Two-Dimensional Spectral Estimation Techniques with Applications to Magnetic Resonance Spectroscopy

by

Frederick J. Frigo, B.S., M.S.

A Dissertation submitted to the Faculty of the Graduate School, Marquette University, in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Milwaukee, Wisconsin
April 14, 2004

This is to certify that we have examined this copy of the dissertation by

Frederick J. Frigo, B.S., M.S.

and have found that it is complete and satisfactory in all respects.

The dissertation has been approved by:

James A. Heinen

Dr. James A. Heinen
Dissertation Director, Department of Electrical and Computer Engineering

Mathew Bernst

Dr. Mathew A. Bernstein
Committee Member, The Mayo Clinic

Ronald H. Brown

Dr. Ronald H. Brown
Committee Member, Department of Electrical and Computer Engineering

Bruce Collick

Dr. Bruce D. Collick
Committee Member, General Electric Healthcare

James E. Richie

Dr. James E. Richie
Committee Member, Department of Electrical and Computer Engineering

Approved on

April 14, 2004

(date)

Abstract

Single-voxel proton magnetic resonance spectroscopy (MRS) is typically used in a clinical setting to quantify metabolites in the human brain. By convention, an MRS absorption spectrum is created by Fourier transformation of phase-corrected raw data acquired during an MRS experiment. An MRS absorption spectrum shows the relative concentrations of certain key metabolites, including N-Acetyl-aspartate (NAA), choline, creatine and others. Certain nonparametric techniques may also be used for MRS analysis. 2D Capon and 2D amplitude and phase estimation (APES) are two relatively new nonparametric methods that can be used effectively to estimate both frequency and damping characteristics of each metabolite. In this dissertation we introduce the weighted 2D Capon, weighted 2D APES, and combined weighted 2D APES / 2D Capon methods. Under certain conditions these methods may provide improved estimation properties and/or reduced computation time, as compared to conventional 2D methods.

Many clinicians routinely use multiple receive coils for magnetic resonance imaging (MRI) studies of the human brain. In conjunction with these exams, it is often desired to perform proton MRS experiments to quantify metabolites from a region of interest. An MRS absorption spectrum can be generated for each coil element; however, interpreting the results from each channel is a tedious process. Combining MRS absorption spectra obtained from an experiment in which multiple receive coils are used would greatly simplify clinical diagnosis. In this dissertation we introduce two methods

for 2D spectral estimation in the case of multi-channel data. To date, no such methods have appeared in the literature. These new methods employ weighted signal averaging and weighted spectrum averaging and use any of the 2D techniques described above. We also introduce a method to optimally estimate the relative channel gains from observed data.

The new techniques developed in this dissertation are evaluated and compared to conventional 2D spectral estimation based on extensive computer simulations written in MATLAB. They are also applied to phantom and *in vivo* MRS data.

Acknowledgements

I would like to express my sincere gratitude to my research director, Dr. James A. Heinen. Like so many other Marquette University engineering students, I consider myself to be very fortunate to have taken classes in Linear Systems Analysis, Digital Signal Processing, Digital Filter Design, and Adaptive Filtering from Dr. Heinen. His contributions to this dissertation have been immense, and his scholarly dedication and patience have been exemplary. Without his mentoring, and involvement, this research would not have been possible. Both of us have learned a great deal about magnetic resonance spectroscopy through this research, and have found the application of signal processing strategies to improving the clinical usefulness of magnetic resonance spectroscopy to be quite fascinating.

I would also like to acknowledge the other members of my committee: Dr. Matthew A. Bernstein of the Mayo Clinic, Dr. Ronald H. Brown of Marquette University, Dr. Bruce D. Collick of General Electric Healthcare, and Dr. James E. Richie of Marquette University. My committee provided a great deal of encouragement and focus for the direction of this work.

I am also grateful to Marquette University, the Society of Jesus, and to the many outstanding professors I have learned from: Dr. Russell J. Niederjohn, Dr. Arthur C. Moeller, Dr. Douglas J. Harris, Dr. Roger H. Johnson, and many others. In addition, I would like to acknowledge the contributions of many individuals (from General Electric

Healthcare unless noted otherwise); Dr. Bryan J. Mock, Dr. Jason A. Polzin, Dr. Thomas E. Raidy of Duke University, Dr. Thoralf Niendorf, Dr. Jeffrey A. Hopkins, Dr. Kevin F. King, Dr. R. Scott Hinks, Dr. Bipin Salunkhe, Dr. Jean H. Brittain, Dr. Steve G. Tan, Dr. Steve S. Chen, Dr. David C. Zhu of the University of Chicago, Dr. Michael W. Bourne of the University Hospital of Wales, Dr. Shawn F. Halpin of the University Hospital of Wales, Dr. Sarah K. Patch, Dr. Elisabeth C. Angelos, Dr. Dale R. Thayer, Dr. Manoj Saranathan, Dr. Sandhya Parameswaran, Louis M. Frigo, David H. Gurr, Charles R. Giordano, Bo J. Pettersson, Daniel J. Zimmerman, and Mike R. Hartley.

Finally, I would like to express my deepest appreciation and gratitude to my wife, Jill, and my children, Katherine, William, Joseph and Elizabeth, for their love and support. Completing this dissertation was a significant challenge, and they provided the motivation to finish. I would also like to thank my parents, Joseph and Irene Frigo, my brother, Louis, and my sisters, Julie and Janette, for their prayers, and helpful advice.

My sister, Julie, fought against the ravages of juvenile diabetes, yet had the courage and tenacity to embark on her own scholarly journey. After graduating from the College of Pharmacy at the University of Wisconsin at Madison, she enrolled in the Ph.D. program at the University of Michigan at Ann Arbor to focus on diabetes research. Her journey ended far too soon, but serves as a source of great inspiration in my journey, and I dedicate this dissertation in her memory.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Single-voxel proton MRS is typically used in a clinical setting to quantify metabolites in the human brain. MRS studies are vitally important for diagnosis, monitoring therapy, and early detection of a number of diseases including neurodegenerative disease associated with normal aging, Alzheimer's disease, non-Alzheimer's dementia, Parkinson's disease, Huntington's disease, Amyotrophic Lateral Sclerosis (ALS), alcohol abuse, epilepsy, stroke, Creutzfeldt-Jakob (mad cow) disease, cerebral-vascular disease and cancer[1].

In a clinical setting, an MRS absorption spectrum is created that shows the relative concentration of certain key metabolites, including NAA, choline, creatine and others. Certain nonparametric techniques may also be used for MRS analysis. 2D Capon and 2D APES are two relatively new nonparametric methods that can be used effectively to estimate both frequency and damping characteristics of each metabolite[2]. One benefit of 2D Capon and 2D APES over conventional techniques for MRS data analysis is improved accuracy for peak detection, especially for peaks that are close to each other. Another benefit of 2D Capon and 2D APES is that estimates of damping are provided in addition to frequency estimates, providing the clinician with an additional parameter to use in the diagnosis and evaluation of results from an MRS study.

Several new techniques for 2D spectral estimation related to 2D Capon and 2D APES will be introduced in this dissertation. Under certain conditions, these techniques show improvement over standard 2D Capon and 2D APES in terms of accuracy in peak detection and/or computational efficiency. Although the new 2D spectral estimation techniques are general purpose in nature, they have been developed as a result of their utility toward the problem of MRS signal processing.

Multiple-channel receive coils are regularly used in a clinical setting for magnetic resonance imaging (MRI) studies since images created from multiple-channel receive coils can offer improved image quality. In conjunction with imaging studies that use multiple-channel receive coils, it is often desired to perform MRS studies during the same examination. An effective technique for creating a combined MRS absorption spectrum from multiple-channel receive coils has been recently introduced by Frigo, Heinen, et al. [3],[4], and has gained clinical acceptance.

Combining 2D spectral estimates obtained from an experiment in which multiple-channel receive coils are used would greatly simplify clinical diagnosis. Two new techniques for multiple-channel 2D spectral estimation will be introduced in this dissertation. The new techniques apply to standard 2D Capon and 2D APES, and to the new 2D spectral estimation methods that have been proposed in this dissertation. The primary motivation to introduce the new multiple-channel 2D spectral estimation techniques was for multiple-channel MRS studies; however, these techniques may be extended to other signal processing applications as well.

We begin our discussion with a brief review of the principles of nuclear magnetic resonance in Chapter 2. The underlying physics and science behind this technology has provided a foundation upon which magnetic resonance (MR) scanners have been designed and built, and the impact of these medical imaging machines to the practice of modern medicine has been extraordinary.

In Chapter 3 we extend our discussion to single-voxel quantitation (SVQ) proton MRS which is typically used in a clinical setting to quantify metabolites in the human brain. Our focus will be to break down and analyze the signal processing steps that are typical for MRS SVQ scans. We have carefully documented each processing step, and have implemented efficient algorithms in MATLAB to create a conventional MRS absorption spectrum from data acquired during an SVQ study. The signal processing steps applied to MRS data are vitally important to provide a framework for the new techniques that will be introduced in later chapters.

Motivated primarily by finding improved techniques for analyzing MRS data, we introduce in Chapter 4 several new nonparametric two-dimensional spectral estimation techniques: the weighted 2D Capon method, the weighted 2D APES method, and the combined weighted 2D APES/Capon method. Each of these techniques provides spectral estimates of damping as well as frequency, making them useful for MRS data analysis.

Multiple-channel receive coils have become increasing popular for magnetic resonance imaging (MRI) since they often provide improved image quality over single channel coils. In a clinical setting, it is common to perform MRS studies along with MRI studies, especially for neurological exams. In Chapter 5 we introduce two new algorithms for multiple-channel 2D spectral estimation that can be used effectively for MRS studies. We also introduce a method to optimally estimate the relative channel gains from observed data.

In Chapter 6, we evaluate the proposed new algorithms of Chapter 4 and Chapter 5 through extensive simulations. The motivation for doing simulations is that the signal processing algorithms can be subjected to controlled conditions, unlike in a standard MRS experiment, where unknown variations can occur from scan to scan.

In Chapter 7 we apply the algorithms developed in Chapters 4 and 5, and simulated in Chapter 6, to data collected during MRS experiments. Examples are provided demonstrating how these algorithms work in MRS studies involving phantoms with known concentrations of metabolites, as well as in a limited number of *in vivo* MRS studies involving human volunteers.

We then reflect on our conclusions in Chapter 8, enlightened by the discussion and examination of the MRS signal processing details that are fundamental to its success in a clinical scenario, but mindful of the potential improvements that can be achieved through new techniques. Based on the results of extensive simulations, we are confident

that these new techniques have the potential to add clinical value and improve the overall quality of spectral analysis for MRS studies.

Chapter 2

Magnetic Resonance Spectroscopy

MRS is used to find the relative spectral amplitudes resulting from frequency components of different molecules. MRS can be used for the quantification of a number of metabolites *in vivo*. Since the hydrogen atom is abundant *in vivo*, many clinical applications for proton MRS have been developed.

MRS is used for diagnosis, surgical planning, monitoring therapy and early detection of the following diseases: neurodegenerative disease associated with normal aging, Alzheimer's disease, non-Alzheimer's dementia, Parkinson's disease, Huntington's disease, amyotrophic lateral sclerosis (ALS), alcohol abuse, epilepsy, stroke, Creutzfeldt-Jakob (mad cow) disease, cerebral-vascular disease, and cancer [1].

Improved data acquisition capabilities, in particular, increased sampling resolution and multiple-channel receive coils, have created opportunities for implementing new signal processing algorithms to improve the sensitivity and accuracy of MRS experiments. The motivation for introducing these new algorithms is to reduce scan time and improve clinical diagnostic capabilities.

2.1 Magnetic Resonance Principles

All nuclei have charge and mass. Some nuclei also possess spin or angular momentum. This spinning charge generates a magnetic field and an associated angular

momentum as first described by Wolfgang Pauli [5]. Inspired by the early work of Pauli, and Rabi[6],[7] two groups of researchers independently discovered the property of nuclear magnetic resonance: Felix Bloch of Stanford University [8] led one group and Edward Purcell of Harvard University [9] led the other group. Bloch and Purcell were jointly awarded the Nobel Prize for physics in 1952 as a result of their discovery.

At first, nuclear magnetic resonance was used primarily in the laboratory by chemists. Later, nuclear magnetic resonance was shown to be useful for medical purposes. Raymond Damadian demonstrated in 1971 that nuclear magnetic resonance spectroscopy could be used *in vivo* to detect cancerous tumors in mice [10]. Paul Lauterbur created the first two-dimensional nuclear magnetic resonance image of human anatomy [11]. For this discovery Lauterbur earned the Nobel Prize in physiology or medicine in 2003 with Sir Peter Mansfield who was also responsible for many innovations in the area of nuclear magnetic resonance [12].

The following is a summary of MR principles adapted from material found in a number of references [13] - [20]. Magnetic resonance originates from the interaction between an atom and an external magnetic field. Atoms with an odd number of either neutrons or protons possess a net nuclear spin or intrinsic nuclear spin angular momentum. Nuclear spin has an associated magnetic field. Nuclear magnetic resonance is based on the interaction of these spins with three types of magnetic fields: the main field, B_0; the radio-frequency field, B_1; and linear gradient fields, G.

Placed in the main static magnetic field, B_0, the net nuclear angular momentum of hydrogen atoms with a single proton will begin to rotate, or precess, about the magnetic field as shown in Fig. 1.

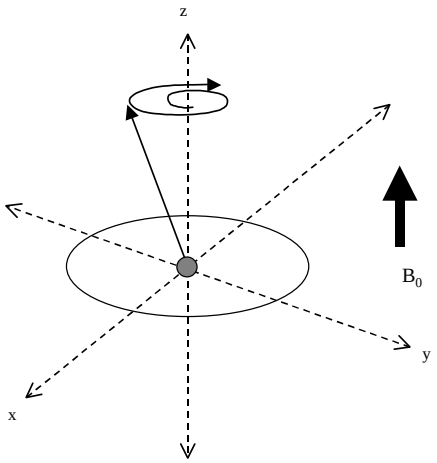

Fig. 1. Net angular momentum of hydrogen.

The frequency of this precession is proportional to the strength of the magnetic field and is given by the Larmor equation:

$$f_0 = \gamma B_0 \qquad (2.1)$$

where f_0 is the Larmor frequency in megahertz (MHz), B_0 is the magnetic field in Tesla (T) and γ is a constant known as the gyromagnetic ratio. For hydrogen with a single proton the value of γ is 42.5774 MHz/T.

To obtain a nuclear magnetic resonance signal, an electromagnetic pulse, B_1, tuned to the resonant frequency of the spins, as described in Eqn. (2.1), is applied in the transverse (*xy*) plane to excite these spins out of equilibrium. After this pulse is turned off, relaxation back to equilibrium occurs and the rotating magnetization vectors induce an electromotive force (EMF) in a receiver coil oriented in such a way as to detect changes in the transverse plane. The generated time signal is known as a free induction decay (FID).

It is possible to perform spatial localization in an MR experiment by applying linear gradient magnetic fields in addition to B_0. Gradient fields in the *x*, *y*, and *z* direction can be combined to provide the spatial localization desired. If a gradient field, G_x, is applied in the *x* direction, then the resonant frequency, $f(x)$, will vary with the *x*-location:

$$f(x) = \gamma(B_0 + G_x x) = f_0 + \gamma G_x x \qquad (2.2)$$

The behavior of the magnetization vector $\mathbf{M}$ is described by the Bloch equation:

$$\frac{d\mathbf{M}}{dt} = \mathbf{M} \times \gamma \mathbf{B} - \frac{\mathrm{M}_x \mathbf{i} + \mathrm{M}_y \mathbf{j}}{T_2} - \frac{(\mathrm{M}_z - \mathrm{M}_0)\mathbf{k}}{T_1} \qquad (2.3)$$

where $\mathbf{i}, \mathbf{j},$ and $\mathbf{k}$ are unit vectors in the *x*, *y*, and *z* directions; M_0 is the equilibrium magnetization arising from the main field B_0; and $\mathbf{B}$ includes the various magnetic fields applied. T_1 is the longitudinal relaxation time constant which characterizes the process of spins returning to equilibrium and T_2 is the transverse relaxation time constant which characterizes the process of decaying transverse magnetization.

Electrons of a particular atom in a molecule may be shared with nearby atoms forming a "cloud" of electrons. The distribution of these electrons is described by quantum mechanics and varies depending on the type of chemical bond. Oxygen-hydrogen bonds have a greater electron density associated with the oxygen atom, whereas carbon-hydrogen bonds are more equally distributed between both atoms. The variation in electron distribution affects the magnetic resonance of a particular atom. The Larmor Eqn. (2.1) can be rewritten with a chemical shielding term, σ_i, which takes into account the variations in different chemical bonds affecting the magnetic resonance frequency:

$$f_i = \gamma \mathrm{B}_0 (1 - \sigma_i) \qquad (2.4)$$

Nuclear magnetic resonance spectroscopy takes advantage of variations in magnetic resonance frequencies due to chemical bonds to determine molecular structure.

The convention is to measure chemical shifts in a manner that is independent of B_0 and can be used for all field strengths:

$$\delta_i = (f_i - f_{ref}) \times 10^6 / f_{ref} \qquad (2.5)$$

where δ_i is the chemical shift measured in parts per million (ppm).

The reference frequency, f_{ref}, defined to have a shift of 0.0 ppm is obtained from the methyl (C-H) groups of tetramethylsilane $(CH_3)_4Si$. Since silicon (Si) has low electronegativity, the shielding of protons is greater than in most other organic molecules, and the chemical shifts thus appear downward in the same direction [21]. As an example, the hydrogen found in water molecules has a shift measured at 4.7 ppm while the hydrogen found in the lipids that are a major constituent of body fat has a shift of 1.2 ppm. This represents a difference of 3.5 ppm, or at 1.5T a difference of approximately 225 Hz.

2.2 Magnetic Resonance Spectroscopy Experiment

Proton MRS is used *in vivo* to measure the concentration of a number of metabolites. MRS is often used for clinical studies of the human brain. High-field magnetic resonance scanners (1.5T or greater) are used to perform most of these studies. Two common types of MRS pulse sequences used in conjunction with clinical studies are point resolved spectroscopy (PRESS) and stimulated echo mode (STEAM) [22].

Both PRESS and STEAM rely on applied gradient fields to isolate a three-dimensional region of interest from which the signal will be measured. PRESS and STEAM differ in how the echoes that give rise to the signal of interest are obtained. PRESS has a higher signal-to-noise ratio than STEAM by a factor of two since the echo in PRESS is created by refocusing the entire net magnetization of the region of interest. STEAM has several advantages over PRESS since it can support a shorter echo time (TE) and has reduced effects from J coupling [23].

Single-voxel quantification (SVQ) and chemical shift imaging (CSI) are two types of spectroscopy applications often used in a clinical setting. SVQ quantifies metabolites in a single volume of interest as shown in Fig. 2. CSI attempts to quantify metabolites in multiple volumes of interest, often overlaying color maps on top of anatomical images to create localized maps showing relative concentrations of metabolites (such as NAA) as shown in Fig. 3.

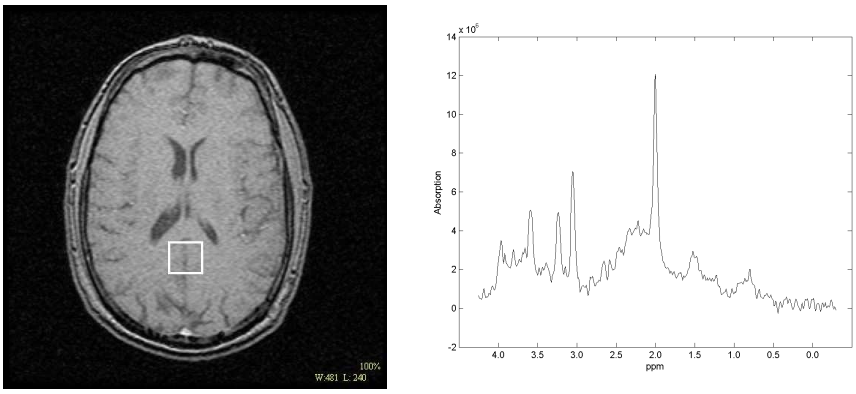

Fig. 2. Single voxel quantification example.

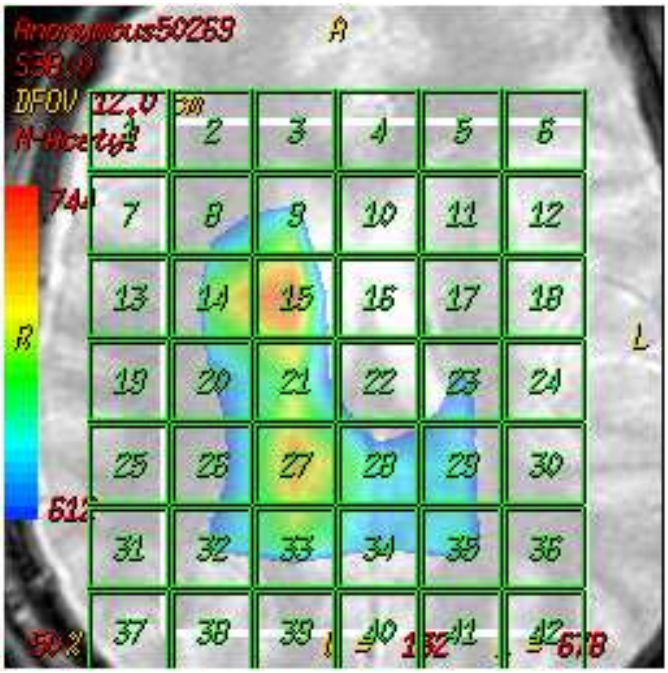

Fig. 3. Chemical shift imaging example.

Our discussion and exploration of proton MRS will focus primarily on SVQ. A typical SVQ MRS scan involves a data acquisition scheme in which a set of non-water-suppressed reference data, along with a set of water-suppressed data is collected. Water-suppression can be achieved *in vivo* using a chemical selective saturation (CHESS) pulse-sequence to reduce the dominant water signal[22],[24]. A typical frame of data obtained from a FID during the non-water-suppressed reference acquisition is shown in Fig. 4, and a typical frame of data obtained during a water-suppressed acquisition is shown in Fig. 5.

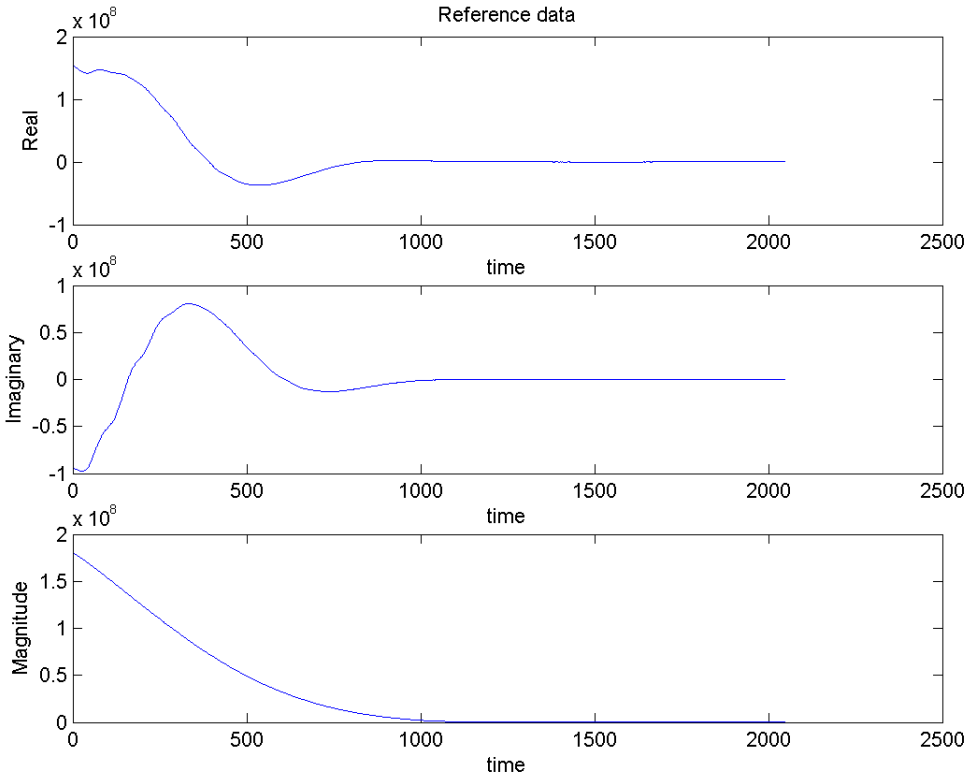

Fig. 4. Raw data frame obtained during MRS reference acquisition.

