Implementation of Wavelet Encoding Spectroscopic Imaging Technique on a 3 Tesla Whole Body MR Scanner

by

Yao Fu

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfilment of the requirements of the degree of

Master of Science

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg, Manitoba, Canada

I hereby declare that I am the sole author of this thesis.

Abstract

Magnetic Resonance Imaging (MRI) uses a magnetic field and low energy radio waves to visualize the internal structure and function of the body. This is one of the most popular technologies currently used for diagnostic imaging. Magnetic Resonance Spectroscopy Imaging (MRSI), complementing MRI, provides a chemical map of the scanned region by providing spatial information about tissue metabolite concentrations. MRSI is being used for early diagnostics to differentiate diseased tissue from normal tissue. However, obtaining metabolic maps with high spatial resolution requires long acquisition times where the patient has to lie still inside the magnet bore (scanner) especially if classical Chemical Shift Imaging (CSI) is used. The need for acquisition time reduction is encountered in many practical applications. In this dissertation, a 3D wavelet based encoding spectroscopic method (WE-SI) is investigated and implemented on a 3 Tesla Siemens Scanner. Compared to CSI, the proposed method is able to reduce acquisition time, and preserves the spatial metabolite distribution. As expected, a decrease in Signal to Noise Ratio (SNR) is noticed in WE-SI data compared to CSI. The dissertation explores important physical principles in MRI and spectroscopic imaging as a background, following by introduction of the wavelet encoding theory and comparison to Fourier encoding. In chapter 3, the implementation of WE-SI on a 3T scanner is detailed. In-vitro and in-vivo results are displayed and discussed in chapter 4, followed by conclusion.

Acknowledgements

I would like to thank my co-supervisors: Dr. Hacene Serrai, Dr. Gabriel Thomas and Dr. Reza Fazel for their guidance, support and encouragement throughout the course of my studies and preparing this thesis.

I also would like to thank Dr. Arkady Major and Dr. Krisztina Malisza for serving on my examining committee, carefully reading my thesis and providing me with helpful feedback and suggestions.

I would like to acknowledge the financial support from my co-supervisor Dr. Hacene Serrai (NSERC-Discovery Grant) and Conference expense support from Department of Electrical and Computer Engineering, University of Manitoba. All support is gratefully acknowledged and appreciated.

Many thanks go to the stuff at National Research Council, Institute for Biodiagnostics. They are very patient and helpful, in particular to Omkar Ijare, Patricia Gervai and Ernie Packulak.

Contents

Chapter 1

Introduction to

Magnetic Resonance Imaging and

Spectroscopic Imaging

Magnetic resonance imaging (MRI) is an imaging technique that has been widely used for diagnosis, characterization, and treatment planning and assessment [1-4]. It is a noninvasive method and has the ability to image a wide variety of soft tissues, making it possible for application to most portions of the population. Magnetic resonance spectroscopic imaging (MRSI) is being increasingly used in situations where MR imaging cannot give a definite diagnosis. MRSI may provide early prognostic information, aid in understanding brain development and metabolism; differentiate between diseased and normal tissue, improve treatment, and reduce risk to the patient. The acquisition of metabolic information from multiple imaged regions often involves long acquisition times, particularly when using classical Chemical Shift Imaging (CSI) [5-7]. At the same time, patients must lie still during the exam to avoid motion artifacts, which is difficult with a long acquisition period.

To reduce acquisition time, several Fourier based approaches using modified high speed imaging sequences have been proposed. Techniques such as Echo Planar Imaging (EPI) [8] and spiral imaging [9] provide metabolite information from different brain regions with high spatial resolution. They are generally limited to two spatial dimensions

and have reduced metabolite spectral resolution. Recently, a non-Fourier encoding MRSI technique based upon wavelet encoding-spectroscopic imaging (WE-SI) was proposed to reduce acquisition time [10-11]. WE-SI utilizes selective radio-frequency (RF) pulses with profiles resembling the shape of wavelets, to sequentially excite a set of predetermined regions of the brain (sub-spaces) of different sizes and locations without the need for full recovery time (*TR*), necessary for spin relaxation, between excitations. In-vivo results on a 1.5T whole body scanner show that WE-SI provides data with high spectral resolution in multiple dimensions and reduces acquisition time relative to CSI.

The increment in magnetic field sensitivity is important as it provides higher signal sensitivity. However, field inhomogeneity increases with higher field strength, which is quite challenging especially for the wavelet encoding technique because the encoding is based on amplitude modulation. To prove the feasibility of WE-SI at a higher field strength and comparing with previously developed 1.5 T WE-SI technique, we implemented WE-SI on a 3 tesla clinical scanner equipped with 32 receive channels. Our goal was to improve data sensitivity and increase the potential of WE-SI by completing its implementation at a higher magnetic field. The fact that the work in this dissertation is accomplished at a completely different platform with higher field strength, requires development of new radio frequency (RF) pulses and the related pulse sequence. Further more, a new ordering algorithm was also developed to automatically select acquisition order for a minimum total acquisition time, and as a result the corresponding reconstruction code was freshly written as well.

Over the years, MRI engineers put considerable efforts into improving hardware, software, and acquisition technologies to increase signal quality and speed up the

acquisition. MRI uses a strong magnetic field to force the nuclear magnetization of hydrogen protons (mostly) to align to the direction of the magnetic field. Radio frequency pulses are used to tip the spins into transverse plane for the scanner to pick up a signal. Depending on the different densities of protons and relaxation parameters of different tissues, different image intensities are obtained, hence an image can be formed.

1.1 Proton Spins with the Magnetic Field

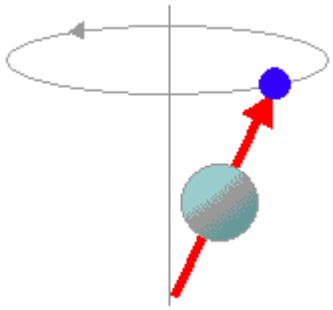


Fig. 1.1 Proton spinning at equilibrium state.

Under an external magnetic field, B_0, protons spin about the field direction. The precession angular frequency for the proton magnetic moment vector is given by

$$\omega_0 = \gamma B_0 \tag{1.1}$$

where γ is the gyromagnetic ratio (or magnetogyric ratio) and is tissue related [12]. In water, the hydrogen proton has a γ value around 2.68×10^8 rad/s/tesla. At a field of 3 tesla, spins processes at radiofrequency of 123.3MHz. This is the Larmor Frequency and equation 1.1 is called the Larmor equation.

As shown in Fig. 1.1, in the equilibrium state the positive charge on the proton interacts with the magnetic field and produces a torque,

$$\overline{N} = \overline{M} \times \overline{B} \tag{1.2}$$

where $\overline{M}$ is the magnetic dipole moment or magnetic moment. The bar symbol represents a vector.

This torque causes protons to rotate about the direction of B_0 with angular momentum $\overline{J}$.

$$\frac{d\overline{J}}{dt} = \overline{N} \tag{1.3}$$

The direct relationship between the magnetic moment and the spin angular momentum vector is found by experiment:

$$\overline{M} = \gamma \cdot \overline{J} \tag{1.4}$$

1.2 Laboratory Frames and Rotating Frames

The laboratory reference frame is a fixed frame which is represented by unprimed x, y and z. It describes the physical dimension of the subject, eg, left-right, head-foot, and anterior-posterior directions.

Another frame (denoted by x', y' and z') is rotating about an arbitrary axis with respect to the fixed frame because of the existence of a magnetic field. In this frame, the rotating vector B_1, known as the radio frequency (RF) field, appears to be stationary.

1.3 Magnetization, Relaxation and the Bloch Equation

From equation 1.2, 1.3 and 1.4, the following differential equation can be derived:

$$\frac{d\overline{M}}{dt} = \gamma \cdot \overline{M} \times \overline{B} \tag{1.5}$$

which is called the "fundamental equation of motion". It is most advantageous to analyze the magnetization and its differential equation in terms of parallel and perpendicular

(with respect to the B_0 field) components defined relative to the static main magnet field. Let $B_{ext} = B_0 \bar{z}$, then

$$M_{\parallel} = M_z \tag{1.6}$$

$$M_{\perp} = M_x \bar{x} + M_y \bar{y} \tag{1.7}$$

$$\frac{dM_z}{dt} = 0 \tag{1.8}$$

$$\frac{dM_{\perp}}{dt} = \gamma M_z \times \overline{B_{ext}} \tag{1.9}$$

The above equations are derived for the equilibrium state, in which case, the protons are not interacting with each other. However, with interacting protons, the protons try to align with the external field through the exchange of energy with the surroundings to achieve the minimum potential energy with magnetization M_0 [20]. The overall trajectory of the tip of the net magnetization vector is shown in Fig. 1.2.

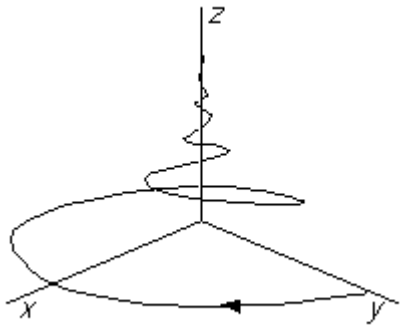


Fig. 1.2 The trajectory of the tip of net magnetization

In the transverse plane, $\overline{M}_{\perp}$ is maximum when spins are tipped into the transverse plane (x,y plane). A loss of energy to the surrounding nuclei causes the return of the excited nuclei from the high energy to the low energy states. This return is an exponential process characterized by:

$$\frac{dM_z}{dt} = \frac{1}{T_1}(M_0 - M_z) \tag{1.10}$$

where T_1 is the experimental 'spin-lattice relaxation time', and is different from tissue to tissue.

Solving 1.10, we have,

$$M_z(t) = M_z(0)e^{-t/T1} + M_0(1 - e^{-t/T1}) \tag{1.11}$$

A loss in transverse magnetization is due to spins in the high and low energy states exchanging energy, but without losing energy to the surrounding lattice. The rate of change is characterized by:

$$\frac{d\overline{M}_\perp}{dt} = \gamma\overline{M}_\perp \times \overline{B}_{ext} - \frac{1}{T_2}\overline{M}_\perp \tag{1.12}$$

T_2 is shorter than T_1 because of the dephasing effect. Combining differential equations in both the longitudinal direction and transverse plane into one vector equation, we obtain the so called Bloch equation[12],

$$\frac{d\overline{M}}{dt} = \gamma\overline{M} \times \overline{B}_{ext} + \frac{1}{T_1}(M_0 - M_z)\overline{z} - \frac{1}{T_2}\overline{M}_\perp \tag{1.13}$$

The complete solution is therefore,

$$M_x(t) = e^{-t/T_2}(M_x(0)\cos\omega_0 t + M_y(0)\sin\omega_0 t) \tag{1.14}$$

$$M_y(t) = e^{-t/T_2}(M_y(0)\cos\omega_0 t - M_x(0)\sin\omega_0 t) \tag{1.15}$$

$$M_z(t) = M_z e^{-t/T_1} + M_0(1 - e^{-t/T_1}) \tag{1.16}$$

1.4 Radio Frequency (RF) Pulses

A B_1 field applied on-resonance for a finite time is called an 'RF pulse' [12]. By adding an RF field B_1 which is at rest in the rotating frame and parallel to x' to the static field, we are able to tip $\overline{M}$ from its equilibrium position. The total external field is

$$\overline{B}_{ext} = B_0\bar{z} + B_1\bar{x}' \tag{1.17}$$

Suppose spins are rotating with frequency ω in the laboratory frame, the rotating spins will generate a magnetic field $-\omega/\gamma$. Therefore, the effective field in the rotating frame is:

$$\overline{B}_{eff} = (B_0 - \frac{\omega}{\gamma})\bar{z} + B_1\bar{x}' \tag{1.18}$$

For most cases, the RF pulse duration is much smaller than the decay of T_1 and T_2. Therefore, spin relaxation with short RF in the rotating frame is similar to the static field described above:

$$M_{x'} = e^{-t/T_2}(M_{x'}(0)\cos\Delta\omega t + M_{y'}(0)\sin\Delta\omega t) \tag{1.19}$$

$$M_{y'} = e^{-t/T_2}(M_{y'}(0)\cos\Delta\omega t - M_{x'}(0)\sin\Delta\omega t) \tag{1.20}$$

$$M_z = M_z(0)e^{-t/T_1} + M_0(1-e^{-t/T_1}) \tag{1.21}$$

with $\Delta\omega$ representing possible deviations from ideal conditions due to static field impurities or variations in the applied RF frequencies.

During the short time (τ) of B_1, we can assume a close to on-resonant condition. Under this assumption, $\Delta\omega$ is much smaller than ω_1 and can be ignored. Therefore, spins are seen only experiencing precession around the x axis with angular frequency ω_1. Therefore, the flip angle induced by the RF field is given by:

$$\Delta\theta = \omega_1 \tau = \gamma B_1 \tau \tag{1.22}$$

1.5 Signal Detection

MR signals are detected by coils according to Faraday's law of electromagnetic induction. A single magnetic moment is analogous to a bar magnet rotating about the z direction. As the magnetic moment rotates, the flux change is picked up by the nearby coils. An electromagnetic force (emf) is generated:

$$emf = -\frac{d\phi}{dt} \tag{1.23}$$

where

$$\phi = \int_{coilarea} \overline{B} d\overline{s} \tag{1.24}$$

The emf induced in the MRI coil is expressed as:

$$emf = -\frac{d}{dt}\phi_M(t) = -\frac{d}{dt}\int_{samples} d^3 r \overline{M}(\overline{r},t) \cdot \overline{B}^{receive}(\overline{r}) \tag{1.25}$$

where $\phi_M(t)$ is the flux, $\overline{B}^{receive}(\overline{r})$ is the coil field sensitivity.

The signal detected by a coil is proportional to the induced emf and expressed as:

$$s(t) \propto \omega_0 \int d^3 r M_\perp(\overline{r},t) B_\perp^*(\overline{r}) \tag{1.26}$$

If we assume the transmit and receive coils produce homogenous fields over the image volume, we have,

$$s(t) \propto \omega_0 B_\perp \int d^3 r \rho(r,t) e^{i\theta_B} \tag{1.27}$$

1.6 Fourier Imaging and Acquisition Method

If we reduce equation 1.27 to a one dimensional case and assume that the transmit coil produces a homogenous field over the image volume, we can write:

$$s(t) = \int dz \rho(z) e^{i\phi_G(z,t)} \tag{1.28}$$

Notice that 1.28 is in the form of the Fourier transform if,

$$\phi_G(z,t) = -2\pi zt \tag{1.29}$$

This equation together with equation 1.24 over a fixed coil area introduce the need for a linearly varying field added to the static field,

$$B_z(z,t) = B_0 + zG(t) \tag{1.30}$$

Equation 1.30 introduces another very important component in MR, namely *G*, called gradient strength. It provides a linear variation added to the homogenous magnetic field B_0. The gradients associated the phase term in equation 1.28 with three dimensional position, which converted equation 1.28 into Fourier transform form. The acquired signals are in the Fourier domain, which is called the K space. Low frequency components are located in the center of this K space, and high frequency components are at the edges.

The extension of one-dimensional imaging to all three dimensions can be written as:

$$s(k_x, k_y, k_z) = \iiint dxdydz \rho(x, y, z) e^{-i2\pi(k_x x + k_y y + k_z z)} \tag{1.31}$$

1.7 MR Spectroscopy

MR spectroscopy provides an encoding of measurable contributions from different metabolites including water, fat, choline, creatine, n-acetyl aspartate and lactate by identifying certain molecular constituents. It is a technique to provide spatial metabolite information.

Protons in different molecules experience a different magnetic shielding effect due to their chemical environments [12]. The Larmor frequency for specific metabolite:

$$w_{oi} = \gamma_i B_0 \tag{1.32}$$

where γ_i is called gyro-magnetic ratio and is constant for each isotope. If we use a broadband transmit RF pulse to excite wide range of frequencies, we would obtain a time signal containing information about the set of all nuclei in the sample. The Free Induction Decay (FID) is a signal in time domain collected by the receiver coil [12]. Normally a Fast Fourier Transform (FFT) is used to convert FID into a spectrum to analyze metabolic peaks.

The small and measurable frequency shift from the Larmor frequency due to magnetic shielding by local environment is called 'chemical shift' σ [12].

$$B_{shifted}(j) = (1-\sigma_j)B_0 \tag{1.33}$$

The chemical shift spectrum is expressed as the fractional shift in parts-per-million (ppm) of the NMR frequency relative to an arbitrary reference compound. Table 1.1 shows common observable proton metabolites with ppm values is given as:

TABLE 1.1

OBSERVABLE PROTON METABOLITES

ppm	Metaboite	Properties
0.9-1.4	Lipids	Products of brain destruction
1.3	Lactate	Product of anaerobic glycolysis
2.0	NAA	Neuronal marker
2.2-2.4	Glutamine/GABA	Neurotransmitters
3.0	Creatine	Energy metabolism
3.2	Choline	Cell membrane marker
3.5	myo-inositol	Glial cell marker, osmolyte hormone receptor mechanisms

1.8 Chemical Shift Imaging (CSI)

Chemical shift imaging (CSI) is a sequence used to record the spectroscopic data for a group of voxels in two or three dimensions. It uses phase information to encode the position by the Fourier encoding technique [5, 7]. A set of FIDs are collected at every k space point. The sequence for three dimensional CSI is shown in Fig. 1.3. Phase encoding is introduced to cover the k-space by varying the gradient amplitudes according to equation 1.31 .

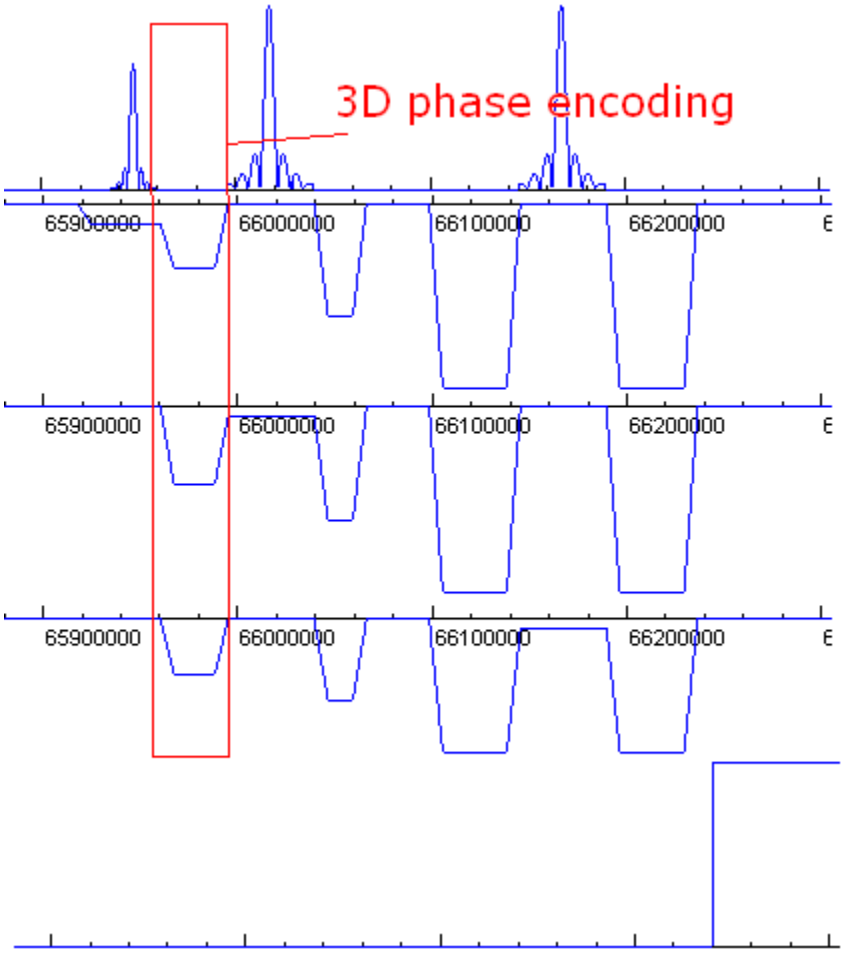


Fig. 1.3 CSI sequence

1.9 CSI Limits

We can only collect finite discrete points in K space in Fourier Imaging. This implies two potential problems common with Fourier Imaging: (1) Aliasing caused by under sampling, and (2) windowing effects because of finite sampling.

1.9.1 Aliasing

Because the K space data is discrete, when converting the K space data to the spatial domain by Fourier transformation, the result is spatial periodic with period L as shown in Fig. 1.4. If L is smaller than the field of view (FOV) A, aliasing occurs and the resultant image wraps around along the under sampled direction (Fig. 1.5). This is also called the folding effect. According to the Nyquist sampling criterion [13], the sampling frequency has to be at least twice the highest frequency：

$$L \geq 2 \times \frac{A}{2} = A \tag{1.34}$$

This means the sampling step in k space must be:

$$\Delta k = \frac{1}{L} \leq \frac{1}{A} \tag{1.35}$$

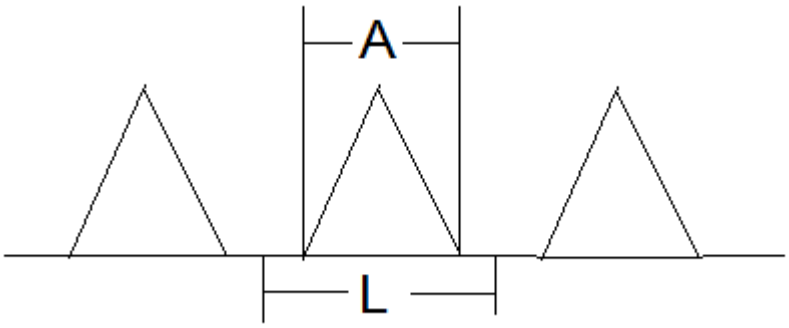


Fig. 1.4 An object with FOV A with period L due to discrete sampling. [12]

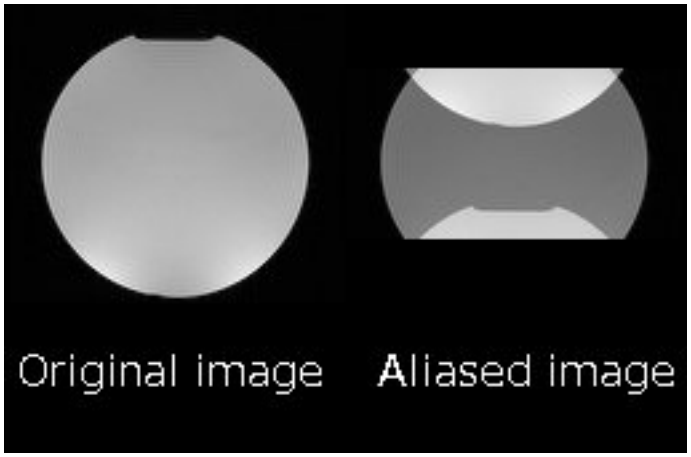


Fig. 1.5 Effect of aliasing on a MR phantom image.

