ABSTRACT

Title of Document: EVALUATION OF TRAUMATIC BRAIN INJURY USING MAGNETIC RESONANCE SPECTROSCOPY

Elijah O. George, Doctor of Philosophy, 2014

Directed By: Professor Rao P. Gullapalli
Department of Bioengineering

Traumatic brain injury (TBI) is responsible for a third of all injury-related deaths in the United States. With the lack of structural imaging biomarkers available for the detection and evaluation of TBI sequelae, unambiguous diagnosis and prognosis in TBI still remain a huge challenge. Furthermore, complications arising from TBI can lead to cognitive, social, emotional and behavioral defects later in life. Even in confirmed cases of head injury, computed tomography (CT) and conventional MR techniques are limited in their ability to predict the neuropsychological outcome of patients. While the initial trauma can induce structural impairment of brain tissue, the bulk of the cerebral dysfunction ensuing from TBI is due to alterations in cellular biochemical processes that occur in the days and weeks following the traumatic incident. There is therefore a need for advanced imaging modalities that are able to probe the more underlying cellular changes that are induced by TBI. Understanding such cellular changes will be useful in predicting patient outcome and designing

interventions to alleviate the injury sequelae. Magnetic Resonance Spectroscopy (MRS) is a non-invasive imaging modality that is capable of detecting cellular metabolic changes in *in vivo* tissue. In this study we will consider the use of MRS as a clinically relevant tool in the diagnostic and prognostic evaluation of TBI. To this end, we have laid out the following specific aims: (i) To understand the nature and implications of neurometabolic sequelae in mild traumatic brain injury (mTBI) by carrying out cross-sectional comparisons of mTBI patients to neurologically healthy subjects at different stages of injury and to determine associations between early neurometabolic patterns and chronic neuropsychological performance in mTBI patients (ii) To develop novel MRS pulse sequence acquisition and data processing techniques that will enable a more thorough neurometabolic evaluation of TBI and enhance quantification of MRS data (iii) To develop automated classification systems in mTBI using early neurometabolic information that will aid discrimination between subjects with and without injury related sequelae and allow the prediction of symptomatic outcome at the later stages of injury.

EVALUATION OF TRAUMATIC BRAIN INJURY USING MAGNETIC RESONANCE SPECTROSCOPY

By

Elijah O. George

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2014

Advisory Committee:
Professor Rao Gullapalli, Chair
Professor Adam Hsieh, Co-chair
Professor Yu Chen
Professor Yang Tao
Professor Panagiotis Dimitrakopoulos, Dean’s Representative

Dedication

To my late parents and late grandmother (Mama Cole Street).

Acknowledgements

My Father in heaven, thank you for lifting my head once again. All the glory goes to You alone.

I am immensely grateful to all the people who have contributed one way or another to the success of this dissertation.

I would like to thank my advisor, Professor Rao Gullapalli for great mentoring and guidance over the years. His flexible and considerate nature allowed me to approach my studies in a manner that allowed optimum productivity.

I'm also grateful to all the members of the Magnetic Resonance Research Center (MRRC) at the University of Maryland, Baltimore School of Medicine. They showed great support and assistance with all my experiments and research work. I am particularly grateful to Steve Roys who provided great technical support for a lot of my projects.

Finally, I would like to send a huge "I love you" to all my family and friends who provided me with invaluable moral, spiritual, financial and emotional support during the course of my graduate school experience. Firstly, to my siblings, Hannah and Joshua - You have been rocks to me. I couldn't have done this without you. Also, to my brother and brother-in-Christ, Fikayo - Thanks for your prayers and spiritual support. They went a long way. Finally, Pastor Adebisi and my New Wine Assembly family - God strategically placed you in my life for times such as these. You have been a huge source of inspiration. I am eternally grateful to you.

DOD Award #W81XWH-12-1-0098
DOD Award # W81XWH-08-1-0725

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ANAM – Automaed Neuropsychological Assessment Metrics
ATP – adenosine triphosphate
AUC – area under the curve
BW - bandwidth
Cho – total choline
Cre – total creatine
CR – Cramer-Rao
CRLB – Cramer-Rao lower bounds
CSV – centrum semiovale
CT – computed tomography
DAI – diffuse axonal injury
DE-MRSI – dual echo magnetic resonance spectroscopy imaging
DTI –diffusion tensor imaging
EDH – epidural hematoma
EMF – electromotive force
ESA – early sub-acute
FID – free induction decay
FOV – field of view
FPR – false positive rate
FT – Fourier transform
FWHM – full width at half maximum
GABA – gamma-amino butyric acid
GCS – Glasgow Comma Scale
GM – gray matter
Glx – glutamate/glutamine
GPC –glycerophosphoryl choline
HIV – Human Immunodeficiency Virus
^{1}H-MRS – proton magnetic resonance spectroscopy
^{1}H-NMR – proton nuclear magnetic resonance
IVH – intra ventricular hemorrhage
LDA – Linear Discriminant Analysis
LG – Lorentz-Gaussian
LOC – loss of consciousness
LSA – late sub-acute
MRI – magnetic resonance imaging
MRS – magnetic resonance spectroscopy
MRSI – magnetic resonance spectroscopy imaging
mI – myo - inositol
mTBI – mild traumatic brain injury
NAA – n-acetyl aspartate

NMR – nuclear magnetic resonance
NP - neuropsychological
PC – phosphoryl choline
PCS - post concussive syndrome
pFDR – positive false discovery rate
PTA – post traumatic amnesia
PTSD – post traumatic stress disorder
ppm – parts per million
PR – pattern recognition
PRESS – Point Resolved Spectroscopy
RF – radio frequency
ROC – receiver operating characteristic
SAH – sub arachnoid hemorrhage
SE – spin echo; single echo
SNR – signal to noise ratio
STE – stimulated echo
STEAM – Stimulated Echo Acquisition Mode
SVM – Support Vector Machines
SVS – single voxel spectroscopy
SWI – susceptibility weighted imaging
TBI – traumatic brain injury
TE – echo time
TMS – tetramethyl silane
TR – repetition time
VOI – volume of interest
WM – white matter

Chapter 1: **Introduction**

Traumatic Brain Injury (TBI) is a leading cause of death and disability in the United States. It is estimated that there are 1.5 million cases of TBI annually, 50,000 of which are fatal.(Thurman et al., 1999) In particular, TBI is a growing concern amongst military personnel as a huge number of soldiers fall casualty during deployment.(Hoge et al., 2008; Toblin et al., 2012; Wilk et al., 2012) TBI can be caused by the direct impact of forces on the head or by forces that induce a relative acceleration or deceleration of the brain with respect to the skull.(Bayly et al., 2005; Sabet et al., 2008) There are two types of injury process that are caused by brain trauma, namely *primary injury* and *secondary injury*. Primary injury is the structural damage that is suffered by cerebral tissue and blood vessels as a result of the initial traumatic insult. These include cerebral hemorrhages, contusions and lacerations. Secondary injury on the other hand may or may not result from primary injury and occurs in the minutes, hours and days following trauma. Secondary injury involves the underlying cellular processes that occur on a more diffuse scale throughout the brain parenchyma. Such processes include diffuse axonal injury (DAI),(Cecil et al., 1998a; Cecil et al., 1998b) excitotoxicity,(Palmer et al., 1993) apoptosis,(Raghupathi et al., 2000) mitochondrial dysfunction and metabolic alterations.(Schuhmann et al., 2003; Verweij et al., 2000) Hence secondary injury has been implicated as the reason for the gradual deterioration and eventual death of some TBI patients.(Park et al., 2008) While occurrences of primary injury can yield contrast enhancement in conventional imagining methods such as computed tomography (CT) and (magnetic resonance imaging) MRI, understanding secondary injury will require *in vivo* imaging techniques that are more telling of the underlying physiological and neurochemical changes. Imaging modalities such as diffusion tensor imaging (DTI) and magnetic resonance spectroscopy (MRS) have been shown to potentially fit well into this role.(Garnett et al., 2000a; Inglese et al., 2005) Occurrences of TBI can be classified according to severity by using the Glasgow Coma Scale (GCS).(Marmarou et al., 2007) The GCS grades an individual’s level of consciousness based on verbal, motor and eye opening reactions to stimuli. The grading is carried out on a scale of 3-15 with 13-15 considered as mild, 9-12 as moderate and 8 or below as severe TBI. In addition to the neurophysiological dysfunction induced by TBI, patients can also suffer from a number of complications in the acute stages of injury and long after the incidence of trauma. Such complications can be physical, cognitive, emotional and behavioral.(Fann et al., 1995) Hence other than the already demanding feat of detecting acute abnormalities with imaging biomarkers of TBI, a major challenge in current TBI research is being able to predict the eventual outcome of patients and the time required for resolution of symptoms. The awareness of eventual patient outcome would go a long way for patients, their family members and care providers towards planning for resumption of regular activity or improving the quality of life as the case may be.

Magnetic resonance spectroscopy (MRS) is an imaging modality that is capable of non-invasively probing metabolic and biochemical abnormalities *in vivo* at the cellular level. Such cellular *in vivo* changes which are often occult to structural imaging modalities are strong markers of impending or ongoing pathology as well as robust tools for patient prognosis and outcome prediction.

The goal of this dissertation was to assess and enhance MRS as a tool for the evaluation of TBI. Firstly, MRS derived measurements are used to determine metabolic deviations of TBI patients from neurologically healthy subjects. With the aid of a variety of statistical and pattern recognition methods, MRS is also explored as a tool for prognostic evaluation and outcome prediction at the later stages of the pathology. Finally, we develop and evaluate novel approaches to MRS acquisition and metabolite quantification.

This dissertation is organized as follows:

Chapter 2 provides background information on the fundamental aspects of nuclear magnetic resonance spectroscopy (NMR), the science of MRS acquisitions and the methodology of post-acquisition processing and metabolite quantification. In addition, the chapter includes a brief background of pattern recognition methods commonly employed for classification using MRS-derived features. Finally a review of trends in the spectroscopic evaluation of TBI is provided.

Chapter 3 discusses metabolic deviations from neurologically healthy subjects in mTBI patients observed at different stages of the pathology. The study also includes an analysis to predict the cognitive outcome of mTBI patients at the later stages of mTBI using MRS measurements. This work led to a conference abstract and presentation at the Proceedings of the 21st annual meeting of the International Society of Magnetic Resonance in Medicine (ISMRM) in 2013 (George *et al.,* 2013a) and subsequently a paper published in the Journal of Neurotrauma in 2014 (George *et al.,* 2014a)

Chapter 4 describes and evaluates a novel MRS acquisition method that is capable of acquiring multiple MRS data sets within a clinically feasible time frame thereby enabling a more thorough evaluation of TBI. This work led to a conference abstract and presentation at the Proceedings of the 21st annual meeting of the ISMRM in 2013 (George *et al.,* 2013b). The manuscript for this study is in preparation to be submitted to the Journal of Magnetic Resonance in Medicine.

Chapter 5 discusses a novel approach to patient diagnosis and symptomatic outcome prediction in mTBI using pattern recognition algorithms. The analysis also includes a cross validation and receiver operating characteristic analyses that evaluate the accuracy of the predictive model in classifying and predicting symptomatic outcome in newly introduced data samples. This work led to a conference abstract at the Proceedings of the 22nd annual meeting of the ISMRM in 2014 (George *et al.,* 2014b).

Chapter 6 introduces a novel approach to enhancing the accuracy with which MRS metabolite signals are quantified. The method involves a voxel-wise application of time-domain filter functions with parameters that adapt to the local conditions of the voxel, toward improving quantification accuracy. This work led to a conference abstract at the Proceedings of the 22nd annual meeting of the ISMRM in 2014 (George *et al.,* 2014c).

Chapter 7 includes a summary of the findings realized herein as well as future directions

Chapter 2: **Background**

2.1 Nuclear Magnetic Resonance Spectroscopy (NMR)

Introduction to NMR

Nuclear Magnetic Resonance (NMR) is a phenomenon in which certain atomic nuclei can absorb and emit electromagnetic radiation while under the influence of an external magnetic field. The science of NMR has led to the evolution and widespread use of a number of biomedical imaging modalities including Magnetic Resonance Imaging (MRI) and MRS. The relative abundance of protons in the body allows MRI to exploit proton NMR (^{1}H-NMR) in generating contrast between different soft tissues. MRS or *in vivo* NMR spectroscopy has also emerged as a useful way to non-invasively investigate the biochemical activity of living tissue. This section provides a brief introduction to the physics of ^{1}H-NMR spectroscopy.

Because protons have nuclear spin and a positive electric charge, they produce a small magnetic field making them behave like tiny magnets or “spins” which generate their own magnetic fields. Under the influence of a strong external magnetic field B_0 (also known as the static magnetic field), there is an induced energy level difference, with the lower energy spins (a little over half of the spin population) aligning in the direction of the magnetic field and the higher energy spins aligning in the opposite direction (**Figure 2.1**). In this state, prior to excitation by any other source of electromagnetic radiation, the spins are said to be in thermal equilibrium. The lower energy state spins become excitable and their behavior after excitation can provide unique information about the nature of the surrounding tissue and the chemical environment.

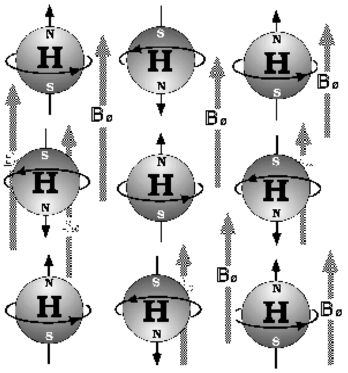


Figure 2.1 Illustration of the behavior of protons while under a static magnetic field B_0. A slight excess of protons aligns in the direction of B_0 to occupy the lower energy state. The remaining protons align opposite to the direction of B_0 to form the higher energy state. (NessAiver, 1996)

Protons possess their own spin about their axes, but when exposed to an external magnetic field, they also precesses about the external magnetic field with a frequency ω_o given by

$$\omega_o = \gamma \mathrm{B}_o \qquad \textbf{Eq. 2.1}$$

ω_o is the precession frequency, also known as the *Larmor* frequency and γ is the gyromagnetic ratio which is a property unique to every atom. During precession, the excitable spins combine to form a magnetization vector M_0 whose resultant is in the direction of the magnetic field. Observing the magnetization requires detecting the precessional motion of the spins. While aligned longitudinally in the direction of the B_0 magnetic field, there is no net motion of the magnetization; hence there is a need to tip the magnetization so it has a component in the transverse plane (the plane perpendicular to the longitudinal axis). This is made possible by another magnetic field B_1 in the transverse plane oscillating in the radio frequency (RF) range. This B_1 field is induced as a pulse during which the magnetization will simultaneously precess about both B_0 and B_1 (**Figure 2.2**). The duration of the pulse determines the angle to which the spins are tipped away from the longitudinal axis. This angle is typically 90^o into the transverse plane (for an excitation pulse) or 180^o (for an inversion pulse).

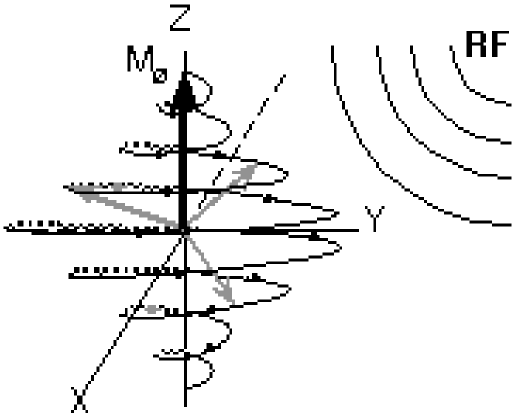


Figure 2.2 Illustration of the motion carried out by the magnetization vector M_o during the application of an RF pulse. The spins precess about both B_o and B_1 causing M_o to undergo a net spiral motion during excitation. (NessAiver, 1996)

Once the transverse magnetization is generated, the net motion of the magnetization induces an electro-motive force (EMF) into a receiver coil as governed by Faraday's law. This induced EMF is what is transduced into an NMR signal.

Relaxation Mechanisms

Upon RF excitation, energy is absorbed by the low-energy state nuclei. As soon as the RF pulse is turned off, these nuclei begin to re-emit the energy causing them to return to thermal equilibrium. This process is known as 'relaxation'. In NMR there are two main mechanisms that govern relaxation of the spins after excitation. These relaxation mechanisms respectively depend on the time constants T1 and T2. These time constants are unique to certain molecular environments or in the case of *in vivo* imaging, certain tissue types. T1 relaxation involves the restoration of the longitudinal magnetization after being tipped away from the longitudinal axis (**Figure 2.3**). With T1 relaxation, most of the energy given off as the relaxation goes as heat into the surrounding tissue lattice. T1 relaxation is governed by the equation

$M_z = M_0 * (1-(1-\cos\alpha) * e^{-t/T1})$ **Eq. 2.2**

If $\alpha = 90$ then,

$M_z = M_0 * (1- e^{-t/T1})$ **Eq. 2.3**

Where M_z is the longitudinal magnetization, α is the flip angle of excitation (the angle with which M_0 is tipped away from the longitudinal axis), t is the time after excitation and T1 is the longitudinal-relaxation time constant.

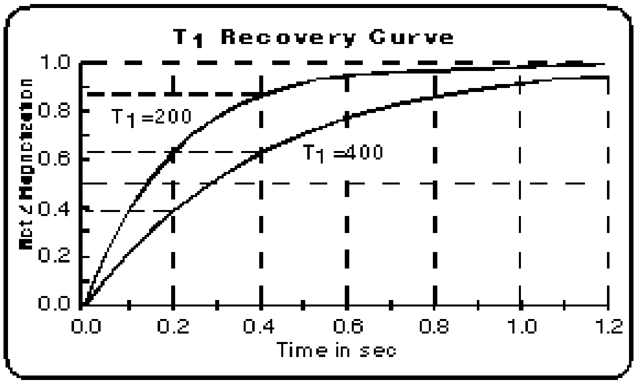


Figure 2.3 The T1 recovery curve of showing the longitudinal magnetization of two different tissue types (T1 = 200 and T1 = 400) after excitation. At a time t = T1, 63.2% of the total longitudinal magnetization has been recovered. (NessAiver, 1996)

T2 relaxation refers to the loss of spin phase-coherence and the resulting disappearance of the transverse magnetization as spins exchange energy between themselves (**Figure 2.4**). While there is no net energy transfer between the spins, the exchange of energy results in increased entropy or 'chaos' of the system, leading to a loss of phase coherence between the spins. This can also be understood as a loss in phase coherence between the spins as the magnetic field generated by each individual spin affects the precession frequencies of other spins as they interact with each other. In the absence of any spatial variations of $B_{0,}$ this spin-spin interaction is the only source of transverse magnetization decay. This decay is described by

$M_{xy} = M_0 * \sin\alpha * e^{-t/T2}$ **Eq. 2.4**

where M_{xy} is the transverse magnetization.

The behavior of the signal is described by a Free Induction Decay (FID).

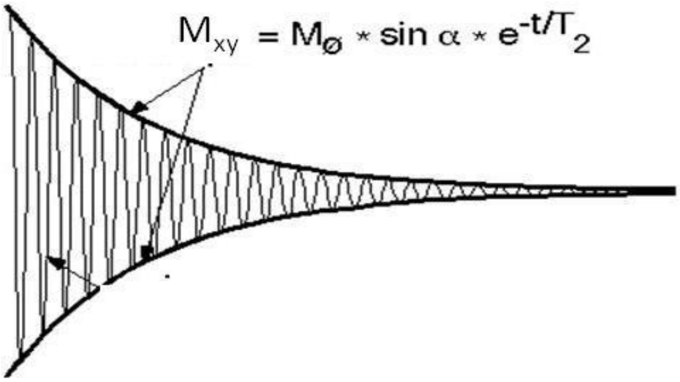


Figure 2.4 Free induction decay describing the signal generated by the transverse magnetization M_{xy} as it precesses about B_o. (NessAiver, 1996)

In reality, the disappearance of the transverse magnetization is also affected by other random factors. Spins also lose phase coherence after excitation if their precession frequencies vary as a result of spatially varying external magnetic fields. Varying external magnetic fields can be induced by non-uniformity of the B_o field or varying magnetic susceptibilities of the surrounding tissue. Hence the transverse relaxation is determined by both the fixed intrinsic T2 relaxation as well as these random factors. T2* is the time constant used to describe the relaxation due to both sources of transverse relaxation. If we replace T2 with T2* in **Eq 2.4**, we obtain

$M_{xy} = M_0 * \sin\alpha * e^{-t/T2*}$ **Eq. 2.5**

Spin Echo Formation

While the signal lost due to intrinsic T2 decay cannot be recovered, spin echoes allow for the recovery of the portion of signal lost due to the random contributions to T2* decay. Spin echoes are formed by applying a refocusing pulse (180^o RF pulse) after excitation. When a refocusing pulse is applied at a time TE/2 after excitation, the magnetization is caused to rotate 180^o in the transverse plane. This rotation allows the spins that had acquired more phase (from having relatively faster precession frequencies) to now lag behind the slower spins with the exact phase they used to exceed the slower spins just before the application of the refocusing pulse. With an additional time of TE/2 after the refocusing pulse (or a total time of TE after excitation), the phase disparities due to field inhomogeneity will be eliminated and a “spin echo” is formed. At this point, any reduction from the original signal is due to fixed T2 relaxation effects alone. This process can be repeated as many times after excitation to continuously eliminate signal losses due to the random contributions to T2* decay (**Figure 2.5**). The time TE is known as the “echo time”. Spin echoes are essential to spatial localization schemes utilized in Magnetic Resonance Spectroscopy pulse sequences.

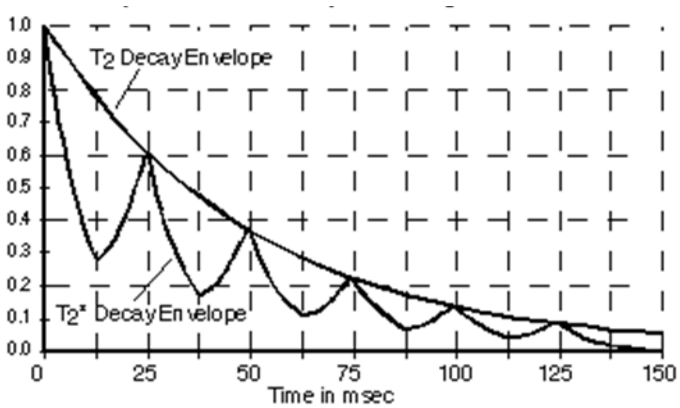


Figure 2.5 The behavior of an NMR signal after initial excitation in which losses due to random contributions to relaxation are restored with a refocusing pulse. Spin echoes are generated at times *TE* = 25, 50, 75, 100 and 125ms after excitation. (NessAiver, 1996)

The Fourier Transform

The FID is a representation of the sum of the signal from all the spins of varying precession frequencies that contribute to magnetization vector. In order to understand information about each of these separate spins, the signal has to be transformed from the time domain where it is acquired to the frequency domain. This makes it possible to separate the signal into its composite resonance frequencies (**Figure 2.6**). The Fourier transformation is a reversible operation that converts signal from the time domain to the frequency domain and vice versa. The conversion of signal from one domain to the other using the Fourier Transform (FT) is described by

$$S(v) = FT[S(t)] = \int_{-\infty}^{\infty} S(t)\exp(-i \quad 2\pi \quad v \qquad t) \quad dt \qquad \textbf{Eq. 2.6}$$

where *v* is the resonance frequency variable and *t* is time variable. *S(*ν*)* is the signal intensity in the frequency domain and *S(t)* is the signal intensity in the time domain. The frequency domain reveals the relative magnitude of the signal at each resonance frequency for that specific TE of acquisition. Hence spins with shorter T2 will have increasingly lower signal as the TE of acquisition becomes longer. Furthermore, the line width of each signal in the frequency domain corresponds to the rate of decay of the signal at that resonance frequency. Faster decaying signals (shorter T2 resonances) will have a broader line width in the frequency domain, while resonances with longer T2 will be represented by a smaller line width. The line width at half the height of the signal in the frequency domain (commonly known as the full width at half maximum (*FWHM*)) is equal to the inverse of the T2 value.

FWHM = 1/T2 **Eq. 2.7**

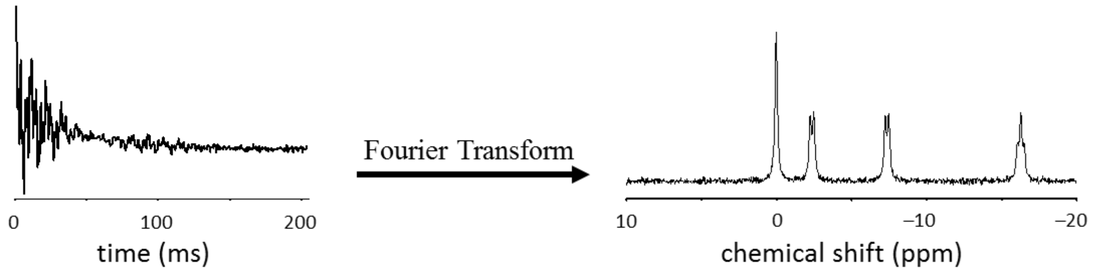


Fig 2.6 The Fourier Transform (FT) is a mathematical operation used to convert an NMR signal from the time domain into the frequency domain. The time domain signal represents the sum of all the composite resonance frequencies present in the signal. The FT allows us to analyze the individual resonance frequencies that contribute to the signal. (De Graaf, 2007)

Chemical Shift

The resonance frequency of any nuclei is dependent on the field B_0 as well as the gyromagnetic ratio. Furthermore, the density of electrons surrounding the nuclei can also lead to variations in the resonance frequency. This property is known as the chemical shift. The electrons surrounding any nuclei create a shielding effect to the nuclei. Because electrons also have spin and a negative charge, under an external magnetic field they behave as tiny magnets rotating in a direction opposite to the precession of the nuclei, thereby subtracting from the strength of the magnetic field experienced by the nuclei. This can allow nuclei of the same atom and under the same magnetic field to precess at different resonance frequencies. The net magnetic field *B* experienced by each nucleus is given by

$$B = B_0(1-\sigma) \qquad \textbf{Eq. 2.8}$$

where σ is the shielding or screening constant.

The chemical shift is not expressed in units of frequency, but in units of parts per million (ppm) which describe the frequency of a spin relative to the frequency of a reference compound. The chemical shift δ of any spin is given by

$$\delta = \frac{v - v_{ref}}{v_{ref}} \times 10^6 \qquad \textbf{Eq. 2.9}$$

Where v is the frequency of the spin of subject and v_{ref} is the frequency of the reference compound. Tetramethylsilane (TMS) is a generally accepted reference compound and has been ascribed a chemical shift of 0ppm (Wishart et al., 1995).

Time-domain Filtering

Time-domain filtering in NMR involves the application of filter or window functions to the NMR signal in the time domain for the purpose of improving the signal-to-noise ratio (SNR), spectral resolution or simply the visual appearance of the spectra. The filtered time domain signal is obtained from the original signal through the following relationship

$f_{\text{filtered}}(t) = f_{\text{original}}(t) \text{ x } f_{\text{filter}}(t)$ **Eq. 2.10**

Where $f_{\text{filtered}}(t)$ is the filtered timed domain signal function. $f_{\text{original}}(t)$ is the original time-domain function prior to application of the filter. $f_{\text{filter}}(t)$ is the applied filter function.

The most common filters employed in NMR signal processing are the exponential weighting filter and the Lorentz-Gaussian filter.

An exponential filter (**Eq. 2.11**) can be used to improve the SNR or spectral resolution of the signal when converted to the frequency domain. By applying a decreasing exponential filter, the noisy data points at the end of the signal are quelled while the points at the beginning of the FID remain relatively untouched ultimately leading to an increase in SNR. A decreasing exponential filter however has the intrinsic property of increasing the spectral linewidths of the frequency domain signal which can diminish spectral resolution. An increasing exponential filter can be used to artificially reduce the linewidths towards enhancing spectral resolution. The reduced linewidths however occur at the expense of measurement sensitivity as the noisy data points at the end of the FID are magnified causing a loss in SNR across the spectrum in the frequency domain. The exponential filter is defined as

$f_{filter}(t) = e^{\pm t/Tw}$ **Eq. 2.11**

Where T_w is the exponential weighting time constant.

A decreasing exponential filter can also be used to reduce frequency domain truncation artifacts in an NMR signal for which the FID acquisition was prematurely truncated. The application of a decreasing exponential filter to a truncated FID allows a more streamlined transition between the acquired data points and the zero-amplitude points at the end of the spectrum. This is illustrated in **Figure 2.7.**

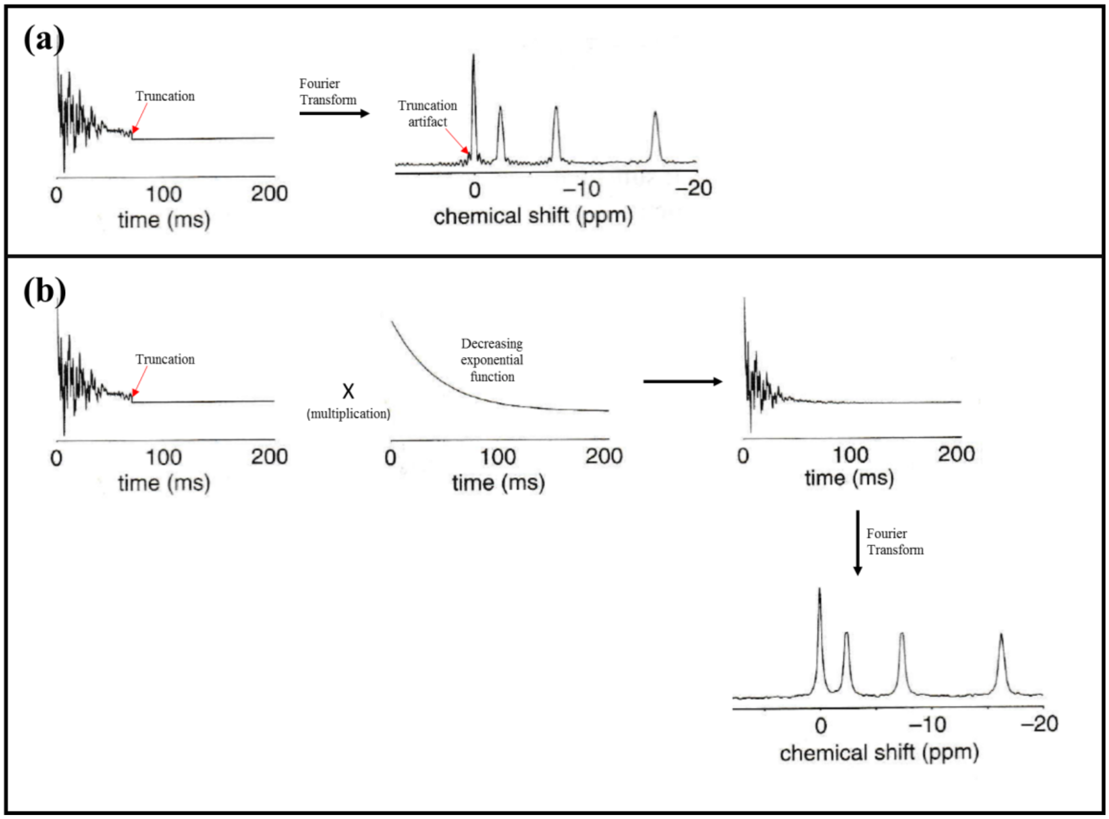


Figure 2.7 The use of an exponential filter in removing truncation artifacts in spectra. (a) The premature truncation of an FID leads to squiggle-like truncation artifacts in spectra after Fourier transform to the frequency domain. (b) The multiplication of the FID by a decreasing exponential function can help to mitigate truncation artifacts by streamlining the transition of the acquired signal into the zero-amplitude points at the end of the spectrum. (de Graaf, 2007)

The Lorentz-Gaussian filter (**Eq. 2.12**) is generally used to convert the lineshape of a frequency domain signal from a Lorentzian shape to a Gaussian shape. Signals with a

Lorentzian lineshape can be harder to integrate in the frequency domain as they produce longer tails and can overlap with neighboring peaks in the NMR spectrum. A Gaussian lineshape however diminishes more rapidly and is desired when resolving marginally separated resonances in a spectrum. The Lorentz-Gausian filter function is defined as

$$F_{filter}(t) = e^{+t\nu_L} e^{-t^2\nu_G^2} \qquad \textbf{Eq. 2.12}$$

$\boldsymbol{F}_{filter}$ is the value of the filter function to be multiplied by the time domain signal at the time (t) of the FID; $\boldsymbol{\nu}_L$(Hz) and $\boldsymbol{\nu}_G$(Hz) are the Lorentzian and Gaussian time constants respectively. In principle, the application of a Lorentz-Gaussian filter suppresses the intrinsic Lorentzian property of the FID when multiplied by exp(+t/T_L), (assuming T_L = T2*) so that exp(+t/T_L) x exp(-t/T2*) = 1. The Gaussian property of the FID will eventually be increased after being multiplied by the "exp(-t^2/T_G^2)" part of the Lorentz-Gaussian filter.

Zero-filling

In truncated NMR acquisitions in which the length of the FID is not adequately sampled, zero filling can be used to simulate an extended acquisition time simply by adding a number of data points with zero signal amplitude at the end of the spectra. Zero filling can be used to artificially enhance the spectral resolution of the signal in frequency domain thereby improving signal quantification. The spectral resolution however can only be enhanced to a finite degree using zero filling, beyond which any further addition of zeros will produce no change in the signal quality. **Figure 2.8** illustrates how zero filling can be used to enhance spectral resolution in truncated time domain signals.

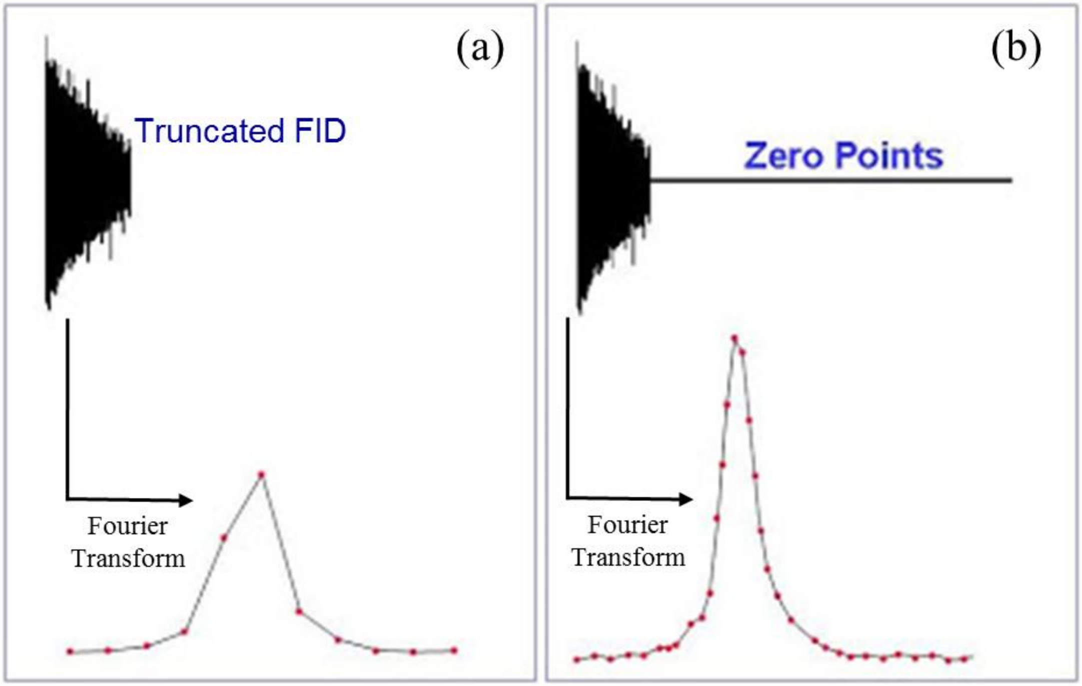


Figure 2.8 Zero filling can be used to artificially improve the spectral resolution and visual appearance of a truncated NMR signal. (a) The truncated time domain signal without zero filling before and after a Fourier transform (b) The truncated time domain signal with zero filling before and after a Fourier transform. (http://u-of-o-nmr-facility.blogspot.com/2007/11/zero-filling.html)

2.2 Proton Magnetic Resonance Spectroscopy Metabolites in the Human Brain

Proton magnetic resonance spectroscopy (^{1}H-MRS) allows non-invasive detection of a number of human brain metabolites *in vivo* **(Figure 2.9)**. This section provides a brief background of some of the most common ^{1}H-MRS detectable metabolites in the human brain.

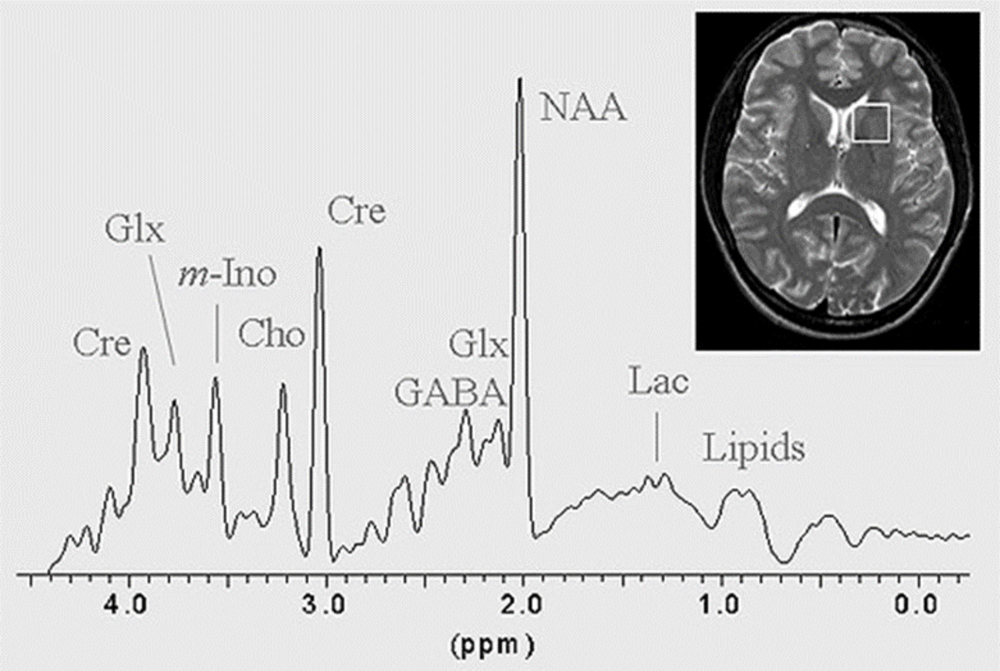


Figure 2.9 The ^{1}H-MRS spectrum of a human brain. (http://www.ncl.ac.uk/magres/research/brain/)

N-Acetyl Aspartate (NAA)

NAA has the largest signal in the ^{1}H-MRS spectrum of a normal human brain. It resonates at a frequency of ~2.01ppm. While its concentration varies throughout the brain, it is believed to be synthesized in the neuronal mitochondria and is dominantly localized within the neurons.(Moffett et al., 2007) Hence its abundance has been generally associated with mitochondrial well-being, neuronal integrity and neuronal density, particularly in traumatic brain injury.(Signoretti et al., 2008; Signoretti et al., 2001) For this reason, NAA decrease is a prominent biomarker in a number of cerebral pathologies including stroke and disseminated encephalomyelitis.(Bizzi et al., 2001; Demougeot et al., 2003) Canavan’s disease is the only known cerebral pathology for which increased NAA levels have been observed.(Wittsack et al., 1996) NAA is also believed to function as an organic osmolyte for regulating cellular water content.(Baslow, 2003) Others have also speculated that NAA serves as a secondary source of acetate for lipid synthesis in glial cells.(Burri et al., 1991)

Total Choline (Cho)

The ^{1}H-MRS signal of Cho is the sum of signals from free choline, glycerophosphorylcholine (GPC) and phosphorylcholine (PC). The signal is measured at ~3.2ppm. While it may be possible to resolve the individual resonances of Cho at higher field strengths, resolving these resonances at clinical magnetic field strengths is seldom achievable as the difference in their chemical shifts are very small. Cho metabolites are believed to be markers of cell membrane turnover as they are key players in pathways of phospholipid metabolism.(Ackerstaff et al., 2003; Zeisel and Blusztajn, 1994) Increases in cerebral Cho levels have been observed in a number of pathological conditions including multiple sclerosis (MS),(Tartaglia et al., 2002) brain tumors,(Herminghaus et al., 2002) and human acquired immunodeficiency syndrome (HIV).(Tracey and Navia, 1996)

Total Creatine (Cre)

The Cre signal in the ^{1}H-MRS spectrum is the sum of signals from phosphocreatine and creatine. Much like the separate signals of total choline, phosphocreatine and creatine cannot be resolved at clinical magnetic field strengths. The Cre signal is measured at ~3.03ppm. Cre metabolites along with ATP are key players in energy metabolism pathways in brain.(Hope et al., 1984) The concentration of Cre within the brain is believed to be unperturbed even in the event of pathology hence Cre has widely been adapted as an internal concentration reference for normalizing other metabolic measurements.(Tartaglia et al., 2002) Caution must however be taken in interpreting results with which Cre is used as a concentration reference as Cre concentrations have been found to be altered in certain pathologic conditions including stroke and TBI.(Friedman et al., 1998; Gideon et al., 1992)

Gamma-Aminobutyric Acid (GABA)

GABA is an inhibitory neurotransmitter with resonances detectable at a number of chemical shifts within the ^{1}H-MRS spectrum (1.89ppm, 2.28ppm and 3.01ppm). All the resonances of GABA however overlap with other signals in the spectrum hence detection is only made possible by special editing techniques.(Keltner et al., 1996) Alterations in cerebral GABA concentration levels have been implicated in alcoholism and substance as well as and psychiatric disorders.(Behar et al., 1999; Sanacora et al., 1999)

Glutamate/Glutamine (Glx)

The signals of Glx can be measured between 2.1ppm and 2.4ppm in the ^{1}H-MRS spectrum. Glutamate is the major excitatory neurotransmitter in the human brain and a precursor for the synthesis of GABA.(Mathews and Diamond, 2003) Glutamate and glutamine are key metabolites in the glutamate-glutamine neurotransmitter cycle during which glutamate is taken up by glial cells and converted to glutamine.(Rothman et al., 1999) Imbalances in glx metabolism and excessive accumulation of glutamate can lead to excitoxicity which has been implicated in a number of pathologies such as TBI and MS.(Pitt et al., 2000; Yi and Hazell, 2006) Resolving the individual resonances of Glx is mostly feasible at higher field strengths and might be necessary to understand the intermediary metabolism of the two metabolites.

Myo-Inositol (mI)

Myo-Inositol (mI) is a hexacylic alchohol and is measured at ~3.56 in the ^{1}H-MRS spectrum. In particular, mI measurments are only obtainable with short echo time MRS measurements as they have characteristically short T2 relaxation times. Myo-Inositol is believed to function as an organic osmolyte in the brain and a marker of glial cell proliferation and glial inflammatory response.(Ashwal et al., 2004; Hattingen et al., 2008) Changes in mI levels have been observed in Alzheimer's disease and TBI.(Ashwal et al., 2004; Miller et al., 1993)

Lactate

Lactate or lactic acid is a carboxylic acid and its production is largely associated with anaerobic respiration in the cell. The lactate doublet is measured at ~1.3ppm in the ^{1}H-MRS spectrum. Increased lactate is commonly observed under hypoxic conditions such as ischemic stroke and tumors.(Graham et al., 1992; Graham et al., 1993; Lai et al., 2002) In the ^{1}H-MRS spectrum of the brain, lactate is usually overlapping with lipids and macromolecules hence its signal is usually measured at intermediate to long echo times where the lipid signal is substantially diminished due to T2 decay or with spectral editing techniques.(Sotak and Freeman, 1988)

2.3 Volume Localization in Magnetic Resonance Spectroscopy

In order to probe the metabolic properties of well-defined tissue regions, it is important to localize the signal acquisition to a volume of interest (VOI) so that ambiguous information from other spatial locations are not included in the acquisition. Also, the presence of extra-cranial lipids can pose a huge source of difficulty during post-processing and metabolite quantification hence it is important to localize the acquired signal to regions free of such signal-contaminating lipids and macromolecules when possible. In addition, the homogeneity of the tissue region localized largely affects the homogeneity of the applied B_0 field, which consequently affects the spectral linewidths and spectral resolution of metabolite signals. A good localization can be used to restrict the signal acquisition to more homogenous tissue regions so that a uniform magnetic field can be achieved; a more uniform field ultimately results in narrower resonance linewidths and improved spectral resolution of the signals. The importance of field homogeneity will be discussed in greater detail in **Section 2.4** below. Localization is generally achieved in MRS by applying three consecutive slice selective RF pulses, each in a different orthogonal direction. With the aid of a magnetic field gradient, each pulse excites a slab of spins with a range of resonance frequencies determined by the magnetic field gradient, in such a way that the slab intersects the desired VOI and only the spins in the VOI are influenced by all three pulses. **Figure 2.10** illustrates how three consecutive RF pulses can be used to localize a VOI. The two most common localization methods utilized in MRS are Stimulated Echo Acquisition Mode (STEAM) and Point Resolved Spectroscopy (PRESS). These will be discussed briefly below.

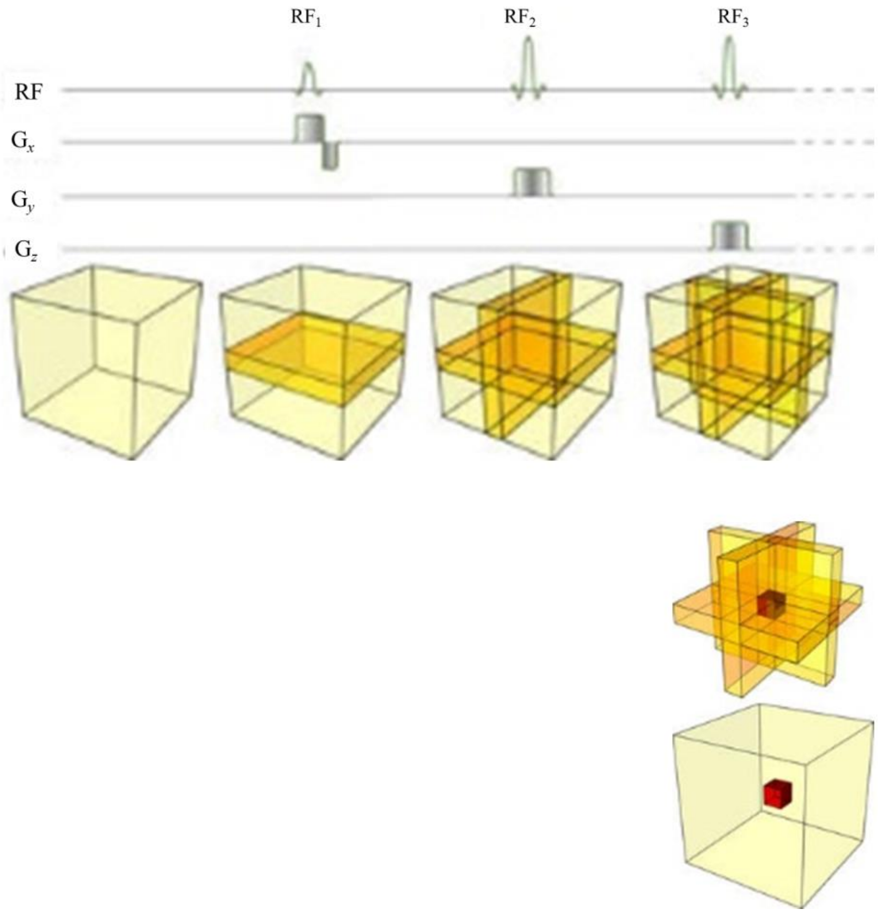


Figure 2.10 Localization of a volume of interest (VOI) using 3 slice selective pulses (RF1, RF2 and RF3) in the presence of slice selective gradient pulses. Each pulse excites a slab of spins in a different orthogonal direction so that the intersection of the 3 slabs forms the VOI. Only spins in the VOI (red box) is influenced by all three RF pulses. (http://www.imaios.com/en/e-Courses/e-MRI/Magnetic-Resonance-Spectroscopy-MRS/single-voxel-spectroscopy)

Stimulated Echo Acquisition Mode (STEAM)

The STEAM pulse sequence utilizes three consecutive 90^0 slice selective RF pulses (each selecting a slab in a different orthogonal plane) for localization of the VOI. The pulse sequence diagram for a STEAM localization scheme is shown in **Figure 2.11**. The first two pulses (pulses 1 and 2) are separated by a delay of TE/2. The last two pulses (pulses 2 and 3) are separated by a delay of TM. Spin echoes (SE) are generated by the combined refocussing effect of pulses 1and 2 (SE12 formed at a time TE), pulses 2 and 3 (SE23 formed at a time TE/2 + 2TM), pulses 1 and 3 (SE13 formed at time TE + 2TM) and pulses 1, 2 and 3 (SE123 formed at a time 2TM). SE123 results from the refocussing by pulses 2 and 3 of the signal excited by pulse 1. In the STEAM sequence however, the signal of interest is the stimulated echo (STE) which is formed after a delay TE/2 following pulse 3. The timing of the formation of the different echoes depends on the duration of the intervals between the pulses. Unambiguous detection of STE without interfering information from other echoes however will require the use of a phase cycling scheme for the individual RF pulses,(Frahm et al., 1987) or the application of magnetic field gradient ‘crushers’ placed between the 2nd and 3rd RF pulses. The undesired FID signals (FID1, FID2 and FID3) formed immediately after each of the 90^0 pulses can be removed by placing identical crushers in between the 1st and 2nd RF pulses, and after the 3rd RF pulse (before the STE) that can dephase the signal coherence. The joint effect of these crushers is the elimination of any signal with a transverse component during the interval between the 2nd and 3rd pulses. The destroyed signal accounts for 50% of the initial excitation. Hence, while the STEAM sequence has the advantage of achieving a lower minimum echo time for the acquired signal compared to the PRESS method, most MRS users are opting to use the PRESS approach as acquisitions using the STEAM method are much lower in SNR.

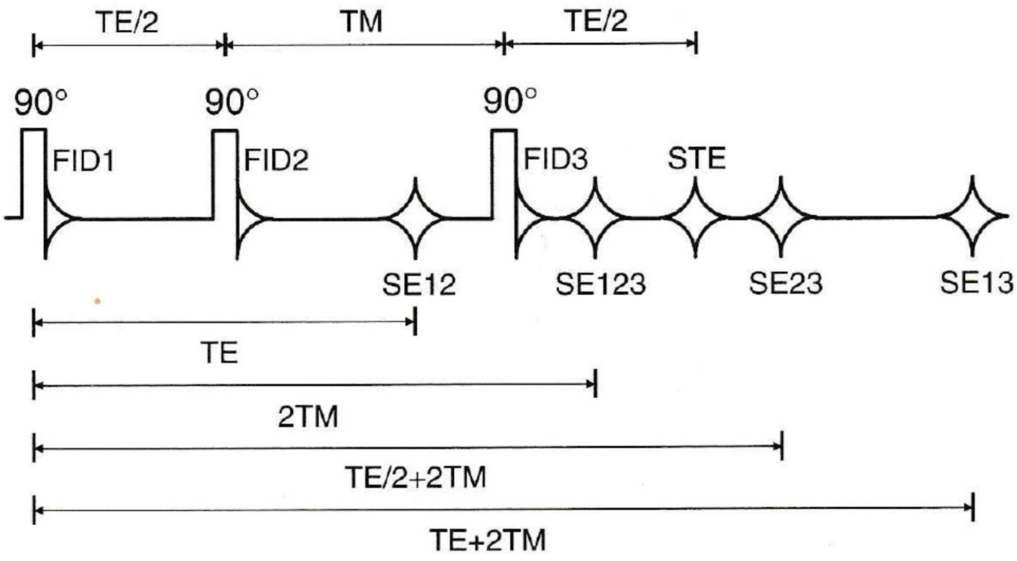


Figure 2.11 A STEAM localization pulse sequence diagram (de Graaf, 2007)

Point Resolved Spectroscopy (PRESS)

In the PRESS localization scheme, the initial 90^0 slice selective excitation is followed by two slice selective refocussing 180^0pulses. The PRESS pulse sequence diagram is shown in **Figure 2.12**. Like the STEAM sequence, each pulse selects a slice in a different orthogonal direction. This allows the initial spin echo formed by the first two pulses to be refocused by the 3rd pulse, while localizing the volume formed by the intersection of the three slices. The echo of choice is the one formed by the last refocussing pulse. Unlike the STEAM approach, the strength of the signal acquired is solely dependent on the echo time of acquisition and the intrinsic relaxation properties of the signal.

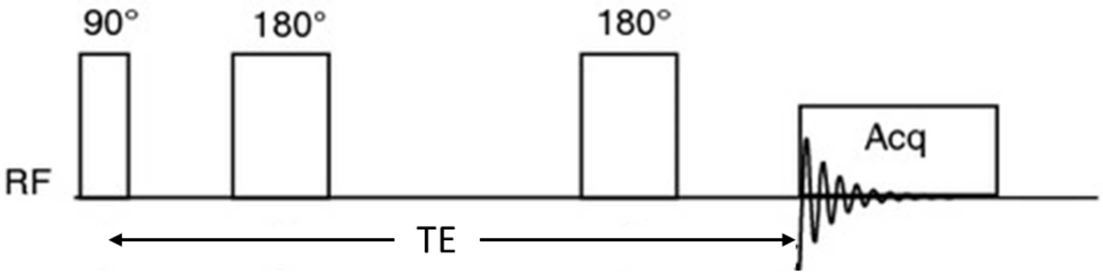


Figure 2.12 A PRESS localization pulse sequence diagram. (http://www.springerimages.com/Images/LifeSciences/1-10.1007_s00249-009-0517-y-1)

2.4 Important Factors in an *In vivo* MRS Acquisition

The quality of an MRS acquisition is largely dependent on a number of factors that go a long way to determine the precision with which eventual quantification of the metabolite signals can be performed. An adequately uniform magnetic field across the tissue sample, precise suppression of tissue water and adequate SNR are necessary to ensure the spectral resolution, sensitivity and overall spectral quality needed for a reliable measurements to be obtained. These are briefly discussed in this section.

Magnetic Field Homogeneity

A non-uniform external magnetic field across the volume of interest gives rise to an increase in the spectral linewidths and overlapping of the resonance signals, ultimately causing a loss in the spectral resolution of the acquisition. This is illustrated in **Figure 2.13** Particularly, resolving closely occurring resonances in the spectrum such as glutamate/glutamine, creatine /phosphocreatine requires a magnetic field that can allow homogeneities of less than 0.1ppm. More so, increased uncertainty in estimating signal amplitude for any resonance is introduced as the resonance linewidth is broadened. The need for optimum magnetic field homogeneity becomes direr at lower acquisition echo times as the presence of a macromolecular baseline can further prohibit unambiguous resolution of resonances.

Magnetic field shimming can be used to optimize the magnetic field homogeneity across the tissue. Shimming is carried out by introducing additional magnetic fields (either with pieces of steel or coils with adjustable current) to the permanent magnetic field in a fashion that allows the magnetic field to be more homogenous.

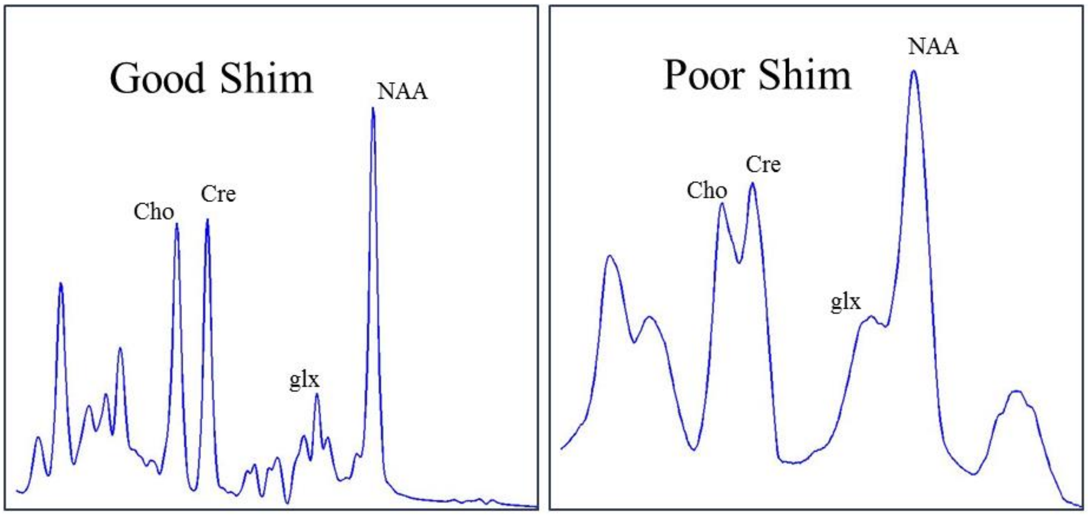


Figure 2.13 Illustration of the effects of B_0 shimming on MRS spectra. A good shim allows closely resonating signals to be properly resolved enabling more accurate spectral quantitation. A poor shim causes closely resonating signals to overlap preventing unambiguous resolution and quantitation

Water Suppression

Water is the most prevalent source of proton signal in the body. To achieve the dynamic range required for detection of much less abundant metabolites, tissue water has to be removed from the MRS signal using water suppression techniques. The presence of the water signal if not adequately suppressed prior to acquisition can also complicate spectral baseline definition as the molecular vibrations of water molecules can introduce side bands into the water signal that spread over a large frequency range in the spectra. Hence a water suppression scheme must precede the pulse sequence for the metabolite signal acquisition. Water suppression in MRS is typically achieved by exciting the water spins alone into the transverse plane using a frequency selective pulse, and de-phasing the coherence of the spins using a magnetic field gradient along the transverse plane. While it is possible to remove unsuppressed water signal with post-acquisition processing algorithms, these algorithms fail in the presence of an asymmetric and or badly phased water signal. Hence effective water suppression schemes prior to acquisition are usually desired.

Sensitivity and Signal-to-Noise Ratio (SNR)

The sensitivity of an MRS acquisition, which is directly proportional to the SNR is a huge factor in determining the reliability of the measurements obtained while using the technique. Needless to say, the detection of minute or subtle changes in tissue metabolic properties will require the use of methods with high levels of sensitivity. Increased SNR in MRS measurements can be realized by increasing the number of signal acquisitions, reducing the receiver bandwidth and increasing the voxel size. Care must however be taken in adjusting these parameters as each of these come at a cost of some other aspect of the acquisition that are also paramount to spectral quality. The number of signal acquisitions is directly proportional to the total scan time. Increased scan time can lead to patient agitation and ultimately motion artifacts in the spectra. Reducing the receiver bandwidth corresponds to reducing the rate at which the FID is sampled. This can lead to poor characterization of the signal, resulting in aliasing artifacts. Increasing the voxel size in a single voxel spectroscopy (SVS) acquisition could mean an increase in the magnetic field inhomogeneity across the VOI as there could be added variation in the tissue types sampled as the VOI is increased. As discussed in the *Magnetic Field Homogeneity* subsection above, this could result in reduced spectral resolution for the acquisition. Where the size of VOI is not increased as in the case of a multi voxel or magnetic resonance spectroscopy imaging (MRSI) acquisition, increasing the voxel size would require a decrease in the image resolution. The optimum choice of acquisition parameters requires careful thought and analysis, and will ultimately depend on the nature of the experiment or examination.

2.5 Data Pre-processing and Spectral Quantification

Before carrying out efficient quantification of metabolites with the acquired MRS data, the data has to be pre-processed so that artifacts resulting from hardware imperfections, molecular and macroscopic motion, and lipid/macromolecular contamination can be removed or alleviated. The stages of data preprocessing and spectral quantification are briefly discussed below.

Data Pre-processing

Pre-processing of the raw MRS data is necessary prior to spectral quantification as artifacts in the spectra can deter unambiguous quantification of metabolite signals. Such pre-processing steps include signal phasing, eddy current correction and baseline correction.

As only the real component of the complex signal is typically used for analysis, the signal can acquire an irregular phase when the real component of the signal is not entirely in absorption mode (i.e when the phase of the signal with respect to the receiver is not zero at time t = 0 of the acquisition). This can be caused by macroscopic motion (such as physiological or gross patient motion) as well as hardware irregularities that allow delays in the acquisition after the formation of the echo. Badly phased spectra can complicate signal quantification and can generate inaccurate measurements when multiple signal acquisitions are added together for the purpose of increasing SNR. Zero and first order phase correction methods can be used to correct phase artifacts in the signal.(Mandal, 2012)

Eddy currents can induce time-varying magnetic fields, in addition to the B_0 field during the acquisition. These additional magnetic fields can perturb the resonance characteristics of the spins ultimately causing distortions in the signal shape. Eddy current corrections can be carried out with an additional spectroscopic scan that acquires the unsuppressed water signal. By monitoring the variation over time of the resonances frequencies of the non-water suppressed time domain signal, phase modulations can be applied to the signal to restore the original frequency behavior; since any irregularities in the resonance frequency can be attributed to the presence of eddy currents.(Zhang et al., 2007) The same corrections applied to the non-water suppressed signal can then be applied to the metabolite scan as the eddy currents affect all the signals in the spectrum.

In many cases, it is not entirely feasible to localize the acquisition of the signal to regions that are free of undesired lipids and macromolecules. Because they resonate at a much broader range of frequencies, these lipids and macromolecules form a contoured baseline on which the other resonances are superimposed, thereby complicating baseline determination and spectral quantification accuracy. Short T2 metabolites and macromolecules have a high transverse relaxation rate as their proton spins are more quickly dephased with respect to each other in the transverse plane. Hence the resultant magnetization is quickly attenuated leading to a rapidly decreasing NMR signal. It is also established that the T2 property largely depends on the molecular weight and mobility of molecules, with the larger and less mobile macromolecules having intrinsically shorter T2 relaxation times.(Schmidt-Rohr et al., 1992) It is therefore possible to minimize the problem of macromolecular contamination using acquisitions with delayed echo times. While it is possible to alleviate the detection of these macromolecules by acquiring the signal at longer echo times, other short T2 metabolites which may be of interest are also not detectable at longer echo times. A popular approach to baseline determination in the presence of a macromolecular background is the spline method, in which the baseline is approximated by a polynomial function and subtracted from the frequency domain signal.(Vanhamme et al., 2000)

Spectral Quantification

Spectral quantification involves the conversion of the measured signal into metabolite concentrations or metabolite ratios. Because the density or concentration of protons is directly proportional to the magnetization at thermal equilibrium in a ^{1}H-MRS acquisition, it is possible to convert signal strength to metabolite concentration with the knowledge of certain acquisition parameters. The relationship between the metabolite signal and metabolite concentration is defined by

$$S_M = NSA \text{ x } Gain_{rec} \text{ x } \omega_0 \text{ x } [M] \text{ x } V \text{ x } f_{sequence} \text{ x } f_{coil} \qquad \textbf{Eq 2.13}$$

Where S_M is the metabolite signal, *NSA* is the number of signal acquisitions, Gain_{rec} is the receiver gain setting, ω_0 is the Lamor frequency, *[M]* is the molar concentration and *V* is the volume of the voxel. $f_{sequence}$ is a function that accounts for the relaxation time constants (T1 and T2) of the metabolite as well as the echo time (TE) and repetition time (TR) of the sequence. f_{coil} is a function that includes parameters related to the quality and geometry of the RF coil.