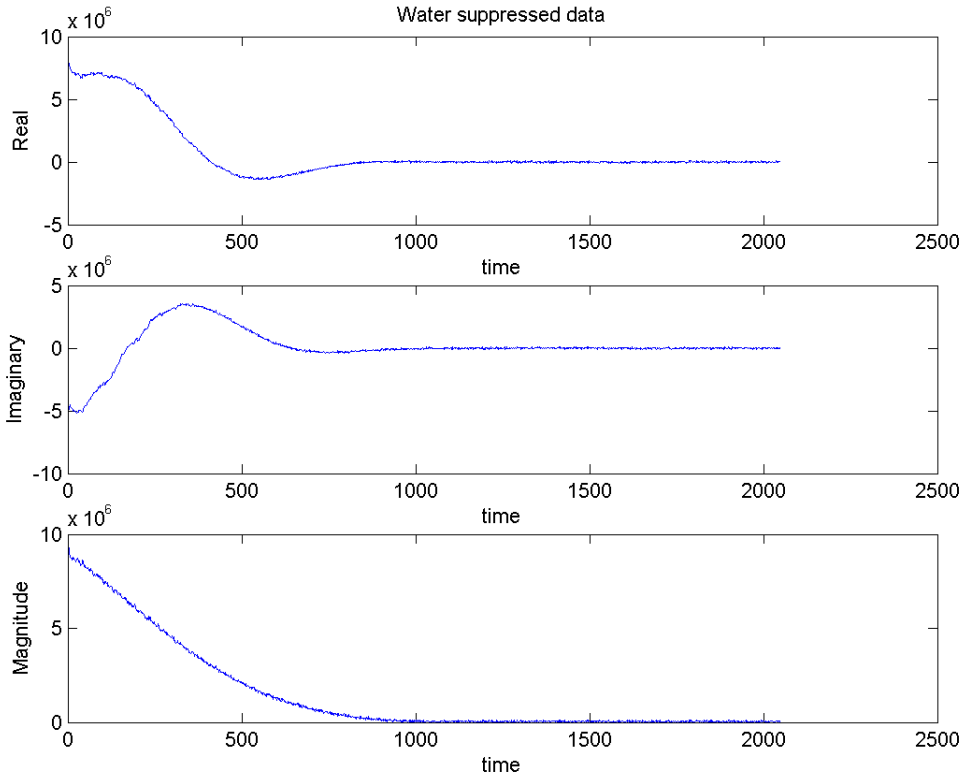

Fig. 5. Raw data frame obtained during water-suppressed MRS acquisition.

An example of the magnitude of the raw data collected during an MRS scan is shown in Fig. 6 with the horizontal axis representing time, the vertical axis representing frame number and color indicating signal amplitude. This scan was performed using a 1.5T General Electric® (GE) MR scanner (Milwaukee, WI, USA) on a 8cc volume contained within the GE MRS phantom, with a TE of 35 msec, a repetition time (TR) of 1500 msec, an acquisition size of 2048 complex data pairs for each frame, a sampling

rate of 16 kHz, and 2 excitations per frame (NEX). The total scan time was 1 minute and 18 seconds to acquire 8 reference frames and 16 water-suppressed frames.

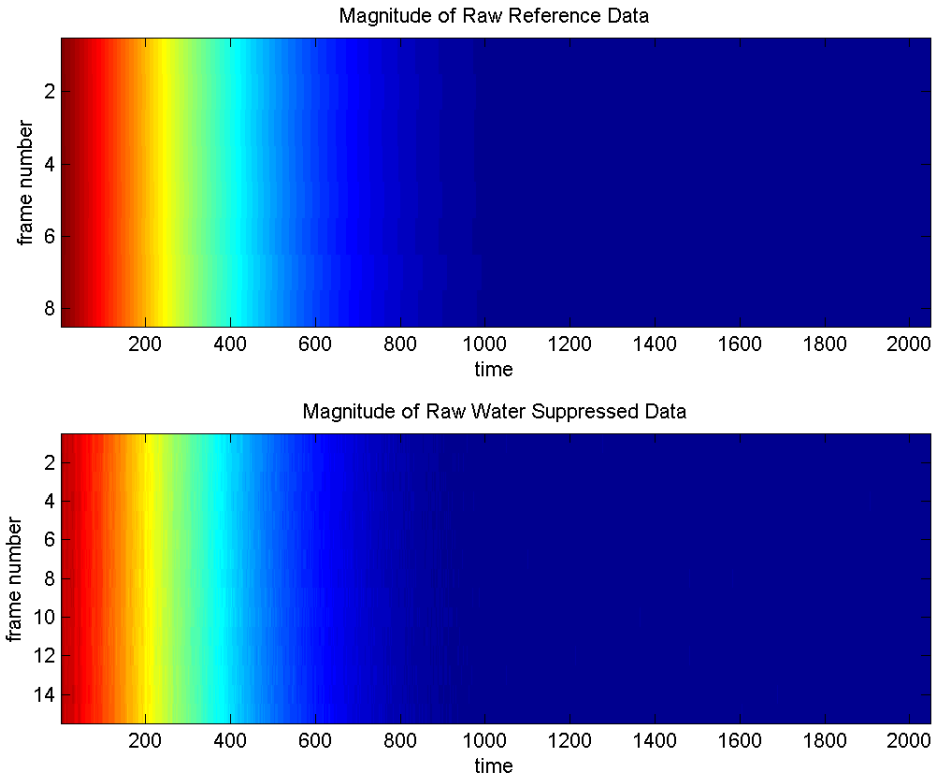

Fig. 6. Magnitude of raw data for an MRS scan.

MRS data acquisition requires that a number of scan parameters be determined for each region of interest. The center frequency, flip angle, transmit gain, receive gain, and shimming parameters are determined for each MRS experiment during prescan. These

parameters are optimized for the particular TE and TR selected for the experiment. Shimming is also extremely important in the region of interest to assure that a homogeneous magnetic field is present since the patient distorts the magnetic field and areas within the patient have different magnetic susceptibility[25]. For MRS experiments conducted *in vivo*, water suppression is an important consideration due to the abundance of water *in vivo*[22].

Many different types of MRS scans can be taken in a clinical setting[26]-[28]. The parameters of these scans are selected to provide the desired clinical information. For example, depending on the metabolite of interest, different TEs may be selected. MRS scans usually have scan TRs greater than one second. The smallest TEs supported are approximately 30 msec. The signal-to-noise ratio (SNR) of the data acquired during an MRS scan can be improved by increasing the number of data frames acquired for a given scan. This increases the scan time however. Another way to increase SNR is to increase the size of the volume under study.

In an MRS experiment it is possible to detect the magnetization that is transverse to the main longitudinal field, B_0, with more than one receive coil simultaneously. Multiple-channel receive coils provide greater sensitivity and improved SNRs for MRI applications [15],[29]-[38]. Multiple-receive coils can also be used effectively for MRS studies [3],[39]-[42]. In Chapter 5 we will extend these techniques to 2D spectral estimation of MRS data from multiple receive coils.

2.3 Summary

In this chapter, we have reviewed the fundamental theory behind magnetic resonance spectroscopy and have introduced how it is used in a clinical setting for *in vivo* studies. We have introduced two conventional pulse-sequences, PRESS and STEAM, that can be used for MRS experiments, and also have described briefly how metabolites can be quantified using SVQ.

Chapter 3

Magnetic Resonance Spectroscopy Processing Techniques

This chapter describes conventional MRS data processing, and in particular, the processing associated with SVQ [22],[43]-[45]. Conventional single-voxel MRS involves collecting a set of non-water suppressed reference data along with a set of water-suppressed data. The processing involves phase correction, removal of residual water present in the water-suppressed data, and computation of an MRS absorption spectrum from which information about metabolite concentration can be determined.

The GE MRS phantom provides an effective model of the human brain. It contains known solutions of metabolites whose concentrations are approximately the same as what might be found in a human brain. The GE MRS phantom contains a solution of 12.5 mM NAA, 10.0 mM creatine (Cr), 3.0 mM choline (Ch), 7.5 mM myoinositol (mI), 12.5 mM L-Glutamic acid (Glu), 5.0 mM lactate, and 0.5 mM γ-aminobutyric acid (GABA). Data acquired during MRS experiments from this phantom provide results from which quantitative assessments can be made.

In this chapter we will review in detail the processing involved in computing an MRS absorption spectrum. We will utilize data acquired using a GE 1.5T MR scanner with a typical SVQ PRESS sequence on an 8 cc volume from the GE MRS phantom, and

show plots of intermediate results as a means of understanding each processing step. A block diagram of this processing scheme is shown in Fig. 7.

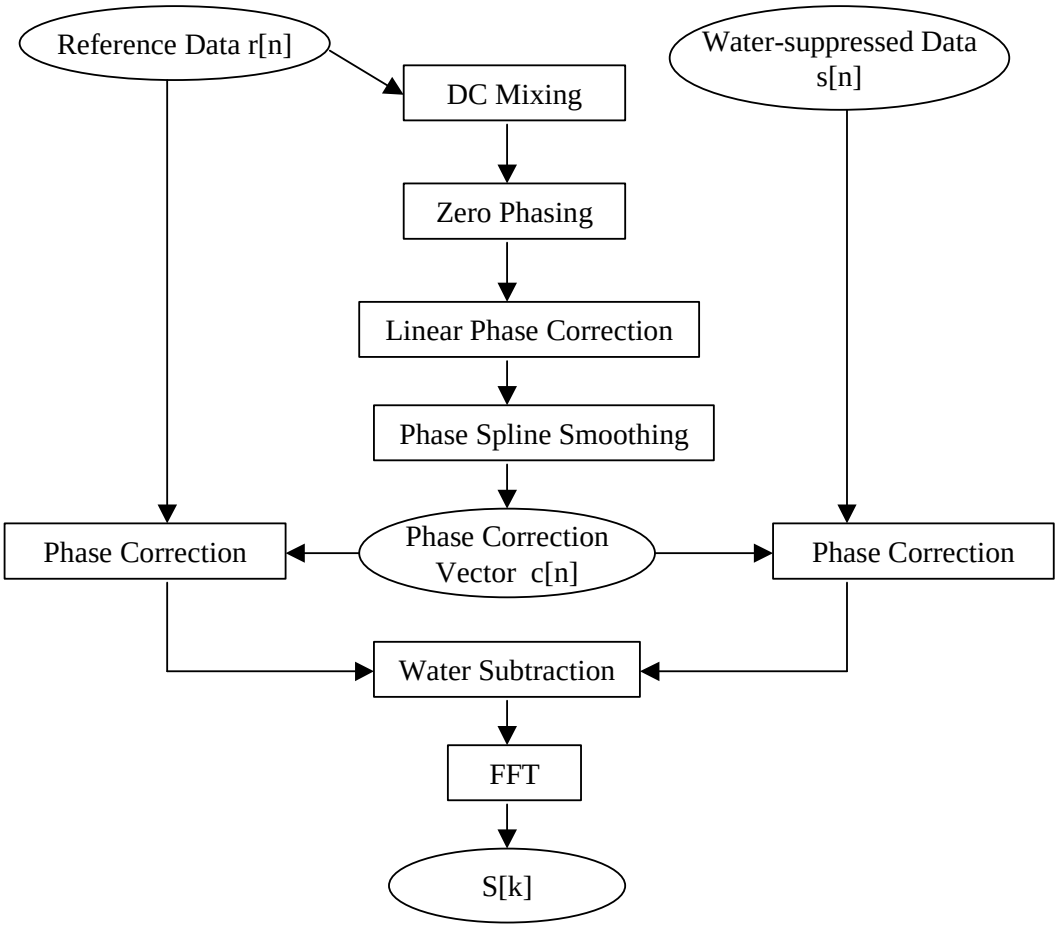

Fig. 7. Block diagram of MRS processing steps.

Non-water suppressed reference data sets are collected and the individual frames are averaged to obtain the discrete-time signal, $r[n]$, where n represents normalized time.

The reference data, $r[n]$, are used to compute a phase correction vector, $c[n]$, through the process of DC mixing, zero phasing, linear phase correction and phase spline smoothing. Water-suppressed data sets are collected and the individual frames are averaged to obtain $s[n]$. The phase correction vector, $c[n]$, is applied to both the reference data, $r[n]$, and the water-suppressed data, $s[n]$. After phase correction, residual water is removed from the water-suppressed data by subtracting an appropriately scaled version of $r[n]$ from $s[n]$. From the phase-corrected water-suppressed data with residual water removed, a Fourier transform is used to compute the MRS absorption spectrum.

3.1 Creating a Phase Correction Vector

After acquiring non-water suppressed reference data, a phase correction vector, $c[n]$, is created. This phase correction vector is applied to both water-suppressed and non-water-suppressed data during the creation of the MRS absorption spectrum. Data acquired during the reference acquisition without water suppression, $r[n]$, will be used to produce a phase correction vector, $c[n]$. Data from the reference acquisition will also be used to produce an estimate of the water signal necessary to perform residual water removal.

3.1.1 Reference Normalization

For this example, the number of reference frames acquired, N_{ref}, is 16. The number of complex data points for each reference frame, N, is 2048. All reference

frames are averaged to produce $r_{raw}[n]$, the averaged water reference data as shown in Fig. 8.

$$r_{raw}[n] = \frac{\sum_{i=1}^{N_{ref}} r_i[n]}{N_{ref}} \quad \text{for} \quad 0 \le n < N \tag{3.1}$$

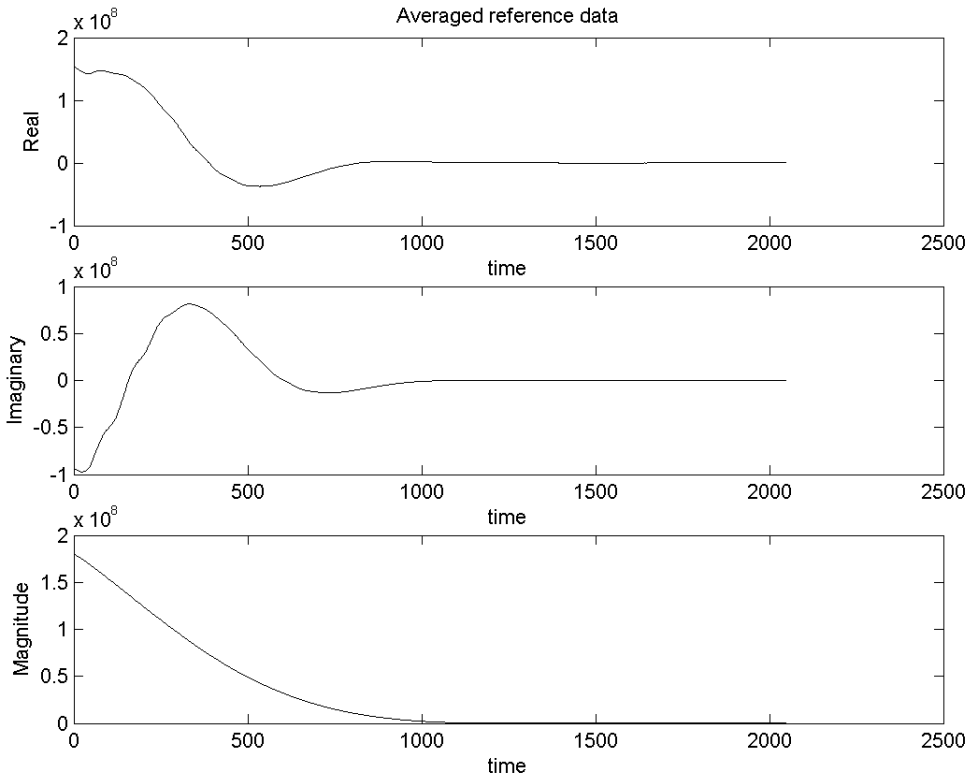

Fig. 8. Averaged non-water-suppressed reference data.

The averaged water reference signal, $r_{raw}[n]$, can also be normalized as:

$$r_{scale} = \max |r_{raw}[n]| \qquad \text{for } 0 \le n < N \tag{3.2}$$

$$r_{norm}[n] = \frac{r_{raw}[n]}{r_{scale}} \qquad \text{for } 0 \le n < N \tag{3.3}$$

3.1.2 DC Mixing

The most dominant frequency component *in vivo* is due to the signal from water, which typically shows up as a very low frequency or DC component since the center frequency for the acquisition is set to the signal from water during prescan. The purpose of DC mixing is to mitigate the effects of water components. First, the normalized averaged water reference data are discrete Fourier transformed:

$$R[k] = \mathcal{F}\{r_{norm}[n]\} \quad \text{for } 0 \le k < N \tag{3.4}$$

The frequency with the largest magnitude, ω_m, from $R[k]$ is determined:

$$k_{peak} = \max |R[k]| \qquad \text{for } 0 \le k < N \tag{3.5}$$

$$\omega_m = \frac{k_{peak} \cdot 2\pi}{N} \tag{3.6}$$

A complex-valued vector containing the frequency component of ω_m from Eqn. (3.6) can then be created as shown in Fig. 9.

$$c_1[n] = \cos(\omega_m n) - j\sin(\omega_m n) \qquad \text{for} \quad 0 \le n < N \tag{3.7}$$

or $$c_1[n] = e^{-j\omega_m n} \qquad \text{for} \quad 0 \le n < N \tag{3.8}$$

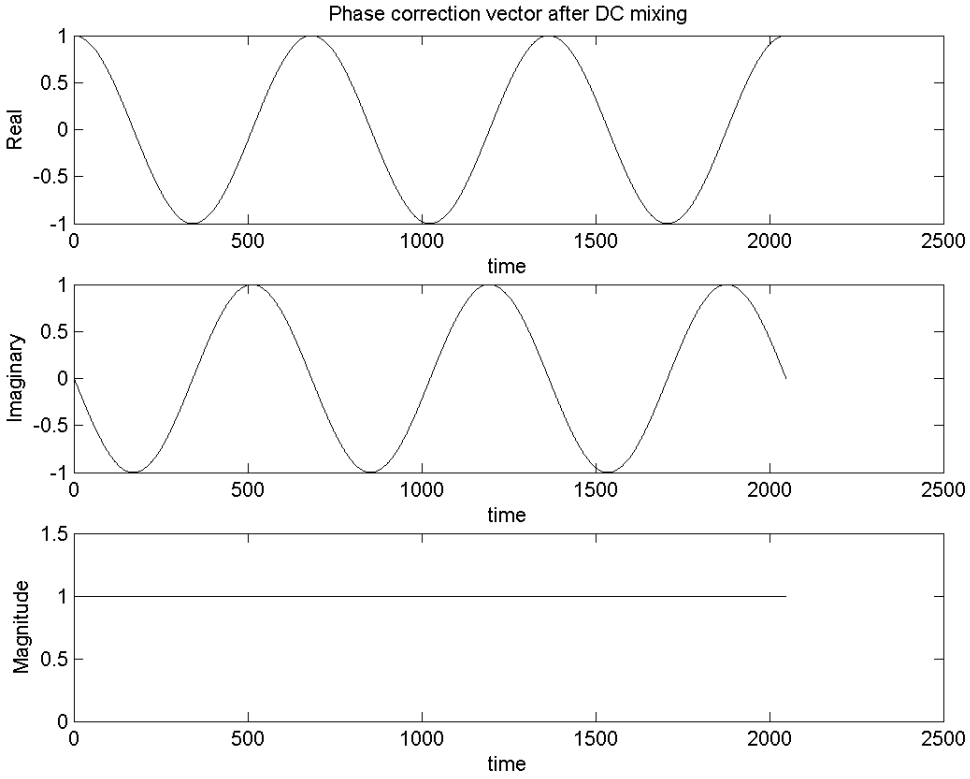

Fig. 9. Phase correction vector after DC mixing.

This result, $c_1[n]$, is applied to the normalized averaged water reference data, $r_{norm}[n]$, to generate $r_{dc}[n]$ as shown in Fig. 10.

$$r_{dc}[n] = r_{norm}[n] \cdot c_1[n] \qquad \text{for } 0 \le n < N \qquad (3.9)$$

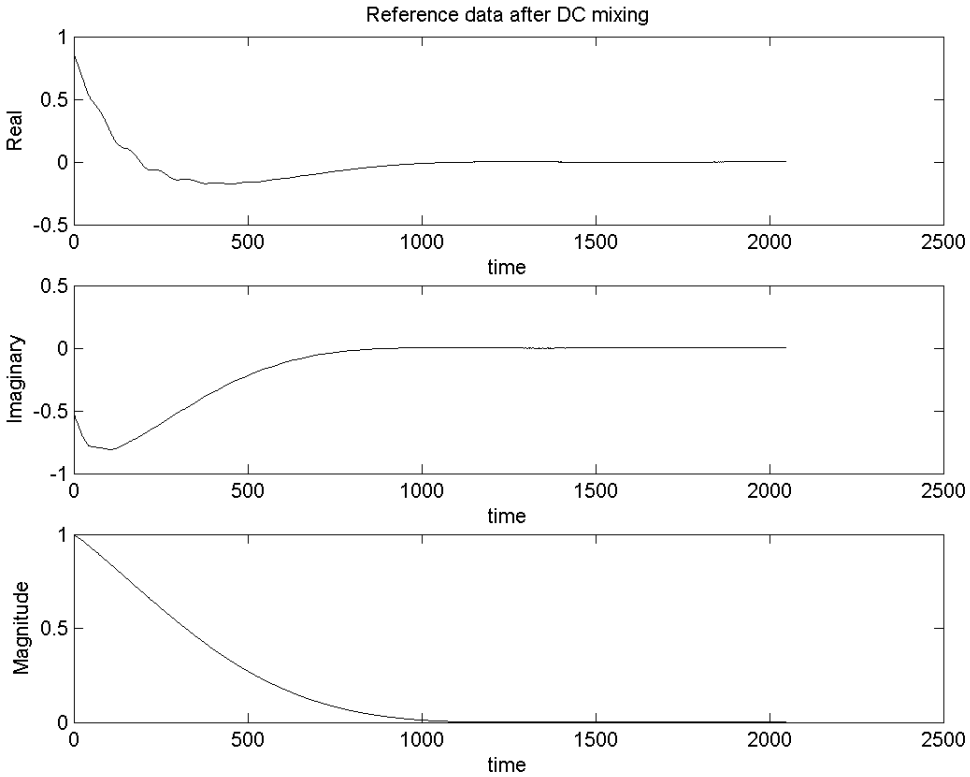

Fig. 10. Reference data after DC mixing.