(1) Suppose we have a boxcar gradient G_f applied on the readout direction, then Δk is defined as:

$$\Delta k = \gamma G_f \Delta t = \frac{1}{L} \leq \frac{1}{A} \tag{1.36}$$

Therefore the acquisition bandwidth has to be:

$$BW \equiv f = \frac{1}{\Delta t} > \gamma G_f A \tag{1.37}$$

(2) Suppose we have a boxcar gradient G_p applied on the phase encoding direction, then Δk is defined as:

$$\Delta k = \gamma G_p \tau_p \tag{1.38}$$

where τ_p is the duration of G_p. From equation 1.35 and 1.38 we have the requirement for phase encoding direction:

$$\tau_p < \frac{1}{\gamma G_p A} \tag{1.39}$$

Situation 2 is responsible for CSI aliasing problems as CSI employs phase encoding in all three Cartesian dimensions. In Seimens' implementation, the smallest gradient strength is calculated by the equality case of equation 1.36:

$$G_{step} = \frac{1}{\gamma A \tau_p} \tag{1.40}$$

This ensures that the aliasing problem is avoided if G_{step} is within the hardware limitation with the desired FOV and resolution. If we assume the center of the FOV is placed at the isocenter of the scanner, so there is no frequency shift for all CSI acquisition steps, and assume that the k space data is uniformly distributed, then the maximum gradient strength required for Fourier encoding is calculated as:

$$G_{\max} = \frac{n}{2} G_{step} \le G_{\max\_allowed} \tag{1.41}$$

Where n is the desired resolution, and

$G_{max\_allowed}$ is the maximum capable gradient that is defined by hardware. If G_{max} exceeds $G_{max\_allowed}$, the gradient step is forced to drop, and aliasing occurs.

In the Siemens' 3T system, the Larmor constant γ is 42.5756 MHz/T, and the maximum capable gradient is 11.547mT/m. Gradient duration is defined as 2800 μs. Therefore equation 1.40 and equation 1.41 require that:

$$\frac{n}{A} \ge \frac{2 \times 11.547 \times 42.5756 \times 2800}{10^6} = 2.7531 \tag{1.42}$$

From equation 1.42 we can see that low resolution with large FOV would likely cause aliasing problems. In cases of aliasing, intuitively we would see metabolite contaminations across the FOV boundary.

A longer gradient duration will delay the echo, yielding a lower signal to noise ratio. A higher capable maximum gradient requires extensive hardware modification.

1.9.2 Windowing Effect

The K space data we collect from the scanner has finite length, which can be mathematically modeled by multiplying the sampled data by a rect function [13]. Then, when we convert it to the spatial domain, this is equivalent to a convolution of the original data with a sinc function. All images include this effect, but a wide rect function gives a narrower sinc function, and therefore can provide negligible bluring. For the same reason, there is a lower limit for spatial resolution.

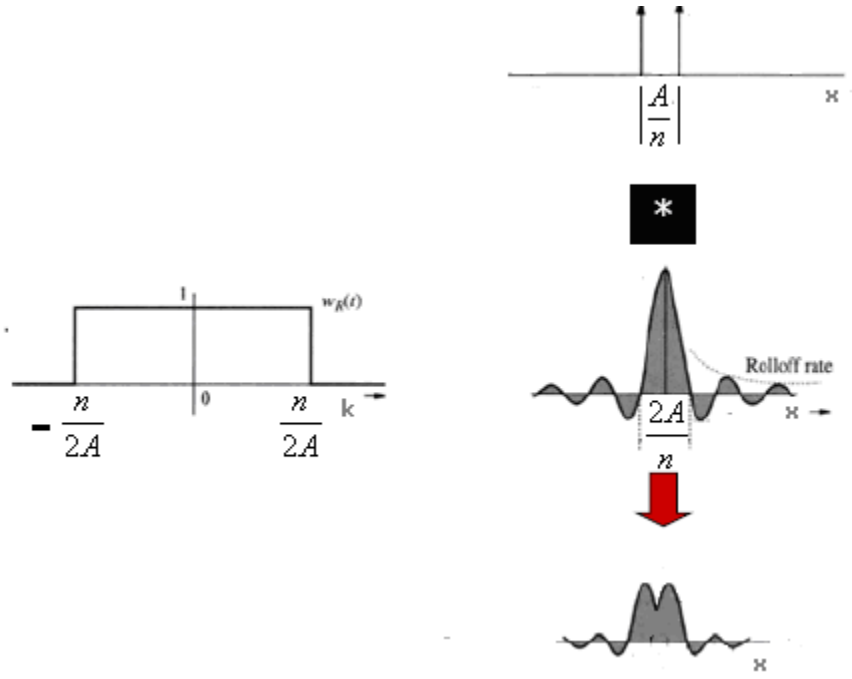


Fig. 1.6 Windowing effect on MR data.

As $\Delta k = \frac{1}{FOV} = \frac{1}{A}$, suppose resolution is *n*, then the rectangle function in K space has a length $\frac{n}{2A}$. The corresponding sinc function in the spatial domain has a main lobe with width equal to $\frac{A}{2n}$. As shown in Fig. 1.6, it is clear that no matter how large the resolution is, the main lobe of the sinc function will exceed the local region defined by a pixel. The contamination caused by the main lobe will only affect the pixels right next to it. The existence of side lobes will produce positive or negative contaminations as well. As the distance between two pixels becomes larger, the level of contamination decreases rapidly.

We can intuitively reduce contamination by increasing resolution. For instance from Fig. 1.6, if we keep the distance between two metabolites to be $\frac{A}{n}$, but increase the resolution to 2*n*, there will be very little contamination. This amount of contamination can be ignored.

1.9.3 Imperfection RF Pulse Profiles

Finite sample points are used to represent the shape of RF pulses. Tails exist on both sides beyond the desired bandwidth. Therefore, substances outside the FOV excited by those tails will be folded into the boundary of the other side. To overcome this problem, usually the FOV of a CSI sequence is set larger than the View Of Interest (VOI). However, by using this method, if we want to keep the voxel size inside the VOI, we need to acquire data at a higher resolution over the entire FOV. As a consequence, acquisition time is increased.

1.9.4 Acquisition Time

CSI uses 3D Fourier phase encoding to encode spatial location. For every encoding step, the whole VOI is excited. Therefore, recovery time (*TR*) is needed for spin relaxation. To acquire a 3D spectroscopic image with resolution n_x, n_y and n_z, the total acquisition time is:

$$T_{total} = TR \times n_x \times n_y \times n_z \tag{1.43}$$

1.10 Other MR Spectroscopic Imaging Techniques

Other spectroscopic imaging methods have been proposed to address some of the problems associated with CSI. These methods include Hadamard spectroscopic imaging (SI) [14] [15], high speed imaging techniques such as EPI and spiral [16], fast spin echo (FSE) [17], Steady State Free Precession (SSFP) and echo-shift methods. For a variety of reasons, each of these methods has only been partially successful in improving the original Fourier-based MRSI technique in terms of overall qualitative and quantitative results.

The Hadamard SI method uses RF pulse modulation in the presence of gradients and manipulates the sign of the acquired MR signal according to the Hadamard matrix to obtain spatial encoding. One major advantage of this approach is the ability to provide metabolite images at low spatial resolution with less cross voxel contamination and high spectral resolution. The technique requires high RF peak powers as the number of voxels increases in order to maintain the low cross voxel contamination.

High speed imaging methods such as EPI [8], Line Scan Echo Planar SI (LSPEPSI) [18] or spiral are very rapid and offer high spatial resolution. However, as they use the readout gradients to collect both spatial and spectral points, they require complicated reconstruction methods to differentiate between these dimensions. In addition, there is a trade-off between spatial and spectral resolution. The point spread functions in the spatial and spectral dimensions can lead to a spread of signal over both dimensions [19]. Furthermore, the SNR tends to decrease compared to the CSI method due to the use of high receiver bandwidths [20].

FSE imaging methods allow imaging of a single chemical species, such as phosphocreatine (PCr), and have been employed to examine human muscle and brain [21]-[23] opriate echo spacing between the refocusing RF pulses is introduced to dephase unwanted spins while fulfilling the CPMG (Carr, Purcell, Meiboom and Gill) condition for the desired spins allowing their signals to be observed. Unfortunately only a limited number of species can be imaged using these methods, with signals that are within a narrow frequency range and are thus difficult to separate.

Echo-shifting methods were introduced in an effort to reduce acquisition time and increase spatial resolution [24]. By time shifting either the readout gradient or the RF pulses, both the spectral dimension and one spatial dimension are encoded. Phase encoding is then used in the remaining spatial dimension. In this case, spectral resolution is sacrificed for shorter acquisition time [25].

Chapter 2

Theory of WESI

As discussed in section 1.9, the CSI technique can provide metabolite images with good SNR and high spectral and spatial resolution at the cost of long acquisition times that result from two factors:

1) The spin-lattice time, T_1 of the metabolites is long (1-2 seconds for proton and 4-6 seconds for phosphorus MR), requiring a long recovery time (*TR*) between phase encoding steps to allow for T_1 relaxation. For a 3D spectroscopic image with spatial resolution n_x, n_y and n_z, the total acquisition time is:

$$T_{total} = n_x \times n_y \times n_z \times Averages \qquad (2.1)$$

As an example, an 8 by 8 by 4 metabolite map with TR = 2 sec using CSI requires at least 9 minutes. To reduce acquisition time, several Fourier-based approaches have been proposed. These Fourier based techniques provide metabolite information with high spatial resolution but are limited in spectral resolution and spatial dimensions. Therefore, obtaining a high spatial resolution metabolic map using CSI requires long acquisition time where the patient has to lie still inside the magnet.

2) A large number of phase encoding steps are required in each spatial dimension to avoid image reconstruction artifacts such as pixel bleed. To utilize CSI clinically, spatial resolution is often sacrificed and *TR* times that are short with respect to T_1 are used, increasing the difficulty of obtaining accurate metabolite concentrations. If the FOV is large, a folding effect results in large contamination at all boundaries since all

three CSI dimensions are phase encoded. If the resolution is low, pixel bleed decreases the data accuracy.

Similar to Fourier encoding that uses acquired k-space data, wavelet encoding uses signals acquired from predetermined sub-spaces to fill the wavelet domain (Fig. 2.1). By replacing the Fourier transform by the wavelet (Haar wavelet) transform, pixel bleed and total acquisition time are reduced with some sacrifice in SNR.

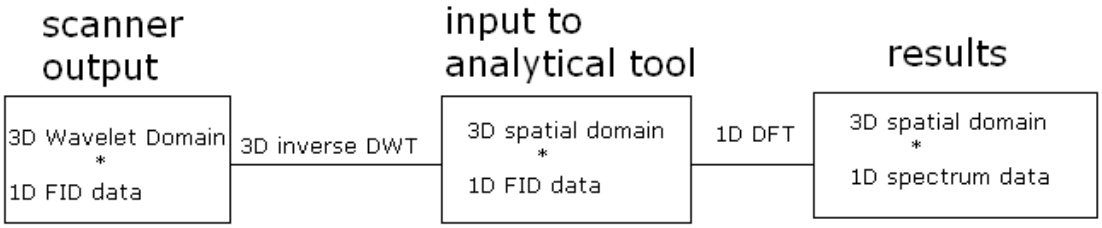


Fig. 2.1 Data processing procedures

2.1 Haar Wavelet and Its Implementation on The MR Scanner

Using this orthogonal basis, the wavelet transform is defined as [26]:

$$F(j,k) = \int \varphi_{j,k}(x) f(x) dx \tag{2.2}$$

As with the Fourier transform, the Inverse wavelet transform can be obtained by [26]:

$$f(x) = \sum_{j,k} \varphi_{j,k}(x) F(j,k) \tag{2.3}$$

The Haar wavelet we are using is defined by:

$$\varphi_{j,k}(x) = \begin{cases} 1 & \text{if } k/2^{j-1} < x < (k+\frac{1}{2})/2^{j-1} \\ -1 & \text{if } (k+\frac{1}{2})/2^{j-1} \le x < (k+1)/2^{j-1} \\ 0 & \text{if } \textit{otherwise} \end{cases} \tag{2.4}$$

For j=1,2,3,…log2N-1, k= 0, 1, …2^{j-1}-1

and,

$$\varphi_{j,k}(x) = 1 \tag{2.5}$$

for j=0

It is clear that the Haar wavelet is a sequence of orthogonal functions that can be used to approximate any continuous real function.

In wavelet encoding, a set of dilated and translated prototype functions called wavelets are used to span a localized space by dividing it into a set of sub-spaces with pre-determined sizes and locations. As shown in equation 2.2, the wavelet transform is a collection of inner products between the original function and the Haar basis. In spectroscopic imaging, this process is achieved using RF pulses with profiles resembling the wavelet shapes [10, 11]. Slice selective excitation and refocusing RF pulses, with single and dual band profiles similar to Haar wavelets, are used in the modified point resolved sequence (PRESS) [27] to acquire three dimensional (3D) wavelet encoding spectroscopic imaging (WE-SI) data. The magnitude of both single and dual RF pulses are set at unity (equation 2.2). The desired spatial resolution in each direction sets the corresponding number of dilations (increases in the localization gradients), and consequently the number of translations (frequency shift) of the Haar wavelets (RF pulses), which are used to collect MR signals from the corresponding sub-spaces [10] [11].

The bandwidths of all RF pulses are set to a fixed value, BW_{rf}. The dilatation of the Haar wavelet is implemented by changing the gradient strength.

$$x_j = \frac{BW_{rf}}{G_x \cdot 2^{j-1}} \tag{2.6}$$

$$G_x = \frac{BW_{rf}}{L} \tag{2.7}$$

Where L is the FOV along x, G_x is the gradient strength needed when the whole FOV is excited.

The translation of the Haar Wavelet is implemented by shifting the center frequency of the RF pulse. When j=0 or 1, the Haar wavelet is centered at the middle of the FOV; otherwise, it is shifted. For encoding steps with the same dilation value j but different translations (k and k+1), a frequency shift with the same value of RF bandwidth is required.

We can simply extend equation 2.2 and 2.3 into three orthogonal spatial dimensions:

$$F(j_1, j_2, j_3, k_1, k_2, k_3) = \iiint \varphi_{j_1,k_1}(x)\varphi_{j_2,k_2}(y)\varphi_{j_3,k_3}(z) f(x,y,z)dxdydz \tag{2.8}$$

$$f(x,y,z) = \sum_{j,k} \varphi_{j1,k1}(x)\varphi_{j2,k2}(y)\varphi_{j3,k3}(z)F(j_1, j_2, j_3, k_1, k_2, k_3) \tag{2.9}$$

2.2 Pulse Sequence Design

As previously mentioned, a WE-SI sequence is generated from a PRESS sequence using RF pulses with profiles resembling Haar functions (single or dual boxcar shape).

One excitation (90º) RF pulses and two refocusing (180º) RF pulses are applied together with three slice selective gradients. RF pulses also can be seen as band filters, which can flip spins about the precession direction if they are rotating within the selective frequency range. By selecting the axis of precession and RF duration, different flip angles (eg. 90º and 180º in our application) can be obtained. Equation 1.31 shows that protons spin at linearly varying frequencies with the presence of gradients. Therefore, the required bandwidth of the RF pulses is:

$$BW \equiv \Delta f = \gamma G_s \Delta z \tag{2.10}$$

According to the wavelet transform, a single RF pulse is only applied when j=0, otherwise a dual RF pulse is applied. An internal loop is added in order to accomplish the dilation and translation for individual encoding steps through changing the gradient strength and center frequency of the RF pulse correspondingly (fSeqRun() in appendix A). Fig. 2.2 shows one step of WE-SI encoding (j_x=2, j_y=0, j_z=0, k_x=0, k_y=0, k_z=0).

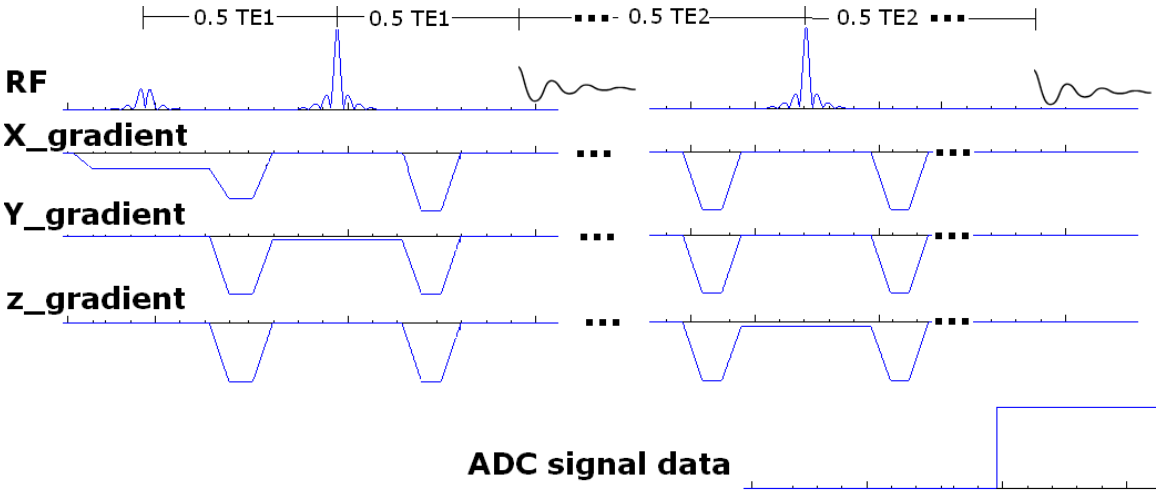


Fig. 2.2: WE-SI pulse sequence design (at encoding step j_x=2, k_x=0; j_y=0, k_y=0; j_z=0, k_z=0).

At this encoding step, the slice selective (90°) RF pulse is on the x axis. Since j_x=2, a dual band 90° RF pulse is required. The gradient strength is calculated by equation 1.4. The first refocusing (180°) RF pulse is selective on Y at j_y=0. Therefore a single band RF pulse is applied. The second refocusing (180°) RF pulse is selective on Z at j_z=0. Therefore a single band RF pulse is applied with the gradient on z-direction.

The 90° RF pulse is also called the excitation pulse, which flips spins within the selected frequency into the transverse plane. It is assumed that the spins are tipped instantaneously into the transverse plane at the center of the RF pulse, starting when spins start dephasing about the x direction for half of the RF pulse duration due to the existence of the x_gradient. In order to rephase spins, we need to apply $G_{rephase}$ with a reverse sign [12]:

$$\left|\frac{\int dtG_{rephase}}{\int dtG_s}\right| = \left|\frac{totalAreaUnderG_{rephase}}{totalAreaUnderG_s}\right| = 50\% \qquad (2.11)$$

There are twelve other gradients applied on all three dimensions with higher magnitude (Fig. 2.2). They are called spoilers, whose function is to rapidly dephase the spins outside the desired region in order to localize the signal. Due to the existence of spoilers, spins rotate at different frequencies depending on their 3D position, over time a phase difference is created and the received signal drops. At a certain time (half TE_1 or half TE_2 as shown in Fig. 2.2), selective spins experience a 180º angle flip, which reverses the phase difference and spins recover to being back in phase in another half TE_1 or half TE_2. For a better explanation, we suppose there are only two spins, one is rotating

faster than the other one. At t=0, they are both flipped into the transverse plane and are inphase. After half TE the faster spin is ahead of the slower spin by $\Delta\theta$. Then both spins experience the 180º RF pulse and the phase difference between the faster and the slower becomes $-\Delta\theta$. Since the faster spin still rotates at a higher frequency, in another half TE, the two are in phase for a second time. Therefore, at t=TE, we obtain a phased signal called an echo. However, for those spins which didn't see the 180º RF pulse, they see a total of $2\Delta\theta$ and since this $\Delta\theta$ is actually randomly distributed between 0 and 2π, the complex signal is cancelled out.

The echo time (TE) is the time when the spins in the selected region rephased again after the second 180º RF pulse. MR signal is collected at t=TE. As spins are rotating while relaxing, the received signal is in complex form.

Hence, to ensure the rephasing process, we need: (1) the area negative lobe in is half of the area of the positive lobe in x direction. (2) in all three dimensions, the areas of the spoiler lobes before and after 180º RF pulses must be equal. In Fig. 2.2, we see the first spoiler in X direction is smaller than others. This is the result of superposition of the refocusing gradient and the spoiler.

To better illustrate how the RF pulses flip spins in order to select the desired region, we show an example with $j_x=2$ $j_y=2$ and $j_z=2$. Since dilations in all three dimensions are non zero, all RF pulses have dual band profile.

Step 1: Apply a selective 90º RF pulse with a selective gradient on the x axis:

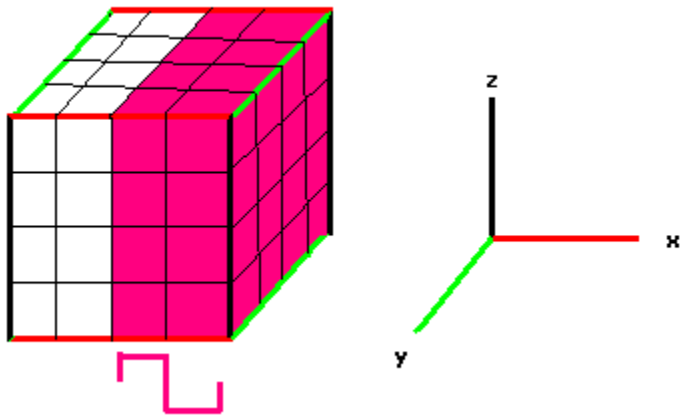


Fig. 2.3 A 90º excitation pulse applied along x with j_x=2.

Half of the spins (color pink) are flipped into the transverse plane; the other half (white) are left untouched and spins rotates around spatial axis z. Suppose the RF pulse is applied along +x' in the rotating frame, during the short duration of B_1 field, spins rotates about +x' axis by either +90º or -90º depending on the position along x. Right after the B1 field, the left part of the pink region has spins lying along +y' and the right part of pink region has spins lying along –y' in the rotating frame.

Step 2: Apply the first selective180º RF pulse on y

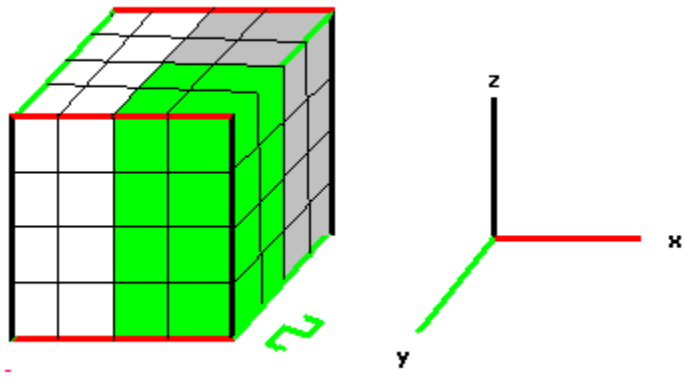


Fig. 2.4 A first 180º reforcosing pulse applied along y with j_y=2.

