phys. JETP (Engl. Transl.)*, **35**:687–9, 1959.

[44] O. Rodrigues. *J. de Mathématiques Pures et Appliquées*, **5**:380–440, 1840.

[45] M. A. M. Santiago and A. N. Vaidya. *J. Phys. A: Math. Gen.*, **9**:897–904, 1976.

[46] M. D. Shuster. *J. Astronaut.*, **41**:439–517, 1993.

[47] D. J. Siminovitch. *J. Magn. Reson. A*, **117**:235–45, 1995.

[48] J. Sivardière. *Am. J. Phys.*, **62**:737–43, 1994.

[49] J. Stuelpnagel. *SIAM Rev.*, **6**:422–30, 1964.

[50] D. R. Truax. *Phys. Rev. D*, **31**:1988–91, 1985.

[51] R. M. Wilcox. *J. Math. Phys.*, **8**:962–82, 1967.

[52] H. Gao J. Zhou and B. C. Sanctuary. *J. Magn. Reson. A*, **101**:119–21, 1993.

[53] C. Ye J. Zhou and B. C. Sanctuary. *J. Chem. Phys.*, **101**:6424–9, 1994.

[54] Altmann S. *Rotations, Quaternions and Double Groups.* Clarendon Press, Oxford, 1986.

[55] C. A. Fyfe. *Solid State NMR for Chemists.* C.F.C Press, Guelph, 1983.