3.1.3 Zero Phasing

By convention, the MRS absorption spectrum is represented as the real part of the Fourier transformation of the water-suppressed data. Prior to Fourier transformation,

water-suppressed data sets are adjusted so that they begin with zero phase. By adjusting the signal to begin with zero phase, it is more accurately represented by the cosine or real components of the Fourier transform [46]. This effectively yields an MRS absorption spectrum with sharp peaks with narrow line-widths[22],[26].

To adjust the reference signal so that it begins with zero phase, the complex conjugate (denoted by $*$) of the first element of the reference data after DC mixing, a complex-valued scalar, A_{zp}, is obtained and multiplied by $r_{dc}[n]$ and $c_1[n]$ as follows and shown in Fig. 11:

$$A_{zp} = r^*[0] \tag{3.10}$$

$$r_{zp}[n] = A_{zp} \cdot r_{dc}[n] \quad \text{for} \quad 0 \le n < N \tag{3.11}$$

This phase adjustment is also applied to the phase correction vector as shown in Fig. 12:

$$c_{zp}[n] = A_{zp} \cdot c_1[n] \quad \text{for} \quad 0 \le n < N \tag{3.12}$$

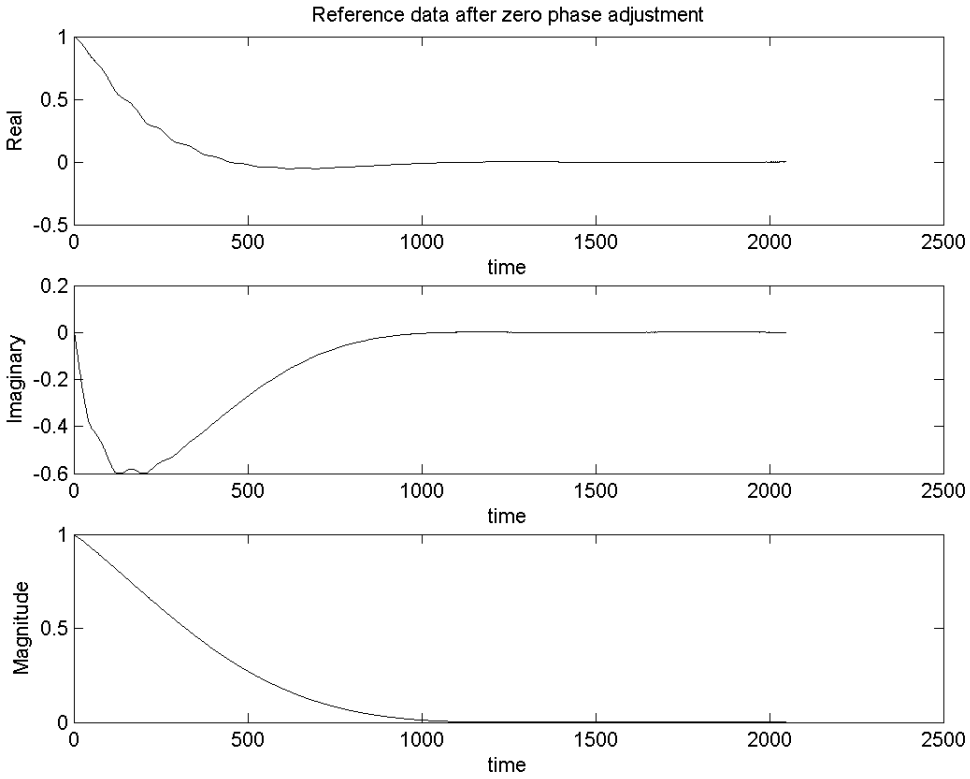

Fig. 11. Reference data after zero-phase adjustment.

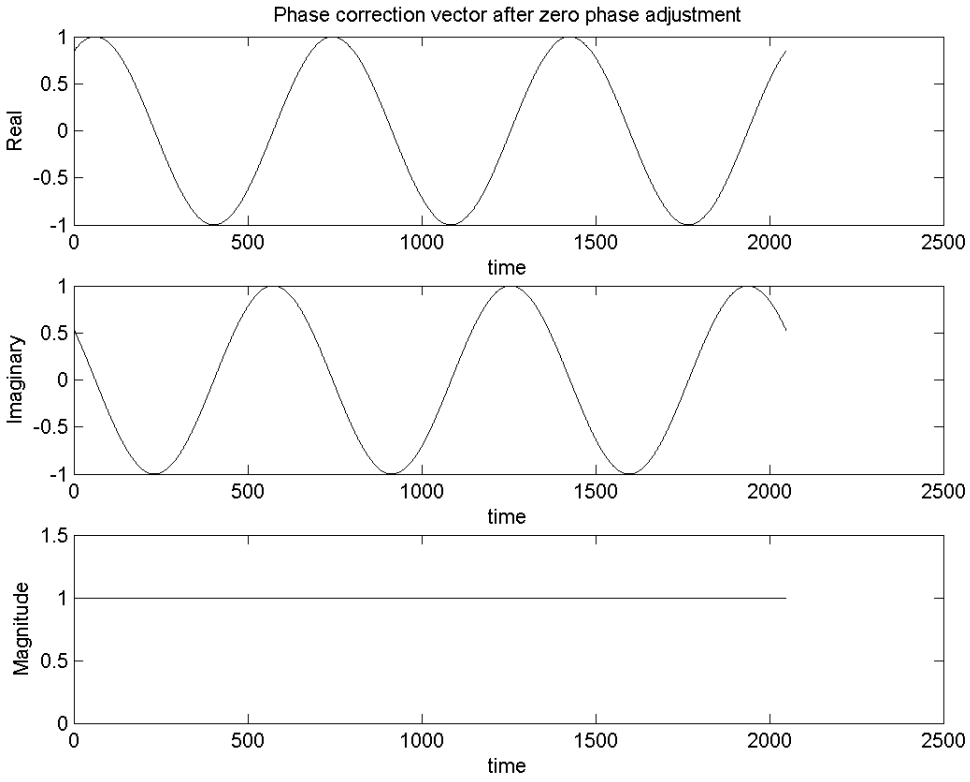

Fig. 12. Phase correction vector after zero-phase adjustment.

3.1.4 Linear Phase Correction

Several techniques have been proposed to correct for linear phase errors present in raw data acquired during MRS studies[47]-[51]. Linear phase correction provides a first-order estimate of phase errors present in the data so that they may be removed. One technique for doing this is to determine the unwrapped phase of $r_{zp}[n]$ as $\phi_{zp}[n]$. Then the last element of $\phi_{zp}[n]$, ϕ_{lp}, is used to generate a complex-valued vector for linear phase correction as shown in Fig. 13.

$$\phi_{zp}[n] = \text{unwrap}\{\sphericalangle r_{zp}[n]\} \qquad \text{for} \;\; 0 \le n < N \tag{3.13}$$

$$\phi_{lp} = \phi_{zp}[N-1] \tag{3.14}$$

The phase discontinuity shown at $time \approx 1500$ in Fig. 13 is located in a low SNR region of the data where the magnitude is close to zero. This phase oscillation is somewhat atypical, and perhaps an improved phase unwrapping technique could be used to eliminate the discontinuity in the unwrapped phase.

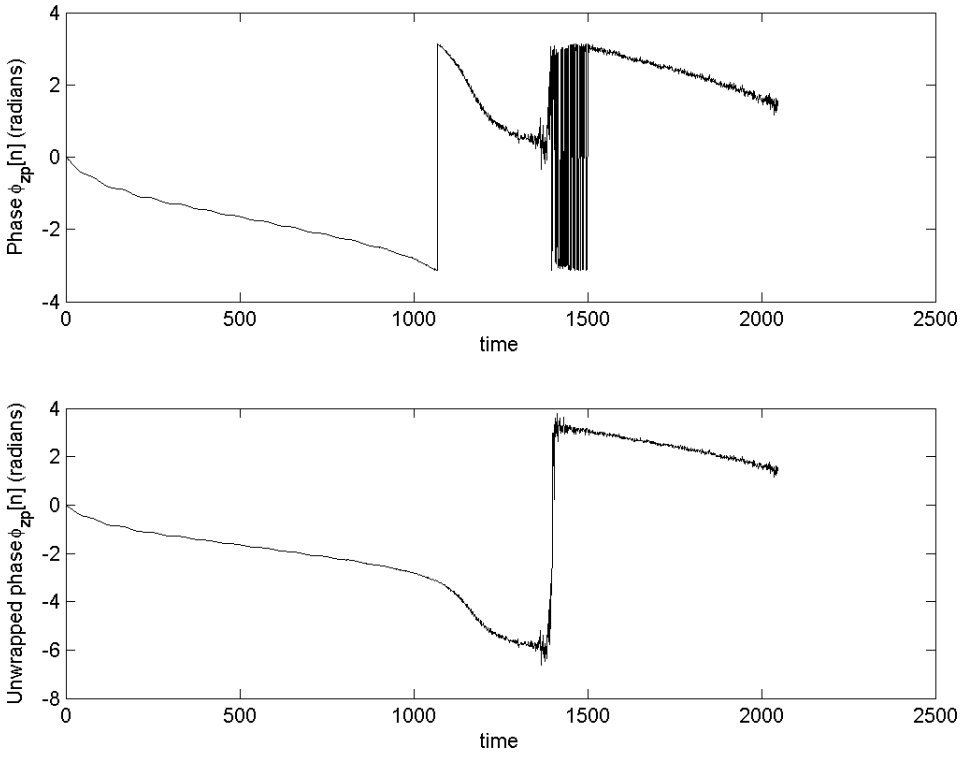

Fig. 13. Phase of DC mixed, zero-phase adjusted reference data.

After the last element of the unwrapped phase, ϕ_{lp}, is determined, a complex-valued linear phase correction vector, $lp[n]$, shown in Fig. 14 is generated as follows:

$$\omega_{lp} = \phi_{lp} \cdot \frac{2\pi}{N} \tag{3.15}$$

$$lp[n] = \cos(\omega_{lp} n) - j \sin(\omega_{lp} n) \quad \text{for} \quad 0 \le n < N \tag{3.16}$$

or $$lp[n] = e^{-j\omega_{lp} n} \quad \text{for} \quad 0 \le n < N \tag{3.17}$$

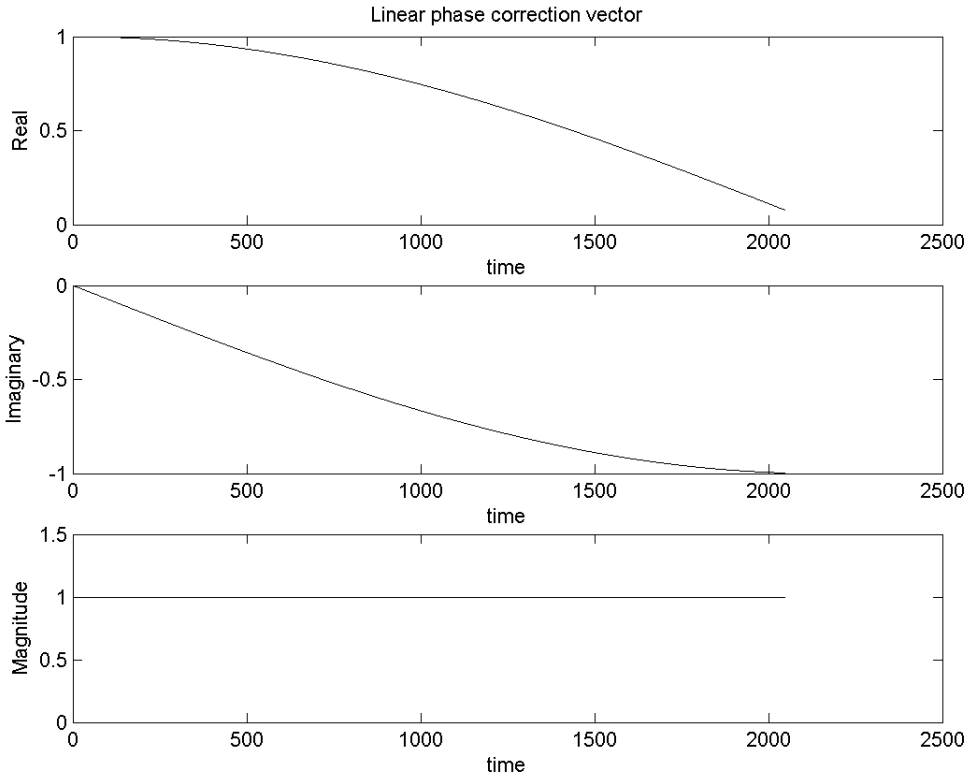

Fig. 14. Linear phase correction vector determined from reference data.

The linear phase correction vector, $lp[n]$, can then be applied to the DC-mixed, zero-phase adjusted reference data, $r_{zp}[n]$, as well as to the DC-mixed, zero-phase adjusted phase correction vector, $c_{zp}[n]$, as follows:

$$r_{lp}[n] = r_{zp}[n] \cdot lp[n] \qquad \text{for } 0 \le n < N \tag{3.18}$$

$$c_{lp}[n] = c_{zp}[n] \cdot lp[n] \qquad \text{for } 0 \le n < N \tag{3.19}$$

or
$$c_{lp}[n] = A_{zp} \cdot c_1[n] \cdot lp[n] \qquad \text{for } 0 \le n < N \tag{3.20}$$

$$c_{lp}[n] = A_{zp} \cdot e^{-j\omega_m n} \cdot e^{-j\omega_{lp} n} \qquad \text{for } 0 \le n < N \tag{3.21}$$

$$c_{lp}[n] = A_{zp} \cdot e^{-j(\omega_m + \omega_{lp}) n} \qquad \text{for } 0 \le n < N \tag{3.22}$$

The resulting reference data, $r_{lp}[n]$, is shown in Fig. 15 and the phase correction vector, $c_{lp}[n]$, is shown in Fig. 16.

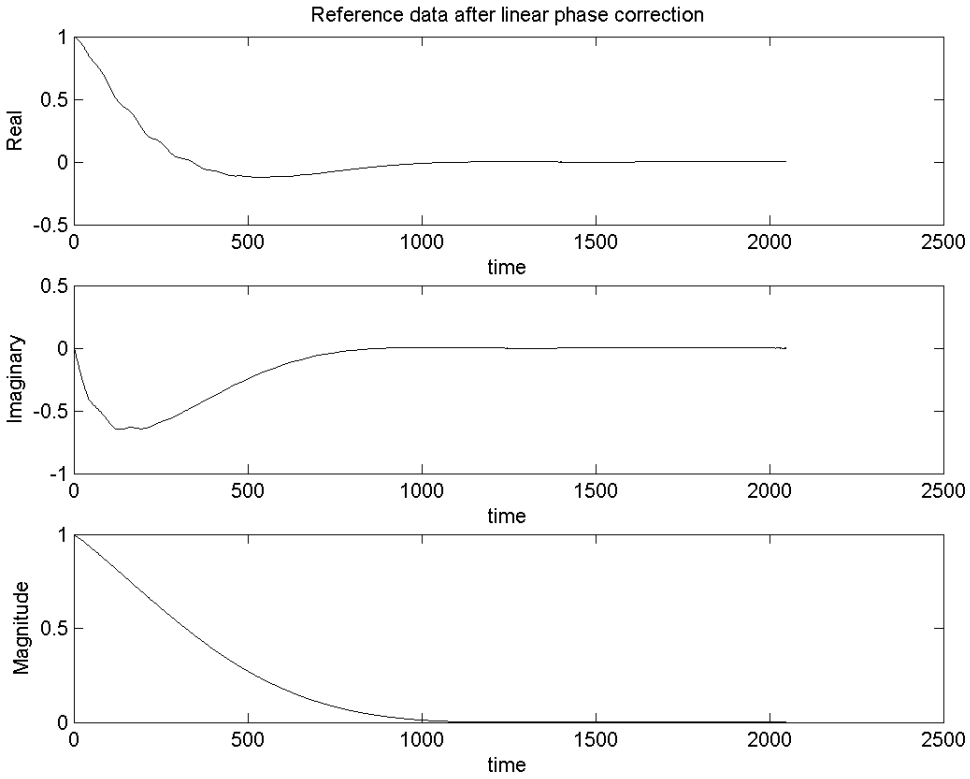

Fig. 15. DC mixed, zero-phase adjusted, linear phase-corrected reference data.

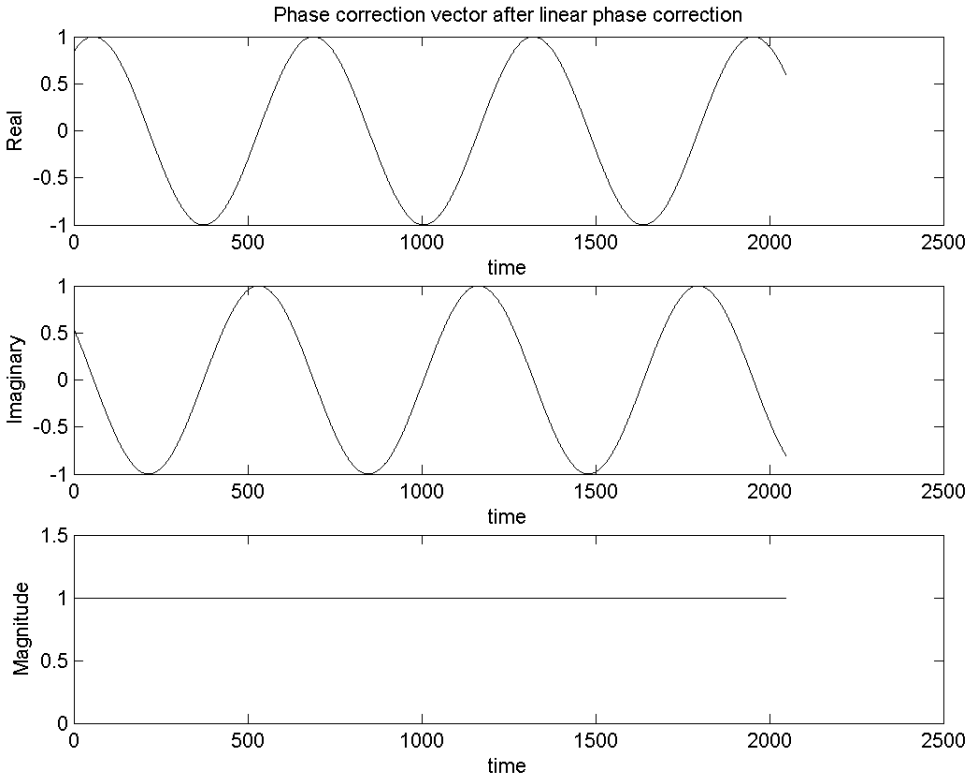

Fig. 16. DC-mixed, zero-phase, linear phase-corrected phase correction vector.

3.1.5 Phase Spline Smoothing

The final step in creating the phase correction vector, $c[n]$, is to smooth the unwrapped phase of the phase correction vector determined by DC mixing, zero phasing and linear phase correction. For data frames with significant noise, or phase discontinuities, this step provides a more stable phase correction vector. The smoothing can be done using the spline method proposed by de Boor [52] (see also [53]). The unwrapped phase of $r_{lp}[n]$ can be determined as:

$$\phi_{lp}[n] = \text{unwrap}\{\measuredangle r_{lp}[n]\} \qquad \text{for} \quad 0 \le n<N \qquad (3.23)$$

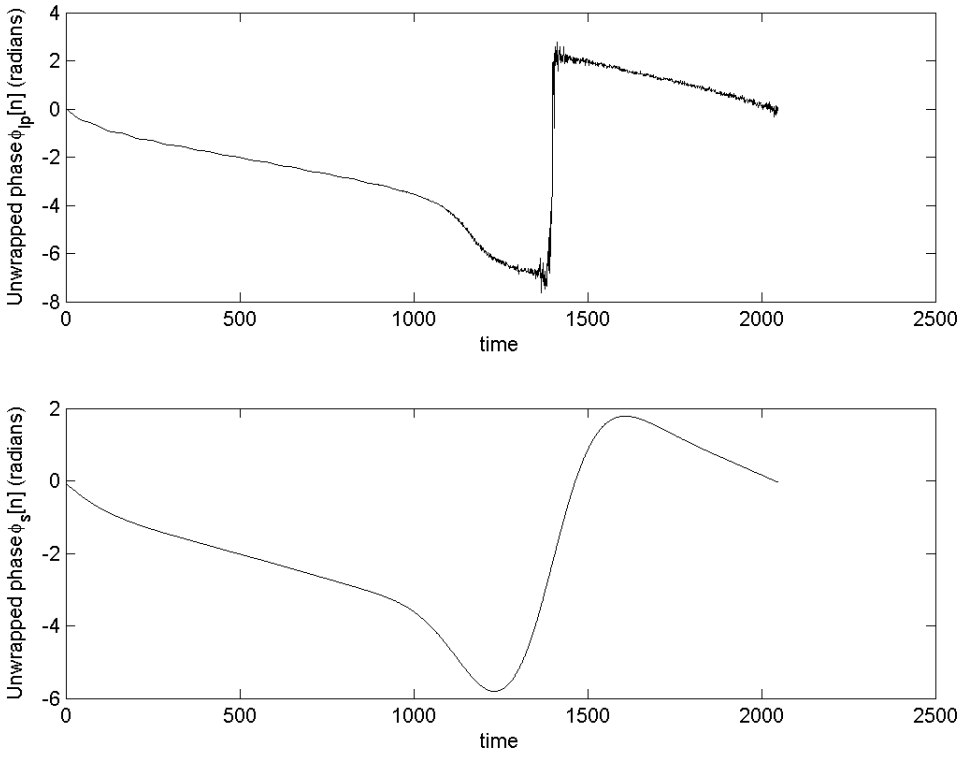

Fig. 17. Spline smoothed phase of phase-corrected reference data.

The unwrapped phase, $\phi_{lp}[n]$, is smoothed to create $\phi_s[n]$ as shown in Fig. 17.

$$\phi_s[n] = smooth\{\phi_{lp}[n]\} \qquad \text{for} \quad 0 \le n < N \qquad (3.24)$$

where $smooth\{\}$ is a spline smoothing function as described by de Boor [52].