The first 180º RF pulse flips the spins in the front part of the cube by 180º in the rotating frame about x'. Hence, spins that were rotating around z (color white) will remain rotating around z but with an opposite sign. Due to the existence of the spoiler gradients, the whole cube experiences dephasing, and only those spins selected by the 180° RF pulse can get rephased. Hence the signal from the grey part is destroyed.

Step 3: Apply the second selective 180º RF pulse on z.

Step 3 is exactly the same as step 2 but with a gradient applied on the remaining axis. Hence, the signal collected at echo time (TE) is only from the desired blue voxel.

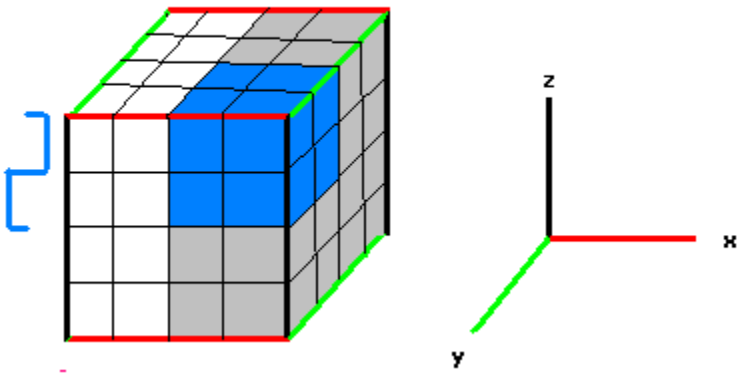


Fig. 2.5 A second 180º refocusing pulse applied along z with j_z=2

2.3 Acquisition Time Reduction in Wavelet Encoding

The same number of acquisition steps is required for Fourier and Wavelet encoding for the same spatial resolution. However, due to the finite support of the wavelet transform, for most encoding steps, only a portion of subject is exited while the

rest is relaxed, whereas for CSI a full TR is required for each step. Therefore, by arranging the excitations in an optimal order, a series of these sub-space signals can be acquired without the need of a full relaxation. The acquisition time is given by [11]:

$$Acq\_time = N_x.N_y.N_z.TR - N_{eff}.\left(TR - TR_{\min}\right), \tag{2.12}$$

where, N_x, N_y, N_z are the desired spatial resolution in x, y, and z respectively, and N_{eff} is given by [11]:

$$N_{eff} = \left[\begin{array}{l} 2^{2M_x}.\left(M_z - 2^{M_z} + 6\right)/3 + 2^{2M_y+1}.\left(M_z - 2^{M_z} + 5\right)/3 + 2^{M_z+M_y+M_x+1} - 2.2^{M_x} \\ +2^{M_y+M_x}\left(M_y - 2.M_z - 8 + M_x\right) + 2^{M_y}\left(-2.M_y + 2 + 2.M_x\right) - 4/3 - P(x) - P(y) \end{array}\right] \tag{2.13}$$

representing the number of times where the sequence is being executed using TR_{min}. TR_{min} is the total time needed to perform the sequence, which includes saturation, water suppression, the actual pulse sequence, and acquisition duration. The difference between TR and TR_{min} represents the extra time needed for spins to get fully relaxed after data collection, and it is also the time reduction of WE-SI comparing to CSI.

The following variables are set to: $M_x = \log_2(N_x)$, $M_y = \log_2(N_y)$, $M_z = \log_2(N_z)$, $P(x) = i.2^{2(i-1)}$, and $P(y) = 2.j.2^{2(j-1)}$, where i and j run from 2 to M_x and M_y respectively. If $M_x = M_y = M_z = M$, N_{eff} is simplified to:

$$N_{eff} = \left[2^{3.M} + 2^{2M}\left(3.M - 8\right)/3 - 4/3 - 3.P\right] = \left[N^3 + N^2\left(3.\log(N) - 8\right)/3 - 4/3 - 3.P\right] \tag{2.14}$$

As shown in Fig. 2.6 where a 4 by 4 by 4 resolution is acquired from the sample, acquisition time is reduced for the higher coefficients.

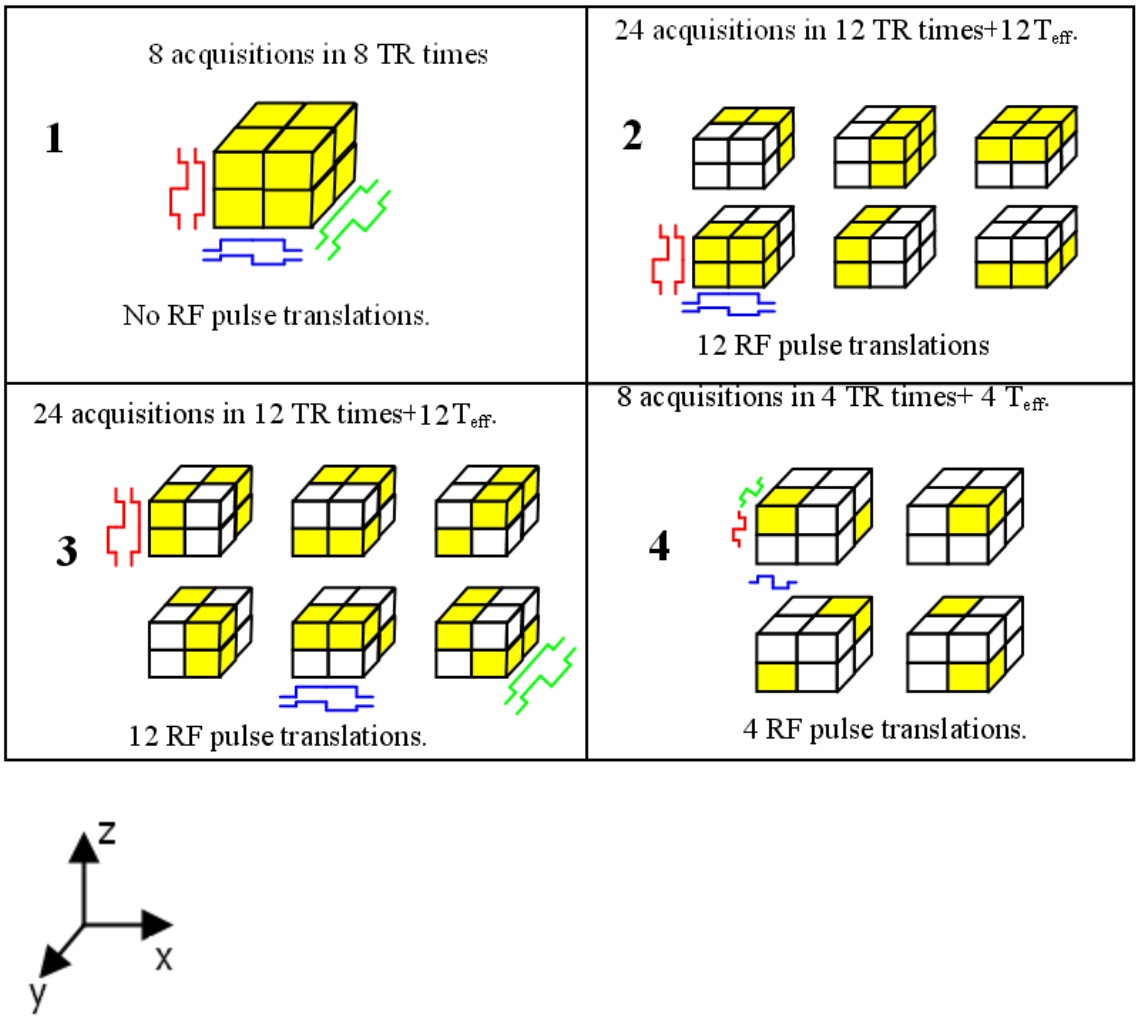


Fig. 2.6 an example of 4 by 4 by 4 wavelet encoding [11].

The required excitation steps and timing are as following:

(a) <u>The total FOV is excited</u>

As shown in Fig. 2.6-1, at lower dialation (j_x=0 or 1, j_y=0 or 1, and j_z=0 or 1), the whole FOV is excited. A TR is required for each acquisition, and a total of 8 acquisitions are

needed for collecting these 8 wavelet encoding coefficients. There is no RF pulse translation needed for all 8 encodings.

$$N_1 = \binom{2}{1}\binom{2}{1}\binom{2}{1} = 8 \tag{2.15}$$

where $\binom{a}{b}$ denotes a combinations of b.

(b) Half of the FOV is excited

There are 24 encoding steps where half of the FOV needs to be excited. This means at one and only one dimension j=2 and k=0 or 1, while in the other two dimension, j could be either 0 or 1 and k can be 0 only.

$$N_{\frac{1}{2}} = \binom{3}{1}\binom{2}{1}\binom{2}{1}\binom{2}{1} = 24 \tag{2.16}$$

The six regions possible are highlighted in yellow in Fig. 2.6-2. To minimize the acquisition time, the six regions are divided into three groups, where the two regions in each group are complementary to each other and they should be excited sequentially. In Fig. 2.6-2, the top and bottom encoding steps from the same column form one group. 90º RF pulses are applied on the direction where j=2, so that only half of the spins are flipped into the transverse plane. The rest of the region experiences two 180º RF pulses: the first one flips the corresponding spins from +z to –z and the second one flips spins from –z back to +z (z here represents the direction of B_0). Therefore spins in the second half of the FOV to keep precessing about +z as if they were never excited. Therefore, the second half of the FOV can be excited without waiting for a full TR.

The time needed for these 24 excitations is: $24TR\text{-}12(TR\text{-}TR_{min})$.

In different groups, scaling factor j=2 occurs in different spatial dimensions, therefore the 90º RF pulses are applied with gradient from corresponding j=2 dimension. We call this process RF pulse switching.

(c) a quarter of the sample is excited

In this case as shown in Fig. 2.6-3, we will have two dimensions with j=2 and the third dimension with j=1 or j=0. Each encoding step excites only a quarter of the sample. Since 90º RF pulses applies with j=2, we are able to excite two regions sequentially without waiting for a full TR in between.

24 encoding steps are needed when only a quarter of the sample is excited.

$$N_{\frac{1}{4}} = \binom{3}{1}\binom{2}{1}\binom{2}{1}\binom{2}{1} = 24 \qquad (2.17)$$

Similar to (b), we can divide all the possibilities into three groups based on where j=0 or j=1 is applied. For minimum acquisition time, steps within one group should be acquired sequentially without waiting for a full TR and the time needed here is 24TR- 12(TR-TR_{min}).

(d) 1/8 sample is excited

As shown in Fig. 2.1-4, there are 8 encodes in this case. Since the minimum dilation is still 2, we can only excite two encoding steps without waiting for the full TR. The time needed is 8TR-4(TR-TR_{min}).

2.4 The Acquisition Order

In order to minimize acquisition time and cross-voxel contamination, and for best signal strength, it is very important to acquire the wavelet encoding steps in a specific order.

First, we want to minimize the total acquisition time by skipping the waiting time for as many steps as possible. The localization property of the wavelet transform allows us to excite different regions of the sample. The next encoding step should try to be locate in a region that was not affected by all three RF , or are fully relaxed by the end of the previous acquisition. We also need to ensure to receive the optimal amount of signal for all acquisition steps. This means spins have to be fully flipped into the transverse plane with minimum projection in the longitude direction.

As shown in the previous section, the designed WE-SI sequence consists of one 90º RF pulse and two 180º RF pulses for each encoding step. Since all pulses are selective, there are a total of six cases that can occur for a spin inside the sample:

(1) The selected region sees one 90º and two 180º RF pulses.

(2) A region with no RF pulses.

(3) A region with two 180º pulses only.

(4) A region with one 90º and one 180º RF pulses only.

(5) A region with one 180º pulse only.

(6) A region with one 90º pulse only.

After finishing one acquisition step, we try to find the best step to be excited next. In an ideal case, we would like to acquire a signal from case (2). This requires the minimum of j_x, j_y and j_z to be greater or equal to 2 and we can collect $2^{\min(jx,\, jy,\, jz)-1}$ steps together

without waiting for a full TR. We would excite regions with same dilations in a diagonal order. The next region to be excited is ((j_x, k_x+1), (j_y, k_y+1), (j_z, k_z+1)).

To further speed up the process, we found that it is safe to collect regions in (3) without waiting for a full TR. In region (3), spins see the first 180º RF pulse and flip to –z immediately following longitudinal magnetization decay

$$SI \propto M_0(1-2e^{-T_1/t}) \tag{2.18}$$

Hence, at

$$t = 0.5(TE_1 + TE_2)$$

$$SI \propto M_0(1-2e^{-2T_1/(TE_1+TE_2)}) \tag{2.19}$$

In our sequence,

$$0.5(TE_1 + TE_2) = 0.5TE \le \frac{75}{2} = 37.5ms$$

whereas T_1 is at around 500ms for brain.

Also at this time, the second 180º RF pulse is applied to flip magnetization by 180º:

$$SI \propto -M_0(1-2e^{-2T_1/(TE_1+TE_2)}) \approx -M_0(1-2) = M_0 \tag{2.20}$$

There is enough time (around 0.5 seconds for ADC acquisition) for the region to get fully relaxed before the current acquisition ends. Therefore, we could acquire the next encoding step from the regions seeing two 180º pulses.

We avoid acquiring data directly from regions in case (4), (5) and (6). If encodes from these regions are needed, we will need to wait a TR for spins to be fully relaxed before the next acquisition.

In order to maximize the number of regions in case(2) and case(3) and to save time we should always play the 90º RF pulse with the highest dilation.

To automatically select the the next encoding step for a minimized total acquisition time, the following algorithm was developed. It requires resolution in all three dimensions to be at a power of two.

```
function b=getHeader2(nx,ny,nz)
for i=0:nx*ny*nz-1
            find(the next encoding step in numerical order);
            if(!find)
                break;
            end
               calculate the dilation and translation for current step

            [Y,who_is_90]=max(scales);

               Put 90 degree RF pulse selective on largest dilation direction;
               Put the first 180 degree RF pulse selective on the second largest dilation
               direction;
               Put the second 180 degree RF pulse selective on the smallest dilation
               direction;

               for j=0:Y-1 // Y encoding steps can be acquired together         //without
               waiting full TR
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5

                    if(scale_1>1)
                        p=mod(ii+j,scale_1);
                        if p==0
                            p=scale_1;
                        end

                        p=p+scale_1;

                    else
                        p=ii;
                    end
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    if(scale_3>1)
                        q=mod(jj+j,scale_3);
                        if(q==0)
                            q=scale_3;
                        end
    %                    q=q+floor(j/scale_3)*scale_3;
                        q=mod(q,scale_2);
                        if(q==0)
                            q=scale_2;
                        end
%                        mod(jj+j,scale_3)
%                        floor(j/scale_3)
%                        scale_3
                        q=q+scale_2;

                    else
                        q=jj;
                    end
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    if(scale_3>1)
                        r= mod(kk+j,scale_3);
                        if(r==0)
                            r=scale_3;
                        end

                        r=r+scale_3;

                    else
                        r=kk;
                    end
                       //flag the encoded steps
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                end
end
```

2.5 Signal to Noise Ratio in Wavelet Encoding

WE-SI excites a distribution of spins at each acquisition time. The size of excited subspaces decreases as wavelet dilation increases, whereas in Fourier encoding, the whole FOV is excited at each acquisition step. Hence, on average, signal to noise ratio is lower in WE-SI than in CSI. As a consequence, the sensitivity in 3D WE-SI compared to Fourier encoding drops by [11]:

$$SNR_{WESI} = SNR_{CSI}\sqrt{\frac{27N_xN_yN_z}{(4+2N_x^2+2N_y^2+N_x^2N_y^2)\cdot(N_z^2+2)}} \tag{2.21}$$

which is approximated to

$$SNR_{WESI} \approx SNR_{CSI}\cdot(3/N)^{3/2} \quad \texttt{if} \quad N_x = N_y = N_z \tag{2.22}$$

2.6 Problems with WE-SI

2.6.1 Chemical shift misregistration

As introduced earlier, both magnetic field gradient (position) and chemical shift can contribute to a frequency difference. For WE-SI, translation is realized by shifting the

center frequency of RF pulses. Hence, for different species, we need different shift frequencies to arrive at the same spatial position. However, there is no easy way to shift different frequencies with a single excitation. A higher bandwidth will help for chemical shift misregistration. Details are shown in chapter 4.

2.6.2 RF pulse imperfection

With finite data points in the time domain, the RF pulse is not a perfect square. The tail and transition bands contaminate the data.

2.6.3 B_0 field inhomogeneity

At higher field strength, it is more difficult to achieve a high degree of B_0 field homogeneity [28]. WE-SI is magnitude modulated and is more sensitive to B_0 inhomogeneity than CSI. Hence WE-SI technique may suffer more from pixel bleed.

Chapter 3

Material and Methods

This chapter briefly introduces the implementation environment, procedures, and the design of test methods.

3.1 Hardware Environment

We are using a 3 Tesla whole body Siemens clinical scanner located at the National Research Council of Canada, Institute of Biodiagnostics. Transmit/Receive head coil using only one receiving channel are attached to the Tray in at head position (shown in Fig. 3.1). The scanner is connected to a user interface computer with controlling and displaying functionalities provided by Siemens.

Fig. 3.1 The Siemens 3T system.

3.2 Programming Environment

Syngo is the common software to all current Siemens Medical Solutions Products (also known as Medcom). SyngoMR is the MR implementation of this software and it is also called Numaris 4. The software was installed into the scanner to control MR data acquisition [29].

The actual MR sequence in Numaris 4 was written in C/C++. The programming frame work is called Integrated Development Environment for Applications (IDEA) which consists of two prime parts: Sequence Development Environment (SDE) and Image Calculation Environment (ICE). Normally, SDE controls the acquisition and gather the raw data files (.dat). These files are then passed to ICE for data processing. For our sequence, ICE is bypassed and is replaced by MATLAB programs written in-home.

The software version we used for our development is SyngoVB15.

A pulse sequence is a program on the scanner that is used to program the hardware for data collection. A sequence can be treated as an object with four main member functions: fSEQInit, fSEQPrep, fSEQRun, and fSEQCheck. Static variables were used to transfer information between different functions.

fSEQInit initializes measurement parameters, set up the boundary limits, and conFig. static objects. fSEQPrep prepares RF pulses, gradients based on the parameters set up by fSEQInit. It also estimates the total energy look ahead calculation. In fSEQrun, sequence timing is created. fSEQcheck overflow the check.

Our WE-SI sequence is based on a single voxel spectroscopy sequence (SVS). However, since the resolution, RF pulses and acquisition order are totally different, we've made extensively modifications in all functions except fSEQcheck. The detailed coding is available in appendix A.

3.3 Manual of Operation for WE-SI Sequence

The WE-SI sequence is a modification of the PRESS sequence (SVS in Siemens). However, since it has multi-resolution for 3D, the sequence control cards are similar to CSI, as shown in Fig. 3.2 to 3.6. Important settings for the WE-SI sequence will be highlighted in this section.

On the left session of the routine card (shown in Fig. 3.2), position and FOV need to be set properly based on previously acquired localizer images. For WE-SI, resolution is closely related to FOV and RF bandwidth. Due to the nature of wavelet encoding, there

is no folding effect that can be introduced by an imperfect excitation pulse (see section 2.9.3). Hence, View of Interest (VOI) should be set equivalent to FOV. On the right of routine card, we should note "Average" should be set to 1 always. The value input for "average" goes to the number of inner loops in the fSeqRun() function of the sequence (Appendix A). This loop will repeat the same acquisition step for "average" times. In this case, since the same region is to be excited, we will always need to wait a full TR time in between, which is very inefficient for WE-SI. The actual averaging functionality to obtain a better SNR is set by the "Measurements" value, found on the "contrast" card. "Measurements" leads an outer loop in fSeqRun() function of the sequence, which will perform all encoding steps and then repeat the whole process from the first encoding step to the last for a total of "Measurements" times. In changing the number of measurements, we are accomplishing the sequence order introduced in 2.4, and acquisition time reduction according to equation 2.13.

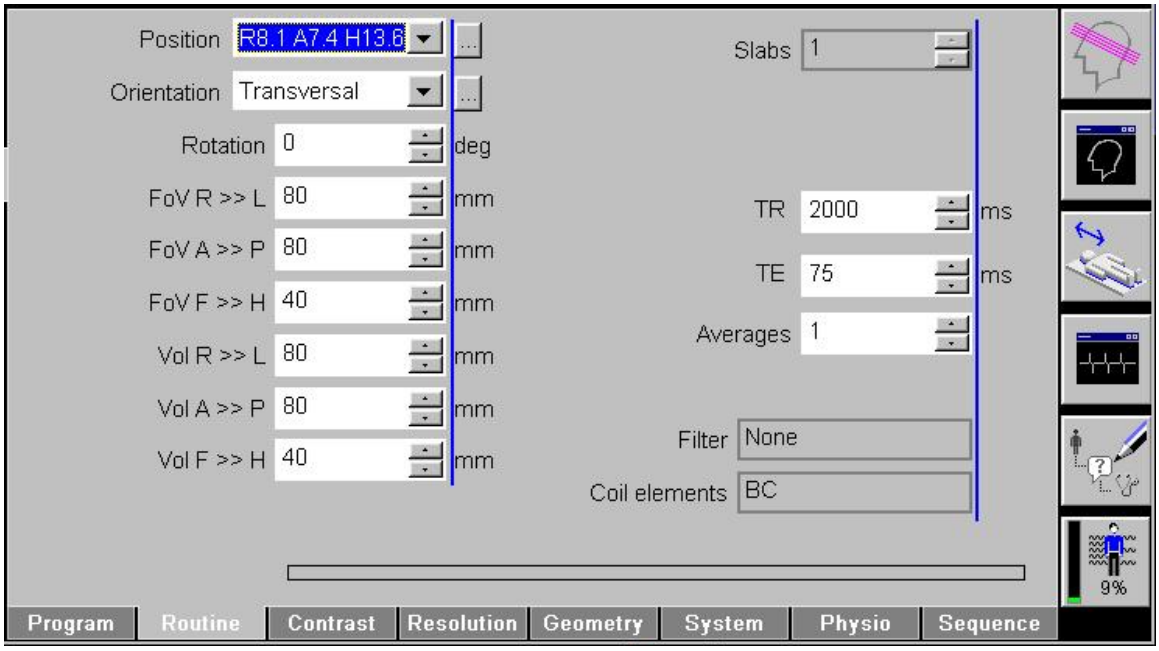


Fig. 3.2 Routine Card.

In the contract card (Fig. 3.3), the setting of "Measurement" increases SNR. Let n be the number of measurements, the SNR is proportional to $\sqrt{n}$.