As it is often difficult to evaluate f_{coil}, it is common practice to determine the concentration of other metabolites using ratios to a reference compound for which the concentration is known. Hence

$$\frac{[M]}{[R]} = \frac{S_M}{S_R} C_{seq} \qquad \textbf{Eq 2.14}$$

Where [R] and S_R are the molar concentration and signal of the reference compound respectively.

$$C_{seq} = \frac{\exp(-TE/T2_M)}{\exp(-TE/T2_R)} \cdot \frac{1-\exp(-TR/T1_M)}{1-\exp(-TR/T1_R)} \qquad \textbf{Eq 2.15}$$

$T1_M$ and $T1_R$ are the longitudinal relaxation time constants for the metabolite and reference compounds respectively. $T2_M$ and $T2_R$ are the transverse relaxation time constants for the metabolite and reference compounds respectively.

Typically, tissue water is used as a reference compound when determining the absolute concentration of metabolites. The tissue water signal is acquired using a non-water suppressed spectroscopic scan. While the molar concentration of water is constant in all tissues, the volume fraction of water can vary with tissue-type, or from voxel to voxel in the case of an MRSI acquisition. Hence it may be necessary to multiply the water concentration by the fraction of the voxel that water comprises in order to obtain an accurate quantification. Information about tissue composition and water content can be obtained by applying segmentation algorithms to structural MRI scans.(Alfano et al., 1997)

As the acquisition of the MRS signal is not carried out under perfect conditions, the signal is best quantified by fitting a model of the signal to the acquired signal. Spectral fitting software such as jMRUI and LCModel have been widely employed for spectral fitting and quantification.(Provencher, 2001; Stefan et al., 2009) Prior knowledge and constraints needed for optimum fitting performance can be obtained from measurements on an in vitro phantom with metabolite composition identical to the tissue being examined or by simulation of a basis set using quantum mechanical properties of the spins.(Ratiney et al., 2005; Stefan et al., 2009)

2.6 Classification

Classification provides a means of differentiating between patients whose neurometabolic patterns are indicative of a pathologic state and individuals who metabolically speaking, have no deviations from a neurologically sound state. By training a pre-existing data set containing the neurometabolic profile of pre-determined members of both groups to recognize neurometabolic patterns unique to each group, we are able to objectively classify previously undiagnosed individuals as belonging to one group or the other. A number of studies have previously investigated the use of *classification* methods in MRS-based TBI evaluation (Auld et al., 1995; Holshouser et al., 2000). The broader groups of MRS studies employing classification methods are studies that focus on classifying human tumors (Howells et al., 1992; Vicente et al., 2013).

Theory

Pattern recognition (PR) is a term used to describe a number of approaches that assign a label or group structure to input data. PR systems have the ability to 'learn' from data and recognize pertinent features of the data that allow it to make generalizations about a previously unseen input. *Classification* is a form of pattern recognition that has been heavily employed in biomedical research to discriminate between patients and healthy subjects (Hagberg et al., 1998), and to classify different forms of a given pathology *e.g.* classifying human tumors (Howells et al., 1992; Vicente et al., 2013). *Classification* approaches can be subdivided into those that utilize *unsupervised learning* and those that utilize *supervised learning*. In unsupervised learning, the classification methods are carried out on the entire data set in order to determine the appropriate group structure. Essentially, the learning procedure is free to determine its own group structure based on whatever features of the data it deems relevant. Examples of such unsupervised learning are cluster analysis and self-organizing artificial neural networks (Sarr et al., 2000; Carpenter et al., 1988). Supervised learning however requires two sets of data; a *training set* and a *validation* or *test set*. The training set consists of the group of objects whose class assignment has been previously determined and will eventually be used to develop a classification system. The test set on the other hand consists of the group of objects whose class is unknown and will be fed to the classification system to determine their class. Examples of supervised learning classification methods include LDA, *Optimal Discriminant Vector* and *Support Vector Machines* (Hagberg et al., 1998).

Perhaps the most widely employed methods for classification using MRS features are Linear Discriminant Analysis (LDA) and Support Vector Machines (SVM). These will be discussed briefly in this section.

Linear Discriminant Analysis (LDA)

LDA is a classification method that determines a linear combination of features that allows optimum discrimination between two or more classes of objects. Each of the various features or variables forms a dimension in a space containing all the classes of data. For example, a set of features in neurometabolite classification studies can include the following: NAA/Cr measurements in the *thalamus*; Cho/Cr measurements in the *centrum semiovale*; and Cr measurements in the *splenium*, *etc*. LDA can be regarded as a dimensionality reduction technique whereby each sample object is projected onto a line by assigning coefficients to these variables. The line of projection will be the direction of optimum separation of classes (**Fig 2.14**).

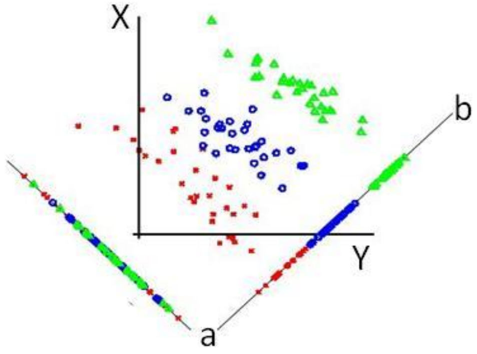


Figure 2.14 Two-Feature (X,Y) data of 3 different groups of objects. Maximum discrimination between the groups can be achieved by projecting the features into a specific line ***a-b*** determined by the LDA algorithm.
(http://courses.ee.sun.ac.za/Pattern_Recognition_813/lectures/lecture01/node6.html)

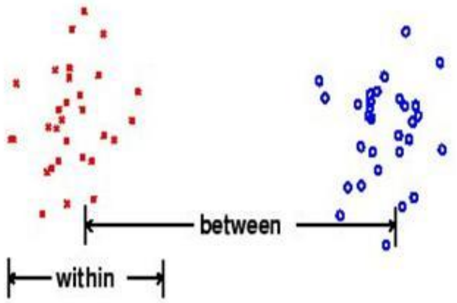


Figure 2.15 Good class separation in LDA is achieved by maximizing the between-class variance while minimizing the within-class variance.
(http://courses.ee.sun.ac.za/Pattern_Recognition_813/lectures/lecture01/node6.html)

Ultimately, the algorithm seeks to maximize the variance between the separate classes, while minimizing the variance within each class (See **Figure 2.15**). Mathematically speaking, this objective is equivalent to diagonalization of the matrix given by

$$Cov_w^{-1/2}.Cov_B.Cov_w^{-1/2} \qquad \textbf{Eq.2.16}$$

Cov_w is the pooled within-class covariance matrix and Cov_B is the between-class covariance matrix.

The covariance matrix describes the inherent variability of the data. The element in the *i,j* position of the covariance matrix is the covariance between the *i*th and *j*th elements of the random vector $\vec{X}$ containing each of the variables of the data.

$$\vec{X} = \begin{matrix} X_1 \\ \vdots \\ X_n \end{matrix} \qquad \textbf{Eq.2.17}$$

where $X_{1\ldots}\, X_n$ are each of the random variables or *features* of the data. The covariance matrix *Cov* is defined as

$$Cov = \begin{bmatrix} \mathrm{E}[(X_1-\mu_1)(X_1-\mu_1)] & \cdots & \mathrm{E}[(X_1-\mu_1)(X_n-\mu_n)] \\ \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n-\mu_n)(X_1-\mu_1)] & \cdots & \mathrm{E}[(X_n-\mu_n)(X_n-\mu_n)] \end{bmatrix} \qquad \textbf{Eq.2.18}$$

E is the mean operator, and $\mu_i = \mathrm{E}(X_i)$

The pooled within-class covariance matrix incorporates the covariance matrices of all the groups we seek to separate.

$$Cov_w = \left(\frac{1}{\sum_{i=1}^{k} n_i - 1}\right) \sum_{i=1}^{k} \frac{Cov_i}{n_i - 1}$$ **Eq.2.19**

Where *Cov_i* is the covariance matrix of group *i*,and *n_i* is the number of objects in group *i.*

Diagonalization of the matrix in equation **Eq.2.16** will yield *eigenvalues* and corresponding *eigenvectors* equal in number to the *number of groups* minus 1. The components of each eigenvector form the coefficients in the so-called ‘discriminant function’. In essence, the components of each vector will form coefficients for the random variables that will allow each object to be projected in the direction that maximizes the separation of groups. The size of the eigenvalue is an indication of how well the each discriminant function separates the groups with the larger eigenvalues achieving better separation.

Support Vector Machines

In the support vector machines (SVM) classifier, the discriminant model is generated by computing two parallel hyperplanes (decision surfaces) in a space defined by the features of the data set. The gap between the two hyperplanes maximally separates the two classes in the training data set. In cases where the data is not linearly separable by hyperplanes, it is possible to apply a kernel function or “kernel trick” which maps the data into a higher dimensional space where linear separation can be more achievable. **Figure 2.16** illustrates how data that originally cannot be separated by linear demarcation can be made separable by a hyperplane after transformation to a higher dimensional space.

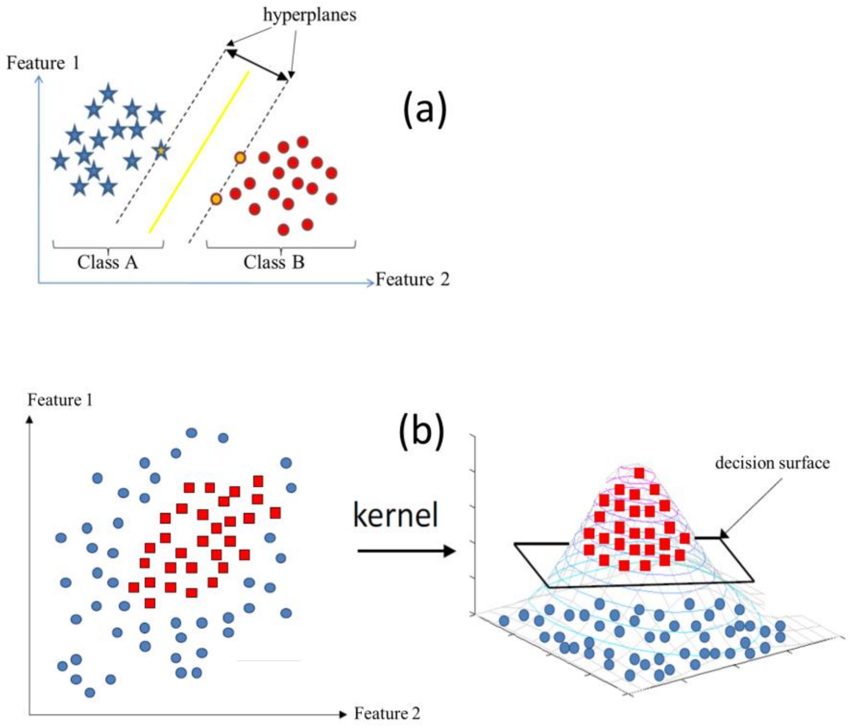


Figure 2.16 (a) linearly separable data can be conveniently classified without the application of a kernel function. (b) When linear separation by a hyperplane is not achievable, a kernel function can be applied to map the data into a higher dimensional space where linear separation can be achieved. (http://www.nyuinformatics.org/downloads/supplements/SVM_Tutorial_2010/Final_WB.pdf)

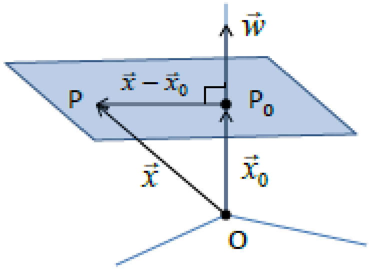


Figure 2.17 Mathematical definition of a hyperplane. (http://www.nyuinformatics.org/downloads/supplements/SVM_Tutorial_2010/Final_WB.pdf)

Figure 2.17 illustrates the definition of a hyperplane in a 3D feature space. A hyperplane is defined by a point *P*₀ and a vector $\vec{w}$ which is perpendicular to the hyperplane at *P*₀. If $\vec{x}_0 = OP_0$ and $\vec{x} = OP$ for any arbitrary point *P* on the hyperplane, then the vector $\vec{x} - \vec{x}_0$ must be perpendicular to $\vec{w}$. Hence,

$$\vec{w} \bullet (\vec{x} - \vec{x}_0) = 0$$

$$\Rightarrow \vec{w} \bullet \vec{x} - \vec{w} \bullet \vec{x}_0 = 0$$

If we define $b = -\vec{w} \bullet \vec{x}_0$, then

$$\vec{w} \bullet \vec{x} + b = 0 \qquad \textbf{Eq. 2.20}$$

Equation 2.20 is the equation for a hyperplane in a multidimensional feature space. The decision boundaries for each class can be further defined by assigning class instances (positive or negative) to each of the classes being separated. **Figure 2.18** illustrates the formulation of demarcating boundaries for class separation.

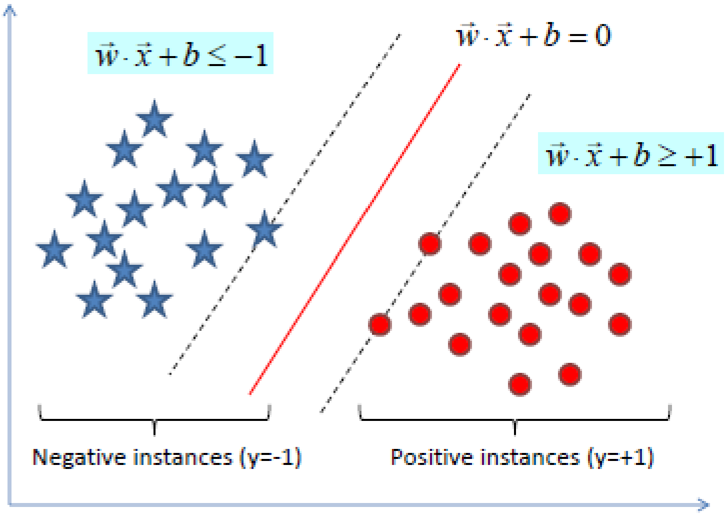

$\vec{w}\cdot\vec{x}+b\leq -1$
$\vec{w}\cdot\vec{x}+b=0$
$\vec{w}\cdot\vec{x}+b\geq +1$
Negative instances (y=-1)
Positive instances (y=+1)

Figure 2.18 Illustration of the assignment of demarcating boundaries for each class. (http://www.nyuinformatics.org/downloads/supplements/SVM_Tutorial_2010/Final_WB.pdf)

Hence, using the equation of a hyperplane in the feature space, it can be shown that the demarcating hyperplanes for each class in a kernel based SVM classifier are defined by

$$g(\vec{x}) = (w \bullet \phi(\vec{x}) + b) \qquad \textbf{Eq. 2.21}$$

$g(\vec{x}) = -1$ for negative instances and $g(\vec{x}) = +1$ for positive instances. The definition of demarcating hyperplanes is illustrated in **Figure 2.18.** $\vec{x}$ is the feature vector for each sample and $\phi(\vec{x})$ is the feature vector after transformation by a kernel function. b and w are geometrical properties of the two hyperplanes. Separation of classes is achieved by using an optimization algorithm to maximize the distance between the two hyperplanes demarcating each class, while limiting the number of sample points within the separation gap. A more elaborate description of the formulation of the SVM algorithm can be found in the paper by Burges.(Burges, 1998)

2.7 A Review of Trends in the Spectroscopic Evaluation of TBI

Neurometabolic Evaluation of TBI

Previous studies have demonstrated the continuously increasing relevance of MRS in the clinical evaluation of TBI. As early as 1995, a pediatric TBI study involving 17 patients revealed increases in tissue lactate (Lac) in regions of brain contusion and infarction (Sutton et al., 1995). Cerebral lactate which is commonly associated with anaerobic glycolysis and/or infiltration of macrophages is believed to be a marker of ischemic or inflammatory conditions following TBI. While additional TBI studies have since reported the presence of lactate following injury (Condon et al., 1998; Hillary et al., 2007; Ross et al., 1998), the scarce evidence of TBI-induced brain lactate is perhaps owing to the duration between injury and time of examination. NAA depression is perhaps the most common neurometabolic change associated with TBI. Cecil et al employed MRS as a modality for detecting diffuse axonal injury (DAI) by examining metabolic white matter changes arising from TBI(Cecil et al., 1998a). The studies revealed reduced NAA/Cre levels in the splenium when compared to healthy controls. NAA depression has since been observed in TBI studies examining all severities of the pathology and across multiple neuroanatomic regions (Garnett et al., 2000b),(Cecil et al., 1998a; Govindaraju et al., 2004),(Brooks et al., 2000; Friedman et al., 1998; Garnett et al., 2000a; Govind et al., 2010; Marino et al., 2007). Such reductions in NAA levels could be either irreversible, indicating permanent neuronal atrophy or could be reversible, in which NAA is restored to normative levels; reversible NAA depression is indicative of a temporary decline in neuronal function (Destefano et al., 1995). The most common changes reported to date involving cell-membrane metabolism marker Cho have been increases relative to the normative levels observed in healthy subjects (Govind et al., 2010; Govindaraju et al., 2004; Sarmento et al., 2009). However a fair amount of ambiguity exists between the Cho-related findings in MRS-TBI studies as a substantial number of studies have reported the lack of significant change with respect to the healthy controls (Gasparovic et al., 2009; Kirov et al., 2007; Vagnozzi et al., 2010; Vagnozzi et al., 2008). The disparities in the findings of differrent studies is likely due to varying examination windows for the patient cohort and perhaps varying severities of TBI cases examined in each study. While it is speculated that cellular Cre levels are unperturbed even in the event of pathology, a number of studies have reported an changes in Cre following TBI (Friedman et al., 1998; Gasparovic et al., 2009; Yeo et al., 2011). Cellular Cre which is measured as the combined signal of creatine and phosphocreatine is believed to be a marker of cellular energy metabolism. Changes in Cre levels following TBI could be a homeostatic response of cerebral cells towards increasing the demand for energy production needed to facilitate cellular repair and restoration of normal cellular physiological conditions. A number of studies have reported an increase in the osmolyte and glial cell activity-marker myo-inositol (mI) following TBI (Ashwal et al., 2004), (Kierans et al., 2014). Myo-inositol increase is also believed to be a marker of inflammatory and edematous conditions following TBI where mI as an organic osmolyte is required to regulate fluid balance in the brain. Recently however, a study investigating sports related concussion in female athletes reported a decrease in mI levels following TBI (Chamard et al., 2013). The later however carried out examinations at least 7 months following injury. This finding is perhaps indicative of chronic cellular and metabolic atrophy where the abundance of mI as a cellular osmolyte could be associated with cell density.

Association of Neurometabolic Measurements with Neuropsychological Variables

A number of studies have previously investigated the association of MRS measurements with neuropsychological test scores (Babikian et al., 2006; Brooks et al., 2000; Friedman et al., 1999; Friedman et al., 1998; Garnett et al., 2000a; Gasparovic et al., 2009; Govind et al., 2010; Yeo et al., 2006). Most reports have focused on NAA as a marker that underlies cognitive function following TBI. One study however revealed that Cre levels in both GM and WM in moderate to severe TBI patients associated positively with NP test scores obtained within 24 hrs of the MRS scan (Friedman et al., 1998). Another study also in moderate to severe TBI reported the positive association between sub-acute Cre levels in GM with concurrent cognitive performance (Friedman et al., 1998). The observation of Cre as a marker of cognitive function could possibly present a more specific metabolic approach to mTBI prognosis as changes in NAA can also be associated with other cellular processes such as lipid synthesis and mitochondrial dysfunction that do not necessarily underlie cognitive function. Indeed changes in total cellular Cre can be more specific to cellular alterations directly associated to recovery from trauma and hence more indicative of long-term functional outcome.

Evaluation of Mild TBI

Mild TBI accounts for 75% of TBI occurences (Cassidy et al., 2004; Tagliaferri et al., 2006). Furthermore, most mTBI cases occur as non-hemorrhagic such that injury-related findings are occult to CT and structural MR. Owing to the deficiencies of structural imagining modalities in accurately diagnosing mTBI, a large number of patients eventually encounter neuropsychological and cognitive deficits in the months and years after the initial trauma. Mild TBI occurrences typically result from a rotational or angular acceleration of the brain as opposed to a direct focal impact. Hence while no gross focal injuries might be observed, the result is a diffuse effect on the overall brain parenchyma. This diffuse injury manifests as subtle changes at the cellular level that can include diffuse axonal injury, inflammation, edema, apoptosis, excitotocicity and mitochondrial dysfunctions. A number of studies have previously investigated neurometabolic alterations in the mTBI cohort. Neurometabolic changes encountered in mTBI can include some of the alterations described above but commonly at a less profound scale than moderate or severe cases of TBI.

Owing to the subtlety in the nature of injury and the lack of neuroimaging markers for the evaluation of mTBI, the thoroughness of the methodology used for evaluation of mTBI induced changes is extremely important. Firstly, the tightness of the examination windows at each stage of the pathology (Acute, sub-acute or Chronic) for patient evaluation could go a long way to determine the ability to detect metabolic changes; varying levels of change in the underlying cellular metabolism occurring as a result of varying examination periods in each patient can introduce noise and diminish the sensitivity with which group changes are detected. Furthermore a large population size is desirable as this will help to reduce the occurrence of false positives and overcome inter-subject variability induced by seemingly extraneous factors such as patient diet and exercise (Greco and Prins, 2013; Wu et al., 2013). Indeed mTBI studies that have been successful in maintaining a prospective and tightly controlled methodology with a substantial population size (N > 25) are few and far between. Vagnozzi et al. carried out a prospective longitudinal study in 28 mTBI athletes after concussion at 3, 15, and 30 days post injury (Vagnozzi et al., 2008). Their findings included decreased NAA/Cre and eventual recovery to baseline levels. In this study however, NAA/Cre values in patients who had suffered a second concussion did not recover to baseline within the 30 examination interval. This is perhaps indicative of a slower recovery process in patients with sub-concussive occurrences. The same authors later published a prospective multicenter study of 40 concussed athletes examined at 3, 15, 22, and 30 DPI (Vagnozzi et al., 2010). The study revealed a reduction of NAA/Cre at the 3-day mark with eventual recovery at 30 days. In a study by Yeo et al investigating sub-acute mTBI (5-21 days post injury) in 32 patients, increased WM Cre and Glx and reduced GM Glx was observed.

Classification Studies in TBI

The use of automated classification methods in the evaluation of TBI presents a less ambiguous approach to patient diagnosis and outcome prediction. This is particularly true for mTBI cases where patient evaluation is carried out using highly subjective self-reported symptoms and field concussion tests. Furthermore automated classification methods can allow a less cumbersome approach to evaluating the numerous changes in metabolic measurements and ratios that result from TBI. A few brain and central nervous system (CNS) injury studies have applied other classification methods to MRS data for the purpose of predicting clinical outcome. Holshouser *et al.* used linear discriminant analysis (LDA) to compare the efficiency of short TE and long TE data in predicting the outcome of children with acute brain injury.(Holshouser et al., 2000) Their analysis revealed that both short and long TE MRS data predicted outcome with an accuracy of 91% in children over one month. In children less than one month, the long TE method performed better with an accuracy of 91% compared to the short TE method which yielded an accuracy of 79%. Auld *et al*. showed that LDA applied to MRS-acquired features alone is able to predict the outcome of children who have sustained acute central nervous system injury with an accuracy of 81%.(Auld et al., 1995) Both of these studies however included children with non-traumatic injuries such as cardiac arrest, hypoxic-ischemic encephalopathy and near-drowning for which the injury sequelae can vastly differ from TBI. Tollard *et al*. showed that by combining MRS and diffusion tensor imaging (DTI) features, an LDA method was able to discriminate between severe TBI patients with unfavorable outcome, those with favorable outcome and control patients with an accuracy of 97%.(Tollard et al., 2009)

Chapter 3: Longitudinal Evaluation of Traumatic Brain Injury: a ^{1}H-MRS Study

3.1 Introduction

It is estimated that annually there are over 1.5 million traumatic brain injury (TBI) occurrences in the United States.(Thurman et al., 1999) More so, studies have also shown that about 12-20% of Iraq and Afghanistan veterans have suffered from TBI.(Hoge et al., 2008) The overwhelming majority of TBI injuries (75 %) are deemed “mild,”(Cassidy et al., 2004; Tagliaferri et al., 2006) with most occurring as non-hemorrhagic contusions or microhemorrhages which often are undetectable by computed tomography (CT) or structural magnetic resonance imaging (MRI).(Garnett et al., 2000b; Mittl et al., 1994) The inability to accurately diagnose mild traumatic brain injury (mTBI) in the acute stage using conventional imaging has resulted in a growing number of patients and veterans who show overt signs of post-traumatic stress disorder (PTSD), depression and other neurological and cognitive deficits in the months following the initial injury.(Kontos et al., 2013) The long term impact of such mild injuries on individuals and their families is a multifaceted problem that constitutes a threat to their physical, social and psychological well-being as well as a huge financial burden on the economy as a whole.(Spelman et al., 2012)

Most mTBI injuries result from the rotational or angular acceleration and/or shearing of the brain.(Bayly et al., 2005; Sabet et al., 2008) Hence the impact is not necessarily focal but may produce a widespread “diffuse” effect on the entire brain parenchyma.

As a result, the bulk of complications arising from mTBI are due to “secondary injury” and the sequelae include diffuse axonal injury (DAI),(Cecil et al., 1998a; Cecil et al., 1998b) inflammation,(Morganti-Kossmann et al., 2001) edema,(Beaumont et al., 2000) apoptosis,(Raghupathi et al., 2000) excitotoxicity,(Palmer et al., 1993) mitochondrial dysfunction,(Verweij et al., 2000),(Xiong et al., 1997) and neurometabolic alterations.(Schuhmann et al., 2003) Such physiological and biochemical changes occurring at the cellular level are rarely detected by conventional CT or MR imaging techniques. The accurate assessment of mTBI therefore necessitates a deeper understanding of changes at the molecular level which leads to changes in the biochemical processes that may precede any discernible macroscopic changes at the tissue level *in vivo*. Magnetic Resonance Spectroscopy (MRS) is one modality that has the potential of providing a non-invasive means for evaluating metabolic changes that occur at the cellular level in mTBI patients.

A number of studies have used MRS to detect neurometabolic changes resulting from mTBI. Perhaps the most common finding is a widespread decrease in the neuronal integrity marker N-Acetyl Asparate (NAA) in both gray matter (GM) and white matter (WM).(Cohen et al., 2007; Govindaraju et al., 2004; Henry et al., 2010; Henry et al., 2011; Sarmento et al., 2009; Vagnozzi et al., 2010; Vagnozzi et al., 2008) However, a fair amount of disparity exists between the studies associated with the cell-membrane turn over marker, choline (Cho). Some studies have found increased Cho in various regions in the brain parenchyma,(Govind et al., 2010; Govindaraju et al., 2004; Sarmento et al., 2009) while others have reported the absence of statistical changes in Cho levels associated with mTBI.(Gasparovic et al., 2009; Kirov et al., 2007; Vagnozzi et al., 2010; Vagnozzi et al., 2008) As total cellular creatine (the combined levels of creatine and phosphocreatine) is believed to be stable with most cerebral abnormalities, it is common practice to normalize the MRS metabolites to total creatine (Cre). Nonetheless, at least two studies have reported an increase in WM Cre among the mTBI population.(Gasparovic et al., 2009; Yeo et al., 2011) Some, but not all of the disparities observed from the various studies are likely due to varying experimental conditions, observation times following injury, and the range of injury severity observed.

Given a growing evidence that mTBI patients (including veterans returning from Iraq and Afghanistan) have the propensity to suffer from long term cognitive and psychological symptoms following injury,(Bay et al., 2012; Hoge et al., 2008; Toblin et al., 2012; Wilk et al., 2012) this group of patients deserves special attention. Although these patients have subtle closed head injuries and are able to function well for the most part, a significant portion of these patients have reported cognitive changes and depression several months following injury.(Wilk et al., 2012) The long-term cognitive, social and emotional effects of mTBI on a patient's life can be immensely profound, yet abnormalities detected by current neuroimaging methods and neuropsychological tests can be subtle or even nonexistent. Hence, conclusive evidence about neurometabolic changes that occur following mTBI and the impact of these changes on overall brain function will require studies that employ a thorough methodology. Firstly, "noise" arising from the variability in the time of examination post injury will need to be quelled by tightly controlled evaluation periods as changes in the underlying physiology can induce metabolic changes even over small time intervals. Furthermore, if we are to garner a genuine sense of the evolution of metabolite levels at different stages in the progression of mTBI, such a practice of tightly controlled examination windows will need to be carried out at multiple stages of the pathology. Secondly, a large population size at each stage of examination is imperative to reducing the occurrence of false positives and overcoming inter-subject variability induced by factors other than patient age that usually go unaccounted for. Indeed seemingly extraneous factors such as exercise and diet have been shown to influence the progression of TBI.(Greco and Prins, 2013; Wu et al., 2013) The current study was designed to address the aforementioned issues that have been a major limitation in prior mTBI spectroscopy literature.

In the current study, we examined mTBI patients at three major stages of mTBI [early sub-acute (ESA), late sub-acute (LSA) and chronic], while strictly controlling the examination windows for each of these stages. We also sought to address the issue of population size in each period of examination by including a substantial number of patients at each time-point studied. Furthermore, our analysis includes a prognostic evaluation of cognitive ability at the chronic stage of mTBI using metabolic measurements obtained at the ESA stage of mTBI. The value of a method that is able to predict chronic cognitive outcome right from the early stages of the pathology cannot be overstated as this can enable caregivers to provide necessary interventions to maximize long term cognitive ability. To our knowledge, this is the first time neurometabolic measurements have been used to predict long term cognitive outcome in the mTBI cohort.

In order to determine which regional metabolic values were sensitive enough for correlation with neuropsychologic data, we compared the measurements from mTBI patients to those of healthy control subjects using MRSI measurements from a number of neuroanatomic regions. The regions examined include the posterior corpus callosum, thalamus, putamen, posterior periventricular white matter and the centrum semiovale (CSV). Our analysis revealed the thalamus and CSV to be the most sensitive regions as measurements in all other regions failed to yield any significant differences between mTBI patients and control subjects. Indeed the interconnectivity with, and proximity to the cerebral cortex,(Behrens et al., 2003; Berman et al., 2004) make these two regions highly vulnerable to injury – the cortex being the major impact site of trauma induced shearing forces. Previous neuroimaging studies have shown that these regions are highly sensitive to TBI-induced changes.(Inglese et al., 2005; Kennedy et al., 2009; Kirov et al., 2007; Squarcina et al., 2012) Subsequently, we assessed metabolic markers in these regions (thalamus and CSV) for their capability to predict neurocognitive outcome 6 months after injury.

3.2 Materials and Methods

Patient Selection

All patients were recruited from the Adams Cowley Shock Trauma Center at the University of Maryland Medical Center as part of an ongoing MagNet Study (Magnetic Resonance Imaging of Neuro Trauma Study). The study was approved by the IRB of the University of Maryland. In the MagNet study, TBI patients of diverse severity [Glasgow Coma Score (GCS) 3-15] were recruited and assessed acutely or at the early sub-acute stage (within 10 days of injury) and were longitudinally assessed sub-acutely (approximately one month following injury) and chronically (approximately 6 months following injury). The full examination involved a combination of advanced MR imaging and neuropsychological assessment using the Automated Neuropsychological Assessment Metrics (ANAM) battery of tests.(Ivins et al., 2009; Kane et al., 2007) Patients were included in this study if they were 18 years of age or older with mechanism of injury indicative of closed head trauma, and positive head CT or altered mental status and/or loss of consciousness. Patients were excluded if they had a history of neurological or psychiatric illnesses, cerebrovascular accidents, brain neoplasms or seizures. Given our interest in metabolic changes among mTBI patients, only patients with an admission GCS of 13 – 15 were included in this study. Neither loss of consciousness (LOC) nor post traumatic amnesia (PTA) was used in determining the severity of TBI as this information was either unavailable or subject to ambiguity when self-reported. It is also important to note that the mTBI cohort examined herein included complicated mTBI patients (patients with positive acute/ESA CT or MRI). **Table 3.1** summarizes the demographic information of healthy controls and mTBI patients examined at each time point. **Table 3.2** provides a description of CT/ structural MRI findings for patients in the complicated mTBI category. Forty-three consecutive mTBI patients (40.63 ± 17.31 yrs, 10 female, 12 complicated) were included in this study on whom MRS data was obtained at the ESA stage within 10 days (5.44 ± 3.15 days) of injury. MRS data from thirty-three consecutive patients (37.64 ± 16.60 yrs, 10 female, 10 complicated) was included from the LSA stage (37.00 ± 12.26 days post injury). MRS data from twenty-seven consecutive patients (40.11 ± 17.34 yrs, 9 female, 8 complicated) was included from the chronic stage (195.30 ± 19.60 days post injury). MRS data from 21 (39.76 ± 18.04 yrs, 8 female) healthy, neurologically intact subjects was used for the control group. Of the patients who underwent an ANAM assessment at the third visit, twenty-three of them also received an MRS assessment at the ESA stage.

	Control	**Early Sub-acute**		**Late Sub-acute**		**Chronic**	
	Value ± StdDev	Value ± StdDev	P-value vs Control	Value ± StdDev	P-value vs Control	Value ± StdDev	P-value vs Control
N	21	43	NA	33	NA	27	NA
Age	39.76 ± 18.04	40.63 ± 17.31	0.854	37.64 ± 16.60	0.659	40.11 ± 17.33	0.946
Female (N)	8	10	NA	10	NA	9	NA
GCS	NA	14.74 ± 0.58	NA	14.79 ± 0.55	NA	14.63 ± 0.68	NA
Education (Years)	15.48 ± 1.60	13.77 ± 2.65	.009	13.73 ± 2.84	.013	14.07 ± 2.56	.033
Days Post Injury	NA	5.44 ± 3.15	NA	37.00 ± 12.26	NA	195.30 ± 19.60	NA
Positive CT (N)	NA	10 (23.26%)	NA	8 (24.24%)	NA	7 (25.92%)	NA
Positive MRI (N)	NA	10 (23.26%)	NA	9 (27.27%)	NA	7 (25.92%)	NA
Positive CT or MRI (N)	NA	12 (27.90%)	NA	10 (30.30%)	NA	8 (29.63%)	NA
PCS+ (N)	NA	NA	NA	19 (57.57%)	NA	15 (55.56%)	NA

Table 4.1 Demographic data summary of control subjects and mTBI patients. Note that patients who had positive CT readings may not necessarily be the same patients with positive MRI readings as findings could have resolved in the duration between CT and MRI scans.

Patient	Age	Gender	MR/CT Findings	ESA	LSA	Chronic
1	58	Male	**CT**: Small focus of IVH **MR**: Multiple punctate hemorrhages in right frontal lobe, left parietal lobe and left cerebellum on SWI; subtle DAI; IVH	✓	✓	✓
2	82	Male	**CT**: Extra axial blood on left side **MR**: Bleeds in occipital and temporal lobes in right frontal and parietal lobes; bilateral SDH and SAH; IVH; parenchymal contusions; right anterior temporal lobe hemorrhage; left posterior temporal lobe hemorrhage; DAI	✓		
3	45	Female	**CT**: Frontal pole parenchymal contusion; SDH along falx on right side **MR**: SDH on right side along temporal convexity; small SDH on left side; bilateral temporal pole contusions with hemorrhage; right frontal lobe hemorrhage; hemorrhage on frontoparietal junction		✓	✓
4	40	Male	**CT**: Negative **MR**: Small frontal and parietal punctate hemorrhages	✓	✓	✓
5	42	Male	**CT**: Convexity SDH with minimal local mass effect extending into falx **MR**: Convexity SDH; SAH visible on FLAIR image.		✓	✓
6	58	Male	**CT**: Negative MR: Right frontal lobe punctate hemorrhage	✓		
7	26	Male	**CT**: Small left frontal lobe cortical contusions **MR**: Punctate hemorrhages in left frontal lobe	✓	✓	
8	29	Male	**CT**: Subtle hemorrhages in bilateral frontal lobe **MR**: Lesion in left temporal lobe; punctate hemorrhages in bilateral temporal and frontal lobes; left parietal DAI	✓		✓
9	53	Male	**CT**: Small left frontal lobe cortical contusion **MR**: Negative	✓	✓	✓
10	53	Male	**CT**: Subcentimeter focus of increased attenuation in right thalamic nuclei **MR**: Small area of right posterior thalamic and sub ependymal hemorrhage; left dorsolateral brainstem contusion	✓	✓	✓
11	48	Male	**CT**: Contusion in left parietal lobe **MR**: Negative	✓		
12	19	Male	**CT**: Small focus of hemorrhage in right caudate **MR**: Subcentimeter right frontal lobe lesion	✓	✓	
13	48	Male	**CT**: Epidural hematoma; SDH or SAH blood in right side; frontal lobe contusions **MR**: Bilateral frontal lobe contusions, bleeds in some midbrain regions; bleeds in right temporal lobe; SAH in parietal lobe; IVH; EDH in right temporal lobe; EDH in right convexity	✓		
14	28	Female	**CT**: Left frontal lobe contusion and punctate hemorrhage **MR**: Left frontal lobe contusions		✓	
15	65	Female	**CT**: Negative **MR**: Left frontotemporal sudural collection with mass effect in left frontal and temporal lobes		✓	
16	26	Female	**CT**: SAH in right temporal lobe; right anterior temporal parenchymal contusion **MR**: Right brainstem punctate hemorrhage	✓		✓

Table 3.2 (*Shown on previous page*) Initial CT and MRI readings for patients with complicated mTBI. Checkmark indicates which mTBI group comprised the patient for comparison to healthy controls.
EDH – Epidural hematoma; DAI – Diffuse axonal injury; IVH – Intra-ventricular hemorrhage; SAH – Subarachnoid hemorrhage; SDH – Subdural hematoma; SWI – Susceptibility weighted imaging;

Although not all patients who obtained MRS at the 6 month point received both the ESA and LSA stage MRS, there was a significant overlap of subjects across all three examination time points. A total of seventeen patients (41.00 ± 19.58 yrs, 4 female, 4 complicated) were examined at all three time points.

Neuropsychological Assessment

The ANAM is a neurocognitive test that was developed by the U.S. military to test a number of cognitive domains including attention, concentration, reaction time, memory, processing speed, decision-making and executive function. It is a computer based assessment consisting of a battery of seven subtests. **Table 3.3** provides a summary of the subtests that were administered and the respective cognitive domains that they assess. The ANAM was found to be effective in detecting cognitive deficits and monitoring recovery in patients who have suffered sports-related concussions.(Bleiberg et al., 2004; Warden et al., 2001) In addition to the numerous scores obtained for each ANAM subtest, the software also computes a throughput score for each subtest which is indicative of an overall cognitive function in that cognitive domain. The throughput score is a metric that combines both speed and accuracy of the test response into a single score. The unit of throughput is "number of correct answers per minute." The weighted throughput score is a single value generated from the throughput scores of each subtest that measures performance across the entire battery with each subtest contributing proportionately to the weighted score.

ANAM Subtest	***Cognitive Function Assessed***
Simple Reaction Time	Visuomotor response time
Code Substitution	Visual search and learning
Procedural Reaction Time	Concentration and attention
Matching to Sample	Spatial processing and visuospatial working memory
Mathematical Processing	Working memory and attention
Code Substitution Delayed	Recognition and memory
Simple Reaction Time 2	Simple Reaction Time test repeated after completion of all other tests
Weighted Throughput	Single score that summarizes performance on all other tests

Table 3.3 List of ANAM subtests used in neurocognitive evaluation of subjects and the corresponding domain of cognitive function assessed.

Evaluation for Persistent Post Concussive Symptoms

Patient evaluation for the persistence of post concussive symptoms (PCS) at the LSA and chronic stages of mTBI was carried out using the Rivermead post-concussion symptoms questionnaire (RPQ).(King et al., 1995) Patients were deemed PCS positive (PCS+) if they experienced any of 4 or more of the major PCS symptoms (headaches, dizziness, sleep abnormalities, trouble concentrating, fatigue, memory problems and irritability). **Table 3.1** includes information on the number of patients with PCS persistence at the later stages of mTBI.

MR Examination

MR exams were performed on a Siemens Tim-Trio 3T MRI (Siemens Medical Solutions, Malvern, PA) scanner using a 12-channel receive only head coil. A high resolution T1-weighted-MPRAGE (TE = 3.44 ms, TR = 2250ms, TI = 900ms, flip angle = 9^o, resolution = 256 x 256 x256, FOV = 22 cm, sl. Thick. = 1.5mm) was acquired for anatomic reference. A 3D phase-encoded point-resolved spectroscopy (3D-PRESS) magnetic resonance spectroscopy imaging (MRSI) sequence was used to obtain spectroscopic data at a TE/TR of 135ms/1300ms over a FOV of 160 x 160 x 106mm^3 and a volume of interest (VOI) covering 106 x 106 x 48mm^3. The acquired resolution and interpolated resolution were 12x12x8 and 16x16x8 respectively with a total acquisition time of 7 minutes and 40 seconds. The VOI was oriented along the AC-PC line and centered about the *corpus callosum*. **Figure 3.1** shows the VOI selected (white box) as well as the location of one of the slices for the 3D-PRESS acquisition. Saturation bands were used to effectively suppress any chemical shift artifacts arising from lipids outside the volume of interest.

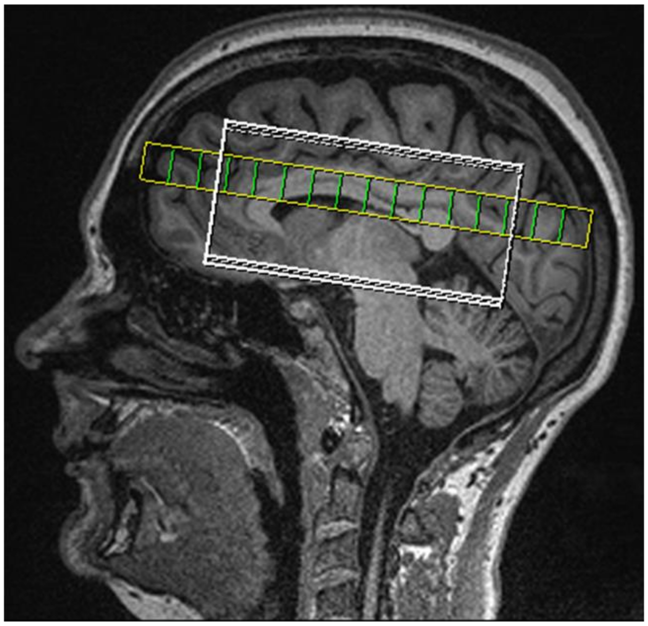


Figure 3.1. Positioning of the spectroscopic volume of interest (white box) in sagittal view. One slice in the MRSI grid is shown (yellow slab). Green bars represent MRSI voxels.

MRS Processing

MRS spectra were quantified offline using LCModel.(Provencher, 2001) LCModel is a frequency domain MRS processing software that compares the measured *in vivo* spectra to a linear combination of spectra from an *in vitro* basis set. The software uses a model that includes line shape and baseline functions, zero-order and first-order phase correction parameters and overall referencing of chemical shifts to fit the *in vitro* basis set to the measured *in vivo* spectra. The Cramer-Rao (CR) bounds generated by the software represent a lower limit of the statistical error of the fitted parameters. In the current study, measured metabolites with CR bounds lower than 15% were excluded from any further analysis. Because absolute Cre measurements were utilized in our metabolic analysis, we conducted additional calibration experiments to confirm that LCModel-generated measurements were in agreement with true creatine concentrations both in phantoms and *in vivo*. LCModel-generated creatine measurements linearly correlated with known concentrations in creatine phantoms made in-house (r^2 = 0.9976, p<<0.01). In addition LCModel-generated Cre measurements linearly correlated with Cre concentrations determined using a water reference scan within the same regions in the human brain (r^2 = 0.9031, p << 0.01).

Regions of Interest

The thalamus and the CSV were used as regions of interests (ROIs) for analysis as shown in **Figure 3.2** on a T1-Weighted MPRAGE image. For each region, spectroscopic measurements were acquired from two MRSI voxels (one voxel from each of the lateral regions of the brain) as shown in **Figure 3.2**. The average measurement from both voxels was used as a single value for each region in the analysis. Other regions in the brain parenchyma were examined for metabolic alterations due to mTBI but failed to yield any meaningful differences when compared to healthy controls.

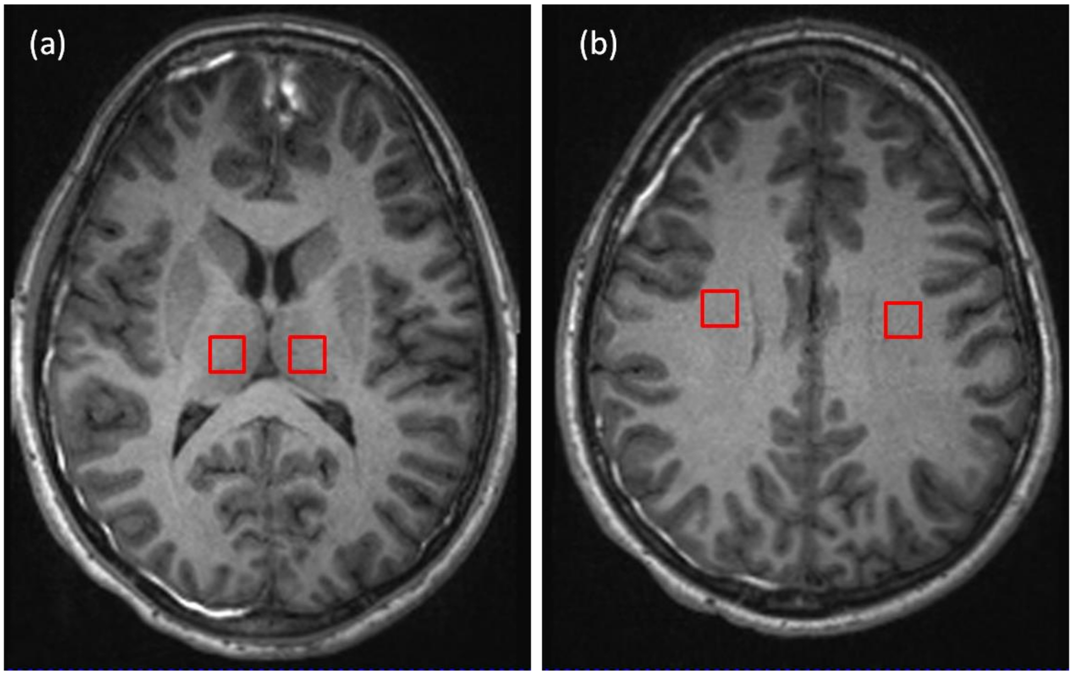


Figure 3.2. Anatomical location of the ROIs [(a) thalamus and (b) centrum semiovale] used to obtain spectroscopic information. Measurements from each lateral side of the ROI were averaged to generate a single value for each region.

Statistical Analysis

Statistical analysis was performed using *SPSS Statistics* version 21.0.0 software (IBM). Because not all the patients were followed through all time points, a two-tailed one-way analysis of variance (ANOVA) test was performed at each time point to determine neurometabolic differences between the patients and the control subjects. To understand the association between ESA metabolic measurements and chronic cognitive capability in mTBI patients, we carried out a two-tailed partial Pearson's correlation of metabolic measurements with ANAM scores with patient age and years-of-education as control variables. Corrections for multiple hypothesis testing was carried out using the positive false discovery rate (pFDR) method.(Storey, 2002) The q-value generated by this method is an analogue of the p-value and is an estimate of the minimum pFDR that can occur once a test statistic is rejected. Alpha was held at .05. Computations for this method were carried out with *MATLAB* R2013b.

Follow-up Analysis

We carried out a separate analysis for the seventeen patients who were followed up at all three time points. This analysis included the neurometabolic comparison to healthy controls and correlation of ESA neurometabolic measurements with chronic ANAM test scores, as was conducted for the overall population. In addition, we conducted a repeated measures ANOVA to evaluate neurometabolic changes in mTBI patients over time. The ESA, LSA and chronic time points were the three levels employed in the repeated measures analysis.

3.3 Results

Regional Metabolic Changes

Thalamus

There were no statistically significant changes in NAA/Cre measurements in the thalamus at any of the observed time points. A decreasing trend ($p = 0.085$, $q = 0.1771$) in NAA/Cre measurements was observed at the ESA stage for the mTBI group when compared to healthy subjects. At the LSA stage, a significant decrease in Cho/Cre ($p = 0.042$, $q = 0.0239$) was observed among the mTBI group when compared to healthy subjects. No statistically significant changes were observed in Cre measurements in the thalamus at any stage of mTBI when compared to healthy subjects. These results are summarized in **Figure 3.3**.

Centrum Semiovale

There were no statistically significant changes in NAA/Cre measurements in the CSV at any stage of mTBI when compared to healthy subjects. At the ESA stage of mTBI, a reduction in Cho/Cre was observed that was not significant (p = 0.082, q = 0.1771) when compared to healthy subjects. Further reduction in Cho/Cre was observed at the LSA stage (p = 0.017, q = 0.0193). Although the Cho/Cre ratio was still reduced by the chronic stage, this reduction was not statistically significant (p = 0.081, q = 0.1616). There were no changes in Cre measurements in the CSV at any stage of mTBI when compared to healthy subjects. These results are summarized in **Figure 3.3**.

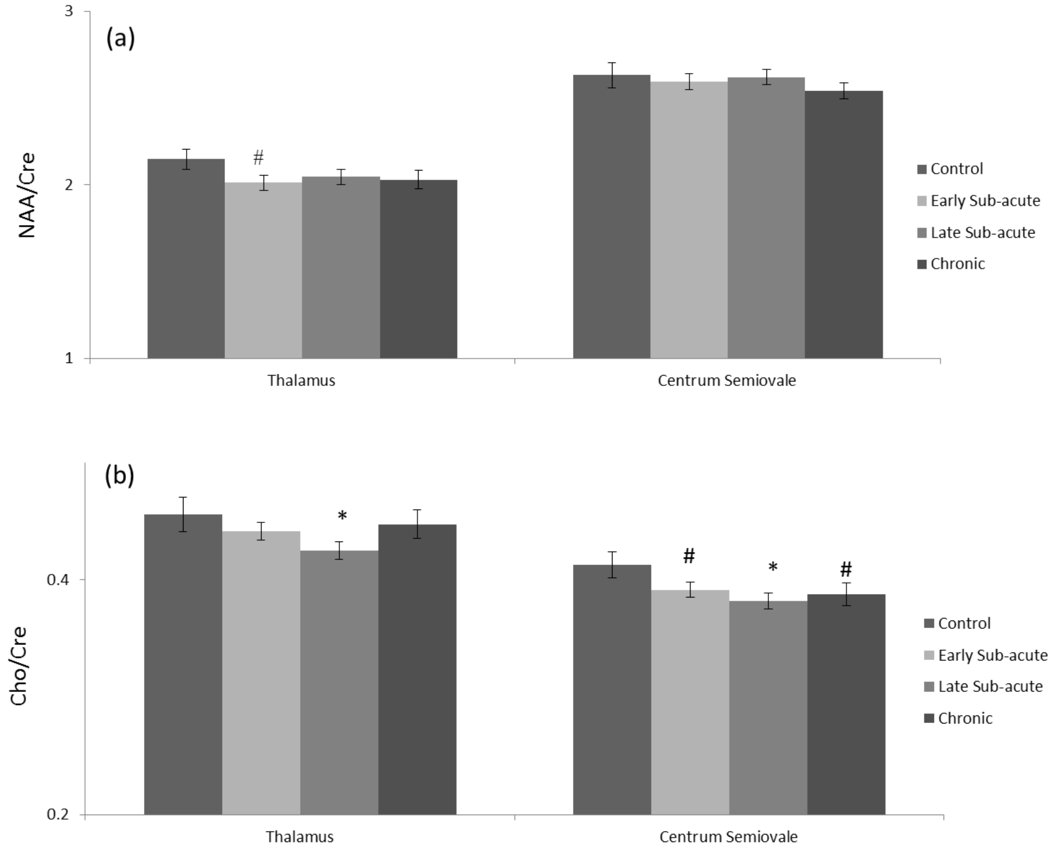


Figure 3.3. Graphical comparison of mTBI patients to healthy subjects using (a) NAA/Cre and (b) Cho/Cre measurements from the thalamus and centrum semiovale. Error bars indicate standard error. *#p<0.1; *p<0.05*

Association of Early Sub-acute Metabolic Measurements with Cognitive Outcome in mTBI Patients

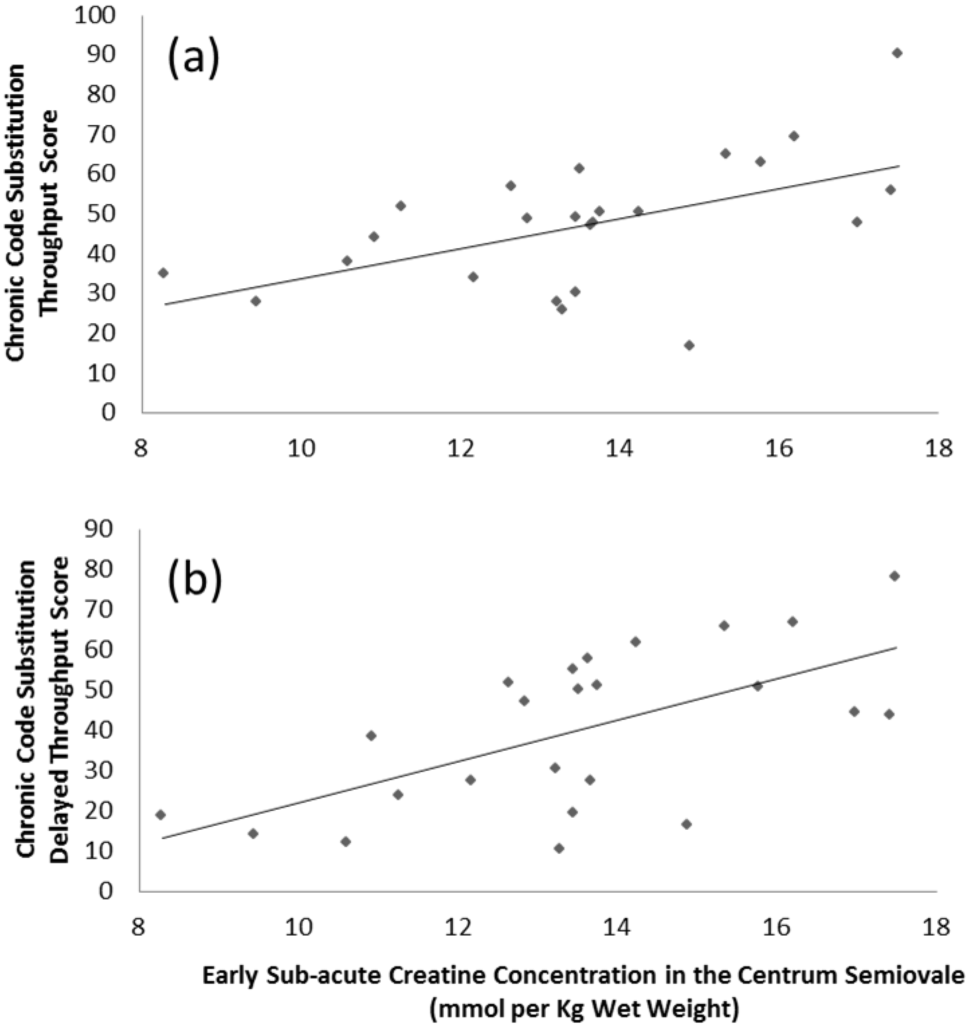
Although a significant positive correlation of ESA thalamic NAA/Cre was observed with the chronic Match-to-sample (r = 0.494, p = 0.017) and Simple Reaction Time (r = 0.462, p = 0.026) throughput scores on the ANAM, these correlations were no longer significant when corrected for patient age and years of education. However, ESA Cre measurements in the CSV did associate significantly with the chronic Code

Substitution Delayed throughput scores (r = 0.497, p = 0.019) and non-significantly with the chronic Code Substitution throughput scores (r = 0.391, p = 0.072) on the ANAM test after correcting for patient age and years of education. **Figure 3.4** summarizes the association of ESA Cre measurements in the CSV with chronic cognitive performance.

Follow-Up and Repeated Measures Analysis

Separate analyses were carried out for the seventeen patients who were examined at all three examination time points. No significant changes in NAA/Cre or Cre measurements were found at any of the examination time points for this group. We observed a trend of reduced thalamic Cho/Cre (p = 0.053, q = 0.0169) at the ESA stage of mTBI compared to healthy subjects. There were no changes in thalamic Cho/Cre at the LSA or chronic stage of mTBI when compared to healthy subjects. In the CSV, we observed a decrease in Cho/Cre at the ESA (p = 0.033, q = 0.0169) and chronic (p = 0.026, q = 0.0175) stages of mTBI. A decreasing trend of reduced choline (p = 0.076, 0.5694) in the CSV was observed in the LSA stage of mTBI when

Figure 3.4. Correlation of early sub-acute creatine concentration in the centrum semiovale with chronic cognitive ability measured by the (a) Code Substitution throughput score (r = 0.391, p = 0.072), (b) the Code Substitution Delayed throughput score (r = 0.497, p = 0.019).



compared to healthy controls. **Figure 3.5** summarizes these results

We also carried out an age and years-of-education corrected correlation of ESA neurometabolic measurements with chronic ANAM scores in mTBI patients. This analysis revealed a positive non-significant association of thalamic NAA/Cre with the Match-to-Sample throughput scores (r = 0.474, p = 0.087) and a positive non-significant association of thalamic Cre with the Simple Reaction Time throughput scores (r = 0.505, p = 0.066). These associations did not survive multiple hypothesis testing.

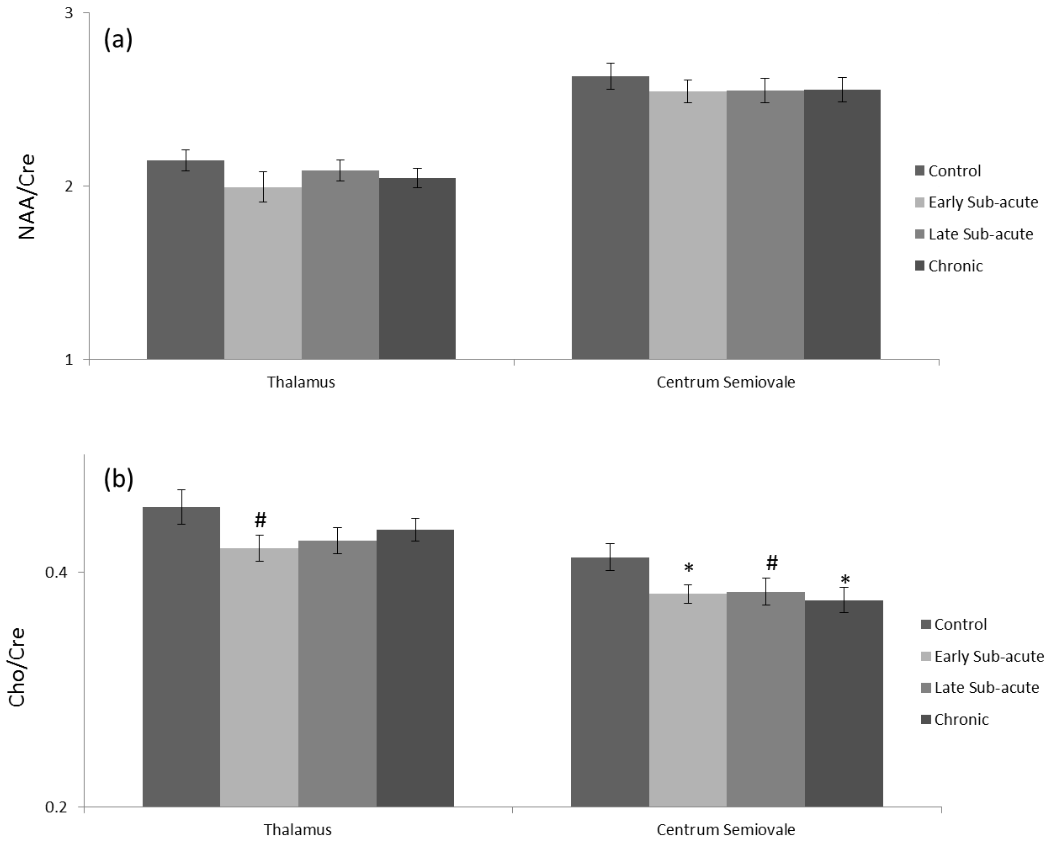


Figure 3.5. Graphical comparison of mTBI patients to healthy subjects using (a) NAA/Cre and (b) Cho/Cre measurements from the thalamus and centrum semiovale. Patients included in this comparison were followed up at all 3 examination time points (N = 17). Error bars indicate standard error. *#p<0.1; *p<0.05*

A repeated measures analysis across all 3 time points for the mTBI patients revealed no significant neurometabolic changes as mTBI progressed.

3.4 Discussion

The main objective of this study was to identify neurometabolic abnormalities at different stages of mTBI and to determine the association of neurometabolic values at the ESA stage with cognitive performance at the chronic stage of mTBI. The impact of mTBI is diffuse in nature, hence abnormalities detected are likely to manifest in a number of anatomical regions within the brain parenchyma.(Kirov et al., 2013) Nonetheless, intrinsic biomechanical properties of certain tissue types allow them to be more vulnerable to traumatic impact, making them regions of higher sensitivity for the detection of pathology. Moreover, the complex inter-connectivity of brain structures allows even regions that are not initially harmed at the instant of trauma to undergo eventual decline in structural and functional integrity, owing to such phenomenon as Wallerian degeneration.(Cecil et al., 1998a; Cecil et al., 1998b)

Our analysis showed that the CSV and thalamus are highly metabolically-sensitive to mTBI. Previous work focusing on diffusion tensor imaging (DTI) evaluation of TBI has indicted the CSV as a region that is highly vulnerable to brain injury.(Inglese et al., 2005; Kennedy et al., 2009) Its proximity and inter-connection with the cortex,(Berman et al., 2004) which essentially is the impact site of trauma-induced shearing forces make it a chief casualty of TBI. Although the thalamus is a much deeper brain structure, it has many projections to various cortical regions,(Behrens et al., 2003) making it a region of interest for studying abnormalities that ensue from TBI.(Kirov et al., 2007). The most novel findings realized in this study were: (a) statistically significant decreases in Cho/Cre values measured in the thalamus and CSV at the LSA stages of injury for the mTBI group when compared to healthy subjects, and (b) positive associations of ESA Cre measurements in the CSV with chronic neuropsychological test performance in mTBI patients after controlling for patient age.