Once the smoothed phase has been calculated, the complex-valued vector with smoothed phase, $p_f[n]$, as shown in Fig. 18 is created as follows:

$$p_f[n] = \cos(\phi_s[n]) - j\sin(\phi_s[n]) \qquad \text{for} \quad 0 \le n < N \qquad (3.25)$$

or $$p_f[n] = e^{-j\phi_s[n]} \qquad \text{for} \quad 0 \le n < N \qquad (3.26)$$

and this is used to generate the final phase correction vector, $c[n]$, shown in Fig. 19 that will be used in spectroscopy processing:

$$c[n] = c_{lp}[n] \cdot p_f[n] \qquad \text{for} \quad 0 \le n < N \qquad (3.27)$$

or

$$c[n] = A_{zp} \cdot c_1[n] \cdot lp[n] \cdot p_f[n] \qquad \text{for} \quad 0 \le n < N \qquad (3.28)$$

$$c[n] = A_{zp} \cdot e^{-j\omega_m n} \cdot e^{-j\omega_{lp} n} \cdot e^{-j\phi_s[n]} \qquad \text{for} \quad 0 \le n < N \qquad (3.29)$$

$$c[n] = A_{zp} \cdot e^{-j[(\omega_m + \omega_{lp})n + \phi_s[n]]} \qquad \text{for} \quad 0 \le n < N \qquad (3.30)$$

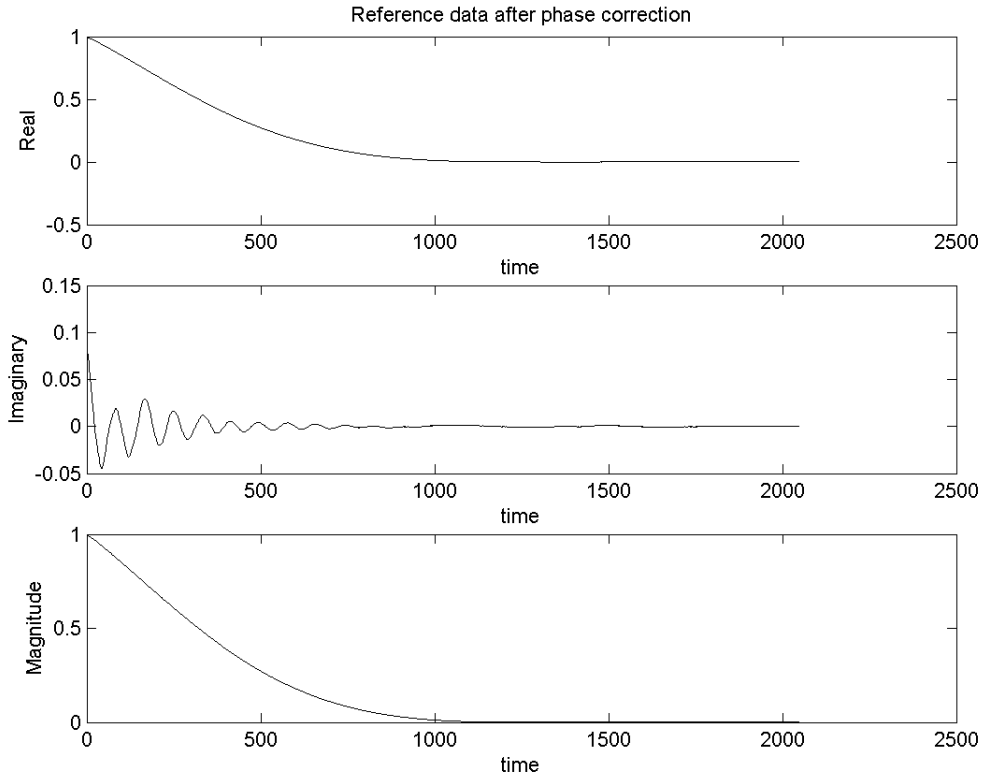

Fig. 18. Spline smoothed phase-corrected reference data.

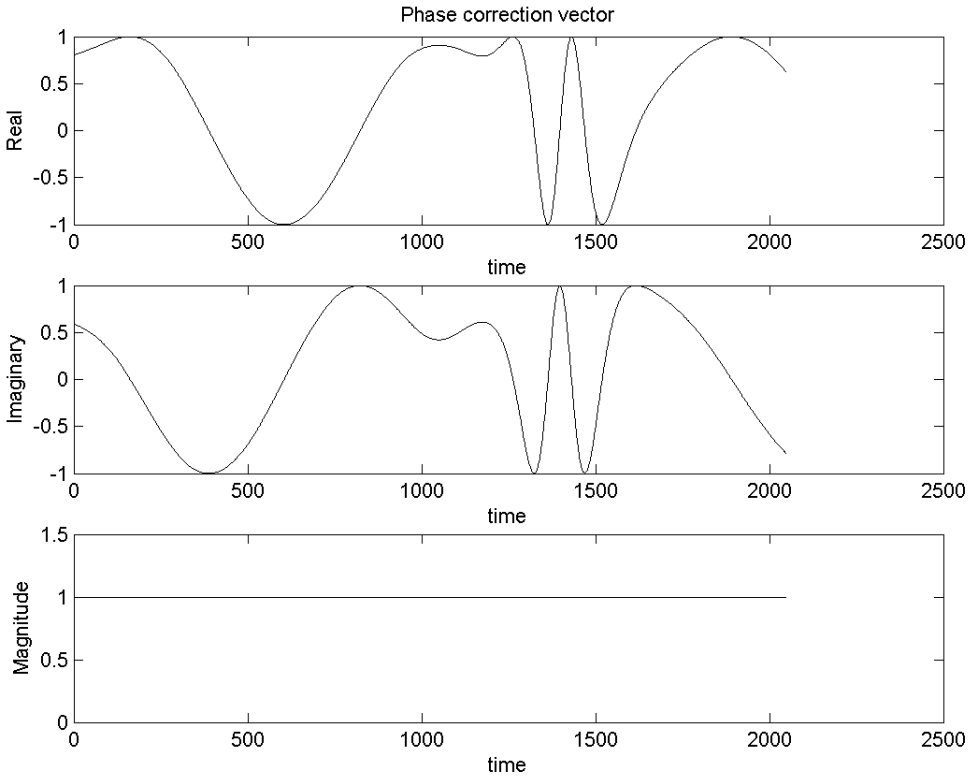

Fig. 19. Final phase correction vector.

3.2 Computation of Absorption Spectrum

Typical SVQ scans involve acquiring two different types of data: a set of reference data from which water is not suppressed, and also a set of water-suppressed data from which signals of metabolites can be identified. An MRS absorption spectrum is created by applying the phase correction vector, $c[n]$, to both the non-water-suppressed

reference data as well as the water-suppressed data. The steps involved to create an MRS absorption spectrum are described in the following sections.

3.2.1 Averaging of Water-Suppressed Signal

If the number of water-suppressed frames acquired for a particular scan is 32 and the NEX for this scan is 2, then a total of 16, or, N_{sig}, water-suppressed frames, $s_i[n]$, are available. If the number of complex data points for each frame is 2048, or N, then the water-suppressed signal can be averaged to produce $s_{raw}[n]$ as shown in Fig. 20 in the following manner:

$$s_{raw}[n] = \frac{\sum_{i=1}^{N_{sig}} s_i[n]}{N_{sig}} \quad \text{for } 0 \le n < N \tag{3.31}$$

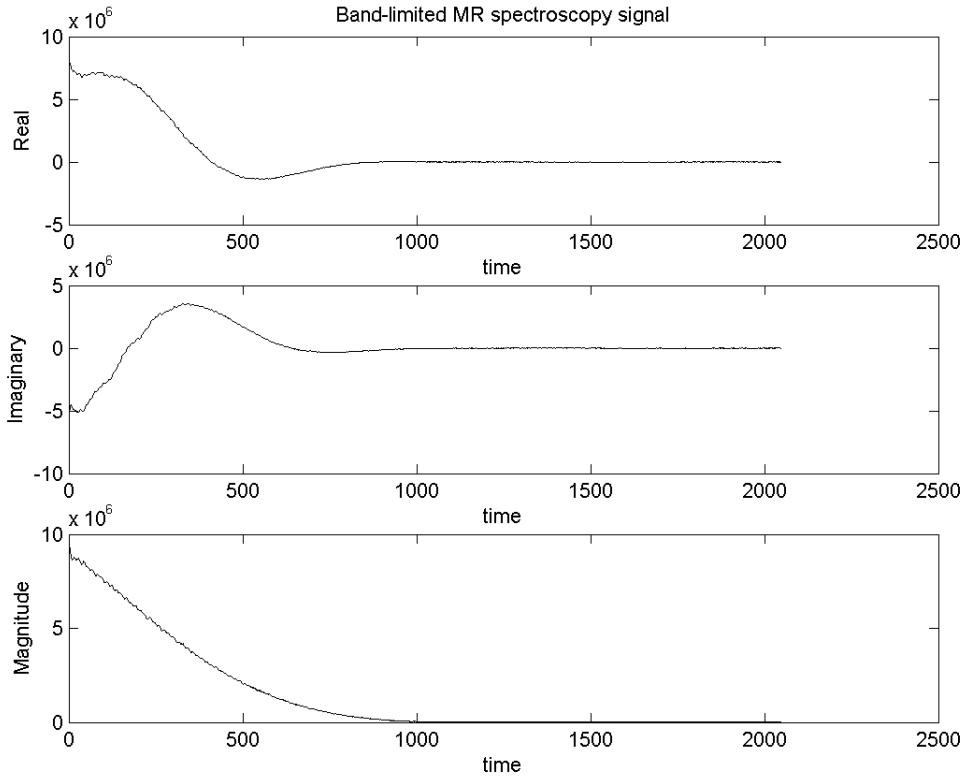

Fig. 20. Water-suppressed MR spectroscopy signal.

3.2.2 Applying Phase Correction Vector

The phase correction vector is applied to both the averaged water-suppressed signal, $s_{raw}[n]$, as shown in Fig. 21 and the averaged reference signal, $r_{raw}[n]$ as shown in Fig. 22 :

$$r_{pc}[n] = r_{raw}[n] \cdot c[n] \qquad \text{for} \quad 0 \le n < N \qquad (3.32)$$

where $r_{raw}[n]$ is given by Eqn. (3.1), and

$$s_{pc}[n] = s_{raw}[n] \cdot c[n] \qquad \text{for} \quad 0 \le n < N \qquad (3.33)$$

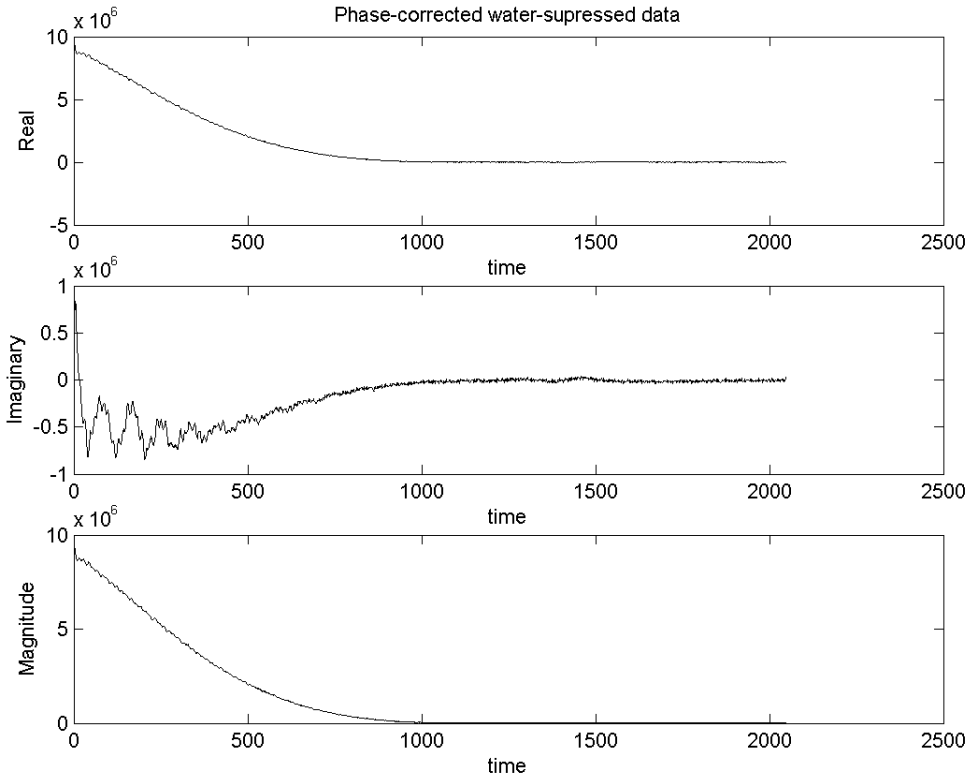

Fig. 21. Water-suppressed MRS data with phase correction applied.

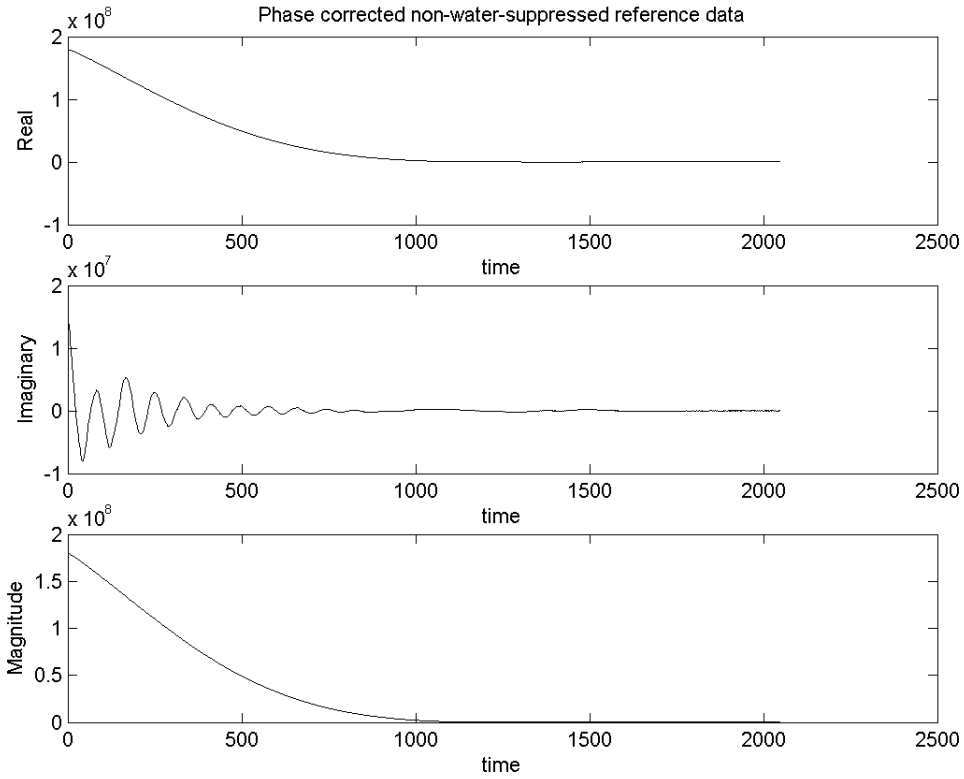

Fig. 22. Reference data with phase correction applied.

3.2.3 Residual Water Removal

The water-suppressed signal, $s_{pc}[n]$, is now subtracted from the reference signal, $r_{pc}[n]$, to create a pure water signal, $s_{pw}[n]$ as shown in Fig. 23 (see [54]-[56]).

$$s_{pw}[n] = r_{pc}[n] - s_{pc}[n] \qquad \text{for} \quad 0 \le n < N \tag{3.34}$$

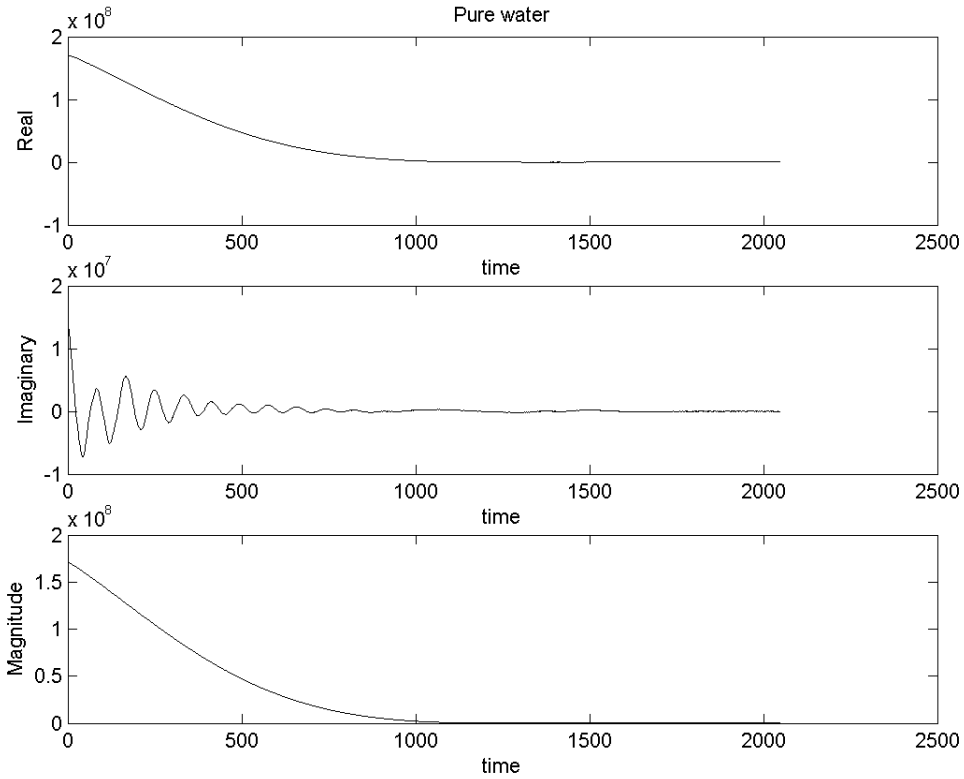

Fig. 23. Pure water signal.

Next a set of processing steps is performed to determine a scale factor, a_{scale}, to allow a scaled version of the pure water as shown in Fig. 23 to be subtracted from the water-suppressed data from which an MRS absorption specturm will be computed. First, the pure water signal, $s_{pw}[n]$, and the phase corrected water suppressed signal, $s_{pc}[n]$ are multiplied by an alternation vector:

$$s_{pwa}[n] = s_{pw}[n] \cdot a[n] \qquad \text{for} \quad 0 \le n < N \tag{3.35}$$

$$s_{pca}[n] = s_{pc}[n] \cdot a[n] \qquad \text{for} \quad 0 \le n < N \tag{3.36}$$

where

$$a[n] = \begin{Bmatrix} 1 & \text{for } n \text{ even} \\ -1 & \text{for } n \text{ odd} \end{Bmatrix} \quad \text{for} \quad 0 \le n < N \tag{3.37}$$

Next a hanning window as shown in Fig. 24 is applied.

$$s_{pww}[n] = s_{pwa}[n] \cdot w_1[n] \qquad \text{for} \quad 0 \le n < N \tag{3.38}$$

$$s_{pcw}[n] = s_{pca}[n] \cdot w_1[n] \qquad \text{for} \quad 0 \le n < N \tag{3.39}$$

where

$$w_1[n] = \begin{Bmatrix} 0.5\left(1\text{-}\cos\left(\dfrac{2\pi\left(n + K/2\right)}{K}\right)\right) & \text{for} \quad 0 \le n < K/2;\ K = 1280 \\ 0 & \text{otherwise} \end{Bmatrix} \tag{3.40}$$

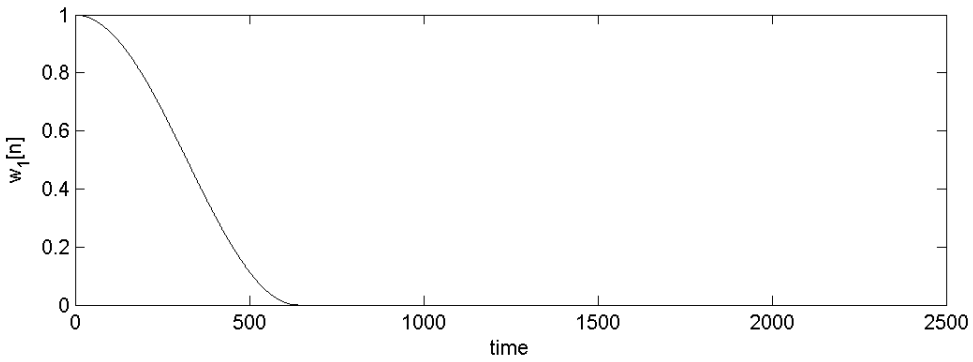

Fig. 24. Apodization window for residual water removal.

A Fourier transform is then performed on the apodized pure water and also on the apodized, phase-corrected, water-suppressed signal as shown in Fig. 25.

$$S_w[k] = \mathcal{F}\{s_{pww}[n]\} \qquad \text{for} \quad 0 \le k < N \tag{3.41}$$

$$S_s[k] = \mathcal{F}\{s_{pcw}[n]\} \qquad \text{for} \quad 0 \le k < N \tag{3.42}$$

and the scale factor, a_{scale}, is then determined from the ratio of the maximum value of the spectrum obtained from the water-suppressed signal to that of pure water .

$$a_w = \max(\mathrm{Re}\{S_w[k]\}) \qquad \text{for} \quad 0 \le k < N \tag{3.43}$$

$$a_s = \max(\mathrm{Re}\{S_s[k]\}) \qquad \text{for} \quad 0 \le k < N \tag{3.44}$$

$$a_{scale} = \frac{a_s}{a_w} \tag{3.45}$$

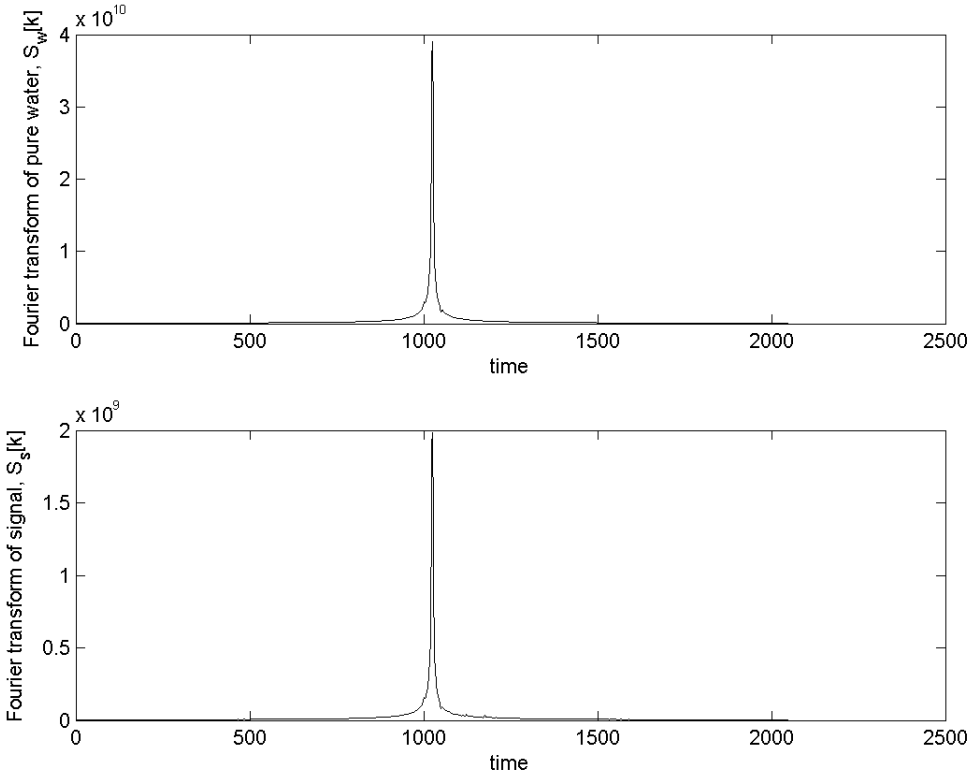

Fig. 25. Fourier transform of pure water and water-suppressed signal.

After the scale factor, a_{scale}, has been determined, it is applied to the windowed, pure water signal, $s_{pww}[n]$, which is then subtracted from the windowed, water-suppressed signal, $s_{pcw}[n]$, to generate the water-subtracted pure signal, $s_s[n]$.

$$s_w[n] = s_{pww}[n] \cdot a_{scale} \qquad \text{for } 0 \le n < N \tag{3.46}$$

$$s_s[n] = s_{pcw}[n] - s_w[n] \qquad \text{for } 0 \le n < N \qquad (3.47)$$

The water-suppressed signal shown in Fig. 20 after phase-correction and residual water removal is given by Eqn. (3.47) and shown in Fig. 26.