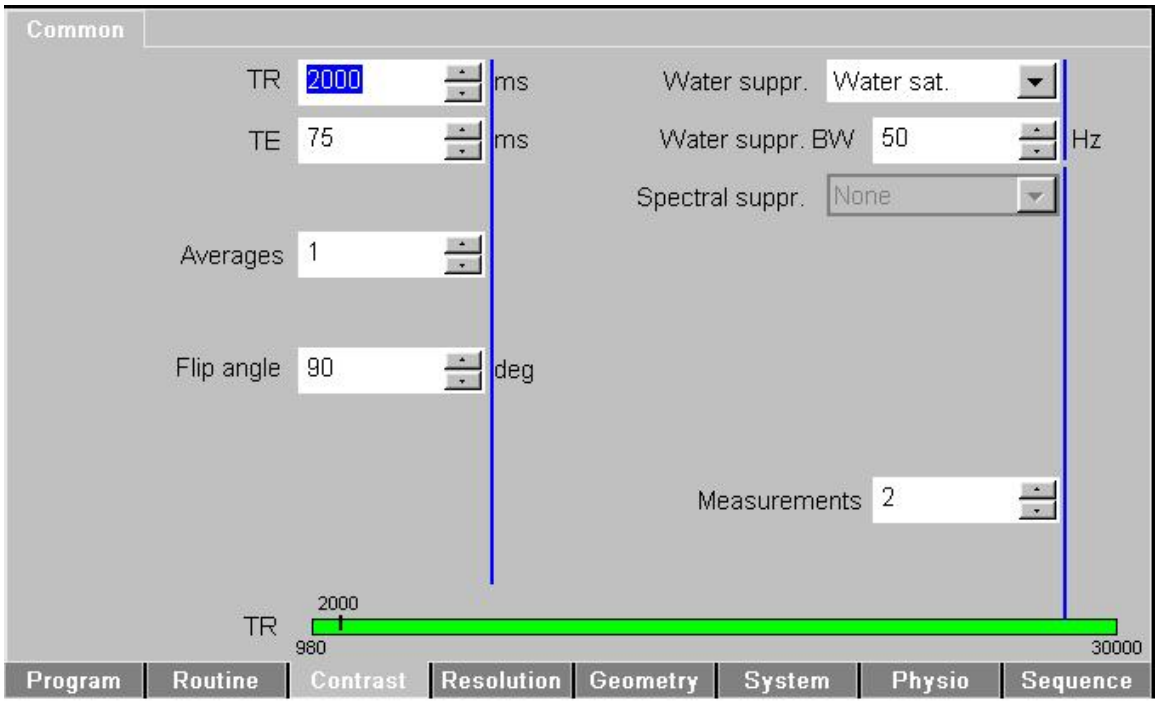


Fig. 3.3 Contrast Card.

In the transmitter/receiver subcard under "system" (Fig. 3.4), we need to manually adjust the transmitter voltage. The reference amplitude is the input voltage to the transmitter (RFPA) required to produce a 180º rotation for a 1 msec (1000 usec) rectangular RF pulse. All RF pulses are scaled to this reference RF pulse. Single 90º and 180º pulses are kept at the default voltage. The profile of dual 90º and 180º RF pulses have both positive and negative lobes (Fig. 3.5). Hence the voltage requires better optimization. From extensive phantom testings, we conclude that the relation between dual RF pulses and reference voltage to be:

V_{D90} =0.75V_{ref};

V_{D180}=2.75V_{ref};

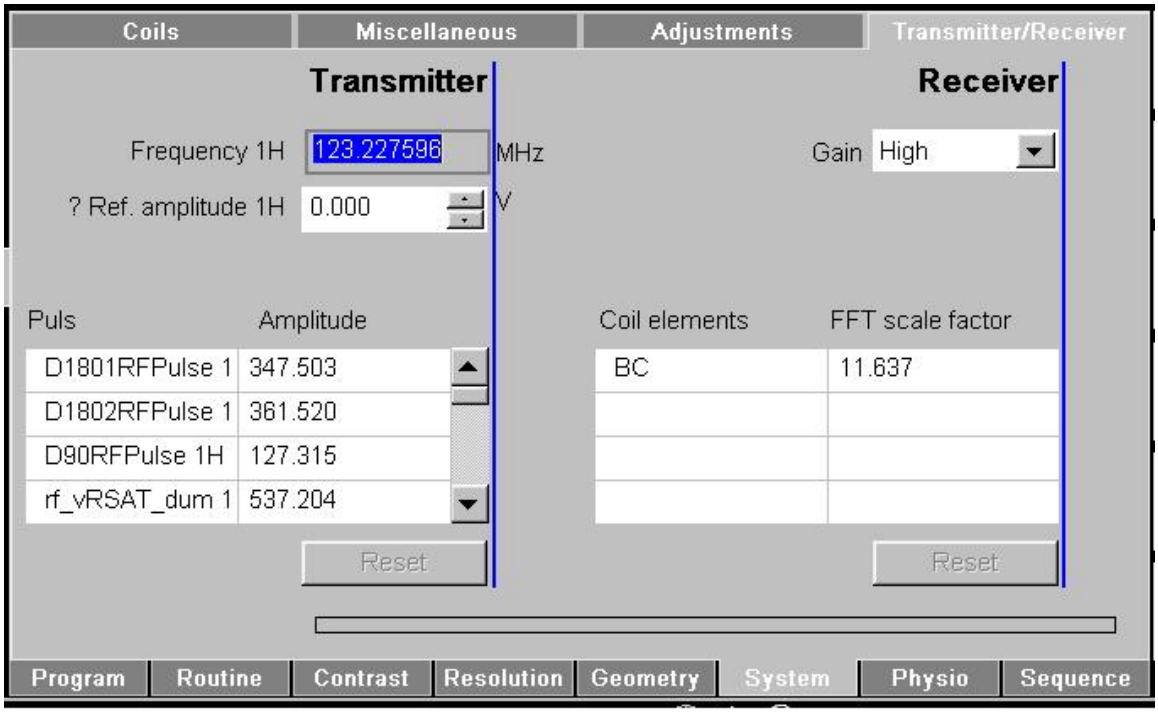


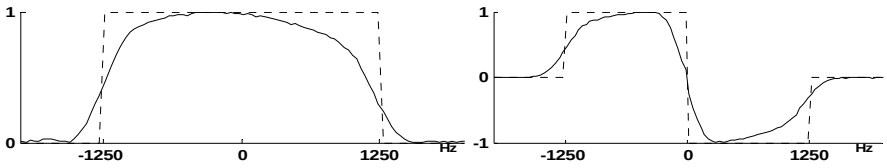
Fig. 3.4 Transmiter/Receiver card

All other settings, such as resolution, saturation band, and TE/TR values are the same as the regular CSI sequence.

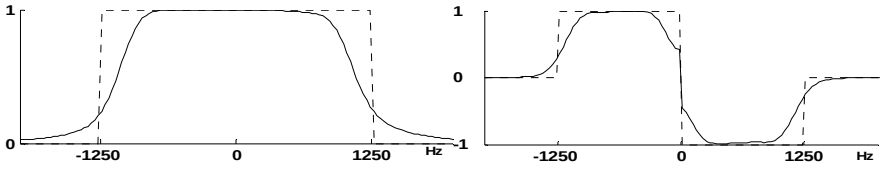
3.4 RF Pulse Parameters

As introduced previously, we have developed the WE-SI technique by modifying the spatially localized PRESS sequence to acquire 3D WE-SI data. Refined sinus cardinal (sinc) functions, representing excitation (90º) and refocusing (180º) RF pulses for the WE-SI sequence, using the Shinnar-Le Roux algorithm [30] are generated. The profiles of these RF pulses, one single and one dual band, resemble scale and Haar wavelet functions respectively (Fig. 3.5). The excitation RF pulse is applied along the slice

direction and the refocusing RF pulses are applied along the phase and read directions by analogy to the imaging sequences [12]. To achieve spatial encoding in the three directions, dilations and translations of the dual band RF pulses detailed elsewhere [11], are achieved by increasing the selection gradient strength and shifting the centre frequency of the RF pulses respectively. The duration and bandwidth of all RF pulses are 5.2 msec and 2500Hz, respectively. Fig. 3.5 shows the spatial profiles of the RF pulses as executed on the scanner (solid line) versus the Haar wavelet profiles (dashed line), where the difference between the two shapes is in the transition band and the edges. This is mainly due to the short duration of the sinc functions of the RF pulses. The signal loss and cross voxel contamination can be corrected by data reconstruction in the inverse wavelet transform.



A. Single 90 RF pulse B. Dual 90 RF pulse



C. Single 180 RF pulse D. Dual 180 RF pulse

Fig. 3.5 Profiles of RF pulses (solid lines) used as Haar functions (dashed lines) in the WE-SI sequence.

3.5 Phantom and Acquisition Parameters

We conducted phantom studies at different spatial resolutions. Low resolution 2×2×2 and 4×4×2 data were acquired with different home-made phantoms (Fig. 3.6), and 8×8×4 data were acquired with a spherical phantom containing known solution with known concentrations. The two home-made phantoms are as shown in figure 3.6. They were made from two rectangular plastic holders containing equally spaced 2×2 and 4×4, 14 mm diameter holes. The size of each phantom is 40 mm × 40 mm and 70 mm × 70 mm. Cylindrical tubes filled with aqueous solutions of metabolites with known concentrations were placed in the holes of the plastic holder, dropped in a container filled with water, which in turn immersed in a cylinder filled with canola oil. Single and dual band RF pulses with profiles resembling Haar wavelet functions (5.2 msec duration and 2500 Hz bandwidth) are used to acquire 3D WE-SI data on a 3T Siemens magnet. The acquisition parameters are TR=2 sec, TE = 45 ms, ADC bandwidth = 2kHz, 1k points, and NEX = 4. Two sets of phantom experiments have been conducted to evaluate the performance of WE-SI versus CSI. In the first set (4x4x4 matrix size), a uniform spherical phantom containing an aqueous solution of brain metabolites with known concentrations (NAA, Creatine and Choline, etc.) is used for SNR calculations and data analysis using LC Model. In the second set (4x4x1 matrix size), a home-made phantom with 16 tubes containing different metabolite solutions with known concentrations is scanned to determine voxel contamination

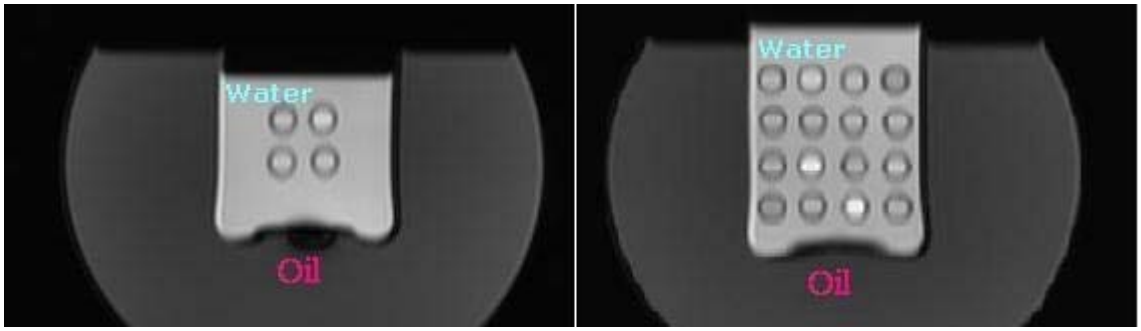


Fig. 3.6 Localization MR images of the 2×2×2 (left) and 4×4×2 (right) phantoms. The box represents the FOV used in the WE-SI sequence.

3.6 In-vivo Tests

Optimized 3D WE-SI sequence based on phantom results were used to conduct studies on the human brain in healthy volunteers. For each healthy subject, a set of standard volumetric anatomical images were obtained. In order to obtain the region-dependent thresholds, it is necessary to have sufficient brain coverage for the WE-SI dataset. Therefore, we will obtained three sets of 3D ^{1}H WE-SI data.

For the first set, the repetition time and echo time was TR/TE=2000/75 ms with two averages, and matrix size of 4 by 4 by 4 with a nominal voxel size approaching 4 cm^3.

The acquisition time were:

1) localizer images (~ 5 minutes)
2) 3D CSI TR/TE = 2000/75 (4 by 4 by 4 Nex = 2, ~ 34 minutes)
3) 3D WE-SI TR/TE = 2000/75 (4 by 4 by 4, Nex = 2, ~ 20 minutes)

For the second set the repetition and echo times were TR/TE = 1500/35 ms with four averages. The matrix size used was be 8 by 8 by 2 with a nominal voxel size approaching 2 cm^3.

The acquisition times were :

4) 3D CSI TR/TE = 1500/35 (8 by 8 by 2, Nex = 2, ~ 4.5 minutes)

5) 3D WE-SI TR/TE = 1500/35 (8 by 8 by 2, Nex = 2, ~ 3 minutes)

For the third set the repetition and echo times were be TR/TE = 1500/35 ms with four averages. The matrix size was 8 by 8 by 4 with a nominal voxel size approaching 1 cm^3.

The acquisition times were:

6) 3D CSI TR/TE = 1500/35 (8 by 8 by 4, Nex = 2, ~ 4.5 minutes)

7) 3D WE-SI TR/TE = 1500/35 (8 by 8 by 4, Nex = 2, ~ 3 minutes)

The total time in the machine for each subject was on the order of 1 hour and 10 minutes. The data was transferred to a computer for analysis using in-house software (Appendix B). Two different echo times were used to check the sensitivity of the method at different echo times, and if the method is able to detect metabolites at short apparent relaxation time.

3.7 Raw Data Gathering

3.7.1 TWIX

TWIX is a built-in function that takes scanner output without any data processing. The data are stored in the order of acquisition. The file contains a common header followed by individual acquisitions. Each individual acquisition contains a header and a data vector. The data vector is a vector of complex numbers represented by two floats, the real part and imaginary part respectively. The actual decoding program is in appendix B.

3.7.2 spectro files

Spectro files contain processed data without headers. These files are accessible by the Siemens on-line analysis software. We need to overwrite a CSI data file with processed WE-SI results in order to display the results on scanner. The MATLAB code that is used to overwrite .spectro files is in appendix C.

3.7.3 rda files

The .rda files are processed data with headers. The header contains information about the patient and all acquisition parameters. rda files are important as they can be directly used by LCmodel, which is a common software for spectroscopy data analysis. We can get .rda files directly from the scanner by going to spectroscopy>options>export rawdata. The Matlab code that is used to create a .rda file is in appendix D.

3.8 Data Analysis Method

In terms of data analysis, two types of software were used. The first one is LCModel [31] which is commonly used by MRS users. It automatically quantizes the absolute and relative concentrations of metabolites and signal to noise ratio. It requires basis functions

for specific pulse sequence type and echo time information. The input to this software is .rda files from the scanner which is taken from the spectroscopy card directly. The outputs are quantization results and figures are similar to Fig. 3.7. The fit of spectrum is given by red curves. The actual quantification of metabolites is given in the table at the left part of the Fig. 3.7.

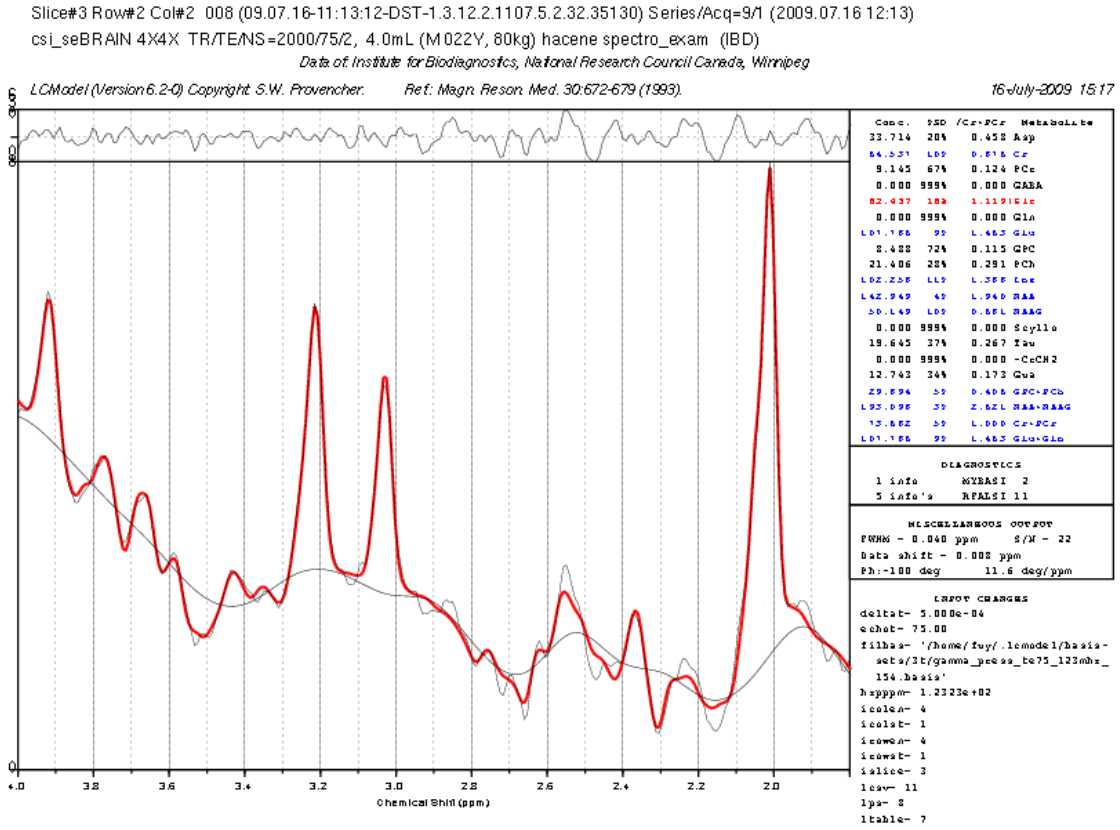


Fig. 3.7 LCModel output

To manually adjust the frequency shift, phase, baseline, and to have a better overlay of the image and the spectrum, Simens online software is used. This is the easiest and quickest way to view the spectrum. But it does not provide absolute quantification and scanner for data processing must be conducted on the scanner.

For our analysis, we overlaid the spectrum evaluation from LCmodel with the imaging from Siemens to better represent our results.

Pre-processing is required regardless of the software. A discrete inverse Haar wavelet transform must be performed on the raw data, instead of a discrete inverse Fourier transform. This pre-processing step is done in MATLAB by first arranging wavelet ecoding results into regular order and then performing an inverse wavelet transformation in all three dimensions. The code is attached in appendix B.

For in-vivo results, a line broadening of 2.5Hz is applied to both WE-SI and CSI results. Line broadening is a method to convolve the resultant spectrum with a Lorentzian function whose width at half height equals a certain frequency value (2500Hz in our case). Line broadening acts as a lowpass filter that filters out the higher frequency, and smoothes the spectrum.

Chapter 4

Results and Discussion

In-vitro and in-vivo results are presented in this chapter.

4.1 Phantom Results

4.1.1 2×2×2 Phantom Test

At this resolution results show accurate information on absolute metabolite quantification. Fig. 4.1 shows the accurate localization findings of the metabolite peaks along with fitting results (red). The estimated metabolite concentration versus the expected is shown in Table 4.1. Voxel contaminations are insignificant at this low resolution. Since no wavelet translation (RF pulse shift) is performed, there is no reduction on acquisition time.

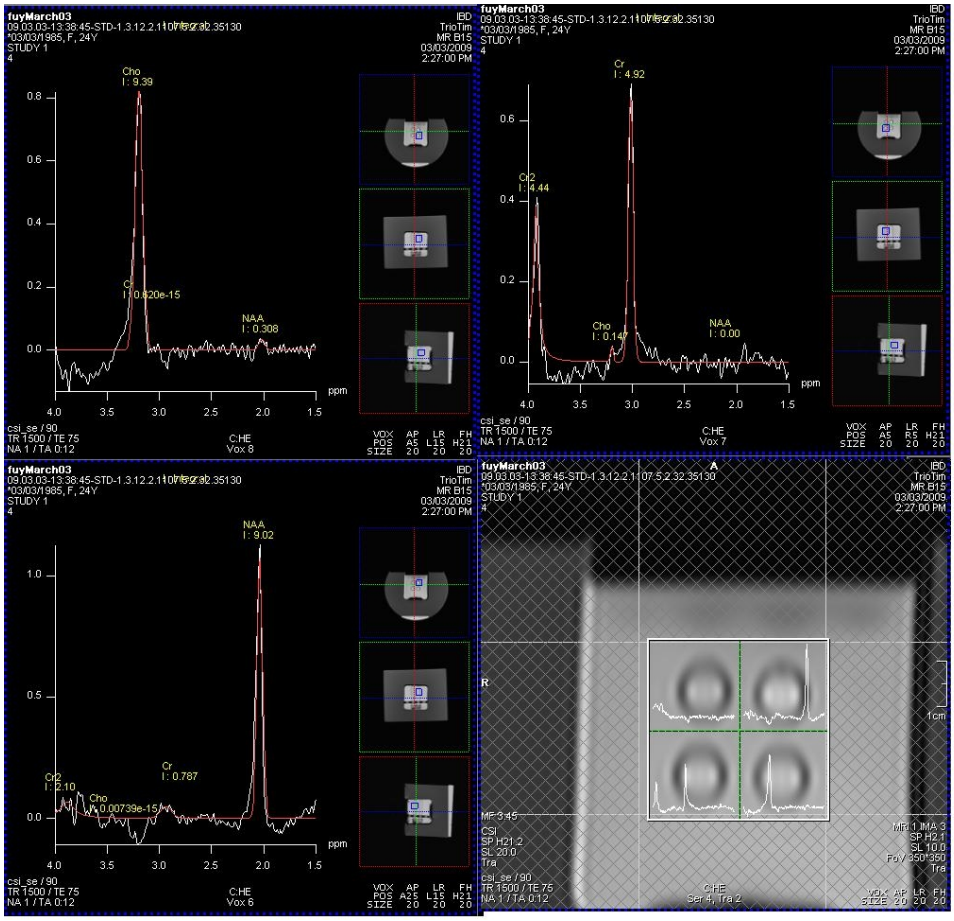


Fig. 4.1 Front axial slice of acquired metabolite spectra for the 2×2×2 WE-SI .

TABLE 4.1

METABOLITE CONSENTRATION ESTIMATE MILLI-MOLAR (MM) BY 2×2×2 PHANTOM

metabolite \ voxel		NAA	Creatine	Choline	Water
NAA (mM)	Measured	144	0	0	0
	True	150	0	0	0
Creatine (mM)	Measured	0	64	4	0
	True	0	65	0	0
Choline (mM)	Measured	0	0	48	0
	True	0	0	50	0

TABLE 4.2

WE-SI TIMING

Resolution	Total Acquisition Time		
	Experiment	Calculated	reference (CSI)
2×2×2	32 sec	32 sec	32 sec
4×4×2	196 sec	191 sec	256 sec
8×8×4	1224 sec	1250 sec	2068 sec

4.1.2 *4x4x2 Phantom Test*

The results also show that WE-SI was able to give accurate metabolite spatial information with low voxel contamination (Table 4.3, Fig. 4.2). We are able to reduce the acquisition time by 23.4% using WE-SI compared to CSI at this resolution (Table 4.2), which are consistent with the theoretical calculations (equation 3.13).