Neurometabolic Differences between Healthy Subjects and mTBI Groups

Our analysis revealed a decreasing trend in measured NAA/Cre levels in the thalamus at the ESA stage of injury when compared to healthy subjects. NAA is an acetyl-amino-acid derivative that resonates at 2.01ppm in the ^{1}H-NMR spectrum. While its exact function *in vivo* is not fully understood, NAA is believed to serve as an osmolyte in neuronal tissue,(Baslow, 2003) and a marker of neuronal integrity and viability.(Demougeot et al., 2001) NAA also is reported to be a source of acetate for myelin synthesis in glial cells.(Burri et al., 1991; Chakraborty et al., 2001) It is produced in the neuronal mitochondria and is predominantly located within the neurons.(Clark, 1998; Moffett et al., 2007) In MRS, the individual signals of NAA and NAAG are difficult to resolve at clinical field strength hence is reported as a combination of NAA and NAAG. While the decrease in NAA observed herein did not reach statistical significance, such changes can be considered worthy of note considering that NAA is always lowered in TBI. The occurrence of decreased NAA/Cre values for mTBI patients agrees well with previously reported changes in NAA/Cre following TBI of all severities.(Garnett et al., 2000b),(Cecil et al., 1998a; Govindaraju et al., 2004),(Brooks et al., 2000; Friedman et al., 1998; Garnett et al., 2000a; Govind et al., 2010; Marino et al., 2007) Trauma induced disruption of neuronal integrity gives rise to mitochondrial dysfunction and a resulting compromise in NAA synthesis.(Signoretti et al., 2001; Vagnozzi et al., 1999; Verweij et al., 2000) A number of studies have linked NAA decrease and recovery to a concomitant fall and rise in ATP levels suggesting that NAA production in the cell is tied to the overall well-being of the mitochondria (the energy factory of the cell).(Signoretti et al., 2008; Signoretti et al., 2001) Wallerian degeneration and axonal disintegration have also been implicated as reasons for the diffuse nature of NAA decrease in the brain due to injury.(Cecil et al., 1998a; Cecil et al., 1998b) While we observed a decreasing trend in NAA/Cre at the ESA stage of mTBI, such changes did not exist at later stages of injury suggesting that the initial perturbation due to injury was recoverable towards normative levels in this patient population. Such recovery in the more chronic stages of TBI is also well documented in the literature,(Destefano et al., 1995; Schuhmann et al., 2003) suggesting that the irreversible loss of neurons may not be the primary reason for NAA depression but that it may be associated with mitochondrial dysfunction experienced by disrupted, yet viable cells that are able to recover after trauma. Similar recovery of NAA after an initial depression has also been observed in other cerebral pathologies such as stroke and disseminated encephalomyelitis.(Bizzi et al., 2001; Demougeot et al., 2003)

Total choline (Cho), whose signal is comprised of free choline and choline containing compounds [(Glycerophosphocholine (GPC) and Phosphocholine (PC)] and is measured at 3.2 ppm in the ^{1}H-NMR spectrum, has been shown to be altered in the progression of TBI. Choline compounds are an integral part of cell membrane phospholipids and myelin in brain tissue. Hence, changes in detected choline have been interpreted to indicate cell membrane turnover and/or myelin breakdown.(Chang et al., 1996; Davie et al., 1993; Yeo et al., 2006). Our studies indicate a decrease in Cho/Cre levels in both the thalamus and CSV at the LSA stage of mTBI. While significant changes in Cho are not typically reported in mTBI,(Gasparovic et al., 2009; Kirov et al., 2007; Vagnozzi et al., 2010; Vagnozzi et al., 2008) a number of mTBI studies have found elevated Cho levels associated with mTBI. (Govind et al., 2010; Govindaraju et al., 2004; Sarmento et al., 2009) Our observed depression of Cho/Cre levels in the mTBI group however, has not been reported in previous mTBI studies. The reasons for such contrast in findings could include differing patient cohorts, examination window and number of subjects included in the analysis. For example, Govind et al,(Govind et al., 2010) included moderate TBI (GCS 10 – 12) patients in the analysis and Sarmento et al,(Sarmento et al., 2009) examined patients strictly with post traumatic headaches. The findings of these studies could be indicative of post-traumatic choline elevations being associated with more severe or symptomatic occurrences of TBI. Govindaraju et al,(Govindaraju et al., 2004) included a smaller patient population (16 mTBI patients) with a much wider examination window (2-30 days) for their mTBI study. Nonetheless, significantly lowered levels of Cho have previously been observed in experimental models of TBI.(Harris et al., 2012; Schuhmann et al., 2003) Similar evidence of Cho depression has been observed in human studies of alcoholism,(Bendszus et al., 2001; Durazzo et al., 2004; Parks et al., 2002) smoking,(Durazzo et al., 2004) combat-related PTSD,(Freeman et al., 1998) stroke,(Graham et al., 1993) and depression.(Renshaw et al., 1997) Indeed, the reduction of Cho/Cre may be suggestive of an increase in Cre and/ or a decrease in Cho in the same region. We postulate that our observed decrease in Cho/Cre resulted from a decrease in Cho as no significant changes were observed in Cre measurements in any of the associated regions. Several studies have concluded that the elevation of Cho implies an inflammatory response(Brooks et al., 2000),(Brooks et al., 2001; Friedman et al., 1999) and possibly glial cell proliferation.(Garnett et al., 2000a; McBride et al., 1995) However, the lowering of Cho in mTBI observed herein is likely due to degradation and eventual phagocytotic clearance of choline-containing membrane fragments that dissociate from the cell upon the occurrence of trauma.(Greenfield et al, 2008; Sbarra and Karnovsky, 1959) The "mild" nature of injury however might not require the desperate inflammatory response that often leads to rapid glial cell proliferation resulting in increased Cho observed by other investigators. The eventual restoration or lack of abnormality in the chronic stage is likely indicative of cellular membrane repair and/or re-myelination of damaged myelin sheaths in white matter.

It is important that such a novel finding as this be taken with caution as we cannot definitively rule out the possibility that observed changes in measured Cho also may be due to reasons unaccounted for such as changes in relaxation properties of Cho induced by injury and the associated changes in the cellular environment. Indeed previous studies have shown that cellular processes such as edema can alter the relaxation times and apparent concentrations of *in vivo* metabolites.(Kamada et al., 1994) Changes in relaxation times of neurometabolites have been observed in other cerebral pathologies such as bipolar disorder and schizophrenia.(Ongur et al., 2010) An in-depth understanding of the dependence of metabolite relaxation on the occurrence and severity of brain injury will require studies involving multiple echo time MRS acquisitions.

The assumption that the metabolite Cre is stable in TBI and various other neurodegenerative disorders is widespread in spectroscopy literature.(Tartaglia et al., 2002) Hence the combined signal of creatine and phosphocreatine which is measured at about 3.03ppm in the ^{1}H-MRS spectrum is used as an internal concentration reference to normalize other metabolite concentrations.(Yeo et al., 2011) Nonetheless, Cre levels have been shown to deviate from normative values in other cerebral pathologies including Parkinson's disease and multiple sclerosis.(Hattingen et al., 2009; Inglese et al., 2003) A previous study has shown increased Cre in mTBI in white matter regions.(Gasparovic et al., 2009) A separate study showed decreased Cre in the more chronic stages of severe TBI.(Friedman et al., 1998) In the current study, no statistical differences were found between Cre levels of the mTBI group and healthy subjects for either region analyzed, ultimately agreeing with the majority of TBI spectroscopic literature. As detected Cre in the brain is believed to originate from intracellular stores of the metabolite, the lack of abnormality in Cre levels observed herein could be indicative of the absence of significant cell death or astroglial proliferation that occurs with more severe TBI, in which case Cre could play a role as a surrogate marker to NAA of cell density.

Correlation of Early Sub-acute Neurometabolic Information with Chronic Neuropsychological Test Performance

Prior to correcting for patient age and years-of-education, we observed a significant positive correlation between ESA thalamic NAA/Cre and two ANAM subtest scores. These correlations were no longer significant after age and years-of-education were accounted for as control variables. While age-related decline in cognitive function has been well established, the effect of years-of-education could reflect the role that pre-injury socio-economic status plays in long-term cognitive outcome of TBI as has been shown by previous studies.(Hoofien et al., 2002) Our analysis revealed a positive association (within mTBI patients) of Cre levels in the CSV with two of the ANAM subtests after correcting for age and years of education as a covariates. These correlations were non-existent in the healthy control group suggesting that the association is unlikely a pre-morbid phenomenon. Hence while alterations in Cre levels may not be a direct effect of mTBI (as no differences in Cre were realized herein between mTBI patients and the healthy control group), these findings do suggest that cellular Cre plays a role in long term cognitive recovery after mTBI. A number of previous studies have looked at the association of MRS metabolites to neuropsychological information in the TBI population.(Babikian et al., 2006; Brooks et al., 2000; Friedman et al., 1999; Friedman et al., 1998; Garnett et al., 2000a; Gasparovic et al., 2009; Govind et al., 2010; Yeo et al., 2006) The majority of previous reports have focused on NAA a marker that modulates cognitive function post TBI. One study showed that Cre levels in normal-appearing occipitoparietal white matter and normal appearing gray matter in patients with moderate to severe trauma (mean 53 ± 23 days post injury) associated positively with neuropsychological test scores indicative of overall cognitive function obtained within 24 hrs of the MRS scan.(Friedman et al., 1998) Another study showed a positive association of gray matter Cre levels in moderate to severe TBI patients at 1.5 months of injury to overall neurocognitive performance at the same time point.(Brooks et al., 2000) To our knowledge, this is the first report revealing the association of ESA Cre measurements with chronic neurocognitive performance in an mTBI cohort. The evolution of Cre as a marker of cognitive function could present a more specific metabolic approach to mTBI prognosis as changes in neuronal marker NAA can also be associated with other cellular processes such as lipid synthesis and reversible mitochondrial dysfunction that do not necessarily underlie long term cognitive function. Changes in total cellular creatine (phosphocreatine + creatine) levels can be more specific to cell loss (even though less prevalent in mTBI) and hence will be more telling of long term functional outcome. Further support for the importance of creatine in cognitive recovery comes from a previous study that revealed that dietary supplementation of creatine can aid recovery in TBI.(Sakellaris et al., 2006; Sullivan et al., 2000) Additional studies with well-defined hypotheses, carried out in a separate mTBI cohort might be necessary to confirm the MRS-ANAM correlations realized herein as the associations did not survive multiple hypothesis testing.

Limitations of this Study

The results presented here are from a controlled prospective study on the sequelae of mTBI. This is a cross-sectional study across various time points following mTBI and provides a general view of the changes in neurometabolites in the thalamus and CSV. The inclusion criteria for mTBI patients employed herein is solely based on initial GCS scores as information on LOC and PTA was either unavailable or subject to ambiguity when self-reported. Indeed additional criteria such as is defined by the American Congress of Rehabilitation medicine (ACRM) include an LOC duration of less than 30 minutes and PTA of less than 24 hours may be necessary for a more thorough evaluation of mTBI. While the study does provide valuable insights into the changes in biochemistry of the brain at a population level, the applicability of these findings to an individual subject will require careful examination of data from a very large population, preferably from different sites. However, it is fair to say that the population size of each time point analysis permits us to reach an objective conclusion about the metabolic and neuropsychological impact of mTBI at each of these time points.

Conclusion

A cross-sectional comparison between mTBI patients and healthy controls revealed decreasing trends in thalamic NAA/Cre levels at the ESA stage of mTBI and decreases in Cho/Cre occurring in the thalamus and CSV at the LSA stage of mTBI. Our analysis also revealed positive correlations between ESA Cre levels and chronic cognitive performances. With mTBI being perhaps the least understood form of brain injury due to the lack of objective neuroimaging markers that underlie the pathology, these findings will jointly help to provide a more efficient evaluation of mTBI patients. The prognostic value of this study will allow caregivers to give accurate advice to patients on when to resume activity, avoiding the risk of repeated injury.

Chapter 4: Dual Echo Magnetic Resonance Spectroscopy Imaging: application to Traumatic Brain Injury

4.1 Introduction

Magnetic Resonance Spectroscopy Imaging (MRSI) allows localized measurement of *in vivo* metabolic composition from a multidimensional array of spatial locations. In earlier and more conventional forms of MRSI,(Brown et al., 1982) spin excitation is followed by a pulsed gradient which encodes spatial information, after which the free induction decay (FID) is sampled in the absence of any field gradients. More recently, other MRSI approaches have been developed to either reduce total scan time or increase the amount of data acquired while maintaining comparable acquisition durations. Fast proton spectroscopic imaging (also known as Turbo spectroscopic imaging [TSI] or RARE mode spectroscopic imaging) is analogous to echo planar imaging (EPI) in the sense that the multiple echoes generated by refocussing the initial excitation are phase encoded differently, ultimately allowing reduced overall scan time (Duyn and Moonen, 1993; Dydak et al., 2006; Traber et al., 1997). In Carr-Purcell-Meiboom-Gill (CPMG) mode spectroscopic imaging (SI), the multiple echoes generated by repeated refocussing are phase encoded uniformly so that multiple data sets (each at different echo time [TE] of acquisition) can be acquired within durations comparable to a conventional single echo (SE) MRSI method.(Dreher and Leidfritz, 1995; Kiefer et al., 1998; Mulkern et al., 1996) The increased information afforded by CPMG mode SI can be used to (i) increase the signal-noise-ratio (SNR) and the number of metabolite signals measurable in the spectrum using short TE acquisitions (ii) reduce problems associated with lipid and macromolecular background convolution such as baseline determination using long TE acquisitions (iii) reduce problems associated with peak overlap by taking advantage of intrinsic spin coupling properties at different TEs (iv) measure T2 relaxation times of different metabolites using consecutive echoes in the echo train and (v) determine the optimum TEs for measurement of different metabolites. In both RARE and CPMG mode SI however, there exists trade-offs between SNR, spectral resolution and acquisition duration that is primarily imposed by intrinsic T2 decay and the need for increased sampling rates for one or more of the sampled echoes. Another form of SI which is becoming the technique of choice for whole-brain metabolic imaging is echo planar spectroscopic imaging (EPSI)(Posse et al., 1994; Posse et al., 1995). In EPSI the readout occurs concurrently with a rapidly changing read-out gradient allowing for simultaneous encoding of spectral and spatial information, ultimately leading to a reduction in overall scan time and the prevention of motion-induced artifacts in the acquisition.

Herein we describe a novel dual-echo MRSI (DE-MRSI) acquisition that is capable of simultaneously acquiring both short and long TE data sets with the same scan time as a standard clinical SE MRSI method. This technique was developed by modification of a standard SE point resolved spectroscopy (PRESS) sequence. The acquisition method employs the CPMG approach to SI while optimizing the data quality achievable by choosing acquisition parameters that best suit the intrinsic properties of each data set. We compare data obtained from the DE-MRSI sequence to data from the standard SE sequence in order to evaluate measurement differences due to the acquisition modification. Also, repeatability analyses were carried out to determine the reproducibility of the acquired measurements. In order to validate the usefulness of this technique in research and clinical studies, the DE-MRSI sequence was tested on traumatic brain injury (TBI) patients as well as neurologically healthy subjects. Measurements from the two groups were cross-sectionally compared to determine how well pathology induced changes can be detected using this method.

4.2 Methods

Sequence Development

The DE-MRSI sequence was developed by modification of a Siemens PRESS SE MRSI sequence. Modifications include an additional 180^{o} pulse after the initial readout to refocus the original signal. The additional pulse is played out simultaneously with slice-selective gradients and flanked by identical gradient crushers immediately before and after the pulse so that only unambiguous information from the refocused signal is localized. **Figure 4.1** shows the pulse sequence diagram for the DE-MRSI technique. Furthermore, in order to accommodate the dual acquisition scheme, the short TE echo is strategically sampled with a dwell-time of 200µs [receiver bandwidth (BW) = 5 kHz, 5 times the BW of the standard acquisition]. The dwell-time for the long TE acquisition remains consistent with the standard acquisition at 1000µs (BW = 1 kHz). Theoretically speaking, the implications of increasing the BW include reduced SNR and reduced digital or spectral resolution for the acquisition. However, with the approach presented herein, the intrinsically high SNR of the short TE acquisition can allow losses in SNR associated with the increased sampling rate to be ultimately tolerated without adverse loses in the sensitivity of the measurement. Furthermore, because short T2 metabolites are of primary interest in short TE acquisitions, the high damping rate and broad spectral line width of short T2 metabolites can allow adequate characterization of the MRS signal despite the reduced digital resolution. The source code for the DE-MRSI sequence has been included in the appendix.

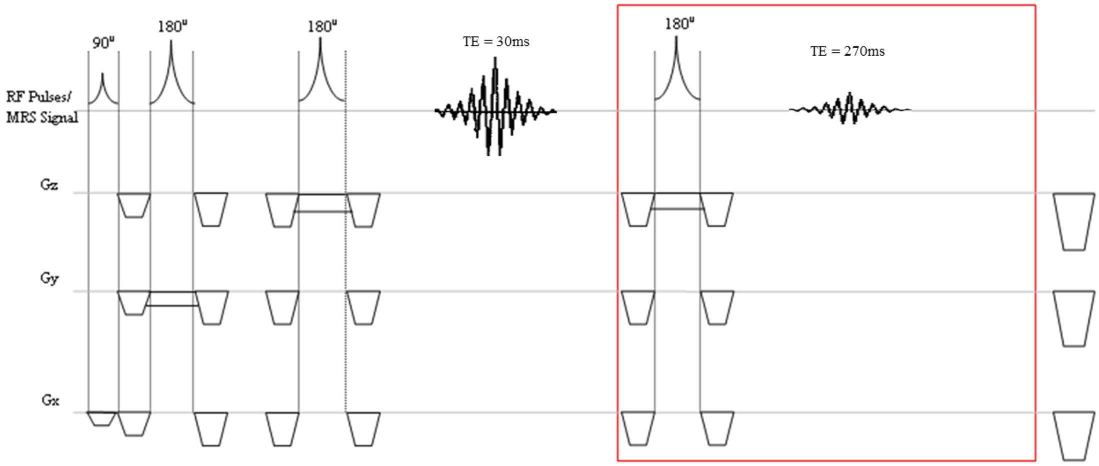
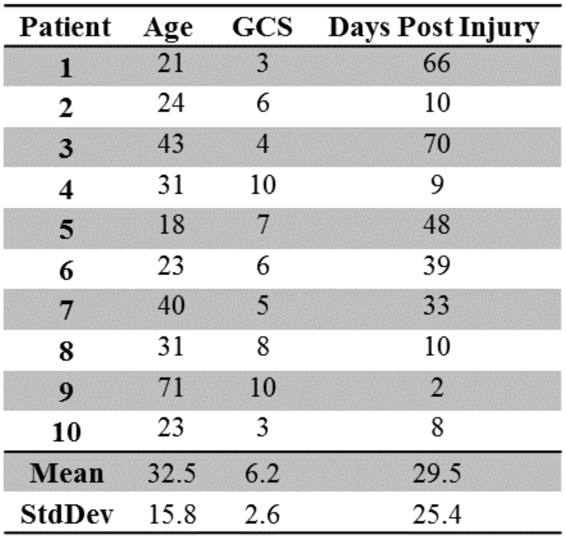


Figure 4.1. Pulse sequence diagram for DE-MRSI technique. Red box delineates modifications made to standard PRESS technique.

Data Acquisition

All scans were implemented on a Siemens Tim-Trio 3T MRI (Siemens Medical Solutions, Malvern, PA) scanner using a 12-channel receive only head coil. A high resolution T1-weighted-MPRAGE (TE = 3.44 ms, TR = 2250ms, TI = 900ms, flip angle = 9^{o}, resolution = 256 x 256 x256, FOV = 22 cm, sl. Thick. = 1.5mm) was acquired for anatomic reference. MRSI scan parameters were as follows: TE1 = 30ms; TE2 = 270ms; TR = 1320ms; FOV = 160x160x106; VOI = 106x106x48; acquired resolution = 12x12x8; interpolated resolution = 16x16x8; total acquisition time = 7min 40 secs; vectors size = 512. TE1 for the DE-MRSI acquisition had a bandwidth of 5KHz while the SE acquisition was carried out with a BW of 1KHz. TE2 for both acquisitions had identical bandwidths of 1KHz. Phantom measurements were carried out on a phantom made in-house with metabolite composition identical to a human brain. Ten moderate/severe TBI patients (GCS 3-10, 29.5 ± 25.4 days post injury) and ten healthy control subjects were scanned using the DE-MRSI sequence. **Table4.1** summarizes the demographic information of the healthy subjects and TBI patients examined. **Figure 4.2** shows the VOI selected (white box) as well as the location of one of the slices for the 3D-PRESS acquisition. Saturation bands were used to effectively suppress any chemical shift artifacts arising from lipids outside the VOI.



Patient	Age	GCS	Days Post Injury
1	21	3	66
2	24	6	10
3	43	4	70
4	31	10	9
5	18	7	48
6	23	6	39
7	40	5	33
8	31	8	10
9	71	10	2
10	23	3	8
Mean	32.5	6.2	29.5
StdDev	15.8	2.6	25.4

Table 4.1 Demographic data summary of TBI patients used for group comparison to healthy control subjects. Glasgow coma scale score (3-15) indicates the level of consciousness at the time of admission with 3 indicating deep unconsciousness. Days post injury indicates the number of days after injury that MR examination was carried out.

Post-processing and Analysis

Metabolite quantification was performed using *LCModel.*(Provencher, 2001) Linear regression analysis was used to compare measurements from the standard SE sequence to measurements from the DE-MRSI sequence in both phantom and human brain. A standard deviation (StdDev) and percentage standard deviation (%StdDev) was used to determine the reproducibility of measurements acquired using the DE-MRSI technique. The %StdDev is the standard deviation given as a percentage of the mean of the measurement. Measurements from six consecutive trials were employed in the analysis. The regions of the human brain analyzed for metabolic alterations due to TBI are shown in **figure 4.3**. Measurements in both sides of lateralized brain regions were averaged to produce one single value for analysis. A one-way analysis

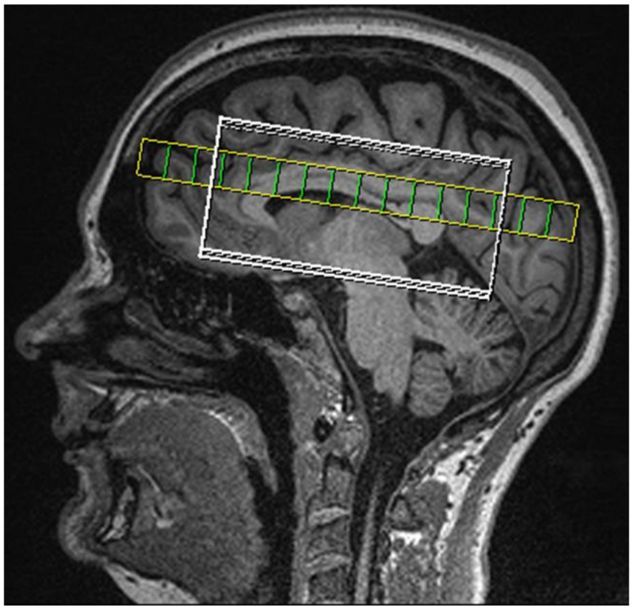


Figure 4.2. Positioning of the spectroscopic volume of interest (white box) in sagittal view. One slice in the MRSI grid is shown (yellow slab). Green bars represent MRSI voxels.

of variance (ANOVA) test was used to determine group differences in metabolite-ratio values between TBI patients and the healthy control group.

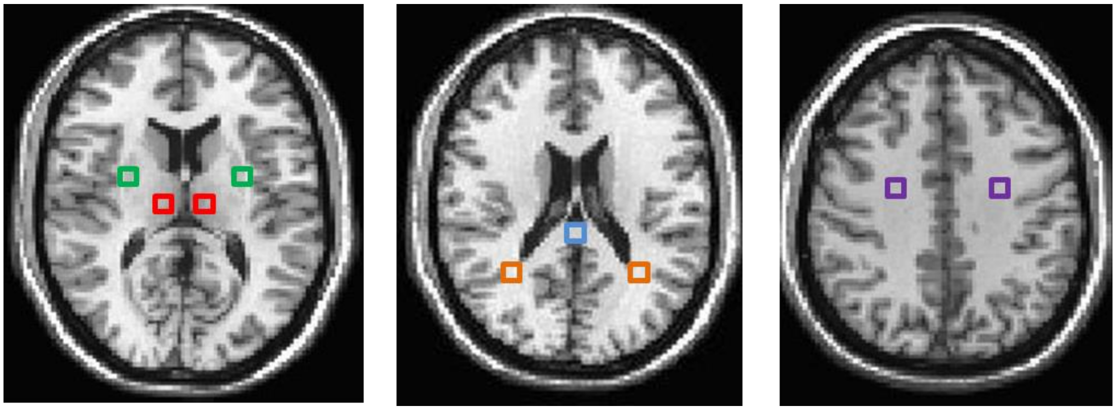


Figure 4.3. Neuroanatomical regions analyzed for TBI induced metabolic changes. Measurements in both sides of lateralized brain regions are averaged to produce one single value for analysis.

🟩 Putamen
🟥 Thalamus
🟦 Splenium
🟩 Periventricular white matter
🟪 Centrum semiovale

4.3 Results

Pulse Sequence Comparison and Repeatability Analysis

Linear regression analysis showed that measurements from the DE-MRSI method were strongly correlated to measurements from the standard SE method in both phantom and human brain. **Table 4.2** summarizes the results of the correlation for different metabolites measured in both long and short TE data sets. While short TE measurements were comparable between both sequences, a loss in the signal measured with the long TE DE-MRSI acquisition was observed when compared to its standard SE counterpart. The repeatability analysis showed reproducible results in consecutive measurements in both phantom and human brain, consistently yielding minimal %StdDev (≤10%). **Table 4.3** summarizes the repeatability analysis.

	TE(ms)	**NAA**		**Cho**		**mI**	
		r	*m*	*r*	*m*	*r*	*m*
^{1}H-MRS Phantom	*30*	0.98	1.41	0.99	1.33	0.98	1.08
	270	0.99	0.57	0.99	0.55	N/A	N/A
Human Brain	*30*	0.96	1.23	0.92	1.22	0.89	1.14
	270	0.91	0.55	0.91	0.57	N/A	N/A

Table 4.2 Linear correlation between measurements from DE-MRSI sequence and standard SE-MRSI sequence in different voxels of ^{1}H-MRS phantom and human brain. *r* and *m* respectively represent the Pearson's coefficient and slope of the linear correlation "measurement$_{\text{DE-MRSI}}$ = *m* * measurement$_{\text{SE-MRSI}}$". mI signal was not detected at TE = 270ms

	TE(ms)	**NAA**			**Cho**		
		Mean	*StdDev*	*%StdDev*	*Mean*	*StdDev*	*%StdDev*
^{1}H-MRS Phantom	*30*	0.99	0.08	7.66	0.41	0.01	3.23
	270	1.37	0.02	1.63	0.16	0.02	10.74
Human Brain	*30*	1.82	0.02	0.99	0.38	0.00	1.30
	270	3.97	0.20	5.01	0.55	0.05	8.27

Table 4.3 Repeatability analysis for DE-MRSI technique on phantom and human brain. Number of measurements = 6. StdDev = Standard deviation, %StdDev is the standard deviation in terms of percentage of the mean.

Comparison of TBI Patients to Healthy Subjects with Short TE DE-MRSI Data

Figure 4.4a shows the comparison between short TE DE-MRSI spectra acquired from the spleniums of a severe TBI patient and a healthy subject. The spectra show a notable increase in measured mI signal and a decrease in NAA signal for the TBI patient when compared to the healthy subject. At the group level, there was an observable trend of reduced NAA/Cre values in all the neuroanatomical regions analyzed for the TBI group when compared to healthy subjects.

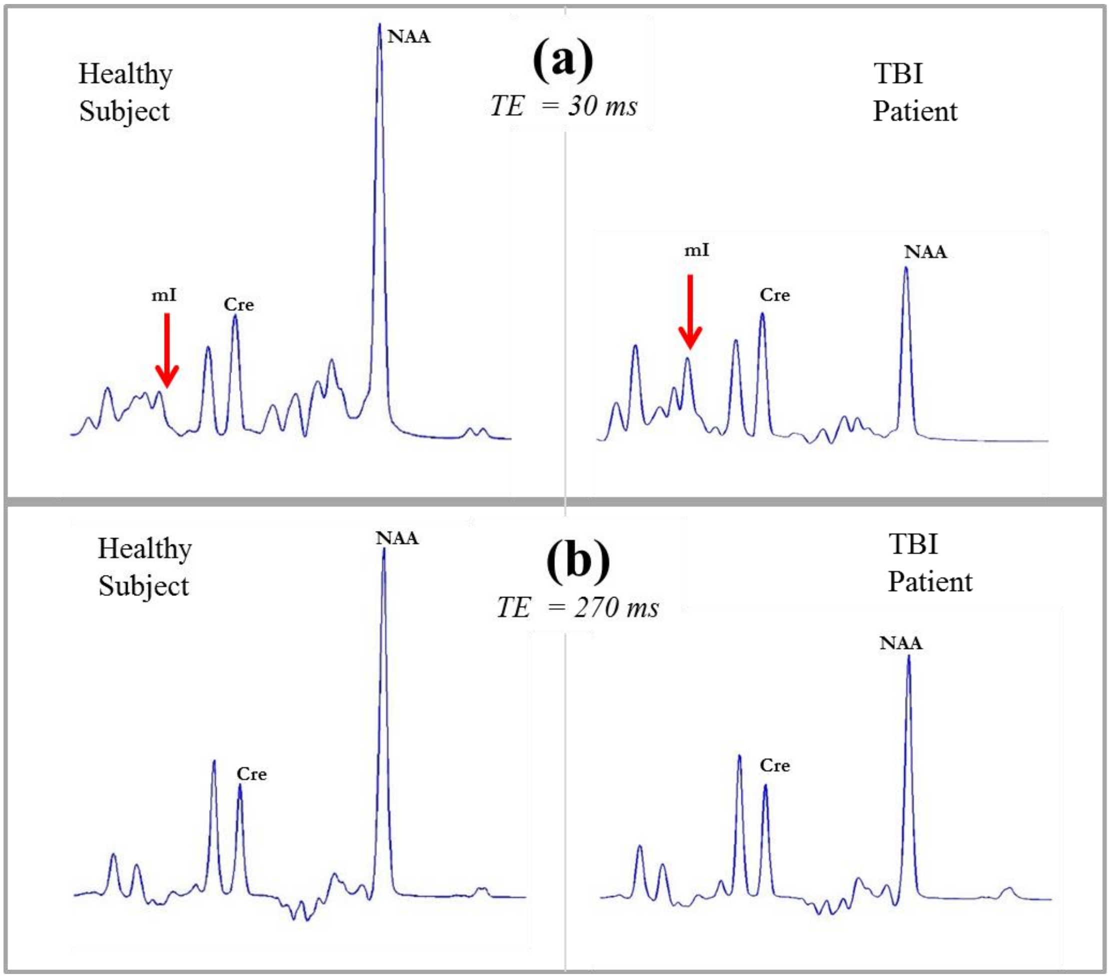


Figure 4.4 (a) Short and (b) long echo time ^{1}H-MRS spectrum from periventricular white matter acquired using DE-MRSI on a healthy control subject (left) and a traumatic brain injury subject (right) [GCS = 3, motor vehicle accident, 68 days post injury, 22 years old]. Notable decreases in NAA can be observed in the TBI patient when compared to the healthy control subject using both long and short echo time spectra. Also, an increase in mI can be observed in the TBI patient compared to the healthy control subject using the short echo time spectra

Furthermore, statistically significant decreases in NAA/Cre values measured in the splenium (p = 0.003) and centrum semiovale (p = 0.030), as well as a non-significant trend of decreased NAA/Cre in the thalamus (p = 0.072 was observed in TBI patients when compared to healthy subjects. *Myo*-inositol measurements revealed a general trend of increased mI/Cre levels in all neuroanatomical regions for the TBI group when compared to healthy subjects. Statistically significant increases in mI/Cre values measured in the *splenium* (p = 0.026), thalamus (p = 0.043) and putamen (p = 0.003) was observed in the TBI group when compared to healthy subjects. There were no statistically relevant differences in Cho/Cre measurements between TBI patients and healthy subjects. **Figure 4.5** summarizes these results.

Comparison of TBI Patients to Healthy Subjects with Long TE DE-MRSI Data

Figure 4.4b shows the comparison between long TE DE-MRSI spectra acquired from the spleniums of a severe TBI patient and a healthy subject. The spectra show a notable decrease in the NAA signal for the TBI patient when compared to the healthy subject. At the group level, there was a general trend of reduced NAA/Cre levels in all neuroanatomical regions for the TBI group when compared to healthy subjects, ultimately agreeing with the short TE data. More so, statistically significant decreases in NAA/Cre values measured in the splenium (p = 0.039) and periventricular white matter (p = 0.027) and a non-significant trend of decreased NAA/Cre in the centrum semiovale (p = 0.079) was observed in TBI patients when compared to healthy subjects. There were no statistical changes in Cho/Cre measurements between TBI patients and healthy subjects. **Figure 4.6** summarizes these results.

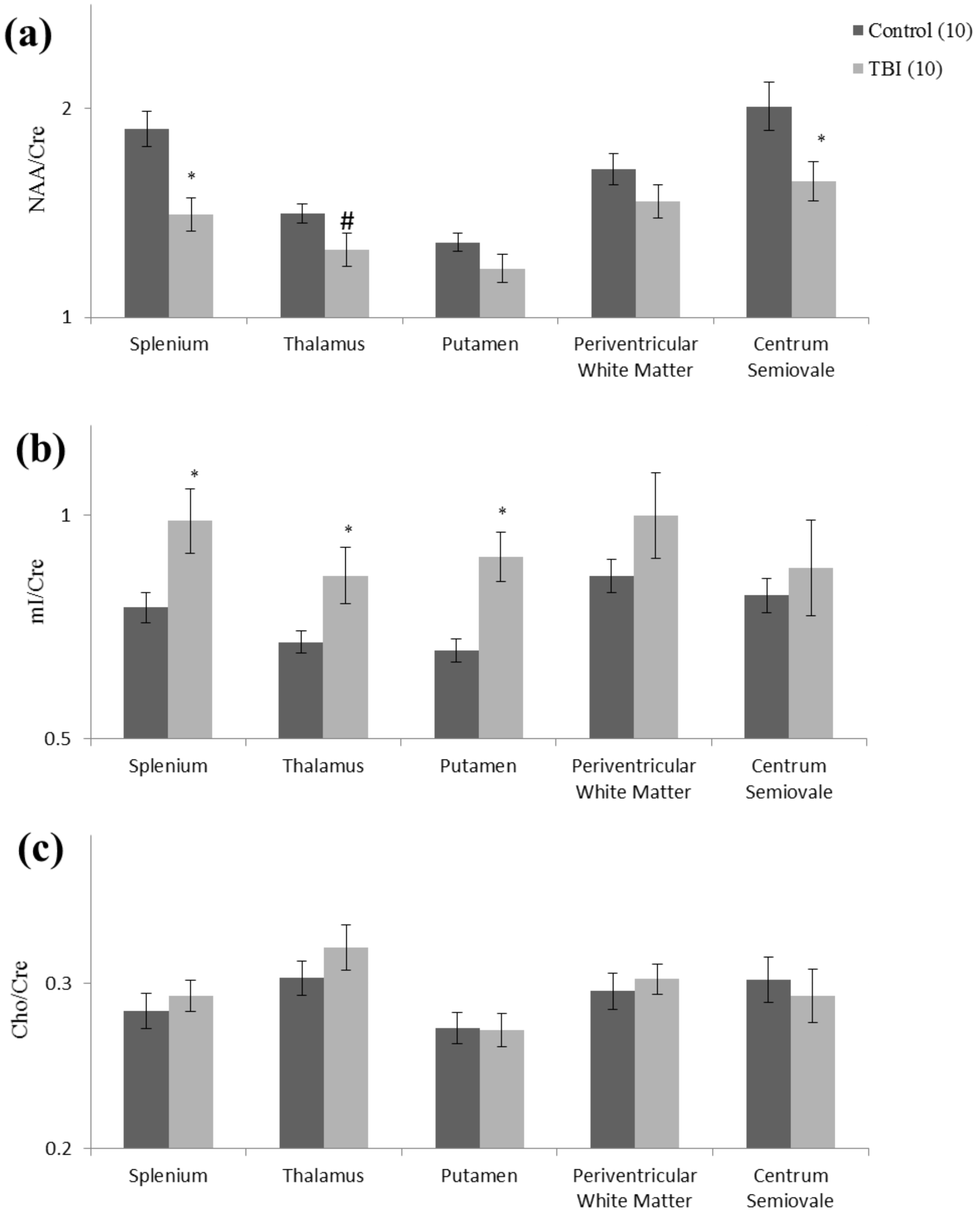


Figure 4.5. Group comparison of TBI patients to healthy subjects using short TE DE-MRSI measurements of (a) NAA/Cre (b) mI/Cre and (c) Cho/Cre measurements from multiple neuroanatomic regions. *#p<0.1; *p<0.05*

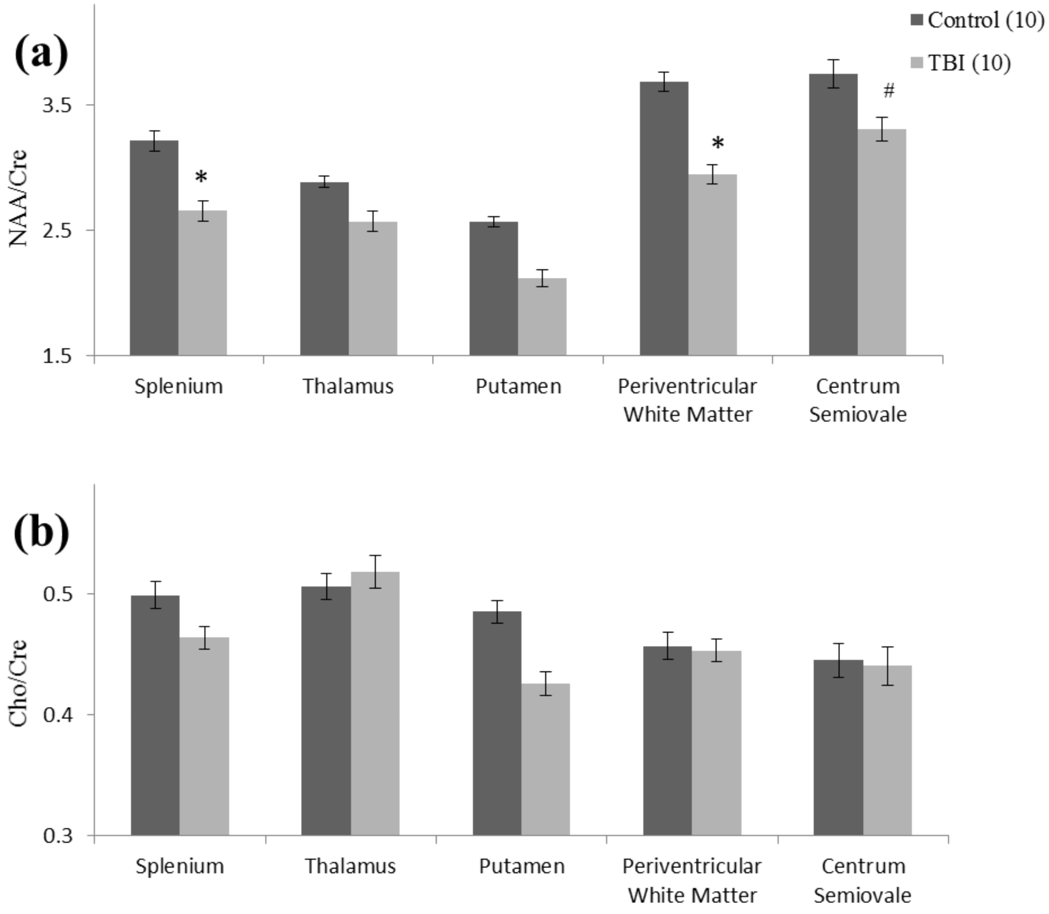


Figure 4.6. Group comparison of TBI patients to healthy subjects using long TE DE-MRSI measurements of (a) NAA/Cre and (b) Cho/Cre measurements from multiple neuroanatomic regions. *#p<0.1; *p<0.05*

4.4 Discussion

A thorough evaluation of the diffuse, underlying metabolic changes that occur in cerebral pathologies will require information from multiple MRSI data sets, each at different echo times (TE) of acquisition. However, with the already prolonged nature of MRSI scans, carrying out multiple spectroscopic acquisitions would be unfeasible in most clinical and research settings. In this study, we presented a novel dual-echo MRSI (DE-MRSI) technique that affords users the capability of acquiring short TE (TE = 30ms) and long TE (TE = 270ms) MRSI data sets in the same duration as a standard clinical SE MRSI method. Previous studies have investigated the use of multi-echo spectroscopic imaging methods. Dreher *et al*., introduced a double echo (TE = 136ms, 272ms, BW = 2100 Hz, sampling duration = 122ms for each echo) multislice technique using Hadmard encoding.(Dreher and Leibfritz, 1994) The same authors later described a 3-echo sequence [(TE = 135ms, 272ms and 408ms), BW = 2100Hz, 256 points] used to measure the intrinsic T2 of different metabolites in rat brain, while achieving a percent standard deviation of 10-15% for repeated measurements.(Dreher and Leidfritz, 1995) Mulkern *et al*., demonstrated the use of CPMG spectroscopic imaging in a group of healthy volunteers and one tumor patient for T2 decay analyses of metabolite resonances.(Mulkern et al., 1996). While the method was shown to be useful in measuring T2 values of the combined choline and creatine signal, it was not successful in resolving the individual resonances of these metabolites for separate quantification. Kiefer *et al*., introduced a time domain parametric spectral analysis method for enhancing spectral resolution in multi-echo MRSI data (6 echoes; TE = 130ms; BW = 4.64kHz; sampling duration = 110ms) acquired in the presence of an inhomogeneous magnetic field.(Kiefer et al., 1998) The measurements from the different echoes were used to determine the optimal TE of acquisition for different resonances in the presence of varying levels of field inhomogeneity. Skinner et al. used varying echo spacing within the same acquisition (48 echoes, BW = ±32kHz) for T2 measurement of myelin water and intracellular/extracellular water in both phantom and normal brain.(Skinner et al., 2007) The results showed that increased inter-echo spacing for the final 15 echoes was optimal for measuring T2 relaxation. None of the above-mentioned methods however, and to the best of our knowledge, no prior multi-echo MRSI study has demonstrated the sensitivity to detect group-wise *in vivo* metabolic changes induced by pathology or perturbations to cellular metabolic function. The lack of sensitivity inherent to most multi echo spectroscopic imaging methods is owing to the reduced SNR and diminished spectral resolution that accompanies increased sampling BW and shortened acquisition durations necessary for acquiring multiple echos in the echo train. While such schemes might be useful in measuring *in vivo* T2 values of metabolites with long TE measurements, the detection of subtle changes in *in vivo* metabolite levels that arise from pathologies like TBI will ultimately be a challenge. The approach to multi-echo MRSI employed herein however, takes advantage of the increased SNR available in short TE acquisitions by (i) including a short TE data set in the acquired train of echoes, and (ii) sampling this short TE data set with a relatively higher BW while maintaining standard acquisition conditions in the long TE acquisition. More so, the inherently broad linewidths of short T2 metabolites obtained at short TE will permit reliable characterization and quantification of the signal in spite of the reduced spectral resolution.

We compared the measurements from the DE-MRSI sequence to measurements from the standard sequence to see how the changes in the acquisition scheme affect metabolite detection. Repeatability studies were conducted by carrying out multiple runs of the technique in succession to determine the reproducibility of the acquired measurements. Finally, we applied the DE-MRSI sequence to a human study of TBI in order to assess its sensitivity in detecting group-wise pathology-induced metabolic changes.

Sequence Comparison and Repeatability Analysis

Our results show a strong linear correlation between measurements obtained using the standard SE MRSI method and corresponding measurements using the DE-MRSI method in both short and long TE data sets. The short TE measurements were comparable between both sequences. With the long TE acquisitions however, there was a decrease in the SNR of measurements obtained with the DE-MRSI sequence when compared to the standard sequence. This signal loss is however expected as the DE-MRSI method incorporates an additional RF pulse and slice selective gradient needed to refocus the signal for the long TE acquisition. Hence the causes of signal reduction could be any combination of RF non-uniformity, gradient non-uniformity or gradient eddy currents.(Simmons et al., 1994) Nonetheless, calibration analyses such as those reported in **Table 4.2** can be used as a correction curve to estimate the value of the unperturbed signal if need be.

The repeatiblity analysis shows that the DE-MRSI technique is able to generate consistently reproducible measurements in both short and long TE acquisitions. Most measurements had %StdDev $\leq$ 5% in phantom as well as human brain. The %StdDev was observed to increase marginally with long TE measurements of Cho/Cre (8.27% in human brain and 10.74% in ^{1}H-MRS phantom). This is however expected as these measurements have intrinsically low SNR which ultimately hampers consitency in the spectral fitting procedure.

TBI Evaluation

The observation herein of widespread increases in mI/Cre levels for the TBI group agrees well with previous TBI studies of mI.(Ashwal et al., 2004) In the progression of TBI, mI is believed to be a marker of glial cell activity.(Garnett et al., 2000a; Garnett et al., 2000b) The alteration of cerebral mI levels has also been implicated in other cerebral pathologies.(Miller et al., 1993; Shimon et al., 1998) Furthermore, both long and short TE DE-MRSI measurements revealed a widespread depression of NAA/Cre levels in the TBI group when compared to healthy subjects. NAA which is known to be a marker of neuronal integrity and viability,(Demougeot et al., 2001) has been shown to be reduced in the occurrence of TBI.(Garnett et al., 2000a; Garnett et al., 2000b) Changes in NAA levels have also been associated with other cerebral pathologies.(Bizzi et al., 2001; Demougeot et al., 2003) While increases in Cho/Cre have been previously observed in studies of severe TBI,(Garnett et al., 2000a; Garnett et al., 2000b) no changes in Cho/Cre were observed herein. This lack of abnormality detected with Cho/Cre measurements could be owing to a number of factors including the limited number of subjects examined and variability in the time of examination relative to the time of injury for each patient. Furthermore, the neuroanatomical patterns of Cho/Cre alterations in the TBI group when compared to healthy subjects were not necessarily consistent between the short TE and long TE data. This is likely due to factors unaccounted for such as the varying relaxation properties of Cho between subjects. Indeed previous studies have shown that cellular processes such as edema (which is prominent in severe TBI) can affect the relaxation times and apparent concentrations of *in vivo* metabolites.(Kamada et al., 1994) Changes in relaxation times of neurometabolites have been observed in other cerebral pathologies such as bipolar disorder and schizophrenia.(Ongur et al., 2010)

Limitations and Conclusion

A current limitation of this study is the limited number of subjects used in the TBI evaluation. Indeed a larger subject population could help to suppress noise arising from inter-subject variability and provide greater statistical power in detecting group-wise changes. Hence it is important that the results of this study are applied with caution particularly in clinical settings. Nonetheless, with the analyses presented herein we have been able to demonstrate the potential of the DE-MRSI technique as a valuable tool for *in vivo* spectroscopic evaluation. DE-MRSI offers a sensitive, time-efficient and thorough approach to *in vivo* metabolic imaging.

Chapter 5: Predicting Injury Status and Symptomatic Outcome with Early Neurometabolic Patterns in Mild Traumatic Brain Injury

5.1 Introduction

Up to 5 million Americans are currently living with TBI related disabilities. More so about 75% of TBI patients are deemed mild, with most abnormalities manifesting as occult to CT or structural MR imaging. Accurate diagnosis and evaluation of mTBI early after injury is crucial towards prescribing the appropriate interventions and making available the proper care that would aid recovery. Unfortunately however, with the lack of robust neuroimaging and neuropsychological (NP) markers of mTBI available with conventional MRI, CT and neuropsychological tests, clinicians are forced to rely on less objective self-reported symptoms and concussion tests to diagnose mTBI (Brenner et al., 2009). Consequently, many mTBI occurrences have gone undiagnosed, with patients not afforded the needed attention and being ill-advised on when to resume military duty or athletic activity. Hence there is a need for a vehicle that can aid unbiased discrimination between patients who have developed early sequelae which are indicative of injury and individuals who have no deviations from a neurologically healthy state. In addition to detecting acute abnormalities with imaging or neuropsychological (NP) markers of mTBI, a current challenge of researchers and clinicians alike is being able to predict the symptomatic outcome of patients at the later stages of injury. The awareness of the of the likelihood of symptom persistence in the later stages of TBI is of immense value as it can allow patients and caregivers to plan towards recovery and resumption of regular activity, particularly in the case of athletes and combat personnel. Conceivably, if an early awareness of the presence of injury-related sequelae and the likelihood of symptom persistence at the later stages of mTBI is to be realized, there is a need for a modality that can probe physiological imbalances at the cellular level right from the onset of the pathology. Magnetic resonance spectroscopy (MRS) has been shown to be capable of unveiling perturbations in cellular metabolic activity that are induced by mTBI. Our goal in this study is to design and evaluate a methodology for patient diagnosis and symptomatic outcome prediction in mTBI using MRS data. We aim to determine whether early metabolic patterns in mTBI can be indicative of initial injury status (the presence or absence of mTBI-related sequelae at the onset of injury) and symptom persistence at the chronic stage (approximately six months after injury). To this end, the support vector machine (SVM) classification method is applied to MRS data in order to (i) distinguish between mTBI patients whose metabolic sequelae are indicative of injury and individuals who neurologically speaking conform to a healthy state and (ii) distinguish between mTBI patients with and without symptom persistence at the later stages of injury.

Classification methods or classifiers are machine learning algorithms that assign a label or group structure to previously ‘unseen’ input data (test or validation data), having developed a discriminant model by learning pertinent features and patterns from data with known classification (training data). These methods have been employed in biomedical research to discriminate between different forms of a known pathology (e.g. classifying human tumors) and to design decision support systems (DSS) for medical intervention.(Dudoit et al., 2002; Tate et al., 2006; Vicente et al., 2013) With the support vector machines (SVM) classifier, the discriminant model is generated by computing two parallel hyperplanes in a space defined by the features of the data set, for which the distances between the two hyperplanes maximally separates the two classes in the training data set. In cases where the data is not linearly separable by hyperplanes, it is possible to apply a kernel function or "kernel trick" which maps the data into a higher dimensional space where linear separation can be more achievable. A more detailed description of the SVM methodology can be found in the paper by Georgiadis *et al*.(Georgiadis et al., 2011)

A few brain and central nervous system (CNS) injury studies have applied other classification methods to MRS data for the purpose of predicting clinical outcome. Holshouser *et al*. used linear discriminant analysis (LDA) to compare the efficiency of short TE and long TE data in predicting the outcome of children with acute brain injury.(Holshouser et al., 2000) Their analysis revealed that both short and long TE MRS data predicted outcome with an accuracy of 91% in children over one month. In children less than one month, the long TE method performed better with an accuracy of 91% compared to the short TE method which yielded an accuracy of 79%. Auld *et al*. showed that LDA applied to MRS-acquired features alone is able to predict the outcome of children who have sustained acute central nervous system injury with an accuracy of 81%.(Auld et al., 1995) Both of these studies however included children with non-traumatic injuries such as cardiac arrest, hypoxic-ischemic encephalopathy and near-drowning for which the injury sequelae can vastly differ from TBI. Tollard *et al*. showed that by combining MRS and diffusion tensor imaging (DTI) features, an LDA method was able to discriminate between severe TBI patients with unfavorable outcome, those with favorable outcome and control patients with an accuracy of 97%.(Tollard et al., 2009) To the best of our knowledge however, this is the first study investigating patient diagnosis and outcome prediction in mTBI by applying the SVM algorithm to MRS data. Indeed Kernel-based SVM methods offer the best approach to classification in clinical studies as the use of LDA may be problematic when handling small, unbalanced or high-dimensionality data sets.(Lukas et al., 2004; Luts et al., 2007) The SVM algorithm has been successfully combined with MRS data in prior studies of brain tumor classification.(Devos et al., 2004; Devos et al., 2005; Georgiadis et al., 2011; Kelm et al., 2007; Lukas et al., 2004; Luts et al., 2007; Menze et al., 2006).

Three studies to evaluate the use of the SVM algorithm applied to MRS features for mTBI evaluation are described herein (i) the discrimination between mTBI patients and healthy subjects followed by a 10-fold cross validation procedure (ii) the discrimination between patients with- and without symptom persistence at the chronic stage of injury using baseline definitions of post concussive syndrome (PCS). This analysis is also followed by a 10-fold cross validation procedure (iii) a receiver operatic characteristic (ROC) curve analysis involving an expansion of ‘study-(ii)’ above in which different levels of symptomatology are employed as the various discrimination threshold settings.

5.2 Methods

Patient Selection

All patients were recruited from the Adams Cowley Shock Trauma Center at the University of Maryland Medical Center as part of an ongoing MagNet Study (Magnetic Resonance Imaging of Neuro Trauma Study). The study was approved by the IRB of the University of Maryland. Sixty-four mTBI patients [Glasgow Coma Scale (GCS) 13-15] were examined at the acute/ early sub-acute (ESA) stage (within 10 days of injury). Of these 64 patients, 41 were also evaluated chronically [~6 months post injury (PI)] for symptom persistence. Patients were included in this study if they were 18 years of age or older with mechanism of injury indicative of closed head trauma, positive head CT or altered mental status and/or loss of consciousness (LOC). Patients were excluded if they had a history of neurological or psychiatric illnesses, cerebrovascular accidents, brain neoplasms or seizures. It is worth indicating that the mTBI cohort examined herein included complicated mTBI patients (patients with positive acute/ESA CT or MRI). MRS data from 32 healthy, neurologically intact subjects was used for the healthy control group. **Table 5.1** summarizes the demographic information of the mTBI patients and control subjects examined. **Table 5.2** summarizes the demographic information of mTBI patients who were evaluated at the chronic stage of injury for symptom persistence.

MR Examination

MR exams were performed on a Siemens Tim-Trio 3T MRI (Siemens Medical Solutions, Malvern, PA) scanner using a 12-channel receive only head coil. A high resolution T1-weighted-MPRAGE (TE = 3.44 ms, TR = 2250ms, TI = 900ms, flip angle = 9o, resolution = 256 x 256 x256, FOV = 22 cm, sl. Thick. = 1.5mm) was acquired for anatomic reference. A 3D phase-encoded point-resolved spectroscopy (3D-PRESS) magnetic resonance spectroscopy imaging (MRSI) sequence was used to obtain spectroscopic data at a TE/TR of 135ms/1300ms over a FOV of 160 x 160 x 106mm3 and a volume of interest (VOI) covering 106 x 106 x 48mm3. The acquired resolution and interpolated resolution were 12x12x8 and 16x16x8 respectively with a total acquisition time of 7 minutes and 40 seconds. The VOI was oriented along the AC-PC line and centered about the corpus callosum. Saturation bands were used to effectively suppress any chemical shift artifacts arising from lipids outside the volume of interest.

	Control	**mTBI (Early Sub-acute)**	
N	32	64	P-value vs control
Age	39.13 ± 17.76	41.64 ± 16.92	0.501
Female (N)	14	11	NA
GCS	NA	14.78± 0.54	NA
Education (Years)	15.16 ± 2.03	13.71 ± 2.56	.007
Days Post Injury	NA	6.17 ± 3.26	NA
Positive CT (N)	NA	19 (29.69%)	NA
Positive MRI (N)	NA	18 (28.13%)	NA

Table 5.1. Demographic data summary of control subjects and mTBI patients examined at the early-subacute stage. Days post injury (DPI) indicates the mean number of days after injury with which patients were examined at each stage of mTBI

	PCS+	PCS-	p-value
N	19	22	
Age	49.31 ± 16.41	37.27 ± 14.96	0.023
Female (N)	7	2	
GCS	14.52 ± 0.75	14.96 ± 0.21	0.017
Education (Years)	13.53 ± 1.90	14.50 ± 2.92	0.232
DPI (<10 days)	7.05 ± 3.47	6.32 ± 3.04	0.485
DPI (~6months)	200.42 ± 33.22	197.23 ± 27.83	0.7457

Table 5.2. Demographic data summary of mTBI patients examined at the chronic stage for symptom persistence. Days post injury (DPI) indicates the mean number of days after injury with which patients were examined at each stage of mTBI

Evaluation for Persistent Post Concussive Symptoms

Patient evaluation for the persistence of post concussive symptoms (PCS) at the chronic stages of mTBI was carried out using the Rivermead post-concussion symptoms questionnaire (RPQ).(King et al., 1995) Per baseline definitions of PCS,(Boake et al., 2005) patients were deemed PCS positive (PCS+) if they experienced any of 3 or more of the major PCS symptoms (headaches, dizziness, sleep abnormalities, trouble concentrating, fatigue, memory problems and irritability). Of the 41 patients examined, 19 were diagnosed as PCS positive (PCS+) and 22 as PCS negative (PCS-)

at the chronic stage of injury

MRS Processing

MRS spectra were quantified offline using LCModel.(Provencher, 2001) LCModel is a frequency domain MRS processing software that compares the measured *in vivo* spectra to a linear combination of spectra from an in vitro basis set. The software uses a model that includes line shape and baseline functions, zero-order and first-order phase correction parameters and overall referencing of chemical shifts to fit the in vitro basis set to the measured *in vivo* spectra. The Cramer-Rao (CR) bounds generated by the software represent a lower limit of the statistical error of the fitted parameters. In the current study, metabolic measurements with CR bounds greater than 15% were excluded from any further analysis.

Support Vector Machine Algorithm Development and Cross Validation

The MRS features utilized in the SVM model were N-acetyl aspartate-to-creatine ratio (NAA/Cre) and Choline-to-creatine ratio (Cho/Cre) values measured in the thalamus and centrum semiovale (CSV). **Figure 5.1** shows the anatomical locations for the regions of interest. Values from each lateral side of the regions were averaged to produce one single value for analysis. Previous studies have shown that measurements from these regions are sensitive to mTBI-induced changes.(George *et al.*, 2014)(Inglese et al., 2005; Kirov et al., 2007)

The SVM algorithm was developed using MATLAB R2013b. With the selected features, a SVM method with a radial basis function (RBF) kernel was used to train a model to predict the class of each subject. An RBF is a function whose value depends solely on the distance to the origin.(Scholkopf et al., 1997) **Figure 5.2** illustrates the procedure for classification using the SVM-RBF algorithm. A 10-fold cross validation method was used to evaluate the accuracy of the model in classifying 'unseen' data samples.(Fushiki, 2011)

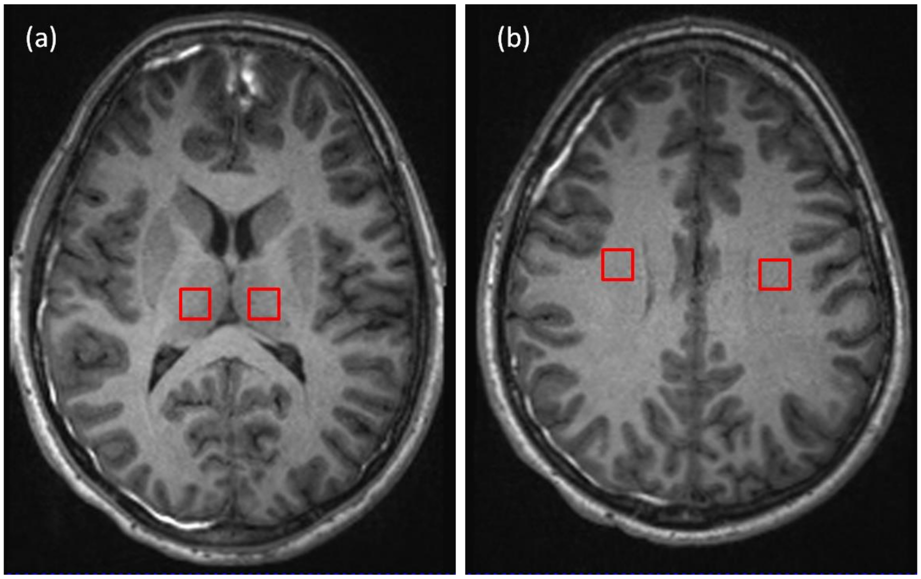


Figure 5.1. Anatomical location of the ROIs [(a) thalamus and (b) centrum semiovale] used to obtain spectroscopic information. Measurements from each lateral side of the ROI were averaged to generate a single value for each region

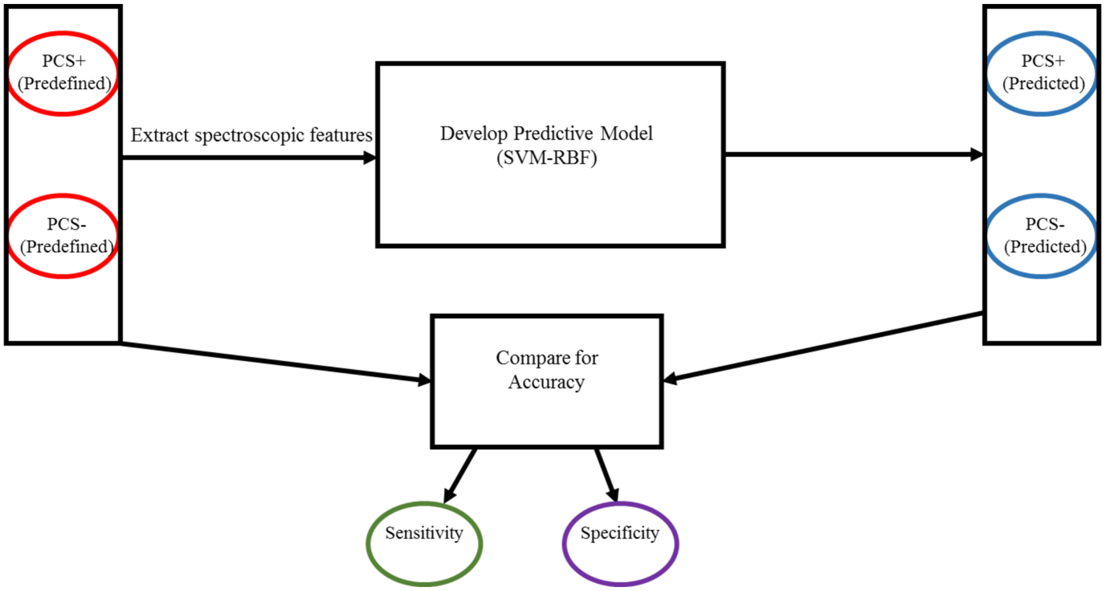


Figure 5.2. Schematic of Classification procedure using SVM-RBF algorithm

Receiver Operating Characteristic Curve Analysis

An ROC curve is a graph used to show the discrimination accuracy of a binary classifier.(Zweig and Campbell, 1993) The curve is generated by plotting the sensitivity or true positive rate (TPR) of the classifier against "1-specificity" or the false positive rate (FPR) at various discrimination threshold settings. An ROC curve can be useful in determining the optimal model from a number of choices that are being considered for use as classifiers. The area under the curve (AUC) is indicative of the accuracy of the predictive model. Hence an area of 1 would represent a perfect test and an area of 0.5 represents a model with completely random predictions.

Herein, ROC analysis was used to evaluate the performance of the SVM classifier in predicting the symptomatic outcome of mTBI patients. The discrimination thresholds employed were based on the number of self-reported symptoms used for PCS evaluation (headaches, dizziness, sleep abnormalities, trouble concentrating, fatigue, memory problems and irritability). Hence each threshold was indicative of the occurrence of *n* or more symptoms (where n = 1, 2…6). Computations for the ROC method were developed using Matlab2013b.