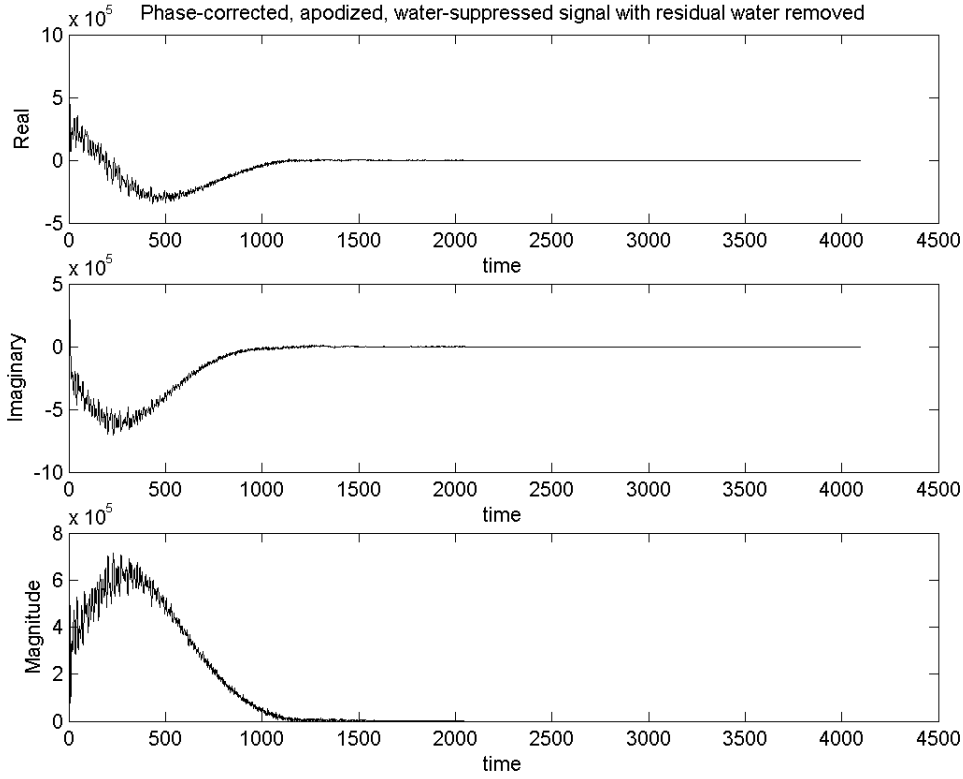

Fig. 26. Phase-corrected water-suppressed signal with residual water removed.

3.2.4 Fourier Transform to Compute Absorption Spectrum

A hanning window shown in Fig. 27 is then applied to the phase-corrected water-suppressed signal with residual water removed.

$$s_{sw}[n] = s_s[n] \cdot w_2[n] \qquad \text{for} \quad 0 \le n < N \tag{3.48}$$

where

$$w_1[n] = \begin{cases} 0.5\left(1\text{-}\cos\left(\dfrac{2\pi\left(n + K/2\right)}{K}\right)\right) & \text{for} \quad 0 \le n < K/2;\ K = 4096 \\ 0 & \text{otherwise} \end{cases} \tag{3.49}$$

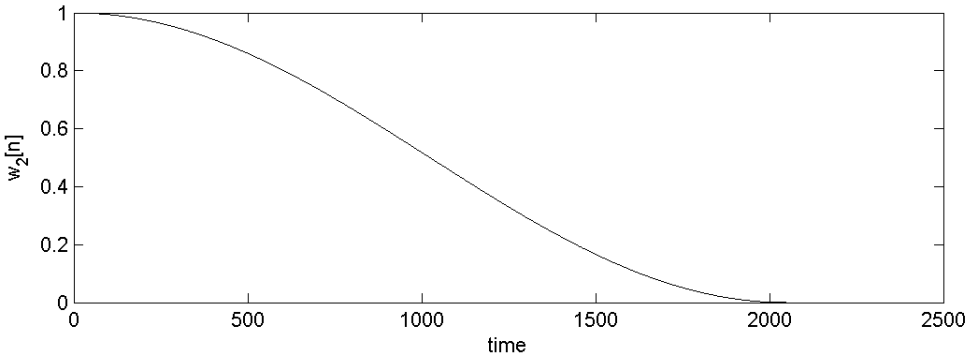

Fig. 27. Apodization window applied prior to final Fourier transform.

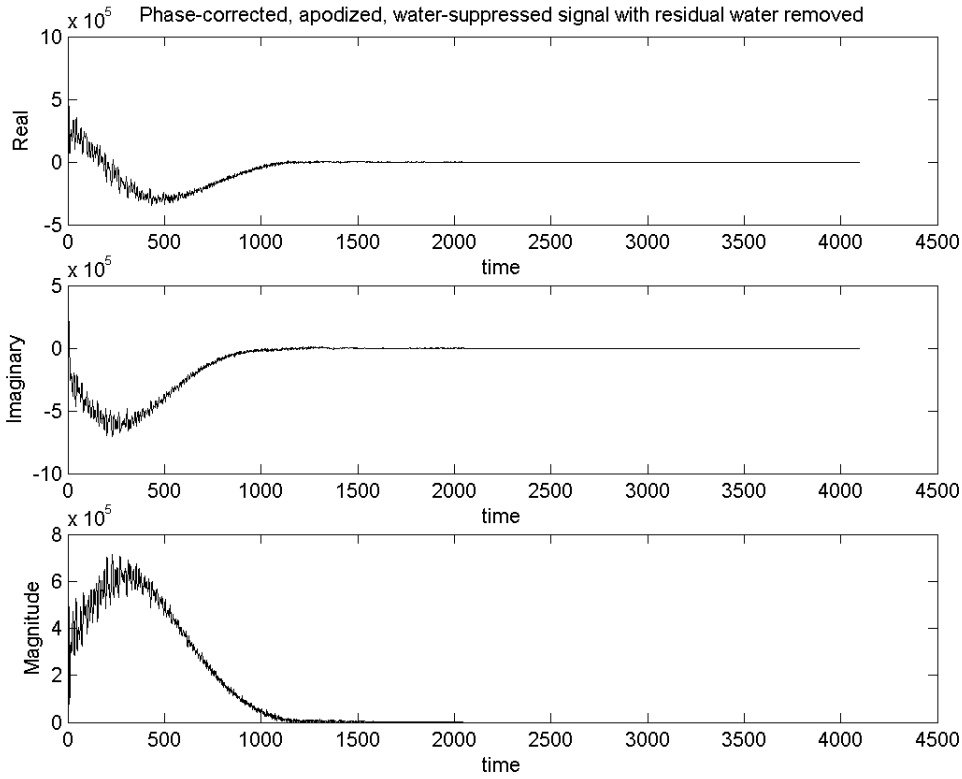

Fig. 28. Phase-corrected, apodized signal with residual water removed.

After apodization, the signal is zero padded as shown in Fig. 28 and Fourier transformed.

$$s_{zp}[n] = \begin{cases} s_{sw}[n] & \text{for} \quad 0 \le n < N \\ 0 & \text{for} \quad N \le n < 2N \end{cases} \qquad (3.50)$$

$$S_f[k] = \mathcal{F}\{s_{zp}[n]\} \qquad \text{for} \quad 0 \le k < 2N \qquad (3.51)$$

The final results, $S_f[k]$, are comprised of the MRS absortion spectrum represented as the real-valued results of the Fourier transform, and the MRS dispersion spectrum represented as the imaginary-valued results of the Fourier transform[22].

$$MRS\ absorption\ spectrum = \mathrm{Re}\{S_f[k]\} \tag{3.52}$$

$$MRS\ dispersion\ spectrum = \mathrm{Im}\{S_f[k]\} \tag{3.53}$$

A typical MRS absorption spectrum used for clinical diagnosis would be displayed in a range from approximately 4.5 ppm to 0.0 ppm.

Fig. 29 is an example of an MRS absorption spectrum obtained from the GE MRS phantom. This MRS experiment was performed using PRESS on an 8 cc volume, with a TE of 35msec and a TR of 1500msec, 2 NEX, 8 reference frames and 16 water-suppressed (CHESS) frames. The total scan time was 1 minute and 18 seconds. The largest peak occurring near 2.0 ppm is from NAA. The doublet near 1.3 ppm is from lactate. The large peak near 3.0 ppm is from creatine. The peak near 3.2 ppm is from choline. The peak near 3.55 ppm is from myoinositol.

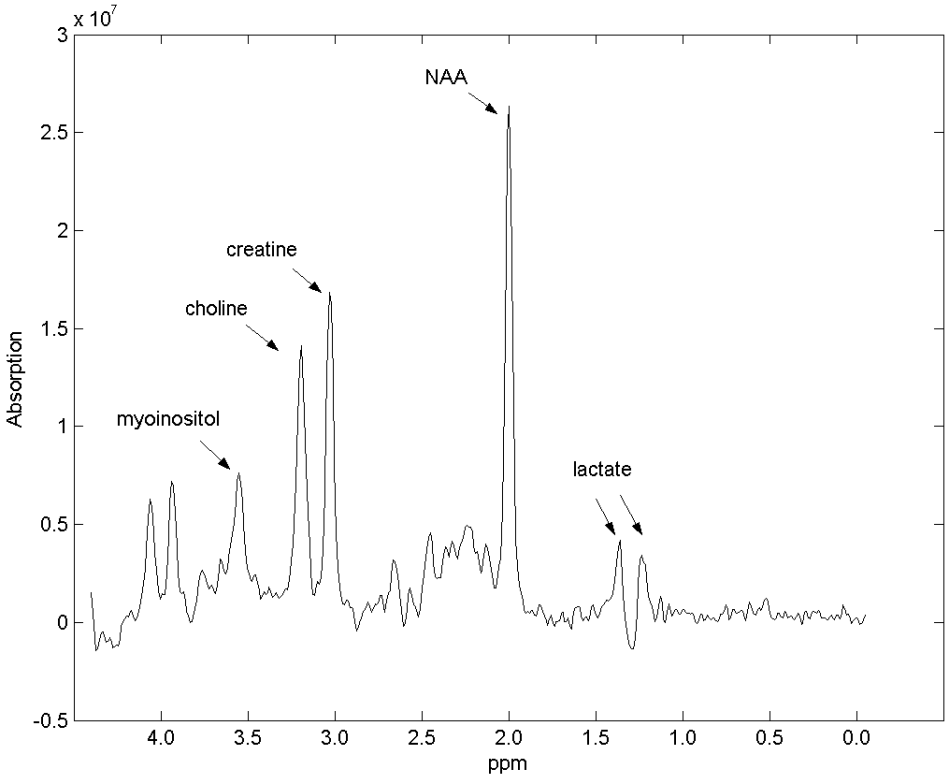

Fig. 29. Absorption spectrum of GE MRS phantom.

For one-dimensional plots representing the absorption spectrum, the peak area or the peak height (for narrow line-widths), representing a particular frequency component can be calculated and compared to other peaks[24]. Often, the creatine peak is used as a reference for *in vivo* MRS studies[22]. In some cases, the mere presence of a particular peak may indicate a clinically significant diagnosis. For example, lactate, which shows up as a doublet near 1.3 ppm, should never occur in a normal human brain, so its presence would potentially indicate damage from acute stroke[17].

3.3 Summary

In this chapter we have reviewed the signal processing steps associated with the creation of an MRS absorption spectrum for SVQ scans. This involves the creation of a phase correction vector from reference data, and the subsequent phase correction of both the non-water-suppressed reference data and the water-suppressed data containing spectral information for the metabolites of interest. Residual water is removed from water-suppressed data by subtracting appropriately scaled reference data prior to creating the MRS absorption spectrum by Fourier transformation. The processing discussed in this chapter provides a foundation upon which spectral estimation improvements will be introduced in later chapters.

Chapter 4

Weighted 2D Spectral Estimation Methods

Several nonparametric techniques in addition to the Fourier transform can be used for spectral analysis of MRS studies. Two nonparametric techniques that can be used for MRS analysis, which employ adaptive filter bank approaches, are the one-dimensional Capon method introduced by J. Capon in 1969 [57],[58] and the one-dimensional APES method [59],[58]. The filter bank approach to power spectral estimation employs a bandpass filter with a fixed bandwidth to estimate the power spectral density (PSD) of a given frequency component. The basic periodogram, is a filter bank approach based on the standard Fourier transform [60] -[62].

Capon and APES are filter bank approaches that improve the estimate of the PSD by creating one data-dependent bandpass filter for each spectral point being estimated[63] - [73]. The bandpass filter is created by a least-squares minimization process, which attempts to minimize the total output power of the filter, yet pass the frequency component of interest unaltered. A second least-squares process is then implemented to estimate the amplitude of the filtered signal. Capon and APES provide more accurate spectral estimates with lower sidelobes and narrower spectral peaks than the Fourier transform based periodogram techniques[59].

The Fourier transform, one-dimensional Capon analysis and one-dimensional APES analysis fail to take into account, at least in a direct way, any damping associated with each signal component. Thus in these methods, because the detected amplitude will be altered by the damping, it is difficult to accurately estimate the amplitudes of the various signal components. To alleviate this problem, two-dimensional Capon and APES techniques have recently been developed. Stoica and Sundin [2] have shown that the 2D Capon and 2D APES methods provide high-resolution two-dimensional MRS results showing both frequency, ω, and damping, σ. Frigo, Heinen, et al., have demonstrated that these methods have clinical utility for MRS since they provide information about T2*, the total transverse relaxation time, for each metabolite [74].

In this chapter we propose three new 2D spectral estimation techniques, which have as their basis the 2D Capon and 2D APES methods introduced by Stoica and Sundin. These are the weighted 2D Capon method, the weighted 2D APES method and the combined weighted 2D APES / 2D Capon method. These techniques incorporate several new design parameters, the proper choice of which may lead to reduced processing time, better peak identification, and/or better estimates of spectral amplitude, phase, and damping.

The theoretical development of these methods closely parallels that followed by Stoica and Sundin [2] in the development of the 2D Capon and 2D APES methods. For brevity, we will not separately review the 2D Capon and 2D APES methods. Rather, since these are special cases of our new techniques, we will develop the new techniques

in detail and at appropriate places point out how our new techniques reduce to the standard 2D Capon and 2D APES methods. Because these derivations are of necessity quite involved, as a convenience for the reader a summary of the salient results is presented in Sect. 4.8 and useful derivations are included in the Appendix (see Sec. A.1 – Sec. A.2). Finally, it is noted that these techniques are generally applicable to a wide variety of spectral estimation problems, not simply to MRS.

4.1 Discrete Spectrum of Sum of Continuous-Time Damped Complex Sinusoids

Consider a continuous-time signal consisting of R damped complex sinusoids:

$$x(t) = \sum_{r=1}^{R} S_r e^{-\sigma_r' t} e^{j\omega_r' t}, \quad t \geq 0 \tag{4.1}$$

where $\sigma_r', \omega_r' \geq 0$ and $S_r = A_r e^{j\theta r}$ is complex with magnitude $A_r > 0$ and phase θ_r. We may also consider a sampled version of $x(t)$ with sampling period T:

$$\begin{aligned} x[n] = x(nT) &= \sum_{r=1}^{R} S_r e^{-\sigma_r' Tn} e^{j\omega_r' Tn}, \quad n \geq 0 \\ &= \sum_{r=1}^{R} S_r e^{-\sigma_r n} e^{j\omega_r n} \\ &= \sum_{r=1}^{R} S_r p_r^n \end{aligned} \tag{4.2}$$

where $\sigma_r = \sigma_r' T$, $\omega_r = \omega_r' T$, $p_r = e^{-\sigma_r} e^{j\omega_r}$, and where T is chosen such that $0 \leq \omega_r < \pi$, for all r, thus satisfying the Nyquist sampling theorem.

One might consider analyzing $x[n]$ using the z-transform, but this presents certain problems. First, $X(z) = \mathcal{Z}\{x[n]\}$ has poles at p_r, i.e., it has a value of ∞ at $z = p_r$. Furthermore, $X(z)$ converges only outside a circle with radius equal to the magnitude of the largest pole.

Alternatively, we may consider using the discrete-time Fourier transform $X(e^{j\omega}) = \mathcal{F}\{x[n]\}$ to study $x[n]$. $X(e^{j\omega})$ converges as long as $\sigma_r > 0$ for all r. However, the effects of σ_r on the magnitude of the various components of $x[n]$ are difficult to quantify using $X(e^{j\omega})$. We therefore define the spectrum $S(\sigma,\omega) = \mathcal{S}\{x[n]\}$ of $x[n]$ as follows:

$$\begin{aligned}\mathcal{S}\{x[n]\} = S(\sigma,\omega) &= \sum_{r=1}^{R} S_r \delta_{\sigma-\sigma_r} \delta_{\omega-\omega_r} \\ &= \sum_{r=1}^{R} A_r e^{j\theta_r} \delta_{\sigma-\sigma_r} \delta_{\omega-\omega_r}\end{aligned} \tag{4.3}$$

where

$$\delta_\rho = \begin{cases} 1, & \rho = 0 \\ 0, & \rho \neq 0 \end{cases} \tag{4.4}$$

We may also extend this definition to a general complex signal, $x[n]$, (implicitly assuming it to be composed of damped complex sinusoids) in the following manner:

$$\mathcal{S}\{x[n]\} = S(\sigma,\omega) = A(\sigma,\omega)e^{j\theta(\sigma,\omega)} \tag{4.5}$$

although at this point it is not clear how to define $A(\sigma,\omega)$ and $\theta(\sigma,\omega)$. We will address this issue shortly.

We may graphically represent $S(\sigma,\omega)$, given by Eqn. (4.3) (for the signal $x[n]$ given in Eqn. (4.2)), in two ways as shown in Fig. 30 and Fig. 31.

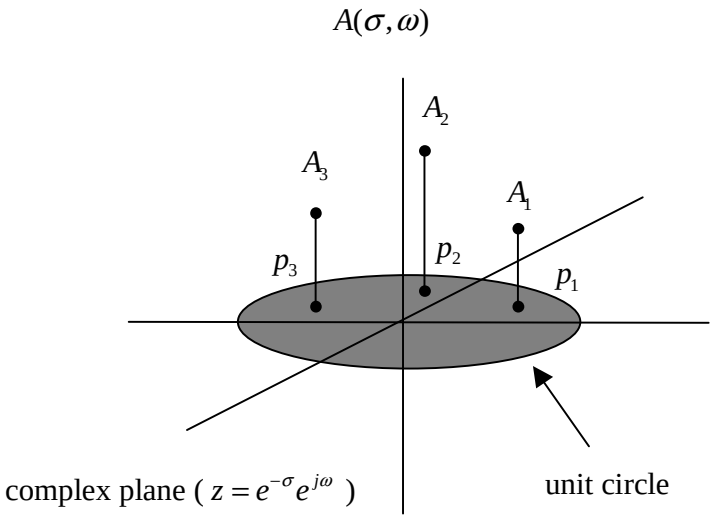

Fig. 30. Polar representation of $A(\sigma,\omega)$.

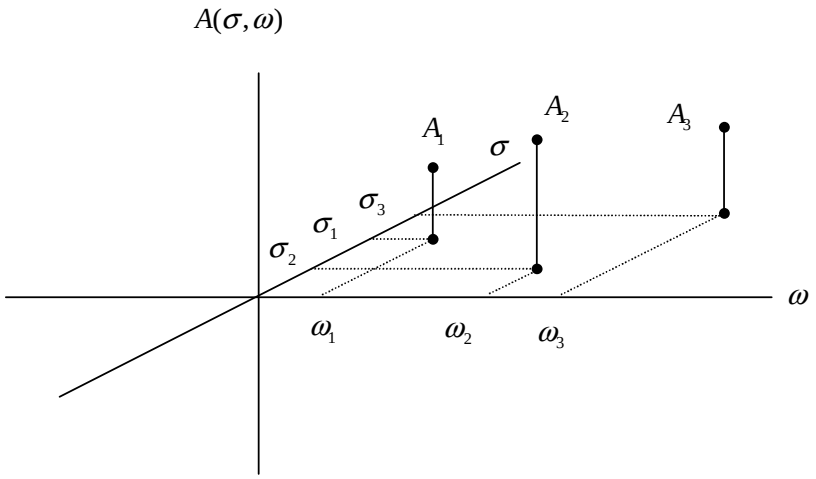

Fig. 31. Rectangular representation of $A(\sigma,\omega)$.

Similar plots could be constructed for $\theta(\sigma,\omega)$. For $S(\sigma,\omega)$ given by Eqn. (4.5), for an arbitrary signal $x[n]$, both $A(\sigma,\omega)$ and $\theta(\sigma,\omega)$ would appear as surfaces.

4.2 Discrete Spectrum of Arbitrary Complex Signals

We now return to the problem of defining $A(\sigma,\omega)$ and $\theta(\sigma,\omega)$ in general. To define $S(\sigma,\omega)$ at an arbitrary point (σ,ω) and for an arbitrary complex signal $x[n]=x(nT)$ we proceed as follows. Assume that $x[n]$ contains a term of the form $Ae^{j\theta}e^{-\sigma n}e^{j\omega n}$. Construct a filter, specific to the point (σ,ω), that passes the signal $Ae^{j\theta}e^{-\sigma n}e^{j\omega n}$ unaltered while significantly attenuating all other such signals. We will employ a finite impulse response (FIR) filter for tractability and implement it in a non-causal manner to avoid start-up transients.

To be specific, assume we have a finite number of samples $x[n] = x(nT),\ n = 0,1,\cdots,N+M-2$. The filter output at the point (σ,ω) is denoted as $x_{\sigma,\omega}[n],\ n = 0,1,\cdots,N-1$, where

$$x_{\sigma,\omega}[n] = \overline{h}^{H}(\sigma,\omega)\overline{x}[n] \tag{4.6}$$

where

$$\overline{h}(\sigma,\omega) = \begin{bmatrix} h_0(\sigma,\omega) \\ h_1(\sigma,\omega) \\ \vdots \\ h_{M-1}(\sigma,\omega) \end{bmatrix} \tag{4.7}$$

is the M -dimensional vector of filter coefficients,

$$\overline{x}[n] = \begin{bmatrix} x[n] \\ x[n+1] \\ \vdots \\ x[n+m+1] \end{bmatrix},\ n = 0,1,\cdots,N-1, \tag{4.8}$$

and H denotes the Hermitian (complex) transpose. This filter will be constructed to meet the conditions:

$$(1) \qquad x_{\sigma,\omega}[n] = x[n] \quad \text{when} \quad x[n] = Ae^{j\theta}e^{-\sigma n}e^{j\omega n} \tag{4.9}$$

and

$$\text{(2)} \qquad \begin{array}{c} x_{\sigma,\omega}[n] \approx 0 \quad \text{when} \quad x[n] = A_0 e^{j\theta_0} e^{-\sigma_0 n} e^{j\omega_0 n}, \\ \sigma_0 \neq \sigma \quad \text{and / or} \quad \omega_0 \neq \omega \end{array} \tag{4.10}$$

We will address the first condition now and return to the second condition later. The first condition requires that

$$\overline{h}^H(\sigma,\omega)\overline{x}[n] = \overline{x}[n], \quad n = 0,1,\cdots,N-1 \tag{4.11}$$

if $x[n] = Ae^{j\theta}e^{-\sigma n}e^{j\omega n}$.