TABLE 4.3

WE-SI EVALUATION FOR THE 4×4×2 PHANTOM

voxel position	Expected Metabolites	Contamination
(1,1)	water	0
(1,2)	NAA	0
(1,3)	Choline	NAA (2%)
(1,4)	Glycine	0
(2,1)	Acetone	0
(2,2)	water	0
(2,3)	NAA	0
(2,4)	Creatine	NAA (9%)
(3,1)	Sarcosine	0
(3,2)	Creatine	0
(3,3)	Choline	NAA 5%)
(3,4)	NAA	0
(4,1)	Glycine	Succinate (6%)
(4,2)	Succinate	0
(4,3)	Sarcosine	Choline (4%); Acetone (12.7 %)
(4,4)	Acetone	Succinate (5%)

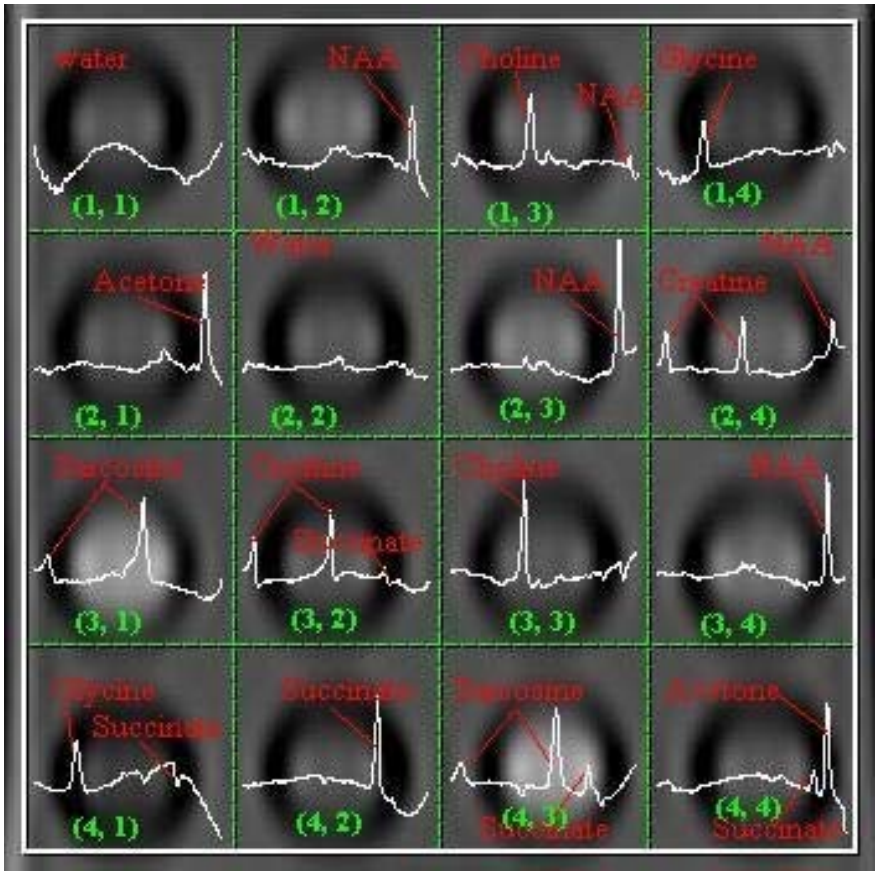


Fig. 4.2: Front axial slice of acquired metabolite spectra for the 4×4×2 WE-SI .

Discussions on Voxel contamination

(1)Voxel contamination in WE-SI at 3 Tesla is mainly due to the RF pulse profiles not perfectly matching the shapes of the Haar functions (Fig. 3.5). The tails of the RF pulse profiles seen as solid lines in Fig. 3.5 extends outside the boxcar shown as a dashed line. These tails pick up a small portion of signal from neighboring voxels.

(2)The transition bands are large causing voxel contamination. To minimize the profile errors, we replaced the Haar function values (1 and -1) in the inverse wavelet transform by numbers obtained from the fit of the RF pulse profiles to boxcar shapes.

(3) Another reason for voxel contamination is the B_0 field inhomogeneity. At higher B_0 field strength, homogeneity is more difficult to achieve, especially with phantoms made from glass vials, plastic holders and containers, which complicates the shimming process for better B_0 field homogeneity due to susceptibility magnetic field effects [28]. As shown in Fig. 4.2, we observe more contaminations in the bottom row, because of the poor shimming at that location.

(4) The nature of chemical shift misregrastration affects more on the metabolites with lower ppm values to a greater extent. Consider NAA with chemical shift at 2.0 ppm. The chemical shift in frequency is calculated as:

$$\Delta f = \Delta ppm \times \omega_0 = (4.7 - 2.2) \times 123.3 = 308.25 Hz \quad (4.1)$$

Therefore if NAA is present within the region of the scanning, we will lose a certain amount of signal depending on the position of NAA, current dilation and translation of the wavelet encoding.

Taking the following example, in which case we only have Water (left) and NAA (right) (Fig. 4.3). If a 1D acquisition of resolution four is performed, the four acquisition steps with their spatial coverage are shown in green bars.

For the step encodings one and two, gradients strength is calculated by:

$$G_x = \frac{BW}{\gamma L} = \frac{2500}{\gamma L} \tag{4.2}$$

Water is resonating at a frequency depending on x:

$$f_w = G_x \cdot x \tag{4.3}$$

NAA is resonating at a frequency 308.25 Hz lower than water frequency. Thus the frequency shift is equivalent to a shift in the spatial domain:

$$\Delta x = \frac{\Delta f}{\gamma G_x} = \frac{\Delta f}{BW} L \tag{4.4}$$

This shift is indicated by the blue box.

For encoding step 1:

Signals collected are not affected by this shift

For encoding step 2:

Due to the dual shape of the RF pulse, part of the NAA signal is cancelled by the NAA signal shifted into the water region (the yellow shaded area) due to their opposite sign. Hence, the total signal energy is weakened.

For encoding step 3 and 4:

Since the x coverage is halved, the gradient strength is doubled. The chemical shift still exists but Δx is halved. As shown in the Fig. 4.3, encoding step 3 picks up NAA signals passing the boundary (yellow shaded), and contamination is observed. At encoding step 4, we were supposed to observe a zero wavelet coefficient because NAA should be divided evenly by positive and negative part of the dual pulse. However, due to the shift, we are most likely getting a small positive value.

This chemical shift misregistration problem is affecting all metabolite peaks. The further the metabolite is from water peak, the more shifted it is going to be in spatial domain. This problem is also more serious at higher field strength. From equations 4.4, it is easy to see that greater RF bandwidth pulses with higher gradient strength will reduce this effect.

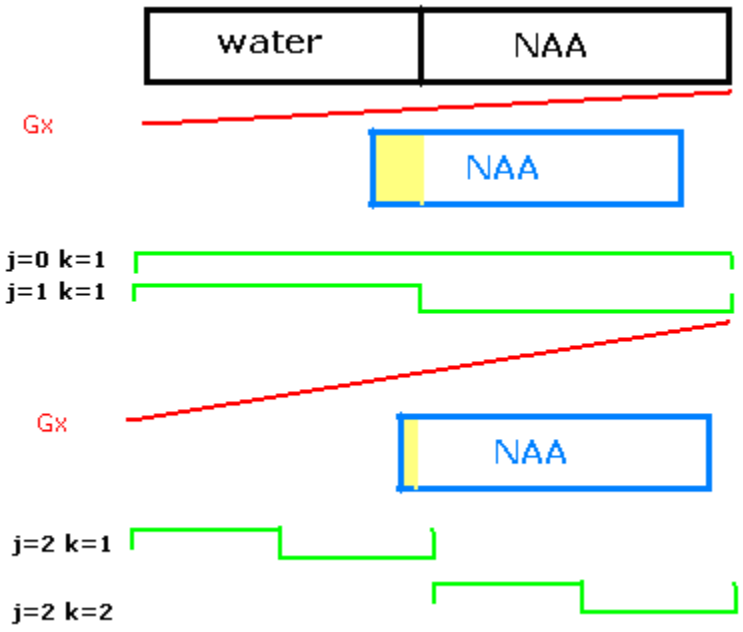


Fig. 4.3 Illustration of chemical shift misregistration.

A. 8x8x4 Spherical Phantom Test

This test was performed to evaluate and compare the SNR of WE-SI versus CSI. As expected the SNR measured from a subset of voxels located at the center of the sphere (B_0 field homogeneity is better at the center of the sphere) is lower by 28.8% compared to CSI. These results are comparable to the calculated ones at 29.9% (equation 2.19). However, compared to results obtained at 1.5 tesla (results not shown), the SNR at 3 tesla is higher. Acquisition time reduction is also obtained at this resolution (Table 4.2).

4.2 In-vivo Results

Figure 4.4 displays the second axial slice spectra from a 4 by 4 by 4 WE-SI and from CSI human subject data, respectively. Fig. 4.5 displays the back axial slice spectra from an 8 by 8 by 2 WE-SI and from CSI human subject data, respectively. Similar relative quantification results of the metabolites were obtained. Table 4.4 shows the mean values and the standard deviations over all the voxels for the six volunteers at each spatial resolution of the metabolites ratios of N-acetyl-aspartate (NAA) and choline peaks to the creatine peak. In table 4.5, acquisition times are given for both WE-SI and CSI at different resolutions. WE-SI is quicker in acquiring MRSI data than CSI. The experimental acquisition times approach the calculated ones.

As expected, the SNR was lower in WE-SI than CSI by a factor of 1.5, 2.7, and 3.2 in the 4 by 4 by 4, 8 by 8 by 2 and 8 by 8 by 4, respectively (Table 4.6). The SNR was calculated as the ratio of the NAA peak intensity and the standard deviation of noise. The SNR values are consistent with Equation 3.21.

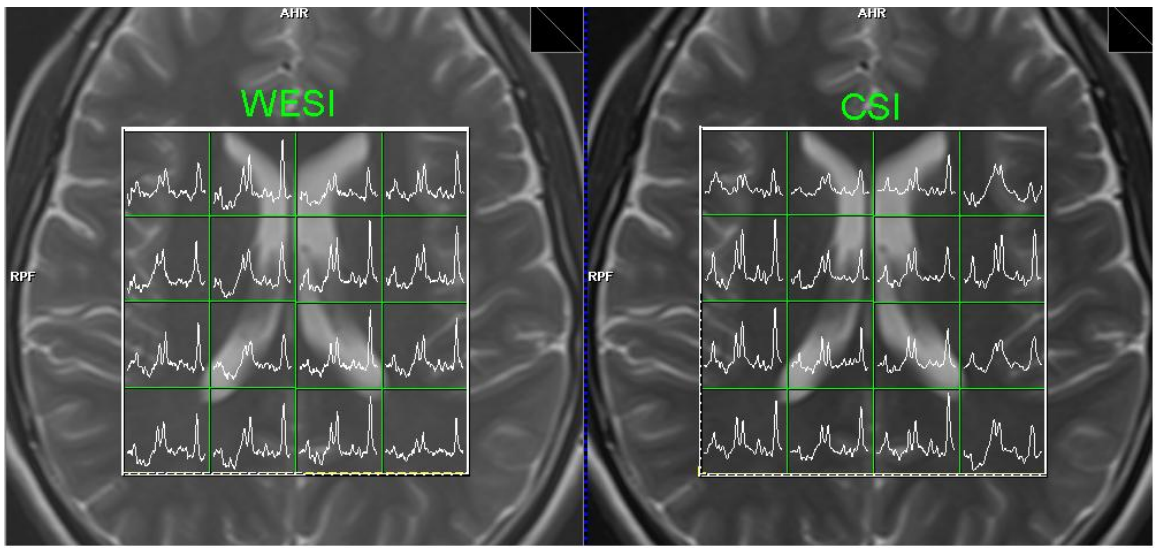


Fig. 4.4 In vivo spectra from the second axial slice of the 4x4x4 WE-SI (left) and CSI (right) data.

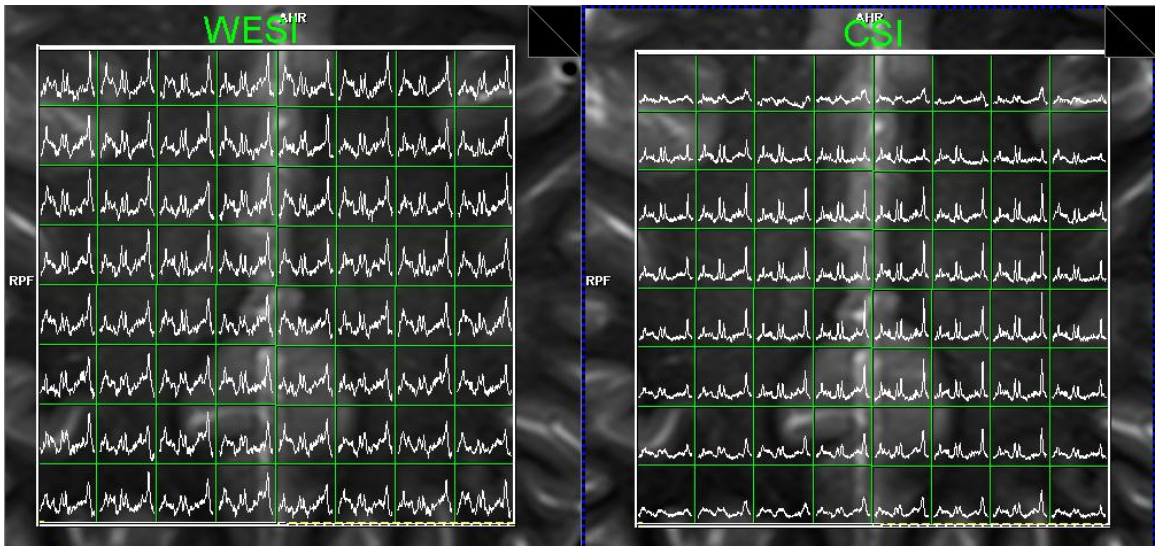


Fig. 4.5, In vivo spectra from the back axial slice of the 8x8x2 WE-SI (left) and CSI (right) data.

Table 4.4 The mean and standard deviation values SNR for different resolutions.

SNR	4x4x4	8x8x2	8x8x4
CSI	18.2500 ± 7.1738	18.6758 ± 7.1982	15.6543 ± 6.1891
WE-SI (experiment)	12.5235 ± 3.4935	7.5000 ± 2.2698	4.9257 ± 1.6256
WE-SI (calculated)	11.8502	6.7867	4.5658

Table 4.5 Experimental acquisition time duration for both WE-SI and CSI at three different resolutions.

	WE-SI	CSI
4x4x4 (TR=2s,avg=2)	3min20sec	4min16sec
8x8x2 (TR=1.5s, avg=4)	8min42sec	12min48sec
8x8x4 (TR=1.5s, avg=4)	17min14sec	25min36sec

Table 4.6 Experimental and calculated SNR values for WE-SI and CSI at three different resolutions.

Mean ± SD		NAA/Cr	Cho/Cr
4x4x4	WE-SI	1.7156 ± 0.4487	0.2699 ± 0.0812
	CSI	1.7742 ± 0.6753	0.2930 ± 0.0762
8x8x2	WE-SI	1.4010 ± 0.4994	0.2944 ± 0.1134
	CSI	1.5031 ± 0.6753	0.2953 ± 0.0841
8x8x4	WE-SI	1.4756 ± 0.6923	0.2835 ± 0.1553
	CSI	1.6025 ± 0.6784	0.2794 ± 0.0865

Chapter 5 Conclusions and Future Work

A three dimensional wavelet encoding method for acquiring magnetic resonance spectroscopic imaging data was presented. The proposed WE-SI was compared to the gold standard CSI technique. This comparison, offers a valuable indication of acquisition time, voxel contamination and sensitivity. In contrast to Fourier encoding which usually works with fine grids over a large FOV, the wavelet encoding, is better for small FOVs with low resolutions. At the same time, WE-SI reduces acquisition time compared to CSI. The reduction in acquisition time is directly proportional to the spatial resolution and dimensions. However, it suffers from lower SNR than CSI, although results obtained at 3 tesla were better in sensitivity than those obtained at 1.5 tesla. In order to increase the SNR, less spatially localized wavelets should be used [32]. A more significant effect of chemical misregistration was also seen at higher field with WE-SI method. Overall, WE-SI is a reliable method at 3 tesla field strength and can be applied to clinical studies.

For future work, we suggest combining WE-SI with a parallel imaging (PI) method for further reduction in acquisition time. This work requires the 3T platform to be equipped with multi-receive channels covering independent and fixed object regions that are used to collect the MR signal (Figure 6.1). Similar to Fourier encoding with PI [33] where a number of k-space lines are not acquired resulting in an aliased image, a RF pulse with redundant Haar wavelets will result a super-imposed image when reconstructing it with a lower scale wavelet (N/R instead of N). Then as with SENSE [33], coil sensitivity maps can be used to separate a super imposed voxel into R different voxels at their corresponding positions. By applying PI, the acquisition time is reduced

by a factor R, called the acceleration factor. A decrease in SNR is also expected. However, since in Wavelet encoded parallel imaging (WE-PI) the RF excitation is not reduced by a factor of R, the SNR drop with PI in WE-SI should be less significant than with CSI.

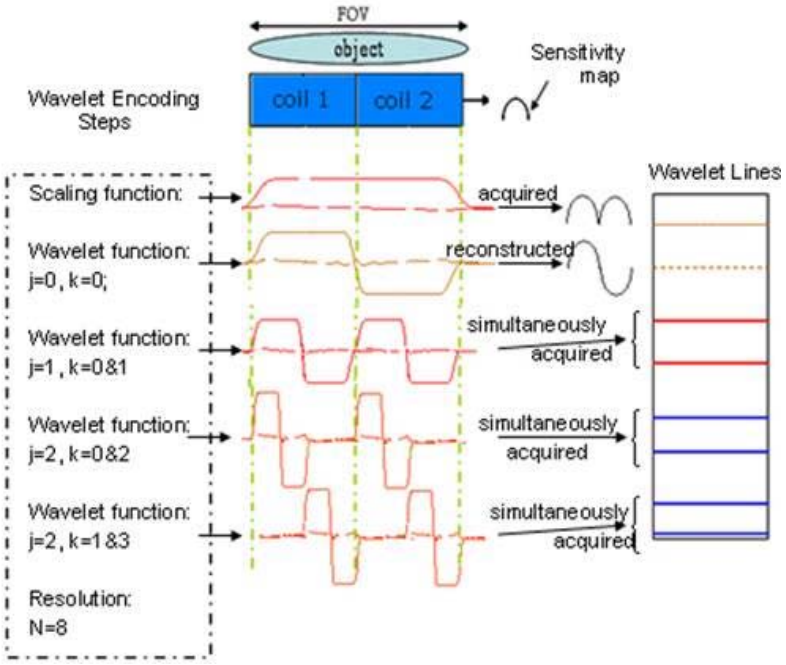


Figure 6.1 Acquired and reconstructed wavelet domain lines with N=8 and R=2. Translation and dilation values k and j are displayed for each wavelet encoding step.

Also, as mentioned in earlier chapters, we need higher RF bandwidth to reduce chemical shift misregistration effect. However, the smaller B1s available at higher fields restrict the Bandwidth available. We should look for methods that either minimize power consumption of RF pulses, or minimize the bandwidth requirements. A good alternative is presented in Goelman's paper [34] as cascaded RF pulses (figure 6.2). In this method, instead of using superposition which requires multiple bandwidths, the two RF pulses are cascaded head and tail to keep power consumption within the limit.

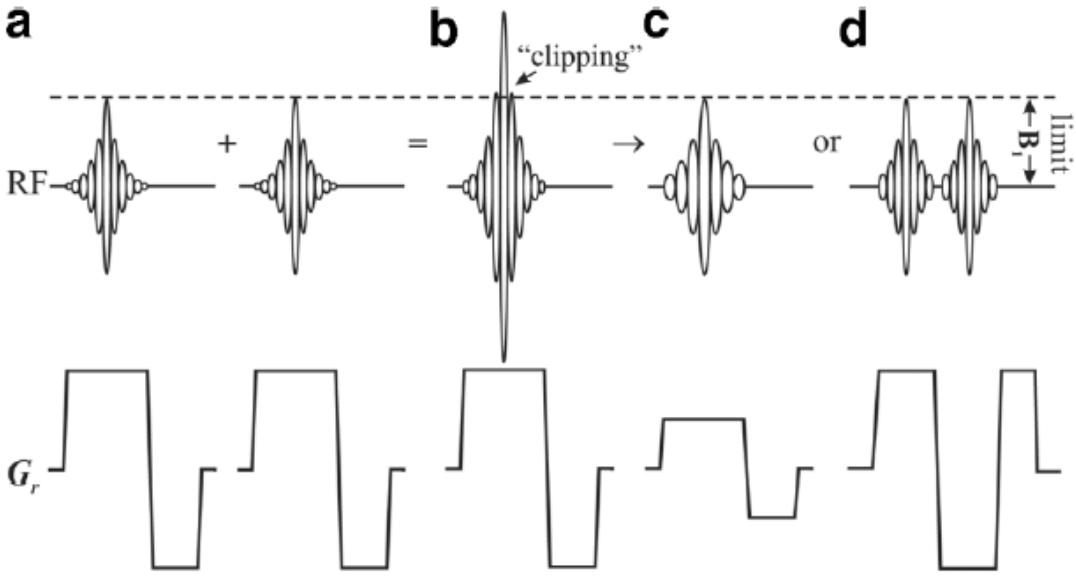


Figure 6.2 Cascaded RF pulses [34].

To reduce data reconstruction artifacts, which are the another source of voxel contamination, wavelets with smoother decay and shorter duration that are less dependent on the profiles of the RF pulses should be tested [32]. As a consequence, shorter RF pulses could be used and data with shorter echo times could be acquired, which increase data sensitivity.

References:

1. Sundgren, PC; Dong, Q; Gomez-Hassan, D; Mukherji, SK; Maly, P; Welsh, R., "Diffusion tensor imaging of the brain: review of clinical applications," *Neuroradiology* vol. 46, p.339 – 350, 2004.

2. Ashwal, S; Holshouser, BA; Tong, KA., "Use of advanced neuroimaging techniques in the evaluation of pediatric traumatic brain injury," *Developmental Neuroscience,* vol. 28, p. 309-326, 2006.