5.3 Results

Discrimination between Acute mTBI Patients and Healthy Subjects

When spectroscopic features from the thalamus and CSV were jointly used in the classification algorithm, acute mTBI patients were correctly differentiated from healthy subjects with an accuracy of 80.21% (sensitivity = 78.13%, specificity = 84.38%). Using features from the thalamus alone, an accuracy of 67.71% (sensitivity = 62.50%, specificity = 78.13%). An accuracy of 71.88% (sensitivity = 79.69%, specificity = 56.25%) was realized when spectroscopic features from the CSV alone were employed. These results are summarized in **Table 5.3**. **Figure 5.3** illustrates the discrimination between acute mTBI patients and healthy subjects when features from the thalamus and CSV were separately used in the SVM model.

In evaluating the performance of the predictive model in classifying 'unseen' data samples, a 10-fold cross validation analysis yielded an accuracy of 69.80% when features from the thalamus and CSV were concurrently used for discrimination. Features from the thalamus and CSV yielded cross validation accuracies of 67.71% and 61.46% respectively when used separately. These results are summarized in **Table 5.3**.

	Sensitivity (%)	Specificity (%)	Accuracy (%)	10 fold CV (%)
Thalamus	62.50	78.13	67.71	67.71
CSV	79.69	56.25	71.88	61.46
Thalamus + CSV	78.13	84.38	80.21	69.80

Table 5.3. Accuracy of predictive model in discriminating between healthy subjects and mTBI patients. 10 fold cross validation (CV) measures performance of model in classifying 'unseen' data samples

Symptomatic Outcome Prediction with Baseline Definitions of Post Concussive Syndrome

In using spectroscopic features from both the thalamus and the CSV, the predictive model achieved an overall accuracy of 88.81% (sensitivity = 94.74%, specificity = 81.82%) in classifying mTBI patients according to symptomatic outcome. An overall accuracy of 80.49% (sensitivity = 78.94%, specificity = 81.82%) was achieved when features from the thalamus alone were used. When features from the CSV alone were used, an overall accuracy of 73.17% (sensitivity = 63.16%, specificity = 81.82%) was achieved. These results are summarized in **Table 5.4**. **Figure 5.4** illustrates the classification of patients according to symptomatic outcome when features from the thalamus alone and CSV alone were used.

A 10-fold cross validation analysis yielded an accuracy of 78.05% when features from both the thalamus and CSV were jointly used. Features from the thalamus and CSV individually yielded cross validation accuracies of 70.73% and 56.1% respectively. These results are summarized in **Table 5.4**.

	Sensitivity (%)	Specificity (%)	Accuracy (%)	10 fold CV (%)
Thalamus	78.95	81.82	80.49	70.73
CSV	68.42	72.73	70.73	56.10
Thalamus + CSV	94.74	81.82	87.80	78.05

Table 5.4. Accuracy of predictive model in discriminating between symptomatic outcome classes of patients. 10 fold cross validation (CV) measures performance of model in classifying ‘unseen’ data samples.

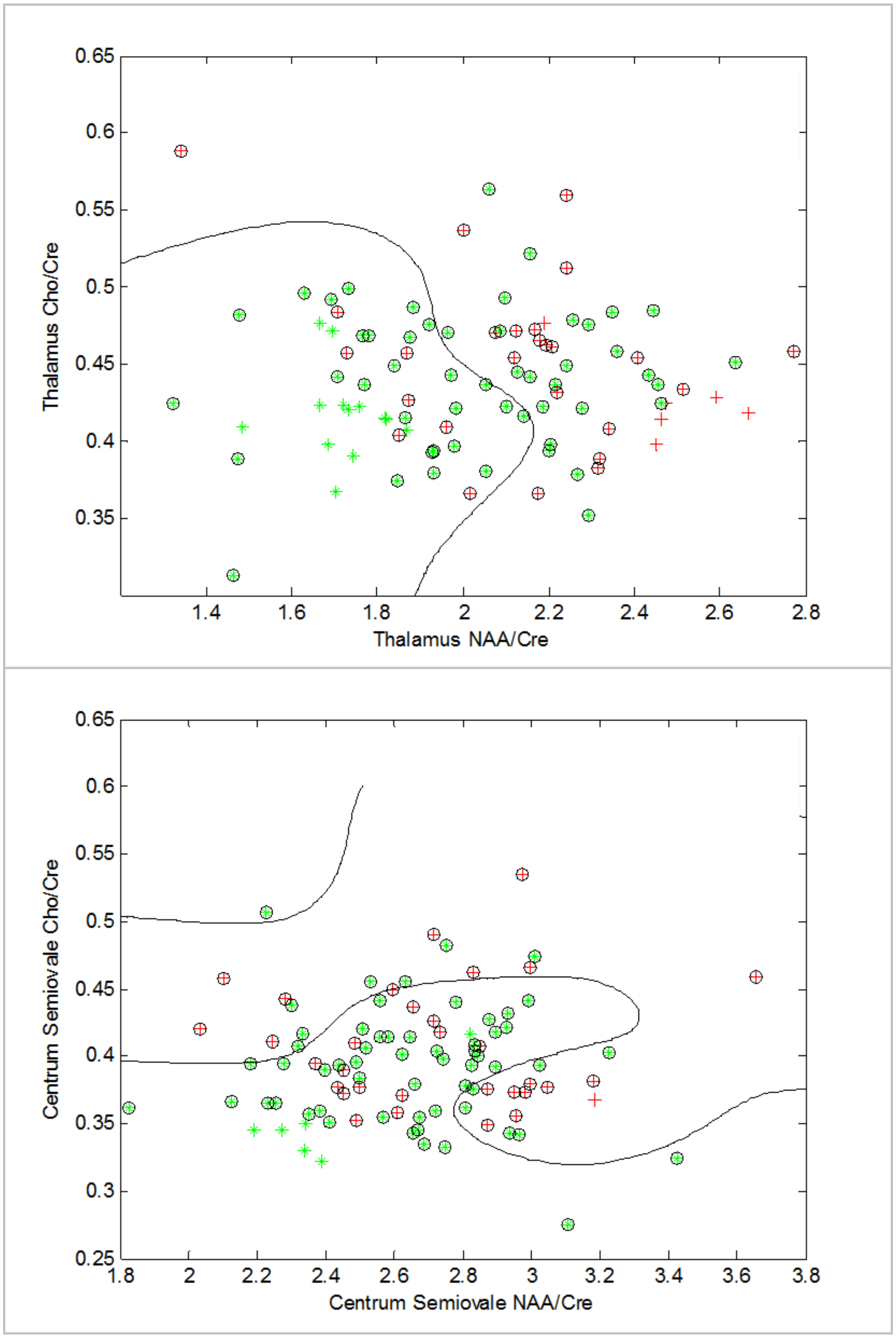

0.65
0.6
0.55
0.5
0.45
0.4
0.35
Thalamus Cho/Cre
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
Thalamus NAA/Cre
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
Centrum Semiovale Cho/Cre
1.8
2
2.2
2.4
2.6
2.8
3
3.2
3.4
3.6
3.8
Centrum Semiovale NAA/Cre

Figure 5.3. (*shown on previous page*) Separation of mTBI patients (*) and healthy controls (+) by predictive SVM model using a radial basis function kernel. Contour lines indicate cross-section of separating hyperplane. Circled markers indicate samples lying on the decision boundry demarcating each class. Shown are the 2-D plots for classification when spectroscopic features from the (a) thalamus and (b) centrum semiovale alone are used in the model.

Receiver Operating Characteristic Curve Analysis

When spectroscopic features from the thalamus and CSV were jointly used for the ROC evaluation, an AUC of 0.9694 was achieved. The thalamus and CSV separately yielded AUCs of 0.8308 and 0.8377 respectively. These results are illustrated in **Figure 5.5**.

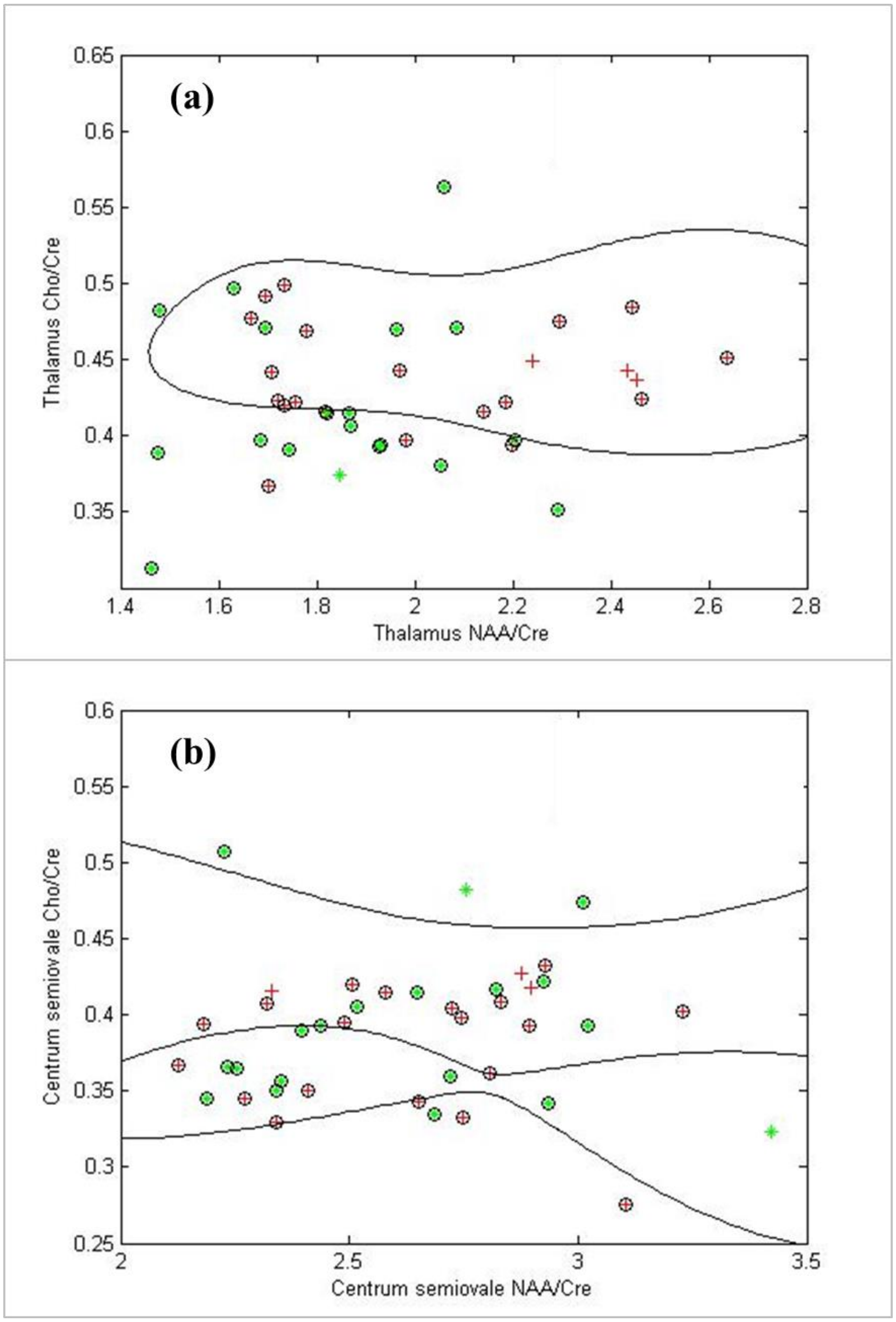


Figure 5.4. Separation of outcome classes [PCS+(*) and PCS-(+)] by predictive SVM model using a radial basis function kernel. Contour lines indicate cross-section of separating hyperplane. Circled markers indicate samples lying on the decision boundry demarcating each class. Shown are the 2-D plots for classification when spectroscopic features from the (a) thalamus and (b) centrum semiovale alone are used in the model.

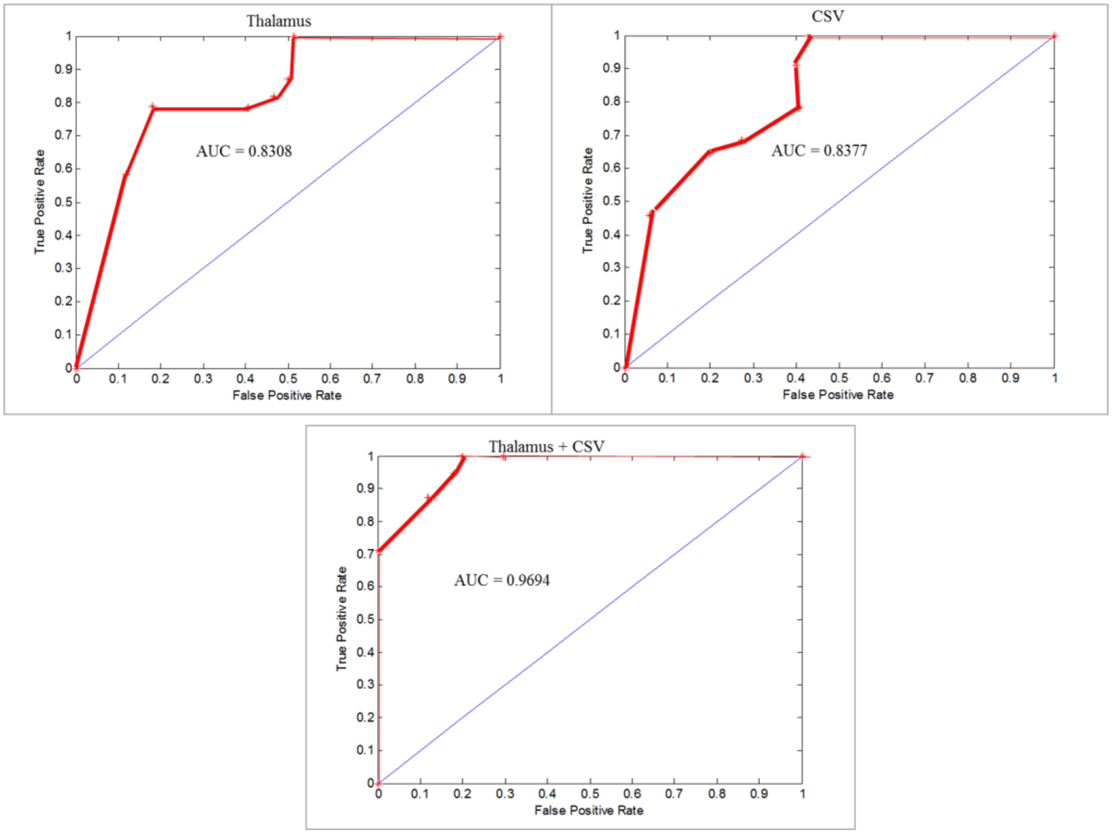


Figure 5.5. Receiver operating characteristic (ROC) curve (True Positive Rate vs. False Positive Rate) analysis for symptomatic outcome prediction in mTBI. Shown are the ROC curves for the thalamus (AUC = 0.8308), CSV (AUC = 0.8377) and Thalamus + CSV (AUC = 0.9694)

5.4 Discussion

The goals of this study were to design and evaluate a vehicle for patient diagnosis and symptomatic outcome prediction in mTBI using early neurometabolic information. The SVM algorithm was applied to acute/ESA MRS features with the aim of (i) discerning between patients whose early metabolic sequelae were indicative of injury and individuals whose metabolic patterns conform to a neurologically healthy state (ii) predicting whether patients would be PCS+ or PCS- at the chronic stage of injury (approximately 6 months PI). A 10-fold cross validation method was used to evaluate the performance of the predictive model in classifying ‘unseen’ data samples.

Furthermore, an ROC analysis was used to evaluate the utility of the SVM model as a classifier in predicting symptomatic outcome. The MRS features employed were NAA/Cre and Cho/Cre measurements in the thalamus and CSV. The rationale for this choice of features was based on previous work from our research lab revealing that these measurements were sensitive to changes resulting from mTBI.(George *et al.*, 2014). Other previous studies have also shown that measurements from these regions are strong markers of mTBI.(Inglese et al., 2005; Kirov et al., 2007) Indeed the choice of biologically relevant features is crucial in a study such as this as it would not only help to avoid overfitting of the model but would also aid interpretability of results. The major findings of this study were that the SVM model predicted injury status and symptomatic outcome with a much greater accuracy when spectroscopic features from the thalamus and CSV were jointly incorporated into the model compared to when features from either region were separately used. The models employing combined features also performed better when evaluated with a cross validation procedure and ROC analysis.

The improved performance realized when features from both regions are jointly incorporated into the SVM model could be elicited by a number of factors. Firstly, because of the diffuse nature of the mTBI pathology within the brain parenchyma, a more thorough characterization of the acute physiological changes that are indicative of injury status and symptomaticity could be realized if measurements from multiple neuroanatomical regions can be used without possibly introducing noisy or irrelevant features. Secondly, because of the interconnectivity of the thalamus and the CSV with the cerebral cortex (the cortex being the impact site of traumatic shearing forces),(Behrens et al., 2003; Berman et al., 2004; Strich, 1961) it could be inferred that the metabolic patterns indicative of mTBI-induced change are similar in both regions and would be much more easily recognized by a pattern recognition scheme and utilized by a predictive model when they are jointly fed into the algorithm. While the model using thalamic features did not perform as accurately as the model utilizing features from both regions, the model using thalamic features consistently predicted injury status and patient outcome with more accuracy than the model using features from the CSV. The thalamus is characterized by a high neuronal density hence changes in the thalamic nuclei would be more indicative of recovery after TBI, particularly in the event of irreversible neuronal loss. More so, the thalamus has been implicated in deficiencies associated with a number of functions used for behavioral evaluation of mTBI including sleep,(Seilhean et al., 1995) memory,(Johnson and Ojemann, 2000) and fatigue.(Niepel et al., 2006) The CSV on the other hand is a white matter region and its exact role in modulating injury sequelae and recovery after mTBI is yet to be understood.

A marginal decline in classification accuracy was observed when predicting initial injury status compared to the accuracy realized in predicting symptomatic outcome, even though a larger number of subjects was employed in the former. Indeed a larger number of subjects should allow the algorithm to more efficiently recognize and learn the pertinent features that are truly indicative of class instance. The reduced classification accuracy observed in the injury status prediction study could be owing to a number of reasons. Firstly, the disparity in the mechanisms and severity of injury commonly observed in mTBI could be a reason for the decline in classifier performance. Some patients suffer injuries that give rise to secondary brain injury, ultimately yielding sequelae which are prevalent throughout the brain parenchyma. Others with more subtle injuries may not manifest signs that are grossly indicative of injury, ultimately allowing their neurometabolic patters to conform to a neurologically healthy state. Hence while it may be easier for a model to predict the eventual outcome of a patient within the mTBI cohort, distinguishing between patients and healthy subjects can be a much harder feat as some patients experience little or no changes in cellular metabolic physiology. Also, the imbalance in the number of subjects included in each class could present a cause for reduced classifier performance. Indeed imbalance in the size of the classes used to train a model can lead to an unfavorable bias in pattern recognition and feature learning ultimately yielding a sub-optimal model fit. In any case however, a larger number of samples contained in any class is always preferred when possible as this will prevent the algorithm from learning the noisy or irrelevant features of the data and avoid over fitting.

Limitations and Conclusion

The chief limitation of this study is the limited number of subjects used in developing and validating the SVM model. A more conclusive classification analysis will require a large number of subjects from different sites. In particular the ability of the classifier to predict injury initial injury status will likely be improved if a more balanced data set is used. Hence there is a need for at least twice the number of data samples on healthy subjects currently utilized. At the very least however, the results of this study has shown the promise of MRS and the SVM classifier in predicting patient outcome in mTBI. Caution should however be exercised in drawing medical conclusions from a study such as this. Owing to the very subtle nature of injury in mTBI and the resulting overlap of features that will occur between classes, classification in mTBI becomes a difficult problem. Unlike moderate or severe TBI, the changes at the tissue or cellular level are less profound hence predicting injury status or patient outcome would be problematic regardless of the selected features or classifiers employed. It is also important to consider that patient outcome cannot be solely dependent on features measured at the early stages of mTBI. Ambiguity in predicting outcome is introduced by seemingly extraneous factors such as diet and exercise that have been shown to influence patient recovery after TBI.(Wu et al., 2013) Future mTBI outcome prediction studies could take this into account for a more thorough evaluation. Improved classification accuracy could also be realized if multiple spectroscopic acquisitions (each at a different echo time of acquisition) are utilized as has been shown by previous studies.(Garcia-Gomez et al., 2008; Holshouser et al., 2000) While MRS has tremendous potential in probing changes at the cellular level, it is still an emerging technique and its utility in mTBI classification studies will only increase as advances in acquisition, post-processing and quantification methods are realized. The SVM algorithm applied to early neurometabolic features offers a promising approach to patient diagnosis and outcome prediction in mTBI.

Chapter 6: Enhancing Spectral Fit Accuracy and Spectral Resolution in Magnetic Resonance Spectroscopy Imaging Datasets

6.1 Introduction

Magnetic Resonance Spectroscopy Imaging (MRSI) is a useful tool for non-invasive metabolic and biochemical evaluation of *in vivo* tissue. Nonetheless, the acquisition of a quantifiable, artifact-free metabolite signal still remains a challenge in most clinical and research settings. Such artifacts are caused by acquisition imperfections such as magnetic field inhomogeneity across the volume of interest (VOI), eddy currents, macroscopic motion, sub-optimal water suppression and lipid and macromolecular contamination. Furthermore, the relatively low abundance of tissue metabolites compared to the much more prevalent tissue-water makes the issue of signal-to-noise-ratio (SNR) one of increased importance for any MRS method, as optimum SNR is needed for detecting minute metabolic changes in tissue. The presence of artifacts coupled with the SNR issues commonly associated with MRS acquisitions inherently induce ambiguity in measurements obtained with MRS techniques. Hence many researchers and clinicians alike have failed to adapt MRS as a "method of choice" for objective diagnosis and detection of tissue abnormalities. The intricacies of artifact contamination and reduced SNR are further compounded in turbo spectroscopic imaging (TSI) and multi-echo MRSI techniques as these methods require acquisition schemes that characteristically lead to a reduction in SNR and spectral resolution, while introducing artifacts into the spectra. While TSI and multi-echo MRSI techniques are useful for decreasing overall scan time and increasing the amount of spectroscopic information obtainable within a single scan, the increased receiver bandwidth needed for rapid FID sampling are adverse to the SNR and spectral resolution achievable. More so, due to T2-imposed SNR limitations at extremely long echo times, these methods require truncated acquisitions of the FID ultimately leading to truncation artifacts in the spectra.

Herein we propose a post-acquisition method that (i) enhances quantification and spectral fitting precision and (ii) seeks to improve spectral resolution in any MRS acquisition. Optimization in (i) and (ii) above are realized by the use of adaptive time domain filter functions which use function parameters that best suit the local conditions from which the spectra is obtained.

In the first study described herein, the glutamate/glutamine (Glx) Cramer-Rao Lower Bound (CRLB) value generated by LCModel,(Provencher, 2001) is minimized using an optimized Lorentz-Gaussian (LG) filter function. The Glx CRLB is chosen as an objective function because of the inherently low SNR of the Glx resonance in ^{1}H-MRS spectrum of the human brain. Hence any measures taken to improve the quantification precision of the Glx signal should presumably have a similar effect on resonances with a larger signal amplitude. We compare the results of applying this optimized method to short echo time dual-echo MRSI (DE-MRSI) (high bandwidth, low spectral resolution) data obtained from a human brain to the same data set processed without the optimization scheme. The short TE DE-MRSI has been described in a previous study from our research lab.(George *et al.*, 2014) Also, the result of this optimization is compared to an identical data set acquired using a standard single echo (SE) acquisition, to show that the linear relationship between both acquisitions is preserved in other metabolites. In the second study, the estimated *in vivo* FWHM is minimized using the optimized LG filter function with the aim of boosting the overall spectral resolution. The FWHM minimization procedure is applied to spectra acquired from a ^{1}H-MRS phantom for which the magnetic field homogeneity has been disrupted. A standard PRESS acquisition was used to obtain data for this analysis.

6.2 Methods

Data Acquisition

All scans were implemented on a Siemens Tim-Trio 3T MRI (Siemens Medical Solutions, Malvern, PA) scanner using a 12-channel receive only head coil. A high resolution T1-weighted-MPRAGE (TE = 3.44 ms, TR = 2250ms, TI = 900ms, flip angle = 9^o, resolution = 256 x 256 x256, FOV = 22 cm, sl. Thick. = 1.5mm) was acquired for anatomic reference. MRSI scan parameters were as follows: TE = 30ms; TR = 1320ms; FOV = 160x160x106; VOI = 106x106x48; acquired resolution = 12x12x8; interpolated resolution = 16x16x8; total acquisition time = 7min 40 secs; vectors size = 512. Data for the first study was acquired from the brain of a neurologically healthy subject. Data for the second study was acquired from a General Electric (GE) braino phantom.

For the first study, the optimized filter method was applied to the short TE DE-MRSI acquisition. This acquisition which is obtained during the first readout of a dual-acquisition scheme is obtained with a relatively high bandwidth (5kHz) while sampling with a vector size of 512 points. Hence spectral resolution and SNR are ultimately compromised. More so, the dual acquisition requires that this readout is abruptly truncated allowing truncation artifacts in the spectra. The results of applying the optimized filter method was compared to data from a standard SE acquisition (bandwidth = 1kHz, vector size = 512 points).

In the second study, the optimized filter method was applied to data from a GE ^{1}H-MRS braino phantom acquired using a standard press SE acquisition, with which the magnetic field homogeneity was disrupted. The FWHM of the water signal within the VOI was measured at 45 Hz and the T2* was measured at 10ms. The results of applying the optimized method to data acquired under an inhomogeneous magnetic field is compared to the same dataset processed without the method in order to evaluate the improvement in spectral resolution realized.

MRS Processing

MRS spectra were fitted and quantified using LCModel. LCModel is a frequency domain MRS processing software that compares the measured *in vivo* spectra to a linear combination of spectra from an *in vitro* basis set. The software uses a model that includes line shape and baseline functions, zero-order and first-order phase correction parameters and overall referencing of chemical shifts to fit information obtained from the *in vitro* basis set to the measured *in vivo* spectra. The Cramer-Rao lower bounds (CRLB) generated by the software for each metabolic measurement represent a lower limit of the statistical error of the fit.

Optimization Algorithm

Voxel-wise quantification precision is enhanced by applying a Lorentz-Gaussian filter function (**Eq 1**) with voxel-specific parameters that objectively minimize the Glx-CRLB or the *in vivo* FWHM values for the voxel. The filter function is defined by

$$F_{filter}(t) = e^{+t\nu_L} e^{-t^2 \nu_G^2} \qquad \textbf{Eqn 6.1}$$

F*filter* is the value of the filter function to be multiplied by the time domain signal at the time (*t*) of the FID; $\boldsymbol{\nu}_L$(Hz) and $\boldsymbol{\nu}_G$(Hz) are the Lorentzian line narrowing and Gaussian line broadening parameters respectively. The optimum $\boldsymbol{\nu}_L$ and $\boldsymbol{\nu}_G$ parameters for each voxel were determined by first repeatedly applying the filter with every combination of $\boldsymbol{\nu}_L$ and $\boldsymbol{\nu}_G$ within the range of 0.01ppm – 0.1 ppm (for both $\boldsymbol{\nu}_L$ and $\boldsymbol{\nu}_G$) in a step-wise fashion, at intervals of 0.01ppm. This range was chosen as glutamate and glutamine can conveniently be resolved with linewidths within this range. More extreme values for these parameters could further compromise the SNR or the spectral resolution of the entire spectra. The combination of $\boldsymbol{\nu}_L$ and $\boldsymbol{\nu}_G$ yielding the lowest Glx-CRLB or *in vivo* FWHM were determined to be the optimum values for any particular voxel.

6.3 Results

Optimization of Spectral Fitting Accuracy

The application of the optimized filter method consistently led to more robust quantification of all metabolites in multiple voxels in the data set. **Figure 6.1** shows the result of applying an optimized LG filter function to a voxel in short TE DE-MRSI data. More importantly, the quantification of the Glx signal in the voxel shown went from unreliable (CRLB>20%) to reliable (CRLB<20%) deeming it suitable for spectroscopic analysis. Perhaps the most visually palpable effect of the application of this method to the data set analyzed was the suppression of the truncation artifacts which is characteristic of most multi echo data sets. In the entire data set, voxel-wise optimization of the CRLB led to an increase in the total number of voxels with reliable Glx quantification (from 41% of the voxels in the data set before application of the optimized LG filters to 72% of the voxels after filters were applied). Even though the Glx-CRLB was used to determine the optimum filter parameters, application of the optimization method either led to a decrease in the CRLB for all

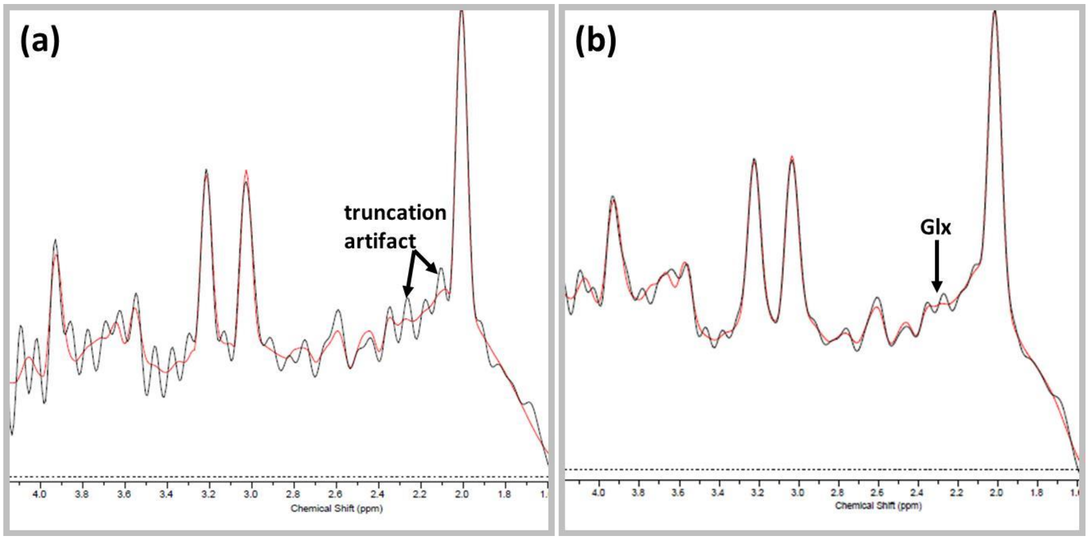


Figure 6.1 ^{1}H-MRSI spectra from a voxel in human brain. The spectra was acquired from short TE DE-MRSI data processed with- (b) and without (a) the optimized LG filtering method. Processing with the optimized method allowed truncation artifacts to be suppressed ultimately leading to more precise quantitation of metabolite signals. In this example, the Glx CRLB was reduced from 35% to 16% after the optimized filter function was applied.

metabolites, or left it unchanged. In addition, the linear relationship expected between data from a standard SE acquisition and data processed using the optimized filter approach is still preserved as shown for NAA/Cre in **Figure 6.2**.

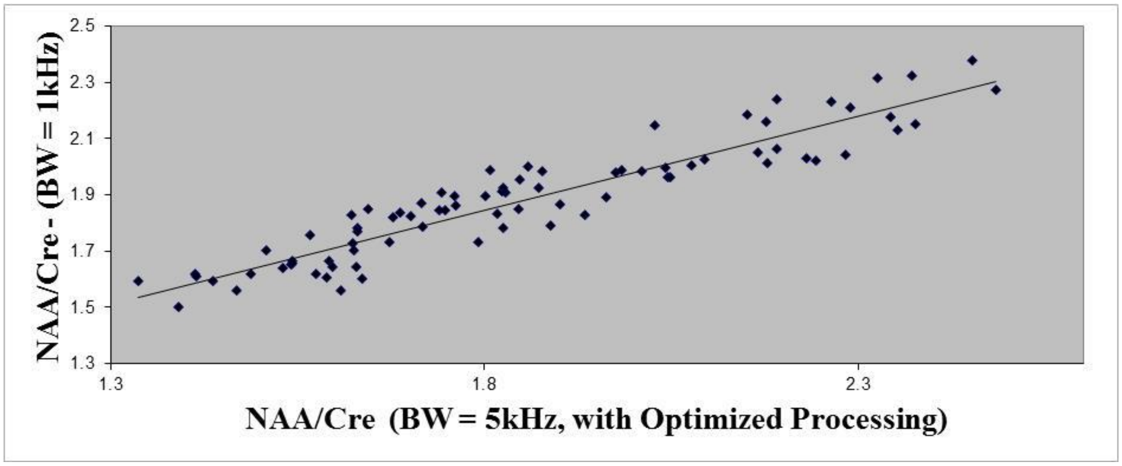


Figure 6.2. Linear correlation of NAA/Cre values in different voxels of a human brain measured by a standard-MRSI method (BW =1kHz) with NAA/Cre values in the corresponding voxels from optimized low spectral resolution (BW=5kHz) MRSI measurements. r = 0.935, p<<<0.001

Optimization of Spectral Resolution

The use of the filter method allowed a marginal reduction in the *in vivo* FWHM when optimized for spectral resolution. **Figure 6.3** illustrates the effect of application of the optimized method when applied to data acquired under poor magnetic field homogeneity conditions. The application of the method when optimized for spectral resolution however consistently led to a reduction in the SNR measured in the voxel. The voxel-wise optimization of the spectral resolution led to a reduction in FWHM for 69% of the voxels in the data set with an average reduction rate of 11% in each voxel when compared to the data set processed without the optimized method.

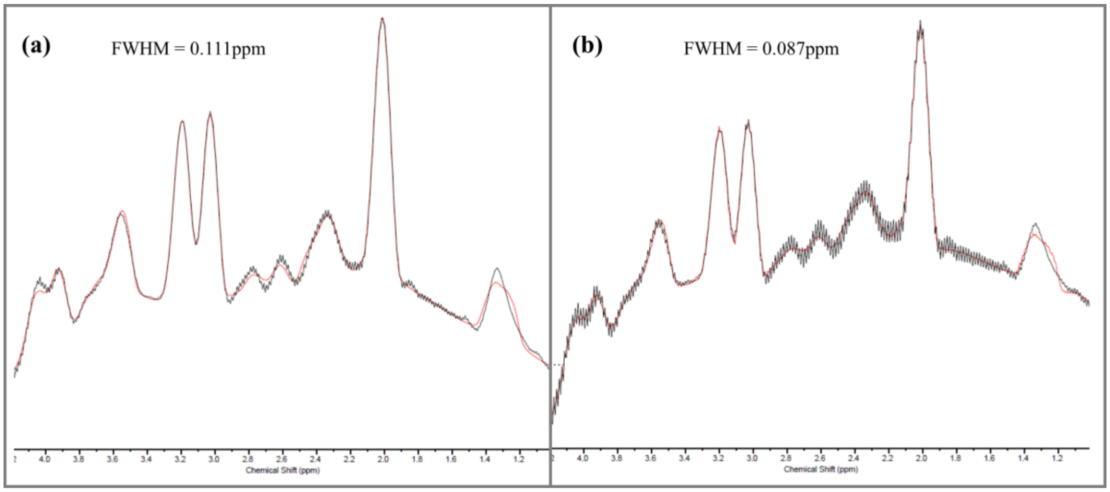


Figure 6.3 ^{1}H-MRSI spectra from a voxel in human brain. The spectra was acquired from standard single echo MRSI data processed with- (b) and without (a) the optimized LG filtering method for improving spectral resolution. Processing with the optimized method yielded measurable reductions in the in vivo FWHM. In this voxel, the in vivo FWHM was reduced from 0.111ppm to 0.087ppm after the optimized filter function was applied. The application of the optimized method for improving spectral resolution however led to a reduction in the measured SNR of the voxel.

6.4 Discussion

Herein adaptive time-domain filtering using the LG filter function was successfully used to improve spectral fitting accuracy and spectral resolution in a MRSI data set. The concept of applying time-domain filter functions to spectroscopic data has been well established and previously employed in other studies.(Ebel et al., 2006; Van Horn et al., 2010; Wang, 1996) To the best of our knowledge however, the current study is the first time the application of adaptive voxel-specific filter functions has been used to enhance spectral fitting accuracy and spectral resolution in MRSI data.

A number of factors could contribute to poor quantification precision in MRSI data. Perhaps the most prominent amongst these are the low SNR of the metabolite signal, poor spectral resolution and spectral artifacts induced by premature truncation of the

FID acquisition. Filter functions can in principle be used to suppress the adverse effects introduced by these factors, but a knowledge of the function parameters that can effectively address the specific combination and severity of the conditions is necessary. More so, in an MRSI data set, the nature and extent of the complications introduced by acquisition deficiencies will vary from voxel to voxel, hence a "one size fits all" approach to filter application might not be optimum. This necessitates an approach to time-domain filtering that can (1) effectively combat the spectral quantification problems that are caused by the myriad of acquisition complications and (2) address such problems in a matter that best suits the local conditions of each voxel. Conceivably, a true validation of the performance of these filter methods would be the efficiency of the method in improving the accuracy of a spectral fit and/or improving the spectral resolution which can create a means for improved quantification reliability. Hence the adaptive approach to filter application introduced herein seeks to address the issues described above while utilizing spectral-fit accuracy and spectral resolution as a yard stick for success. The dual line-narrowing and line-broadening property of the LG function (depending on the values of the parameter used in the function) equips it with the capability of addressing various complications within the voxel, towards improving spectral fit accuracy and spectral resolution.

In the first study, the increase in the number of voxels with a reliable quantification of low SNR metabolites realized upon the application of the optimized filter method strengthens the utility of the MRS technique to which it is applied. Herein, Glx signal quantification precision was employed as the objective for the optimization scheme. As glutamate is the major excitatory neurotransmitter in the brain, imbalances in Glx levels have been implicated in a number of cerebral pathologies and excitotoxic conditions including traumatic brain injury (TBI),(Gasparovic et al., 2009) cirrhosis,(Cordoba et al., 2001) and post-stroke depression.(Glodzik-Sobanska et al., 2006) Hence the application of the methods proposed herein can aid a thorough evaluation of such pathologies across multiple regions in the brain parenchyma. More so, our analysis revealed that the application of the optimized filter method did not perturb the values of the preexisting measurements when metabolic ratios where used. Hence the use of the optimized method can allow a valid comparison to results from other data sets processed without the method.

In the second study, the increase in the spectral resolution realized across the entire data set can potentially be useful in resolving closely separated resonances particularly in short TE data sets where the presences of broadly overlapping lipids and macromolecules can further hinder the resolution of smaller resonances. The application of this method for enhancing spectral resolution must however be carried out with caution as the same phenomena responsible for narrowing of the resonance linewidth can also allow reductions in the SNR of the spectra. A narrowing of the resonance linewidth is ultimately brought about by the application of an increasing window function across the FID in the time domain. An increasing function can lead to an amplification of the 'noisy' data points at the end of the FID thereby causing a reduction in SNR. Hence it is necessary to ensure that the boost in spectral resolution will ultimately lead to an increase in quantification precision. This will largely depend on the spectral fitting method utilized.

It is important to note that with this voxel specific filter approach, comparisons across different voxels within the same subject or comparisons between different subjects are best carried out using metabolite ratios, as the function parameters applied to each voxel may vary substantially. The use of metabolic ratios can algebraically cancel out the voxel-specific factors introduced by the filter without undoing the spectral fitting or spectral resolution enhancing effect. If absolute concentration values are to be determined using this method, identical voxel-wise filters will need to be applied to a separate spectroscopic non-water suppressed acquisition so tissue water can serve as a reference metabolite.

Limitations and Conclusions

A current limitation of this study is the computationally expensive approach used to determine the optimum filter parameters for each voxel. Future studies will involve the use of derivative-free global optimization algorithms in determining the appropriate parameters. Herein we have demonstrated the relevance of voxel specific processing of data for accurate quantification of metabolites through adaptive filter methods in large scale spectroscopic data sets. This method could potentially help to improve the reliability and utility of MRSI techniques in clinical and research settings.

Chapter 7: Summary and Future Directions

The goal of this dissertation was to employ and assess magnetic resonance spectroscopy (MRS) as a tool for the evaluation of traumatic brain injury (TBI). To this end, MRS was used to detect group-wise neurometabolic changes and to predict cognitive outcome in TBI patients. Also, by employing sophisticated pattern recognition and model evaluation techniques, we designed and evaluated a system for improved patient diagnosis and outcome prediction in mild TBI (mTBI). Furthermore, novel methods for enhancing MRS acquisition and metabolite quantification in TBI and other neuroimaging studies were developed and evaluated. This chapter provides a summary of this dissertation and future directions for the research presented herein.

7.1 Longitudinal Evaluation of Mild Traumatic Brain Injury

In chapter 3, we carried out a cross sectional evaluation of the neurometabolic changes that occur at different stages of mTBI. Patients were evaluated at the early sub-acute (ESA) stage (within 10 days of injury), the late sub-acute (LSA) stage (within 1 month of injury) and chronic stage [6 months post injury (PI)]. Also, MRS measurements were used to predict the cognitive outcome of mTBI patients at the chronic stage of injury. The ANAM (Automated Neuropsychological Assessment Metrics) was used to evaluate the cognitive performance of patients at the chronic stage of injury. Our results revealed the reduction of Cho/Cre values measured in the thalamus and centrum semiovale (CSV) at the sub-acute stage of injury in mTBI patients when compared neurologically healthy subjects. In addition, Cre values measured in the CSV at the ESA stage positively associated with chronic ANAM scores measuring performance in delayed and immediate code substitution. These results jointly show that metabolic measurements in the thalamus and CSV can potentially serve as diagnostic and prognostic markers of mTBI. The findings of this study will help to advance the understanding of neurometabolic sequelae in mTBI.

7.2 Dual Echo Magnetic Resonance Spectroscopy Imaging

In chapter 4, we introduced dual echo magnetic resonance spectroscopy imaging (DE-MRSI), a novel MRS acquisition method that was capable of acquiring 2 spectroscopic data sets, each at a different echo time (TE = 30ms and TE = 270ms), with the same acquisition time as a standard clinical single echo (SE) technique. The strategic choice of acquisition parameters utilized allowed the implementation of the dual echo scheme in a fashion that prevented the adverse losses in sensitivity and decline in quantification accuracy commonly associated with multi-echo and fast spectroscopic imaging techniques. Furthermore measurements from this technique were compared with identical measurements from the standard SE counterpart to determine how the dual acquisition scheme would affect the acquired measurements. The DE-MRSI technique was also tested on TBI patients and healthy subjects to determine if the measurements were sensitive enough to detect group-wise neuropathologic changes. The results showed that measurements acquired using the DE-MRSI method strongly correlated with the corresponding measurements acquired using a standard SE technique in both phantom and human brain. Furthermore, measurements from the DE-MRSI technique were capable of detecting group-wise changes in various neuroanatomic regions in TBI patients. DE-MRSI potentially presents a thorough and clinically feasible approach to spectroscopic imaging in neuropathologic conditions such as TBI.

7.3 Predicting Injury Status and Symptomatic Outcome with Early Neurometabolic Patterns in Mild Traumatic Brain Injury

In Chapter 5, we designed and evaluated a novel method for predicting initial injury status and symptomatic outcome in mTBI. The support vector machine algorithm was applied to MRS features with the aim of (i) discriminating between patients whose neurometabolic patterns were indicative of injury-related sequelae and individuals whose neurometabolic patterns showed no deviation from a neurologically healthy state and (ii) distinguishing between patients with and without symptom persistence at the chronic stage of injury. The features utilized in the classification algorithm were NAA/Cre and Cho/Cre values measured in both the thalamus and CSV. The radial basis function (RBF) kernel was employed in the SVM algorithm to map the features to a higher dimensional space towards achieving optimal separation of classes. Symptom persistence was evaluated using the Rivermead Post Concussive Symptom Questionaire. Furthermore, the performance of the discriminant model was evaluated using a 10-fold cross validation procedure. In the outcome prediction study, an ROC analysis was used to evaluate the utility of the classifier by considering the accuracy at different levels of symptom persistence; each level of symptom persistence pertains to a certain number of self-reported symptoms. The major finding of this study was that the concurrent use of MRS measurements from the thalamus and CSV in the SVM model allowed improved classification compared to when these measurements were used separately. We also observed a slightly better performance of the SVM algorithm in predicting symptomatic outcome compared to the performance in classifying subjects according to initial injury status. We speculate that this decline in performance is due to the disparity in injury severity and mechanism of injury that exists across the mTBI cohort that would ultimately hinder an unbiased recognition of pertinent neurometabolic patterns by the discriminant model. This reduced classifier performance could also be elicited by the imbalance in the size of data samples in each class used in training the SVM model. The application of the SVM algorithm to MRS-derived neurometabolic features provides a promising approach to predicting injury status and symptomatic outcome in mTBI.

7.4 Voxel-Wise Enhancement of Spectral Fit Accuracy and Spectral Resolution in MRSI Data Sets

In chapter 6, we described a novel approach to enhancing the accuracy of spectral fitting and improving spectra resolution in MRSI data. In particular, the Lorent-Gaussian (LG) time domain filter was optimized on a voxel-wise basis for improved spectral fitting and spectral resolution measured by the glutamate/glutamine (glx) Cramer-Rao lower bounds (CRLB) and the *in vivo* full width at half maximum (FWHM) respectively. The method for optimization of the spectral fitting accuracy was applied to data acquired with the short TE DE-MRSI (an acquisition characterized by low SNR and truncation artifacts). The results showed the optimized method allowed the suppression of truncation artifacts in the spectra and improved metabolite quantification in over 72% of the voxels analyzed. The method for improving the spectral resolution was applied to a short TE single echo data set acquired in the presence of an inhomogeneous magnetic field. The application of the method allowed a boost in the spectral resolution in 69% of the voxels in the data set with an average improvement of 11% in each voxel when compared to the data set processed without the optimized approach. The results however showed that the improved spectral resolution in many cases occurred at the expense of the signal-to-noise ratio (SNR) hence caution must be taken to ensure that the application of this method ultimately aids improved spectral quantification if so desired. Voxel-wise enhancement of spectral fitting accuracy and spectral resolution with optimized time domain filtering methods could potentially help to improve the reliability and utility of MRSI techniques in research and clinical settings.

7.5 Future Directions

In the foreseeable future, we hope to carry out a multimodal evaluation of TBI in which MRS-derived neurometabolic features will be correlated with other neuroimaging modalities that could perhaps enable a more concrete understanding of the TBI sequelae. Such modalities include diffusion tensor imaging (DTI) and diffusion kirtosis imaging (DKI) which measure the diffusion properties of water at the cellular level, ultimately providing a sense of the microstructural properties of brain tisue; functional magnetic resonance imaging (fMRI) which measures neural activity (both during rest and during the execution of cerebral tasks) by monitoring changes in cerebral blood oxygenation; and arterial spin labeling (ASL) which measures blood perfusion. Such studies could also include classification studies combining features from MRS and the above-mentioned modalities.

As an extension to the work presented in chapters 4 and 5, we hope to carry out classification studies utilizing MRS features acquired at different echo times to aid classification accuracy in TBI. Indeed previous work has shown that the combined use of spectroscopic features acquired at different echo times can allow a boost in classification accuracy.(Garcia-Gomez et al., 2008; Holshouser et al., 2000) It is also of great interest to employ other kernel functions, namely the exponential kernel and polynomial kernel functions in the SVM classification algorithm that could perhaps be more suitable for classification in TBI spectroscopic data sets. Also, we plan to incorporate global optimization methods in determining kernel function parameters that will also aid the applicability of the SVM algorithm to TBI spectroscopic data sets. Global optimization methods will also be extremely useful in the work presented in chapter 6 whereby instead of the "brute-force" step wise approach utilized in finding the optimum function parameters, a derivative-free global optimization method could be used in determining the optimum parameters for each voxel. This will greatly help to reduce computation time and allow a more exhaustive and computationally inexpensive search for optimum function parameters.