Under these circumstances,

$$\overline{x}[n] = Ae^{j\theta}\begin{bmatrix} e^{(-\sigma+j\omega)n} \\ e^{(-\sigma+j\omega)(n+1)} \\ \vdots \\ e^{(-\sigma+j\omega)(n+M-1)} \end{bmatrix} \tag{4.12}$$

or

$$\overline{x}[n] = \overline{s}(\sigma,\omega)Ae^{j\theta}e^{(-\sigma+j\omega)n} \tag{4.13}$$

where

$$\overline{s}(\sigma,\omega) = \begin{bmatrix} 1 \\ e^{(-\sigma+j\omega)} \\ e^{2(-\sigma+j\omega)} \\ \vdots \\ e^{(M-1)(-\sigma+j\omega)} \end{bmatrix}. \tag{4.14}$$

Thus Eqn. (4.11) becomes

$$\bar{h}^H(\sigma,\omega)\bar{s}(\sigma,\omega)Ae^{j\theta}e^{(-\sigma+j\omega)n} = Ae^{j\theta}e^{(-\sigma+j\omega)n},\ n=0,1,\cdots,N-1. \quad (4.15)$$

Clearly, Eqn. (4.15) will be satisfied if and only if

$$\bar{h}^H(\sigma,\omega)\bar{s}(\sigma,\omega)=1. \quad (4.16)$$

Thus the first condition will be met exactly if $\bar{h}(\sigma,\omega)$ satisfies Eqn. (4.16). There is no easy or unique way to deal with the second condition. There are a number of ways to deal with Eqn. (4.11), each leading to a different spectral estimation method.

4.3 Estimating the Spectrum, $S(\sigma,\omega)$, from Filter Output, $x_{\sigma,\omega}[n]$

At this point we consider the problem of estimating $S(\sigma,\omega)$ assuming $x_{\sigma,\omega}[n]$ is available. We will later consider several methods for determining $\bar{h}(\sigma,\omega)$, each leading to a different version of $x_{\sigma,\omega}[n]$. Specifically, we assume we have $x_{\sigma,\omega}[n],\ n=0,1,\cdots,N-1$. If $\bar{h}(\sigma,\omega)$ has been appropriately designed, then

$$x_{\sigma,\omega}[n] \approx Ae^{j\theta}e^{-\sigma n}e^{j\omega n} \quad (4.17)$$

More precisely,

$$x_{\sigma,\omega}[n] = S(\sigma,\omega)e^{-\sigma n}e^{j\omega n} + \varepsilon[n] \quad (4.18)$$

since $S(\sigma,\omega) = Ae^{j\omega}$, where the error $\varepsilon[n]$ is presumably "small."

It is noted that, if $x[n] = Ae^{j\theta}e^{-\sigma n}e^{j\omega n}$ exactly and if our measurement of $x[n]$ is completely noise-free, then, evaluating Eqn. (4.18) at $n = 0$, we have the simple result

$$S(\sigma,\omega) = x_{\sigma,\omega}[0] \tag{4.19}$$

In practice, however, $\varepsilon[n]$ will contain remnants of other signal components and noise, so Eqn. (4.19) may be in error. We will thus proceed to obtain an estimate of $S(\sigma,\omega)$ from $x_{\sigma,\omega}[n]$ in such a manner as to minimize a weighted sum of the squared-error terms over some range of values of n. Specifically, we will choose $S(\sigma,\omega)$ that minimizes

$$J = \sum_{n=0}^{K-1} \left|\varepsilon[n]\right|^2 e^{-2\alpha n} = \sum_{n=0}^{K-1} \varepsilon[n]\varepsilon^*[n]e^{-2\alpha n} \tag{4.20}$$

where * denotes complex conjugate, $1 \le K \le N$, and where, from Eqn. (4.18),

$$\varepsilon[n] = x_{\sigma,\omega}[n] - S(\sigma,\omega)e^{-\sigma n}e^{j\omega n} \tag{4.21}$$

The quantity α may be zero (equally weighing all the error terms), positive (more heavily weighing "early" terms), or even slightly negative (more heavily weighing "late" terms). Furthermore, α may be a function of σ. It is noted that in the current literature [2], α is always chosen to be zero and K to be N.

Temporarily denoting $S(\sigma,\omega)$ as $S = B + jC$ and $e^{-\sigma}e^{j\omega}$ as p, we have

$$J = \sum_{n=0}^{K-1}\left[x_{\sigma,\omega}[n] - (B + jC)p^n\right]\left[x^*_{\sigma,\omega}[n] - (B - jC)p^{*n}\right]e^{-2\alpha n} \qquad (4.22)$$

Following standard minimization techniques, $S(\sigma,\omega)$ must satisfy

$$\frac{\partial J}{\partial B} = 0 \qquad (4.23)$$

and

$$\frac{\partial J}{\partial C} = 0 \qquad (4.24)$$

This leads to

$$\frac{\partial J}{\partial B} = \sum_{n=0}^{K-1}\left(\left[x_{\sigma,\omega}[n] - Sp^n\right]\left(-p^{*n}\right) + \left[x^*_{\sigma,\omega}[n] - S^*p^{*n}\right]\left(-p^n\right)\right)e^{-2\alpha n} = 0 \qquad (4.25)$$

and

$$\frac{\partial J}{\partial C} = \sum_{n=0}^{K-1}\left(\left[x_{\sigma,\omega}[n] - Sp^n\right]\left(jp^{*n}\right) + \left[x^*_{\sigma,\omega}[n] - S^*p^{*n}\right]\left(-jp^n\right)\right)e^{-2\alpha n} = 0 \qquad (4.26)$$

Subtracting Eqn. (4.25) from Eqn. (4.26) after canceling the j in Eqn. (4.26) results in

$$2\sum_{n=0}^{K-1}\left[x_{\sigma,\omega}[n] - Sp^n\right]p^{*n}e^{-2\alpha n} = 0 \qquad (4.27)$$

Similarly, adding these equations results in

$$2\sum_{n=0}^{K-1}\left[x^{*}_{\sigma,\omega}[n]-S^{*}p^{*n}\right] p^{n}e^{-2\alpha n}=0 \qquad (4.28)$$

which is simply the conjugate of Eqn. (4.27). Thus, after canceling the 2, both lead to

$$\sum_{n=0}^{K-1}\left[x_{\sigma,\omega}[n]-Sp^{n}\right] p^{*n}e^{-2\alpha n}=0 \qquad (4.29)$$

or,

$$S\sum_{n=0}^{K-1} p^{n}p^{*n}e^{-2\alpha n}=\sum_{n=0}^{K-1} x_{\sigma,\omega}[n]p^{*n}e^{-2\alpha n} \qquad (4.30)$$

or,

$$S(\sigma,\omega)\sum_{n=0}^{K-1} e^{-\sigma n}e^{j\omega n}e^{-\sigma n}e^{-2\alpha n}=\sum_{n=0}^{K-1} x_{\sigma,\omega}[n]e^{-\sigma n}e^{-j\omega n}e^{-2\alpha n} \qquad (4.31)$$

Employing Eqn. (4.6), this becomes

$$S(\sigma,\omega)\sum_{n=0}^{K-1} e^{-2(\sigma+\alpha)n}=\overline{h}^{H}(\sigma,\omega)\sum_{n=0}^{K-1}\overline{x}[n]e^{-(\sigma+2\alpha)n}e^{-j\omega n} \qquad (4.32)$$

Thus

$$S(\sigma,\omega) = A(\sigma,\omega)e^{j\theta(\sigma,\omega)}$$

$$= \frac{1}{L_K(\sigma+\alpha)} \bar{h}^H(\sigma,\omega)\bar{X}_K(\sigma+2\alpha,\omega) \qquad (4.33)$$

where

$$L_K(\sigma) = \sum_{n=0}^{K-1} e^{-2\sigma n}$$

$$= \begin{cases} \dfrac{1-e^{-2\sigma K}}{1-e^{-2\sigma}}, & \sigma \neq 0 \\ K, & \sigma = 0 \end{cases} \qquad (4.34)$$

and

$$\bar{X}_K(\sigma,\omega) = \sum_{n=0}^{K-1} \bar{x}[n]e^{-\sigma n}e^{-j\omega n} \qquad (4.35)$$

where the properties of the geometric sum have been used in Eqn. (4.34). It is noted that this reduces to the standard result found in the literature [2] when $\alpha = 0$ and $K = N$, namely

$$S(\sigma,\omega) = \frac{1}{L_N(\sigma)} \bar{h}^H(\sigma,\omega)\bar{X}_N(\sigma,\omega) \qquad (4.36)$$

Further, it is observed that when $K = 1$, Eqn. (4.33) reduces to

$$S(\sigma,\omega) = \bar{h}^H(\sigma,\omega)\bar{x}[0] \tag{4.37}$$

It is also noted that when $\alpha = -\sigma/2$

$$S(\sigma,\omega) = \frac{1}{K}\bar{h}^H(\sigma,\omega)\bar{X}_K(0,\omega) \tag{4.38}$$

where $\bar{X}_K$ is now a function only of ω.

We have not as yet addressed the determination of $\bar{h}(\sigma,\omega)$. Several alternative approaches are possible. These are discussed in the following sections.

4.4 Weighted 2D Capon Method

In this approach we minimize the weighted total energy in the filter output $x_{\sigma,\omega}[n]$ while passing the signal component at frequency, ω, and damping, σ, unaltered. We thus define

$$J_C = \sum_{n-0}^{N-1} \left| x_{\sigma,\omega}[n] \right|^2 e^{-2\beta n} \tag{4.39}$$

In the conventional 2D Capon method $\beta = 0$ [2]. Just as in the case of α in Eqn. (4.20) β may be zero, positive or slightly negative. Furthermore, β may be a function of σ. We have, using Eqn. (4.6),

$$J_C = \sum_{n=0}^{N-1} \left| \bar{h}^H(\sigma,\omega)\bar{x}[n] \right|^2 e^{-2\beta n}$$

$$= \sum_{n=0}^{N-1} (\bar{h}^H(\sigma,\omega)\bar{x}[n])(\bar{h}^T(\sigma,\omega)\bar{x}^*[n])e^{-2\beta n} \qquad (4.40)$$

since $|a|^2 = aa^*$. Using the fact that the second factor in Eqn. (4.40) is scalar and is thus its own transpose,

$$J_C = \sum_{n=0}^{N-1} \bar{h}^H(\sigma,\omega)\bar{x}[n]\bar{x}^H[n]\bar{h}(\sigma,\omega)e^{-2\beta n}$$

$$= \bar{h}^H(\sigma,\omega)\sum_{n=0}^{N-1} \bar{x}[n]\bar{x}^H[n]e^{-2\beta n}\bar{h}(\sigma,\omega)$$

$$J_C = \bar{h}^H(\sigma,\omega)R_\beta \bar{h}(\sigma,\omega) \qquad (4.41)$$

where

$$R_\beta = \sum_{n=0}^{N-1} \bar{x}[n]\bar{x}^H[n]e^{-2\beta n} \qquad (4.42)$$

We thus wish to choose $\bar{h}(\sigma,\omega)$ to minimize

$$J_C = \bar{h}^H(\sigma,\omega)R_\beta \bar{h}(\sigma,\omega) \qquad (4.43)$$

subject to

$$\overline{h}(\sigma,\omega)\overline{s}(\sigma,\omega)=1 \tag{4.44}$$

as required by Eqn. (4.16). This is a standard quadratic minimization problem with solution (see Appendix Sec. A.1),

$$\overline{h}_C(\sigma,\omega)=\frac{R_\beta^{-1}\overline{s}(\sigma,\omega)}{\overline{s}^H(\sigma,\omega)R_\beta^{-1}\overline{s}(\sigma,\omega)} \tag{4.45}$$

With Eqn. (4.33), this leads to the weighted 2D Capon spectrum

$$S_C(\sigma,\omega)=\frac{1}{L_K(\sigma+\alpha)}\frac{\overline{s}(\sigma,\omega)R_\beta^{-1}}{\overline{s}^H(\sigma,\omega)R_\beta^{-1}\overline{s}(\sigma,\omega)}\overline{X}_N(\sigma+2\alpha,\omega) \tag{4.46}$$

since R_β^{-1} is Hermitian. In the standard literature [2], $\beta=0$, $\alpha=0$, and $K=N$, for which Eqn. (4.46) reduces to the conventional 2D Capon spectrum

$$S_C(\sigma,\omega)=\frac{1}{L_N(\sigma)}\frac{\overline{s}(\sigma,\omega)R_0^{-1}}{\overline{s}^H(\sigma,\omega)R_0^{-1}\overline{s}(\sigma,\omega)}\overline{X}_N(\sigma,\omega) \tag{4.47}$$

4.5 2D Capon Method in Frequency Domain

As an alternative to the time-domain minimization criterion of Eqn. (4.39), we here consider a frequency domain approach. From Eqn. (4.33), assuming $K=N$, we have

$$S(\sigma,\omega)=\frac{1}{L_N(\sigma+\alpha)}\overline{h}^H(\sigma,\omega)\overline{X}_N(\sigma+2\alpha,\omega) \tag{4.48}$$

Now, assuming $\bar{h}(\sigma,\omega)$ to be fixed, we can consider the spectral estimate at another frequency, ω_k, given by

$$S_\omega(\sigma,\omega_k)=\frac{1}{L_N(\sigma+\alpha)}\bar{h}(\sigma,\omega)\bar{X}_N(\sigma+2\alpha,\omega_k) \tag{4.49}$$

Letting $\omega_k=\frac{2\pi}{N}k,\ \ k=0,1,\cdots,N-1$, we form

$$J_C'=\frac{L_N^2(\sigma+\alpha)}{N}\sum_{k=0}^{N-1}\left|S_\omega(\sigma,\omega_k)\right|^2 \tag{4.50}$$

and will choose $\bar{h}(\sigma,\omega)$ to minimize J_C' subject to Eqn. (4.16). The positive constant in front of the summation is added for convenience and does not affect the minimization process. Proceeding,

$$J_C'=\frac{L_N^2(\sigma+\alpha)}{N}\sum_{k=0}^{N-1}S_\omega(\sigma,\omega_k)S_\omega^H(\sigma,\omega_k)$$

$$=\frac{1}{N}\sum_{k=0}^{N-1}\bar{h}^H(\sigma,\omega)\bar{X}_N(\sigma+2\alpha,\omega_k)\bar{X}_N^H(\sigma+2\alpha,\omega_k)\bar{h}(\sigma,\omega)$$

$$J_C'=\frac{1}{N}\bar{h}^H(\sigma,\omega)\sum_{k=0}^{N-1}\bar{X}_N(\sigma+2\alpha,\omega_k)\bar{X}_N^H(\sigma+2\alpha,\omega_k)\bar{h}(\sigma,\omega) \tag{4.51}$$

The summation in Eqn. (4.51) may be expanded as follows:

$$\sum_{k=0}^{N-1} \overline{X}_N(\sigma+2\alpha,\omega_k)\overline{X}_N^H(\sigma+2\alpha,\omega_k)$$

$$=\sum_{k=0}^{N-1}\left(\sum_{n=0}^{N-1}\overline{x}[n]e^{-(\sigma+2\alpha)n}e^{-j\frac{2\pi}{N}kn}\cdot\sum_{m=0}^{N-1}\overline{x}[m]e^{-(\sigma+2\alpha)m}e^{j\frac{2\pi}{N}km}\right)$$

$$=\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}\overline{x}[n]\overline{x}^H[m]e^{-(\sigma+2\alpha)(n+m)}\sum_{k=0}^{N-1}e^{-j\frac{2\pi}{N}k(n-m)} \qquad (4.52)$$

It may be shown that [95]

$$\sum_{k=0}^{N-1}e^{-j\frac{2\pi}{N}k(n-m)}=\begin{cases}N, & m=n\\ 0, & m\neq n\end{cases} \qquad (4.53)$$

Hence, from Eqn. (4.52),

$$\sum_{k=0}^{N-1}\overline{X}_N(\sigma+2\alpha,\omega_k)\overline{X}_N^H(\sigma+2\alpha,\omega_k)=N\sum_{n=0}^{N-1}\overline{x}[n]\overline{x}^H[n]e^{-2(\sigma+2\alpha)n} \qquad (4.54)$$

Substituting Eqn. (4.54) into Eqn. (4.51) results in

$$J_C'=\overline{h}^H(\sigma,\omega)\sum_{n=0}^{N-1}\overline{x}[n]\overline{x}^H[n]e^{-2(\sigma+2\alpha)n}\overline{h}(\sigma,\omega)$$

$$=\overline{h}^H(\sigma,\omega)R_{\sigma+2\alpha}\overline{h}(\sigma,\omega) \qquad (4.55)$$

This is exactly Eqn. (4.41) with $\beta=\sigma+2\alpha$. Thus the frequency domain version of the 2D Capon method is simply the weighted 2D Capon method with a particular choice for

β. This development is of value, however, because it suggests the possibility of using $\beta = \sigma + constant$ in the weighted 2D Capon method.

4.6 Weighted 2D APES Method

In this approach we minimize the weighted least squares fitting error between the filter output and the signal component at the desired frequency ω and damping σ, while passing that component unaltered. (APES is an acronym for ***a****mplitude and* ***p****hase* ***es****timation*.) We define

$$J_A = \sum_{n=0}^{N-1} \left| x_{\sigma,\omega}[n] - S(\sigma,\omega) e^{(-\sigma + j\omega)n} \right|^2 e^{-2\beta n} \qquad (4.56)$$

where $S(\sigma,\omega)$ is the spectral estimate given by Eqn. (4.33). Here, however, it is convenient to choose $K = N$ and $\alpha = \beta$ in Eqn. (4.33). The usual comments apply to β. It is noted that when $\beta = 0$, J_A corresponds to the minimization criterion employed in the conventional 2D APES method introduced by Stoica and Sundin[2]. Thus

$$J_A = \sum_{n=0}^{N-1} \left| \bar{h}^H(\sigma,\omega)\bar{x}[n] - \frac{1}{L_N(\sigma+\beta)} \bar{h}^H(\sigma,\omega)\bar{X}_N(\sigma+2\beta,\omega) e^{(-\sigma + j\omega)n} \right|^2 e^{-2\beta n} \qquad (4.57)$$

which is to be minimized over $\bar{h}(\sigma,\omega)$ subject to Eqn. (4.16). We now have

$$J_A = \sum_{n=0}^{N-1} \overline{h}^H(\sigma,\omega)\left(\overline{x}[n] - \frac{1}{L_N(\sigma+\beta)} \overline{X}_N(\sigma+2\beta,\omega) e^{(-\sigma+j\omega)n} \right)$$

$$\cdot \left(\overline{x}^H[n] - \frac{1}{L_N(\sigma+\beta)} \overline{X}_N^H(\sigma+2\beta,\omega) e^{(-\sigma-j\omega)n} \right) e^{-2\beta n} \overline{h}(\sigma,\omega)$$

$$J_A = \overline{h}^H(\sigma,\omega)\left(\sum_{n=0}^{N-1} \overline{x}[n]\overline{x}^H[n] e^{-2\beta n} - \frac{1}{L_N(\sigma+\beta)} \overline{X}_N(\sigma+2\beta,\omega) \sum_{n=0}^{N-1} \overline{x}^H[n] e^{(-\sigma-2\beta+j\omega)n} \right.$$

$$- \frac{1}{L_N(\sigma+\beta)} \sum_{n=0}^{N-1} \overline{x}[n] e^{(-\sigma-2\beta-j\omega)n} \overline{X}_N^H(\sigma+2\beta,\omega)$$

$$\left. + \frac{1}{L_N^2(\sigma+\beta)} \sum_{n=0}^{N-1} e^{(-\sigma+j\omega)n} e^{(-\sigma-j\omega)n} e^{-2\beta n} \overline{X}_N(\sigma+2\beta,\omega) \overline{X}_N^H(\sigma+2\beta,\omega) \right) \overline{h}(\sigma,\omega)$$

$$J_A = \overline{h}^H(\sigma,\omega)\left(R_\beta - \frac{1}{L_N(\sigma+\beta)} \overline{X}_N(\sigma+2\beta,\omega) \overline{X}_N^H(\sigma+2\beta,\omega) \right.$$

$$- \frac{1}{L_N(\sigma+\beta)} \overline{X}_N(\sigma+2\beta,\omega) \overline{X}_N^H(\sigma+2\beta,\omega)$$

$$\left. + \frac{1}{L_N^2(\sigma+\beta)} \sum_{n=0}^{N-1} e^{-2(\sigma+\beta)n} \overline{X}_N(\sigma+2\beta,\omega) \overline{X}_N^H(\sigma+2\beta,\omega) \right) \overline{h}(\sigma,\omega)$$

$$J_A = \overline{h}^H(\sigma,\omega)\left(R_\beta - \frac{1}{L_N(\sigma+\beta)} \overline{X}_N(\sigma+2\beta,\omega) \overline{X}_N^H(\sigma+2\beta,\omega) \right) \overline{h}(\sigma,\omega) \qquad (4.58)$$

since

$$\sum_{n=0}^{N-1} e^{-2(\sigma+\beta)n} = L_N(\sigma+\beta) \tag{4.59}$$

Thus

$$J_A = \bar{h}^H(\sigma,\omega) Q_\beta(\sigma,\omega) \bar{h}(\sigma,\omega) \tag{4.60}$$

where

$$Q_\beta(\sigma,\omega) = R_\beta - \frac{1}{L_N(\sigma+\beta)} \bar{X}_N(\sigma+2\beta,\omega) \bar{X}_N^H(\sigma+2\beta,\omega) \tag{4.61}$$

which is to be minimized subject to Eqn. (4.16). Other than the fact that $Q_\beta(\sigma,\omega)$ is a function of σ and ω (whereas R_β is not), J_A is exactly of the form of J_C in Eqn. (4.41) in the weighted 2D Capon approach. Thus

$$\bar{h}_A(\sigma,\omega) = \frac{Q_\beta^{-1}(\sigma,\omega)\bar{s}(\sigma,\omega)}{\bar{s}^H(\sigma,\omega) Q_\beta^{-1}(\sigma,\omega)\bar{s}(\sigma,\omega)} \tag{4.62}$$

which leads to the weighted 2D APES spectrum

$$S_A(\sigma,\omega) = \frac{1}{L_N(\sigma+\beta)} \frac{\bar{s}^H(\sigma,\omega) Q_\beta^{-1}(\sigma,\omega)}{\bar{s}^H(\sigma,\omega) Q_\beta^{-1}(\sigma,\omega)\bar{s}(\sigma,\omega)} \bar{X}_N(\sigma+2\beta,\omega) \tag{4.63}$$

since $Q_\beta^{-1}(\sigma,\omega)$ is Hermitian. In the standard literature [2], $\beta = 0$, for which Eqn. (4.63) reduces to the conventional 2D APES spectrum

$$S_A(\sigma,\omega) = \frac{1}{L_N(\sigma)} \frac{\bar{s}^H(\sigma,\omega)Q_0^{-1}(\sigma,\omega)}{\bar{s}^H(\sigma,\omega)Q_0^{-1}(\sigma,\omega)\bar{s}(\sigma,\omega)} \bar{X}_N(\sigma,\omega) \qquad (4.64)$$

It is noted that the Matrix Inversion Lemma [60] (see Appendix Sec. A.2) may be used to facilitate the computation of $Q_\beta^{-1}(\sigma,\omega)$, leading to

$$Q_\beta^{-1}(\sigma,\omega) = R_\beta^{-1} + \frac{R_\beta^{-1}\bar{X}_N(\sigma+2\beta,\omega)\bar{X}_N(\sigma+2\beta,\omega)R_\beta^{-1}}{L_N(\sigma+\beta) - \bar{X}_N^H(\sigma+2\beta,\omega)R_\beta^{-1}\bar{X}_N(\sigma+2\beta,\omega)} \qquad (4.65)$$

requiring the inversion only of R_β.