3. Hanstock, C; Faden, A; Bendall, M; Vink, R., "Diffusion-weighted imaging differentiates ischemic tissue from traumatized tissue," in *ISMRM*, Stroke 1994, vol. 25, p. 843-848.

4. Sehgal, V; Delproposto, Z; Haacke, EM; Tong, KA; Wycliffe, N; Kido, DK; Xu, Y; Neelavalli, J; Haddar, D; Reichenbach, JR., "Clinical applications of neuroimaging with susceptibility weighted imaging," *J Magn Reson Imaging,* vol. 22, p. 439-450, 2005.

5. Gruber, S.; Stadlbauer, A.; Mlynarik, V.; Gatterbauer, B.; Roessler, K.; Moser, E., "Proton magnetic resonance spectroscopic imaging in brain tumor diagnosis," in *Neurosurgery Clinics of North America,* 2005, vol. 16, (1), p.101-114,..

6. Hetherington, H. P.; Kim, J. H.; Pan, J. W.; Spencer, D. D., "H-1 and P-31 spectroscopic imaging of epilepsy: Spectroscopic and histologic correlations," in *Epilepsia* vol. 45 (4), p. 17-23, 2004.

7. McKnight, T. R., "Proton magnetic resonance spectroscopic evaluation of brain tumor metabolism," Seminars in Oncology, 2004, 31, (5), 605-617.

8. Oshio, K.; Kyriakos, W.; Mulkern, R. V., "Line scan echo planar spectroscopic imaging," *Magnetic Resonance in Medicine*, vol. 44, (4), p. 521-524, 2000.

9. Adalsteinsson, E.; Irarrazabal, P.; Topp, S.; Meyer, C.; Macovski, A.; Spielman, D. M., "Volumetric spectroscopic imaging with spiral-based k-space trajectories," *Magnetic Resonance in Medicine,* vol. 39(6), p.889-898, 1998.

10. J. Weaver, Y. Xu, D. Healy, J. Drisoll, "Wavelet-Encoded MR imaging," *Magnetic Resonance in Medicine,* vol. 24, p.275-287, 1992.

11. Young, R.; Serrai, H., "Implementation of three-dimensional wavelet encoding spectroscopic imaging: In vivo application and method comparison," *Magnetic Resonance in Medicine,* vol. 61 (1), p. 6-15, 2009.

12. E. M. Haacke, R. W. Brown, M. R. Thompson, R. Venkatesan. **Magnetic Resonance Imaging: physical principles and sequence design**. New York: Wiley-liss, 1999

13. B.P. Lathi, **Signal Processing & Linear Systems**. Oxford: Oxford University Press, 1998.

14. Bolinger, L.; Leigh Jr., J. S., "Hadamard Spectroscopic Imaging (HSI) for multi-volume localization," *Journal of Magnetic Resonance,* vol. 80, p.62-167, 1988.

15. Gonen, O.; AriasMendoza, F.; Goelman, G., "3D localized in vivo H-1 spectroscopy of human brain by using a hybrid of 1D-Hadamard with 2D chemical shift imaging," *Magnetic Resonance in Medicine*, vol. 37 (5), p. 644-650, 1997.

16. Posse, S.; Gioacchino, T.; Risinger, O.; Bihan, D., "High-speed 1H spectroscopic imaging in human brain by echo-planar spatial-spectral encoding," *Magnetic Resonance in Medicine*, vol. 37, p.644-650, 1995.

17. Duyn, J.; Moonen, C., "Fast proton spectroscopic imaging of human brain using multiple spin-echoes," *Magnetic Resonance in Medicine* vol. 30, p.409-414, 1993.

18. Hanson, L. G.; Schaumburg, K.; Paulson, O. B., “Reconstruction strategy for echo planar spectroscopy and its application to partially undersampled imaging,” *Magnetic Resonance in Medicine*, vol. 44(3), p. 412-417, 2000.

19. Adalsteinsson, E.; Irarrazabal, P.; Topp, S.; Meyer, C.; Macovski, A.; Spielman, D. M., “Volumetric spectroscopic imaging with spiral-based k-space trajectories,” *Magnetic Resonance in Medicine*, vol. 39 (6), p. 889-898, 1998.

20. Pohmann, R.; vonKienlin, M.; Haase, A., “Theoretical evaluation and comparison of fast chemical shift imaging methods,” *Journal of Magnetic Resonance*, vol. 129 (2), p.145-160, 1997.

21. Chao, H.; Bowers, J. L.; Holtzman, D.; Mulkern, R. V., “RARE imaging of PCr in human forearm muscles,” *Journal of Magnetic Resonance Imaging*, vol. 7 (6), p. 1048-1055, 1997.

22. Duyn, J.; Moonen, C., “Fast proton spectroscopic imaging of human brain using multiple spin-echoes,” *Magnetic Resonance in Medicine* 1993, 30, 409-414.

23. Greenman, R. L.; Elliott, M. A.; Vandenborne, K.; Schnall, M. D.; Lenkinski, R. E., “Fast imaging of phosphocreatine using a RARE pulse sequence,” *Magnetic Resonance in Medicine*, vol. 39 (5), p. 851-854, 1998.

24. Haase, A., Snapshot FLASH MRI. “Applications to T1, T2 and chemical-shift imaging,” *Magnetic Resonance in Medicine*, vol. 13, p. 77-89, 1990.

25. Hugg, J. W.; Maudsley, A. A.; Weiner, M. W.; Matson, G. B., “Comparison of k-space sampling schemes for multidimensional MR spectroscopic imaging,” *Magnetic Resonance in Medicine* vol. 36 (3), p. 469-473, 1996.

26. I. Daubechies, **Ten lectures on wavelet**, in CBMS. Philadelphia, PA: SIAM, 1994

27. Bottomley, “Selective volume method for performing localized NMR spectroscopy,” United States Patent 4,480,228, Oct 30, 1984

28. Haacke, E. M., In “Magnetic Resonance Imaging,” Wiley, Eds. 1999.

29. Siemens MRI Stuff, Siemens IDEA user’s Manual, Siemens.

30. J. Pauly, P. Le Roux, D. Nishimura, and A. Macovski, “Parameter Relations for theShinnar-Le Roux Selective Excitation Pulse Design Algorithm,” in *IEEE Trans. Med. Imaging*, vol. 10, p.53-65, March 1991.

31. Stephen Provencher, “LCModel & LCMgui User’s Manual,” August 2007.

32. Weaver, J.; Healy, D., “Signal-to-noise ratios and effective repetition times for wavelet encoding and encoding with wavelet packet bases,” in *Journal of Magnetic Resonance (A)*, vol. 113, p.1-10, 1995.

33. Klaas P. Pruessmann, Markus Weiger, Markus B. Scheidegger, and Peter Boesiger, “SENSE: Sensitivity Encoding for Fast MRI,” in *Magnetic Resonance in Medicine*, vol. 42, p. 952-962, 1999.

34. Gadi Goelman, Songtao Liu, and Oded Gonen, “Reducing Voxel Bleed in Hadamard-Encoded MRI and MRS,” in *Magnetic Resonance in Medicine*, vol. 55, p.1460–1465, 2006.