Appendix

Dual Echo Magnetic Resonance Spectroscopy Imaging Source Code

```
//---------------------------------------------------------------------------
//  Copyright (C) Siemens AG 1998  All Rights Reserved.  Confidential
//---------------------------------------------------------------------------
//
// Project: NUMARIS/4
//
//    File: \n4\pkg\MrServers\MrSpecAcq\csi_se\csi_se.cpp
//
//  Author: AdvOnc
//
//    Date: 12.01.2007
//
//    Lang: C++
//
// Descrip: Dual Echo MRSI (modified from CSI spin echo sequence with outer
volume suppression) *Modifications to original sequence are included in sections
preceded by “Elijah”
//
// EGA Requirement Key: As shown on the following lines:
//
//   Abbrev.   Translation                                   Relevant for
//   -------   -----------                                   ------------
//   EGA-ALL   {:IMPLEMENT:N4_EGA_MRSpec_SW_LokalisierSW::}      all of
the keys below
//   EGA-01    {:IMPLEMENT:000_EGA_BildOri_SW_SequenzROVz::}    GP
          polarity
//   EGA-02    {:IMPLEMENT:000_EGA_BildPos_SW_SequenzSSelVz::}    GS
          polarity
//   EGA-03    {:IMPLEMENT:000_EGA_BildMass_SW_SequenzROPC::}     GP
          amplitude
//   EGA-04    {:IMPLEMENT:000_EGA_BildPos_SW_SequenzSSel::}      GS
          amplitude
//   EGA-05    {:IMPLEMENT:000_EGA_BildPos_SW_NCOFrequenzSSel::}  SRF
frequency
//   EGA-06    {:IMPLEMENT:000_EGA_BildPos_SW_NCOFrequenzRO::}
Readout frequency/phase
//   EGA-07    {:IMPLEMENT:000_EGA_MRSpec_SW_RefbildValid::}
Positioning/Reference Images must be NDIS images
//
// Variant {csi_se}: -Dcsi_se
```

```
//

/*] END: */


// additional EGA keys compared to svs_se: EGA-01, EGA-03, EGA-06



/////////////////////////////////////////////////////////////////////////////
// Include files
/////////////////////////////////////////////////////////////////////////////


//#include  "MrCommon/MrNFramework/MrTrace/MPCUTrace/MPCUtrace.h"
//#include  "MrServers/MrImaging/libSBB/SeqBuildBlock.h"  // for SeqBuildBlock
class functions
#include  "MrServers/MrProtSrv/MrProt/MrProt.h"
#include  "MrServers/MrMeasSrv/SeqIF/libRT/sSYNC.h"
#include  "MrServers/MrMeasSrv/SeqIF/libRT/sREADOUT.h"
#include  "MrServers/MrImaging/libSBB/libSBB.h"
#include  "MrServers/MrImaging/libSBB/SBBRSat.h"
#include  "MrServers/MrPerAns/PerProxies/GCProxy.h"
#include  "MrServers/MrImaging/seq/SystemProperties.h"
//#include  "MrServers/MrHardwareControl/MC4C40/TX/DSP_TXU.h" // sSample




// solve and try handlers are included
#include "MrServers/MrSpecAcq/spectro_ui\spectro_ui.h"
//lTRNeededSpectro declared extern in spectro_ui.h, and defined in spectro_ui.cpp
//lTENeededSpecSupp declared extern in spectro_ui.h, and defined in spectro_ui.cpp

// chanched Rsat-SBB
  #include "MrServers/MrSpecAcq/sbb_ovs/sbb_fixed_ovs.cpp"
  #include "MrServers/MrSpecAcq/sbb_ovs/sbb_ovs.cpp"




// Debug Flag for UT: _DVP_DEBUG_UT


 // #define _OWN_DEBUG_011
 //  #define _DVP_DEBUG_01
 //  #define _OWN_DEBUG_02
 //  #define _DVP_DEBUG_UT
 //  #define _OWN_DEBUG_fixSBB
 //    #define _OWN_DEBUG_timing
 //        #define _OWN_DEBUG_SpecSupp

  //   #define _DVP_DEBUG
  //  #define _DVP_DEBUG_UT
  //   #define _DVP_DEBUG_UT


//static GPAProxy  theGPA;

//////////////////////////////////////////////////////////////////////////////
// Prototypes of local functions
//////////////////////////////////////////////////////////////////////////////

static NLS_STATUS fSEQRunKernel
(
  MrProt        *pMrProt,
  SeqLim        *pSeqLim,
  SeqExpo       *pSeqExpo,
  long          lKernelMode
 );

//static void print_slicepos( char *, sSLICE_POS * );

/////////////////////////////////////////////////////////////////////
// Slice position information (rotation matrix and voxel position)
/////////////////////////////////////////////////////////////////////

 sSLICE_POS                ss_fov;    //no more static to get value in SBB_OVS
 sSLICE_POS                ss_voi;    //no more static to get value in SBB_OVS


////////////////////////////////////////////////////////////////////
// Sync Bits
////////////////////////////////////////////////////////////////////

static sSYNC          ss_osc1 ("ss_osc1");

//////////////////////////////////////////////////////////
// RF-Pulses and NO-Events
//////////////////////////////////////////////////////////
```

```
 sRF_PULSE_EXT ss_rf_exc( "ss_rf_exc" );     //no more static to get value in
SBB_OVS
 sRF_PULSE_EXT ss_rf_pi_sl( "ss_rf_pi_sl" ); //no more static to get value in
SBB_OVS
 sRF_PULSE_EXT ss_rf_pi_ph( "ss_rf_pi_ph" ); //no more static to get value in
SBB_OVS
static sRF_PULSE_EXT ss_rf_dummy( "ss_rf_dummy" );

static sFREQ_PHASE ss_ph_s_exc( "ss_ph_s_exc" );
static sFREQ_PHASE ss_ph_n_exc( "ss_ph_n_exc" );
static sFREQ_PHASE ss_ph_s_pi_sl( "ss_ph_s_pi_sl" );
static sFREQ_PHASE ss_ph_n_pi_sl( "ss_ph_n_pi_sl" );
static sFREQ_PHASE ss_ph_s_pi_ph( "ss_ph_s_pi_ph" );
static sFREQ_PHASE ss_ph_n_pi_ph( "ss_ph_n_pi_ph" );
static const long  MAX_RF_PI_PULSE_DURATION = 5200;      // max pulse
duration of 180 refoc. pulses in us


// water and fat suppression

static sRF_PULSE_GAUSS ss_rf_ws1( "ss_rf_ws1" );
static sRF_PULSE_GAUSS ss_rf_ws2( "ss_rf_ws2" );
static sRF_PULSE_GAUSS ss_rf_ws3( "ss_rf_ws3" );



#define NO_POINTS_ARB   (4096)
static sRF_PULSE_ARB ss_rf_mega1( "ss_rf_mega1" );
static sRF_PULSE_ARB ss_rf_mega2( "ss_rf_mega2" );
static sSample ss_mega_samples[NO_POINTS_ARB];

double add_refoc( long frequ_offset, long duration, double attenuation, float *arr );
double arr2sample_arr( long sz, float *arr, sSample *sample_arr );


static sFREQ_PHASE ss_ph_s_ws( "ss_ph_s_ws" );
static sFREQ_PHASE ss_ph_n_ws( "ss_ph_n_ws" );

////////////////////////////////////////////////////////////////////
// Readout Events
////////////////////////////////////////////////////////////////////
//Elijah
/////
static sREADOUT  ss_adc1("ss_adc1"); // static structure - ADC event 1

static sFREQ_PHASE ss_ph_s_adc1( "ss_ph_s_adc1" );
static sFREQ_PHASE ss_ph_n_adc1( "ss_ph_n_adc1" );

static sREADOUT  ss_adc2("ss_adc2"); // static structure - ADC event 2

static sFREQ_PHASE ss_ph_s_adc2( "ss_ph_s_adc2" );  // need also a different
phase cycling for ADC2
static sFREQ_PHASE ss_ph_n_adc2( "ss_ph_n_adc2" );
/////


///////////////////////////////////////////////////////////////////
//  OVS Building Block
///////////////////////////////////////////////////////////////////



//#define MaxNrVarRSATS 8              // Maximum number of allowed RSats
defined in SBBRSat


static  SBBList                 mySBBList;

static  SeqBuildBlockRSatSpec    RSat[MaxNrVarRSATS]; //= {&mySBBList,
&mySBBList, &mySBBList, &mySBBList, &mySBBList, &mySBBList,
&mySBBList, &mySBBList};
long    lScanTimeOVSSats;


static  SBBList               my_fixed_rsat_SBBList;

static  SeqBuildBlockRSatSpecfixed  fixedRSat[MaxNrFixedRsats];// = {
&my_fixed_rsat_SBBList, &my_fixed_rsat_SBBList, &my_fixed_rsat_SBBList,
&my_fixed_rsat_SBBList};

long    lScanTimeOVSSats_fixed,
                    time_to_excit_in_sequ;        // time from the start of the sequence
itself up to the middle of the excitation pulse
double dsat_readoutFOV;                             // max excited spatial region (enlarged
VoI with max chem. shift)
double dsat_phaseFOV;                                       // max excited spatial region
(enlarged VoI with max chem. shift)


double  dGSAmplitudereduction_factor_90;
double  dGSAmplitudereduction_factor_pi;
```

```
///////////////////////////////////////////////////////////////////
// Gradient Pulses
///////////////////////////////////////////////////////////////////

/////////////////////////
// slice selection pulses
/////////////////////////

static sGRAD_PULSE ss_grad_exc( "ss_grad_exc" );     // gradient during excit.
// static sGRAD_PULSE ss_grad_ref( "ss_grad_ref" );  // refocussing gradient
static sGRAD_PULSE ss_grad_pi_sl( "ss_grad_pi_sl" ); // gradient during pi along
SL
static sGRAD_PULSE ss_grad_pi_ph( "ss_grad_pi_ph" ); // gradient during pi along
PH

////////////////////////
// phase encoding pulses
////////////////////////

static sGRAD_PULSE ss_encod_sl( "ss_encod_sl" );
static sGRAD_PULSE ss_encod_ph( "ss_encod_ph" );
static sGRAD_PULSE ss_encod_ro( "ss_encod_ro" );

/////////////////
// spoiler pulses
/////////////////

static sGRAD_PULSE ss_sp1_ph("ss_sp1_ph"); // spoiler 1 - phase
static sGRAD_PULSE ss_sp1_ro("ss_sp1_ro"); // spoiler 1 - readout
static sGRAD_PULSE ss_sp1_sl("ss_sp1_ph"); // spoiler 1 - slice

static sGRAD_PULSE ss_sp2_ph("ss_sp2_ph"); // spoiler 2 - phase
static sGRAD_PULSE ss_sp2_ro("ss_sp2_ro"); // spoiler 2 - readout
static sGRAD_PULSE ss_sp2_sl("ss_sp2_ph"); // spoiler 2 - slice

static sGRAD_PULSE ss_wssp_ph("ss_wssp_ph"); // water suppression spoiler -
phase
static sGRAD_PULSE ss_wssp_ro("ss_wssp_ro"); // water suppression spoiler -
readout
static sGRAD_PULSE ss_wssp_sl("ss_wssp_ph"); // water suppression spoiler - slice

static sGRAD_PULSE ss_mega1_ph("ss_mega1_ph"); // mega suppression spoiler -
phase
```

```
static sGRAD_PULSE ss_mega1_ro("ss_mega1_ro"); // mega suppression spoiler -
readout
static sGRAD_PULSE ss_mega1_sl("ss_mega1_sl"); // mega suppression spoiler -
slice
static sGRAD_PULSE ss_mega2_ph("ss_mega2_ph"); // mega suppression spoiler -
phase
static sGRAD_PULSE ss_mega2_ro("ss_mega2_ro"); // mega suppression spoiler -
readout
static sGRAD_PULSE ss_mega2_sl("ss_mega2_sl"); // mega suppression spoiler -
slice

static sGRAD_PULSE ss_finsp_sl("ss_finsp_sl"); // final spoiler - slice
static sGRAD_PULSE ss_finsp_ro("ss_finsp_ro"); // final spoiler - readout
static sGRAD_PULSE ss_finsp_ph("ss_finsp_ph"); // final spoiler - phase


////////////////////////////////////////////////////////////////////////
// MR Spectroscopy Sequence Functionality
////////////////////////////////////////////////////////////////////////

//static mrspec_seq_lib spec_lib;

//Elijah
/////
//static double sa_phase_array[16][4]; // static array
//static double sa_phase_array[16][5]; // add one entry to inlcude phase cycling for the
2nd ADC
/////
//Elijah
/////
//static long sl_phase_cycle; //Included by Elijah
/////
static long sl_excit_delay, sl_trueTE1, sl_trueTE2, sl_samplesBeforeEcho,
sl_trueTE2_minfix,
        sl_timeBeforeEcho, sl_aqu_fill_before, sl_aqu_fill_after;

//static long wsat_delay_betw_ws1_ws2,                    // additional delay
between wsat pulses
                        //wsat_delay_betw_ws2_ws3,        // additional delay
between wsat pulses
                        //sl_timeBeforeEcho, sl_aqu_fill_before, sl_aqu_fill_after;

//Elijah
///// add for second echo
static long sl_fill_before3rdRF, sl_fill_after3rdRF, sl_samplesBeforeEcho2,
        sl_timeBeforeEcho2, sl_aqu_fill_before2, sl_aqu_fill_after2;
```

```
/////

// CSI variables

static const short SIZE_OF_COMPLEXFLOAT = 8 ;              // win NT 4.0
static const long  MAX_FINAL_DATASIZE  = 128 * 1024 * 1024;   // 128MB e.g.
16*16*16*1024*4 (*8byte) ;
static long        sl_act_final_datasize ;
static const long  MAX_N_CSI_ENCODES = 32*32*32;
static long        sl_n_csi_encodes;
static short       ssh_1st_csi_addr[ MAX_N_CSI_ENCODES ],
                                    ssh_2nd_csi_addr[ MAX_N_CSI_ENCODES ],
                                    ssh_3rd_csi_addr[ MAX_N_CSI_ENCODES ];
static short ssh_csi_weight[ MAX_N_CSI_ENCODES ];
static short ssh_1st_csi_addr_offset, ssh_2nd_csi_addr_offset,
ssh_3rd_csi_addr_offset;
static double sd_1st_csi_grad_step, sd_2nd_csi_grad_step, sd_3rd_csi_grad_step;
static double sd_1st_csi_grad_offset, sd_2nd_csi_grad_offset,
sd_3rd_csi_grad_offset;
static double sd_read_pos, sd_phase_pos, sd_slice_pos;

// there is no such constant in csequence.h
#ifndef GRAD_RASTER_TIME
#define GRAD_RASTER_TIME (10)
#endif

// redefine some UI properties
// the functions defined below are registered with the UI within fSEQInit()

#ifndef VXWORKS

#include "MrServers/MrProtSrv/MrProtocol/libUILink/UILinkSelection.h"
#include "MrServers/MrProtSrv/MrProtocol/UILink/MrStdNameTags.h"
#include "MrServers/MrProtSrv/MrProtocol/UILink/StdProtRes/StdProtRes.h"



// set the "save uncombined" checkbox to true, which is the only mode supported by
VA11A


static bool returnSaveUncombinedValue(LINK_BOOL_TYPE* const, long)
{
   return true;
}
/*
```

```
static mrstd::vector<bool>
returnSaveUncombinedOptions(const LINK_BOOL_TYPE* const _this, bool&
verify, long)
{
   verify = false;
   mrstd::vector<bool> option(1);
   option[0] = true;
   return option;
}
*/
#endif
// VXWORKS

////////////////////////////////////////////////////////////////////////
// additional functionality NOT available in product
////////////////////////////////////////////////////////////////////////

#ifdef _DVP_DEBUG

// in order to read ASCII information during fSEQprep()

#include  "MrServers/MrSpecAcq/mrspec_seq_lib/CmdLineArg.h"
#include  "MrServers/MrSpecAcq/mrspec_seq_lib/CmdLineArg.cpp"
#include  "MrServers/MrSpecAcq/mrspec_seq_lib/mrspec_seq_lib.h"
#include  "MrServers/MrSpecAcq/mrspec_seq_lib/mrspec_seq_lib.cpp"
extern CmdLineArg __cla;

#endif



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

/*] END: */

NLS_STATUS fSEQInit( SeqLim *pSeqLim )
```

```
{
 static const char *ptModule = {"fSEQInit"};

 NLS_STATUS  lStatus = SEQU__NORMAL;
 // using SEQU_SEQ_NOT_INITIALIZED to indicate failure of this function


 GCProxy      theGCProxy;       // Declaration of Gradient Coil Proxy


 //////////////////////////////////////////////////////////////////////////////////////
 // let me introduce myself...
 //////////////////////////////////////////////////////////////////////////////////////

 pSeqLim->setMyOrigFilename ( __FILE__ );
 pSeqLim->setSequenceOwner  ( SEQ_OWNER_SIEMENS );
 pSeqLim->setSequenceHintText( (char *) "\n\
 Application: Spectroscopy \n\
     Basics: CSI, Spin-Echo \n\
      Build: "__DATE__"   "__TIME__"\n");

 pSeqLim->isSVSSequence( TRUE );
 pSeqLim->isCSISequence( TRUE );
 pSeqLim->setSequenceCard( SEQ::SEQUENCE_CARD_SPECTROSCOPY );





        // * --------------------------------------------------------------------------- *
  // * Add RSats to SBBList                                            *
  // * --------------------------------------------------------------------------- *


  //  Loop counter variable RSats
  long        lI;
  for ( lI=0; lI<MaxNrVarRSATS; lI++ )  {
     RSat[lI].addToSBBList(&mySBBList);
  }




  // Loop counter fixed RSats
  for ( lI=0; lI<MaxNrFixedRsats; lI++ )  {
     fixedRSat[lI].addToSBBList(&mySBBList);
         }
 ////////////////////////////////////////////////////////////////////////////////
 // the system requirements: frequency, and gradient power
 ////////////////////////////////////////////////////////////////////////////////


 pSeqLim->setAllowedFrequency(40200000, 125000000 ); // Hz, extended for 3T
 pSeqLim->setRequiredGradAmpl( 16.0 );      //  mT/m /* but we have 20 on
TUBRBO gradients */
 pSeqLim->setRequiredGradSlewRate( 25.0 );  // (mT/m)/ms, /* TUBRBO gradients
*/
 pSeqLim->setGradients( SEQ::GRAD_NORMAL ); // SEQ::GRAD_FAST,
SEQ::GRAD_WHISPER);
 pSeqLim->setForcePositioningOnNDIS(SEQ::ON);  /*! EGA-07 !*/

         // since Iso-Centre Scanning, this parameters forces the sequence

         // to be  positioned only on NDIS images

 ////////////////////////////////////////////////////////////////////////////////
 // VectorSize of time domain signal  (         min,         max,     mult,       def  )
 // this is a spectroscopy specific variable
 ////////////////////////////////////////////////////////////////////////////////

 pSeqLim->setVectorSize( 512, 2048, SEQ::BASE2, 1024 );
 pSeqLim->setReadoutOSFactor( 2. ); // default
 pSeqLim->setRemoveOversampling( SEQ::YES, SEQ::NO );

 ////////////////////////////////////////////////////////////////////////////////
 // Base matrix size of the image
 ////////////////////////////////////////////////////////////////////////////////

 pSeqLim->setBaseResolution( 8,32, SEQ::INC_NORMAL, 16 );  // def 16
 pSeqLim->setPELines( 8, 32, 1, 16);
 pSeqLim->setMaxPhaseResolution( 2 );

 pSeqLim->setfinalMatrixSizeRead( 8, 32, SEQ::BASE2, 16 );
 pSeqLim->setfinalMatrixSizePhase( 8, 32, SEQ::BASE2, 16 );


 ////////////////////////////////////////////////////////////////////////////////
 // slices and partitions
 ////////////////////////////////////////////////////////////////////////////////
 //pSeqLim->setConcatenations( 1, 1, 1, 1 ); // this is the default, concat. does not
show up in the UI
 pSeqLim->setSlices( 1, 1, 1, 1);
 //pSeqLim->setMultiSliceMode( SEQ::MSM_INTERLEAVED,
SEQ::MSM_SEQUENTIAL );
 //pSeqLim->setSliceSeriesMode( SEQ::INTERLEAVED, SEQ::ASCENDING,
SEQ::DESCENDING );
 //pSeqLim->enableSliceShift(); // this is the default
 //pSeqLim->enableMSMA(); // this is the default, MSMA multi-slice, multi-angle,
                                                       // is prevented by setSlices( 1, 1, 1, 1 )
                                                       // useful in multislice mode to prevent
different slice-groups
 //pSeqLim->enableOffcenter(); // this is the default
 //pSeqLim->setAllowedSliceOrientation (SEQ::DOUBLE_OBLIQUE); // this is the
default
 // pSeqLim->setMinSliceResolution (0.5);  // this is the default
 // pSeqLim->setEllipticalScanning (ES_OFF);


 ////////////////////////////////////////////////////////////////////////////////////////
 // 3D CSI
 ////////////////////////////////////////////////////////////////////////////////////////

 pSeqLim->setDimension( SEQ::DIM_3, SEQ::DIM_2 ); // order sensitive !,
switching from 3D to 2D once is required to generate a consistent 2D protocol;
especially for OutOfPlanePhaseSteps and NumberOfFrames
 //pSeqLim->getDimension().setDisplayMode( SEQ::DM_OFF ); // prevents display
of the dimension parameter
 pSeqLim->setPartition( 8,32, SEQ::INC_NORMAL, 8 );
 pSeqLim->setfinalMatrixSizeSlice( 8, 16, SEQ::BASE2, 8 );
 pSeqLim->setImagesPerSlab( 8, 32, SEQ::BASE2, 8 ); // should be redundant!
 pSeqLim->setSlabThickness( 55, 300 );             // 3D FOV limits
 pSeqLim->set3DPartThickness( 2, 40, 1, 15);
 pSeqLim->setMinSliceResolution( 0.5 );



 ////////////////////////////////////////////////////////////////////////////////////
 // Bandwidth used for data acqu. (   min, max, inc, def   )
 ////////////////////////////////////////////////////////////////////////////////////

 //Elijah
///// Need to let the 1st ADC be much shorter for the 2nd ADC to happen at TE ~
280ms. but doesn't seem to be able to set for multiple bandwidth. So fix the first one
to higher one.
 //pSeqLim->setBandWidth( 0, 1000, 2000, 10, 1000 );
 pSeqLim->setBandWidth( 0, 1000, 6000, 10, 5000 ); //increased to 2500 to realize a
shorter ACQ-window

 pSeqLim->setBandWidth( 1, 1000, 6000, 10, 1000 );
/////


 ////////////////////////////////////////////////////////////////////////////////
 // Echo Time                          ( No.,      min,      max,      inc,      def)  ;
 ////////////////////////////////////////////////////////////////////////////////
 //Elijah
 /////
 pSeqLim->setContrasts(2,2,1,2);
 /////

 pSeqLim->setTE( 0, 30000, 1500000, 1000, 30000 );

//Elijah
 /////
 pSeqLim->setTE( 1, 30000, 2000000, 1000, 280000 );  //  TE max = 2.0 sec new
since VB15
   /////

 ////////////////////////////////////////////////////////////////////////////////
 // Repetition Time                    ( No.,      min,      max,      inc,      def)  ;
 ////////////////////////////////////////////////////////////////////////////////
//Elijah
 /////
 //pSeqLim->setTR(0, 500000, 10000000, 10000, 1600000 );
 pSeqLim->setTR(0, 200000, 30000000, 10000, 2000000 );  //min TR  = 200ms
//max =  30sec  sinde  VB15A
 /////
 pSeqLim->setRepetitions( 0, 7, 1, 0);   //2D max 8, 3D max 4 realized with
GetLimitHandler
 pSeqLim->setRepetitionsDelayTime( 0, 10000000, 100000, 0);

 ////////////////////////////////////////////////////////////////////////////////
 // CSI FOV
 ////////////////////////////////////////////////////////////////////////////////

 pSeqLim->setReadoutFOV( 55, 300, 1, 120 );
 pSeqLim->setPhaseFOV( 55, 300, 1, 120 );
 pSeqLim->setSliceThickness( 5.0, 40.0, 1.0, 20.0 );  // identical to VoISizeSlice in
2D CSI
    //pSeqLim->setSliceDistanceFactor( 0.0, 8.0, 0.01, 0.20 );
 ////////////////////////////////////////////////////////////////////////////////////
 // VOI definition              (       min,         max,         inc,          def)  ;
 // these are spectroscopy specific variables
 ////////////////////////////////////////////////////////////////////////////////////

 pSeqLim->setVoIPosCor( -150, 150, .1, 0);
 pSeqLim->setVoIPosSag( -150, 150, .1, 0);
 pSeqLim->setVoIPosTra( -150, 150, .1, 0);

 pSeqLim->setVoISizePhase( 30, 160, 1, 60 );
 pSeqLim->setVoISizeReadout( 30, 160, 1, 60 );
 pSeqLim->setVoISizeSlice( 30, 160, 1, 40); // 3D CSI

 ////////////////////////////////////////////////////////////////////////////////////
 // RF                              (     min,       max,      inc,     def)  ;
 ////////////////////////////////////////////////////////////////////////////////////

   pSeqLim->setFlipAngle( 0.000, 180.000, 1.000, 90.000 );
   pSeqLim->setExtSrfFilename( "%MEASCONST%/extrf_spec.pls" );


 /////////////////////////////////////////////////////////////////////////////////
 // phase cycling
 // ... is switched off in CSI sequences
 /////////////////////////////////////////////////////////////////////////////////

 //pSeqLim->setPhaseCyclingType( SEQ::PHASE_CYCLING_NONE );
 //pSeqLim->setPhaseCyclingType( SEQ::PHASE_CYCLING_AUTO, /* default */
 //                         SEQ::PHASE_CYCLING_NONE,
 //                         SEQ::PHASE_CYCLING_TWOSTEP,
 //                         SEQ::PHASE_CYCLING_EIGHTSTEP,
 //                         SEQ::PHASE_CYCLING_EXORCYCLE,
 //                         SEQ::PHASE_CYCLING_SIXTEENSTEP_EXOR );

 /////////////////////////////////////////////////////////////////////////////////
 // phase encoding type
 // this is a spectroscopy specific variable
 /////////////////////////////////////////////////////////////////////////////////

 pSeqLim->setPhaseEncodingType( SEQ::PHASE_ENCODING_FULL,
           SEQ::PHASE_ENCODING_WEIGHTED,
SEQ::PHASE_ENCODING_ELLIPTICAL  );
```

```
  ////////////////////////////////////////////////////////////////////////////////////
  // water suppression (default setting )
  ////////////////////////////////////////////////////////////////////////////////////
  // WATER_SUPPRESSION_OFF: No WET RF or Gradients
  // WATER_SUPPRESSION_RF_OFF: for the reference scan: only WET Gradient
spoiler are On
  pSeqLim->setWaterSuppression( SEQ::WATER_SATURATION,
SEQ::WATER_SUPPRESSION_OFF, SEQ::WATER_SUPPRESSION_WEAK,
SEQ::WATER_SUPPRESSION_RF_OFF);


  ////////////////////////////////////////////////////////////////////////////////////
  // Preperation Pulses                (     min,     max,     inc,     def)  ;
  // this is a spectroscopy specific variable
  ////////////////////////////////////////////////////////////////////////////////////

  pSeqLim->setPreparingScans( 0, 16, 1, 4);    //  def 4

  ////////////////////////////////////////////////////////////////////////////////////
  // Acquisition delay                 (     min,     max,     inc,     def)  ;
  // this is a spectroscopy specific variable
  ////////////////////////////////////////////////////////////////////////////////////
  /*
  pSeqLim->setAcquisitionDelay( 0, +100000, 1000, 0 );
  */

  ////////////////////////////////////////////////////////////////////////////////////
  // Averages / Repetitions            (    min,     max,     inc,     def)  ;
  ////////////////////////////////////////////////////////////////////////////////////

  pSeqLim->setAverages( 1, 64, 1, 1 );

  ///////////////////////////////////////////////////////////////////////////////////
  // Bandwidth of Water Excitation Pulses     (    min,     max,     inc,     def)  ;
  ////////////////////////////////////////////////////////////////////////////////////

  pSeqLim->setRfBandwidth( 20, 80, 5, 35 );


  // -------------------------------------------------------------------------------------
  // Loop control
  // -------------------------------------------------------------------------------------

  pSeqLim->setAveragingMode             (SEQ::INNER_LOOP,
SEQ::OUTER_LOOP);
```

```
  ////////////////////////////////////////////////////////////////////////////////
  // spectroscopy sequences do NOT use the raw data filters of the imaging sequences
  ////////////////////////////////////////////////////////////////////////////////

  pSeqLim->setFilterType( SEQ::HAMMING,  SEQ::PRESCAN_NORMALIZE   );

  ////////////////////////////////////////////////////////////////////////////////
  // default adjust procedures
  ////////////////////////////////////////////////////////////////////////////////

  pSeqLim->setAdjShim( SEQ::ADJSHIM_ADVANCED,
SEQ::ADJSHIM_STANDARD, SEQ::ADJSHIM_TUNEUP );
  pSeqLim->setAdjWatSup( SEQ::ENABLE, SEQ::DISABLE );



          ////////////////////////////////////////////////////////////////////////////////
          // new with VA21A
          ////////////////////////////////////////////////////////////////////////////////

          pSeqLim->setSliceSelectDeltaFrequency( -5.0, 0.0, 0.01, 0.0 ); // suitable
range for 1H


          ////////////////////////////////////////////////////////////////////////////////
          // Outer Volume Suppression  new with VA25A
          ////////////////////////////////////////////////////////////////////////////////

    //setOuterVolumeSuppression (SEQ::Switch def, SEQ::Switch mode2 =
(SEQ::Switch) 0);
    pSeqLim->setOuterVolumeSuppression ( SEQ::OFF, SEQ::ON );  // def OFF



          ////////////////////////////////////////////////////////////////////////////////
          // additional delta frequency for RSats new with VA25A
          ////////////////////////////////////////////////////////////////////////////////


    //setRSatDeltaFrequency (double dMinimum, double dMaximum, double
dIncrement, double dDefault);
```

```
   //pSeqLim->setRSatMode                  (     SEQ::RSAT_REG,
SEQ::RSAT_QUICK       );
   pSeqLim->setRSats                    (        0,  MaxNrVarRSATS,     1,        0  );
   pSeqLim->setRSatThickness            (     8.000,   150.000,     1.000,    30.000);
//min = 8mm since VA15A
   pSeqLim->setRSatDeltaFrequency                        (      -5.0,       0.0,      0.01,
0);



         //////////////////////////////////////////////////////////////////////////////////////
         // spectral suppression  new with VA25A
         //////////////////////////////////////////////////////////////////////////////////////


    //setSpectralSuppression (SEQ::SpectralSuppression def,
SEQ::SpectralSuppression mode2 = (SEQ::SpectralSuppression) 0,
SEQ::SpectralSuppression mode3 = (SEQ::SpectralSuppression) 0);
         pSeqLim->setSpectralSuppression ( SEQ::SPEC_SUPPR_NONE,
SEQ::SPEC_SUPPR_LIPID, SEQ::SPEC_SUPPR_LIPID_WATER );


   //setSpecLipidSupprBandwidth (double dMinimum, double dMaximum, double
dIncrement, double dDefault);
   pSeqLim->setSpecLipidSupprBandwidth ( 0.80, 2.50, 0.05, 0.9);            // in ppm
for 1.55 the pulse dur. is 25.6 ms at 1.5T systems



   //setSpecLipidSupprDeltaPos (double dMinimum, double dMaximum, double
dIncrement, double dDefault);
         pSeqLim->setSpecLipidSupprDeltaPos ( -6.0, -3.0, 0.01, -3.4);   // in ppm
default for lipid



   //setSpecWaterSupprBandwidth (double dMinimum, double dMaximum, double
dIncrement, double dDefault);
         pSeqLim->setSpecWaterSupprBandwidth ( 0.80, 2.50, 0.05, 0.9);   // in ppm



   //setSpecWaterSupprDeltaPos (double dMinimum, double dMaximum, double
dIncrement, double dDefault);
         pSeqLim->setSpecWaterSupprDeltaPos ( -0.5, 3.0, 0.01, 0.0);    /// in ppm
```

```
///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///////
  // the coil combine mode, we don't want to have; adaptive coil combine is not
possible, because then save uncombined is not allowed

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///////

          ParLimOption<SEQ::CoilCombineMode>& coilCombine = pSeqLim-
>getCoilCombineMode();
          coilCombine.set(SEQ::COILCOMBINE_SUM_OF_SQUARES);
          coilCombine.setDisplayMode(SEQ::DM_OFF);



  ////////////////////////////////////////////////////////////////////////////////////////
  // Data Receive & Image calculation
  ////////////////////////////////////////////////////////////////////////////////////////

          pSeqLim->setICEProgramFilename (
"%SiemensIceProgs%\\IcePrgSpectroscopy");





   ////////////////////////////////////////////////////////////////////////////////////////
  // register the redefined functions specified above with the UI
  ////////////////////////////////////////////////////////////////////////////////////////


#ifndef VXWORKS

  LINK_BOOL_TYPE* pBool = _search<LINK_BOOL_TYPE>( pSeqLim,
MR_TAG_SAVE_UNCOMBINED );
  if(  1 == 0 )  // the save uncombined box is availabe since VB11A
  {
    pBool->registerGetValueHandler( returnSaveUncombinedValue ); //
MRUILinkSelection.h
    pBool->registerGetOptionsHandler( NULL /*returnSaveUncombinedOptions*/ );
    pBool->registerSetValueHandler( NULL );
  }
```

```
#endif




#ifndef VXWORKS
        // UI-utilities defined in spectro_ui.cpp
        Init_GetLimitHandler_NofMeasurementLimit  (pSeqLim);   // overload
GetLimitHandler Number of Measurement
        Init_GetLimitHandler_RFBandwidth          (pSeqLim);   // overload
GetLimitHandler RFBandwith of WET pulse

        Init_SolveDimNofMeasConflict              (pSeqLim);   // solve handler
Dimension - Number of Measurement Conflict
        Init_SolveVecSizeTRConflict               (pSeqLim);   // solve handler increase
vector size -> increase TR
        Init_SolveBandWidthTRConflict             (pSeqLim);   // solve handler RF
BW decreased -> increase TRmin

   // WET
        Init_SolveWaterSuppSelectionTRConflict    (pSeqLim);     // solve handler
Weak Water Supp On -> increase TRmin
        Init_SolveWaterSuppBWTRConflict                        (pSeqLim);   // solve
handler Water Suppression BW decrease -> increase TRmin

        // spectral suppression
        Init_SolveSpectralSuppTETRConflict       (pSeqLim);       // solve handler
SpectralSuppression ON -> increase TE  && TRmin
 //  Init_SolveSpectralWaterSuppBWTEConflict   (pSeqLim);   // solve handler
Spectral Water Supp BW decrease -> increase TEmin // does not work jet

        //FullyExcitedVoI
        Init_SolveFullyExcitedVoITRConflict       (pSeqLim);   // solve handler
FullyExcitedVoI ON -> increase TRmin


#endif



return (lStatus);


}
```

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

/*] END: */

NLS_STATUS fSEQPrep
(
  MrProt     *pMrProt,     /* IMP: Measurement protocol  */
  SeqLim     *pSeqLim,     /* IMP: Sequence limits       */
  SeqExpo    *pSeqExpo     /* EXP: Returned values       */
)
{
  static const char *ptModule          = {"fSEQPrep"};
  static const long ECHO_DELAY = 200;                // time we have to start earlier !!

  NLS_STATUS   lStatus = SEQU__NORMAL;
  // using SEQU_SEQ_NOT_PREPARED to indicate failure of this function

  long          lKernelRequestsPerMeasurement = 0;
  double        dMeasureTimeUsec       = 0.;  /* Measurement time (usec)   */
  double        dTotalMeasureTimeMsec = 0.;  /* Total measurement time    */
  //double        dScanTimeTrigHalt     = 0.;/* Trigger halt block time   */
  //long lNoiseMeasTime = 0;

  double        dRfEnergyInSBBs                                  = 0.; /* RF energy in
SBB calls    */
  double        dRfEnergyInSBBs_fixed_rsats       = 0.; /* RF energy in SBB calls    */
  double        dRfEnergyInSRFs                                  = 0.; /* RF energy in
SRF         */

  long lFrequency, ramptime, sp1dur, sp2dur, encoddur, wsatpulsedur, l,n, tau, tau1,
tau2, tau3, h2osup_dur, finalspoil_dur;

  //Elijah
  ///// for second echo
  long n2;
  /////
  double sp1ampl, sp2ampl, wsspgradmoment, d, dmin, dmax, max_grad_ampl,
        alpha_pi_sl, alpha_pi_ph, refoc, voxelshift_read, voxelshift_phase,
voxelshift_slice, gauss_width_td;
  char name_pi_sl[32], name_pi_ph[32], name_exc[32];



  double       d_rf_pi_ph_GSAmplitude = 0;                 /* new slice select gradient
with reduced strength for in-plane 180 */
  double       d_rf_exc_GSAmplitude   = 0;                 /* new slice select gradient with
reduced strength for in-plane 90  */
  long         lI                     = 0;                 /* helper variables          */

  char         ptIdentdummy[7];                           // * Ident strings for sat pulses *




  if( pMrProt->preScanNormalizeFilter().on() )
          pMrProt->preScanNormalizeFilter().storeCXIma( true ); // exception, change
a protocol during prepare() !!!


  //////////////////////////////////////////////////////////////////////////////////////////
  // Get the current nucleus from the protocol
  //////////////////////////////////////////////////////////////////////////////////////////
  MeasNucleus mainNucleus(pMrProt->txSpec().nucleusInfoArray()[0].nucleus());
  double larmorconst = mainNucleus.getLarmorConst();



////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///////
  // the coil combine mode, we don't want to have; adaptive coil combine is not
possible, because then save uncombined is not allowed

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///////

  if ( pMrProt->coilCombineMode() !=
SEQ::COILCOMBINE_SUM_OF_SQUARES) return SEQU_ERROR;


  ////////////////////////////////////////////////////////////////////////////////////////
  // Set up DICOM header
  ////////////////////////////////////////////////////////////////////////

  // missing, sr!

  ////////////////////////////////////////////////////////////////////////
  // compute VoI and FoV orientation and position
  ////////////////////////////////////////////////////////////////////////

  if( !(ss_voi.prep( pMrProt, pSeqLim, pMrProt->spectroscopy().VoI(), 0 )) )
     return SEQU_SEQ_NOT_PREPARED;

  // in the case of multi-slice CSI we would need to prepare more than the one slice
  // in this case, the function fSUPrepSlicePosArray() might be helpful
  if( !(ss_fov.prep( pMrProt, pSeqLim, pMrProt->sliceSeries()[0] , 0 )) )
     return SEQU_SEQ_NOT_PREPARED;

  ////////////////////////////////////////////////////////////////////////
  // compute general limitations
  ////////////////////////////////////////////////////////////////////////
   /*
   Turbo Gradient system:

         GPA Type                              = K2217_500V_300A
         GPA GradMaxAmplAbsolute             [mT/m] = 20
         GPA GradMaxAmplNominal              [mT/m] = 16
         GPA GradMaxAmpl                     [mT/m] = 16 / 16 / 16
         GPA GradMinRiseTimeAbsolute      [us/(mT/m)] = 40
         GPA GradMinRiseTime              [us/(mT/m)] = 40 / 40 / 60
         GPA GradClipRiseTime             [us/(mT/m)] = 30
         GPA GradMaxSlewRateAbsolute      [mT/(m*ms)] = 25
         GPA GradMaxSlewRate              [mT/(m*ms)] = 25 / 25 / 16.6667
         GPA GradClipSlewRate             [mT/(m*ms)] = 33.3333
   */


  ramptime = 700; /* us */    /* since VB11A  */

  // max_grad_ampl = .001 * ramptime * pSeqLim->getRequiredGradSlewRate(); //
20 mT/m
  // check with pSeqLim->getRequiredGradAmpl() dis-abled
  max_grad_ampl =  20.0 / sqrt( 3. ); //     20.0 / sqrt( 3. ) = 11.5 mT/m

  #ifdef _OWN_DEBUG_011
  cout <<
"=============================================================" <<
endl;
```

```
 cout << "==>ramptime:                              " << ramptime <<  "us" << endl;
 cout << "==>max_grad_ampl:                            " << max_grad_ampl <<
"mT/m" << endl;
 #endif




 //////////////////////////////////////////////////////////////////////////
 // set product sequence default values
 //////////////////////////////////////////////////////////////////////////

 encoddur = 2800;

 sp1ampl = 7.;
 sp1dur = 2000;     // mega spoiler duration
 sp2ampl = 11.5;
 sp2dur = 4000;
 wsspgradmoment = 168.; // resulting from VA12B: 33.6 ms * 5 mT/m
 strcpy( name_exc, "hsinc_400_8750" );
 strcpy( name_pi_sl, "mao_400_4" );
 strcpy( name_pi_ph, "mao_400_4" );
 alpha_pi_sl = 180.;
 alpha_pi_ph = 180.;
 refoc = .515;
 voxelshift_read = voxelshift_phase = voxelshift_slice = .5;



 //////////////////////////////////////////////////////////////////////////
 // get prep-information from sequence param file
 //////////////////////////////////////////////////////////////////////////

 #ifdef _DVP_DEBUG // this flag is NOT set during product sequence compilation

 mrspec_seq_lib spec_lib;

 if( ( spec_lib.get_parameters( "csi_se_param.asc" ) ) ){

   __cla.set_opt( "RAMPTIME", 1 );
   __cla.set_opt( "SP1AMPL", 1 );
   __cla.set_opt( "SP1DUR", 1 );
   __cla.set_opt( "SP2AMPL", 1 );
   __cla.set_opt( "SP2DUR", 1 );
   __cla.set_opt( "WSSPGRADMOMENT", 1 );
   __cla.set_opt( "NAME_EXC", 1 );
   __cla.set_opt( "NAME_PI_SL", 1 );
   __cla.set_opt( "NAME_PI_PH", 1 );
   __cla.set_opt( "ALPHA_PI_SL", 1 );
   __cla.set_opt( "ALPHA_PI_PH", 1 );
           __cla.set_opt( "REFOC", 1 );
           __cla.set_opt( "VOXELSHIFT_READ", 1 );
           __cla.set_opt( "VOXELSHIFT_PHASE", 1 );
           __cla.set_opt( "VOXELSHIFT_SLICE", 1 );
           //__cla.set_opt( "ACQUIRED_SAMPLES_BEFORE_ECHO", 1 );  //TS


   if( __cla.is_opt( "RAMPTIME" ) ){
       l = __cla.arg2int( "RAMPTIME", 1 );
       if( l > 200 && l < 1000 )
           ramptime = l;
   }
   if( __cla.is_opt( "SP1AMPL" ) ){
       d = __cla.arg2flt( "SP1AMPL", 1 );
       if( d > 1 && d < max_grad_ampl )
           sp1ampl = d;
   }
   if( __cla.is_opt( "SP1DUR" ) ){
       l = __cla.arg2int( "SP1DUR", 1 );
       if( l > 100 && l < 5000 )
           sp1dur = l;
   }
   if( __cla.is_opt( "SP2AMPL" ) ){
       d = __cla.arg2flt( "SP2AMPL", 1 );
       if( d > 1 && d < max_grad_ampl )
           sp2ampl = d;
   }
   if( __cla.is_opt( "SP2DUR" ) ){
       l = __cla.arg2int( "SP2DUR", 1 );
       if( l > 100 && l < 5000 )
           sp2dur = l;
   }
   if( __cla.is_opt( "WSSPGRADMOMENT" ) ){
       d = __cla.arg2flt( "WSSPGRADMOMENT", 1 );
       if( d > 1 && d < 300 )
           wsspgradmoment = d;
   }

            if( __cla.is_opt( "REFOC" ) ){
       d = __cla.arg2flt( "REFOC", 1 );
       if( d > 0 && d < 1 )
           refoc = d;
```

```
     }

     if( __cla.is_opt( "NAME_EXC" ) )
        strcpy( name_exc, __cla.arg2str( "NAME_EXC", 1 ) );
     if( __cla.is_opt( "NAME_PI_SL" ) )
        strcpy( name_pi_sl, __cla.arg2str( "NAME_PI_SL", 1 ) );
     if( __cla.is_opt( "NAME_PI_PH" ) )
        strcpy( name_pi_ph, __cla.arg2str( "NAME_PI_PH", 1 ) );

     if( __cla.is_opt( "ALPHA_PI_SL" ) ){
         d = __cla.arg2flt( "ALPHA_PI_SL", 1 );
         if( d > 0 && d < 180. )
             alpha_pi_sl = d;
     }
     if( __cla.is_opt( "ALPHA_PI_PH" ) ){
         d = __cla.arg2flt( "ALPHA_PI_PH", 1 );
         if( d > 0 && d < 180. )
             alpha_pi_ph = d;
     }

             if( __cla.is_opt( "VOXELSHIFT_READ" ) )
                          voxelshift_read = __cla.arg2flt( "VOXELSHIFT_READ", 1 );

             if( __cla.is_opt( "VOXELSHIFT_PHASE" ) )
                          voxelshift_phase = __cla.arg2flt( "VOXELSHIFT_PHASE", 1 );

             if( __cla.is_opt( "VOXELSHIFT_SLICE" ) )
                          voxelshift_slice = __cla.arg2flt( "VOXELSHIFT_SLICE", 1 );


  }
  else{
      fprintf( stdout, "\nfSeqPrep():"
                        "\ncan't find customized sequence parameters; using default values\n\n"
);
  }

  fprintf( stdout, "\nfSeqPrep(): customize-able sequence parameters are\n"
                    "\nramptime %d"
                    "\nsp1dur %d \nsp1ampl %f \nsp2dur %d \nsp2ampl %f
\nwsspgradmoment %f"
                    "\nname_exc %s \nname_pi_sl %s \nname_pi_ph %s"
                    "\nrefoc %f"
                            "\nalpha_pi_sl %f \nalpha_pi_ph %f\n\n",
                    ramptime, sp1dur, sp1ampl, sp2dur, sp2ampl, wsspgradmoment,
name_exc, name_pi_sl, name_pi_ph,
                 refoc, alpha_pi_sl, alpha_pi_ph );
  #endif

  //////////////////////////////////////////////////////////////////////////
  // Prepare Osc. Bit
  //////////////////////////////////////////////////////////////////////////

  //ss_osc1.setIdent( RTEIDENT_Osc0 );
  ss_osc1.lCode     = SYNCCODE_OSC0;
  ss_osc1.lDuration = OSCTRIGGERTIMEus; /* 10 see
...\Measurement\Sequence\csequence.h */


  //////////////////////////////////////////////////////////////////////////
  // Prepare the RF pulse structures
  //////////////////////////////////////////////////////////////////////////

  //////////////////////////////////////////////
  // excitation pulse (READOUT direction)
  ///////////////////////////////////// ////////

  ss_rf_exc.setTypeExcitation();
  ss_rf_exc.setDuration( 2600 ) ;
  ss_rf_exc.setFlipAngle( pMrProt->flipAngle() );
  ss_rf_exc.setInitialPhase( 0 );
  ss_rf_exc.setFamilyName( name_exc );
  ss_rf_exc.setThickness( pMrProt->spectroscopy().VoI().readoutFOV() ); /*! EGA-
04; EGA-02; EGA-05 !*/

  if( !( ss_rf_exc.prepExternal( pMrProt, pSeqExpo ) ) )
      return ss_rf_exc.getNLSStatus();




    // reduction of the slice gradients for VoI excitation
    // dGSAmplitudenreduction_factor increases the VOI by factor 2 of the max.
chemical shift displacement (1.85ppm)
          // BWTimeProduct of Pi/2-pulse: 8.75
          // empiric factor: 0.78
    dGSAmplitudereduction_factor_90 = 0.78 * (1 - 2 * (1.85 * (pMrProt-
>txSpec().frequency()*1E-6) /(8.75/(ss_rf_exc.getDuration()*1E-6)) ));  //with
empirical factor 0.78 to compensate the slice profile
          d_rf_exc_GSAmplitude = dGSAmplitudereduction_factor_90 *
ss_rf_exc.getGSAmplitude();
```

```
         #ifdef _DVP_DEBUG_UT
         cout << endl;
   cout  << "   --------------- Calc. of fully excited VoI Slice Grad (RO / 90)   -----------
----   "              << endl;
   cout << "   Frequency:                              " << pMrProt->txSpec().frequency()
<< endl;
         cout << "   ss_rf_exc.getDuration():                  " << ss_rf_exc.getDuration()
<< endl;
   cout << "==>Bandwidth of pulse:                        " <<
(8.75/(ss_rf_exc.getDuration()*1E-6)) << endl;
   cout << "==>alt: ss_rf_exc.getGSAmplitude():           " <<
ss_rf_exc.getGSAmplitude() << endl;
   cout << "==>neu: d_rf_exc_GSAmplitude:                 " << d_rf_exc_GSAmplitude
<< endl;
   cout << "==>dGSAmplitudereduction_factor_90:           " <<
dGSAmplitudereduction_factor_90 << endl;
 // cout << "==>dGSAmplitudereduction_factor_90 (worst case):  " << (0.78 * (1 - 2
* (1.85*123/(8.75/(2600*1E-6))))) << endl;
 // cout << "==>Gneu/Galt:                             " <<
(d_rf_exc_GSAmplitude/ss_rf_exc.getGSAmplitude()) << endl;
//  cout << "## VOIalt R >> L    (exc.)                 =" <<
(8.75/(ss_rf_exc.getDuration()*1E-6)) /(ss_rf_exc.getGSAmplitude()*larmorconst)
<< endl;
//  cout << "## VOIneu R >> L                           =" <<
(8.75/(ss_rf_exc.getDuration()*1E-6)) /(d_rf_exc_GSAmplitude *larmorconst) <<
endl;
//  cout << "## zu saettigen min R >> L                 =" <<
((8.75/(ss_rf_exc.getDuration()*1E-6))/d_rf_exc_GSAmplitude -
   //
(8.75/(ss_rf_exc.getDuration()*1E-6))/ss_rf_exc.getGSAmplitude()) /larmorconst  <<
endl;
         cout << endl;
   #endif




  // 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
```

```
        if (!(pMrProt->spectroscopy().outerVolumeSuppression()) )
        {   // fully excited VOI = OFF
                lFrequency = (long)( .5 + ss_rf_exc.getGSAmplitude() * larmorconst *
ss_voi.getSliceOffCenterRO() );  /*! EGA-05 !*/
        }
        else
        {   //cout << " ++++++++   fully_excited_VOI = ON  , lfrequency offset  , RO
++++++++  "  << endl;
                lFrequency = (long)( .5 + d_rf_exc_GSAmplitude * larmorconst *
ss_voi.getSliceOffCenterRO() );     /*! EGA-05 !*/
        }

  // correction of chemical shift displacement artefact
  lFrequency += (long)(pMrProt->txSpec().frequency() /* in Hz */ * 1E-6 * pMrProt-
>spectroscopy().dDeltaFrequency) /* in ppm */;  /*! EGA-05 !*/
  ss_ph_s_exc.setFrequency( lFrequency );                              /*! EGA-05
!*/
  ss_ph_n_exc.setFrequency( 0L );


        #ifdef _DVP_DEBUG
        cout << " -------------------------frequency shift VOI in RO direction -------------
-----"   << endl;
   cout << "## ss_voi.getSliceOffCenterRO()   (shift RO):  " <<
ss_voi.getSliceOffCenterRO()                    << endl;
   cout << "## larmorconst:                              " << larmorconst
                          << endl;
   cout << "## ss_ph_s_exc.getFrequency() (frequ shift)    " <<
ss_ph_s_exc.getFrequency()                      << endl;
        cout << "## d_rf_exc_GSAmplitude:                     " <<
d_rf_exc_GSAmplitude                                    << endl;
        cout << "## d_rf_exc_GSAmplitude*gamma:               " <<
d_rf_exc_GSAmplitude*  larmorconst << endl;
   cout << endl;
   #endif


  // computation of start-phase for off-resonant pulses as in
  // \n4\comp\measurement\sequence\libRT\sFREQ_PHASE::prepSet()
  // pulse asymmetry: 0.5 (== symmetric pulse)
  // input units:
  // [Frequency] = Hz
  // [pulse duration] = us
  // output unit:
  // [Phase] = deg.
  // the frequency (and pulse) dependent phase portion
  ss_ph_s_exc.setPhase( - lFrequency * (360./1e6) *
                                                  ss_rf_exc.getDuration() *
ss_rf_exc.getAsymmetry() /* = .5 */ );
  ss_ph_n_exc.setPhase(  - lFrequency * (360./1e6) *
                                                  ss_rf_exc.getDuration() * (1. -
ss_rf_exc.getAsymmetry()) /* = .5 */  );

  #ifdef _DVP_DEBUG
    fprintf (stdout,  "\nfSEQPrep(): SLICE direction properties"
                  "\nactual flip angle %f deg. \nactual pulse duration %f"
                  "\nVoI size %f mm \ngradient strength normal VoI %f mT/m"
                  "\npos. offset %f mm \n frequ. offset %d Hz \n\n",
                  (float)(ss_rf_exc.getActualFlipAngle()),
                  (float)(ss_rf_exc.getDuration()),
                  (float)(ss_rf_exc.getThickness()),
                  (float)(ss_rf_exc.getGSAmplitude()),
                  (float)(ss_voi.getSliceOffCenterRO()),
                  (int)lFrequency );
  #endif

  ////////////////////////////////////////
  // refocussing pulse (SLICE direction)
  ////////////////////////////////////////

    #ifndef VXWORKS
        {
                              // this code determines the shortest PI-pulse duration

                              // since an RF-pulse may not be prepared twice, we use a
dummy pulse

                              // getActualFlipAngle() does currently NOT work on the
scanner
                              // hence, we compute the pulse duration only on the host and
store
                              // it in the protocoll, which is then available to the host

                              ss_rf_dummy.setDuration( 2600 );
                              ss_rf_dummy.setFlipAngle( 180. );
                              ss_rf_dummy.setFamilyName( name_pi_sl ); // note that the
duration computed here is used for BOTH refocussing pulses

                         // which is might not that you want to have if the pulses are different
```

```
                // (i.e. name_pi_sl != name_pi_phase )

                // however, in the product the pulses are equal
                        ss_rf_dummy.setThickness( 40 );
                        if( !(ss_rf_dummy.prepExternal( pMrProt, pSeqExpo ) ) )
                                  return ss_rf_dummy.getNLSStatus();

                        #ifdef _DVP_DEBUG
                                fprintf( stdout, "\nfSEQPrep(): flip angle of dummy PI
pulse at %d pulse duration: %f deg.\n\n",
                                        ss_rf_dummy.getDuration(),
ss_rf_dummy.getActualFlipAngle() );
                        #endif


                        long dur;
                        double angle;
                        if( (angle = ss_rf_dummy.getActualFlipAngle()) > 0 ){

                        dur = (int)(.5 + 2600 * 180. / angle); // [dur] = us
                                dur = ( dur%200 ) ? (1+dur/200)*200 : dur; // rounding
up to a multiple of 200 us
                                if( dur > MAX_RF_PI_PULSE_DURATION )        //
upper limit pulse duration to avoid excessive displacement
                                        dur = MAX_RF_PI_PULSE_DURATION;
        // due to chemical shift
                        }
                        else{

                                dur = MAX_RF_PI_PULSE_DURATION;
                                TRACE_PUT2(TC_INFO, TF_SEQ, "NEVER
HAPPEN Warning from %s:"
                                                    "\nduration of PI pulse could not be
determined;"
                                                                          "\nthe default
duration of %d us is used.", ptModule, dur );
                                #ifdef _DVP_DEBUG
                                        fprintf( stdout, "\nfSEQPrep(): NEVER
HAPPEN Warning:"
                                                    "\nduration of PI pulse could not be
determined;"
                                                                          "\nthe default
duration of %d us is used.", dur );
                                #endif
                        }
```

```
                                pMrProt->utilityParameter()[_UtilPar_180dumpul] = dur;
                                // pMrProt->utilityParameter()[_UtilPar_180dumpul] = 3200;
// UT pulse duration
            }
            #endif


            #ifdef _OWN_DEBUG01
                cout << "pMrProt->utilityParameter()[_UtilPar_180dumpul] 180degree
pulse =" << pMrProt->utilityParameter()[_UtilPar_180dumpul] << endl;

            #endif



  ss_rf_pi_sl.setTypeRefocussing();
  ss_rf_pi_sl.setDuration( (alpha_pi_sl > 90) ? pMrProt-
>utilityParameter()[_UtilPar_180dumpul] : 2600 ) ;
  ss_rf_pi_sl.setFlipAngle( alpha_pi_sl );
  ss_rf_pi_sl.setInitialPhase( 0 );
  ss_rf_pi_sl.setFamilyName( name_pi_sl );
  ss_rf_pi_sl.setThickness( pMrProt->spectroscopy().VoI().thickness() ); /*! EGA-04;
EGA-02; EGA-05 !*/

  if( !( ss_rf_pi_sl.prepExternal( pMrProt, pSeqExpo ) ) )
      return ss_rf_pi_sl.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 + ss_rf_pi_sl.getGSAmplitude() *
                larmorconst * ss_voi.getSliceShift() );  /*! EGA-05 !*/
  // correction of chemical shift displacement artefact
  lFrequency += (long)(pMrProt->txSpec().frequency() /* in Hz */ * 1E-6 * pMrProt-
>spectroscopy().dDeltaFrequency) /* in ppm */;  /*! EGA-05 !*/
  ss_ph_s_pi_sl.setFrequency( lFrequency );                          /*! EGA-05 !*/
  ss_ph_n_pi_sl.setFrequency( 0L );

  // computation of start-phase for off-resonant pulses as in
  // \n4\comp\measurement\sequence\libRT\sFREQ_PHASE::prepSet()
  // pulse asymmetry: 0.5 (== symmetric pulse)
  // input units:
  // [Frequency] = Hz
  // [pulse duration] = us
  // output unit:
  // [Phase] = deg.

  // the frequency (and pulse) dependent phase portion
  ss_ph_s_pi_sl.setPhase( - lFrequency * (360./1e6) *
                                                        ss_rf_pi_sl.getDuration() *
ss_rf_pi_sl.getAsymmetry() /* = .5 */ );
  ss_ph_n_pi_sl.setPhase(  - lFrequency * (360./1e6) *
                                                        ss_rf_pi_sl.getDuration() * (1. -
ss_rf_pi_sl.getAsymmetry()) /* = .5 */  );
  // add 90 deg.s to maintain CPMG condition
  ss_ph_s_pi_sl.increasePhase( 90. );
  ss_ph_n_pi_sl.decreasePhase( 90. );

  #ifdef _DVP_DEBUG
    fprintf( stdout, "\nfSEQPrep(): READOUT direction properties"
                "\nactual flip angle %f deg. \nactual pulse duration %f"
                "\nVoI size %f mm \ngradient strength %f mT/m"
                "\npos. offset %f mm \n frequ. offset %d Hz \n\n",
                (float)(ss_rf_pi_sl.getActualFlipAngle()),
                (float)(ss_rf_pi_sl.getDuration()),
                (float)(ss_rf_pi_sl.getThickness()),
                (float)(ss_rf_pi_sl.getGSAmplitude()),
                (float)(ss_voi.getSliceShift()),
                (int)lFrequency );
  #endif

  ///////////////////////////////////////////////////////////////////////////////////////
  // refocussing pulse (PHASE direction)
  ///////////////////////////////////////////////////////////////////////////////////////

  ss_rf_pi_ph.setTypeRefocussing();
  ss_rf_pi_ph.setDuration( (alpha_pi_ph > 90) ? pMrProt-
>utilityParameter()[_UtilPar_180dumpul] : 2600 );
  ss_rf_pi_ph.setFlipAngle( alpha_pi_ph );
  ss_rf_pi_ph.setInitialPhase( 0 );
  ss_rf_pi_ph.setFamilyName( name_pi_ph );
  ss_rf_pi_ph.setThickness( pMrProt->spectroscopy().VoI().phaseFOV() ); /*! EGA-
04; EGA-02; EGA-05 !*/

  if( !( ss_rf_pi_ph.prepExternal( pMrProt, pSeqExpo ) ) )
```

```
     return ss_rf_pi_ph.getNLSStatus();

   // reduction of the slice gradients for VoI excitation
   dGSAmplitudereduction_factor_pi = 0.78 * (1 - 2 * (1.85*(pMrProt-
>txSpec().frequency()*1E-6)/(6.0/(ss_rf_pi_ph.getDuration()*1E-6))));  //with
empirical factor 0.78 to compensate the slice profile
         d_rf_pi_ph_GSAmplitude = dGSAmplitudereduction_factor_pi *
ss_rf_pi_ph.getGSAmplitude();

         // for the calcultion of the min FOV in case fully excited VoI = On
   // FoVmin = 1.1/dGSAmplitudereduction_factor_pi * VoI(UI) =
fullyexcitedVoiFOVmin_factor * VoI(UI)
         // this parameter is needed in MRUILinkSpecCSI.cpp for the tool tip
         pMrProt->utilityParameter()[_UtilPar_FullyExcitedVoiFoVmin_factor] =
(long)(0.5+(1.1*1000/dGSAmplitudereduction_factor_pi));  // *1000 to keep past
comma digits



   #ifdef _OWN_DEBUG_01
         cout <<  endl;
         cout << "pMrProt-
>utilityParameter()[_UtilPar_FullyExcitedVoiFoVmin_factor] =" << pMrProt-
>utilityParameter()[_UtilPar_FullyExcitedVoiFoVmin_factor] << endl;

   cout << "min FOV_ro                                          :" << pMrProt-
>utilityParameter()[_UtilPar_FullyExcitedVoiFoVmin_factor]/1000.*ss_rf_exc.getT
hickness()   << endl;
         cout << "min FOV_ph                                          :" << pMrProt-
>utilityParameter()[_UtilPar_FullyExcitedVoiFoVmin_factor]/1000.*ss_rf_pi_ph.get
Thickness() << endl;
         cout <<  endl;
   #endif



         #ifdef _DVP_DEBUG_UT
   cout << "   ---------------- Calc. of fully excited VoI Slice Grad (PH / 180)   ----------
--   "              << endl;
   cout << "   ss_rf_pi_ph.getDuration():                " << ss_rf_pi_ph.getDuration()
<< endl;
   //cout << "==>Frequency:                              " << pMrProt-
>txSpec().frequency() << endl;
   cout << "==>Bandwidth of pulse:                     " <<
6.0/(ss_rf_pi_ph.getDuration()*1E-6) << endl;
   cout << "==>alt: ss_rf_pi_ph.getGSAmplitude():         " <<
ss_rf_pi_ph.getGSAmplitude() << endl;
```

```
   cout << "==>neu: d_rf_pi_ph_GSAmplitude:                " <<
d_rf_pi_ph_GSAmplitude << endl;
   cout << "==>dGSAmplitudereduction_factor_pi:            " <<
dGSAmplitudereduction_factor_pi << endl;
   //cout << "==>dGSAmplitudereduction_factor_pi (worst case): " << (0.78*(1 - 2 *
(1.85*123/(6.0/(MAX_RF_PI_PULSE_DURATION*1E-6))))) << endl;
   //cout << "==>Gneu/Galt                                  " <<
(d_rf_pi_ph_GSAmplitude/ss_rf_pi_ph.getGSAmplitude()) << endl;

         /*
         cout << "## VOIalt A >> P    (refoc.)                 =" <<
(6.0/(ss_rf_pi_ph.getDuration()*1E-6))
/(ss_rf_pi_ph.getGSAmplitude()*larmorconst) << endl;
   cout << "## VOIneu A >> P                           =" <<
(6.0/(ss_rf_pi_ph.getDuration()*1E-6)) /(d_rf_pi_ph_GSAmplitude *larmorconst) <<
endl;
   cout << "## zu saettigen min A >> P                  =" <<
((6.0/(ss_rf_pi_ph.getDuration()*1E-6))/d_rf_pi_ph_GSAmplitude -

                       (6.0/(ss_rf_pi_ph.getDuration()*1E-
6))/ss_rf_pi_ph.getGSAmplitude()) /larmorconst  << endl;
   cout << "## VOIneu max  A >> P   63Mhz                =" <<  160.0/(0.78*(1 - 2 *
(1.85*63 /(6.0/(6000*1E-6)))))   << endl;
         cout << "## VOIneu max  A >> P  123Mhz                =" <<  160.0/(0.78*(1 -
2 * (1.85*123/(6.0/(6000*1E-6)))))   << endl;
         cout << "## Saettiger 160mm  VOI u. Pulsl=6000; 123MHz =" <<
160.0/(0.78*(1 - 2 * (1.85*123/(6.0/(6000*1E-6)))))    -    160.0  << endl ;
   */
         cout << endl;
         #endif




  // 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

   if (!(pMrProt->spectroscopy().outerVolumeSuppression()) )
         {         // cout << " ++++++++   fully_excited_VOI = OFF  , lfrequency offset
PE    ++++++++  "  << endl;
```

```
                lFrequency = (long)( .5 + ss_rf_pi_ph.getGSAmplitude() *
larmorconst * ss_voi.getSliceOffCenterPE() );  /*! EGA-05 !*/
        }
        else
        {  // cout << " ++++++++   fully_excited_VOI = ON  , lfrequency offset PE
++++++++  "  << endl;
                lFrequency = (long)( .5 + d_rf_pi_ph_GSAmplitude   *  larmorconst *
ss_voi.getSliceOffCenterPE() );   /*! EGA-05 !*/
             // cout << "## lFrequency offset VOI new          ="  << lFrequency
                                << endl;

        }

  // correction of chemical shift displacement artefact
  lFrequency += (long)(pMrProt->txSpec().frequency() /* in Hz */ * 1E-6 * pMrProt-
>spectroscopy().dDeltaFrequency) /* in ppm */;  /*! EGA-05 !*/
  ss_ph_s_pi_ph.setFrequency( lFrequency );                              /*! EGA-05
!*/
  ss_ph_n_pi_ph.setFrequency( 0L );




  #ifdef _DVP_DEBUG
        cout << " ------------------------- frequency shift VOI in phase direction --------
----------------"<< endl;
        cout << "## larmorconst                          ="  <<  larmorconst
                                                << endl;
        cout << "## pMrProt->txSpec().frequency()          ="  <<  pMrProt-
>txSpec().frequency()                         << endl;
        cout << "## pMrProt->spectroscopy().dDeltaFrequency ="  <<  pMrProt-
>spectroscopy().dDeltaFrequency  << endl;
        cout << "## ss_voi.getSliceOffCenterPE()                         ="  <<
ss_voi.getSliceOffCenterPE()            << endl;
    cout << "## Grundoffset fully VOI                 ="  << (long)( .5 +
d_rf_pi_ph_GSAmplitude *  larmorconst * ss_voi.getSliceOffCenterPE()) << endl ;
    cout << "## Grundoffset normal VOI                 ="  << (long)( .5 +
ss_rf_pi_ph.getGSAmplitude()*  larmorconst * ss_voi.getSliceOffCenterPE()) <<
endl ;
        cout << "## lFrequency                           ="  << lFrequency
                        << endl;
        cout << "## d_rf_pi_ph_GSAmplitude                                   ="  <<
d_rf_pi_ph_GSAmplitude  << endl;
    cout << "d_rf_pi_ph_GSAmplitude*gamma                               ="  <<
d_rf_pi_ph_GSAmplitude*  larmorconst << endl;
```

```
        cout << "==>ss_ph_s_pi_ph.getFrequency():        ="  <<
ss_ph_s_pi_ph.getFrequency() << endl;
    cout << endl;
  #endif




  // computation of start-phase for off-resonant pulses as in
  // \n4\comp\measurement\sequence\libRT\sFREQ_PHASE::prepSet()
  // pulse asymmetry: 0.5 (== symmetric pulse)
  // input units:
  // [Frequency] = Hz
  // [pulse duration] = us
  // output unit:
  // [Phase] = deg.

  // the frequency (and pulse) dependent phase portion
  ss_ph_s_pi_ph.setPhase( - lFrequency * (360./1e6) *
                                              ss_rf_pi_ph.getDuration() *
ss_rf_pi_ph.getAsymmetry() /* = .5 */ );
  ss_ph_n_pi_ph.setPhase(  - lFrequency * (360./1e6) *
                                              ss_rf_pi_ph.getDuration() * (1. -
ss_rf_pi_ph.getAsymmetry()) /* = .5 */  );
  // add 90 deg.s to maintain CPMG condition
  ss_ph_s_pi_ph.increasePhase( 90. );
  ss_ph_n_pi_ph.decreasePhase( 90. );


  #ifdef _DVP_DEBUG
    fprintf( stdout, "\nfSEQPrep(): PHASE direction properties"
             "\nactual flip angle %f deg. \nactual pulse duration %f"
             "\nVoI size %f mm \ngradient strength normal VoI %f mT/m"
             "\npos. offset %f mm \n frequ. offset %d Hz \n\n",
             (float)(ss_rf_pi_ph.getActualFlipAngle()),
             (float)(ss_rf_pi_ph.getDuration()),
             (float)(ss_rf_pi_ph.getThickness()),
                                  //(float)(d_rf_pi_ph_GSAmplitude),
             (float)(ss_rf_pi_ph.getGSAmplitude()),
             (float)(ss_voi.getSliceOffCenterPE()),
             (int)lFrequency );
  #endif

  // water suppression
  // ... as described by Ogg et al.,JMR, B 104, p. 1-10, 1994.
  wsatpulsedur = (long)(25600 * 35. / pMrProt-
>spectroscopy().RFExcitationBandwidth()); // all water suppression pulses are of
equal duration
  wsatpulsedur = ( wsatpulsedur%100 ) ? (wsatpulsedur/100)*100 : wsatpulsedur; //
rounding down to a multiple of 100 us
  // with VA15A, the bandwidth of the pulses is NOT any more scaled using the
empirical PulseWidthTimeDomain factor
  // which is kept constant at 35 (the VA12B default value),
  // but by scaling the pulse duration
  // given that the extreme values of the excitation bandwidth are 20 ... 60,
pulsedurations range from 44800 to 14800 us
  gauss_width_td = 35.;


  ss_rf_ws1.setTypeExcitation();
  ss_rf_ws1.setDuration( wsatpulsedur ) ; // in us
  ss_rf_ws1.setSamples( wsatpulsedur/100 );
  ss_rf_ws1.setFlipAngle( 89.2 );   // Ogg: 89.2
  ss_rf_ws1.setInitialPhase( 0.0 ); // not used
  ss_rf_ws1.setThickness( 10.0 );   // not used
  ss_rf_ws1.setFlipAngleCorrection();
  if( ! ( ss_rf_ws1.prepGauss( pMrProt, pSeqExpo, gauss_width_td ))){
    TRACE_PUT1(TC_INFO, TF_SEQ, "\nfSEQPrep():  %s\n",
              "can't prepare water suppression pulse" );
    return ss_rf_ws1.getNLSStatus();
  }

  ss_rf_ws2.setTypeExcitation();
  ss_rf_ws2.setDuration( wsatpulsedur ) ; // in us
  ss_rf_ws2.setSamples( wsatpulsedur/100 );
  ss_rf_ws2.setFlipAngle( 83.4 );   // Ogg: 83.4
  ss_rf_ws2.setInitialPhase( 0.0 ); // not used
  ss_rf_ws2.setThickness( 10.0 );   // not used
  ss_rf_ws2.setFlipAngleCorrection();
  if( ! ( ss_rf_ws2.prepGauss( pMrProt, pSeqExpo, gauss_width_td ))){
    TRACE_PUT1(TC_INFO, TF_SEQ, "\nfSEQPrep():  %s\n",
              "can't prepare water suppression pulse" );
    return ss_rf_ws2.getNLSStatus();
  }

  ss_rf_ws3.setTypeExcitation();
  ss_rf_ws3.setDuration( wsatpulsedur ) ; // in us
  ss_rf_ws3.setSamples( wsatpulsedur/100 );
  ss_rf_ws3.setFlipAngle( 160.8 );
  ss_rf_ws3.setInitialPhase( 0.0 ); // not used
  ss_rf_ws3.setThickness( 10.0 );   // not used
```

```
 ss_rf_ws3.setFlipAngleCorrection();
 if( ! ( ss_rf_ws3.prepGauss( pMrProt, pSeqExpo, gauss_width_td ))){
    TRACE_PUT1(TC_INFO, TF_SEQ, "\nfSEQPrep():  %s\n",
            "can't prepare water suppression pulse" );
    return ss_rf_ws3.getNLSStatus();
 }





        // #----------------------------------------------------------------------#
  // #
                                                            #
  // #                              MEGA for lipid AND water suppression
                          #
        // #
                                                                      #
  // #----------------------------------------------------------------------#


        const double MEGABwTimeProduct = 2521600.;  // near 25600us * 98.5 Hz
empiric factor measured with sequence pulseprofile
        const long no_mega_samples = 512;
        long mega_pulse_dur = 25600,  mega_pulse_dur1 = 25600,  mega_pulse_dur2
= 25600;  // 25.6 ms
        float mega_arr1[no_mega_samples*2];
        long i, delta_frequ1,delta_frequ2,delta_frequ = 0L;



        // mega for lipids and water

if( pMrProt->spectroscopy().SpectralSuppression() ==
SEQ::SPEC_SUPPR_LIPID_WATER  )
        {

          for( i=0; i<no_mega_samples*2; i++ )
                        mega_arr1[i] = 0.0f;



     // #          first suppression frequency offset
```

```
                mega_pulse_dur1 = (long)(0.5 + MEGABwTimeProduct/(pMrProt-
>spectroscopy().SpecLipidSupprBandwidth()* pMrProt->txSpec().frequency() * 1E-
6) );
                mega_pulse_dur1 = ( mega_pulse_dur1%(no_mega_samples/2) ) ?
(1+mega_pulse_dur1/(no_mega_samples/2))*(no_mega_samples/2) :
mega_pulse_dur1; // rounding up to a multiple of 256 us (each sample point must
have .5us)
                delta_frequ1 = (long)(.5 - pMrProt-
>spectroscopy().SpecLipidSupprDeltaPos() * pMrProt->txSpec().frequency() * 1E-6
);


      // #      second suppression frequency offset
                mega_pulse_dur2 = (long)(0.5 + MEGABwTimeProduct/(pMrProt-
>spectroscopy().SpecWaterSupprBandwidth() * pMrProt->txSpec().frequency() *
1E-6) );
                mega_pulse_dur2 = ( mega_pulse_dur2%(no_mega_samples/2) ) ?
(1+mega_pulse_dur2/(no_mega_samples/2))*(no_mega_samples/2) :
mega_pulse_dur2; // rounding up to a multiple of 256 us (each sample point must
have .5us)
                delta_frequ2 = (long)(.5 - pMrProt-
>spectroscopy().SpecWaterSupprDeltaPos() * pMrProt->txSpec().frequency() * 1E-6
);




        // in case of flip anngle as UI parameter  use the following code: fraction of
second flip angle devided through the first flip angle
        double attenuation_mixedpul = ( 180. ) / ( 180. ); // att. factor of h20 flip angle
/ lipid flip angle
        double mega_ampl1, mega_ampl2;
        mega_ampl1 = add_refoc( delta_frequ1,  mega_pulse_dur1, 1.0, mega_arr1 );
// first pulse shape
        mega_ampl2 = add_refoc( delta_frequ2,  mega_pulse_dur2,
attenuation_mixedpul, mega_arr1 ); // second pulse shape
        //cout << "attenuation_mixedpul "  << attenuation_mixedpul << endl;
        //cout << "mega_ampl1 von mixed "  << mega_ampl1 << endl ;
   //cout << "mega_ampl2 von mixed "  << mega_ampl2 << endl ;



                // mega_arr convertion
        arr2sample_arr( no_mega_samples , mega_arr1, ss_mega_samples );
   ss_rf_mega1.setTypeUndefined();
   ss_rf_mega1.setDuration( mega_pulse_dur1 ) ; // in us
   ss_rf_mega1.setSamples( no_mega_samples );
   ss_rf_mega1.setFlipAngle( 270.);                    // for mixed pulses: 180 degree is
realised with 270
   ss_rf_mega1.setInitialPhase( 0.0 );                 // not used
   ss_rf_mega1.setThickness( 10.0 );                   // not used
   ss_rf_mega1.setFlipAngleCorrection();



         if( !( ss_rf_mega1.prepArbitrary( pMrProt, pSeqExpo, ss_mega_samples,
mega_ampl2) ) ){
     TRACE_PUT1(TC_INFO, TF_SEQ, "\nfSEQPrep():  %s\n",
             "can't prepare mega suppression pulse, increase TE, TR (>XX ms)" );
     return ss_rf_mega1.getNLSStatus();
         }



         ss_rf_mega2.setTypeUndefined();
   ss_rf_mega2.setDuration( mega_pulse_dur1 ) ; // the pulse duration calculated with
the lipid suppr BW; for mixed pulses, the BWs for lipid and water are coupled in th
UI
   ss_rf_mega2.setSamples( no_mega_samples );
   ss_rf_mega2.setFlipAngle( 270.  );                  //
   ss_rf_mega2.setInitialPhase( 0.0 );                 // not used
   ss_rf_mega2.setThickness( 10.0 );                   // not used
   ss_rf_mega2.setFlipAngleCorrection();


         if( !( ss_rf_mega2.prepArbitrary( pMrProt, pSeqExpo, ss_mega_samples,
mega_ampl2 ) ) ){
         //if( !( ss_rf_mega2.prepExternal( pMrProt, pSeqExpo ) ) ){
     TRACE_PUT1(TC_INFO, TF_SEQ, "\nfSEQPrep():  %s\n",
             "can't prepare mega suppression pulse, increase TE, TR (>XX ms)" );
     return ss_rf_mega2.getNLSStatus();
         }


   #ifdef  _OWN_DEBUG_SpecSupp
           cout << "@@@@@@@@@@@@@@@@@@@@@@@@
SPEC_SUPPR_LIPID_WATER
@@@@@@@@@@@@@@@@@@@@@@@@"<< endl;
```

```
            cout << "MEGA1: pMrProt->spectroscopy().SpecLipidSupprDeltaPos()  "<<
pMrProt->spectroscopy().SpecLipidSupprDeltaPos() << endl;
            cout << "MEGA2: pMrProt->spectroscopy().SpecWaterSupprDeltaPos()
"<< pMrProt->spectroscopy().SpecWaterSupprDeltaPos()  << endl;

            cout << "MEGA1: pMrProt-
>spectroscopy().SpecLipidSupprBandwidth()"<< pMrProt-
>spectroscopy().SpecLipidSupprBandwidth() << endl;
            cout << "MEGA2: pMrProt-
>spectroscopy().SpecWaterSupprBandwidth()"<< pMrProt-
>spectroscopy().SpecWaterSupprBandwidth() << endl;
      cout << "must be the same value !!!!"<< endl;

      cout << "mega_pulse_dur1       =    "<<  mega_pulse_dur1    << endl;
            cout << "delta_frequ1         =    "<<  delta_frequ1    << endl;

            cout << "mega_pulse_dur2       =    "<<  mega_pulse_dur2    << endl;
            cout << "delta_frequ2         =    "<<  delta_frequ2    << endl;



            cout <<
"@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@"<< endl;
          #endif