4.7 Combined Weighted 2D APES / 2D Capon Method

The weighted 2D Capon method and the weighted 2D APES method each offer certain advantages. It might therefore be expected that a compromise of these advantages might be obtained by combining these methods. To accomplish this, define

$$J_{AC} = (1-\gamma)J_C + \gamma J_A \qquad (4.66)$$

where $0 \le \gamma \le 1$. For convenience, we choose $K = N$ and $\alpha = \beta$ in the weighted 2D Capon method. Using Eqns. (4.41) and (4.60) we have

$$J_{AC} = (1-\gamma)\bar{h}^H(\sigma,\omega)R_\beta\bar{h}(\sigma,\omega) + \gamma\bar{h}^H(\sigma,\omega)Q_\beta(\sigma,\omega)\bar{h}(\sigma,\omega)$$

$$J_{AC} = \bar{h}^H(\sigma,\omega)\left((1-\gamma)R_\beta + \gamma Q_\beta(\sigma,\omega)\right)\bar{h}(\sigma,\omega) \qquad (4.67)$$

With Eqn. (4.61) this becomes

$$J_{AC} = \bar{h}^H(\sigma,\omega)\left((1-\gamma)R_\beta + \gamma R_\beta - \frac{\gamma}{L_N(\sigma+\beta)}\bar{X}_N(\sigma+2\beta,\omega)\bar{X}_N^H(\sigma+2\beta,\omega)\right)\bar{h}(\sigma,\omega)$$

$$J_{AC} = \bar{h}^H(\sigma,\omega)\left(R_\beta - \frac{\gamma}{L_N(\sigma+\beta)}\bar{X}_N(\sigma+2\beta,\omega)\bar{X}_N^H(\sigma+2\beta,\omega)\right)\bar{h}(\sigma,\omega) \qquad (4.68)$$

or

$$J_{AC} = \bar{h}^H(\sigma,\omega)Q_{\beta,\gamma}(\sigma,\omega)\bar{h}(\sigma,\omega) \qquad (4.69)$$

where

$$Q_{\beta,\gamma}(\sigma,\omega) = R_\beta - \frac{\gamma}{L_N(\sigma+\beta)}\bar{X}_N(\sigma+2\beta,\omega)\bar{X}_N^H(\sigma+2\beta,\omega) \qquad (4.70)$$

As usual, $\bar{h}(\sigma,\omega)$ is chosen to minimize J_{AC} subject to Eqn. (4.16). We thus have

$$\bar{h}_{AC}(\sigma,\omega) = \frac{Q_{\beta,\gamma}^{-1}(\sigma,\omega)\bar{s}(\sigma,\omega)}{\bar{s}^H(\sigma,\omega)Q_{\beta,\gamma}^{-1}(\sigma,\omega)\bar{s}(\sigma,\omega)} \qquad (4.71)$$

and

$$S_{AC}(\sigma,\omega)=\frac{1}{L_N(\sigma+\beta)}\frac{\overline{s}^H(\sigma,\omega)Q_{\beta,\gamma}^{-1}(\sigma,\omega)}{\overline{s}^H(\sigma,\omega)Q_{\beta,\gamma}^{-1}(\sigma,\omega)\overline{s}(\sigma,\omega)}\overline{X}_N(\sigma+2\beta,\omega) \qquad (4.72)$$

It is seen that the combined method is nothing more than a parameterized version of the weighted 2D APES method, which reduces to the normal weighted 2D APES method when $\gamma=1$ and to the weighted 2D Capon method (with $K=N$) when $\gamma=0$. As before, the Matrix Inversion Lemma [60] can be employed to rewrite $Q_{\beta,\gamma}^{-1}(\sigma,\omega)$ as

$$Q_{\beta,\gamma}^{-1}(\sigma,\omega)=R_\beta^{-1}+\frac{\gamma R_\beta^{-1}\overline{X}_N(\sigma+2\beta,\omega)\overline{X}_N^H(\sigma+2\beta,\omega)R_\beta^{-1}}{L_N(\sigma+\beta)-\gamma\overline{X}_N^H(\sigma+2\beta,\omega)R_\beta^{-1}\overline{X}_N(\sigma+2\beta,\omega)} \qquad (4.73)$$

4.8 Summary of Methods

For the sake of clarity, in this section we will briefly summarize the development presented earlier in this chapter. We assume we have a signal of the form

$$x[n]=\sum_{\sigma,\omega}S(\sigma,\omega)s^{-\sigma n}e^{j\omega n}+noise,\quad n=0,1,\cdots,N+M-2 \qquad (4.74)$$

where

$$S(\sigma,\omega)=A(\sigma,\omega)e^{j\theta(\sigma,\omega)} \qquad (4.75)$$

and where we seek to estimate $S(\sigma,\omega)$ based on $x[n]$.

4.8.1 General Framework for Techniques

All of the techniques are based on the following two stage methodology:

1.) At each (σ,ω) of interest, construct a length-M FIR filter $\bar{h}(\sigma,\omega)$ for $x[n]$ that passes the signal component at (σ,ω) and attenuates all others, resulting in the filter output $x_{\sigma,\omega}[n]$.

2.) From $x_{\sigma,\omega}[n]$ estimate $S(\sigma,\omega)$.

4.8.2 Existing Techniques

4.8.2.1 2D Capon

1.) Choose the filter $\bar{h}(\sigma,\omega)$ that minimizes

$$J_C = \sum_{n=0}^{N-1} \left| x_{\sigma,\omega}[n] \right|^2 \qquad (4.76)$$

subject to $\bar{h}^H(\sigma,\omega)\bar{s}(\sigma,\omega) = 1$ (which ensures that the component at (σ,ω) is passed unaltered).

2.) Choose the estimate $S(\sigma,\omega)$ that minimizes

$$J = \sum_{n=0}^{N-1} \left| \varepsilon[n] \right|^2 \qquad (4.77)$$

where $\varepsilon[n] = x_{\sigma,\omega}[n] - S(\sigma,\omega)e^{-\sigma n}e^{j\omega n}$

This results in

$$S_C(\sigma,\omega) = \frac{1}{L_N(\sigma)} \frac{\bar{s}^H(\sigma,\omega)R_0^{-1}}{\bar{s}^H(\sigma,\omega)R_0^{-1}\bar{s}(\sigma,\omega)} \bar{X}_N(\sigma,\omega) \tag{4.78}$$

4.8.2.2 2D APES

1.) Choose the filter $\bar{h}(\sigma,\omega)$ that minimizes

$$J_A = \sum_{n=0}^{N-1} \left|\varepsilon[n]\right|^2 \tag{4.79}$$

subject to $\bar{h}^H(\sigma,\omega)\bar{s}(\sigma,\omega) = 1$.

2.) Choose the estimate $S(\sigma,\omega)$ that minimizes

$$J = \sum_{n=0}^{N-1} \left|\varepsilon[n]\right|^2 \tag{4.80}$$

This results in

$$S_A(\sigma,\omega) = \frac{1}{L_N(\sigma)} \frac{\bar{s}^H(\sigma,\omega)Q_0^{-1}(\sigma,\omega)}{\bar{s}^H(\sigma,\omega)Q_0^{-1}(\sigma,\omega)\bar{s}(\sigma,\omega)} \bar{X}_N(\sigma,\omega) \tag{4.81}$$

4.8.3 New Techniques

In these proposed techniques certain choices of the parameters $K, \alpha, \beta,$ and γ can lead to improved estimation properties and/or reduced computation time.

4.8.3.1 Weighted 2D Capon

1.) Choose the filter $\overline{h}(\sigma,\omega)$ that minimizes

$$J_C(\beta,K)=\sum_{n=0}^{K-1}\left|x_{\sigma,\omega}[n]\right|^2 e^{-2\beta n} \tag{4.82}$$

subject to $\overline{h}^H(\sigma,\omega)\overline{s}(\sigma,\omega)=1$.

2.) Choose the estimate $S(\sigma,\omega)$ that minimizes

$$J(\alpha)=\sum_{n=0}^{N-1}\left|\varepsilon[n]\right|^2 e^{-2\alpha n} \tag{4.83}$$

This results in

$$S_C(\sigma,\omega)=\frac{1}{L_N(\sigma+\alpha)}\,\frac{\overline{s}^H(\sigma,\omega)R_\beta^{-1}}{\overline{s}^H(\sigma,\omega)R_\beta^{-1}\overline{s}(\sigma,\omega)}\,\overline{X}_K(\sigma+2\alpha,\omega) \tag{4.84}$$

4.8.3.2 Weighted 2D APES

1.) Choose the filter $\overline{h}(\sigma,\omega)$ that minimizes

$$J_A(\beta)=\sum_{n=0}^{N-1}\left|\varepsilon[n]\right|^2 e^{-2\beta n} \tag{4.85}$$

subject to $\overline{h}^H(\sigma,\omega)\overline{s}(\sigma,\omega)=1$.

2.) Choose the estimate $S(\sigma,\omega)$ that minimizes

$$J(\beta) = \sum_{n=0}^{N-1} \left| \varepsilon[n] \right|^2 e^{-2\beta n} \qquad (4.86)$$

This results in

$$S_A(\sigma,\omega) = \frac{1}{L_N(\sigma+\beta)} \frac{\overline{s}^H(\sigma,\omega) Q_\beta^{-1}(\sigma,\omega)}{\overline{s}^H(\sigma,\omega) Q_\beta^{-1}(\sigma,\omega) \overline{s}(\sigma,\omega)} \overline{X}_N(\sigma+2\beta,\omega) \qquad (4.87)$$

4.8.3.3 Combined Weighted 2D APES / 2D Capon

1.) Choose the filter $\overline{h}(\sigma,\omega)$ that minimizes

$$J_{AC}(\beta,\gamma) = (1-\gamma) J_C(\beta,N) + \gamma J_A(\beta) \qquad (4.88)$$

subject to $\overline{h}^H(\sigma,\omega)\overline{s}(\sigma,\omega) = 1$.

2.) Choose the estimate $S(\sigma,\omega)$ that minimizes

$$J(\beta) = \sum_{n=0}^{N-1} \left| \varepsilon[n] \right|^2 e^{-2\beta n} \qquad (4.89)$$

This results in

$$S_{AC}(\sigma,\omega) = \frac{1}{L_N(\sigma+\beta)} \frac{\overline{s}^H(\sigma,\omega) Q_{\beta,\gamma}^{-1}(\sigma,\omega)}{\overline{s}^H(\sigma,\omega) Q_{\beta,\gamma}^{-1}(\sigma,\omega) \overline{s}(\sigma,\omega)} \overline{X}_N(\sigma+2\beta,\omega) \qquad (4.90)$$

which reduces to the weighted 2D APES result when $\gamma = 1$ and the weighted 2D Capon result (with $K = N$) when $\gamma = 0$.

4.9 Computational Considerations

We may evaluate the weighted 2D Capon spectrum, $S_C(\sigma,\omega)$, and the combined weighted 2D APES / 2D Capon spectrum, $S_{AC}(\sigma,\omega)$, for specific values of σ and ω. (It is not necessary to separately discuss $S_A(\sigma,\omega)$, since it is a special case of $S_{AC}(\sigma,\omega)$.) It is generally of interest, however, to evaluate these quantities over an equally spaced rectangular grid of σ and ω values. Certain computational efficiencies may be exploited if this is done. Assume

$$\sigma_m = m\Delta\sigma, \quad m = 0,1,\cdots,N_\sigma - 1, \quad \Delta\sigma > 0 \tag{4.91}$$

and

$$\omega_k = k\Delta\omega, \quad k = 0,1,\cdots,N_\omega - 1, \quad \Delta\omega > 0 \tag{4.92}$$

That is, we are interested in N_σ values of σ and N_ω values of ω. To ensure that $0 \le \omega < \pi$, we choose

$$\Delta\omega = \frac{\pi}{N_\omega} \tag{4.93}$$

We now consider the evaluation of $\bar{X}_K(\sigma + 2\alpha,\omega)$ and $\bar{X}_N(\sigma + 2\beta,\omega)$, as required by the methods described above. It suffices to consider only $\bar{X}_K(\sigma + 2\alpha,\omega)$ where $0 \le K \le N$ and where α may be a function of σ. Thus

$$\bar{X}_K(\sigma+2\alpha,\omega)=\sum_{n=0}^{K-1}\bar{x}[n]e^{-(\sigma+2\alpha)n}e^{-j\omega n} \tag{4.94}$$

At the grid points this becomes

$$\bar{X}_K(\sigma_m+2\alpha,\omega_k)=\sum_{n=0}^{K-1}\bar{x}[n]e^{-(m\Delta\sigma+2\alpha)n}e^{-j\frac{2\pi}{2N_\omega}kn} \tag{4.95}$$

or

$$\bar{X}_K(\sigma_m+2\alpha,\omega_k)=\sum_{n=0}^{2N_\omega-1}\bar{x}_e[n]e^{-(m\Delta\sigma+2\alpha)n}e^{-j\frac{2\pi}{2N_\omega}kn} \tag{4.96}$$

where

$$\bar{x}_e[n]=\begin{cases}\bar{x}[n], & n=0,1,\cdots,K-1\\ 0, & n=K,K+1,\cdots,2N_\omega-1\end{cases} \tag{4.97}$$

and where we have assumed $2N_\omega \geq K$, or $N_\omega \geq K/2$.

From Eqn. (4.96), it is seen that $\bar{X}_K(\sigma_m+2\alpha,\omega_k)$ is the length-$2N_\omega$ vector discrete Fourier transform of the quantity $\bar{x}_e[n]e^{-(m\Delta\sigma+2\alpha)n}$, which may be efficiently evaluated using the fast Fourier transform (FFT). Thus, for a fixed m, $\bar{X}_K(\sigma_m+2\alpha,\omega_k)$ may be determined for all k, $k=0,1,\cdots,N_\omega-1$, using one length-$2N_\omega$ vector FFT (in this case one vector FFT consists of M ordinary FFT's). It is further observed that if α

is chosen as $\alpha = -m\Delta\sigma/2$, $\bar{X}_K(\sigma_m + 2\alpha, \omega_k)$ becomes the FFT of $\bar{x}_e[n]$, independent of m. Thus, one vector FFT suffices for all m, $m = 0,1,\cdots,N_\sigma - 1$.

As another issue, the evaluation of $S_C(\sigma,\omega)$ and $S_{AC}(\sigma,\omega)$ require the inversion of R_β and $Q_{\beta,\gamma}(\sigma,\omega)$, respectively. If β is fixed, then R_β needs to be inverted only once over the entire grid and, if the Matrix Inversion Lemma[60] is used, the inversion of $Q_{\beta,\gamma}(\sigma,\omega)$ likewise requires only one inversion of R_β. On the other hand, if β is a function of σ, then, in both cases, R_β must be inverted once for each m, $m = 0,1,\cdots,N_\sigma - 1$.

Table 1 lists the number of length-$2N_\omega$ FFT's and $M \times M$ matrix inversions for various situations of interest.

Method	α	β	FFT's	Matrix Inversions
Weighted 2D Capon ($K = N$)	not $-\sigma/2$	constant	MN_σ	1
Weighted 2D Capon ($K = N$)	not $-\sigma/2$	function(σ)	MN_σ	N_σ
Weighted 2D Capon ($K = N$)	$-\sigma/2$	constant	M	1
Weighted 2D Capon ($K = N$)	$-\sigma/2$	function(σ)	M	N_σ
Weighted 2D Capon ($K = 1$)	-	constant	0	1
Weighted 2D Capon ($K = 1$)	-	function(σ)	0	N_σ
Combined Weighted 2D APES / 2D Capon	-	constant	MN_σ	1
Combined Weighted 2D APES / 2D Capon	-	function(σ), not $-\sigma/2$	MN_σ	N_σ
Combined Weighted 2D APES / 2D Capon	-	$-\sigma/2$	M	N_σ

Table 1. Computational Requirements.

4.10 Example

The following simple example will hopefully provide the reader with a better understanding of the concepts presented in this chapter. We consider the signal

$$\begin{aligned} x[n] = {} & 2e^{j(\pi/3)}e^{-0.003n}e^{j(20\pi/256)n} + \\ & 4e^{-j(\pi/6)}e^{-0.008n}e^{j(22\pi/256)n} + \\ & 2e^{j(\pi)}e^{-0.001n}e^{j(50\pi/256)n} + \\ & 4e^{j(\pi/6)}e^{-0.008n}e^{j(210\pi/256)n} + \\ & 2e^{-j(\pi/3)}e^{-0.003n}e^{j(220\pi/256)n} \end{aligned} \tag{4.98}$$

with added Gaussian white noise (refer to Eqn. (4.74)). This signal has the spectral peaks shown in Table 2.

ω	σ	A	θ
$20\pi/256$	-0.003	2	$\pi/3$
$22\pi/256$	-0.008	4	$-\pi/6$
$50\pi/256$	-0.001	2	π
$210\pi/256$	-0.008	4	$\pi/6$
$220\pi/256$	-0.003	2	$-\pi/3$

Table 2. Parameters for simulated signal.

In the first instance we assume an SNR of 48 dB. Using the conventional 2D Capon method with $N = 512$ and $M = 128$, we obtain the estimated spectrum $S(\sigma,\omega) = A(\sigma,\omega)e^{j\theta(\sigma,\omega)}$. Fig. 32 is a 3-dimensional plot of $A(\sigma,\omega)$ vs. σ and ω. Fig. 33 is a contour plot of $A(\sigma,\omega)$. In both cases the five spectral peaks are evident.

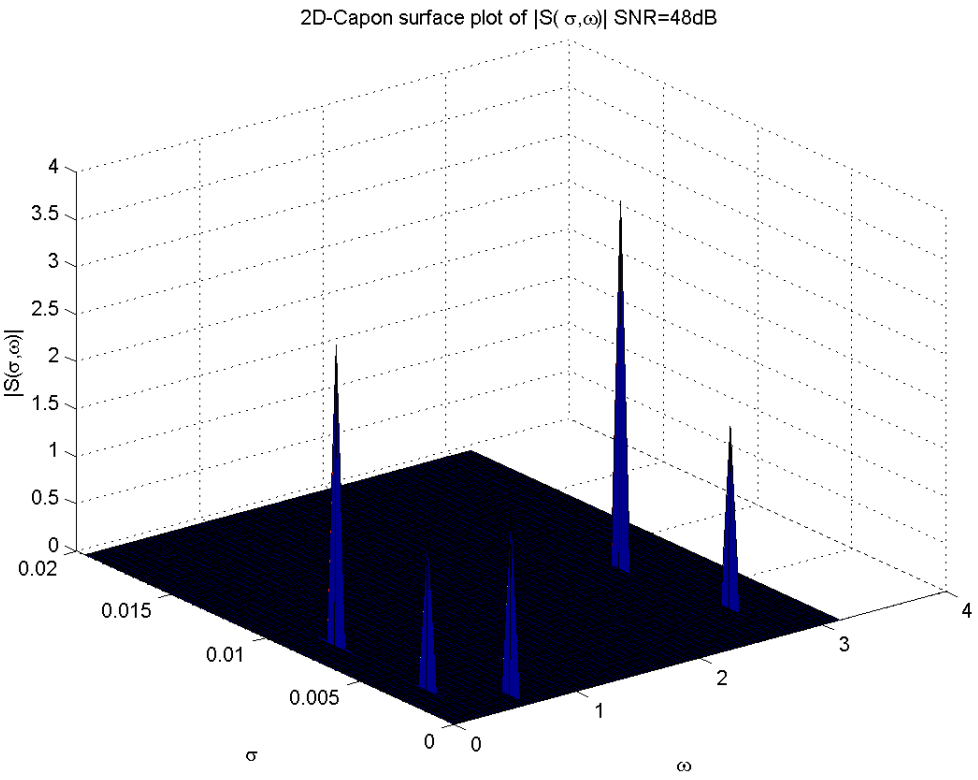

Fig. 32. 2D Capon surface of simulated signal with SNR=48dB.

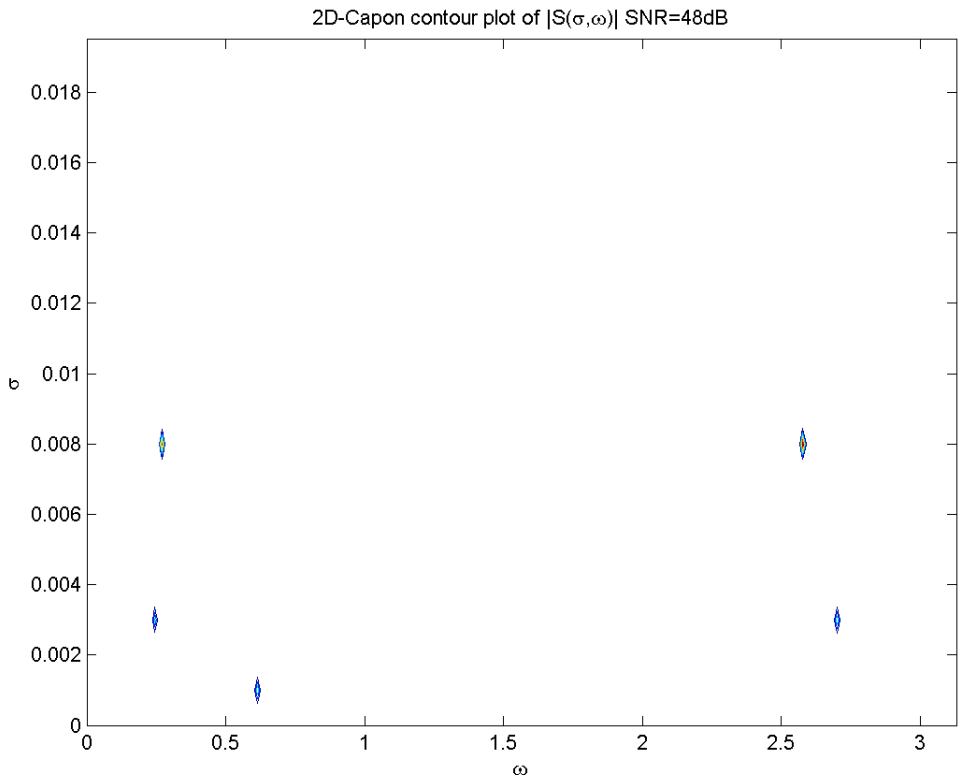

Fig. 33. 2D Capon contour of simulated signal with SNR=48dB.

Fig. 34 and Fig. 35 are similar to Fig. 32 and Fig. 33, respectively, but with an SNR of 12 dB. The effect of greater noise is seen as a broadening of the peaks in the σ direction.