Appendix A

```
…
static const long  RFDuration = 5200;
static const long  RFBandwidth=2500;
static sRF_PULSE_ARB D90RFPulse("D90RFPulse");
static sFREQ_PHASE D90PhSet( "D90PhSet" );
static sFREQ_PHASE D90PhNeg( "D90PhNeg" );


static sRF_PULSE_ARB D1801RFPulse("D1801RFPulse");
static sFREQ_PHASE D1801PhSet( "D1801PhSet" );
static sFREQ_PHASE D1801PhNeg( "D1801PhNeg" );


static sRF_PULSE_ARB D1802RFPulse("D1802RFPulse");
static sFREQ_PHASE D1802PhSet( "D1802PhSet" );
static sFREQ_PHASE D1802PhNeg( "D1802PhNeg" );

static sRF_PULSE_ARB D1821RFPulse("D1821RFPulse");
static sFREQ_PHASE D1821PhSet( "D1821PhSet" );
static sFREQ_PHASE D1821PhNeg( "D1821PhNeg" );

static sRF_PULSE_ARB D1822RFPulse("D1822RFPulse");
static sFREQ_PHASE D1822PhSet( "D1822PhSet" );
static sFREQ_PHASE D1822PhNeg( "D1822PhNeg" );



static sRF_PULSE_ARB S90RFPulse("S90RFPulse");
static sFREQ_PHASE S90PhSet( "S90PhSet" );
static sFREQ_PHASE S90PhNeg( "S90PhNeg" );

static sRF_PULSE_ARB S180RFPulse("S180RFPulse");
static sFREQ_PHASE S180PhSet( "S180PhSet" );
static sFREQ_PHASE S180PhNeg( "S180PhNeg" );

static sRF_PULSE_ARB S182RFPulse("S182RFPulse");
static sFREQ_PHASE S182PhSet( "S182PhSet" );
static sFREQ_PHASE S182PhNeg( "S182PhNeg" );



…
static long ro_off_freq, ph_off_freq, sl_off_freq
```

```
static short DorS_X, DorS_Y, DorS_Z, D1801_2, n_rd, n_ph, n_sl;
static short who_is_90,Y, WAIT;


…
 *********************************************************************\
*
* Name        : fSEQInit
*
* Description : Defines the hard limits for the Seq/Change dialog.
*
* Return      : An NLS status code.
*
\*********************************************************************\


…
  pSeqLim->setBaseResolution(1,128, SEQ::INC_NORMAL,2);


  pSeqLim->setPELines( 1, 128, 1, 2);

  pSeqLim->setMaxPhaseResolution( 2 );
  pSeqLim->setPartition( 1, 128, 1, 2);

  pSeqLim->setfinalMatrixSizeRead(1, 128, SEQ::BASE2,2);         //This is RL direction
  pSeqLim->setfinalMatrixSizePhase( 1, 128, SEQ::BASE2,2);  //This is AP direction
  pSeqLim->setfinalMatrixSizeSlice( 1, 128, SEQ::BASE2,2);        // THis is HF
(equivalent to SI) direction


  ////////////////////////////////////
  // 3D dimension
  ////////////////////////////////////
  pSeqLim->setDimension( SEQ::DIM_3,SEQ::DIM_2);
  pSeqLim->setPartition( 1,128, SEQ::INC_NORMAL, 4 );
  pSeqLim->setfinalMatrixSizeSlice( 1, 128, SEQ::BASE2, 4 );
  pSeqLim->setImagesPerSlab( 1, 128, SEQ::BASE2, 4 );
  pSeqLim->setSlabThickness( 10, 300 );
  pSeqLim->set3DPartThickness( 1, 128, 1, 15);
  pSeqLim->setMinSliceResolution ( 0.5 );




…
```

```
 ****************************************************************\
*
* Name        : fSEQPrep
*
* Description : Prepares everything that the sequence needs at run time.
*
* Return      : An NLS status code.
*
\*****************************************************************


…
 RealAmpls=0;
 ImagAmpls=0;

  for (k=0; k<S90_LEN; ++k) {

    RealAmpls += pulseS90Shape[k].flAbs * cos(pulseS90Shape[k].flPha);

    ImagAmpls += pulseS90Shape[k].flAbs * sin(pulseS90Shape[k].flPha);

  }

  EffAmplIntegs=sqrt(RealAmpls*RealAmpls + ImagAmpls*ImagAmpls);



  S90RFPulse.setTypeExcitation                    ();

  S90RFPulse.setDuration                                  (RFDuration) ;

  S90RFPulse.setFlipAngle                                      (90.);

  S90RFPulse.setInitialPhase                              (0);

  S90RFPulse.setSamples                                          (S90_LEN);
  S90RFPulse.setGSAmplitude(2500.0/larmorconst / pMrProt-
>sliceSeries().front().readoutFOV());
  if (!S90RFPulse.prepArbitrary(pMrProt,pSeqExpo,pulseS90Shape,EffAmplIntegs))

    return S90RFPulse.getNLSStatus();



  // computation of the frequency offset which defines the voxel position
  // input units:
```

```
  // [GSAmplitude] = mT / m
  // [LarmorConst / (2 pi)] = MHz / T
  // [VoxelPosition] = mm
  // output unit:
  // [Frequency] = Hz

  lFrequency = (long)( .5 + S90RFPulse.getGSAmplitude() * larmorconst *
ss_slc.getSliceOffCenterRO() );

  lFrequency += (long)(pMrProt->txSpec().frequency() * 1E-6 * pMrProt-
>spectroscopy().dDeltaFrequency) ;

  ro_off_freq=lFrequency;


  // Dule 90
  for (k=0; k<D90_LEN; ++k) {

          if(k<262){
    RealAmpls += pulseD90Shape[k].flAbs * cos(pulseD90Shape[k].flPha);

    ImagAmpls += pulseD90Shape[k].flAbs * sin(pulseD90Shape[k].flPha);
          }
          else{
    RealAmpls += -pulseD90Shape[k].flAbs * cos(pulseD90Shape[k].flPha);

    ImagAmpls += -pulseD90Shape[k].flAbs * sin(pulseD90Shape[k].flPha);
          }


  }

  EffAmplIntegs=sqrt(RealAmpls*RealAmpls + ImagAmpls*ImagAmpls);



  D90RFPulse.setTypeExcitation                    ();

  D90RFPulse.setDuration                                  (RFDuration) ;

  D90RFPulse.setFlipAngle                                       (90.0);

  D90RFPulse.setInitialPhase                              (0);

  D90RFPulse.setSamples                                         (D90_LEN);
```

```
  D90RFPulse.setGSAmplitude(2500.0/larmorconst / pMrProt-
>sliceSeries().front().readoutFOV());
  if (!D90RFPulse.prepArbitrary(pMrProt,pSeqExpo,pulseD90Shape,EffAmplIntegs))

    return D90RFPulse.getNLSStatus();


//////////////////Dule 180
 RealAmpls=0;
 ImagAmpls=0;

 for (k=0; k<D1801_LEN; ++k) {
          if(k>252){
    RealAmpls += pulseD1801Shape[k].flAbs * cos(pulseD1801Shape[k].flPha);

    ImagAmpls += pulseD1801Shape[k].flAbs * sin(pulseD1801Shape[k].flPha);
          }
          else{
                RealAmpls += -pulseD1801Shape[k].flAbs *
cos(pulseD1801Shape[k].flPha);

                ImagAmpls += -pulseD1801Shape[k].flAbs *
sin(pulseD1801Shape[k].flPha);
          }

  }

  EffAmplIntegs=sqrt(RealAmpls*RealAmpls + ImagAmpls*ImagAmpls);



  D1801RFPulse.setTypeExcitation                ();

  D1801RFPulse.setDuration                        (RFDuration) ;

  D1801RFPulse.setFlipAngle                        (180.0);

  D1801RFPulse.setInitialPhase                (0);

  D1801RFPulse.setSamples                        (D1801_LEN);

  D1801RFPulse.setGSAmplitude(2500.0/larmorconst / pMrProt-
>sliceSeries().front().phaseFOV());
  if
(!D1801RFPulse.prepArbitrary(pMrProt,pSeqExpo,pulseD1801Shape,EffAmplIntegs))
```

```
   return D1801RFPulse.getNLSStatus();



////////////////////////////////Dule 182

  D1821RFPulse=D1801RFPulse;
  D1821RFPulse.setGSAmplitude(2500.0/larmorconst / pMrProt-
>sliceSeries().front().thickness());
  if
(!D1821RFPulse.prepArbitrary(pMrProt,pSeqExpo,pulseD1801Shape,EffAmplIntegs))

   return D1821RFPulse.getNLSStatus();




 RealAmpls=0;
 ImagAmpls=0;

 for (k=0; k<D1802_LEN; ++k) {
        if(k>270){
    RealAmpls += pulseD1802Shape[k].flAbs * cos(pulseD1802Shape[k].flPha);

    ImagAmpls += pulseD1802Shape[k].flAbs * sin(pulseD1802Shape[k].flPha);
        }
        else{
                RealAmpls += -pulseD1802Shape[k].flAbs *
cos(pulseD1802Shape[k].flPha);

                ImagAmpls += -pulseD1802Shape[k].flAbs *
sin(pulseD1802Shape[k].flPha);
        }

  }

  EffAmplIntegs=sqrt(RealAmpls*RealAmpls + ImagAmpls*ImagAmpls);



  D1802RFPulse.setTypeExcitation                ();

  D1802RFPulse.setDuration                              (RFDuration) ;

  D1802RFPulse.setFlipAngle                                     (180.0);
```

```
 D1802RFPulse.setInitialPhase                    (0);

 D1802RFPulse.setSamples                         (D1802_LEN);

 D1802RFPulse.setGSAmplitude(2500.0/larmorconst / pMrProt-
>sliceSeries().front().phaseFOV());
 if
(!D1802RFPulse.prepArbitrary(pMrProt,pSeqExpo,pulseD1802Shape,EffAmplIntegs))

   return D1802RFPulse.getNLSStatus();




  D1822RFPulse=D1802RFPulse;
 D1822RFPulse.setGSAmplitude(2500.0/larmorconst / pMrProt-
>sliceSeries().front().thickness());
 if
(!D1822RFPulse.prepArbitrary(pMrProt,pSeqExpo,pulseD1802Shape,EffAmplIntegs))

   return D1822RFPulse.getNLSStatus();




////////////////////////////////Single 180
RealAmpls=0;
 ImagAmpls=0;

 for (k=0; k<S180_LEN; ++k) {

    RealAmpls += pulseS180Shape[k].flAbs * cos(pulseS180Shape[k].flPha);

    ImagAmpls += pulseS180Shape[k].flAbs * sin(pulseS180Shape[k].flPha);

  }

 EffAmplIntegs=sqrt(RealAmpls*RealAmpls + ImagAmpls*ImagAmpls);
```

```
 S180RFPulse.setTypeExcitation                    ();

 S180RFPulse.setDuration                                (RFDuration) ;

 S180RFPulse.setFlipAngle                                 (180);

 S180RFPulse.setInitialPhase                        (0);

 S180RFPulse.setSamples                                  (S180_LEN);
 S180RFPulse.setGSAmplitude(2500/larmorconst / pMrProt-
>sliceSeries().front().phaseFOV());
 if (!S180RFPulse.prepArbitrary(pMrProt,pSeqExpo,pulseS180Shape,EffAmplIntegs))

   return S180RFPulse.getNLSStatus();



 lFrequency = (long)( .5 + S180RFPulse.getGSAmplitude() * larmorconst *
ss_slc.getSliceOffCenterPE() );
 lFrequency += (long)(pMrProt->txSpec().frequency() * 1E-6 * pMrProt-
>spectroscopy().dDeltaFrequency) ;

 ph_off_freq=lFrequency;

////////////////////////////S 182

         S182RFPulse=S180RFPulse;
         S182RFPulse.setGSAmplitude(2500.0/larmorconst / pMrProt-
>sliceSeries().front().thickness());
          if
(!S182RFPulse.prepArbitrary(pMrProt,pSeqExpo,pulseS180Shape,EffAmplIntegs))

   return S182RFPulse.getNLSStatus();


 lFrequency = (long)( .5 + S182RFPulse.getGSAmplitude() * larmorconst *
ss_slc.getSliceShift() );
 lFrequency += (long)(pMrProt->txSpec().frequency() * 1E-6 * pMrProt-
>spectroscopy().dDeltaFrequency) ;

 sl_off_freq=lFrequency;
```

```
/*[ Function
*************************************************************************\
*
* Name        : fSEQCheck
*
* Description : Checks the real-time sequence for gradient overflows.
*
* Return      : An NLS status code.
*
\*************************************************************************

…
*************************************************************************\
*
* Name        : fSEQRun
*
* Description : Executes the real-time sequence.
*
* Return      : An NLS status code.
*
\*************************************************************************


…


  double x_grad=2500.0/larmorconst / pMrProt->sliceSeries().front().readoutFOV(),
         z_grad=2500.0/larmorconst / pMrProt->sliceSeries().front().thickness(),
         y_grad=2500.0/larmorconst / pMrProt->sliceSeries().front().phaseFOV();
  long                    n_sl ,n_ph, n_rd;
  long       scale_sl, scale_rd, scale_ph, scale_1,scale_2,scale_3, xii,xjj,xkk, myflag, temp;



  n_rd = pMrProt->kSpace().baseResolution();

  n_ph = pMrProt->kSpace().phaseEncodingLines();

  n_sl = pMrProt->kSpace().partitions();


…
```

```
  ////////////////////////////////////////////////////////////
  // execute repetition loop
  ////////////////////////////////////////////////////////////

  n_rep = pMrProt->repetitions() + 1;
  for( k=0; k<n_rep; k++ ){

              short chkboard[32*32*32];
              int ii;
              for (ii=0; ii<32*32*32; ii++)
                        chkboard[ii]=0;

              ss_adc1.Mdh.setCrep( k );


  ////////////////////////////////////////////////////////////
  // execute prepare loop
  ////////////////////////////////////////////////////////////

  fRTSetReadoutEnable( 0 );  // disable ADC events
    for( i=0; i<n_prep; i++ ){


….
  ////////////////////////////////////////////////////////////
  // execute acquisition loop
  ////////////////////////////////////////////////////////////


  ////////////////////added Jan 2008/////////////////////
long lfrequency;
double dPhase;

  for (j=0; j<n_rd*n_ph*n_sl; j++){
              who_is_90=1;
              short found, ii,jj,kk;
              for (chk=0; chk<n_rd*n_ph*n_sl;chk++){
                        found=0;
                        if(chkboard[chk]==0){
                                  j=chk;
                                  found=1;
                                  break;
                        }

              }
```

```
        if(found==0){

                continue;
        }

        int count_rd=j%n_rd+1;
        int count_ph=(j/n_rd)%n_ph+1;
        int count_sl=j/n_rd/n_ph+1;
int cnt;



        scale_rd=1;
        scale_ph=1;
        scale_sl=1;
        scale_1=1;
        scale_2=1;
        scale_3=1;
        myflag=1;


        cnt=2;
        while(cnt<32){
                if(count_rd>cnt && count_rd<=2*cnt)
                        scale_rd=cnt;
                if(count_ph>cnt && count_ph<=2*cnt)
                        scale_ph=cnt;
                if(count_sl>cnt && count_sl<=2*cnt)
                        scale_sl=cnt;
                cnt=cnt*2;
        }

if(scale_rd>=scale_sl && scale_rd>=scale_ph){
                Y=scale_rd;
                who_is_90=1;
                if(scale_ph>=scale_sl){
                        myflag=1;
                   scale_1=scale_rd;
                        scale_2=scale_ph;
                        scale_3=scale_sl;
                        ii=count_rd;
                        jj=count_ph;
                        kk=count_sl;
                }
                else{
                        myflag=2;
```

```
                scale_1=scale_rd;
                scale_2=scale_sl;
                scale_3=scale_ph;
                ii=count_rd;
                jj=count_sl;
                kk=count_ph;
        }
}

if(scale_ph>scale_rd && scale_ph>=scale_sl){
        Y=scale_ph;
        who_is_90=2;
        if(scale_rd>=scale_sl){
                myflag=3;
           scale_1=scale_ph;
                scale_2=scale_rd;
                scale_3=scale_sl;
                ii=count_ph;
                jj=count_rd;
                kk=count_sl;
        }
        else{
                myflag=4;
                scale_1=scale_ph;
                scale_2=scale_sl;
                scale_3=scale_rd;
                ii=count_ph;
                jj=count_sl;
                kk=count_rd;
        }
}

if(scale_sl>scale_ph && scale_sl>scale_rd){
        Y=scale_sl;
        who_is_90=3;
        if(scale_ph>=scale_sl){
                myflag=5;
           scale_1=scale_sl;
                scale_2=scale_rd;
                scale_3=scale_ph;
                ii=count_sl;
                jj=count_rd;
                kk=count_ph;
        }
        else{
                myflag=6;
```

```
                        scale_1=scale_sl;
                        scale_2=scale_ph;
                        scale_3=scale_rd;
                        ii=count_sl;
                        jj=count_ph;
                        kk=count_rd;
                    }
                }

                //  the following two lines turns off the pulse switching functionality

//          who_is_90=1;
//          Y=1;
////////////////////////////////////////////////////////////////////////
        for (cnt=0; cnt<Y; cnt++){
                int p,q,r;
                    WAIT=Y==(cnt+1)?1:0;
            if(scale_1>1){
                        p= (ii + cnt)%scale_1;
                        p= p==0?scale_1:p;
                        p= p+scale_1;
                    }
                    else
                        p=ii;

                    if(scale_3>1){
                        q= (jj + cnt)%scale_3;
                        q= q==0?scale_3:q;
                    //          q=q+cnt/scale_3*scale_3;
                        q=q%scale_2;
                        q= q==0?scale_2:q;
                        q= q+scale_2;
                    }
                    else
                        q=jj;

                    if(scale_3>1){
                        r= (kk + cnt)%scale_3;
                        r= r==0?scale_3:r;
                        r= r+scale_3;
                    }
                    else
                        r=kk;

                    if(myflag==2){
                        temp=q;
```

```
                q=r;
                r=temp;
            }
            if(myflag==3){
                temp=p;
                p=q;
                q=temp;
            }
            if(myflag==4){
                temp=p;
                p=r;
                r=q;
                q=temp;
            }
            if(myflag==5){
                temp=p;
                p=q;
                q=r;
                r=temp;
            }
            if(myflag==6){
                temp=p;
                p=r;
                r=temp;
            }
if(chkboard[p-1+(q-1)*n_rd+(r-1)*n_rd*n_ph]){

              if(myflag==1)
          q=q+scale_3;

        if(myflag==2)
          r=r+scale_3;

        if(myflag==3)
          p=p+scale_3;

        if(myflag==4)
          r=r+scale_3;

        if(myflag==5)
          p=p+scale_3;

        if(myflag==6)
          q=q+scale_3;
            }
if(chkboard[p-1+(q-1)*n_rd+(r-1)*n_rd*n_ph])
```

```
                cout<<"ordering Error !!!"<<endl;

            count_rd=p;
            count_ph=q;
            count_sl=r;

            cout << "==>p,q,r,y:                    " << p<<q<<r<<Y <<
endl;
            chkboard[p-1+(q-1)*n_rd+(r-1)*n_rd*n_ph]=1;









    if(who_is_90==1){
                DorS_X=(count_rd==1)?1:0;
                DorS_Y=(count_ph==1)?1:0;
                DorS_Z=(count_sl==1)?1:0;          //1--single 0---dule

                D90RFPulse.setInitialPhase                  (0);
                D1801RFPulse.setInitialPhase                (0);
                D1802RFPulse.setInitialPhase                (0);
                D1821RFPulse.setInitialPhase                (0);
                D1822RFPulse.setInitialPhase                (0);

                D90RFPulse.setGSAmplitude(scale_rd*x_grad);
                ss_grad_exc.setAmplitude(scale_rd*x_grad);
                ss_grad_ref.setAmplitude(11.5- ((0.515*5200+0.5*800 )
*scale_rd*x_grad / 4000 ));
                D1801RFPulse.setGSAmplitude(scale_ph*y_grad);
                D1802RFPulse.setGSAmplitude(scale_ph*y_grad);
                if(count_ph>1)
                    ss_grad_pi_ph.setAmplitude(scale_ph*y_grad);
                else
                    ss_grad_pi_ph.setAmplitude(scale_ph*y_grad/1.25);


                D1821RFPulse.setGSAmplitude(scale_sl*z_grad);
```

```
                        D1822RFPulse.setGSAmplitude(scale_sl*z_grad);
                        if(count_sl>1)
                                ss_grad_pi_sl.setAmplitude(scale_sl*z_grad);
                        else
                                 ss_grad_pi_sl.setAmplitude(scale_sl*z_grad/1.25);

                        D1822RFPulse.setInitialPhase                                (0);
//                      lfrequency= ro_off_freq*scale_rd;
//                      cout<<"lfreq"<<lfrequency<<endl;
                        lfrequency =  D90RFPulse.getGSAmplitude() * larmorconst *
ss_slc.getSliceOffCenterRO()-335 ;


                        if(count_rd>2)
                                lfrequency+=(-3.0/2*scale_rd+count_rd-
0.5)*RFBandwidth;
                        lfrequency=  (long)( .5 +lfrequency);
                    lfrequency += (long)(pMrProt->txSpec().frequency() * 1E-6 *
pMrProt->spectroscopy().dDeltaFrequency) ;

                        if(count_rd==1){

                    S90PhSet.setFrequency( lfrequency );
                        S90PhNeg.setFrequency( 0L );

                    dPhase = - lfrequency * (360./1e6) * 2600;
                        S90PhSet.setPhase( dPhase );
                S90PhNeg.setPhase( - lfrequency * (360./1e6) * 2600 );
                        }

                        D90PhSet.setFrequency( lfrequency );
                        D90PhNeg.setFrequency( 0L );
                        dPhase = -lfrequency * (360./1e6) * 2600;
                        D90PhSet.setPhase( dPhase );
                        D90PhNeg.setPhase( -lfrequency * (360./1e6) * 2600 );



//                      lfrequency= ph_off_freq*scale_ph;
                        cout << "x_freq  " <<lfrequency<<endl;
                        lfrequency = D1801RFPulse.getGSAmplitude() * larmorconst *
ss_slc.getSliceOffCenterPE()-335 ;

//                      cout<<"lFreq  "<<lfrequency<<endl;
                        if(count_ph>2)
```

```
                                        lfrequency+=(-3.0/2*scale_ph+count_ph-
0.5)*RFBandwidth;
                        lfrequency=  (long)( .5 +lfrequency);
                    lfrequency += (long)(pMrProt->txSpec().frequency() * 1E-6 *
pMrProt->spectroscopy().dDeltaFrequency) ;

                        if(count_ph==1){
                                S180PhSet.setFrequency( long(ph_off_freq/1.25-335) );
                                S180PhNeg.setFrequency( 0L );

                                dPhase = - long(ph_off_freq/1.25-335) * (360./1e6) *
S180RFPulse.getDuration() * 0.5;
                                S180PhSet.setPhase( dPhase );
                              S180PhNeg.setPhase( dPhase );
                                 cout << "y_freq:S180  "
<<S180PhSet.getFrequency()<<endl;
                        }
                        D1801PhSet.setFrequency( lfrequency );
                        D1801PhNeg.setFrequency( 0L );
                        D1802PhSet.setFrequency( lfrequency );
                        D1802PhNeg.setFrequency( 0L );

                        dPhase = -lfrequency * (360./1e6) * 2600;
                        D1801PhSet.setPhase( dPhase);
                    D1801PhNeg.setPhase(dPhase);
                    D1802PhSet.setPhase( dPhase );
                    D1802PhNeg.setPhase( dPhase);



//                      lfrequency= sl_off_freq*scale_sl;
                       cout<<"y_freq  "<<lfrequency<<endl;
                       lfrequency = D1821RFPulse.getGSAmplitude() * larmorconst *
ss_slc.getSliceShift()-335 ;

//                      cout<<"lFreq  "<<lfrequency<<endl;
                        if(count_sl>2)
                                        lfrequency+=(-3.0/2*scale_sl+count_sl-
0.5)*RFBandwidth;

                       lfrequency=  (long)( .5 +lfrequency);
                  lfrequency += (long)(pMrProt->txSpec().frequency() * 1E-6 * pMrProt-
>spectroscopy().dDeltaFrequency) ;
                        cout<<"z_freq  "<<lfrequency<<endl;

                        if(count_sl==1){
```

```
                        S182PhSet.setFrequency( long(sl_off_freq/1.25-335)  );
                            S182PhNeg.setFrequency( 0L );

                            dPhase = - long(sl_off_freq/1.25-335)* (360./1e6) *
S182RFPulse.getDuration() * 0.5;
                            S182PhSet.setPhase( dPhase );
                            S182PhNeg.setPhase( dPhase );
                            cout<<"z_freq  S182
"<<S182PhSet.getFrequency()<<endl;
                        }
                        D1821PhSet.setFrequency( lfrequency );
/*! EGA-05 !*/
                        D1821PhNeg.setFrequency( 0L );
                        D1822PhSet.setFrequency( lfrequency );
/*! EGA-05 !*/
                        D1822PhNeg.setFrequency( 0L );
                            dPhase = -lfrequency * (360./1e6) * 2600;
                        D1821PhSet.setPhase( dPhase);
                        D1821PhNeg.setPhase( dPhase);
                        D1822PhSet.setPhase( dPhase );
                        D1822PhNeg.setPhase( dPhase );
                }

                if(who_is_90==2){
                        DorS_X=(count_ph==1)?1:0;
                   DorS_Y=(count_rd==1)?1:0;
                        DorS_Z=(count_sl==1)?1:0;          //1--single 0---dule



                        S90RFPulse.setInitialPhase
                                    (0);
                        S180RFPulse.setInitialPhase         (0);

                        D90RFPulse.setInitialPhase                              (0);
                        D1801RFPulse.setInitialPhase                              (0);
                        D1802RFPulse.setInitialPhase                              (0);
                        D1821RFPulse.setInitialPhase                              (0);
                        D1822RFPulse.setInitialPhase                              (0);


                        D90RFPulse.setGSAmplitude(scale_ph*y_grad);
                        ss_grad_exc.setAmplitude(scale_ph*y_grad);
                        ss_grad_ref.setAmplitude(11.5- ((0.515*5200+0.5*800 )
*scale_ph*y_grad / 4000 ));
```

```
                    D1801RFPulse.setGSAmplitude(scale_rd*x_grad);
                    D1802RFPulse.setGSAmplitude(scale_rd*x_grad);
                    if(count_rd>1)
                    ss_grad_pi_ph.setAmplitude(scale_rd*x_grad);
                    else

        ss_grad_pi_ph.setAmplitude(scale_rd*x_grad/1.25);

                    D1821RFPulse.setGSAmplitude(scale_sl*z_grad);
                    D1822RFPulse.setGSAmplitude(scale_sl*z_grad);
                    if(count_sl>1)
                    ss_grad_pi_sl.setAmplitude(scale_sl*z_grad);
                    else
                        ss_grad_pi_sl.setAmplitude(scale_sl*z_grad/1.25);
```

```
        lfrequency = D90RFPulse.getGSAmplitude() * larmorconst *
ss_slc.getSliceOffCenterPE()-335 ;
                    if(count_ph>2)
                        lfrequency+=(-3.0/2*scale_ph+count_ph-
0.5)*RFBandwidth;
                    lfrequency=  (long)( .5 +lfrequency);
                lfrequency += (long)(pMrProt->txSpec().frequency() * 1E-6 *
pMrProt->spectroscopy().dDeltaFrequency) ;

                    D90PhSet.setFrequency( lfrequency );
                    D90PhNeg.setFrequency( 0L );
                    dPhase = -lfrequency * (360./1e6) * 2600;
                    D90PhSet.setPhase( dPhase );
                    D90PhNeg.setPhase(  -lfrequency*(360./1e6) * 2600);
                    if(count_ph==1){

                S90PhSet.setFrequency( lfrequency );
                    S90PhNeg.setFrequency( 0L );
```

```
                    dPhase = - lfrequency * (360./1e6) * 2600;
                        S90PhSet.setPhase( dPhase );
                S90PhNeg.setPhase( - lfrequency * (360./1e6) * 2600 );
                        }
                        cout<<"pulse switched        "<<"y_freq:"<<lfrequency<<endl;
                        lfrequency = D1801RFPulse.getGSAmplitude() * larmorconst *
ss_slc.getSliceOffCenterRO() -335;

                        if(count_rd>2)
                                lfrequency+=(-3.0/2*scale_rd+count_rd-
0.5)*RFBandwidth;
                        lfrequency=  (long)( .5 +lfrequency);
                    lfrequency += (long)(pMrProt->txSpec().frequency() * 1E-6 *
pMrProt->spectroscopy().dDeltaFrequency) ;
                        D1801PhSet.setFrequency( lfrequency );
                        D1801PhNeg.setFrequency( 0L );
                        D1802PhSet.setFrequency( lfrequency );
                        D1802PhNeg.setFrequency( 0L );

                        dPhase = -lfrequency * (360./1e6) * 2600;
                        D1801PhSet.setPhase( dPhase);
                    D1801PhNeg.setPhase( dPhase );
                    D1802PhSet.setPhase( dPhase );
                    D1802PhNeg.setPhase( dPhase);

                        if(count_rd==1){
                            S180PhSet.setFrequency( long(ro_off_freq/1.25-335)  );
                            S180PhNeg.setFrequency( 0L );

                            dPhase = - long(ro_off_freq/1.25-335) * (360./1e6) *
S180RFPulse.getDuration() * 0.5;
                            S180PhSet.setPhase( dPhase );
                          S180PhNeg.setPhase( dPhase );
                            cout << "pulse swiched,x_freq:S180  "
<<S180PhSet.getFrequency()<<endl;
                        }

                        cout<<"pulse switched        "<<"x_freq:"<<lfrequency<<endl;
                        lfrequency = D1821RFPulse.getGSAmplitude() * larmorconst *
ss_slc.getSliceShift()-335;
                        if(count_sl>2)
                                lfrequency+=(-3.0/2*scale_sl+count_sl-
0.5)*RFBandwidth;
                        lfrequency=  (long)( .5 +lfrequency);
```

```
                    lfrequency += (long)(pMrProt->txSpec().frequency() * 1E-6 *
pMrProt->spectroscopy().dDeltaFrequency) ;

                    D1821PhSet.setFrequency( lfrequency );
/*! EGA-05 !*/
                    D1821PhNeg.setFrequency( 0L );
                    D1822PhSet.setFrequency( lfrequency );
/*! EGA-05 !*/
                    D1822PhNeg.setFrequency( 0L );
                    dPhase = -lfrequency * (360./1e6) * 2600;
                    D1821PhSet.setPhase( dPhase);
                    D1821PhNeg.setPhase( dPhase);
                    D1822PhSet.setPhase( dPhase );
                    D1822PhNeg.setPhase( dPhase );
                    if(count_sl==1){

                      S182PhSet.setFrequency( long(sl_off_freq /1.25-335) );
                        S182PhNeg.setFrequency( 0L );

                        dPhase = -  long(sl_off_freq /1.25-335)  * (360./1e6) *
S182RFPulse.getDuration() * 0.5;
                        S182PhSet.setPhase( dPhase );
                        S182PhNeg.setPhase( dPhase );
                        cout<<"pulse switched          "<<"z_freq:
S182"<<S182PhSet.getFrequency()<<endl;
                    }
                    cout<<"pulse switched        "<<"z_freq:"<<lfrequency<<endl;
                }

                if(who_is_90==3){
                    DorS_X=(count_sl==1)?1:0;
                    DorS_Y=(count_ph==1)?1:0;
                    DorS_Z=(count_rd==1)?1:0;                    //1--single 0---dule

                    ss_grad_exc.setAmplitude(scale_sl*z_grad);
                    ss_grad_ref.setAmplitude(11.5- ((0.515*5200+0.5*800 )
*scale_sl*z_grad / 4000 ));
                    if(count_ph>1)
                    ss_grad_pi_ph.setAmplitude(scale_ph*y_grad);
                    else
                        ss_grad_pi_ph.setAmplitude(scale_ph*y_grad/1.25);
                    if(count_rd>1)
                    ss_grad_pi_sl.setAmplitude(scale_rd*x_grad);
                    else
                                ss_grad_pi_sl.setAmplitude(scale_rd*x_grad/1.25);
```

```
                    lfrequency= sl_off_freq*scale_sl-335;
                    if(count_sl>2)
                        lfrequency+=(-3.0/2*scale_sl+count_sl-
0.5)*RFBandwidth;

                    D90PhSet.setFrequency( lfrequency );
                    D90PhNeg.setFrequency( 0L );
                    dPhase = -lfrequency * (360./1e6) * 2600;
                    D90PhSet.setPhase( dPhase );
                    D90PhNeg.setPhase( -lfrequency*(360./1e6) * 2600 );
                    if(count_sl==1){

                S90PhSet.setFrequency( lfrequency);
                    S90PhNeg.setFrequency( 0L );

                dPhase = -lfrequency * (360./1e6) * 2600;
                    S90PhSet.setPhase( dPhase );
            S90PhNeg.setPhase( dPhase );
                    }

                     cout << "pulse swiched to 3: z_freq" <<lfrequency<<endl;
                    lfrequency= ph_off_freq*scale_ph-335;

                    if(count_ph>2)
                                lfrequency+=(-3.0/2*scale_ph+count_ph-
0.5)*RFBandwidth;
                    D1801PhSet.setFrequency( lfrequency );
                    D1801PhNeg.setFrequency( 0L );
                    D1802PhSet.setFrequency( lfrequency );
                    D1802PhNeg.setFrequency( 0L );

                    dPhase = -lfrequency * (360./1e6) * 2600;
                    D1801PhSet.setPhase( dPhase);
                D1801PhNeg.setPhase( dPhase );
                D1802PhSet.setPhase( dPhase );
                D1802PhNeg.setPhase( dPhase );

                    if(count_ph==1){
                      S180PhSet.setFrequency( long(ph_off_freq /1.25-335) );
                        S180PhNeg.setFrequency( 0L );

                        dPhase = - long(ph_off_freq /1.25-335) * (360./1e6) *
S180RFPulse.getDuration() * 0.5;
                        S180PhSet.setPhase( dPhase );
                      S180PhNeg.setPhase( dPhase );
```

```
                                cout << "pulse swiched to 3:y_freq, S180 " <<
S180PhSet.getFrequency() <<endl;
                        }
                cout << "pulse swiched to 3: y_freq " << lfrequency<<endl;
                        lfrequency= ro_off_freq*scale_rd-335;

                        if(count_rd>2)
                                lfrequency+=(-3.0/2*scale_rd+count_rd-
0.5)*RFBandwidth;


                        D1821PhSet.setFrequency( lfrequency );
/*! EGA-05 !*/
                        D1821PhNeg.setFrequency( 0L );
                        D1822PhSet.setFrequency( lfrequency );
/*! EGA-05 !*/
                        D1822PhNeg.setFrequency( 0L );
                        dPhase = -lfrequency * (360./1e6) * 2600;
                        D1821PhSet.setPhase( dPhase);
                        D1821PhNeg.setPhase(  dPhase);
                        D1822PhSet.setPhase( dPhase );
                        D1822PhNeg.setPhase( dPhase );
                        if(count_rd==1){

                          S182PhSet.setFrequency( long(ro_off_freq/1.25-335) );
                                S182PhNeg.setFrequency( 0L );

                                dPhase = -  long(ro_off_freq/1.25-335)* (360./1e6) *
S182RFPulse.getDuration() * 0.5;
                                S182PhSet.setPhase( dPhase );
                                S182PhNeg.setPhase( dPhase );
                                        cout << "pulse swiched to 3:x_freq, S182 " <<
S182PhSet.getFrequency() <<endl;
                        }
                       cout << "pulse swiched to 3: x_freq " << lfrequency<<endl;
                }

                for( i=0; i<n_ave; i++ ){
                        D1801_2=k%2;
                        //D1801_2=0;
                        ss_adc1.Mdh.setCset( i ); // averages


                        ss_adc1.Mdh.setClin( count_rd-1 );
                        ss_adc1.Mdh.setCphs( count_ph-1 );
                        ss_adc1.Mdh.setCseg( count_sl-1 );
```

```
                //          ss_adc1.Mdh.setFirstScanInSlice( !i && !j );
                //          ss_adc1.Mdh.setLastScanInSlice( j==(n_rd*n_ph*n_sl)-1 &&
i==(n_ave-1) );



                            lStatus = fSEQRunKernel( pMrProt, pSeqLim, pSeqExpo,
KERNEL_CHECK );
                            CheckStatusPR(lStatus,"fSEQRunKernel");
                  }

                }
        }


  if( k < (n_rep-1) ){
          CheckStatusPB ( lStatus = fSBBMeasRepetDelaysRun( pMrProt, pSeqLim,
pSeqExpo, k ),"fSBBMeasRepetDelaysRun" );

  }

  } // end repetition loop


…


/*[ Function
*************************************************************************\
*
* Name       : fSEQRunKernel
*
* Description : Executes the basic timing of the real-time sequence.
*             This function is called by the function (libSBB)fSEQRunStd.
*
* Return     : An NLS status code.
*
\*************************************************************************
```

```
…

/************************************ S E Q U E N C E   T I M I N G
************************************/
/*          Start Time      |    NCO    | SRF  | ADC  |    Gradient Events    | Sync
*/
/*            (usec)        |   Event   | Event | Event | phase | read  | slice | Event             */
/*fRTEI(                     ,         ,      ,      ,      ,      ,       );   [ Clock]*/
/*************************************************************************
….


  if(who_is_90==1)
   fRTEI(lT,   0, 0, /*A*/ 0, 0,                                                    &ss_grad_exc,0,0);
  if(who_is_90==2)
          fRTEI(lT,   0, 0, /*A*/ 0,
&ss_grad_exc,0,0,0);
  if(who_is_90==3)
           fRTEI(lT,   0, 0, /*A*/ 0, 0,
0,&ss_grad_exc,0);


  if(!DorS_X){
          fRTEI(lT+= (ss_grad_exc.getDuration() - D90RFPulse.getDuration()),
&D90PhSet, &D90RFPulse,0,/*A*/0,0,0,0);
          if(who_is_90==1)
          fRTEI(lT+= (D90RFPulse.getDuration()),
&D90PhNeg,0,/*A*/0,&ss_sp1_ph,&ss_grad_ref,&ss_sp1_sl,0);
          if(who_is_90==2)
                    fRTEI(lT+= (D90RFPulse.getDuration()),
&D90PhNeg,0,/*A*/0,&ss_grad_ref,&ss_sp1_ph,&ss_sp1_sl,0);
          if(who_is_90==3)
                    fRTEI(lT+= (D90RFPulse.getDuration()),
&D90PhNeg,0,/*A*/0,&ss_sp1_ph,&ss_sp1_sl,&ss_grad_ref,0);



  }
  else{
          fRTEI(lT+= (ss_grad_exc.getDuration() - S90RFPulse.getDuration()),
&S90PhSet, &S90RFPulse,0,/*A*/0,0,0,0);
          if(who_is_90==1)
          fRTEI(lT+= (S90RFPulse.getDuration()),
&S90PhNeg,0,/*A*/0,&ss_sp1_ph,&ss_grad_ref,&ss_sp1_sl,0);
          if(who_is_90==2)
```

```
                fRTEI(lT+= (S90RFPulse.getDuration()),
&S90PhNeg,0,/*A*/0,&ss_grad_ref,&ss_sp1_ph,&ss_sp1_sl,0);
        if(who_is_90==3)
                fRTEI(lT+= (S90RFPulse.getDuration()),
&S90PhNeg,0,/*A*/0,&ss_sp1_ph,&ss_sp1_sl,&ss_grad_ref,0);


 }

 // slice select rephasing, 1st refocussing pulse


 if(who_is_90==2)
          fRTEI(lT+= (ss_sp1_sl.getDuration()), 0,0,/*A*/0,0,&ss_grad_pi_ph, 0,0);
 else
         fRTEI(lT+= (ss_sp1_sl.getDuration()), 0,0,/*A*/0,&ss_grad_pi_ph, 0,0,0);



 if(!DorS_Y){
          if(D1801_2){
        fRTEI(lT+= (ss_grad_pi_ph.getRampUpTime()), &D1801PhSet,
&D1801RFPulse, 0,0,0,0,0 );
        fRTEI(lT+= (RFDuration), &D1801PhNeg, 0,0,
&ss_sp1_ph,&ss_sp1_ro,&ss_sp1_sl,0);
          }
          else{
        fRTEI(lT+= (ss_grad_pi_ph.getRampUpTime()), &D1802PhSet,
&D1802RFPulse, 0,/*A*/0,0,0,0 );
        fRTEI(lT+= (RFDuration), &D1802PhNeg, 0,/*A*/0,
&ss_sp1_ph,&ss_sp1_ro,&ss_sp1_sl,0);
          }

 }
 else{
        fRTEI(lT+= (ss_grad_pi_ph.getRampUpTime()), &S180PhSet, &S180RFPulse,
0,/*A*/0,0,0,0 );
        fRTEI(lT+= (RFDuration), &S180PhNeg, 0,/*A*/0,
&ss_sp1_ph,&ss_sp1_ro,&ss_sp1_sl,0);
 }


 // 2nd refocussing pulse

 fRTEI(lT+= (ss_sp1_sl.getDuration() + ss_sp1_sl.getRampDownTime() + sl_trueTE1),
                                0,0,/*A*/0,&ss_sp2_ph,&ss_sp2_ro,&ss_sp2_sl,0);
```

```
 // hier wird der 2.Spoiler angeschaltet
 if(who_is_90==3)
   fRTEI(lT+= (ss_sp2_sl.getDuration()), 0,0,/*A*/0,0,&ss_grad_pi_sl,0, 0);
 else
        fRTEI(lT+= (ss_sp2_sl.getDuration()), 0,0,/*A*/0,0,0,&ss_grad_pi_sl, 0);
 if(!DorS_Z){
          if(D1801_2){
        fRTEI(lT+= (ss_grad_pi_sl.getRampUpTime()), &D1821PhSet,
&D1821RFPulse, 0,0,0,0,0 );
        fRTEI(lT+= (RFDuration), &D1821PhNeg, 0,0,
&ss_sp2_ph,&ss_sp2_ro,&ss_sp2_sl,0);
          }
          else{
        fRTEI(lT+= (ss_grad_pi_sl.getRampUpTime()), &D1822PhSet,
&D1822RFPulse, 0,/*A*/0,0,0,0 );
        fRTEI(lT+= (RFDuration), &D1822PhNeg, 0,/*A*/0,
&ss_sp2_ph,&ss_sp2_ro,&ss_sp2_sl,0);
          }
        }
 else{
        fRTEI(lT+= (ss_grad_pi_sl.getRampUpTime()), &S182PhSet, &S182RFPulse,
0,/*A*/0,0,0,0 );
   fRTEI(lT+= (RFDuration), &S182PhNeg, 0,/*A*/0,
&ss_sp2_ph,&ss_sp2_ro,&ss_sp2_sl,0);
 }
 // acquisition

 //fRTEI(lT+= (ss_sp2_sl.getDuration() + ss_sp2_sl.getRampDownTime() + sl_trueTE2
 //            /* -  pMrProt->spectroscopy().acquisitionDelay()*/ ),
&ss_ph_s_adc,0,&ss_adc1,0,0,0,0);
 fRTEI(lT+= (ss_sp2_sl.getDuration() + ss_sp2_sl.getRampDownTime() +
sl_aqu_fill_before ), &ss_ph_s_adc,0,&ss_adc1,0,0,0,0);


 // final spoiling

 // fRTEI(lT+=(1000 +
ss_adc1.getRoundedDuration(GRAD_RASTER_TIME)),&ss_ph_n_adc,0,0,
&ss_finsp_ro, &ss_finsp_ph, &ss_finsp_sl, 0 );
 fRTEI(lT+=(1000 + ss_adc1.getRoundedDuration(GRAD_RASTER_TIME) +
sl_aqu_fill_after),&ss_ph_n_adc,0,0, &ss_finsp_ro, &ss_finsp_ph, &ss_finsp_sl, 0 );
 fRTEI(lT+=(ss_finsp_sl.getDuration() + ss_finsp_sl.getRampDownTime()),
0,0,0,0,0,0,0 );
```

```
  if(WAIT){
  //        if(Y<2)
                    dt=pMrProt->tr()[0]-lT-lTextra;
//          else
//                    dt=pMrProt->tr()[0]-lT*(Y-1)-lTextra;
//          dt=dt>0?dt:0;
            fRTEI(lT+=  dt, 0,0,0,0,0,0,0);
  }


  ////////////////////////////////////////////////////////////////////////////////
  // do testing and close the event block
  ////////////////////////////////////////////////////////////////////////////////

  mSEQTest(pMrProt, pSeqLim, pSeqExpo, RTEB_ClockCheck, 10, 0 /*lLine*/,
0/*lSliceIndex*/, 0, 0) ;
  mSEQTest(pMrProt, pSeqLim, pSeqExpo, ulTestIdent    , 10, 0/*lLine*/,
0/*lSliceIndex*/, 0, 0) ;
  CheckStatusPB(lStatus = fRTEBFinish(),"fRTEBFinish [*0010*]");

FINISHED:

  return(lStatus);
}
```