}



 else { // only if NOT  (MEAG1 == ON && MEGA2 == ON)
                    // either lipid oder water suppression will be executed

           if(    (pMrProt->spectroscopy().SpectralSuppression() ==
SEQ::SPEC_SUPPR_LIPID)
               ||(pMrProt->spectroscopy().SpectralSuppression() ==
SEQ::SPEC_SUPPR_WATER) )

           {
                    #ifdef  _OWN_DEBUG_SpecSupp
            cout << "@@@@@@@@@@@@@@@    SEQ::SPEC_SUPPR_LIPID
or SEQ::SPEC_SUPPR_WATER   @@@@@@@@@@@@@"<< endl;
```

```
            cout << "MEGA1: pMrProt->spectroscopy().SpecLipidSupprDeltaPos()  "<<
pMrProt->spectroscopy().SpecLipidSupprDeltaPos() << endl;
            cout << "MEGA2: pMrProt->spectroscopy().SpecWaterSupprDeltaPos()
"<< pMrProt->spectroscopy().SpecWaterSupprDeltaPos()  << endl;

            cout << "MEGA1: pMrProt-
>spectroscopy().SpecLipidSupprBandwidth()"<< pMrProt-
>spectroscopy().SpecLipidSupprBandwidth() << endl;
            cout << "MEGA2: pMrProt-
>spectroscopy().SpecWaterSupprBandwidth()"<< pMrProt-
>spectroscopy().SpecWaterSupprBandwidth() << endl;
     cout << "must not be the same value !!!!"<< endl;
            cout <<
"@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@"<< endl;
                     #endif



                    for( i=0; i<no_mega_samples*2; i++ )
                              mega_arr1[i] = 0.0f;





         // #----------------------------------------------------------------------#
   // #
                                                                  #
   // #                              MEGA for lipid suppression
                                     #
         // #
                                                                            #
   // #----------------------------------------------------------------------#

                              // MEGA pulse prep
                     if( (pMrProt->spectroscopy().SpectralSuppression() ==
SEQ::SPEC_SUPPR_LIPID) )
                     {

                              mega_pulse_dur = (long)(0.5 +
MEGABwTimeProduct/(pMrProt->spectroscopy().SpecLipidSupprBandwidth()*
pMrProt->txSpec().frequency() * 1E-6) );
                        mega_pulse_dur = ( mega_pulse_dur%(no_mega_samples/2) ) ?
(1+mega_pulse_dur/(no_mega_samples/2))*(no_mega_samples/2) : mega_pulse_dur;
// rounding up to a multiple of 256 us (each sample point must have .5us)
```

```
                        delta_frequ = (long)(0.5 - pMrProt-
>spectroscopy().SpecLipidSupprDeltaPos() * pMrProt->txSpec().frequency() * 1E-6
);
                        #ifdef  _OWN_DEBUG_SpecSupp
                        cout << "mega_pulse_dur_lipid     =    "<<  mega_pulse_dur
<< endl;
                        cout << "delta_frequ_lipid       =    "<<  delta_frequ       <<
endl;
                        #endif
                 }



   // #----------------------------------------------------------------------#
   // #
                                                        #
   // #                             MEGA for water suppression
                                                #
        // #
                                                                #
   // #----------------------------------------------------------------------#

                        // MEGA pulse prep
                 if( (pMrProt->spectroscopy().SpectralSuppression() ==
SEQ::SPEC_SUPPR_WATER) )
                 {

                        mega_pulse_dur = (long)(0.5 +
MEGABwTimeProduct/(pMrProt->spectroscopy().SpecWaterSupprBandwidth() *
pMrProt->txSpec().frequency() * 1E-6) ) ;
                  mega_pulse_dur = ( mega_pulse_dur%(no_mega_samples/2) ) ?
(1+mega_pulse_dur/(no_mega_samples/2))*(no_mega_samples/2) : mega_pulse_dur;
// rounding up to a multiple of 256 us (each sample point must have .5us)

                        delta_frequ = (long)(0.5 - pMrProt-
>spectroscopy().SpecWaterSupprDeltaPos()  * pMrProt->txSpec().frequency() * 1E-
6 );
                        #ifdef  _OWN_DEBUG_SpecSupp
                        cout << "mega_pulse_dur_water     =    "<<  mega_pulse_dur
<< endl;
                        cout << "delta_frequ_water       =    "<<  delta_frequ       <<
endl;
                        #endif
                 }
   double mega_ampl1;
         mega_ampl1 = add_refoc( delta_frequ,  mega_pulse_dur, 1.0, mega_arr1 ); //
mega for lipids


                  // mega_arr convertion
         arr2sample_arr( no_mega_samples , mega_arr1, ss_mega_samples);



   ss_rf_mega1.setTypeUndefined();
   ss_rf_mega1.setDuration( mega_pulse_dur );  // in us
   ss_rf_mega1.setSamples( no_mega_samples );
         ss_rf_mega1.setFlipAngle( 180. );
         ss_rf_mega1.setInitialPhase( 0.0 );                     // not used
   ss_rf_mega1.setThickness( 10.0 );                  // not used
   ss_rf_mega1.setFlipAngleCorrection();


         if( !( ss_rf_mega1.prepArbitrary( pMrProt, pSeqExpo, ss_mega_samples,
mega_ampl1) ) ){
    TRACE_PUT1(TC_INFO, TF_SEQ, "\nfSEQPrep():  %s\n",
            "can't prepare mega suppression pulse, increase TE (>XX ms)" );
    return ss_rf_mega1.getNLSStatus();
         }




         ss_rf_mega2.setTypeUndefined();
   ss_rf_mega2.setDuration( mega_pulse_dur );       // in us
   ss_rf_mega2.setSamples( no_mega_samples );
         ss_rf_mega2.setFlipAngle( 180. );
   ss_rf_mega2.setInitialPhase( 0.0 );                // not used
   ss_rf_mega2.setThickness( 10.0 );                  // not used
   ss_rf_mega2.setFlipAngleCorrection();


         if( !( ss_rf_mega2.prepArbitrary( pMrProt, pSeqExpo, ss_mega_samples,
mega_ampl1 ) ) ){
    TRACE_PUT1(TC_INFO, TF_SEQ, "\nfSEQPrep():  %s\n",
            "can't prepare mega suppression pulse, increase TE (>XX ms)" );
    return ss_rf_mega2.getNLSStatus();
```

```
          }
  }

} // endif spectral suppression pulses




// phase for MAGA pulses:
  ss_ph_s_ws.setFrequency( 0L );
  ss_ph_n_ws.setFrequency( 0L );
  ss_ph_s_ws.setPhase( 0 );
  ss_ph_n_ws.setPhase( 0 );


  //////////////////////////////////////////////////////////////////////
  // compute VoI- and FoV-dependance for ovs-sequence with reduced gradients
  // for in-plane VoI excitation
  //////////////////////////////////////////////////////////////////////


//if (pMrProt->spectroscopy().outerVolumeSuppression())    //
//         {  // cout << " ++++++++   fully_excited_VOI = ON  , calc VoI- and FoV-
dependance  ++++++++  "  << endl;



            // the min FOV must have he size of the excited region of the new VoI, to
make shure, that there is no overfolding in the Spectro VOI
            // these constraints are not realized to prevent protocol-problems; a tool tip
informs about the suggested min FoV

            dsat_readoutFOV = ( (2*  (1 / dGSAmplitudereduction_factor_pi) ) - 1) *
ss_rf_exc.getThickness();
            dsat_phaseFOV   = ( (2 * (1 / dGSAmplitudereduction_factor_pi) ) - 1) *
ss_rf_pi_ph.getThickness();




            #ifdef _OWN_DEBUG_01
                    cout << "==========================================" <<
endl;
                    cout << "==>dGSAmplitudereduction_factor_pi:                         " <<
dGSAmplitudereduction_factor_pi << endl;
```

```
			cout << "==>((2 * (1 / dGSAmplitudereduction_factor_pi)) - 1):
" << ((2 * (1 / dGSAmplitudereduction_factor_pi)) - 1) << endl;
			cout << "==>(1 + (0.6 * ((1 / dGSAmplitudereduction_factor_pi) -
1))):  " << (1 + (0.6 * ((1 / dGSAmplitudereduction_factor_pi) - 1))) << endl;
			cout << "==>(1 + (2*0.55 * ((1 / dGSAmplitudereduction_factor_pi) -
1))):  " << (1 + (2*0.55 * ((1 / dGSAmplitudereduction_factor_pi) - 1))) << endl;
			cout << "==>phVOI:                                   " <<
ss_rf_pi_ph.getThickness() << endl;
			cout << "==>phFOV:                                   " << pMrProt-
>sliceSeries().front().phaseFOV() << endl;
			cout << "==>(2f-1) * VOI:                            " << (((2 * (1 /
dGSAmplitudereduction_factor_pi)) - 1) * ss_rf_exc.getThickness()) << endl;
			cout << "==>(1+2*0.55(f-1))*VOI :                    " << ((1 +
(2 * 0.55 * ((1 / dGSAmplitudereduction_factor_pi) - 1))) *
ss_rf_pi_ph.getThickness()) << endl;
			cout << "===========================================" <<
endl;
		#endif

//		}




  //////////////////////////////////////////////////////////////////////////////////////
  //  Prepare the readout frequency/phase event
  //////////////////////////////////////////////////////////////////////////////////////
//Elijah
/////
  ss_ph_s_adc1.setFrequency( 0L );
  ss_ph_n_adc1.setFrequency( 0L );
  ss_ph_s_adc1.setPhase( 0 );
  ss_ph_n_adc1.setPhase( 0 );

  ss_ph_s_adc2.setFrequency( 0L );
  ss_ph_n_adc2.setFrequency( 0L );
  ss_ph_s_adc2.setPhase( 0 );
  ss_ph_n_adc2.setPhase( 0 );
  /////

  //////////////////////////////////////////////////////////////////////////////////////
  // compute grid position
  //////////////////////////////////////////////////////////////////////////////////////

  sd_read_pos = ss_fov.getSliceOffCenterRO();
  sd_phase_pos = ss_fov.getSliceOffCenterPE();
  sd_slice_pos = ss_fov.getSliceShift();

  if( !( (n = pMrProt->spectroscopy().finalMatrixSizeRead()) % 2 ) ) // this is
currently always the case
            sd_read_pos += voxelshift_read * pMrProt-
>sliceSeries().front().readoutFOV() / (double) n;

  if( !( (n = pMrProt->spectroscopy().finalMatrixSizePhase()) % 2 ) ) // this is
currently always the case
            sd_phase_pos += voxelshift_phase * pMrProt-
>sliceSeries().front().phaseFOV() / (double) n;

  if( !( (n = pMrProt->spectroscopy().finalMatrixSizeSlice()) % 2 ) ) // this is currently
always the case
            sd_slice_pos += voxelshift_slice * pMrProt-
>sliceSeries().front().thickness() / (double) n;


  #ifdef _DVP_DEBUG
      fprintf( stdout, "voxelshift read %f phase %f slice %f\n voxelposition read %f
phase %f slice %f\n",
                              voxelshift_read, voxelshift_phase,
voxelshift_slice,
                              sd_read_pos, sd_phase_pos, sd_slice_pos );

  #endif


#ifdef _OWN_DEBUG_02
  cout << "========================================" << endl;
  cout << "===========Positioning===================" << endl;
  cout << "==>ss_fov.getSliceShift:       " << ss_fov.getSliceShift() << endl;
  cout << "==>ss_rf_pi_sl.getThickness:   " << ss_rf_pi_sl.getThickness() << endl;
  cout << " " << endl;

  cout << "==>ss_fov.getSliceOffCenterRO: " << ss_fov.getSliceOffCenterRO() <<
endl;
  cout << "==>readoutFOV:                " << pMrProt-
>sliceSeries().front().readoutFOV() << endl;
  cout << "==>ss_rf_exc.getThickness:     " << ss_rf_exc.getThickness() << endl;
  cout << "==>rsat1_pos_RO:              " << ss_fov.getSliceOffCenterRO() +
(ss_rf_exc.getThickness()/2) + ((pMrProt->sliceSeries().front().readoutFOV()/2) -
(ss_rf_exc.getThickness()/2))/2 << endl;
  cout << "==>rsat2_pos_RO:              " << ss_fov.getSliceOffCenterRO() -
(ss_rf_exc.getThickness()/2) - ((pMrProt->sliceSeries().front().readoutFOV()/2) -
(ss_rf_exc.getThickness()/2))/2 << endl;
```

```
  cout << "==>rsati_RO_thickness:         " << ((pMrProt-
>sliceSeries().front().readoutFOV()/2) - (ss_rf_exc.getThickness()/2)) << endl;

  cout << " " << endl;
  cout << "==>ss_fov.getSliceOffCenterPE: " << ss_fov.getSliceOffCenterPE() <<
endl;
  cout << "==>phaseFOV  :                " << pMrProt-
>sliceSeries().front().phaseFOV() << endl;
  cout << "==>ss_rf_pi_ph.getThickness:   " << ss_rf_pi_ph.getThickness() << endl;
  cout << "==>rsat1_pos_PH:              " << ss_fov.getSliceOffCenterPE() +
(ss_rf_pi_ph.getThickness()/2) + ((pMrProt->sliceSeries().front().phaseFOV()/2) -
(ss_rf_pi_ph.getThickness()/2))/2 << endl;
  cout << "==>rsat2_pos_PH:              " << ss_fov.getSliceOffCenterPE() -
(ss_rf_pi_ph.getThickness()/2) - ((pMrProt->sliceSeries().front().phaseFOV()/2) -
(ss_rf_pi_ph.getThickness()/2))/2 << endl;
  cout << "==>rsati_PH_thickness:        " << ((pMrProt-
>sliceSeries().front().phaseFOV()/2) - (ss_rf_pi_ph.getThickness()/2)) << endl;


  cout << "===========Positioning==================" << endl;
  cout << "========================================" << endl;
#endif




  //////////////////////////////////////////////////////////////////////////////////////
  // Prepare the gradient pulse structures
  //////////////////////////////////////////////////////////////////////////////////////

  // gradient during excitation

  if (!(pMrProt->spectroscopy().outerVolumeSuppression() ))
  {
          //cout << " ++++++++   fully_excited_VOI = OFF , excited VOI  ++++++++
"  << endl;
          if( !( ss_grad_exc.prepAmplitude( ramptime,                          /* ramp-up
time */
                              ramptime + 1000 + ss_rf_exc.getDuration(), /* duration =
ramp-up time + flat-top */
                              ramptime,                           /* ramp-down time */
                              ss_rf_exc.getGSAmplitude() ) ) ||        /*! EGA-04; EGA-02
!*/
      !(  ss_grad_exc.check() ) )
       return ss_grad_exc.getNLSStatus();
  } else
  {
            //cout << " ++++++++   fully_excited_VOI = ON  , excited  VOI
++++++++  "  << endl;
            if( !( ss_grad_exc.prepAmplitude( ramptime,                          /* ramp-up
time */
                              ramptime + 1000 + ss_rf_exc.getDuration(), /* duration =
ramp-up time + flat-top */
                              ramptime,                            /* ramp-down time */
                              d_rf_exc_GSAmplitude ) )    ||         /*! EGA-04; EGA-02 !*/
     !(  ss_grad_exc.check() ) )
      return ss_grad_exc.getNLSStatus();
  }




  // the refocussing gradient
  // is balanced with the spoiler after the 1st refocussing puls
  /*
  if( !( ss_grad_ref.prepAmplitude( ramptime,
                              sp1dur,
                              ramptime,
                              sp1ampl -
                              (refoc * ss_rf_exc.getDuration() +
                              .5 * ss_grad_exc.getRampDownTime() ) *
                              ss_rf_exc.getGSAmplitude() / sp1dur ) ) ||

          //d_rf_exc_GSAmplitude / sp1dur )) ||  // if fully excited VOI = ON
     !(  ss_grad_ref.check() ) )
      return ss_grad_ref.getNLSStatus();
          */



  // slice selection readout gradient

  if( !( ss_grad_pi_sl.prepAmplitude( ramptime,                          /* ramp-up time
*/
                              ramptime + ss_rf_pi_sl.getDuration(),      /* duration = ramp-
up time + flat-top */
                              ramptime,                            /* ramp-down time */
                              ss_rf_pi_sl.getGSAmplitude() ) ) ||       /*! EGA-04; EGA-02
!*/
     !(  ss_grad_pi_sl.check() ) )
      return ss_grad_pi_sl.getNLSStatus();
  // slice selection phase gradient

          if (!(pMrProt->spectroscopy().outerVolumeSuppression() ))
          {         //cout << " ++++++++   fully_excited_VOI = OFF , excited
VOI_pi_ph   ++++++++  "  << endl;
                    if( !( ss_grad_pi_ph.prepAmplitude( ramptime,                          /*
ramp-up time */
                               ramptime + ss_rf_pi_ph.getDuration(),      /* duration = ramp-
up time + flat-top */
                               ramptime,                            /* ramp-down time */
                               ss_rf_pi_ph.getGSAmplitude()) ) ||        /*! EGA-04; EGA-02
!*/
     !(  ss_grad_pi_ph.check() ) )
      return ss_grad_pi_ph.getNLSStatus();
          }
          else
          {         //cout << " ++++++++   fully_excited_VOI = ON  , excited
VOI_pi_ph   ++++++++  "  << endl;
                    if( !( ss_grad_pi_ph.prepAmplitude( ramptime,                          //
ramp-up time
                               ramptime + ss_rf_pi_ph.getDuration(),      // duration = ramp-
up time + flat-top
                               ramptime,                            // ramp-down time
                               d_rf_pi_ph_GSAmplitude  ) ) ||           //! EGA-04; EGA-02 !
     !(  ss_grad_pi_ph.check() ) )
      return ss_grad_pi_ph.getNLSStatus();
          }


  // compute phase encoding gradients

  // 1st phase encoding direction is READOUT
  // refocusing grad with spoiler
  sd_1st_csi_grad_offset  = sp1ampl * sp1dur; // gradient moment due to spoiling
          if (!(pMrProt->spectroscopy().outerVolumeSuppression() ))
          {//cout << " ++++++++   fully_excited_VOI = OFF , exc_refocGrad
++++++++  "  << endl;
          sd_1st_csi_grad_offset -= (refoc * ss_rf_exc.getDuration() + .5 *
ss_grad_exc.getRampDownTime() ) *
                        ss_rf_exc.getGSAmplitude(); // gradient moment due to spoiling
and slice rephasing
          } else
```

```
            {//cout << " ++++++++   fully_excited_VOI = ON , exc_refocGrad
++++++++  "  << endl;
             sd_1st_csi_grad_offset -= (refoc * ss_rf_exc.getDuration() + .5 *
ss_grad_exc.getRampDownTime() ) *
                         d_rf_exc_GSAmplitude; // gradient moment due to spoiling and
slice rephasing
            }

  sd_1st_csi_grad_offset /= (double)encoddur; // gradient offset strength



  // [FOV] = mm
  // [larmorConst] = MHz / T
  // [encoddur] = us
  sd_1st_csi_grad_step =  1.0E6 / (larmorconst * pMrProt-
>sliceSeries().front().readoutFOV() * encoddur); /*! EGA-03 !*/

  // do some checking
  dmin = sd_1st_csi_grad_offset - sd_1st_csi_grad_step*(int)(.5+.5*pMrProt-
>kSpace().baseResolution());
  dmax = sd_1st_csi_grad_offset + sd_1st_csi_grad_step*(int)(.5+.5*pMrProt-
>kSpace().baseResolution());

  // gradient overflow
  if(  fabs( dmin ) > max_grad_ampl || fabs( dmax ) > max_grad_ampl ){
              TRACE_PUT2(TC_INFO, TF_SEQ, "Warning from %s: \n"
                              "phase encod. gradient READOUT dir. of %f mT/m cannot be
realized;",
                                                                        ptModule, (fabs( dmin )
> fabs( dmax ) ? fabs( dmin ) : fabs( dmax )) );
           return SEQU_SEQ_NOT_PREPARED;
  }



  // gradient ramping
  // before: slice encoding during excitation, after: NULL

            if (!(pMrProt->spectroscopy().outerVolumeSuppression() ))
            {
              if( fabs( dmin - ss_rf_exc.getGSAmplitude() ) > max_grad_ampl ||
                         fabs( dmax - ss_rf_exc.getGSAmplitude() ) > max_grad_ampl ){

                         TRACE_PUT2(TC_INFO, TF_SEQ, "Warning from %s: \n"
                              "phase encod. gradient READOUT dir. of %f mT/m causes
invalid ramping;",
                                                                    ptModule, (fabs( dmin )
> fabs( dmax ) ? fabs( dmin ) : fabs( dmax )) );
        return SEQU_SEQ_NOT_PREPARED;
          }
         }else
          {
                 if( fabs( dmin - d_rf_exc_GSAmplitude ) > max_grad_ampl ||
                         fabs( dmax - d_rf_exc_GSAmplitude ) > max_grad_ampl ){

                   TRACE_PUT2(TC_INFO, TF_SEQ, "Warning from %s: \n"
                         "phase encod. gradient READOUT dir. of %f mT/m causes
invalid ramping;",
                                                                    ptModule, (fabs( dmin )
> fabs( dmax ) ? fabs( dmin ) : fabs( dmax )) );
        return SEQU_SEQ_NOT_PREPARED;
                 }
         }



  // 2nd phase encoding direction is PHASE

  sd_2nd_csi_grad_offset = sp1ampl * sp1dur; // gradient moment due to spoiling
  sd_2nd_csi_grad_offset /= (double)encoddur; // gradient offset strength

  // [FOV] = mm
  // [larmorConst] = MHz / T
  // [encoddur] = us
  sd_2nd_csi_grad_step =  1.0E6 / (larmorconst * pMrProt-
>sliceSeries().front().phaseFOV() * encoddur); /*! EGA-03 !*/

  // do some checking
  dmin = sd_2nd_csi_grad_offset - sd_2nd_csi_grad_step*(int)(.5+.5*pMrProt-
>kSpace().phaseEncodingLines());
  dmax = sd_2nd_csi_grad_offset + sd_2nd_csi_grad_step*(int)(.5+.5*pMrProt-
>kSpace().phaseEncodingLines());

  // gradient overflow
  if(  fabs( dmin ) > max_grad_ampl || fabs( dmax ) > max_grad_ampl ){
         TRACE_PUT2(TC_INFO, TF_SEQ, "Warning from %s: \n"
                         "phase encod. gradient PHASE dir. of %f mT/m cannot be
realized;",
                                                                    ptModule, (fabs( dmin )
> fabs( dmax ) ? fabs( dmin ) : fabs( dmax )) );
        return SEQU_SEQ_NOT_PREPARED;
  }
  // gradient ramping
  // before: NULL, after: slice encoding in PHASE direction

 if (!(pMrProt->spectroscopy().outerVolumeSuppression() ))
           {
             if( fabs( dmin - ss_rf_pi_ph.getGSAmplitude() ) > max_grad_ampl ||
             fabs( dmax - ss_rf_pi_ph.getGSAmplitude() ) > max_grad_ampl ){
             TRACE_PUT2(TC_INFO, TF_SEQ, "Warning from %s: \n"
                                 "phase encod. gradient PHASE dir. of %f mT/m causes invalid
ramping;",
                                                                          ptModule, (fabs( dmin )
> fabs( dmax ) ? fabs( dmin ) : fabs( dmax )) );
          return SEQU_SEQ_NOT_PREPARED;
             }
 }else
 {
           if( fabs( dmin - d_rf_pi_ph_GSAmplitude ) > max_grad_ampl ||
             fabs( dmax - d_rf_pi_ph_GSAmplitude ) > max_grad_ampl ){
             TRACE_PUT2(TC_INFO, TF_SEQ, "Warning from %s: \n"
                                 "phase encod. gradient PHASE dir. of %f mT/m causes invalid
ramping;",
                                                                          ptModule, (fabs( dmin )
> fabs( dmax ) ? fabs( dmin ) : fabs( dmax )) );
          return SEQU_SEQ_NOT_PREPARED;
             }
 }




  // 3rd phase encoding direction is SLICE

  sd_3rd_csi_grad_offset = sp1ampl * sp1dur; // gradient moment due to spoiling
  sd_3rd_csi_grad_offset /= (double)encoddur; // gradient offset strength

  // [FOV] = mm
  // [larmorConst] = MHz / T
  // [encoddur] = us
  sd_3rd_csi_grad_step =  1.0E6 / ( larmorconst * pMrProt-
>sliceSeries().aFront().thickness() * encoddur); /*! EGA-03 !*/ /* 3D CSI */

  // do some checking
  dmin = sd_3rd_csi_grad_offset - sd_3rd_csi_grad_step*(int)(.5+.5*pMrProt-
>kSpace().partitions());
  dmax = sd_3rd_csi_grad_offset + sd_3rd_csi_grad_step*(int)(.5+.5*pMrProt-
>kSpace().partitions());
```

```
  // gradient overflow
  if(  fabs( dmin ) > max_grad_ampl || fabs( dmax ) > max_grad_ampl ){
            TRACE_PUT2(TC_INFO, TF_SEQ, "Warning from %s: \n"
                          "phase encod. gradient SLICE dir. of %f mT/m cannot be
realized;",
                                                           ptModule, (fabs( dmin )
> fabs( dmax ) ? fabs( dmin ) : fabs( dmax )) );
          return SEQU_SEQ_NOT_PREPARED;
  }
  // gradient ramping
  // before: NULL, after: NULL
  // -> no checking needed

#ifdef _DVP_DEBUG
  fprintf( stdout, "\n 1st FOV %f offset %f step %f \n 2nd FOV %f offset %f step %f
\n 3rd FOV %f offset %f step %f\n\n",
            pMrProt->sliceSeries().front().readoutFOV(), sd_1st_csi_grad_offset,
sd_1st_csi_grad_step,
            pMrProt->sliceSeries().front().phaseFOV(), sd_2nd_csi_grad_offset,
sd_2nd_csi_grad_step,
            pMrProt->sliceSeries().aFront().thickness(), sd_3rd_csi_grad_offset,
sd_3rd_csi_grad_step );
#endif

  ////////////////////////////////////////////////////////////////////////
  // prepare encoding gradient timing
  // the amplitude is set within fSeqRun()
  ////////////////////////////////////////////////////////////////////////

  ss_encod_sl.set( ramptime, encoddur, ramptime );
  ss_encod_ph.set( ramptime, encoddur, ramptime );
  ss_encod_ro.set( ramptime, encoddur, ramptime );

  ////////////////////////////////////////////////////////////////////////
  // compute the CSI gradient table
  ////////////////////////////////////////////////////////////////////////

#ifdef _DVP_DEBUG
  fprintf( stdout, "\n meas res. %d %d %d \n final res. %d %d %d \n",
             pMrProt->kSpace().baseResolution(),
                   pMrProt->kSpace().phaseEncodingLines(),
      pMrProt->kSpace().partitions(),
                   pMrProt->spectroscopy().finalMatrixSizeRead(),
                   pMrProt->spectroscopy().finalMatrixSizePhase(),
                   pMrProt->spectroscopy().finalMatrixSizeSlice()
            );
#endif

  // calculate the number of requests of each kernel call, too
  lKernelRequestsPerMeasurement = pMrProt->spectroscopy().preparingScans();

  { // make variables local

  // 2D/3D

    long a1, a2, a3, d1, d2, d3, dd1, dd2, dd3, nave, lowd1, uppd1, lowd2, uppd2,
lowd3, uppd3;
            unsigned char full = (pMrProt->spectroscopy().phaseEncodingType() ==
SEQ::PHASE_ENCODING_FULL) ? 1 : 0,
                  weight = (pMrProt->spectroscopy().phaseEncodingType() ==
SEQ::PHASE_ENCODING_WEIGHTED) ? 1 : 0;
            double dist;

            d1 = pMrProt->kSpace().baseResolution();
            d2 = pMrProt->kSpace().phaseEncodingLines();
            d3 = pMrProt->kSpace().partitions();
            dd1 = pMrProt->spectroscopy().finalMatrixSizeRead();
            dd2 = pMrProt->spectroscopy().finalMatrixSizePhase();
            dd3 = (pMrProt->kSpace().dimension() == SEQ::DIM_3) ? pMrProt-
>spectroscopy().finalMatrixSizeSlice() : 1; // it would be more elegant ...

            //Elijah
            /////
            //cout<<"pMrProt->kSpace().baseResolution()="<<pMrProt-
>kSpace().baseResolution()<<"pMrProt-
>kSpace().phaseEncodingLines()="<<pMrProt-
>kSpace().phaseEncodingLines()<<"pMrProt->kSpace().partitions()="<<pMrProt-
>kSpace().partitions();
            /////
            // ... if this behaviour could be guaranteed for the return value; sadly, with
VA21 this is not always the case

            if( d1*d2*d3 > MAX_N_CSI_ENCODES ){
                  TRACE_PUT2(TC_INFO, TF_SEQ, "Warning from %s: \n"
                        "can't store %d phase encoding steps\n\n",
                                                            ptModule,
(int)(d1*d2*d3) );
        return SEQU_SEQ_NOT_PREPARED;
            }
            if( dd1 < d1 ){
                  if( !( pSeqLim->isContextPrepForBinarySearch() ) ){
```

```
                                    TRACE_PUT3(TC_INFO, TF_SEQ, "Warning from
%s: \n"
                          "final matrix size %d < numb. phase encoding steps %d along
READ \n\n",
                                                                          ptModule, (int)dd1,
(int)d1 );
                   }
      return SEQU_SEQ_NOT_PREPARED;
          }
          if( dd2 < d2 ){
                   if( !( pSeqLim->isContextPrepForBinarySearch() ) ){
                           TRACE_PUT3(TC_INFO, TF_SEQ, "Warning from %s: \n"
                          "final matrix size %d < numb. phase encoding steps %d along
PHASE \n\n",
                                                                          ptModule, (int)dd2,
(int)d2 );
                   }
      return SEQU_SEQ_NOT_PREPARED;
          }
          if( dd3 < d3 ){
             if( !( pSeqLim->isContextPrepForBinarySearch() ) ){
                           TRACE_PUT3(TC_INFO, TF_SEQ, "Warning from %s: \n"
                          "final matrix size %d < numb. phase encoding steps %d along
SLICE \n\n",
                                                                          ptModule, (int)dd3,
(int)d3 );
                   }
      return SEQU_SEQ_NOT_PREPARED;
          }

          nave = pMrProt->averages();

          lowd1 = -d1/2;
          uppd1 = (d1%2) ? d1/2 : d1/2-1;
          lowd2 = -d2/2;
          uppd2 = (d2%2) ? d2/2 : d2/2-1;
          lowd3 = -d3/2;
          uppd3 = (d3%2) ? d3/2 : d3/2-1;

          ssh_1st_csi_addr_offset = (short) dd1/2;
          ssh_2nd_csi_addr_offset = (short) dd2/2;
          ssh_3rd_csi_addr_offset = (short) dd3/2;

          sl_n_csi_encodes = 0;

          for( a3=lowd3; a3<=uppd3; a3++ )
```

```
            for( a2=lowd2; a2<=uppd2; a2++ )
                    for( a1=lowd1; a1<=uppd1; a1++ ){

                            if( full ){

                                      ssh_1st_csi_addr[ sl_n_csi_encodes ] = (short) a1; /*!
EGA-01 !*/
                                      ssh_2nd_csi_addr[ sl_n_csi_encodes ] = (short) a2; /*!
EGA-01 !*/
                                      ssh_3rd_csi_addr[ sl_n_csi_encodes ] = (short) a3; /*!
EGA-01 !*/
/*
                                      if ( (ssh_1st_csi_addr[ sl_n_csi_encodes ] == 0) &&
(ssh_2nd_csi_addr[ sl_n_csi_encodes ] == 0) && (ssh_3rd_csi_addr[
sl_n_csi_encodes ] == 0) )
                                      {
                                         cout << "a1: " << a1 << ";   a2: " << a2 << ";  a3: "
<< a3 << ";   sl_n_csi_encodes: " << sl_n_csi_encodes << ";   a1: " << a1 << endl;;
                                                cout << "ssh_1st_csi_addr[ sl_n_csi_encodes ]
== 0 !!" << endl;
                                                cout << "sl_n_csi_encodes: " <<
sl_n_csi_encodes << ";   a1: " << a1 << endl;
                                                cout << "ssh_2nd_csi_addr[ sl_n_csi_encodes ]
== 0 !!" << endl;
                                                cout << "sl_n_csi_encodes: " <<
sl_n_csi_encodes << ";   a2: " << a1 << endl;
                                                cout << "ssh_3rd_csi_addr[ sl_n_csi_encodes ]
== 0 !!" << endl;
                                                cout << "sl_n_csi_encodes: " <<
sl_n_csi_encodes << ";   a3: " << a1 << endl;
                                                cout <<
"============================================================
==========: "  << endl;
                                      }

                                      if ( ssh_1st_csi_addr[ sl_n_csi_encodes ] == 0 )
                                      {
                                                cout << "ssh_1st_csi_addr[ sl_n_csi_encodes ]
== 0 !!" << endl;
                                                cout << "sl_n_csi_encodes: " <<
sl_n_csi_encodes << ";   a1: " << a1 << endl;
                                      }
                                      if ( ssh_2nd_csi_addr[ sl_n_csi_encodes ] == 0 )
                                      {
                                                cout << "ssh_2nd_csi_addr[ sl_n_csi_encodes ]
== 0 !!" << endl;
```

```
                    cout << "sl_n_csi_encodes: " <<
sl_n_csi_encodes << ";   a2: " << a1 << endl;
                }
                if ( ssh_3rd_csi_addr[ sl_n_csi_encodes ] == 0 )
                {
                    cout << "ssh_3rd_csi_addr[ sl_n_csi_encodes ]
== 0 !!" << endl;
                    cout << "sl_n_csi_encodes: " <<
sl_n_csi_encodes << ";   a3: " << a1 << endl;
                }
*/
                ssh_csi_weight[ sl_n_csi_encodes ] = nave;
                lKernelRequestsPerMeasurement += nave;
                sl_n_csi_encodes++;

                /*
                fprintf( stdout, "\n adresses %d %d %d weight %d",
                                                ssh_1st_csi_addr[
sl_n_csi_encodes-1 ],
                                                ssh_2nd_csi_addr[
sl_n_csi_encodes-1 ],
                                                ssh_3rd_csi_addr[
sl_n_csi_encodes-1 ],
                                                ssh_csi_weight[
sl_n_csi_encodes-1  ] );
                */
            }
            else{ // elliptical or weighted

                // compute the radial distance
                d = (uppd1 == 0) ? 0 : (a1/(double)uppd1);
                dist = d*d;
                d = (uppd2 == 0) ? 0 : (a2/(double)uppd2);
                dist += d*d;
                d = (uppd3 == 0) ? 0 : (a3/(double)uppd3);
                dist += d*d;
                dist = sqrt( dist );

                if( dist <= 1 ){

                    ssh_1st_csi_addr[ sl_n_csi_encodes ] = (short)
a1; /*! EGA-01 !*/
                    ssh_2nd_csi_addr[ sl_n_csi_encodes ] = (short)
a2; /*! EGA-01 !*/
                    ssh_3rd_csi_addr[ sl_n_csi_encodes ] = (short)
a3; /*! EGA-01 !*/
```

```
                    if( weight )
                        ssh_csi_weight[ sl_n_csi_encodes ] =
(short)( .5 + (nave-1) * (.5+.5*cos( M_PI * dist ) ) + 1 );
                    else
                        ssh_csi_weight[ sl_n_csi_encodes ] =
nave;

                    lKernelRequestsPerMeasurement +=
ssh_csi_weight[ sl_n_csi_encodes ];
                    sl_n_csi_encodes++;

                    /*
                    fprintf( stdout, "\n adresses %d %d %d weight
%d",
                                    ssh_1st_csi_addr[
sl_n_csi_encodes-1 ],
                                    ssh_2nd_csi_addr[
sl_n_csi_encodes-1 ],
                                    ssh_3rd_csi_addr[
sl_n_csi_encodes-1 ],
                                    ssh_csi_weight[
sl_n_csi_encodes-1  ] );
                                    */
                }
            }
        }


  } // end of local CSI gradient table computation

  #ifdef _DVP_DEBUG
    fprintf( stdout, "\n phase encod steps %d\n\n", sl_n_csi_encodes );
  #endif





  // ------------------------------------------------------------------------------------
  // time in the PRESS sequence itself from the start to the middle of the excitation
pulse
  // ------------------------------------------------------------------------------------

  time_to_excit_in_sequ = 100 + (ss_grad_exc.getDuration() -
ss_rf_exc.getDuration()) + (ss_rf_exc.getDuration()/2);
```

```
  #ifdef _OWN_DEBUG_02
    cout << "==>time_to_excit_in_sequ (in csi-Sequ):               " <<
time_to_excit_in_sequ << endl;
  #endif



  // ------------------------
  // configure the OVS SBB
  // ------------------------


    for (lI=0; lI<MaxNrVarRSATS; lI++) {
          RSat[lI].setRequestsPerMeasurement (lKernelRequestsPerMeasurement);
          //RSat[lI].adaptFlipAngle (pMrProt, pSeqLim, lI,
(time_to_excit_in_sequ+lScanTimeOVSSats_fixed));
  }



  // ---------------------------
  // configure the fixed OVS SBB
  // ---------------------------


  for (lI=0; lI<MaxNrFixedRsats; lI++) {
        fixedRSat[lI].setRequestsPerMeasurement (lKernelRequestsPerMeasurement);
                     }




  // ------------------------------------------
  // Prepare all SBBs derived from SeqBuildBlock
  // ------------------------------------------

  // my_fixed_rsat_SBB must be prepared first: pulse duration of fixed rsats is needed
within the normal RSatSBB for flip angle calculations

     if (!my_fixed_rsat_SBBList.prepSBBAll (pMrProt, pSeqLim, pSeqExpo,
&dRfEnergyInSBBs_fixed_rsats /*, dsat_phaseFOV, dsat_phaseFOV */))
     return(my_fixed_rsat_SBBList.getpSBBLastPrep()->getNLSStatus()) ;

            block_iterations = 1;  // one means, that no additional iteration is done for
CSI
```

```
                                                              // in this sequence code an
iteration is not implemented; the block_nr is set to one;

    if (!mySBBList.prepSBBAll (pMrProt, pSeqLim, pSeqExpo, &dRfEnergyInSBBs
/*, dsat_phaseFOV, dsat_phaseFOV */))
    return(mySBBList.getpSBBLastPrep()->getNLSStatus()) ;



  //----------------------------------------------------------
  // Calculate OVSSat time

  //----------------------------------------------------------
  lScanTimeOVSSats = 0;

  for (lI=0;lI<MaxNrVarRSATS;lI++) {
    lScanTimeOVSSats  +=  RSat[lI].getDurationPerRequest();
    //cout << "RSat["<<lI<<"].getDurationPerRequest()      ="<<
RSat[lI].getDurationPerRequest() << endl;
  }


  //----------------------------------------------------------
  // Calculate OVSSat time      for fixed rsats

  //----------------------------------------------------------
  lScanTimeOVSSats_fixed = 0;

  for (lI=0;lI<MaxNrFixedRsats;lI++) {
    lScanTimeOVSSats_fixed  +=  fixedRSat[lI].getDurationPerRequest();
    //cout << "fixedRSat["<<lI<<"].getDurationPerRequest() ="<<
fixedRSat[lI].getDurationPerRequest() << endl;

  }





  //////////////////////////////////////////////////////////////////////////
  // Calculate the total measurement time, including measurement repeats
  //////////////////////////////////////////////////////////////////////////

  dMeasureTimeUsec = ((double)lKernelRequestsPerMeasurement) * pMrProt-
>tr()[0];
```

```
      /* + (double) psLOOP->lTokTokTokTime + dScanTimeTrigHalt + (double)
lNoiseMeasTime ; */

  lStatus=fSBBMeasRepetDelaysPrep( pMrProt, pSeqLim, pSeqExpo,
(dMeasureTimeUsec/1000.),

            &dTotalMeasureTimeMsec );
  CheckStatusPB(lStatus,"fSBBMeasRepetDelaysPrep") ;

  ////////////////////////////////////////////////////////////////////////////////////////////////
  // include the repetitions to the number of kernel requests AFTER calculating
dTotalMeasureTimeMsec,
  // since otherwise the repetitions would have been taken into account twice
  /////////////////////////////////////////////////////////////////////////////////////////////////

  lKernelRequestsPerMeasurement *=  (pMrProt->repetitions() + 1 );

  // estimate RF energy
  dRfEnergyInSRFs += lKernelRequestsPerMeasurement *
ss_rf_exc.getPulseEnergyWs();
  dRfEnergyInSRFs += lKernelRequestsPerMeasurement *
ss_rf_pi_sl.getPulseEnergyWs();
  dRfEnergyInSRFs += lKernelRequestsPerMeasurement *
ss_rf_pi_ph.getPulseEnergyWs();

  //Elijah
  ///// Add antoher RF
  dRfEnergyInSRFs += lKernelRequestsPerMeasurement *
ss_rf_pi_sl.getPulseEnergyWs();
/////

  if( pMrProt->preparationPulses().waterSuppression() ==
SEQ::WATER_SATURATION ||
              pMrProt->preparationPulses().waterSuppression() ==
SEQ::WATER_SUPPRESSION_WEAK){
            dRfEnergyInSRFs += lKernelRequestsPerMeasurement *
ss_rf_ws1.getPulseEnergyWs();
            dRfEnergyInSRFs += lKernelRequestsPerMeasurement *
ss_rf_ws2.getPulseEnergyWs();
            dRfEnergyInSRFs += lKernelRequestsPerMeasurement *
ss_rf_ws3.getPulseEnergyWs();
  }
```

```
        // spectral Suppression
 if( pMrProt->spectroscopy().SpectralSuppression() !=  SEQ::SPEC_SUPPR_NONE
)
 {
  dRfEnergyInSRFs  += lKernelRequestsPerMeasurement *
ss_rf_mega1.getPulseEnergyWs() * (pMrProt->repetitions()+1) ;
  dRfEnergyInSRFs  += lKernelRequestsPerMeasurement *
ss_rf_mega2.getPulseEnergyWs() * (pMrProt->repetitions()+1);
 }




 // spoiler 1

 if( !( ss_sp1_ph.prepAmplitude( ramptime, sp1dur, ramptime, sp1ampl ) ) ||
      !( ss_sp1_ph.check() ) )
    return ss_sp1_ph.getNLSStatus();

 if( !( ss_sp1_ro.prepAmplitude( ramptime, sp1dur, ramptime, sp1ampl ) ) ||
      !( ss_sp1_ro.check() ) )
    return ss_sp1_ro.getNLSStatus();

 if( !( ss_sp1_sl.prepAmplitude( ramptime, sp1dur, ramptime, sp1ampl ) ) ||
      !( ss_sp1_sl.check() ) )
    return ss_sp1_sl.getNLSStatus();

 // spoiler 2

 if( !( ss_sp2_ph.prepAmplitude( ramptime, sp2dur, ramptime, sp2ampl ) ) ||
      !( ss_sp2_ph.check() ) )
    return ss_sp2_ph.getNLSStatus();

 if( !( ss_sp2_ro.prepAmplitude( ramptime, sp2dur, ramptime, sp2ampl ) ) ||
      !( ss_sp2_ro.check() ) )
    return ss_sp2_ro.getNLSStatus();

 if( !( ss_sp2_sl.prepAmplitude( ramptime, sp2dur, ramptime, sp2ampl ) ) ||
      !( ss_sp2_sl.check() ) )
    return ss_sp2_sl.getNLSStatus();


 // spectral suppression spoilers


   l = (sp1dur/GRAD_RASTER_TIME)*GRAD_RASTER_TIME;
   if( !( ss_mega1_sl.prepAmplitude( ramptime, l, ramptime, sp2ampl ) ) || !(
ss_mega1_sl.check() ) )
      return ss_mega1_sl.getNLSStatus();
   if( !( ss_mega2_sl.prepAmplitude( ramptime, l, ramptime, -1.0 * sp2ampl ) ) || !(
ss_mega2_sl.check() ) )
      return ss_mega2_sl.getNLSStatus();
   if( !( ss_mega1_ro.prepAmplitude( ramptime, l, ramptime, sp2ampl ) ) || !(
ss_mega1_ro.check() ) )
      return ss_mega1_ro.getNLSStatus();
   if( !( ss_mega2_ro.prepAmplitude( ramptime, l, ramptime, -1.0 * sp2ampl ) ) || !(
ss_mega2_ro.check() ) )
      return ss_mega2_ro.getNLSStatus();
   if( !( ss_mega1_ph.prepAmplitude( ramptime, l, ramptime, sp2ampl ) ) || !(
ss_mega1_ph.check() ) )
      return ss_mega1_ph.getNLSStatus();
   if( !( ss_mega2_ph.prepAmplitude( ramptime, l, ramptime, -1.0 * sp2ampl ) ) || !(
ss_mega2_ph.check() ) )
      return ss_mega2_ph.getNLSStatus();



  // water suppression spoiler

  tau1 = 60000; // delay between water suppression pulses  // 48000 would be min for
spoilgradiampl < 16
  tau2 = 60000; // delay between water suppression pulses
  tau3 = 60000; // delay between water suppression pulse 3 and excitation


  if (lScanTimeOVSSats_fixed > 0)
  { // in case fully excited VoI is switched On, tau must be increased to 80000
        tau1 = 80000; // delay between water suppression pulses
        tau2 = 80000; // delay between water suppression pulses
        tau3 = 80000; // delay between water suppression pulse 3 and excitation
  }


  if (lScanTimeOVSSats > 0)
  { // if the up to 8 variable RSats and 4 fixed RSats are used, tau must be increased to
150000
        tau1 = 150000; // delay between water suppression pulses
        tau2 = 150000; // delay between water suppression pulses
        tau3 = 150000; // delay between water suppression pulse 3 and excitation
  }
  // the water suppression pulses or the excitation pulse must not be longer than tau
  // compute duration to fill spacing between RF-pulses with the spoiler gradient

   // WS spoiler1
   l = tau1 - (long)(0.5+ .5 * ( ss_rf_ws1.getDuration() + ss_rf_ws2.getDuration() ) +
ramptime ); // max. duration for spoiler gradient (flat top + 1 ramp)
   l = fSDSRoundUpGRT(l);
   d = 1000. * wsspgradmoment / (double) l; // d = gradient amplitude




  if( d > max_grad_ampl ){
            d = max_grad_ampl;

            #ifdef _DVP_DEBUG
                     fprintf( stdout, "\nfSEQPrep(): wat.suppr. gradient moment had to be
clipped."
                                                       "\n specified value: %f; realized value:
%f (mT ms / m)\n" ,

          wsspgradmoment, max_grad_ampl * l * .001 );

            #endif
  }



  if( !( ss_wssp_ph.prepAmplitude( ramptime, l, ramptime, d ) ) || !(
ss_wssp_ph.check() ) )
      return ss_wssp_ph.getNLSStatus();



  // WS spoiler 2
  l = tau2 - (long)(0.5+ 0.5 * ( ss_rf_ws2.getDuration() + ss_rf_ws3.getDuration() ) +
ramptime );
  l = (l/GRAD_RASTER_TIME)*GRAD_RASTER_TIME;
  d = 1000. * wsspgradmoment / (double) l; // d = gradient amplitude



  if( d > max_grad_ampl ){
            d = max_grad_ampl;
            #ifdef _DVP_DEBUG
                     fprintf( stdout, "\nfSEQPrep(): wat.suppr. gradient moment had to be
clipped."
```

```
                                                    "\n specified value: %f; realized value:
%f (mT ms / m)\n" ,

          wsspgradmoment, max_grad_ampl * l * .001 );
           #endif

  }



  if( !( ss_wssp_ro.prepAmplitude( ramptime, l, ramptime, d ) ) || !(
ss_wssp_ro.check() ) )
     return ss_wssp_ro.getNLSStatus();





  // compute minimal spoil duration for WS spoiler 3

  long    lwssp_sl_grad_ampl;
  long    lwssp_sl_duration ;

    lwssp_sl_grad_ampl =  (long)(0.5+pSeqLim->getRequiredGradAmpl());  // 16 mT
          lwssp_sl_duration  =  (long)(0.5+wsspgradmoment*1000
/lwssp_sl_grad_ampl);
          lwssp_sl_duration  =
(lwssp_sl_duration/GRAD_RASTER_TIME)*GRAD_RASTER_TIME;

  //if( !( ss_wssp_sl.prepAmplitude( ramptime, l, ramptime, d ) ) ||
  if( !( ss_wssp_sl.prepAmplitude( ramptime, lwssp_sl_duration, ramptime,
lwssp_sl_grad_ampl ) ) ||   /* l -> lwssp_sl_duration */
       !( ss_wssp_sl.check() ) )
     return ss_wssp_sl.getNLSStatus();


  // weaker water suppression option to keep some water signal for postprocessing
  if( pMrProt->preparationPulses().waterSuppression() ==
SEQ::WATER_SUPPRESSION_WEAK)
          { tau3 = 100000;
           if (lScanTimeOVSSats > 0)
                    tau3 = 150000;  // this time is needed to include all RSats
          }
     //  remark: the reference scan (SEQ::WATER_SUPPRESSION_RF_OFF) will be
executed with upper value of tau3
           //  -> the eddy current compensation will not be perfect in this mode
  if( ss_rf_exc.getDuration() > ss_rf_ws3.getDuration() ){ // in the unlikeley case that
the excit. pulse is longer than the wat. sup. pulse

            // starting with VA15A and variable duration suppression pulses, this
NEVER HAPPEN case is not any more supported
            #ifdef _DVP_DEBUG
                    fprintf( stdout, "\nfSEQPrep(): the excitation pulse (dur. %d us) must
NOT be longer than the "
                                                        "\npreceeding water suppr. pulse (dur.
%d us).\n",
                                                                  ss_rf_exc.getDuration(),
ss_rf_ws3.getDuration() );

            #endif
            return SEQU_SEQ_NOT_PREPARED;
  }
  else{ // the usual case that the excit. pulse is shorter than the wat. sup. pulse
     // ... use the same spoil duration l as before, but compute sl_excit_delay

            // sl_excit_delay is the time between last WET-spoiler ramped down and
sliceselction  of SE-CSI experiment -100us  - lScanTimeOVSSats -
lScanTimeOVSSats_fixed

            sl_excit_delay = (long)(.5 + tau3 - (lwssp_sl_duration + ramptime +
.5*(ss_rf_ws3.getDuration() + ss_rf_exc.getDuration()) +
                                  (ss_grad_exc.getDuration() - ss_rf_exc.getDuration()) )  -
100);    // -100 aus der event timing table

     #ifdef _OWN_DEBUG_01
     cout << "==>sl_excit_delay (before subraction of ovs-time): " << sl_excit_delay
<< endl;
     #endif

            sl_excit_delay -= lScanTimeOVSSats + lScanTimeOVSSats_fixed;



            if ( sl_excit_delay < 0)
            return SEQU_SEQ_NOT_PREPARED;
            // here we may run into protocoll inconsitencies, if a certain amount of RSats
is allowed with a min. calulated pulse duration,
            // and with another patient the Rsat-pulse duration may get longer, and the
number of allowed RSats may be reduced !
```

```
     // for tau = 150 and WATER Supp == Off 12 RSats with longest possible RSat-
duration fit into tau3
            // WATER Supp == OFF: in TRmin-calc MaxScanTimeAllRSats is included
so that  TRmin does not depend on the coil or coil load

     if( sl_excit_delay < 0 )
                    //
                    // the 8 RSatduration have a limited  max. duration, so that the
sl_excit_delay  can not be zero.

        sl_excit_delay = 0;
     sl_excit_delay =
(sl_excit_delay/GRAD_RASTER_TIME)*GRAD_RASTER_TIME;

  }



  if( pMrProt->preparationPulses().waterSuppression() ==
SEQ::WATER_SUPPRESSION_OFF)
    h2osup_dur = 0;  // since VA25A SEQ::WATER_SUPPRESSION_OFF means no
WET at all
  else
    h2osup_dur = tau1 + tau2 + tau3 + ss_rf_ws1.getDuration();  // rounding up a bit




#ifdef _OWN_DEBUG_01
  cout  << "   ------  watersuppression -------   "                << endl;
  cout  << "tau1                        = "<< tau1               << endl;
  cout  << "tau2                        = "<< tau2               << endl;
  cout  << "tau3                        = "<< tau3               << endl;
  cout  << ".5*(ss_rf_ws1.getDuration()+ss_rf_ws2.getDuration())+ramptime ="
<<(long)(0.5+ .5 * ( ss_rf_ws1.getDuration() + ss_rf_ws2.getDuration() ) + ramptime
) << endl;
  cout  << "h2osup_dur                  = "<< h2osup_dur          << endl;
  cout  << "free CHESS fill-time         = "<< sl_excit_delay +lScanTimeOVSSats +
lScanTimeOVSSats_fixed << endl;
  cout
                                                   << endl;

  cout  << "   ------ RSat within WET   -------   "
          << endl;
  cout  << "lScanTimeOVSSats            = "  << lScanTimeOVSSats           << endl;
  cout  << "dRfEnergyInSBBs             = "  << dRfEnergyInSBBs            << endl;
  cout
                                                 << endl;

  cout  << "   ------ fixed RSat within WET   -------   "
                  << endl;
  cout  << "lScanTimeOVSSats_fixed       = "  << lScanTimeOVSSats_fixed
                  << endl;
  cout  << "dRfEnergyInSBBs_fixed_rsats  = "  << dRfEnergyInSBBs_fixed_rsats << 
endl;
  cout
                                                 << endl;



  cout  << "resttime after X fixed/normalRSATs = "  << sl_excit_delay       << endl;
  cout  << "d: = adapted Spoil-ampl      = "  << d                        << endl  ;
  cout  << "last Spoil-Ampl.            = "  << lwssp_sl_grad_ampl        << endl  ;
  cout  << "ss_wssp_ro.getDuration()     = "  << ss_wssp_ro.getDuration()   << endl ;
  cout  << "ss_wssp_ro.getTotalTime()    = "  << ss_wssp_ro.getTotalTime()  << endl 
;
  cout  << "lwssp_sl_duration           = "  << lwssp_sl_duration         << endl ;
  cout  << "ss_rf_exc.getDuration()      = "  << ss_rf_exc.getDuration()    << endl ;
  cout  << "pMrProt->tr()[0]            = "  <<  pMrProt->tr()[0]         << endl;
#endif




  // final spoiling pulses

  tau = 20000;

  if( !( ss_finsp_ph.prepAmplitude( ramptime, tau, ramptime, 5. ) ) ||
      !( ss_finsp_ph.check() ) )
     return ss_finsp_ph.getNLSStatus();

  if( !( ss_finsp_ro.prepAmplitude( ramptime, tau, ramptime, 5. ) ) ||
      !( ss_finsp_ro.check() ) )
     return ss_finsp_ro.getNLSStatus();

  if( !( ss_finsp_sl.prepAmplitude( ramptime, tau, ramptime, 5. ) ) ||
      !( ss_finsp_sl.check() ) )
     return ss_finsp_sl.getNLSStatus();

  finalspoil_dur = tau + ramptime;
  ////////////////////////////////////////////////////////////////////////////////////////
  // set the receiver gain
  ////////////////////////////////////////////////////////////////////////////////////////

  // use high gain in SVS sequences

  lStatus = fSSLSetRxGain( K_RX_GAIN_CODE_HIGH, pMrProt, pSeqLim );
  CheckStatusPB (lStatus, "fSSLSetRxGain") ;

  //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  // compute some sequence parameters
  //////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

  // sl_trueTE1 is the fill delay applied between the refocussing pulses which realizes
TE
  // note that TE = 2 * (delay between the refocussing pulses) = 2 * deltaT

  sl_trueTE1 = (long)(.5 + .5 * pMrProt->te()[0] -
                    (.5 * (ss_rf_pi_ph.getDuration() + ss_rf_pi_sl.getDuration()) +  /*
pulses   */
                      ss_sp1_sl.getTotalTime() + ss_sp2_sl.getTotalTime() ));       /*
spoiling */

  if(pMrProt->spectroscopy().SpectralSuppression() !=  SEQ::SPEC_SUPPR_NONE)
  {
            sl_trueTE1 -= (long) (0.5 + ss_mega1_sl.getTotalTime() +
ss_mega2_sl.getTotalTime() + fSDSRoundUpGRT(ss_rf_mega1.getDuration()) );
  }
  sl_trueTE1 = (sl_trueTE1/GRAD_RASTER_TIME)*GRAD_RASTER_TIME;


  // sl_trueTE2 is the eventfree filltime until echo center
  // the ADC event starts direct after the last spoiler grad (after slice refocusing or
MEGA-pulses) so that data are sampled before echo center
  // Note that the delay between the center of the last RF pulse and the start of the
acquisition is
  // given by    deltaT (== .5*TE) - (delay between excit. and first refoc. pulse)
  // as long as all ramptimes are equal:
  // sl_trueTE2 = sl_trueTE1 -(.5*ss_rf_exc.getDuration() +
ss_encod_sl.getTotalTime() -ss_sp1_sl.getTotalTime())


  /* VA21B
  sl_trueTE2 = (long)(.5 + .5 * pMrProt->te()[0] -
                  ( .5*(ss_rf_exc.getDuration() + ss_rf_pi_ph.getDuration()) +
                  ss_encod_sl.getDuration() + ss_grad_pi_ph.getRampUpTime() ) );

  sl_trueTE2 -=(long)(.5 + ss_sp2_sl.getDuration() +
ss_sp2_sl.getRampDownTime()+ .5 * ss_rf_pi_sl.getDuration());
  */



  sl_trueTE2  =  (long)( .5 + .5 * pMrProt->te()[0]   -
                  ( .5 *ss_rf_exc.getDuration()      +
                                                     .5 *ss_rf_pi_ph.getDuration()    + .5
*ss_rf_pi_sl.getDuration() +
                                                         ss_encod_sl.getTotalTime()   +
ss_sp2_sl.getTotalTime()        ) );




  if(pMrProt->spectroscopy().SpectralSuppression() !=  SEQ::SPEC_SUPPR_NONE)
  {
            sl_trueTE2 -= (long) (0.5 + ss_mega1_ph.getTotalTime() +
ss_mega2_ph.getTotalTime() + fSDSRoundUpGRT(ss_rf_mega2.getDuration()) );
  }

  sl_trueTE2 = (sl_trueTE2/GRAD_RASTER_TIME)*GRAD_RASTER_TIME;


  if( sl_trueTE1 < 0 || sl_trueTE2 < 0 ){
     if( !(pSeqLim-> isContextPrepForBinarySearch()) ){
                          TRACE_PUT4(TC_INFO, TF_SEQ, "NEVER HAPPEN
Warning from %s: \n"
                                                          "TE = %d us cannot be realized;\n"
                                                                                  "trueTE1 %d,
trueTE2 %d\n",
                                                                 ptModule, (int)pMrProt->te()[0],
(int)sl_trueTE1, (int)sl_trueTE2 );
            }
     return SEQU_SEQ_NOT_PREPARED;

  }


  // to prevent protocol-inconsistencies: calculate a fixed trueTE2_min
  // for longest possible pi-pulses and use it for TR calculation
  // the new TR now is independent from pulse duration variations which depend for
ex. to the coil loading
  // and might therefor vary during one acquisition
  // sl_trueTE2_minfix is smaller or equal sl_trueTE2 and is only used for TRmin
calculation

   sl_trueTE2_minfix =  (long)(.5 + .5 * pMrProt->te()[0] -
                      ( .5 *ss_rf_exc.getDuration()     +
                                                          .5 *MAX_RF_PI_PULSE_DURATION    +
.5 *MAX_RF_PI_PULSE_DURATION  +    // longest possible pulse duration
                                                              ss_encod_sl.getTotalTime()  +
ss_sp2_sl.getTotalTime()     )  ); // can be negative

   if(pMrProt->spectroscopy().SpectralSuppression() !=  SEQ::SPEC_SUPPR_NONE
)
   {
            sl_trueTE2_minfix -= (long) (0.5+ ss_mega1_ph.getTotalTime() +
ss_mega2_ph.getTotalTime() + fSDSRoundUpGRT(ss_rf_mega2.getDuration()) );
   }

          sl_trueTE2_minfix =
(sl_trueTE2_minfix/GRAD_RASTER_TIME)*GRAD_RASTER_TIME;



  ////////////////////////////////////////////////////////////////////////////////////////////////////
  // prepare the readout-structure
  ////////////////////////////////////////////////////////////////////////////////////////////////////

  // Note that sl_trueTE2 is not used in the sequence kernel.
  // Instead, sampling starts immediately after the last gradient pulse.
  // The sampling points acquired before the echo are disregarded by the ICE-Prg.
  // They are however useful to allow the digital filter to adjust

  ss_adc1.setDwellTime( 10*(int)(.5 + 1E8 /  (double) (pMrProt->bandWidth(
pSeqLim->getReadoutOSFactor() ))[0] ) ); // in ns

  #ifdef _OWN_DEBUG_011
  cout <<
"============================================================" <<
endl;
  cout << "==>sl_trueTE2:                                    " << sl_trueTE2 <<  "us" <<
endl;
  cout << "==>ss_adc1.getDwellTime():                          " <<
ss_adc1.getDwellTime() <<  "ns" << endl;
  cout << "==>(sl_trueTE2-ECHO_DELAY) / (.001 * ss_adc1.getDwellTime()):  "
<< ((sl_trueTE2-ECHO_DELAY) / (.001 * ss_adc1.getDwellTime())) << endl;
```

```
  cout << "==>floor((sl_trueTE2-ECHO_DELAY) / (.001 *
ss_adc1.getDwellTime())): " << floor(((sl_trueTE2-ECHO_DELAY) / (.001 *
ss_adc1.getDwellTime()))) << endl;
  #endif


  if (pMrProt->spectroscopy().vectorSize() >= 2048){
      sl_samplesBeforeEcho = 0;
            // compute the number of acquired samples
            n = pMrProt->spectroscopy().vectorSize();
            // sl_trueTE2 as computed above
  }
  else{
            sl_samplesBeforeEcho = (long)floor( (sl_trueTE2-ECHO_DELAY) / (.001 *
ss_adc1.getDwellTime()) ); // no OS yet, floor instead of ceil
            if (sl_samplesBeforeEcho < 0) {
                      sl_samplesBeforeEcho = 0;
                      #ifdef _OWN_DEBUG_011
                        cout << "sl_samplesBeforeEcho forced to 0 !!!!" << endl; // this
should not happen
                      #endif
            }
      // compute the number of acquired samples
      // the 8 addditional points are useful to avoid signal distortions induced by the
removal of oversampling
      n =  sl_samplesBeforeEcho + pMrProt->spectroscopy().vectorSize() + 8;
            //sl_trueTE2 = 0; set to null after TRmin calculation
  }


  #ifdef _OWN_DEBUG_011
  cout << "==>vector-size:                                    " << pMrProt-
>spectroscopy().vectorSize() << endl;
  cout << "==>sl_samplesBeforeEcho (mit floor):                 " <<
sl_samplesBeforeEcho << endl;
  cout << "==>number of acquired samples (sl_samplesBeforeEcho+vector_size+8): "
<< n << endl;
  #endif

  sl_timeBeforeEcho = sl_samplesBeforeEcho * ss_adc1.getDwellTime();

  sl_aqu_fill_before = sl_trueTE2 - ECHO_DELAY - (long)(0.5 + (.001 *
ss_adc1.getDwellTime() * sl_samplesBeforeEcho));   // empirical value of ~200us
was found
  if (sl_aqu_fill_before < 0) {
            sl_aqu_fill_before = 0;
            #ifdef _OWN_DEBUG_011
              cout << "sl_aqu_fill_before forced to 0 !!!!" << endl;  // this should not
happen
            #endif
  }

  sl_aqu_fill_after = ( sl_aqu_fill_before%GRAD_RASTER_TIME) ?
(1+sl_aqu_fill_before/GRAD_RASTER_TIME)*GRAD_RASTER_TIME :
sl_aqu_fill_before; //rounding, temp value
  sl_aqu_fill_after = sl_aqu_fill_after - sl_aqu_fill_before;

  #ifdef _OWN_DEBUG_011
  cout <<
"============================================================" <<
endl;
  cout << "==>sl_timeBeforeEcho=samplesBeforeEcho *  DwellTime=:   " <<
sl_timeBeforeEcho << "ns" << endl;
  cout << "==>sl_aqu_fill_before ( including -ECHO_DELAY in us):             " <<
sl_aqu_fill_before << endl;
  cout << "==>sl_aqu_fill_after:                              " << sl_aqu_fill_after << endl;
  cout <<
"============================================================" <<
endl;
  #endif



  // round it up to the next integer multiple of 32 after OS (a requirement of the host-
img.reco. connection SW)
  n = ( n%16 ) ? (1 + (n/16)) * 16 : n;

  ss_adc1.setColumns( n );
  // the argument does NOT take OS into account; this is done automatically by
MEAS-SW


  sl_samplesBeforeEcho = (long)( .5 +  sl_samplesBeforeEcho * pSeqLim-
>getReadoutOSFactor() ); // include OS, ab VA12A in pSeqLim


  #ifdef _DVP_DEBUG
  fprintf( stdout, "\n samples before echo %d", sl_samplesBeforeEcho );
  #endif


  #ifdef _DVP_DEBUG_UT
  cout << "==>TE:                                             " << pMrProt->te()[0] << endl;
  cout << "==>ss_rf_pi_sl.getDuration:                          " <<
ss_rf_pi_sl.getDuration() << endl;
  cout << "==>vector-size:                                      " << pMrProt-
>spectroscopy().vectorSize() << endl;
  cout << "==>adc1-columns (nach Aufrunden auf Vielfaches von 16): " << n <<
endl;
  cout << "==>ss_adc1.getDwellTime() (without oversampling):       " <<
ss_adc1.getDwellTime() <<  " ns" <<endl;
  cout << "==>total adc-duration (with samples before echo):       " << n *
ss_adc1.getDwellTime() <<  " ns" <<endl;
  cout << "==>sl_samplesBeforeEcho (factor 2 because of OS):       " <<
sl_samplesBeforeEcho << endl;
  cout << "==>pSeqLim->getReadoutOSFactor():                       " << pSeqLim-
>getReadoutOSFactor() << endl;
  cout << "==>adc-time before echo (DwellTime * samplesBeforeEcho):" <<
sl_samplesBeforeEcho / 2 * ss_adc1.getDwellTime() << endl;
  cout <<
"================================================================" <<
endl;
  #endif

//Elijah
///// Calcuate timing for third ADC   - JZ
//    The 3rd RF will happen at 0.5 * [TE(1)-TE(0)]
//    Dwelltime: ss_adc1.getDwellTime() is for without oversamping.
//    Seems the sequence is always doing oversampling no matter what you choose.
//    So ADC duration after TE[0] is (as calculated above):  (n - sl_samplesBeforeEcho
/ 2)  * ss_adc1.getDwellTime()
//    For RF pulse, we will be just using the 2nd RF pulse, as well as the spoiler.


  sl_fill_before3rdRF = (long)( .5 + 0.5 * (pMrProt->te()[1] - pMrProt->te()[0])
                            - (n - sl_samplesBeforeEcho / 2)  * (.001 *
ss_adc1.getDwellTime())
                            - (.5 *ss_rf_pi_sl.getDuration() + ss_sp2_sl.getTotalTime()));

  sl_fill_before3rdRF = ( sl_fill_before3rdRF
/GRAD_RASTER_TIME)*GRAD_RASTER_TIME;

  sl_fill_after3rdRF = (long)( .5 + 0.5 * (pMrProt->te()[1] - pMrProt->te()[0])
                           - (.5 *ss_rf_pi_sl.getDuration() + ss_sp2_sl.getTotalTime()));

  sl_fill_after3rdRF = ( sl_fill_after3rdRF
/GRAD_RASTER_TIME)*GRAD_RASTER_TIME;
```

```
//  cout << " sl_fill_before3rdRF =                 " << sl_fill_before3rdRF << endl;
//  cout << " sl_fill_after3rdRF =                 " << sl_fill_after3rdRF << endl;


  if( sl_fill_before3rdRF < 0 || sl_fill_after3rdRF < 0 ){
                        if( !(pSeqLim-> isContextPrepForBinarySearch()) ){
                        TRACE_PUT4(TC_INFO, TF_SEQ, "NEVER HAPPEN Warning
from %s: \n"
                                                            "TE = %d us cannot be realized;\n"

"sl_fill_before3rdRF %d, sl_fill_after3rdRF %d\n",
                                                                        ptModule, pMrProt->te()[1],
sl_fill_before3rdRF, sl_fill_after3rdRF );
                               }
                        return SEQU_SEQ_NOT_PREPARED;
    }

  // As the 1st ADC, the 2nd ADC, it will also start right after the 3rd RF but samples
before echo will be discarded

  ss_adc2.setDwellTime( 10*(int)(.5 + 1E8 /  (double) ( pMrProt->bandWidth(
pSeqLim->getReadoutOSFactor() ))[1] ) ); // in ns


  if (pMrProt->spectroscopy().vectorSize() >= 2048){
      sl_samplesBeforeEcho2 = 0;
             // compute the number of acquired samples
             n2 = pMrProt->spectroscopy().vectorSize();

  }
  else{
             sl_samplesBeforeEcho2 = (long)floor( (sl_fill_after3rdRF-ECHO_DELAY)
/ (.001 * ss_adc2.getDwellTime()) ); // no OS yet, floor instead of ceil
             if (sl_samplesBeforeEcho2 < 0) {
                        sl_samplesBeforeEcho2 = 0;
                        //if( !(pSeqLim-> isContextPrepForBinarySearch()) ){
                          #ifdef _DVP_DEBUG
                          cout << "Never HAPPEN warning: 2nd echo:
sl_samplesBeforeEcho2 forced to 0 !!!!" << endl; // this should not happen
                          #endif
                        //}
                        //return SEQU_SEQ_NOT_PREPARED;  //is not needed here,
because with this we can not run into acute timing problems
             }
      // compute the number of acquired samples
```

```
     // the 8 addditional points are useful to avoid signal distortions induced by the
removal of oversampling
     n2 =  sl_samplesBeforeEcho2 + pMrProt->spectroscopy().vectorSize() + 8;

  }

   #ifdef _OWN_DEBUG_011
   cout << "==>2nd echo: sl_samplesBeforeEcho2 (mit floor):                   " <<
sl_samplesBeforeEcho2 << endl;
   cout << "==>2nd echo: ss_adc2.getDwellTime() (without oversampling):       " <<
ss_adc2.getDwellTime() <<  " ns" <<endl;
   cout << "==>2nd echo: number of acquired samples
(sl_samplesBeforeEcho2+vector_size+8): " << n2 << endl;
   #endif

  sl_timeBeforeEcho2 = sl_samplesBeforeEcho2 * ss_adc2.getDwellTime();

  sl_aqu_fill_before2 = sl_fill_after3rdRF - ECHO_DELAY - (long)(.001 *
ss_adc2.getDwellTime() * sl_samplesBeforeEcho2);
  if (sl_aqu_fill_before2 < 0) {
            sl_aqu_fill_before2 = 0;
             #ifdef _DVP_DEBUG
            // if( !(pSeqLim-> isContextPrepForBinarySearch()) ){
             cout << "2nd echo: sl_aqu_fill_before2 forced to 0 !!" << endl;  // this
should never happen
            //  }
      //return SEQU_SEQ_NOT_PREPARED;
      #endif

  }

  sl_aqu_fill_after2 = ( sl_aqu_fill_before2%GRAD_RASTER_TIME) ?
(1+sl_aqu_fill_before2/GRAD_RASTER_TIME)*GRAD_RASTER_TIME :
sl_aqu_fill_before2; //rounding, temp value
  sl_aqu_fill_after2 = sl_aqu_fill_after2 - sl_aqu_fill_before2;

  #ifdef _OWN_DEBUG_011
  cout <<
"==========================================================" <<
endl;
  cout << "==>2nd echo: sl_timeBeforeEcho2=samplesBeforeEcho2 *
DwellTime2=:   " << sl_timeBeforeEcho2 << "ns" << endl;
  cout << "==>2nd echo: sl_aqu_fill_before2 ( including -ECHO_DELAY):        " <<
sl_aqu_fill_before2 << endl;
  cout << "==>2nd echo: sl_aqu_fill_after2:                              " <<
sl_aqu_fill_after2 << endl;
```

```
  cout << 
"============================================================" << 
endl;
  #endif


  // round it up to the next integer multiple of 32 after OS (a requirement of the host-
img.reco. connection SW)
  n2 = ( n2%16 ) ? (1 + (n2/16)) * 16 : n2;

  ss_adc2.setColumns( n2 );
  // the argument does NOT take OS into account; this is done automatically by 
MEAS-SW




  #ifdef _DVP_DEBUG
  fprintf( stdout, "\n 2nd echo: samples before echo %d", sl_samplesBeforeEcho2 );
  #endif

  sl_samplesBeforeEcho2 = (long)( .5 +  sl_samplesBeforeEcho2 * pSeqLim-
>getReadoutOSFactor() ); // include OS, ab VA12A in pSeqLim

  #ifdef _DVP_DEBUG
  fprintf( stdout, "\n 2nd echo: samples before echo (including ReadoutOSFactor) %d
\n", sl_samplesBeforeEcho2 );
  #endif

  #ifdef _DVP_DEBUG_UT
  cout << "==>2nd echo: adc2-columns (nach Aufrunden auf Vielfaches von 16): " << 
n2 << endl;
  cout << "==>2nd echo: ss_adc2.getDwellTime() (without oversampling):       " << 
ss_adc2.getDwellTime() <<  " ns" <<endl;
  cout << "==>2nd echo: total adc-duration (with samples before echo):       " << n2 * 
ss_adc2.getDwellTime() <<  " ns" <<endl;
  cout << "==>2nd echo: sl_samplesBeforeEcho (factor 2 because of OS):       " << 
sl_samplesBeforeEcho2 << endl;
  cout << "==>2nd echo: adc-time before echo (DwellTime * samplesBeforeEcho):" 
<< sl_samplesBeforeEcho2 / 2 * ss_adc2.getDwellTime() << endl;
  cout << 
"============================================================" << 
endl;
  #endif

/////
```

```
  ///////////////////////////////////////////////////////////////////////
  // checking of sequence & Output of SEQU_ERROR to actuate solve handler
  ///////////////////////////////////////////////////////////////////////

   long dd3;
   dd3 = (pMrProt->kSpace().dimension() == SEQ::DIM_3) ? pMrProt-
>spectroscopy().finalMatrixSizeSlice() : 1; // the value of pMrProt-
>spectroscopy().finalMatrixSizeSlice() must be 1 in case 2D; but is initialized with 8
even in case of 2D


    // max final datasize must be bellow MAX_FINAL_DATASIZE
   sl_act_final_datasize = pMrProt->spectroscopy().vectorSize()*pMrProt-
>spectroscopy().finalMatrixSizeRead()*
                                    pMrProt->spectroscopy().finalMatrixSizePhase()* dd3 *
                                                    (pMrProt->repetitions() + 1);



   if( sl_act_final_datasize*SIZE_OF_COMPLEXFLOAT  >
MAX_FINAL_DATASIZE  ||
            sl_act_final_datasize*SIZE_OF_COMPLEXFLOAT * pMrProt-
>coilInfo().Meas().getNumOfUsedRxChan() > 4 * MAX_FINAL_DATASIZE ||
            pMrProt->coilInfo().Meas().getNumOfUsedRxChan () > 16 )
     return SEQU_ERROR;



  // number of measurements > 4 and 3D CSI is forbidden

   if ((pMrProt->repetitions() + 1 > 4)  &&
                      (pMrProt->kSpace().dimension() == SEQ::DIM_3)  )
      return SEQU_ERROR;



  // VecSizeTRConflict & BandWidthTRConflict
  // lScanTimeOVSSats is always included in h2osup_dur, so it must not be beared in
mind in TRmin
  // the too long getRoundedDuration() is corrected with the substaction of sl_trueTE2
  // and to prevent prot inconsistencies, use sl_trueTE2_minfix instead of sl_trueTE2



     // with trueTE2 = 0 -> calculate TEmin;
    // since ECHO_DELAY is included: sl_aqu_fill_before = 0 -> calculation of
TEmin  is not changed

            lTENeededSpecSupp = ss_rf_exc.getDuration() + 2*
ss_encod_sl.getTotalTime() + ss_rf_pi_ph.getDuration()      // echo1 Mitte
                                                    + ss_rf_pi_sl.getDuration()  +  2*
ss_sp2_sl.getTotalTime() ;
            lTENeededSpecSupp  += 2*ss_mega1_ph.getTotalTime() +
2*ss_mega2_ph.getTotalTime() +
2*fSDSRoundUpGRT(ss_rf_mega1.getDuration()); //  megaRF1.dur = megaRF2.dur
            lTENeededSpecSupp  =
(lTENeededSpecSupp/GRAD_RASTER_TIME)*GRAD_RASTER_TIME;


  // VA21B
  // lTRNeededSpectro =   ss_adc1.getRoundedDuration() - sl_trueTE2 + pMrProt-
>te()[0] + h2osup_dur + finalspoil_dur + 5000;
  // VA25A

   // Adaption of TRneeded:
           // the adc1 acquisition time is calculated as product of dwelltime and
(vectorsize+samples before echo).
           // the samples before echo depend on sl_trueTE2 and are rounded to a integer
multiple of 16
   // this rounding effect results in different TRneeded with and without spectral supp.
           // this rounding difference is compensated here



           long l_n_roundingdiff_compensation;
               l_n_roundingdiff_compensation =0;

           if ( pMrProt->spectroscopy().SpectralSuppression() !=
SEQ::SPEC_SUPPR_NONE )
           {
                           long
sl_samplesBeforeEcho_withoutSpecSupp,n_withoutSpecSupp,

           lhelpTime_withoutSpecSupp,lhelpTime_withSpecSupp,
                                           l_trueTE2_withoutSpecSupp;
                           l_trueTE2_withoutSpecSupp=  sl_trueTE2 - ECHO_DELAY +
(long)(0.5 + ss_mega1_ph.getTotalTime() + ss_mega2_ph.getTotalTime() +
fSDSRoundUpGRT(ss_rf_mega2.getDuration()) );
                           l_trueTE2_withoutSpecSupp=
(l_trueTE2_withoutSpecSupp/GRAD_RASTER_TIME)*GRAD_RASTER_TIME;
```

```
                sl_samplesBeforeEcho_withoutSpecSupp
=(long)floor(l_trueTE2_withoutSpecSupp/(.001 * ss_adc1.getDwellTime()) );
//rounding up
                n_withoutSpecSupp =
sl_samplesBeforeEcho_withoutSpecSupp + pMrProt->spectroscopy().vectorSize() +
8;
                n_withoutSpecSupp = (n_withoutSpecSupp%16 ) ? (1 +
(n_withoutSpecSupp /16)) * 16 : n_withoutSpecSupp;      // rounding up to a integer
multiple of 16

            // this time is added to TRneeded when Spec Supp is OFF
                lhelpTime_withoutSpecSupp  =  -
l_trueTE2_withoutSpecSupp +  (long)(0.5+(n_withoutSpecSupp *.001 *
ss_adc1.getDwellTime()));
                //  this time is added to TRneeded when Spec Supp is ON
                lhelpTime_withSpecSupp     =  -  ( sl_trueTE2 -
ECHO_DELAY ) + ss_adc1.getRoundedDuration() ;
                l_n_roundingdiff_compensation =
lhelpTime_withoutSpecSupp -lhelpTime_withSpecSupp;  // can be nagative

                if (pMrProt->spectroscopy().vectorSize() >= 2048)
//sl_samplesBeforeEcho = 0;
                    {n_withoutSpecSupp = pMrProt-
>spectroscopy().vectorSize();
                    l_n_roundingdiff_compensation = 0;}

                if (abs(l_n_roundingdiff_compensation) > 16000)  // 16000 =
001 * ss_adc1.getmaxDwellTime()*16
                    {
                    #ifdef _DVP_DEBUG
                    fprintf( stdout, "\nfSEQPrep(): the roundingdiff.
compensation time (dur. %d us) must NOT be longer than 1600 \n",
                                abs(l_n_roundingdiff_compensation));
                    #endif
                    return SEQU_SEQ_NOT_PREPARED;
                    }
                /*
                cout << "sl_samplesBeforeEcho_withoutSpecSupp ="<<
sl_samplesBeforeEcho_withoutSpecSupp << endl;
                cout << "l_trueTE2_withoutSpecSupp          ="<<
l_trueTE2_withoutSpecSupp      << endl ;
                cout << "sl_trueTE2_withSpecSupp            ="<< sl_trueTE2
<< endl ;
                cout << "n_withoutSpecSupp                  ="<<
n_withoutSpecSupp    << endl;
```

```
                        cout << "n_withSpecSupp                  ="<<
ss_adc1.getColumns() << endl;
                        cout << "lhelpTime_withoutSpecSupp        ="<<
lhelpTime_withoutSpecSupp << endl;
                        cout << "lhelpTime_withSpecSupp           ="<<
lhelpTime_withSpecSupp << endl;
                        cout << "l_n_roundingdiff_compensation      ="<<
l_n_roundingdiff_compensation << endl << endl;
                        */
        }


//Elijah
/////Next few lines were removed by Elijah and correct lTRNeededSpectro is
calculated using second echo
        //if (h2osup_dur == 0)  // mode WATER_SUPPRESSION_OFF
         //{  const long MaxScanTimeAllRSats = 4*(8660) + 500  + 8*(9680) + 500;
// 4 fixed RSats + 8 RSats with max pulse durations in us
          //  lTRNeededSpectro  =  pMrProt->te()[0] - ( sl_trueTE2_minfix -
ECHO_DELAY ) + ss_adc1.getRoundedDuration() + h2osup_dur
                //                                    +  sl_aqu_fill_before +
sl_aqu_fill_after                // additional times due to echo time correction
                                                      //+  lScanTimeOVSSats +
lScanTimeOVSSats_fixed     // with the variable times in TRmin shorter TR can be
realized, but this could lead to protocol inconsistencies
                        //                            +  MaxScanTimeAllRSats
                                   // the RSats need their own time if
WATER_SUPPRESSION=OFF
               //          +  (long)(0.5+0.5*ss_rf_exc.getDuration())  + finalspoil_dur
                            //                   +  l_n_roundingdiff_compensation
+16000;                         // this calculation realizes an equal TRmin for spectral
supp ON or OFF;
                                                              // the 16000 additional us assure,
that  a neg l_samplesbeforeecho_roundingdiff_dur does not reduce TRneeded below
the real needed time

//                  if (lTRNeededSpectro  > pMrProt->tr()[0] ){
        //                          if( !(pSeqLim-> isContextPrepForBinarySearch()) ){
                //                  TRACE_PUT2(TC_INFO, TF_SEQ, "Warning from
%s: \n"
                        //                    "TR = %d us cannot be realized;",
                            //        ptModule, (int)pMrProt->tr()[0] );
                        //}
                        //return SEQU_ERROR;
                        //}
         //}
```

```
    //else
    // {
        //lTRNeededSpectro  =  pMrProt->te()[0] - ( sl_trueTE2_minfix -
ECHO_DELAY ) + ss_adc1.getRoundedDuration() + h2osup_dur
            //                      +  sl_aqu_fill_before +
sl_aqu_fill_after            // additional times due to echo time correction
        //      +  (long)(0.5+0.5*ss_rf_exc.getDuration())  + finalspoil_dur
                //              +  l_n_roundingdiff_compensation  +
16000;          // this calculation realizes an equal TRmin for spectral supp ON or
OFF;
                                        // the 16000 additional us assure,
that  a neg l_samplesbeforeecho_roundingdiff_dur does not reduce TRneeded below
the real needed time


//              if (lTRNeededSpectro  > pMrProt->tr()[0] ){
    //                      if( !(pSeqLim-> isContextPrepForBinarySearch()) ){
        //                  TRACE_PUT2(TC_INFO, TF_SEQ, "Warning from
%s: \n"
            //                  "TR = %d us cannot be realized;",
            //                  ptModule, (int)pMrProt->tr()[0] );
        //      }
        //      return SEQU_ERROR;
        //      }
    // }

//Elijah
///// Use the 2nd echo for TR calculation

            //if (h2osup_dur == 0)  // mode
WATER_SUPPRESSION_(WET)_OFF
          if( pMrProt->preparationPulses().waterSuppression() ==
SEQ::WATER_SUPPRESSION_OFF)
    {
                if (lScanTimeOVSSats == 0)  // calc of shortest
possilbe TR without RSats && without WaterSupp
                {
//                                  lTRNeededSpectro  =  pMrProt->te()[0] - (
sl_trueTE2_minfix - ECHO_DELAY ) + ss_adc1.getRoundedDuration()
//                                          +  sl_aqu_fill_before +
sl_aqu_fill_after            // additional times due to echo time correction
//                      +  (long)(0.5+0.5*ss_rf_exc.getDuration())  + finalspoil_dur
//                                          // + 1000;
//                                          +  l_n_roundingdiff_compensation
+16000;                     // this calculation realizes an equal TRmin for spectral
supp ON or OFF;
```

```
                                                              // the 16000 additional us assure,
that  a neg l_samplesbeforeecho_roundingdiff_dur does not reduce TRneeded below
the real needed time

            lTRNeededSpectro = pMrProt->te()[1] - (sl_fill_after3rdRF -
ECHO_DELAY) + ss_adc2.getRoundedDuration()
                  + sl_aqu_fill_before2 + sl_aqu_fill_after
                        +  (long)(0.5+0.5*ss_rf_exc.getDuration())  + finalspoil_dur
                                                        // + 1000;
                                                        +  l_n_roundingdiff_compensation
+16000;                          // this calculation realizes an equal TRmin for spectral
supp ON or OFF;
                                                              // the 16000 additional us assure,
that  a neg l_samplesbeforeecho_roundingdiff_dur does not reduce TRneeded below
the real needed time

                                   }
                                   else
                                   {/*
                                            // one could reduce TRmin for the case
WATER_SUPPRESSION_OFF || RSats on
                                   const long MaxScanTimeAllRSats =  8*(6656+2000) +
500;       // 8 RSats with max pulse durations in us
                                   lTRNeededSpectro  =  pMrProt->te()[0] - (
sl_trueTE2_minfix - ECHO_DELAY ) + ss_adc1.getRoundedDuration()
                                                        +  sl_aqu_fill_before +
sl_aqu_fill_after                // additional times due to echo time correction
                                                      //+  3 * lScanTimeOVSSats
// with the variable times in TRmin shorter TR can be realized, but this could lead to
protocol inconsistencies
                                                        +  3 * MaxScanTimeAllRSats
                                             // the 3 RSats blocks need their own time if
WATER_SUPPRESSION=OFF

                                                                      // in a first step, we will
not differentiate between 1 or 3 blocks for TRmin
                        +  (long)(0.5+0.5*ss_rf_exc.getDuration())  + finalspoil_dur
                                                        //+ 1000;
                                                        +  l_n_roundingdiff_compensation
+16000;                          // this calculation realizes an equal TRmin for spectral
supp ON or OFF;

                                                                      // the 16000 additional us
assure, that  a neg l_samplesbeforeecho_roundingdiff_dur does not reduce TRneeded
below the real needed time
                              */
```

```
                                        //  the delays between the RSat-blocks, are the same
as above, where the RSat-blocks are included in WET-schema
//                                      lTRNeededSpectro  =  pMrProt->te()[0] - (
sl_trueTE2_minfix - ECHO_DELAY ) + ss_adc1.getRoundedDuration() +
h2osup_dur /* -sl_WET1_delay  the first RSat-delay can be canceled out
savings:60us */
//                                                          +  sl_aqu_fill_before +
sl_aqu_fill_after                // additional times due to echo time correction
//                         +  (long)(0.5+0.5*ss_rf_exc.getDuration())  + finalspoil_dur
//                                                          //+ 1000 ;
//                                                          +  l_n_roundingdiff_compensation  +
16000;            // this calculation realizes an equal TRmin for spectral supp ON or
OFF;
                                        lTRNeededSpectro  =  pMrProt->te()[1] - (
sl_fill_after3rdRF - ECHO_DELAY ) + ss_adc2.getRoundedDuration() +
h2osup_dur /* -sl_WET1_delay  the first RSat-delay can be canceled out
savings:60us */
                                                            +  sl_aqu_fill_before2 +
sl_aqu_fill_after2               // additional times due to echo time correction
                           +  (long)(0.5+0.5*ss_rf_exc.getDuration())  + finalspoil_dur
                                                            //+ 1000 ;
                                                            +  l_n_roundingdiff_compensation  +
16000;            // this calculation realizes an equal TRmin for spectral supp ON or
OFF;
                                                                  // the 16000 additional us assure,
that  a neg l_samplesbeforeecho_roundingdiff_dur does not reduce TRneeded below
the real needed time
                                        }

                  if (lTRNeededSpectro  > pMrProt->tr()[0] ){
                                        if( !(pSeqLim-> isContextPrepForBinarySearch()) ){
                                        TRACE_PUT2(TC_INFO, TF_SEQ, "Warning from
%s: \n"
                                                  "TR = %d us cannot be realized;",
                                                  ptModule, (int)pMrProt->tr()[0] );
                              }
                              return SEQU_ERROR;
                              }
          }
          else                                                    // mode
WATER_SUPPRESSION_(WET)_ON


          {
//                lTRNeededSpectro  =  pMrProt->te()[0] - ( sl_trueTE2_minfix -
ECHO_DELAY ) + ss_adc1.getRoundedDuration() + h2osup_dur
```

```
//                                                  +  sl_aqu_fill_before +
sl_aqu_fill_after              // additional times due to echo time correction
//                         +  (long)(0.5+0.5*ss_rf_exc.getDuration())  + finalspoil_dur
//                                                  //+ 1000 ;
//                                                  +  l_n_roundingdiff_compensation  +
16000;          // this calculation realizes an equal TRmin for spectral supp ON or
OFF;
                lTRNeededSpectro  =  pMrProt->te()[1] - ( sl_fill_after3rdRF -
ECHO_DELAY ) + ss_adc2.getRoundedDuration() + h2osup_dur
                                                    +  sl_aqu_fill_before2 +
sl_aqu_fill_after2             // additional times due to echo time correction
                           +  (long)(0.5+0.5*ss_rf_exc.getDuration())  + finalspoil_dur
                                                    //+ 1000 ;
                                                    +  l_n_roundingdiff_compensation  +
16000;          // this calculation realizes an equal TRmin for spectral supp ON or
OFF;
                                                          // the 16000 additional us assure,
that  a neg l_samplesbeforeecho_roundingdiff_dur does not reduce TRneeded below
the real needed time

                if (lTRNeededSpectro  > pMrProt->tr()[0] ){
                                if( !(pSeqLim-> isContextPrepForBinarySearch()) ){
                                TRACE_PUT2(TC_INFO, TF_SEQ, "Warning from
%s: \n"
                                        "TR = %d us cannot be realized;",
                                        ptModule, (int)pMrProt->tr()[0] );
                        }
                        return SEQU_ERROR;
                        }
         }
/////



        #ifdef _OWN_DEBUG01
                cout << "lTRNeededSpectro     ="<< lTRNeededSpectro << endl;
        #endif


 /*
  if (lTRNeededSpectro  > pMrProt->tr()[0] ){
         if( !(pSeqLim-> isContextPrepForBinarySearch()) ){
                TRACE_PUT2(TC_INFO, TF_SEQ, "Warning from %s: \n"
                                        "TR = %d us cannot be realized;",
                                        ptModule, pMrProt->tr()[0] );
         }
                return SEQU_ERROR;
  }
  */




#ifdef _OWN_DEBUG_timing
                cout  << "   ------ Timing SE  -------   "              << endl;
                double TEmin, TEmin_mega,TEmin_mega_fix ;
                TEmin =      ss_rf_exc.getDuration() + 2.* ss_sp1_sl.getTotalTime() +
ss_rf_pi_ph.getDuration() + ss_rf_pi_sl.getDuration() +  2.*
ss_sp2_sl.getTotalTime()  + 2* 0.0;

                TEmin_mega = ss_rf_exc.getDuration() + 2.*
ss_sp2_sl.getTotalTime() + ss_rf_pi_ph.getDuration()
                        + ss_rf_pi_sl.getDuration() + 2* ss_encod_sl.getTotalTime() +
                        2*ss_mega1_ph.getTotalTime() +
2*ss_mega2_ph.getTotalTime() +
2*fSDSRoundUpGRT(ss_rf_mega2.getDuration()); //

                TEmin_mega_fix = ss_rf_exc.getDuration() + 2.*
ss_encod_sl.getTotalTime() + MAX_RF_PI_PULSE_DURATION   // echo1 Mitte
                        + MAX_RF_PI_PULSE_DURATION +  2*
ss_sp2_sl.getTotalTime() +
                        + 2*ss_mega1_ph.getTotalTime() +
2*ss_mega2_ph.getTotalTime() +
2*fSDSRoundUpGRT(ss_rf_mega1.getDuration()); // echo2 Mitte ; megaRF1.dur =
megaRF2.dur

                cout  << "pMrProt->te()[0]            = "    << pMrProt->te()[0]
<< endl  ;
                cout  << "TEmin_erreichbar            = "    << TEmin
<< endl;
                cout  << "TEmin_mega_erreichbar       = "    << TEmin_mega
                                 << endl;
                //cout  << "TEmin_mega_fixerreichbar    = "  << TEmin_mega_fix
                                 << endl;
                cout  << "lTENeededSpectro                  = "    <<
lTENeededSpecSupp                    << endl;
                cout  << "lTRNeededSpectro            ="     << lTRNeededSpectro <<
endl;
                cout  << "sl_trueTE1                  = "  << sl_trueTE1
                << endl ;
                //cout  << "sl_trueTE1-0.5*ss_rf_exc.getDuration()= " << sl_trueTE1-
0.5*ss_rf_exc.getDuration() << endl;
```

```
                //cout  << "0.5*ss_rf_exc.getDuration()  = "  <<
0.5*ss_rf_exc.getDuration() << endl;
                cout  << "sl_trueTE2              = "  << sl_trueTE2          << endl ;
                //cout  << " MEGA_komplett_einmal      = "  << (long)
(ss_mega1_sl.getTotalTime() + ss_mega2_sl.getTotalTime() +
fSDSRoundUpGRT(ss_rf_mega1.getDuration())) << endl;
                cout  << "sl_trueTE2_minfix        = "    <<  sl_trueTE2_minfix
<< endl ;
                cout  << "pMrProt->te()[0]         = "    <<  pMrProt->te()[0]
                 << endl;
                cout  << "ss_adc1.getRoundedDuration() = "    <<
ss_adc1.getRoundedDuration() << endl;
                cout  << "(long)(0.5+0.5*ss_rf_exc.getDuration())  + finalspoil_dur +
1000 = " <<  (long)(0.5+0.5*ss_rf_exc.getDuration())  + finalspoil_dur + 1000
                << endl;
                cout  << "h2osup_dur              = "    <<  h2osup_dur
                    << endl;

//Elijah
          /////
          cout  << "pMrProt->te()[1]         = "    << pMrProt->te()[1]          <<
endl;
          cout  << "ss_adc1.getRoundedDuration() = "    <<
ss_adc1.getRoundedDuration() << endl;
          cout  << "sl_fill_before3rdRF       = "    << sl_fill_before3rdRF
         << endl ;
          cout  << "sl_fill_after3rdRF        = "    << sl_fill_after3rdRF         << endl
;
    /////
                #endif



                if (pMrProt->spectroscopy().vectorSize() < 2048)
                        sl_trueTE2 = 0;

                //Elijah
/////
  if (pMrProt->spectroscopy().vectorSize() < 2048)
    sl_fill_after3rdRF = 0;
/////


#ifdef _OWN_DEBUG_02
                cout << "SEQ::INNER_LOOP: " << SEQ::INNER_LOOP << endl;
                cout << "SEQ::OUTER_LOOP: " << SEQ::OUTER_LOOP << endl;
```

```
                    if (pMrProt->kSpace().averagingMode() == SEQ::OUTER_LOOP)
// acquisition outside lines loop
                    {
                              cout << endl;
                              cout << " SEQ::OUTER_LOOP:     ";
                              cout << "pMrProt->kSpace().averagingMode(): " << pMrProt-
>kSpace().averagingMode();
                    }
                    else  // acquisition loop inside lines loop
                    {
                              cout << endl;
                              cout << " SEQ::INNER_LOOP:     ";
                              cout << "pMrProt->kSpace().averagingMode(): " << pMrProt-
>kSpace().averagingMode();
                    }
                    if (pMrProt->kSpace().averagingMode() == SEQ::INNER_LOOP)    //
acquisition inside lines loop / Short Term / normal
                    {
                              cout << endl;
                              cout << " SEQ::INNER_LOOP:     ";
                              cout << "pMrProt->kSpace().averagingMode(): " << pMrProt-
>kSpace().averagingMode();
                    }
#endif




            ////////////////////////////////////////////////////////////////////////////////////////
            // export parameters to ICE program
  ////////////////////////////////////////////////////////////////////////////////////////

  pSeqExpo->setMeasureTimeMin( dMeasureTimeUsec/60000000.0 );
  pSeqExpo->setTotalMeasureTimeMin( dTotalMeasureTimeMsec/60000.0 );
  pSeqExpo->setRFEnergyInSequence_Ws( mainNucleus, dRfEnergyInSRFs +
dRfEnergyInSBBs + dRfEnergyInSBBs_fixed_rsats);
  //pSeqExpo->setMeasuredPELines( 1 );
  pSeqExpo->setSequenceString( "csi_se" );
  pSeqExpo->setSeqShortString( "csi_se" );


FINISHED:

  return(lStatus);
}
```

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

/*] END: */

NLS_STATUS fSEQCheck
(
  MrProt       *pMrProt,           /* IMP: user choice parameters  */
  SeqLim       *pSeqLim,           /* IMP: limits from fSEQInit()  */
  SeqExpo      *pSeqExpo,          /* IMP: exports from fSEQPrep() */
  SEQCheckMode *pSEQCheckMode      /* unused                       */
)
{

  static const char *ptModule = {"fSEQCheck"};
  NLS_STATUS lStatus = SEQU__NORMAL;
  // may use SEQU_SEQ_NOT_CHECKED to indicate failure

  //////////////////////////////////////////////////////////////////////////
  // execute kernel for checking (GSWD look ahead functionality)
  //////////////////////////////////////////////////////////////////////////

   if( !( ss_encod_sl.prepAmplitude(  sd_3rd_csi_grad_offset +

ssh_3rd_csi_addr[0] * sd_3rd_csi_grad_step ) ) ||
                     !( ss_encod_sl.check() ) )
                     return ss_encod_sl.getNLSStatus();

   if( !( ss_encod_ro.prepAmplitude(  sd_1st_csi_grad_offset +

ssh_1st_csi_addr[0] * sd_1st_csi_grad_step ) ) ||
                     !( ss_encod_ro.check() ) )
                     return ss_encod_ro.getNLSStatus();

   if( !( ss_encod_ph.prepAmplitude(  sd_2nd_csi_grad_offset +
ssh_2nd_csi_addr[0] * sd_2nd_csi_grad_step ) ) ||
                         !( ss_encod_ph.check() ) )
                         return ss_encod_ph.getNLSStatus();


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

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

  return SEQU__NORMAL;
}



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

/*] END: */

NLS_STATUS fSEQRun
(
  MrProt  *pMrProt,    /* IMP: user choice parameters  */
  SeqLim  *pSeqLim,    /* IMP: limits from fSEQInit()  */
  SeqExpo *pSeqExpo    /* IMP: exports from fSEQPrep() */
)
{
  static const char *ptModule = {"fSEQRun"};
  NLS_STATUS lStatus          = SEQU__NORMAL;
  double excit_phase, aqc_phase;
  long nave;

  mPrintTrace1 (DEBUG_RUN, DEBUG_CALL, "() <%s> started", pSeqLim-
>getLinkedSeqFilename() ) ;
```

```
  ////////////////////////////////////////////////////////////
  //  initialization of the unit test function
  ////////////////////////////////////////////////////////////

  mSEQTest(pMrProt,pSeqLim,pSeqExpo,RTEB_ORIGIN_fSEQRunStart,0,0,0,0,0);


  #ifdef _DVP_DEBUG
          print_slicepos( "VoI", &ss_voi );
          print_slicepos( "FoV", &ss_fov );
  #endif

  ////////////////////////////////////////////////////////////
  //  set looping parameters
  ////////////////////////////////////////////////////////////

  long i,j,n_prep = pMrProt->spectroscopy().preparingScans(), k, n_rep;

  ss_adc1.Mdh.setClin( 0 ); // 1st
  ss_adc1.Mdh.setCphs( 0 ); // 2nd
  ss_adc1.Mdh.setCseg( 0 ); // 3rd
  ss_adc1.Mdh.setCeco( 0 ); // echo number
  ss_adc1.Mdh.setCset( 0 ); // averages
  ss_adc1.Mdh.setCslc( 0 ); // slice number
  ss_adc1.Mdh.setCrep( 0 ); // repetitions

  ////////////////////////////////////////////////////////////
  // other Mdh info
  ////////////////////////////////////////////////////////////

  ss_adc1.Mdh.setFreeParameterByIndex( 0, (unsigned short) sl_samplesBeforeEcho
);

  //Elijah
  ///// Set up Mdh for 2nd echo
  ss_adc2.Mdh.setClin( 0 ); // 1st
  ss_adc2.Mdh.setCphs( 0 ); // 2nd
  ss_adc2.Mdh.setCseg( 0 ); // 3rd
  ///%%
  ss_adc2.Mdh.setCeco( 1 ); // echo number
  //ss_adc2.Mdh.setCeco( 0); // echo number
  //%%
  ss_adc2.Mdh.setCset( 0 ); // averages
  ss_adc2.Mdh.setCslc( 0 ); // slice number
  ss_adc2.Mdh.setCrep( 0 ); // repetitions
  ss_adc2.Mdh.setFreeParameterByIndex( 0, (unsigned short) sl_samplesBeforeEcho2
);

  /////

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

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

             ss_adc1.Mdh.setCrep( k );

             //Elijah

/////
     ss_adc2.Mdh.setCrep( k );
/////

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

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

     lStatus = fSEQRunKernel( pMrProt, pSeqLim, pSeqExpo, KERNEL_CHECK );
     CheckStatusPR(lStatus,"fSEQRunKernel");
  }
  fRTSetReadoutEnable( 1 );  // enable ADC events





  //////////////////////////////////////////////////////////////
  // execute acquisition loop
  //////////////////////////////////////////////////////////////

#ifdef _OWN_DEBUG_02
  cout << "SEQ::INNER_LOOP: " << SEQ::INNER_LOOP << endl;
  cout << "SEQ::OUTER_LOOP: " << SEQ::OUTER_LOOP << endl;
  if (pMrProt->kSpace().averagingMode() == SEQ::OUTER_LOOP)    // acquisition
outside lines loop
```

```
  {
            cout << endl;
            cout << " SEQ::OUTER_LOOP:     ";
            cout << "pMrProt->kSpace().averagingMode(): " << pMrProt-
>kSpace().averagingMode();
  }
  else  // acquisition loop inside lines loop
  {
            cout << endl;
            cout << " SEQ::INNER_LOOP:     ";
            cout << "pMrProt->kSpace().averagingMode(): " << pMrProt-
>kSpace().averagingMode();
  }
  if (pMrProt->kSpace().averagingMode() == SEQ::INNER_LOOP)    // acquisition
inside lines loop / Short Term / normal
  {
            cout << endl;
            cout << " SEQ::INNER_LOOP:     ";
            cout << "pMrProt->kSpace().averagingMode(): " << pMrProt-
>kSpace().averagingMode();
  }
#endif



if (pMrProt->kSpace().averagingMode() == SEQ::INNER_LOOP)    // acquisition
inside lines loop / Short Term / normal
{
          cout << endl;
          cout << " SEQ::INNER_LOOP  << endl ";

  for( i=0; i<sl_n_csi_encodes; i++ ){

          #ifdef _DVP_DEBUG
            fprintf( stdout, "\n ave %d csi1 %d csi2 %d csi3 %d\n",
                                    ssh_csi_weight[i],
                                    ssh_1st_csi_addr[i],
                                    ssh_2nd_csi_addr[i],
                                    ssh_3rd_csi_addr[i] );
          #endif

            if( !( ss_encod_sl.prepAmplitude(  sd_3rd_csi_grad_offset +

ssh_3rd_csi_addr[i] * sd_3rd_csi_grad_step ) ) ||
                    !( ss_encod_sl.check() ) )
                    return ss_encod_sl.getNLSStatus();
            if( !( ss_encod_ro.prepAmplitude(  sd_1st_csi_grad_offset +

ssh_1st_csi_addr[i] * sd_1st_csi_grad_step ) ) ||
                      !( ss_encod_ro.check() ) )
                      return ss_encod_ro.getNLSStatus();


            if( !( ss_encod_ph.prepAmplitude(  sd_2nd_csi_grad_offset +

ssh_2nd_csi_addr[i] * sd_2nd_csi_grad_step ) ) ||
                      !( ss_encod_ph.check() ) )
                      return ss_encod_ph.getNLSStatus();

            for( j=0; j<ssh_csi_weight[i]; j++ ){ // averages

                    ss_adc1.Mdh.setCset( j ); // averages
                    ss_adc1.Mdh.setClin( ssh_1st_csi_addr[i] + ssh_1st_csi_addr_offset );
                    ss_adc1.Mdh.setCphs( ssh_2nd_csi_addr[i] + ssh_2nd_csi_addr_offset
);
                    ss_adc1.Mdh.setCseg( ssh_3rd_csi_addr[i] + ssh_3rd_csi_addr_offset
);


                    // flags for extracting time-stamps
                    ss_adc1.Mdh.setFirstScanInSlice( !i && !j );
                    ss_adc1.Mdh.setLastScanInSlice( i==(sl_n_csi_encodes-1) &&
j==(ssh_csi_weight[i]-1) );

          //Elijah
          /////
          ss_adc2.Mdh.setCset( j ); // averages
                    //ss_adc2.Mdh.setClin( ssh_1st_csi_addr[i] + ssh_1st_csi_addr_offset
);
                    //ss_adc2.Mdh.setCphs( ssh_2nd_csi_addr[i] +
ssh_2nd_csi_addr_offset );
                    //ss_adc2.Mdh.setCseg( ssh_3rd_csi_addr[i] +
ssh_3rd_csi_addr_offset );

                    ss_adc2.Mdh.setClin( ssh_1st_csi_addr[0] + ssh_1st_csi_addr_offset +
ssh_1st_csi_addr[sl_n_csi_encodes-1] + ssh_1st_csi_addr_offset -
(ssh_1st_csi_addr[i] + ssh_1st_csi_addr_offset) );
                    ss_adc2.Mdh.setCphs( ssh_2nd_csi_addr[0] +
ssh_2nd_csi_addr_offset + ssh_2nd_csi_addr[sl_n_csi_encodes-1] +
ssh_2nd_csi_addr_offset - (ssh_2nd_csi_addr[i] + ssh_2nd_csi_addr_offset) );
```

```
                ss_adc2.Mdh.setCseg( ssh_3rd_csi_addr[0] + ssh_3rd_csi_addr_offset
+ ssh_3rd_csi_addr[sl_n_csi_encodes-1] + ssh_3rd_csi_addr_offset -
(ssh_3rd_csi_addr[i] + ssh_3rd_csi_addr_offset) );


                cout<<"\n"<<"min"<<ssh_1st_csi_addr[0] + ssh_1st_csi_addr_offset
<<"max" <<ssh_1st_csi_addr[sl_n_csi_encodes-1] + ssh_1st_csi_addr_offset;
                cout <<"\n"<<" adc1clin="<<ssh_1st_csi_addr[i] +
ssh_1st_csi_addr_offset <<" adc2clin="<<ssh_1st_csi_addr[0] +
ssh_1st_csi_addr_offset + ssh_1st_csi_addr[sl_n_csi_encodes-1] +
ssh_1st_csi_addr_offset - (ssh_1st_csi_addr[i] + ssh_1st_csi_addr_offset)<<";" << "
adc1cphs="<<ssh_2nd_csi_addr[i] + ssh_2nd_csi_addr_offset << " adc2cphs="<<
ssh_2nd_csi_addr[0] + ssh_2nd_csi_addr_offset +
ssh_2nd_csi_addr[sl_n_csi_encodes-1] + ssh_2nd_csi_addr_offset -
(ssh_2nd_csi_addr[i] + ssh_2nd_csi_addr_offset)  <<";"<< "
adc1cseg="<<ssh_3rd_csi_addr[i] + ssh_3rd_csi_addr_offset << "
adc2cseg="<<ssh_3rd_csi_addr[0] + ssh_3rd_csi_addr_offset +
ssh_3rd_csi_addr[sl_n_csi_encodes-1] + ssh_3rd_csi_addr_offset -
(ssh_3rd_csi_addr[i] + ssh_3rd_csi_addr_offset) ;

                //d1 = pMrProt->kSpace().baseResolution();
         //d2 = pMrProt->kSpace().phaseEncodingLines();
         //d3 = pMrProt->kSpace().partitions();

                // flags for extracting time-stamps
                ss_adc2.Mdh.setFirstScanInSlice( !i && !j );
                ss_adc2.Mdh.setLastScanInSlice( i==(sl_n_csi_encodes-1) &&
j==(ssh_csi_weight[i]-1) );
         /////
      // realize off-centre FoV positions by incrementing the phase of the excitation
pulses
                // from step to step

                excit_phase =  360. * (
                        - ssh_1st_csi_addr[i] * sd_read_pos / pMrProt-
>sliceSeries().front().readoutFOV()
                        - ssh_2nd_csi_addr[i] * sd_phase_pos / pMrProt-
>sliceSeries().front().phaseFOV()
                        - ssh_3rd_csi_addr[i] * sd_slice_pos / pMrProt-
>sliceSeries().front().thickness() ); /*! EGA-06 !*/

                if( (ssh_1st_csi_addr[i] ^ ssh_2nd_csi_addr[i] ^ ssh_3rd_csi_addr[i] ^
j) & 1 ){
                        excit_phase += 180.;
                        aqc_phase = 180.;
                }
```

```
        else
            aqc_phase = 0;

        ss_ph_s_exc.increasePhase( excit_phase );
        ss_ph_n_exc.decreasePhase( excit_phase );

        //Elijah
        //The following lines removed by Elijah to specify phase event for 2
separate ADCs
        //ss_ph_s_adc.increasePhase( aqc_phase );
        //ss_ph_n_adc.decreasePhase( aqc_phase );

        //Elijah
        /////
        ss_ph_s_adc1.increasePhase( aqc_phase );
        ss_ph_n_adc1.decreasePhase( aqc_phase );

        ss_ph_s_adc2.increasePhase( aqc_phase );
        ss_ph_n_adc2.decreasePhase( aqc_phase );
        /////

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

        // undo phase cycling

        ss_ph_s_exc.decreasePhase( excit_phase );
        ss_ph_n_exc.increasePhase( excit_phase );

        //The following lines removed by elijah and replaced below with 2
separate adc phase events
        //ss_ph_s_adc.decreasePhase( aqc_phase );
        //ss_ph_n_adc.increasePhase( aqc_phase );

        //elijah
        /////
        ss_ph_s_adc1.decreasePhase( aqc_phase );
        ss_ph_n_adc1.increasePhase( aqc_phase );

        ss_ph_s_adc2.decreasePhase( aqc_phase );
        ss_ph_n_adc2.increasePhase( aqc_phase );
        /////

     } // end averaging loop
 } // end encoding loop
} // if (pMrProt->kSpace().averagingMode() == SEQ::INNER_LOOP)    //
acquisition inside lines loop / Short Term / normal


if (pMrProt->kSpace().averagingMode() == SEQ::OUTER_LOOP)    // acquisition
outside lines loop
{

  cout << " SEQ::OUTER_LOOP  << endl ";

  nave = pMrProt->averages();
  for ( j=0; j<nave; j++)          // averages
  {
          for( i=0; i<sl_n_csi_encodes; i++ )  // PE steps
          {
#ifdef _OWN_DEBUG_02
                  cout << endl;
                  cout << "Kind of averaging" << endl;
                  cout << "Average: " << j+1 << ";   j: " << j << endl;
                  cout << "max. Average: nave:" << nave << endl;
                  cout << "PE step:" << i << endl;
                  cout << "ssh_csi_weight[i] of this PE step:" << ssh_csi_weight[i] <<
endl;
                  cout << "total number of PE steps = sl_n_csi_encodes: " <<
sl_n_csi_encodes << endl;
#endif

                  if ( ssh_csi_weight[i] > j)              // measure if this additional scan
is really necessary
                  {
#ifdef _OWN_DEBUG_02
                          cout << "YES: ssh_csi_weight[i]: " << ssh_csi_weight[i] << "
is > j: " << j << endl;
#endif
                          if( !( ss_encod_sl.prepAmplitude(  sd_3rd_csi_grad_offset +
                                  ssh_3rd_csi_addr[i] * sd_3rd_csi_grad_step ) ) ||
                                  !( ss_encod_sl.check() ) )
                                  return ss_encod_sl.getNLSStatus();

                          if( !( ss_encod_ro.prepAmplitude(  sd_1st_csi_grad_offset +
                                  ssh_1st_csi_addr[i] * sd_1st_csi_grad_step ) ) ||
                                  !( ss_encod_ro.check() ) )
                                  return ss_encod_ro.getNLSStatus();

                          if( !( ss_encod_ph.prepAmplitude(  sd_2nd_csi_grad_offset +
                                  ssh_2nd_csi_addr[i] * sd_2nd_csi_grad_step ) ) ||
                                !( ss_encod_ph.check() ) )
                                return ss_encod_ph.getNLSStatus();

/*
                        // cout << "3rd: ss_encod_sl.getAmplitude" << *ss_encod_sl
<< endl;
                        // cout << "1st: sd_1st_csi_grad_offset: " <<
sd_1st_csi_grad_offset << endl;
                        // cout << "2nd: sd_2nd_csi_grad_offset: " <<
sd_2nd_csi_grad_offset << endl;
                        // cout << "3rd: sd_3rd_csi_grad_offset: " <<
sd_3rd_csi_grad_offset << endl;
                        cout << "1st: ssh_1st_csi_addr[i]: " << ssh_1st_csi_addr[i] <<
endl;
                        cout << "2nd: ssh_2nd_csi_addr[i]: " << ssh_2nd_csi_addr[i]
<< endl;
                        cout << "3rd: ssh_3rd_csi_addr[i]: " << ssh_3rd_csi_addr[i]
<< endl;
                        cout << "1st: ssh_1st_csi_addr[i] * sd_1st_csi_grad_step: "
<< ssh_1st_csi_addr[i] * sd_1st_csi_grad_step << endl;
                        cout << "2nd: ssh_2nd_csi_addr[i] * sd_2nd_csi_grad_step: "
<< ssh_2nd_csi_addr[i] * sd_2nd_csi_grad_step << endl;
                        cout << "3rd: ssh_3rd_csi_addr[i] * sd_3rd_csi_grad_step: "
<< ssh_3rd_csi_addr[i] * sd_3rd_csi_grad_step << endl;
                        // cout << "1st: ss_encod_ro.getAmplitude: " <<
ss_encod_ro.getAmplitude() << endl;
                        // cout << "2nd: ss_encod_ph.getAmplitude: " <<
ss_encod_ph.getAmplitude() << endl;
                        // cout << "3rd: ss_encod_sl.getAmplitude: " <<
ss_encod_sl.getAmplitude() << endl;
*/



                        ss_adc1.Mdh.setCset( j ); // averages
                        ss_adc1.Mdh.setClin( ssh_1st_csi_addr[i] +
ssh_1st_csi_addr_offset );
                        ss_adc1.Mdh.setCphs( ssh_2nd_csi_addr[i] +
ssh_2nd_csi_addr_offset );
                        ss_adc1.Mdh.setCseg( ssh_3rd_csi_addr[i] +
ssh_3rd_csi_addr_offset );

                        // flags for extracting time-stamps
                        ss_adc1.Mdh.setFirstScanInSlice( !i && !j );
                        ss_adc1.Mdh.setLastScanInSlice( i==(sl_n_csi_encodes-1)
&& j==(ssh_csi_weight[i]-1) );
```

```
            //Elijah
            /////

            ss_adc2.Mdh.setClin( ssh_1st_csi_addr[0] +
ssh_1st_csi_addr_offset + ssh_1st_csi_addr[sl_n_csi_encodes-1] +
ssh_1st_csi_addr_offset - (ssh_1st_csi_addr[i] + ssh_1st_csi_addr_offset) );
        ss_adc2.Mdh.setCphs( ssh_2nd_csi_addr[0] +
ssh_2nd_csi_addr_offset + ssh_2nd_csi_addr[sl_n_csi_encodes-1] +
ssh_2nd_csi_addr_offset - (ssh_2nd_csi_addr[i] + ssh_2nd_csi_addr_offset) );
        ss_adc2.Mdh.setCseg( ssh_3rd_csi_addr[0] + ssh_3rd_csi_addr_offset
+ ssh_3rd_csi_addr[sl_n_csi_encodes-1] + ssh_3rd_csi_addr_offset -
(ssh_3rd_csi_addr[i] + ssh_3rd_csi_addr_offset) );

            // flags for extracting time-stamps
            ss_adc2.Mdh.setFirstScanInSlice( !i && !j );
            ss_adc2.Mdh.setLastScanInSlice( i==(sl_n_csi_encodes-1)
&& j==(ssh_csi_weight[i]-1) );
            /////

            // realize off-centre FoV positions by incrementing the phase
of the excitation pulses
            // from step to step

            excit_phase =  360. * (
                - ssh_1st_csi_addr[i] * sd_read_pos / pMrProt-
>sliceSeries().front().readoutFOV()
                - ssh_2nd_csi_addr[i] * sd_phase_pos / pMrProt-
>sliceSeries().front().phaseFOV()
                - ssh_3rd_csi_addr[i] * sd_slice_pos / pMrProt-
>sliceSeries().front().thickness() ); /*! EGA-06 !*/

            if( (ssh_1st_csi_addr[i] ^ ssh_2nd_csi_addr[i] ^
ssh_3rd_csi_addr[i] ^ j) & 1 ){
                excit_phase += 180.;
                aqc_phase = 180.;
            }
            else
                aqc_phase = 0;

            ss_ph_s_exc.increasePhase( excit_phase );
            ss_ph_n_exc.decreasePhase( excit_phase );

            //The following two lines were removed by elijah and
replaced with two separate adc phase events
            //ss_ph_s_adc.increasePhase( aqc_phase );
            //ss_ph_n_adc.decreasePhase( aqc_phase );
```

```
                                        //Elijah
                                        /////
                                        ss_ph_s_adc1.increasePhase( aqc_phase );
                                        ss_ph_n_adc1.decreasePhase( aqc_phase );

                                        ss_ph_s_adc2.increasePhase( aqc_phase );
                                        ss_ph_n_adc2.decreasePhase( aqc_phase );
                                        /////

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

                                        // undo phase cycling

                                        ss_ph_s_exc.decreasePhase( excit_phase );
                                        ss_ph_n_exc.increasePhase( excit_phase );

                                       //Elijah
                                       /////
                                       //The following two lines were removed by elijah and replaced
with two separate adc phase events
                                       //ss_ph_s_adc.decreasePhase( aqc_phase );
                                        //ss_ph_n_adc.increasePhase( aqc_phase );

                                        ss_ph_s_adc1.decreasePhase( aqc_phase );
                                        ss_ph_n_adc1.increasePhase( aqc_phase );

                                        ss_ph_s_adc2.decreasePhase( aqc_phase );
                                        ss_ph_n_adc2.increasePhase( aqc_phase );
                                        /////
                            }                   // if ( ssh_csi_weight[i] >= j)
                            else
                            {
                                        cout << "NO: no scan " << endl;
                            }

                }                   // PE steps
  }                  // averages
} // if (pMrProt->kSpace().averagingMode() == SEQ::OUTER_LOOP)    //
acquisition outside lines loop

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

```
  }
  }

FINISHED:



mSEQTest(pMrProt,pSeqLim,pSeqExpo,RTEB_ORIGIN_fSEQRunFinish,0,0,0,0,0);
  mPrintTrace1 (DEBUG_RUN, DEBUG_CALL | DEBUG_RETURN, "() <%s>
finished",
                        pSeqLim->getLinkedSeqFilename() ) ;
  return(lStatus);
}




/*[ 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.
*
\*********************************************************************
********/

/*] END: */

static NLS_STATUS fSEQRunKernel
(
  MrProt        *pMrProt,
  SeqLim        *pSeqLim,
  SeqExpo       *pSeqExpo,
  long          lKernelMode
)
{
  static const char *ptModule         = {"fSEQRunKernel"} ;
  NLS_STATUS         lStatus          = SEQU__NORMAL ;
  unsigned long      ulTestIdent      = 0 ;
  long lT;
  long lTextra;
```

```
 //////////////////////////////////////////////////////////////////////////////////////////////////
 // for the sequence unit test
 //////////////////////////////////////////////////////////////////////////////////////////////////

 if (lKernelMode == KERNEL_CHECK)
   ulTestIdent = RTEB_ORIGIN_fSEQCheck;
 else
   ulTestIdent = RTEB_ORIGIN_fSEQRunKernel;

  ///////////////////////////////////////////////////////////////////////
 // to pass the MDH to the ICE prg.
 ///////////////////////////////////////////////////////////////////////

 ss_adc1.Mdh.setEvalInfoMask( MDH_ONLINE );

 //Elijah
 /////
 ss_adc2.Mdh.setEvalInfoMask( MDH_ONLINE );
 /////

 //////////////////////////////////////////////////////////////////////////////////////////////////
 // open this event block
 //////////////////////////////////////////////////////////////////////////////////////////////////


 fRTEBInit( &(ss_voi.m_sROT_MATRIX) );
 // it is annoying that opening of an event block is always connected to
 // calculating the rotation matrix which needs to be calculated only once in
 // single slice sequences

 // this timing schemes requires these pre-conditions to be met:
 // - flat top time of exc.gradient > duration of exc. RF puls
 // - ramp down time spoiler == ramp up time slice selection
 // - flat top durarion slice selection gradients == RF pulse duration
 // - simultanously applied spoiling gradients need to be of equal duration



 lT=0;
/************************************* 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]*/
/*****************************************************************
***********************************/

  fRTEI(lT+=                0,          0,       0,       0,       0,       0,       0,&ss_osc1);
  fRTEI(lT+=  ss_osc1.lDuration,         0,       0,       0,       0,       0,       0,        0);




  if(pMrProt->preparationPulses().waterSuppression() !=
SEQ::WATER_SUPPRESSION_OFF)
  {   // water suppression
            // ... as described by Ogg et al.,JMR, B 104, p. 1-10, 1994.

            if (pMrProt->preparationPulses().waterSuppression() ==
SEQ::WATER_SATURATION ||
                       pMrProt->preparationPulses().waterSuppression() ==
SEQ::WATER_SUPPRESSION_WEAK)
                      fRTEI(lT+= 1000, &ss_ph_s_ws, &ss_rf_ws1,/*A*/ 0,0,0,0,0);
            else
                      fRTEI(lT+= 1000, &ss_ph_s_ws, 0,/*A*/ 0,0,0,0,0);

            fRTEI( lT+= ss_rf_ws1.getDuration(),  &ss_ph_n_ws, 0, /*A*/ 0,
&ss_wssp_ph,0,0,0);

            if (pMrProt->preparationPulses().waterSuppression() ==
SEQ::WATER_SATURATION ||
                       pMrProt->preparationPulses().waterSuppression() ==
SEQ::WATER_SUPPRESSION_WEAK)
                      fRTEI(lT+=
(ss_wssp_ph.getDuration()+ss_wssp_ph.getRampDownTime()), &ss_ph_s_ws,
&ss_rf_ws2,/*A*/ 0,0,0,0,0);
            else
                      fRTEI(lT+=
(ss_wssp_ph.getDuration()+ss_wssp_ph.getRampDownTime()), &ss_ph_s_ws,
0,/*A*/ 0,0,0,0,0);

            fRTEI( lT+= ss_rf_ws2.getDuration(),  &ss_ph_n_ws, 0, /*A*/ 0, 0,
&ss_wssp_ro,0,0);

            if (pMrProt->preparationPulses().waterSuppression() ==
SEQ::WATER_SATURATION ||
                       pMrProt->preparationPulses().waterSuppression() ==
SEQ::WATER_SUPPRESSION_WEAK)
```

```
                        fRTEI(lT+=
(ss_wssp_ro.getDuration()+ss_wssp_ro.getRampDownTime()), &ss_ph_s_ws,
&ss_rf_ws3,/*A*/ 0,0,0,0,0);
            else
                        fRTEI(lT+=
(ss_wssp_ro.getDuration()+ss_wssp_ro.getRampDownTime()), &ss_ph_s_ws,
0,/*A*/ 0,0,0,0,0);

            fRTEI( lT+= ss_rf_ws3.getDuration(),  &ss_ph_n_ws, 0, /*A*/ 0, 0,
0,&ss_wssp_sl, 0);
            fRTEI( lT+= (ss_wssp_sl.getTotalTime() + sl_excit_delay ), 0,0,/*A*/
0,0,0,0,0);
  }




  CheckStatusPB(lStatus = fRTEBFinish(),"fRTEBFinish [*0010*]");


  /* ------------------------------------------------------------------ */
  /* Execute SBB_OVS between last WET-pulse and CSI_SE excitation          */
  /* the scan time for 8 RSATs and 4 fixed RSats must fit within this pause*/
  /* max. pulse duration of the free RSats is 7680 us
*/
  /* ------------------------------------------------------------------ */

  block_nr =1; // we use only the RSat block number 1 for csi_se
  int lI;
  for (lI=MaxNrVarRSATS-1; lI>=0; lI--) {                                    //send
RSats in inverse order
            if (! RSat[lI].run(pMrProt, pSeqLim, pSeqExpo, &ss_voi) ) {
                              return   (RSat[lI].getNLSStatus()) ;
                     }
  }

  lTextra = lT + lScanTimeOVSSats;



  /* ------------------------------------------------------------------ */
  /* Here four fixed rsat pulses around the VOI !!                      */
  /* these RSats are only active if fully_excited_VOI option = ON          */
  /* Execute SBB_fixed_OVS between last WET-pulse or last free RSATs             */
  /* and CSI_SE excitation
                                        */
  /* the scan time for total OVS must be shorter than sl_excit_delay       */
  /* ------------------------------------------------------------------ */


  int lI2;
  for (lI2=MaxNrFixedRsats-1; lI2>=0; lI2--) {
            //send fixed RSats in inverse order
              if (! fixedRSat[lI2].run(pMrProt, pSeqLim, pSeqExpo, &ss_voi) ) {
                                return  (fixedRSat[lI2].getNLSStatus()) ;
                       }
  }
  lTextra +=  lScanTimeOVSSats_fixed;                  // OVS (rsats and fixed_rsats and
water suppression time




  fRTEBInit( &(ss_voi.m_sROT_MATRIX) );
  lT=0;