Appendix B WESI Reconstruction Code

```
nx=4;
ny=4;
nz=1;
average=16;
vector_size=1024;
TE=75; % ms
ADCBandwidth=2000; %Hz

[filename , pathname ] = uigetfile('*.dat', 'Select an DAT file')
filename=[pathname, filename];
fid = fopen(filename,'r');

  SI=WLCons(90, nx);
  SI=fliplr(SI);

  VI=WLCons(180,ny);
  V2I=WLCons(180,nz);

bytes_to_skip = fread(fid, 1, 'uint32');
fseek(fid,bytes_to_skip,'bof');


A = fread(fid,'float32');
samplesbeforeecho=floor((0.5*TE*1000-1.5*5200-4800-4800-
300)*ADCBandwidth/1000000)+1;
no_samples=ceil((samplesbeforeecho+vector_size+8)/16)*16 ;
fillup=no_samples-samplesbeforeecho-vector_size-8;
no_samples=no_samples+16
AA=A(33:length(A));
B=zeros(1,2*nx*ny*nz*vector_size*average);
offset=1;
for (i=1:nx*ny*nz*average)

    temp=AA((i-1)*2*(no_samples)+1+samplesbeforeecho*2:(i-
1)*2*(no_samples)+vector_size*2+samplesbeforeecho*2);
    B(offset:offset+vector_size*2-1)=temp;
    offset=offset+vector_size*2;
end

N= B(1:2:length(B))+sqrt(-1)*B(2:2:length(B));

no_samples=vector_size;
M=reshape(N,no_samples,nx*ny*nz,average);



M=mean(M,3);



M=reshape(M,no_samples,nx*ny*nz);
```

```
wesi=M(:,1);
headers=getHeader2(nx,ny,nz);
data=zeros(no_samples,nx,ny,nz);

offset=1;
for i=1:nx*ny*nz
    headers(i,:)

data(:,headers(i,1),headers(i,2),headers(i,3))=M(offset:offset+no_sampl
es-1);
    offset=offset+no_samples;
end
clear M;
M=data;


data=zeros(size(M));
for i = 1:no_samples
   temp = reshape(M(i,:,:,:),nx,ny,nz)  ;            %(s , v , v_2)
   temp = transpose3D_3(temp);                   %(v2, s, v)
  % disp(temp);
   temp = times3D(inv(SI),temp);   %(v2 , s , v)
   temp = transpose3D_2(temp);          %(v,s,v2)
  % disp(temp);
   temp = times3D(inv(VI),temp);                  %(v , s , v_2)
   temp = transpose3D(temp);            %(s,v,v2)
  % disp(temp);
   temp = times3D(inv(V2I),temp);                 %(s , v , v_2)
   temp = transpose3D_2(temp);
  % disp(temp);

  % this is an error
   data(i,:,:,:) = temp;
end


fdata=zeros(size(data));
data2rda=zeros(size(data));
cnt=0;
for k=1:nz
    figure;
hold on;
    for j=1:ny
        for i=1:nx
            cnt=mod(cnt+1,nx*ny);
            if(cnt==0)
                cnt=nx*ny;
            end

            fdata(:,i,j,k)=fftshift(fft((data(:,i,j,k))));
            data2rda(:,i,j,k)=conj(data(:,i,j,k))*10000;
            h=subplot(ny,nx,cnt);
            hold on;
            y=fdata(320:480,i,j,k);
```

```
            plot(real(y*1000));
                 hold on
              plot(imag(y*1000),'r');


set(gca,'YLim',[-0.5 0.5]*1.5); axis('off');
        end
    end
end




function wavelet=WLCons(intensity)
dilation=log2(intensity);
wavelet=zeros(intensity,intensity);
scale=intensity;
translation=0;
rowdone=1;
wavelet(1,:)=ones(1,intensity);
for i=2:1:intensity
    k=log2(intensity/scale);
    for j=1:1:intensity
        if(translation<j &j<=translation+scale/2)
            wavelet(i,j)=1;
        end
        if( translation+scale/2<j&j<=translation+scale)
            wavelet(i,j)=-1;
        end
        if(j==translation+scale)
            translation=translation+scale;
            break;
        end
    end

    if(i==rowdone+intensity/scale)
        scale=scale/2;
        translation=0;
        rowdone=i;
    end
end
```

```
function A = transpose3D(M)
% switches the second and first elements.

for i = 1:size(M,3)
    for j=1:size(M,2)
        for k=1:size(M,1)
           A(j,k,i) = M(k,j,i);
        end
    end
end

function A = transpose3D_2(M)
% switches the first and third elements

for j = 1:size(M,2)
    for i=1:size(M,3)
        for k=1:size(M,1)
           A(i,j,k) = M(k,j,i);
        end
    end
end
function A = transpose3D_3(M)
% switches the third and second values

for j = 1:size(M,1)
    for i=1:size(M,2)
        for k=1:size(M,3)
           A(j,k,i) = M(j,i,k);
        end
    end
end

function A = times3D(M,N)
% multiplies together 2 3D matrices.

for i = 1:size(N,3)
    for j=1:size(N,2)
    A(:,j,i) = M * N(:,j,i);
    end

end

function b=getHeader2(nx,ny,nz)

% nx=8;
% ny=8;
% nz=4;

a=zeros(1,32*32*32);
b=zeros(nx*ny*nz,3);

cnt=0;
for i=0:nx*ny*nz-1
            m=find( a(1:nx*ny*nz)==0);
```

```
    if(numel(m)==0)
        break;
    end

    i=m(1)-1;
    xii=mod(i,nx)+1;
    xjj=mod(floor(i/nx),ny)+1;
    xkk=floor(i/nx/ny)+1;
%   a(ii,jj,kk)=1;

    scale_x=ceil(log2(xii));
    scale_y=ceil(log2(xjj));
    scale_z=ceil(log2(xkk));
    scales=[scale_x scale_y scale_z];
    for i=1:3
       if(scales(i)<1)
           scales(i)=1;
       end
    end
    scales=2.^(scales-1);
    scale_x=scales(1);
    scale_y=scales(2);
    scale_z=scales(3);
    [Y,who_is_90]=max(scales);

    if(who_is_90==1)
        if(scale_y>=scale_z)
            myflag=1;
            scale_1=scale_x;
            scale_2=scale_y;
            scale_3=scale_z;
            ii=xii;
            jj=xjj;
            kk=xkk;
        else
            myflag=2;
            scale_1=scale_x;
            scale_2=scale_z;
            scale_3=scale_y;
            ii=xii;
            jj=xkk;
            kk=xjj;
        end
    end
    if(who_is_90==2)
        if(scale_x>=scale_z)
            myflag=3;
            scale_1=scale_y;
            scale_2=scale_x;
            scale_3=scale_z;
            ii=xjj;
            jj=xii;
            kk=xkk;
        else
            myflag=4;
```

```
                scale_1=scale_y;
                scale_2=scale_z;
                scale_3=scale_x;
                ii=xjj;
                jj=xkk;
                kk=xii;
            end
        end
        if(who_is_90==3)
            if(scale_x>=scale_y)
                myflag=5;
                scale_1=scale_z;
                scale_2=scale_x;
                scale_3=scale_y;
                ii=xkk;
                jj=xii;
                kk=xjj;
            else
                myflag=6;
                scale_1=scale_z;
                scale_2=scale_y;
                scale_3=scale_x;
                ii=xkk;
                jj=xjj;
                kk=xii;
            end
        end

        [xii xjj xkk]

        for j=0:Y-1
            cnt=cnt+1;



 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5

                if(scale_1>1)
                    p=mod(ii+j,scale_1);
                    if p==0
                        p=scale_1;
                    end

                    p=p+scale_1;

                else
                    p=ii;
                end
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                if(scale_3>1)
                    q=mod(jj+j,scale_3);
                    if(q==0)
                        q=scale_3;
                    end
%                     q=q+floor(j/scale_3)*scale_3;
```

```
                        q=mod(q,scale_2);
                        if(q==0)
                            q=scale_2;
                        end
%                        mod(jj+j,scale_3)
%                        floor(j/scale_3)
%                        scale_3
                        q=q+scale_2;

                    else
                        q=jj;
                    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                    if(scale_3>1)
                        r= mod(kk+j,scale_3);
                        if(r==0)
                            r=scale_3;
                        end

                        r=r+scale_3;

                    else
                        r=kk;
                    end

                    if(myflag==2)
                        temp=q;
                        q=r;
                        r=temp
                    end
                    if(myflag==3)
                        temp=p;
                        p=q;
                        q=temp;
                    end
                    if(myflag==4)
                        temp=p;
                        p=r;
                        r=q;
                        q=temp;
                    end
                    if(myflag==5)
                        temp=p;
                        p=q;
                        q=r;
                        r=temp;
                    end
                    if(myflag==6)
                        temp=p;
                        p=r;
                        r=temp;
                    end


                %   p+(q-1)*nx+(r-1)*nx*ny
                if(a(p+(q-1)*nx+(r-1)*nx*ny)==1)
```

```
                    if(myflag==1)
                        q=q+scale_3;
                    end
                    if(myflag==2)
                        r=r+scale_3;
                    end
                    if(myflag==3)
                        p=p+scale_3;
                    end
                    if(myflag==4)
                        r=r+scale_3;
                    end
                    if(myflag==5)
                        p=p+scale_3;
                    end
                    if(myflag==6)
                        q=q+scale_3
                    end
                end

                    a(p+(q-1)*nx+(r-1)*nx*ny)=1;

                    b(cnt,1)=p;
                    b(cnt,2)=q;
                    b(cnt,3)=r;
%                     if(who_is_90==1)
%
%                         b(:,(q-Y-1)*(ny/Y)+1:(q-Y)*(ny/Y),:)

                end
end
```

Appendix C Code for Online Analysis

```
%
clc;
[filename , pathname ] = uigetfile('*.spectro', 'Select an RDA file')
nx=8;
ny=1;
nz=1;
output_filename=[pathname, filename];
outfile=fopen(output_filename,'w');
 % for m=1:2
  for k=1:nz
  for j=1:ny
  for i=1:nx
 for n=1:1024
 fwrite(outfile,real(data2rda(n,i,j,k))*1,'float32','ieee-le');
 fwrite(outfile,imag(data2rda(n,i,j,k))*1,'float32','ieee-le');
  end
  end
  end
  end
%  end

fclose all;
```

Appendix D Code for LCModel Analysis

```
%
% Read spectroscopy data from Siemens machine
%
% Read a .rda file
%
%
clc;
[filename , pathname ] = uigetfile('*.rda', 'Select an RDA file')
rda_filename = [pathname , filename]; %'c:/data/spectroscopy/spec raw
data/MrSpec.20020531.160701.rda'

fid = fopen(rda_filename);
myfilename=sprintf('modified_%s',filename);
output_filename=[pathname, myfilename];
outfile=fopen(output_filename,'w');

head_start_text = '>>> Begin of header <<<';
head_end_text   = '>>> End of header <<<';

tline = fgets(fid)
fwrite(outfile,tline);
while (isempty(strfind(tline , head_end_text)))

    tline = fgets(fid)
    fwrite(outfile,tline);
    if ( isempty(strfind (tline , head_start_text)) + isempty(strfind
(tline , head_end_text )) == 2)


        % Store this data in the appropriate format

        occurence_of_colon = findstr(':',tline);
        variable = tline(1:occurence_of_colon-1) ;
        value    = tline(occurence_of_colon+1 : length(tline)) ;

        switch variable
        case { 'PatientID' , 'PatientName' , 'StudyDescription' ,
'PatientBirthDate' , 'StudyDate' , 'StudyTime' , 'PatientAge' ,
'SeriesDate' , ...
                    'SeriesTime' , 'SeriesDescription' , 'ProtocolName'
, 'PatientPosition' , 'ModelName' , 'StationName' , 'InstitutionName' ,
...
                    'DeviceSerialNumber', 'InstanceDate' ,
'InstanceTime' , 'InstanceComments' , 'SequenceName' ,
'SequenceDescription' , 'Nucleus' ,...
                    'TransmitCoil' }
            eval(['rda.' , variable , ' = value ']);
        case { 'PatientSex' }
            % Sex converter! (int to M,F,U)
            switch value
            case 0
```

```
                rda.sex = 'Unknown';
            case 1
                rda.sex = 'Male';
            case 2

                rda.sex = 'Female';
            end

        case {  'SeriesNumber' , 'InstanceNumber' , 'AcquisitionNumber'
, 'NumOfPhaseEncodingSteps' , 'NumberOfRows' , 'NumberOfColumns' ,
'VectorSize' }
            %Integers
            eval(['rda.' , variable , ' = str2num(value) ']);
        case { 'PatientWeight' , 'TR' , 'TE' , 'TM' , 'DwellTime' ,
'NumberOfAverages' , 'MRFrequency' , 'MagneticFieldStrength' ,
'FlipAngle' , ...
                    'SliceThickness' ,  'FoVHeight' , 'FoVWidth' ,
'PercentOfRectFoV' , 'PixelSpacingRow' , 'PixelSpacingCol'}
            %Floats
            eval(['rda.' , variable , ' = str2num(value) ']);
        case {'SoftwareVersion[0]' }
            rda.software_version = value;
        case {'CSIMatrixSize[0]' }
            rda.CSIMatrix_Size(1) = str2num(value);
        case {'CSIMatrixSize[1]' }
            rda.CSIMatrix_Size(2) = str2num(value);
        case {'CSIMatrixSize[2]' }
            rda.CSIMatrix_Size(3) = str2num(value);
        case {'PositionVector[0]' }
            rda.PositionVector(1) = str2num(value);
        case {'PositionVector[1]' }
            rda.PositionVector(2) = str2num(value);
        case {'PositionVector[2]' }
            rda.PositionVector(3) = str2num(value);
        case {'RowVector[0]' }
            rda.RowVector(1) = str2num(value);
        case {'RowVector[1]' }
            rda.RowVector(2) = str2num(value);
        case {'RowVector[2]' }
            rda.RowVector(3) = str2num(value);
        case {'ColumnVector[0]' }
            rda.ColumnVector(1) = str2num(value);
        case {'ColumnVector[1]' }
            rda.ColumnVector(2) = str2num(value);
        case {'ColumnVector[2]' }
            rda.ColumnVector(3) = str2num(value);

        otherwise
            % We don't know what this variable is.  Report this just to
keep things clear
            disp(['Unrecognised variable ' , variable ]);
        end

    else
        % Don't bother storing this bit of the output
    end
```

```
end

%
% So now we should have got to the point after the header text
%
% Siemens documentation suggests that the data should be in a double
complex format (8bytes for real, and 8 for imaginary?)
%

bytes_per_point = 16;
complex_data = fread(fid , rda.CSIMatrix_Size(1) *
rda.CSIMatrix_Size(1) *rda.CSIMatrix_Size(1) *rda.VectorSize * 2 ,
'double');

%fread(fid , 1, 'double');  %This was a check to confirm that we had
read all the data (it passed!)

fclose(fid);

% Now convert this data into something meaningful

 %Reshape so that we can get the real and imaginary separated
 hmm = reshape(complex_data,  2 , rda.VectorSize ,
rda.CSIMatrix_Size(1) ,  rda.CSIMatrix_Size(2) ,  rda.CSIMatrix_Size(3)
);

 %Combine the real and imaginary into the complex matrix
 hmm_complex = complex(hmm(1,:,:,:,:),hmm(2,:,:,:,:));

 %Remove the redundant first element in the array
 Time_domain_data = reshape(hmm_complex, rda.VectorSize ,
rda.CSIMatrix_Size(1) ,  rda.CSIMatrix_Size(2) ,
rda.CSIMatrix_Size(3));



%  for k=1:rda.CSIMatrix_Size(3)
%      for j=1: rda.CSIMatrix_Size(2)
%          for i=1:rda.CSIMatrix_Size(1)
%              fdata(:,i,j,k)=fftshift(fft(conj(
Time_domain_data(:,i,j,k))));
%          end
%      end
%  end
%

 %riplot(Time_domain_data(:,1,1,1))
 %% insert the time domain data here
  for l=1:4
  for n=1:4
  for j=1:4
 for k=1:1024
 fwrite(outfile,real(data2rda(k,j,n,1)),'double');
```

```
 fwrite(outfile,imag(data2rda(k,j,n,1)),'double');
 end
  end
  end
  end

fclose all;
```