/*
cout << endl;
cout << "rot. matrix" << endl;
cout << "ss_voi->m_sROT_MATRIX.dMat[0][0]" << ss_voi-
>m_sROT_MATRIX.dMat[0][0] << endl;
cout << "ss_voi->m_sROT_MATRIX.dMat[1][0]" <<
ss_voi.m_sROT_MATRIX.dMat[1][0] << endl;
cout << "ss_voi->m_sROT_MATRIX.dMat[2][0]" <<
m_sROT_MATRIX.dMat[2][0] << endl;
cout << endl;

cout << "ss_voi->m_sROT_MATRIX.dMat[0][1]" << ss_voi-
>m_sROT_MATRIX.dMat[0][1] << endl;
cout << "ss_voi->m_sROT_MATRIX.dMat[1][1]" << ss_voi-
>m_sROT_MATRIX.dMat[1][1] << endl;
cout << "ss_voi->m_sROT_MATRIX.dMat[2][1]" << ss_voi-
>m_sROT_MATRIX.dMat[2][1] << endl;
cout << endl;
cout << "ss_voi->m_sROT_MATRIX.dMat[0][2]" << ss_voi-
>m_sROT_MATRIX.dMat[0][2] << endl;
cout << "ss_voi->m_sROT_MATRIX.dMat[1][2]" << ss_voi-
>m_sROT_MATRIX.dMat[1][2] << endl;
```

```
cout << "ss_voi->m_sROT_MATRIX.dMat[2][2]" << ss_voi-
>m_sROT_MATRIX.dMat[2][2] << endl;
*/



  // excitation

  fRTEI(lT+=  100 , 0,0,/*A*/ 0,0,&ss_grad_exc,0,0);
  fRTEI(lT+= (ss_grad_exc.getDuration() - ss_rf_exc.getDuration()), &ss_ph_s_exc,
&ss_rf_exc,0,/*A*/0,0,0,0);

  // slice select rephasing, 1st refocussing pulse



  fRTEI(lT+= (ss_rf_exc.getDuration()),
&ss_ph_n_exc,0,/*A*/0,&ss_encod_ph,&ss_encod_ro,&ss_encod_sl,0);
  fRTEI(lT+= (ss_encod_sl.getDuration()), 0,0,/*A*/0,&ss_grad_pi_ph, 0,0,0);
  fRTEI(lT+= (ss_grad_pi_ph.getRampUpTime()), &ss_ph_s_pi_ph, &ss_rf_pi_ph,
0,/*A*/0,0,0,0 );
  fRTEI(lT+= (ss_rf_pi_ph.getDuration()), &ss_ph_n_pi_ph, 0,/*A*/0,
&ss_sp1_ph,&ss_sp1_ro,&ss_sp1_sl,0);




  // spectral suppression mega pulse 1
  if( pMrProt->spectroscopy().SpectralSuppression() !=  SEQ::SPEC_SUPPR_NONE
)
  {
            fRTEI(lT+= (ss_sp1_sl.getDuration() + ss_sp1_sl.getRampDownTime() +
sl_trueTE1),    0,        0,        /*A*/0,          0,        &ss_mega1_ro,
          &ss_mega1_sl,           0);
            fRTEI(lT+= (ss_mega1_sl.getDuration() +
ss_mega1_sl.getRampDownTime()),         &ss_ph_s_ws, &ss_rf_mega1,   /*A*/0
,0,0,0,0);
            fRTEI(lT+= (fSDSRoundUpGRT(ss_rf_mega1.getDuration())),
0,  0,/*A*/0,0,&ss_mega2_ro,&ss_mega2_sl,0);

            lT+= (long) (ss_mega1_sl.getDuration() +
ss_mega1_sl.getRampDownTime() -
                      (ss_sp1_sl.getDuration() + ss_sp1_sl.getRampDownTime() +
sl_trueTE1));
  }


   // 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);
  fRTEI(lT+= (ss_sp2_sl.getDuration()), 0,0,/*A*/0,0,0,&ss_grad_pi_sl, 0);
  fRTEI(lT+= (ss_grad_pi_sl.getRampUpTime()), &ss_ph_s_pi_sl, &ss_rf_pi_sl,
0,/*A*/0,0,0,0 );
  fRTEI(lT+= (ss_rf_pi_sl.getDuration()), &ss_ph_n_pi_sl, 0,/*A*/0,
&ss_sp2_ph,&ss_sp2_ro,&ss_sp2_sl,0);



  // spectral suppression mega pulse 2
  if( pMrProt->spectroscopy().SpectralSuppression() !=  SEQ::SPEC_SUPPR_NONE
)
  {
            fRTEI(lT+= (ss_sp2_sl.getDuration() + ss_sp2_sl.getRampDownTime()),
0,0,/*A*/0,&ss_mega1_ph,&ss_mega1_ro,0,0);
            fRTEI(lT+= (ss_mega2_sl.getDuration() +
ss_mega2_sl.getRampDownTime()),&ss_ph_s_ws, &ss_rf_mega2,/*A*/0,0,0,0,0);
            fRTEI(lT+= (fSDSRoundUpGRT(ss_rf_mega2.getDuration()))               ,
0,0,/*A*/0,&ss_mega2_ph,&ss_mega2_ro,0,0);
            lT+= (long) (ss_mega2_sl.getDuration() +
ss_mega2_sl.getRampDownTime() -
                        (ss_sp2_sl.getDuration() + ss_sp2_sl.getRampDownTime()));
  }

  // 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_adc1,0,&ss_adc1,0,0,0,0);
//Elijah -  next line was added because phase of 1st adc was set but not reset in
original csi code
/////
fRTEI(lT+=
ss_adc1.getRoundedDuration(GRAD_RASTER_TIME),&ss_ph_n_adc1,0,0,0,0,0,0);
/////

//Elijah
///// 3rd refocussing pulse and 2nd ADC
  fRTEI(lT+=(sl_aqu_fill_after + sl_fill_before3rdRF),
              0,0,/*A*/0,&ss_sp2_ph,&ss_sp2_ro,&ss_sp2_sl,0);
  //  fRTEI(lT+=(ss_sp2_sl.getTotalTime()),&ss_ph_s_pi_sl, &ss_rf_pi_sl,
0,/*A*/0,0,0,0 );
    fRTEI(lT+= (ss_sp2_sl.getDuration()), 0, 0, 0, 0, 0, &ss_grad_pi_sl, 0);
    fRTEI(lT+= (ss_grad_pi_sl.getRampUpTime()), &ss_ph_s_pi_sl, &ss_rf_pi_sl,
0,/*A*/0,0,0,0 );




  fRTEI(lT+=(ss_rf_pi_sl.getDuration()), &ss_ph_n_pi_sl, 0,/*A*/0,
&ss_sp2_ph,&ss_sp2_ro,&ss_sp2_sl,0);

  fRTEI(lT+=(ss_sp2_sl.getDuration() + ss_sp2_sl.getRampDownTime() +
sl_aqu_fill_before2), &ss_ph_s_adc2,0,&ss_adc2,0,0,0,0);
  fRTEI(lT+=(1000 + ss_adc2.getRoundedDuration(GRAD_RASTER_TIME) +
sl_aqu_fill_after2),&ss_ph_n_adc2,0,0, &ss_finsp_ro, &ss_finsp_ph, &ss_finsp_sl, 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 );
 //This spoiling is used by Elijah instead of the default preceeding one
 fRTEI(lT+=(ss_finsp_sl.getDuration() + ss_finsp_sl.getRampDownTime()),
0,0,0,0,0,0,0 );


  // TR fill
  fRTEI(lT+=  (pMrProt->tr()[0] - lT  - lTextra ), 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);
}

/*
static void print_slicepos( char *nm, sSLICE_POS *sp )
{
        fprintf( stdout, "\n slice pos. of %s", nm );
        fprintf( stdout, "\n off-centre readout %g", sp->getSliceOffCenterRO() );
        fprintf( stdout, "\n off-centre phase %g", sp->getSliceOffCenterPE() );
        fprintf( stdout, "\n off-centre slice %g", sp->getSliceShift() );
        fprintf( stdout, "\n rot. matrix");
        fprintf( stdout, "\n %1.6f \t %1.6f \t %1.6f",
                        sp->m_sROT_MATRIX.dMat[0][0], sp-
>m_sROT_MATRIX.dMat[0][1], sp->m_sROT_MATRIX.dMat[0][2] );
        fprintf( stdout, "\n %1.6f \t %1.6f \t %1.6f",
                        sp->m_sROT_MATRIX.dMat[1][0], sp-
>m_sROT_MATRIX.dMat[1][1], sp->m_sROT_MATRIX.dMat[1][2] );
        fprintf( stdout, "\n %1.6f \t %1.6f \t %1.6f",
                        sp->m_sROT_MATRIX.dMat[2][0], sp-
>m_sROT_MATRIX.dMat[2][1], sp->m_sROT_MATRIX.dMat[2][2] );
        fprintf( stdout, "\n\n" );
}
*/


  /* --------------------------------------------------------------------------------    */
  /*  function add_refoc (long frequ_offset, long duration, double attenuation, float
*arr)  */
  /*
                                                                                        */
  /*  to any pulse with 512 sample points,  the pulse shape  SE5120A180.cpp with an
*/
  /*  optional frequence offset is added
                                                          */
  /*  pulses with multiple frequence bands can be created
                                              */
  /* ---------------------------------------------------------------------------------   */



        double add_refoc( long frequ_offset, long duration, double attenuation, float
*arr )
        {
                // add a MAO pulse to the compex array arr of size sz
```

```
            const long SZ = 512;
            const double M_2PI = 2 * 3.14159265359;
            //float ampl[SZ], pha[SZ];


            //    consistent with normal pulse preperation: the phase offset is
positive for neg. delta frequency
            //    (use neg. values for the delta frequency in the UI)
            //    for 1.5 und 3 Tesla exists a socalled  Kehrladenband: frequence
axis is turned arround neg -> positiv;
            //    attention: for different nuclei this turn arround of the frequency
axis might not be correct !!!
            double phase;
            double phase_offset =  M_2PI * (double) frequ_offset * (double)
duration *  1E-6; // the entire frequ. shift range
            double phase_increment = phase_offset / (double) SZ; // the frequ.
shift between 2 pulse samples
            phase_offset *= (- .5); // running from - phase _offset/2 .... +
phase_offset/2
            long j;


            // include numerical values of the SE5120A180 pulse
            #include "MrServers/MrSpecAcq/spectro_ui/SE5120A180.cpp"




            phase = phase_offset;
            double sum = 0;
            for( j=0; j<SZ; j++ ){

                arr[j*2]   += (float)(attenuation * ampl[j] * cos( phase + pha[j]
)); // add the pulse to the array, real part
                sum += ampl[j] * cos( pha[j] ); // do NOT take the phase due to
the frequ. shift into account
                arr[j*2+1] += (float)(attenuation * ampl[j] * sin( phase + pha[j]
)); // add the pulse to the array, imaginary part

                phase += phase_increment;
            }


           // for IDL:
            /*
            FILE *fp;
```

```
                  long i;
                  //  fp = fopen( "D\\RSI\\IDL55\\libjan\\pulseshape\\megacalc.pro",
"w" );
                  fp = fopen( "megacalc.pro", "w" );
                  fprintf(fp, "pro megacalc, complarr   \n");
                  fprintf( fp, "ampl = fltarr(512) \npha = fltarr (512)");
                  for( i=0; i<SZ; i++ )
                          fprintf( fp, " \nampl[%d] = %f;  \npha[%d] = %f;", i, arr[i*2],
i, arr[i*2+1]);
                  fprintf( fp, "\nplot, pha \noplot, ampl");
                  fprintf( fp, "\ncomplarr = complex(temporary(ampl) ,temporary(pha))
\nend" );

                  fclose( fp );
                  */


                  return sum; // due to pulse definition, this sum is normalized
         }






 /* ---------------------------------------------------------------------------- */
 /* function double arr2sample_arr( long sz, float *arr, sSample *sample_arr )     */
 /*
                                                             */
 /* pulse array conversion
                                                          */
 /* from x + iy to -> abs exp( i pha )
                                            */
 /*
                                                                          */
 /* ----------------------------------------------------------------------------          */



        double arr2sample_arr( long sz, float *arr, sSample *sample_arr )
        {
                // from x + iy to -> abs exp( i pha )
                double M_2PI = 2 * 3.14159265359;
                double max = 0;
        long i;
        for( i=0; i<sz; i++ ){
            sample_arr[i].flAbs = (float) sqrt( arr[i*2] * arr[i*2] +
arr[i*2+1] * arr[i*2+1] );
            if( sample_arr[i].flAbs > max )
                max = sample_arr[i].flAbs;
            sample_arr[i].flPha = (float)(M_PI + atan2( arr[i*2+1], arr[i*2]
) );
            if( sample_arr[i].flPha < 0 ){ // -180 ... + 180 --> 0 ... 360
                sample_arr[i].flPha = (float)(sample_arr[i].flPha +
M_2PI);
            }
        }

        // normalize and compute complex amplitude sum
        double sum=0;
        max = 1./max;
        for( i=0; i<sz; i++ ){

            sample_arr[i].flAbs *= (float)max;
            sum += sample_arr[i].flAbs * (cos( sample_arr[i].flPha ) + sin(
sample_arr[i].flPha ) );
            // this sum is not really helpful for computing a flipangle if the
pulse is complex e.g. due to a frequ.shift
        }

        // write pulse to a PTA file
        /*
        FILE *fp;
        fp = fopen( "stefanpuls.pta", "w" );
        fprintf( fp, "\nPOWERINT:    %f", sum );
        for( i=0; i<sz; i++ )
            fprintf( fp, "\n%f\t%f\t;(%d)", sample_arr[i].flAbs,
sample_arr[i].flPha, i );
        fclose( fp );
        */

        return sum;
    }
```

Bibliography

Ackerstaff, E., K. Glunde, and Z. M. Bhujwalla, 2003, Choline phospholipid metabolism: A target in cancer cells?: Journal of Cellular Biochemistry, v. 90, p. 525-533.

Alfano, B., A. Brunetti, E. M. Covelli, M. Quarantelli, M. R. Panico, A. Ciarmiello, and M. Salvatore, 1997, Unsupervised, automated segmentation of the normal brain using a multispectral relaxometric magnetic resonance approach: Magnetic Resonance in Medicine, v. 37, p. 84-93.

Ashwal, S., B. Holshouser, K. Tong, T. Serna, R. Osterdock, M. Gross, and D. Kido, 2004, Proton spectroscopy detected myoinositol in children with traumatic brain injury: Pediatric Research, v. 56, p. 630-638.

Auld, K. L., S. Ashwal, B. A. Holshouser, L. G. Tomasi, R. M. Perkin, B. D. Ross, and D. B. Hinshaw, 1995, PROTON MAGNETIC-RESONANCE SPECTROSCOPY IN CHILDREN WITH ACUTE CENTRAL-NERVOUS-SYSTEM INJURY: Pediatric Neurology, v. 12, p. 323-334.

Babikian, T., M. C. Freier, S. Ashwal, M. L. Riggs, T. Burley, and B. A. Holshouser, 2006, MR spectroscopy: Predicting long-term neuropsychological outcome following pediatric TBI: Journal of Magnetic Resonance Imaging, v. 24.

Baslow, M. H., 2003, N-acetylaspartate in the vertebrate brain: Metabolism and function: Neurochemical Research, v. 28.

Bay, E., C. Kalpakjian, and B. Giordani, 2012, Determinants of subjective memory complaints in community-dwelling adults with mild-to-moderate traumatic brain injury: Brain Injury, v. 26, p. 941-949.

Bayly, P. V., T. S. Cohen, E. P. Leister, D. Ajo, E. C. Leuthardt, and G. M. Genin, 2005, Deformation of the human brain induced by mild acceleration: Journal of Neurotrauma, v. 22, p. 845-856.

Beaumont, A., A. Marmarou, K. Hayasaki, P. Barzo, P. Fatouros, F. Corwin, C. Marmarou, and J. Dunbar, 2000, The permissive nature of blood brain barrier (BBB) opening in edema formation following traumatic brain injury: Acta neurochirurgica. Supplement, v. 76.

Behar, K. L., D. L. Rothman, K. F. Petersen, M. Hooten, R. Delaney, O. A. C. Petroff, G. I. Shulman, V. Navarro, I. L. Petrakis, D. S. Charney, and J. H. Krystal, 1999, Preliminary evidence of low cortical GABA levels in localized H-1-MR spectra of alcohol-dependent and hepatic encephalopathy patients: American Journal of Psychiatry, v. 156, p. 952-954.

Behrens, T. E. J., H. Johansen-Berg, M. W. Woolrich, S. M. Smith, C. A. M. Wheeler-Kingshott, P. A. Boulby, G. J. Barker, E. L. Sillery, K. Sheehan, O. Ciccarelli, A. J. Thompson, J. M. Brady, and P. M. Matthews, 2003, Non-invasive mapping of connections between human thalamus and cortex using diffusion imaging: Nature Neuroscience, v. 6.

Bendszus, M., H. G. Weijers, G. Wiesbeck, M. Warmuth-Metz, A. J. Bartsch, S. Engels, J. Boning, and L. Solymosi, 2001, Sequential MR imaging and proton

MR spectroscopy in patients who underwent recent detoxification for chronic alcoholism: Correlation with clinical and neuropsychological data: American Journal of Neuroradiology, v. 22.

Berman, J. I., M. S. Berger, P. Mukherjee, and R. G. Henry, 2004, Diffusion-tensor imaging-guided tracking of fibers of the pyramidal tract combined with intraoperative cortical stimulation mapping in patients with gliomas: Journal of Neurosurgery, v. 101, p. 66-72.

Bizzi, A., A. M. Ulug, T. O. Crawford, T. Passe, M. Bugiani, R. N. Bryan, and P. B. Barker, 2001, Quantitative proton MR spectroscopic imaging in acute disseminated encephalomyelitis: American Journal of Neuroradiology, v. 22.

Bleiberg, J., A. N. Cernich, K. Cameron, W. Y. Sun, K. Peck, J. Ecklund, D. Reeves, J. Uhorchak, M. B. Sparling, and D. L. Warden, 2004, Duration of cognitive impairment after sports concussion: Neurosurgery, v. 54.

Boake, C., S. R. McCauley, H. S. Levin, C. Pedroza, C. E. Contant, J. X. Song, S. A. Brown, H. Goodman, S. I. Brundage, and P. J. Diaz-Marchan, 2005, Diagnostic criteria for postconcussional syndrome after mild to moderate traumatic brain injury: Journal of Neuropsychiatry and Clinical Neurosciences, v. 17, p. 350-356.

Brooks, W. M., S. D. Friedman, and C. Gasparovic, 2001, Magnetic resonance spectroscopy in traumatic brain injury: Journal of Head Trauma Rehabilitation, v. 16, p. 149-164.

Brooks, W. M., C. A. Stidley, H. Petropoulos, R. E. Jung, D. C. Weers, S. D. Friedman, M. A. Barlow, W. L. Sibbitt, and R. A. Yeo, 2000, Metabolic and cognitive response to human traumatic brain injury: A quantitative proton magnetic resonance study: Journal of Neurotrauma, v. 17.

Brown, T. R., B. M. Kincaid, and K. Ugurbil, 1982, NMR chemical shift imaging in three dimensions: Proceedings of the National Academy of Sciences of the United States of America, v. 79, p. 3523-6.

Burges, C. J. C., 1998, A tutorial on Support Vector Machines for pattern recognition: Data Mining and Knowledge Discovery, v. 2, p. 121-167.

Burri, R., C. Steffen, and N. Herschkowitz, 1991, N-ACETYL-L-ASPARTATE IS A MAJOR SOURCE OF ACETYL GROUPS FOR LIPID-SYNTHESIS DURING RAT-BRAIN DEVELOPMENT: Developmental Neuroscience, v. 13.

Carpenter, G. A., and S. Grossberg, 1988, THE ART OF ADAPTIVE PATTERN-RECOGNITION BY A SELF-ORGANIZING NEURAL NETWORK: Computer, v. 21, p. 77-88.

Cassidy, J. D., L. J. Carroll, P. M. Peloso, J. Borg, H. von Holst, L. Holm, J. Kraus, and V. G. Coronado, 2004, Incidence, risk factors and prevention of mild traumatic brain injury: Results of the WHO Collaborating Centre Task Force on Mild Traumatic Brain Injury: Journal of Rehabilitation Medicine, v. 36, p. 28-60.

Cecil, K. M., E. C. Hills, E. Sandel, D. H. Smith, T. K. McIntosh, L. J. Mannon, G. P. Sinson, L. J. Bagley, R. I. Grossman, and R. E. Lenkinski, 1998a, Proton magnetic resonance spectroscopy for detection of axonal injury in the splenium of the corpus callosum of brain-injured patients: Journal of Neurosurgery, v. 88, p. 795-801.

Cecil, K. M., R. E. Lenkinski, D. F. Meaney, T. K. McIntosh, and D. H. Smith, 1998b, High-field proton magnetic resonance spectroscopy of a swine model for axonal injury: Journal of Neurochemistry, v. 70, p. 2038-2044.

Chakraborty, G., P. Mekala, D. Yahya, G. S. Wu, and R. W. Ledeen, 2001, Intraneuronal N-acetylaspartate supplies acetyl groups for myelin lipid synthesis: evidence for myelin-associated aspartoacylase: Journal of Neurochemistry, v. 78.

Chamard, E., M. Lassonde, L. Henry, J. Tremblay, Y. Boulanger, L. De Beaumont, and H. Theoret, 2013, Neurometabolic and microstructural alterations following a sports-related concussion in female athletes: Brain Injury, v. 27, p. 1038-1046.

Chang, L., T. Ernst, R. E. Poland, and D. J. Jenden, 1996, In vivo proton magnetic resonance spectroscopy of the normal aging human brain: Life Sciences, v. 58.

Clark, J. B., 1998, N-acetyl aspartate: A marker for neuronal loss or mitochondrial dysfunction: Developmental Neuroscience, v. 20.

Cohen, B. A., M. Inglese, H. Rusinek, J. S. Babb, R. I. Grossman, and O. Gonen, 2007, Proton MR spectroscopy and MRI-volumetry in mild traumatic brain injury: American Journal of Neuroradiology, v. 28.

Condon, B., D. Oluoch-Olunya, D. Hadley, G. Teasdale, and A. Wagstaff, 1998, Early 1H magnetic resonance spectroscopy of acute head injury: Four cases: Journal of Neurotrauma, v. 15, p. 563-571.

Cordoba, J., J. Alonso, A. Rovira, C. Jacas, F. Sanpedro, L. Castells, V. Vargas, C. Margarit, J. Kulisewsky, R. Esteban, and J. Guardia, 2001, The development of low-grade cerebral edema in cirrhosis is supported by the evolution of H-1-magnetic resonance abnormalities after liver transplantation: Journal of Hepatology, v. 35, p. 598-604.

Davie, C. A., C. P. Hawkins, G. J. Barker, A. Brennan, P. S. Tofts, D. H. Miller, and W. I. McDonald, 1993, DETECTION OF MYELIN BREAKDOWN PRODUCTS BY PROTON MAGNETIC-RESONANCE SPECTROSCOPY: Lancet, v. 341.

De Graaf, R.A., 2007. In vivo NMR Spectroscopy: Principles and Techniques. 2nd ed. Wiley

Demougeot, C., N. Bertrand, A. Prigent-Tessier, P. Garnier, C. Mossiat, M. Giroud, C. Marie, and A. Beley, 2003, Reversible loss of N-acetyl-aspartate in rats subjected to long-term focal cerebral ischemia: Journal of Cerebral Blood Flow and Metabolism, v. 23.

Demougeot, C., P. Garnier, C. Mossiat, N. Bertrand, M. Giroud, A. Beley, and C. Marie, 2001, N-Acetylaspartate, a marker of both cellular dysfunction and neuronal loss: its relevance to studies of acute brain injury: Journal of Neurochemistry, v. 77.

Destefano, N., P. M. Matthews, and D. L. Arnold, 1995, REVERSIBLE DECREASES IN N-ACETYLASPARTATE AFTER ACUTE BRAIN INJURY: Magnetic Resonance in Medicine, v. 34.

Devos, A., L. Lukas, J. A. K. Suykens, L. Vanhamme, A. R. Tate, F. A. Howe, C. Majos, A. Moreno-Torres, M. van der Graaf, C. Arus, and S. Van Huffel, 2004, Classification of brain tumours using short echo time H-1 MR spectra: Journal of Magnetic Resonance, v. 170, p. 164-175.

Devos, A., A. W. Simonetti, M. van der Graaf, L. Lukas, J. A. K. Suykens, L. Vanhamme, L. M. C. Buydens, A. Heerschap, and S. Van Huffel, 2005, The use of multivariate MR imaging intensities versus metabolic data from MR spectroscopic imaging for brain tumour classification: Journal of Magnetic Resonance, v. 173, p. 218-228.

Dreher, W., and D. Leibfritz, 1994, DOUBLE-ECHO MULTISLICE PROTON SPECTROSCOPIC IMAGING USING HADAMARD SLICE ENCODING: Magnetic Resonance in Medicine, v. 31, p. 596-600.

Dreher, W., and D. Leidfritz, 1995, PARAMETRIC MULTIECHO PROTON SPECTROSCOPIC IMAGING - APPLICATION TO THE RAT-BRAIN IN-VIVO: Magnetic Resonance Imaging, v. 13, p. 753-761.

Dudoit, S., J. Fridlyand, and T. P. Speed, 2002, Comparison of discrimination methods for the classification of tumors using gene expression data: Journal of the American Statistical Association, v. 97, p. 77-87.

Durazzo, T. C., S. Gazdzinski, P. Banys, and D. J. Meyerhoff, 2004, Cigarette smoking exacerbates chronic alcohol-induced brain damage: A preliminary metabolite imaging study: Alcoholism-Clinical and Experimental Research, v. 28.

Duyn, J. H., and C. T. W. Moonen, 1993, FAST PROTON SPECTROSCOPIC IMAGING OF HUMAN BRAIN USING MULTIPLE SPIN-ECHOES: Magnetic Resonance in Medicine, v. 30, p. 409-414.

Dydak, U., D. Meier, R. Lamerichs, and P. Boesiger, 2006, Trading spectral separation at 3T for acquisition speed in multi spin-echo spectroscopic imaging: American Journal of Neuroradiology, v. 27, p. 1441-1446.

Ebel, A., W. Dreher, and D. Leibfritz, 2006, Effects of zero-filling and apodization on spectral integrals in discrete Fourier-transform spectroscopy of noisy data: Journal of Magnetic Resonance, v. 182, p. 330-338.

Fann, J. R., W. J. Katon, J. M. Uomoto, and P. C. Esselman, 1995, PSYCHIATRIC-DISORDERS AND FUNCTIONAL DISABILITY IN OUTPATIENTS WITH TRAUMATIC BRAIN INJURIES: American Journal of Psychiatry, v. 152, p. 1493-1499.

Frahm, J., K. D. Merboldt, and W. Hanicke, 1987, LOCALIZED PROTON SPECTROSCOPY USING STIMULATED ECHOES: Journal of Magnetic Resonance, v. 72, p. 502-508.

Freeman, T. W., D. Cardwell, C. N. Karson, and R. A. Komoroski, 1998, In vivo proton magnetic resonance spectroscopy of the medial temporal lobes of subjects with combat-related posttraumatic stress disorder: Magnetic Resonance in Medicine, v. 40.

Friedman, S. D., W. M. Brooks, R. E. Jung, S. J. Chiulli, J. H. Sloan, B. T. Montoya, B. L. Hart, and R. A. Yeol, 1999, Quantitative proton MRS predicts outcome after traumatic brain injury: Neurology, v. 52.

Friedman, S. D., W. M. Brooks, R. E. Jung, B. L. Hart, and R. A. Yeo, 1998, Proton MR spectroscopic findings correspond to neuropsychological function in traumatic brain injury: American Journal of Neuroradiology, v. 19.

Fushiki, T., 2011, Estimation of prediction error by using K-fold cross-validation: Statistics and Computing, v. 21, p. 137-146.

Garcia-Gomez, J. M., S. Tortajada, C. Vidal, M. Julia-Sape, J. Luts, A. Moreno-Torres, S. Van Huffel, C. Arus, and M. Robles, 2008, The effect of combining two echo times in automatic brain tumor classification by MRS: Nmr in Biomedicine, v. 21, p. 1112-1125.

Garnett, M. R., A. M. Blamire, R. G. Corkill, T. A. D. Cadoux-Hudson, B. Rajagopalan, and P. Styles, 2000a, Early proton magnetic resonance spectroscopy in normal-appearing brain correlates with outcome in patients following traumatic brain injury: Brain, v. 123.

Garnett, M. R., A. M. Blamire, B. Rajagopalan, P. Styles, and T. A. D. Cadoux-Hudson, 2000b, Evidence for cellular damage in normal-appearing white matter correlates with injury severity in patients following traumatic brain injury - A magnetic resonance spectroscopy study: Brain, v. 123, p. 1403-1409.

Gasparovic, C., R. Yeo, M. Mannell, J. Ling, R. Elgie, J. Phillips, D. Doezema, and A. R. Mayer, 2009, Neurometabolite Concentrations in Gray and White Matter in Mild Traumatic Brain Injury: An H-1-Magnetic Resonance Spectroscopy Study: Journal of Neurotrauma, v. 26.

George, E.O., Roys, S., Zhuo J., Sours, C., Janowich, J., Stoica, S., Rosenberg, J., Gullapalli, R.P, 2013, Longitudinal Evaluation of Mild Traumatic Brain Injury: A 1H-MRS Study. 21st Annual Proceedings of the International Society of Magnetic Resonance in Medicine

George, E.O., Roys, S., Zhuo J., Gullapalli, R.P, 2013, Dual-Echo Magnetic Resonance Spectroscopy Imaging. 21st Annual Proceedings of the International Society of Magnetic Resonance in Medicine

George, E.O., Roys, S., Sours, C., Rosenberg, J., Zhuo J., Kathirkamanthan, S., Gullapalli, R.P, 2014, Longitudinal Evaluation of Mild Traumatic Brain Injury: A 1H-MRS Study. Journal of Neurotrauma

George, E.O., Roys, S., Zhuo J., Sours, Rosenberg, J., Gullapalli, R.P, 2014, Predicting symptomatic outcome in mild traumatic brain injury with support vector machines: a 1H-MRS Study. 22nd Annual Proceedings of the International Society of Magnetic Resonance in Medicine

George, E.O., Roys, S., Gullapalli, R.P, 2014, Enhancing quantitation precision in multiecho spectroscopic imaging. 22nd Annual Proceedings of the International Society of Magnetic Resonance in Medicine

Georgiadis, P., S. Kostopoulos, D. Cavouras, D. Glotsos, I. Kalatzis, K. Sifaki, M. Malamas, E. Solomou, and G. Nikiforidis, 2011, Quantitative combination of volumetric MR imaging and MR spectroscopy data for the discrimination of meningiomas from metastatic brain tumors by means of pattern recognition: Magnetic Resonance Imaging, v. 29, p. 525-535.

Gideon, P., O. Henriksen, B. Sperling, P. Christiansen, T. S. Olsen, H. S. Jorgensen, and P. Arliensoborg, 1992, EARLY TIME COURSE OF N-

ACETYLASPARTATE, CREATINE AND PHOSPHOCREATINE, AND COMPOUNDS CONTAINING CHOLINE IN THE BRAIN AFTER ACUTE STROKE - A PROTON MAGNETIC-RESONANCE SPECTROSCOPY STUDY: Stroke, v. 23, p. 1566-1572.

Glodzik-Sobanska, L., A. Slowik, P. McHugh, B. Sobiecka, J. Kozub, K. E. Rich, A. Urbanik, and A. Szczudlik, 2006, Single voxel proton magnetic resonance spectroscopy in post-stroke depression: Psychiatry Research-Neuroimaging, v. 148, p. 111-120.

Govind, V., S. Gold, K. Kaliannan, G. Saigal, S. Falcone, K. L. Arheart, L. Harris, J. Jagid, and A. A. Maudsley, 2010, Whole-Brain Proton MR Spectroscopic Imaging of Mild-to-Moderate Traumatic Brain Injury and Correlation with Neuropsychological Deficits: Journal of Neurotrauma, v. 27.

Govindaraju, V., G. E. Gauger, G. T. Manley, A. Ebel, M. Meeker, and A. A. Maudsley, 2004, Volumetric proton spectroscopic imaging of mild traumatic brain injury: American Journal of Neuroradiology, v. 25.

Graham, G. D., A. M. Blamire, A. M. Howseman, D. L. Rothman, P. B. Fayad, L. M. Brass, O. A. C. Petroff, R. G. Shulman, and J. W. Prichard, 1992, PROTON MAGNETIC-RESONANCE SPECTROSCOPY OF CEREBRAL LACTATE AND OTHER METABOLITES IN STROKE PATIENTS: Stroke, v. 23, p. 333-340.

Graham, G. D., A. M. Blamire, D. L. Rothman, L. M. Brass, P. B. Fayad, O. A. C. Petroff, and J. W. Prichard, 1993, EARLY TEMPORAL VARIATION OF CEREBRAL METABOLITES AFTER HUMAN STROKE - A PROTON MAGNETIC-RESONANCE SPECTROSCOPY STUDY: Stroke, v. 24.

Greco, T., and M. L. Prins, 2013, Traumatic Brain Injury and Diet: Journal of Child Neurology, v. 28, p. 983-988.

Greenfield, J., Love, S., Louis, D.N., Ellison, D.W., 2008, Greenfield's Neuropathology. Volume 1. 8th ed. London: Hodder Arnold.

Harris, J. L., H.-W. Yeh, I.-Y. Choi, P. Lee, N. E. Berman, R. H. Swerdlow, S. C. Craciunas, and W. M. Brooks, 2012, Altered neurochemical profile after traumatic brain injury: H-1-MRS biomarkers of pathological mechanisms: Journal of Cerebral Blood Flow and Metabolism, v. 32, p. 2122-2134.

Hattingen, E., J. Magerkurth, U. Pilatus, A. Mozer, C. Seifried, H. Steinmetz, F. Zanella, and R. Hilker, 2009, Phosphorus and proton magnetic resonance spectroscopy demonstrates mitochondrial dysfunction in early and advanced Parkinson's disease: Brain, v. 132.

Hattingen, E., P. Raab, K. Franz, F. E. Zanella, H. Lanfermann, and U. Pilatus, 2008, Myo-Inositol: a marker of reactive astrogliosis in glial tumors?: Nmr in Biomedicine, v. 21, p. 233-241.

Henry, L. C., S. Tremblay, Y. Boulanger, D. Ellemberg, and M. Lassonde, 2010, Neurometabolic Changes in the Acute Phase after Sports Concussions Correlate with Symptom Severity: Journal of Neurotrauma, v. 27.

Henry, L. C., S. Tremblay, S. Leclerc, A. Khiat, Y. Boulanger, D. Ellemberg, and M. Lassonde, 2011, Metabolic changes in concussed American football players during the acute and chronic post-injury phases: Bmc Neurology, v. 11.

Herminghaus, S., U. Pilatus, W. Moller-Hartmann, P. Raab, H. Lanfermann, W. Schlote, and F. E. Zanella, 2002, Increased choline levels coincide with enhanced proliferative activity of human neuroepithelial brain tumors: Nmr in Biomedicine, v. 15, p. 385-392.

Hillary, F. G., W. C. Liu, H. M. Genova, A. H. Maniker, K. Kepler, B. D. Greenwald, B. M. Cortese, A. Homnick, and J. Deluca, 2007, Examining lactate in severe TBI using proton magnetic resonance spectroscopy: Brain Injury, v. 21, p. 981-991.

Hoge, C. W., D. McGurk, J. L. Thomas, A. L. Cox, C. C. Engel, and C. A. Castro, 2008, Mild traumatic brain injury in US Soldiers returning from Iraq: New England Journal of Medicine, v. 358.

Holshouser, B. A., S. Ashwal, S. Shu, and D. B. Hinshaw, 2000, Proton MR spectroscopy in children with acute brain injury: Comparison of short and long echo time acquisitions: Jmri-Journal of Magnetic Resonance Imaging, v. 11, p. 9-19.

Hoofien, D., E. Vakil, A. Gilboa, P. J. Donovick, and O. H. R. Barak, 2002, Comparison of the predictive power of socioeconomic variables, severity of injury and age on long-term outcome of traumatic brain injury: sample-specific variables versus factors as predictors: Brain Injury, v. 16, p. 9-27.

Hope, P. L., E. B. Cady, P. S. Tofts, P. A. Hamilton, A. M. D. Costello, D. T. Delpy, A. Chu, E. O. R. Reynolds, and D. R. Wilkie, 1984, CEREBRAL ENERGY-METABOLISM STUDIED WITH PHOSPHORUS NMR-SPECTROSCOPY IN NORMAL AND BIRTH-ASPHYXIATED INFANTS: Lancet, v. 2, p. 366-370.

Inglese, M., B. S. Y. Li, H. Rusinek, J. S. Babb, R. I. Grossman, and O. Gonen, 2003, Diffusely elevated cerebral choline and creatine in relapsing-remitting multiple sclerosis: Magnetic Resonance in Medicine, v. 50.

Inglese, M., S. Makani, G. Johnson, B. A. Cohen, J. A. Silver, O. Gonen, and R. I. Grossman, 2005, Diffuse axonal injury in mild traumatic brain injury: a diffusion tensor imaging study: Journal of Neurosurgery, v. 103, p. 298-303.

Ivins, B. J., R. Kane, and K. A. Schwab, 2009, Performance on the Automated Neuropsychological Assessment Metrics in a Nonclinical Sample of Soldiers Screened for Mild TBI After Returning From Iraq and Afghanistan: A Descriptive Analysis: Journal of Head Trauma Rehabilitation, v. 24.

Johnson, M. D., and G. A. Ojemann, 2000, The role of the human thalamus in language and memory: Evidence from electrophysiological studies: Brain and Cognition, v. 42, p. 218-230.

Kamada, K., K. Houkin, Y. Iwasaki, H. Abe, and T. Kashiwaba, 1994, In vivo proton magnetic resonance spectroscopy for metabolic changes of human brain edema: Neurologia medico-chirurgica, v. 34, p. 676-81.

Kane, R. L., T. Roebuck-Spencer, P. Short, M. Kabat, and J. Wilken, 2007, Identifying and monitoring cognitive deficits in clinical populations using Automated Neuropsychological Assessment Metrics (ANAM) tests: Archives of Clinical Neuropsychology, v. 22.

Kelm, B. M., B. H. Menze, C. M. Zechmann, K. T. Baudendistel, and F. A. Hamprecht, 2007, Automated estimation of tumor probability in prostate magnetic resonance spectroscopic imaging: Pattern recognition vs quantification: Magnetic Resonance in Medicine, v. 57, p. 150-159.

Keltner, J. R., L. L. Wald, J. D. Christensen, L. C. Maas, C. M. Moore, B. M. Cohen, and P. F. Renshaw, 1996, A technique for detecting GABA in the human brain with PRESS localization and optimized refocusing spectral editing radiofrequency pulses: Magnetic Resonance in Medicine, v. 36, p. 458-461.

Kennedy, M. R. T., J. R. Wozniak, R. L. Muetzel, B. A. Mueller, H.-H. Chiou, K. Pantekoek, and K. O. Lim, 2009, White matter and neurocognitive changes in adults with chronic traumatic brain injury: Journal of the International Neuropsychological Society, v. 15, p. 130-136.

Kiefer, A. P., V. Govindaraju, G. B. Matson, M. W. Weiner, and A. A. Maudsley, 1998, Multiple-echo proton spectroscopic imaging using time domain parametric spectral analysis: Magnetic Resonance in Medicine, v. 39, p. 528-538.

Kierans, A. S., V. Kirov, I. I. Gonen, O. Haemer, G. Nisenbaum, E. Babb, J. Grossman, R.I. and W. L. Yvonne, 2014,Myoinositol and glutamate complex neurometabolite abnormality after mild traumatic brain injury: Neurology

King, N. S., S. Crawford, F. J. Wenden, N. E. G. Moss, and D. T. Wade, 1995, THE RIVERMEAD POST CONCUSSION SYMPTOMS QUESTIONNAIRE - A MEASURE OF SYMPTOMS COMMONLY EXPERIENCED AFTER HEAD-INJURY AND ITS RELIABILITY: Journal of Neurology, v. 242, p. 587-592.

Kirov, I., L. Fleysher, J. S. Babb, J. M. Silver, R. I. Grossman, and O. Gonen, 2007, Characterizing 'mild' in traumatic brain injury with proton MR spectroscopy in the thalamus: Initial findings: Brain Injury, v. 21.

Kirov, I. I., A. Tal, J. S. Babb, Y. W. Lui, R. I. Grossman, and O. Gonen, 2013, Diffuse axonal injury in mild traumatic brain injury: a 3D multivoxel proton MR spectroscopy study: Journal of Neurology, v. 260, p. 242-252.

Kontos, A. P., R. S. Kotwal, R. J. Elbin, R. H. Lutz, R. D. Forsten, P. J. Benson, and K. M. Guskiewicz, 2013, Residual Effects of Combat-Related Mild Traumatic Brain Injury: Journal of Neurotrauma, v. 30, p. 680-686.

Lai, P. H., J. T. Ho, W. L. Chen, S. S. Hsu, J. S. Wang, H. B. Pan, and C. F. Yang, 2002, Brain abscess and necrotic brain tumor: Discrimination with proton MR spectroscopy and diffusion-weighted imaging: American Journal of Neuroradiology, v. 23, p. 1369-1377.

Lukas, L., A. Devos, J. A. K. Suykens, L. Vanhamme, F. A. Howe, C. Majos, A. Moreno-Torres, M. Van Der Graaf, A. R. Tate, C. Arus, and S. Van Huffel, 2004, Brain tumor classification based on long echo proton MRS signals: Artificial Intelligence in Medicine, v. 31, p. 73-89.

Luts, J., A. Heerschap, J. A. K. Suykens, and S. Van Huffel, 2007, A combined MRI and MRSI based multiclass system for brain tumour recognition using LS-SVMs with class probabilities and feature selection: Artificial Intelligence in Medicine, v. 40, p. 87-102.

Mandal, P. K., 2012, In vivo proton magnetic resonance spectroscopic signal processing for the absolute quantitation of brain metabolites: European Journal of Radiology, v. 81, p. E653-E664.

Marino, S., E. Zei, M. Battaglini, C. Vittori, A. Buscalferri, P. Bramanti, A. Federico, and N. De Stefano, 2007, Acute metabolic brain changes following traumatic brain injury and their relevance to clinical severity and outcome: Journal of Neurology Neurosurgery and Psychiatry, v. 78.

Marmarou, A., J. Lu, I. Butcher, G. S. McHugh, G. D. Murray, E. W. Steyerberg, N. A. Mushkudiani, S. Choi, and A. I. R. Maas, 2007, Prognostic value of the Glasgow coma scale and pupil reactivity in traumatic brain injury assessed pre-hospital and on enrollment: An IMPACT analysis: Journal of Neurotrauma, v. 24, p. 270-280.

Mathews, G. C., and J. S. Diamond, 2003, Neuronal glutamate uptake contributes to GABA synthesis and inhibitory synaptic strength: Journal of Neuroscience, v. 23, p. 2040-2048.

McBride, D. Q., B. L. Miller, D. L. Nikas, S. Buchthal, L. Chang, F. Chiang, and R. A. Booth, 1995, Analysis of brain tumors using H-1 magnetic resonance spectroscopy: Surgical Neurology, v. 44.

Menze, B. H., M. P. Lichy, P. Bachert, B. M. Kelm, H. P. Schlemmer, and F. A. Hamprecht, 2006, Optimal classification of long echo time in vivo magnetic resonance spectra in the detection of recurrent brain tumors: Nmr in Biomedicine, v. 19, p. 599-609.

Miller, B. L., R. A. Moats, T. Shonk, T. Ernst, S. Woolley, and B. D. Ross, 1993, ALZHEIMER-DISEASE - DEPICTION OF INCREASED CEREBRAL MYOINOSITOL WITH PROTON MR SPECTROSCOPY: Radiology, v. 187, p. 433-437.

Mittl, R. L., R. I. Grossman, J. F. Hiehle, R. W. Hurst, D. R. Kauder, T. A. Gennarelli, and G. W. Alburger, 1994, PREVALENCE OF MR EVIDENCE OF DIFFUSE AXONAL INJURY IN PATIENTS WITH MILD HEAD-INJURY AND NORMAL HEAD CT FINDINGS: American Journal of Neuroradiology, v. 15, p. 1583-1589.

Moffett, J. R., B. Ross, P. Arun, C. N. Madhavarao, and A. M. A. Namboodiri, 2007, N-acetylaspartate in the CNS: From neurodiagnostics to neurobiology: Progress in Neurobiology, v. 81.

Morganti-Kossmann, M. C., M. Rancan, V. I. Otto, P. F. Stahel, and T. Kossmann, 2001, Role of cerebral inflammation after traumatic brain injury: A revisited concept: Shock, v. 16.

Mulkern, R. V., J. Q. Meng, K. Oshio, and A. A. Tzika, 1996, Line scan imaging of brain metabolites with CPMG sequences at 1.5 tesla: Jmri-Journal of Magnetic Resonance Imaging, v. 6, p. 399-405.

NessAiver, M., 1996, All You Really Need to KNow About MRI Physics. 1st ed.

Niepel, G., C. R. Tench, P. S. Morgan, N. Evangelou, D. P. Auer, and C. S. Constantinescu, 2006, Deep gray matter and fatigue in MS - A T1 relaxation time study: Journal of Neurology, v. 253, p. 896-902.

Ongur, D., A. P. Prescot, J. E. Jensen, E. D. Rouse, B. M. Cohen, P. F. Renshaw, and D. P. Olson, 2010, T(2) Relaxation Time Abnormalities in Bipolar Disorder and Schizophrenia: Magnetic Resonance in Medicine, v. 63, p. 1-8.

Palmer, A. M., D. W. Marion, M. L. Botscheller, P. E. Swedlow, S. D. Styren, and S. T. Dekosky, 1993, TRAUMATIC BRAIN INJURY-INDUCED

EXCITOTOXICITY ASSESSED IN A CONTROLLED CORTICAL IMPACT MODEL: Journal of Neurochemistry, v. 61.

Park, E., J. D. Bell, and A. J. Baker, 2008, Traumatic brain injury: Can the consequences be stopped?: Canadian Medical Association Journal, v. 178, p. 1163-1170.

Parks, M. H., B. M. Dawant, W. R. Riddle, S. L. Hartmann, M. S. Dietrich, M. K. Nickel, R. R. Price, and P. R. Martin, 2002, Longitudinal brain metabolic characterization of chronic alcoholics with proton magnetic resonance spectroscopy: Alcoholism-Clinical and Experimental Research, v. 26.

Pitt, D., P. Werner, and C. S. Raine, 2000, Glutamate excitotoxicity in a model of multiple sclerosis: Nature Medicine, v. 6, p. 67-70.

Posse, S., C. Decarli, and D. Lebihan, 1994, 3-DIMENSIONAL ECHO-PLANAR MR SPECTROSCOPIC IMAGING AT SHORT ECHO TIMES IN THE HUMAN BRAIN: Radiology, v. 192, p. 733-738.

Posse, S., G. Tedeschi, R. Risinger, R. Ogg, and D. Lebihan, 1995, HIGH-SPEED H-1 SPECTROSCOPIC IMAGING IN HUMAN BRAIN BY ECHO-PLANAR SPATIAL-SPECTRAL ENCODING: Magnetic Resonance in Medicine, v. 33, p. 34-40.

Provencher, S. W., 2001, Automatic quantitation of localized in vivo H-1 spectra with LCModel: Nmr in Biomedicine, v. 14.

Raghupathi, R., D. I. Graham, and T. K. McIntosh, 2000, Apoptosis after traumatic brain injury: Journal of Neurotrauma, v. 17.

Ratiney, H., M. Sdika, Y. Coenradie, S. Cavassila, D. van Ormondt, and D. Graveron-Demilly, 2005, Time-domain semi-parametric estimation based on a metabolite basis set: Nmr in Biomedicine, v. 18, p. 1-13.

Renshaw, P. F., B. Lafer, S. M. Babb, M. Fava, A. L. Stoll, J. D. Christensen, C. M. Moore, D. A. YurgelunTodd, C. M. Bonello, S. S. Pillay, A. J. Rothschild, A. A. Nierenberg, J. F. Rosenbaum, and B. M. Bruce, 1997, Basal ganglia choline levels in depression and response to fluoxetine treatment: An in vivo proton magnetic resonance spectroscopy study: Biological Psychiatry, v. 41.

Ross, B. D., T. Ernst, R. Kreis, L. J. Haseler, S. Bayer, E. Danielsen, S. Bluml, T. Shonk, J. C. Mandigo, W. Caton, C. Clark, S. W. Jensen, N. L. Lehman, E. Arcinue, R. Pudenz, and C. H. Shelden, 1998, H-1 MRS in acute traumatic brain injury: Jmri-Journal of Magnetic Resonance Imaging, v. 8.

Rothman, D. L., N. R. Sibson, F. Hyder, J. Shen, K. L. Behar, and R. G. Shulman, 1999, In vivo nuclear magnetic resonance spectroscopy studies of the relationship between the glutamate-glutamine neurotransmitter cycle and functional neuroenergetics: Philosophical Transactions of the Royal Society B-Biological Sciences, v. 354, p. 1165-1177.

Sabet, A. A., E. Christoforou, B. Zatlin, G. M. Genin, and P. V. Bayly, 2008, Deformation of the human brain induced by mild angular head acceleration: Journal of Biomechanics, v. 41, p. 307-315.

Sakellaris, G., M. Kotsiou, M. Tamiolaki, G. Kalostos, E. Tsapaki, M. Spanaki, M. Spilioti, G. Charissis, and A. Evangeliou, 2006, Prevention of complications related to traumatic brain injury in children and adolescents with creatine administration: An open label randomized pilot study: Journal of Trauma-Injury Infection and Critical Care, v. 61, p. 322-329.

Sanacora, G., G. F. Mason, D. L. Rothman, K. L. Behar, F. Hyder, O. A. C. Petroff, R. M. Berman, D. S. Charney, and J. H. Krystal, 1999, Reduced cortical gamma-aminobutyric acid levels in depressed patients determined by proton magnetic resonance spectroscopy: Archives of General Psychiatry, v. 56, p. 1043-1047.

Sarmento, E., P. Moreira, C. Brito, J. Souza, C. Jevoux, and M. Bigal, 2009, Proton Spectroscopy in Patients With Post-Traumatic Headache Attributed to Mild Head Injury: Headache, v. 49.

Sbarra, A. J., and M. L. Karnovsky, 1959, The biochemical basis of phagocytosis. I. Metabolic changes during the ingestion of particles by polymorphonuclear leukocytes: The Journal of biological chemistry, v. 234.

Schmidt-Rohr, K., J. Clauss, and H. W. Spiess, 1992, CORRELATION OF STRUCTURE, MOBILITY, AND MORPHOLOGICAL INFORMATION IN HETEROGENEOUS POLYMER MATERIALS BY 2-DIMENSIONAL WIDELINE-SEPARATION NMR-SPECTROSCOPY: Macromolecules, v. 25, p. 3273-3277.

Scholkopf, B., K. K. Sung, C. J. C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, 1997, Comparing support vector machines with Gaussian kernels to radial basis function classifiers: Ieee Transactions on Signal Processing, v. 45, p. 2758-2765.

Schuhmann, M. U., D. Stiller, M. Skardelly, J. Bernarding, P. M. Klinge, A. Samii, M. Samii, and T. Brinker, 2003, Metabolic changes in the vicinity of brain contusions: a proton magnetic resonance spectroscopy and histology study: Journal of Neurotrauma, v. 20.

Seilhean, D., C. Duyckaerts, and J. J. Hauw, 1995, FATAL FAMILIAL INSOMNIA AND PRION DISEASES: Revue Neurologique, v. 151, p. 225-230.

Shimon, H., Y. Sobolev, M. Davidson, V. Haroutunian, R. H. Belmaker, and G. Agam, 1998, Inositol levels are decreased in postmortem brain of schizophrenic patients: Biological Psychiatry, v. 44, p. 428-432.

Signoretti, S., A. Marmarou, G. A. Aygok, P. P. Fatouros, G. Portella, and R. M. Bullock, 2008, Assessment of mitochondrial impairment in traumatic brain injury using high-resolution proton magnetic resonance spectroscopy: Journal of Neurosurgery, v. 108.

Signoretti, S., A. Marmarou, B. Tavazzi, G. Lazzarino, A. Beaumont, and R. Vagnozzi, 2001, N-acetylaspartate reduction as a measure of injury severity and mitochondrial dysfunction following diffuse traumatic brain injury: Journal of Neurotrauma, v. 18.

Simmons, A., P. S. Tofts, G. J. Barker, and S. R. Arridge, 1994, SOURCES OF INTENSITY NONUNIFORMITY IN SPIN-ECHO IMAGES AT 1.5-T: Magnetic Resonance in Medicine, v. 32, p. 121-128.

Skinner, M. G., S. H. Kolind, and A. L. MacKay, 2007, The effect of varying echo spacing within a multiecho acquisition: better characterization of long T-2 components: Magnetic Resonance Imaging, v. 25, p. 840-847.

Sotak, C. H., and D. M. Freeman, 1988, A METHOD FOR VOLUME-LOCALIZED LACTATE EDITING USING ZERO-QUANTUM COHERENCE CREATED IN A STIMULATED-ECHO PULSE SEQUENCE: Journal of Magnetic Resonance, v. 77, p. 382-388.

Spelman, J. F., S. C. Hunt, K. H. Seal, and A. L. Burgo-Black, 2012, Post Deployment Care for Returning Combat Veterans: Journal of General Internal Medicine, v. 27, p. 1200-1209.

Squarcina, L., A. Bertoldo, T. E. Ham, R. Heckemann, and D. J. Sharp, 2012, A robust method for investigating thalamic white matter tracts after traumatic brain injury: Neuroimage, v. 63, p. 779-788.

Stefan, D., F. Di Cesare, A. Andrasescu, E. Popa, A. Lazariev, E. Vescovo, O. Strbak, S. Williams, Z. Starcuk, M. Cabanas, D. van Ormondt, and D. Graveron-Demilly, 2009, Quantitation of magnetic resonance spectroscopy signals: the jMRUI software package: Measurement Science & Technology, v. 20.

Storey, J. D., 2002, A direct approach to false discovery rates: Journal of the Royal Statistical Society Series B-Statistical Methodology, v. 64, p. 479-498.

Strich, S. J., 1961, SHEARING OF NERVE FIBRES AS A CAUSE OF BRAIN DAMAGE DUE TO HEAD INJURY - A PATHOLOGICAL STUDY OF 20 CASES: Lancet, v. 2, p. 443-&.

Sullivan, P. G., J. D. Geiger, M. P. Mattson, and S. W. Scheff, 2000, Dietary supplement creatine protects against traumatic brain injury: Annals of Neurology, v. 48, p. 723-729.

Sutton, L. N., Z. Wang, A. C. Duhaime, D. Costarino, R. Sauter, and R. Zimmerman, 1995, TISSUE LACTATE IN PEDIATRIC HEAD TRAUMA - A CLINICAL-STUDY USING H-1-NMR SPECTROSCOPY: Pediatric Neurosurgery, v. 22, p. 81-87.

Tagliaferri, F., C. Compagnone, M. Korsic, F. Servadei, and J. Kraus, 2006, A systematic review of brain injury epidemiology in Europe: Acta Neurochirurgica, v. 148, p. 255-268.

Tartaglia, M. C., S. Narayanan, N. De Stefano, R. Arnaoutelis, S. B. Antel, S. J. Francis, A. C. Santos, Y. Lapierre, and D. L. Arnold, 2002, Choline is increased in pre-lesional normal appearing white matter in multiple sclerosis: Journal of Neurology, v. 249, p. 1382-1390.

Tate, A. R., J. Underwood, D. M. Acosta, M. Julia-Sape, C. Majos, A. Moreno-Torres, F. A. Howe, M. van der Graaf, V. Lefournier, M. M. Murphy, A. Loosemore, C. Ladroue, P. Wesseling, J. L. Bosson, M. E. Cabanas, A. W. Simonetti, W. Gajewicz, J. Calvar, A. Capdevila, P. R. Wilkins, B. A. Bell, C. Remy, A. Heerschap, D. Watson, J. R. Griffiths, and C. Arus, 2006, Development of a decision support system for diagnosis and grading of brain tumours using in vivo magnetic resonance single voxel spectra: Nmr in Biomedicine, v. 19, p. 411-434.

Thurman, D. J., C. Alverson, K. A. Dunn, J. Guerrero, and J. E. Sniezek, 1999, Traumatic brain injury in the United States: A public health perspective: Journal of Head Trauma Rehabilitation, v. 14, p. 602-615.

Toblin, R. L., L. A. Riviere, J. L. Thomas, A. B. Adler, B. C. Kok, and C. W. Hoge, 2012, Grief and physical health outcomes in US soldiers returning from combat: Journal of Affective Disorders, v. 136.

Tollard, E., D. Galanaud, V. Perlbarg, P. Sanchez-Pena, Y. Le Fur, L. Abdennour, P. Cozzone, S. Lehericy, J. Chiras, and L. Puybasset, 2009, Experience of diffusion tensor imaging and H-1 spectroscopy for outcome prediction in severe traumatic brain injury: Preliminary results: Critical Care Medicine, v. 37, p. 1448-1455.

Traber, F., W. Block, R. Lamerichs, E. Keller, and H. H. Schild, 1997, Reduction of measurement time by H-1-MR turbo spectroscopic imaging of the brain: Rofo-Fortschritte Auf Dem Gebiet Der Rontgenstrahlen Und Der Bildgebenden Verfahren, v. 166, p. 221-229.

Tracey, I., and B. A. Navia, 1996, Brain choline-containing compounds are elevated in HIV-positive patients before the onset of AIDS dementia complex: A proton magnetic resonance spectroscopic study (vol 46, pg 783, 1996): Neurology, v. 46, p. 1787-1787.

Vagnozzi, R., A. Marmarou, B. Tavazzi, S. Signoretti, D. Di Pierro, F. Del Bolgia, A. M. Amorini, G. Fazzina, S. Sherkat, and G. Lazzarino, 1999, Changes of cerebral energy metabolism and lipid peroxidation in rats leading to mitochondrial dysfunction after diffuse brain injury: Journal of Neurotrauma, v. 16.

Vagnozzi, R., S. Signoretti, L. Cristofori, F. Alessandrini, R. Floris, E. Isgro, A. Ria, S. Marziale, G. Zoccatelli, B. Tavazzi, F. Del Bolgia, R. Sorge, S. P. Broglio, T. K. McIntosh, and G. Lazzarino, 2010, Assessment of metabolic brain damage and recovery following mild traumatic brain injury: a multicentre, proton magnetic resonance spectroscopic study in concussed patients: Brain, v. 133.

Vagnozzi, R., S. Signoretti, B. Tavazzi, R. Floris, A. Ludovici, S. Marziali, G. Tarascio, A. M. Amorini, V. Di Pietro, R. Delfini, and G. Lazzarino, 2008, Temporal window of metabolic brain vulnerability to concussion: A pilot (1)H magnetic resonance spectroscopic study in concussed athletes - Part III: Neurosurgery, v. 62.

Van Horn, W. D., A. J. Beel, C. Kang, and C. R. Sanders, 2010, The impact of window functions on NMR-based paramagnetic relaxation enhancement measurements in membrane proteins: Biochimica Et Biophysica Acta-Biomembranes, v. 1798, p. 140-149.

Vanhamme, L., T. Sundin, P. Van Hecke, S. Van Huffel, and R. Pintelon, 2000, Frequency-selective quantification of biomedical magnetic resonance spectroscopy data: Journal of Magnetic Resonance, v. 143, p. 1-16.

Verweij, B. H., J. P. Muizelaar, F. C. Vinas, P. L. Peterson, Y. Xiong, and C. P. Lee, 2000, Impaired cerebral mitochondrial function after traumatic brain injury in humans: Journal of Neurosurgery, v. 93.

Vicente, J., E. Fuster-Garcia, S. Tortajada, J. M. Garcia-Gomez, N. Davies, K. Natarajan, M. Wilson, R. G. Grundy, P. Wesseling, D. Monleon, B. Celda, M. Robles, and A. C. Peet, 2013, Accurate classification of childhood brain tumours by in vivo(1)H MRS - A multi-centre study: European journal of cancer (Oxford, England : 1990), v. 49, p. 658-67.

Wang, Z. Y., 1996, Improvement of measurement precision in absorption spectra by apodization: Magnetic Resonance in Medicine, v. 35, p. 917-920.

Warden, D. L., J. Bleiberg, K. L. Cameron, J. Ecklund, J. Walter, M. B. Sparling, D. Reeves, K. Y. Reynolds, and R. Arciero, 2001, Persistent prolongation of simple reaction time in sports concussion: Neurology, v. 57.

Wilk, J. E., R. K. Herrell, G. H. Wynn, L. A. Riviere, and C. W. Hoge, 2012, Mild Traumatic Brain Injury (Concussion), Posttraumatic Stress Disorder, and Depression in US Soldiers Involved in Combat Deployments: Association With Postdeployment Symptoms: Psychosomatic Medicine, v. 74.

Wittsack, H. J., H. Kugel, B. Roth, and W. Heindel, 1996, Quantitative measurement with localized H-1 MR spectroscopy in children with Canavan's disease: Jmri-Journal of Magnetic Resonance Imaging, v. 6, p. 889-893.

Wu, A., Z. Ying, and F. Gomez-Pinilla, 2013, Exercise facilitates the action of dietary DHA on functional recovery after brain trauma: Neuroscience, v. 248, p. 655-63.

Xiong, Y., Q. Gu, P. L. Peterson, J. P. Muizelaar, and C. P. Lee, 1997, Mitochondrial dysfunction and calcium perturbation induced by traumatic brain injury: Journal of Neurotrauma, v. 14.

Yeo, R. A., C. Gasparovic, F. Merideth, D. Ruhl, D. Doezema, and A. R. Mayer, 2011, A Longitudinal Proton Magnetic Resonance Spectroscopy Study of Mild Traumatic Brain Injury: Journal of Neurotrauma, v. 28.

Yeo, R. A., J. P. Phillips, R. E. Jung, A. J. Brown, R. C. Campbell, and W. M. Brooks, 2006, Magnetic resonance spectroscopy detects brain injury and predicts cognitive functioning in children with brain injuries: Journal of Neurotrauma, v. 23.

Yi, J. H., and A. S. Hazell, 2006, Excitotoxic mechanisms and the role of astrocytic glutamate transporters in traumatic brain injury: Neurochemistry International, v. 48, p. 394-403.

Zeisel, S. H., and J. K. Blusztajn, 1994, CHOLINE AND HUMAN-NUTRITION: Annual Review of Nutrition, v. 14, p. 269-296.

Zhang, Y., S. Marenco, and J. Shen, 2007, Correction of frequency and phase variations induced by eddy currents in localized spectroscopy with multiple echo times: Magnetic Resonance in Medicine, v. 58, p. 174-178.

Zweig, M. H., and G. Campbell, 1993, RECEIVER OPERATING CHARACTERISTIC (ROC) PLOTS - A FUNDAMENTAL EVALUATION TOOL IN CLINICAL MEDICINE (VOL 39, PG 561, 1993): Clinical Chemistry, v. 39, p. 1589-1589.