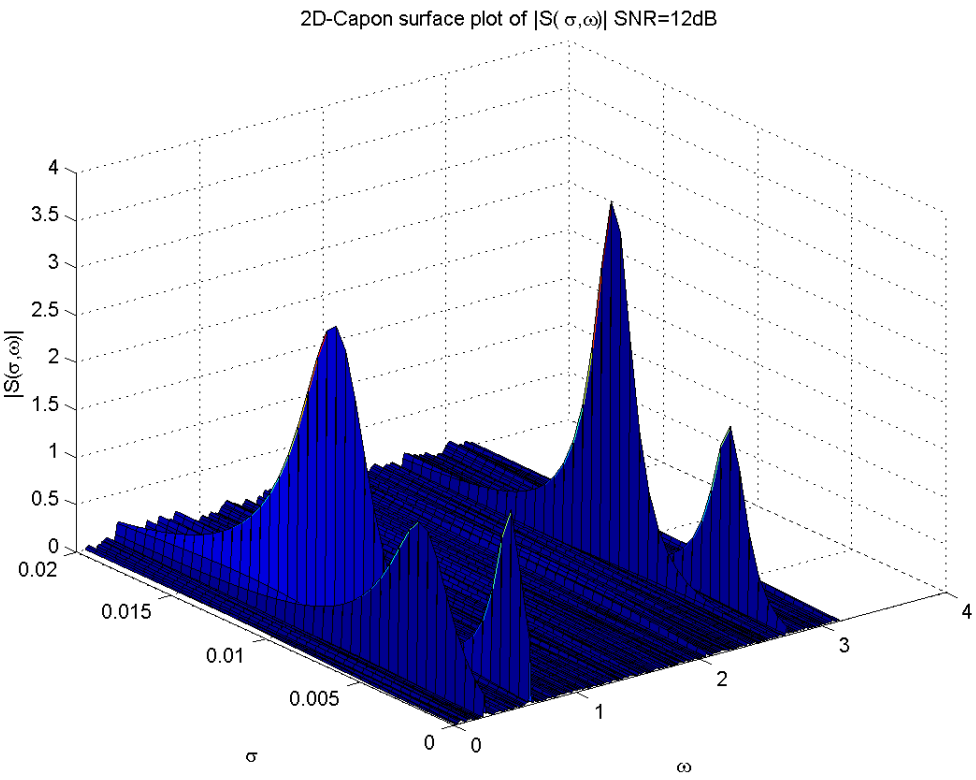

Fig. 34. 2D Capon surface of simulated signal with SNR=12dB.

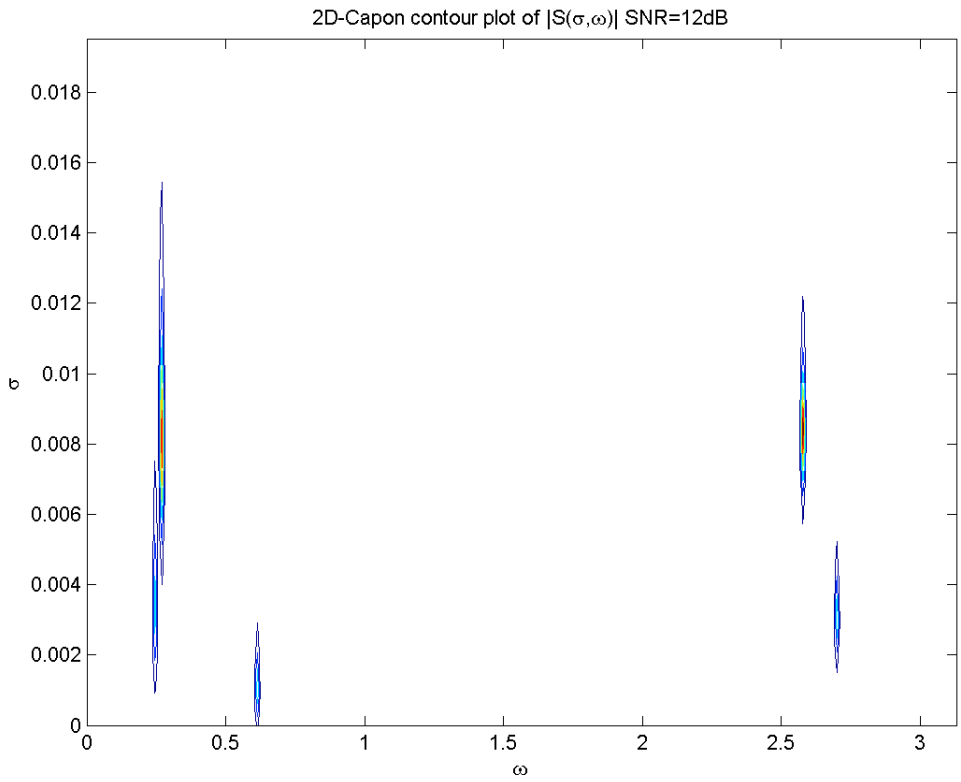

Fig. 35. 2D Capon contour of simulated signal with SNR=12dB.

4.11 Summary

In this chapter we have introduced several new two-dimensional spectral estimation techniques: the weighted 2D Capon method, the weighted 2D APES method and the combined weighted 2D APES / 2D Capon method. It was shown that the conventional 2D Capon method and the conventional 2D APES method are special cases of the new methods introduced in this chapter.

The primary motivation for the introduction of these new techniques is to improve spectral estimation capabilities, peak detection reliability, accuracy, and in some cases to increase the overall performance of the spectral estimation processing time. The two-dimensional techniques introduced in this chapter are capable of estimating both frequency and damping characteristics of a signal and are well-suited to use for MRS analysis.

Chapter 5

Multiple-Channel Weighted 2D Spectral Estimation Methods

Many advancements have been made in the area of MR imaging with multiple receive coils. The most common technique for creating a single MR image from multiple-receive coils was introduced by Roemer, et al.[29]. Since multiple receive coils are widely used in a clinical setting for imaging, it is also desirable to combine MRS spectra from multiple receive coils. Several techniques for combining MRS absorption spectra from multiple receive channels have been proposed [39]-[42]. One technique for combining MRS absorption spectra from multiple receive channels which has gained clinical acceptance was developed by Frigo, Heinen, et al. [3].

In analyzing MRS results from multiple receive coils, the data from each coil may be processed separately to create a result for each coil. Signal characteristics from each coil depend on a number of factors including the orientation of the coil with respect to the B_0 field, the proximity of the coil to the volume generating the signal, coil loading, coil-to-coil coupling effects, and the permeability and permitivity of the medium through which the radio-frequency signal travels [75] prior to being received by the coil elements. Regions of interest close to receive coil elements benefit from improved SNRs[30],[37].

In this chapter we will extend the weighted 2D spectral estimation methods developed in Chapter 4 to the case of multiple-channel data. We will consider both weighted signal averaging and weighted spectrum averaging. Although motivated by the application to MRS signals, the development, as in Chapter 4, is cast in a general setting.

5.1 Extensions to Multiple Observations

In this case we seek to estimate $\mathcal{S}\{x[n]\} = S(\sigma, \omega)$ based on multiple observations of the signal $x[n]$ (multi-channel case). Here we assume we have C noisy scaled observations $x_i[n],\ i = 1, 2, \cdots, C,$ of the complex signal $x[n]$, for $n = 0, 1, \cdots, N + M - 2$. More precisely,

$$x_i[n] = g_i x[n] + \upsilon_i[n],\quad i = 1, 2, \cdots, C \qquad (5.1)$$

where the $\upsilon_i[n]$ are noise terms and the g_i's are complex constants (channel gains). The noise terms $\upsilon_i[n]$ are assumed to be zero mean with variances $\mathcal{E}\left\{\left|\upsilon_i\right|^2\right\} = \rho_i^2 > 0,\ i = 1, 2, \cdots, C$. We further assume that the $\upsilon_i[n]$'s are mutually uncorrelated and uncorrelated with $x[n]$. We will initially assume that the g_i's and ρ_i's are known, and later consider the possibility of estimating these quantities from the data if they are unknown.

5.2 Estimating the Basic Signal, $x[n]$, from Observations, $x_i[n]$'s

For the first technique to be considered later we require an estimate, $\hat{x}[n]$, of $x[n]$. We will choose to seek an estimate of the following form (weighted average):

$$\hat{x}[n] = \sum_{i=1}^{C} w_i^* x_i[n], \quad n = 0,1,\cdots, N+M-2 \tag{5.2}$$

where the w_i's are complex constants. We may also write

$$\hat{x}[n] = \tilde{w}^H \tilde{x}[n] \tag{5.3}$$

where

$$\tilde{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_C \end{bmatrix} \tag{5.4}$$

and

$$\tilde{x}[n] = \begin{bmatrix} x_1[n] \\ x_2[n] \\ \vdots \\ x_C[n] \end{bmatrix} \tag{5.5}$$

We may also write

$$\tilde{x}[n] = \tilde{g}x[n] + \tilde{v}[n] \tag{5.6}$$

where

$$\tilde{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_C \end{bmatrix} \tag{5.7}$$

and

$$\tilde{v}[n] = \begin{bmatrix} v_1[n] \\ v_2[n] \\ \vdots \\ v_C[n] \end{bmatrix} \tag{5.8}$$

Thus

$$\hat{x}[n] = \tilde{w}^H \tilde{g}x[n] + \tilde{w}^H \tilde{v}[n] \tag{5.9}$$

In order to ensure that $\hat{x}[n]$ will be equal to $x[n]$ in the absence of noise, and is thus properly scaled, we require that

$$\tilde{w}^H \tilde{g} = 1 \tag{5.10}$$

Then Eqn. (5.8) becomes

$$\hat{x}[n] = x[n] + \tilde{w}^H \tilde{v}[n] \tag{5.11}$$

We will now determine the optimal $\tilde{w}$, that is, the one that minimizes the energy in $\tilde{w}^H \tilde{v}[n]$. We thus choose

$$J_w = \sum_{n=0}^{N+M-2} \left| \tilde{w}^H \tilde{v}[n] \right|^2$$

$$= \sum_{n=0}^{N+M-2} \tilde{w}^H \tilde{v}[n] \tilde{v}^H[n] \tilde{w}$$

$$J_w = \tilde{w}^H \left(\sum_{n=0}^{N+M-2} \tilde{v}[n] \tilde{v}^H[n] \right) \tilde{w} \qquad (5.12)$$

Since the $v_i[n]$'s are mutually uncorrelated, the off-diagonal terms of $\tilde{v}[n]\tilde{v}^H[n]$ will sum to zero (approximately) and the diagonal terms will sum (approximately) to ρ_i^2, respectively, provided that $N+M$ is reasonably large. Thus, J_w becomes, at least approximately

$$J_w = \tilde{w}^H R_w \tilde{w} \qquad (5.13)$$

where

$$R_w = \begin{bmatrix} \rho_1^2 & 0 & \cdots & 0 \\ 0 & \rho_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \rho_C^2 \end{bmatrix} \qquad (5.14)$$

J_w in Eqn. (5.13) is thus to be minimized subject to Eqn. (5.10). This has solution (see Appendix Sec. A.1)

$$\tilde{w} = \frac{R_w^{-1}\tilde{g}}{\tilde{g}^H R_w^{-1}\tilde{g}} \tag{5.15}$$

thus providing the optimum weight vector. In the case where $\rho_i^2 = \rho^2, \quad i = 1, 2, \cdots, C$, that is, in the case of equal noise variances, Eqn. (5.15) simplifies to

$$\tilde{w} = \frac{\tilde{g}}{\tilde{g}^H \tilde{g}} \tag{5.16}$$

or

$$w_i = \frac{g_i}{\sum_{j=1}^{C} |g_j|^2}, \quad i = 1, 2, \cdots, C \tag{5.17}$$

5.3 Estimating the 2D Spectrum, $S(\sigma, \omega)$

We now consider two approaches to estimating $S(\sigma, \omega)$ in the case of multiple observations.

5.3.1 Weighted Signal Averaging

Here we use Eqn. (5.3) with the weights given by Eqn. (5.15) to obtain the estimate

$$\hat{x}[n] = \frac{\tilde{g}^H R_w^{-1} \tilde{x}[n]}{\tilde{g}^H R_w^{-1} \tilde{g}} \qquad (5.18)$$

of $x[n]$. Then we simply replace $x[n]$ by $\hat{x}[n]$ in any of the methods in Chapter 4 for determining $S(\sigma, \omega)$.

5.3.2 Weighted Spectrum Averaging

In this case, we first apply any of the methods of Chapter 4 to obtain the spectrum $S_i(\sigma, \omega)$ of $x_i[n], \quad i = 1, 2, \cdots, C$. We then form the weighted average spectrum

$$\hat{S}(\sigma, \omega) = \sum_{i=1}^{C} w_i^* S_i(\sigma, \omega) \qquad (5.19)$$

at each value of σ and ω under consideration. If we were to use the w_i's from Eqn. (5.15) and <u>if</u> the spectrum operator $\mathcal{S}\{\cdot\}$ were linear, then Eqn. (5.19) would produce the same spectrum as that obtained by the weighted signal averaging method. However, $\mathcal{S}\{\cdot\}$ is <u>not</u> a linear operator. (Although $\mathcal{S}\{cx[n]\} = c\mathcal{S}\{x[n]\}$, $\mathcal{S}\{x_1[n] + x_2[n]\} \neq \mathcal{S}\{x_1[n]\} + \mathcal{S}\{x_2[n]\}$.) It is therefore not clear what values to choose for the w_i's in Eqn. (5.19) but those of Eqn. (5.15) would seem to be reasonable. It is noted that this method requires C spectrum determinations, and is thus considerably more (C times more) computationally intensive than the weighted signal averaging method, which requires only one spectrum determination.

5.4 Estimating Channel Gains, g_i's , from Observed Data

Thus far we have assumed that the g_i's (required in each of the methods for determining the spectrum) are known. In situations where this is not the case we may use the observed data to obtain estimates of the g_i's to use in determining the spectrum. Unfortunately, it is clearly impossible to determine the g_i's in an absolute sense, since we do not know $x[n]$. However, we may estimate the g_i's in a relative sense, i.e., within a multiplicative constant.

From Eqn. (5.1), we have

$$\frac{1}{g_i} x_i[n] = x[n] + \frac{1}{g_i} \upsilon_i[n], \quad i = 0,1,\cdots,C, \quad n = 0,1,\cdots,N+M-2 \qquad (5.20)$$

Thus $\frac{1}{g_i} x_i[n],\ i = 1,2,\cdots,C$ may all be viewed as estimates of $x[n]$. In the absence of noise, we would thus have

$$\frac{1}{g_j} x_j[n] = \frac{1}{g_i} x_i[n] \qquad (5.21)$$

or

$$g_i x_j[n] = g_j x_i[n] \qquad (5.22)$$

for all combinations of i and j. We will thus attempt to find a set of g_i's, $i = 1, 2, \cdots, C$, so as to minimize

$$J^g = \sum_{n=0}^{N+M-2} \sum_{j=1}^{C} \sum_{\substack{i=1 \\ i \neq j}}^{C} \left| g_i x_j[n] - g_j x_i[n] \right|^2 \tag{5.23}$$

Obviously $g_i = 0,\ i = 1, 2, \cdots, C$, minimizes J^g, but this is clearly unsuitable for our purposes. We will therefore, somewhat arbitrarily, impose the additional restriction that

$$\tilde{g}^H \underline{1} = 1 \tag{5.24}$$

where $\tilde{g}$ is given by Eqn. (5.7) and where

$$\underline{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{5.25}$$

The g_i's thus obtained will be estimates (arbitrarily summing to one) of the true g_i's, to within a multiplicative constant. To carry out the minimization process we must find $\frac{\partial J^g}{\partial g_k}$ for each k, $k = 1, 2, \cdots, C$. We can simplify this process for a specific k by looking at only the terms of J^g that contain g_k. These terms are given by

$$J_k^g = \sum_{n=0}^{N+M-2}\left[\sum_{\substack{i=1\\ i\neq k}}^{C}\left|g_i x_k[n]-g_k x_i[n]\right|^2+\sum_{\substack{j=1\\ j\neq k}}^{C}\left|g_k x_j[n]-g_j x_k[n]\right|^2\right]$$

$$=\sum_{n=0}^{N+M-2}\left[\sum_{\substack{i=1\\ i\neq k}}^{C}\left|g_i x_k[n]-g_k x_i[n]\right|^2+\sum_{\substack{i=1\\ i\neq k}}^{C}\left|g_k x_i[n]-g_i x_k[n]\right|^2\right]$$

$$J_k^g = 2\sum_{n=0}^{N+M-2}\left[\sum_{\substack{i=1\\ i\neq k}}^{C}\left|g_i x_k[n]-g_k x_i[n]\right|^2\right] \tag{5.26}$$

This is of the form of Eqn. (4.22), so by an analogous procedure to that which produced Eqn. (4.29), we have

$$\sum_{n=0}^{N+M-2}\left(\sum_{\substack{i=1\\ i\neq k}}^{C} g_i x_k[n]-g_k\sum_{\substack{i=1\\ i\neq k}}^{C} x_i[n]\right)x_i^*[n]=0 \tag{5.27}$$

or

$$\sum_{\substack{i=1\\ i\neq k}}^{C} g_i\left(\sum_{n=0}^{N+M-2} x_k[n]x_i^*[n]\right)-g_k\left(\sum_{\substack{i=1\\ i\neq k}}^{C} x_i[n]x_i^*[n]\right)=0 \tag{5.28}$$

or

$$\sum_{\substack{i=1\\ i\neq k}}^{C} g_i r_{ki}-g_k\sum_{\substack{i=1\\ i\neq k}}^{C} r_{ii}=0 \tag{5.29}$$

where

$$r_{ij} = \sum_{n=0}^{N+M-2} x_i[n] x_j^*[n] \tag{5.30}$$

Defining

$$r_g = \sum_{i=1}^{C} r_{ii} \tag{5.31}$$

Eqn. (5.29) may be rewritten as

$$\sum_{\substack{i=1 \\ i \neq k}}^{C} g_i r_{ki} + g_k (r_{kk} - r_g) = 0 \tag{5.32}$$

Combining Eqn. (5.32) for all $k = 1, 2, \cdots, C$, we have

$$R_g \tilde{g} = 0 \tag{5.33}$$

where

$$R_g = \begin{bmatrix} r_{11} - r_g & r_{12} & \cdots & r_{1C} \\ r_{21} & r_{22} - r_g & \cdots & r_{2C} \\ \vdots & \vdots & & \vdots \\ r_{C1} & r_{C2} & \cdots & r_{CC} - r_g \end{bmatrix} \tag{5.34}$$

We thus want to choose $\tilde{g}$ so as to make $R_g\tilde{g}$ as close as possible, in some sense, to zero, subject to $\tilde{g}^H \underline{1} = 1$. We will choose therefore to minimize

$$J_g = \tilde{g}^H R_g^H R_g \tilde{g} \tag{5.35}$$

subject to

$$\tilde{g}^H \underline{1} = 1 \tag{5.36}$$

This has solution (see Appendix Sec. A.1)

$$\hat{\tilde{g}} = \frac{\left(R_g^H R_g\right)^{-1} \underline{1}}{\underline{1}^H \left(R_g^H R_g\right)^{-1} \underline{1}} \tag{5.37}$$

which provides our estimate of $\hat{\tilde{g}}$ of $\tilde{g}$, to within a multiplicative constant.

5.5 Estimating Noise Variances, ρ_i's, from Observed Data

Just as in the case of g_i's, we have thus far assumed that the ρ_i's are known. If the ρ_i's are not known, we may be able to obtain estimates of them, provided that we have sufficient data and provided that no signal components have zero damping. To accomplish this, we assume that we have observations $x_i[n]$, $i = 1, 2, \cdots, C$, for $n = 0, 1, \cdots, N_f - 1$ where $N_f \geq N + M - 1$. We further assume that for n in the range $n = N_s, N_s + 1, \cdots, N_f - 1$, all signal components are sufficiently small so as to be

negligible. Then in this range we will have $x_i[n] \approx \upsilon_i[n]$ and we may thus form the estimates

$$\hat{\rho}_i^2 = \frac{1}{N_f - N_s} \sum_{n=N_s}^{N_f-1} \left| x_i[n] \right|^2$$

$$\hat{\rho}_i^2 = \frac{1}{N_f - N_s} \sum_{n=N_s}^{N_f-1} x_i[n] x_i^*[n], \quad i = 1, 2, \cdots, C \qquad (5.38)$$

5.6 Summary

In this chapter we have introduced two methods for performing two-dimensional spectral estimation from multiple-channel data: weighted signal averaging and weighted spectrum averaging. Both techniques offer simple and effective means of creating a single combined spectral estimate from multiple-channel data. We also introduced a method to optimally estimate the relative channel gains from observed data. From a clinical standpoint with respect to MRS, these methods greatly simplify interpretation of results from multiple-channel data.

Chapter 6

Evaluation of 2D Spectral Estimation Methods

6.1 Introduction

In this chapter we evaluate the proposed new algorithms introduced in Chapters 4 and 5 through an extensive series of simulations in MATLAB® on known data sets. By performing these simulations we can measure and quantify the performance of the new two-dimensional spectral estimation techniques under a controlled environment. The types of input signals used [76] and the corresponding SNRs will parallel those typically found in MRS data; however the signals used[77] and the techniques used to evaluate the results are general purpose in nature and not unique to MRS applications.

6.2 Definitions

Several definitions are relevant to interpreting the results of the simulations done in this chapter. Two-dimensional spectral estimation techniques are used to estimate both frequency and damping. As such, it serves as a convenience to consider a three-dimensional coordinate system such as the one shown in Fig. 31 where one axis is defined as the frequency axis, ω, another axis represents damping, σ, and the third axis represents the magnitude of the spectral estimate, $|S(\sigma,\omega)|$.

For a simple input signal, $x[n]$, consisting of a single damped sinusoid, one would expect the two-dimensional surface representing the magnitude of the spectral

estimate, $|S(\sigma,\omega)|$, to contain a single "peak" corresponding to the frequency (ω) and damping factor (σ) of the input signal.

6.2.1 Peak Spectrum

When performing two-dimensional spectral estimates in the presence of noise, the surface representing the magnitude of the spectral estimate, $|S(\sigma,\omega)|$, in the (σ,ω) plane may contain widened peaks making it difficult to distinguish with great accuracy where the peak occurs. Therefore, for convenience, we may greatly simplify the process of determining where a peak occurs by developing a peak-enhancement utility to find peaks and represent them as Dirac delta functions, ignoring other data values by representing them as zero. In this manner, a series of impulse functions similar to the ideal ones shown in Fig. 31 can be visualized to represent peak occurrences. It is noted that potential peaks occurring at the largest value of σ analyzed are not included since it cannot be certain that they are indeed peaks.

6.2.2 Noise Threshold for Peak Spectrum

In some cases we may wish to ignore very small peaks arising primarily from noise. In this case we can define a noise threshold, below which all peaks are ignored. This threshold may be based on a percentage of the maximum peak found in a given surface, or it may represent a pre-defined noise floor. If a peak is detected, but falls below this noise threshold, we simply ignore it.

6.2.3 Projected Peak Spectrum

From the peak spectrum defined in Sec. 6.2.1 we may wish to project the maximum peak found along the σ axis onto the ω axis creating a projected peak spectrum. This is analogous to the spectrum obtained from a one-dimensional Fourier transform in that it contains only frequency information about the signal.

6.2.4 Simple Example

Using the signal defined by Eqn. (4.98) and Table 2 with an $SNR = 10$ dB, we now provide illustrative examples typical of the results we will examine in more detail in the simulations carried out in this chapter. For the conventional 2D Capon method with $N = 512$ and $M = 128$, we obtain the estimated spectrum $S(\sigma,\omega) = A(\sigma,\omega)e^{j\theta(\sigma,\omega)}$. Fig. 36 is a 3-dimensional plot of $|S(\sigma,\omega)|$ vs. σ and ω while Fig. 39 shows the same surface with peak-enhancement using a noise threshold of zero. Fig. 37 is a contour plot of $|S(\sigma,\omega)|$ vs. σ and ω while Fig. 40 shows the same contour with peak-enhancement. Fig. 38 is a projection of $|S(\sigma,\omega)|$ along σ onto the ω axis and Fig. 41 shows a similar plot with peak-enhancement. Fig. 42 is also provided to illustrate how the standard Fourier transform of this signal compares to the other techniques.

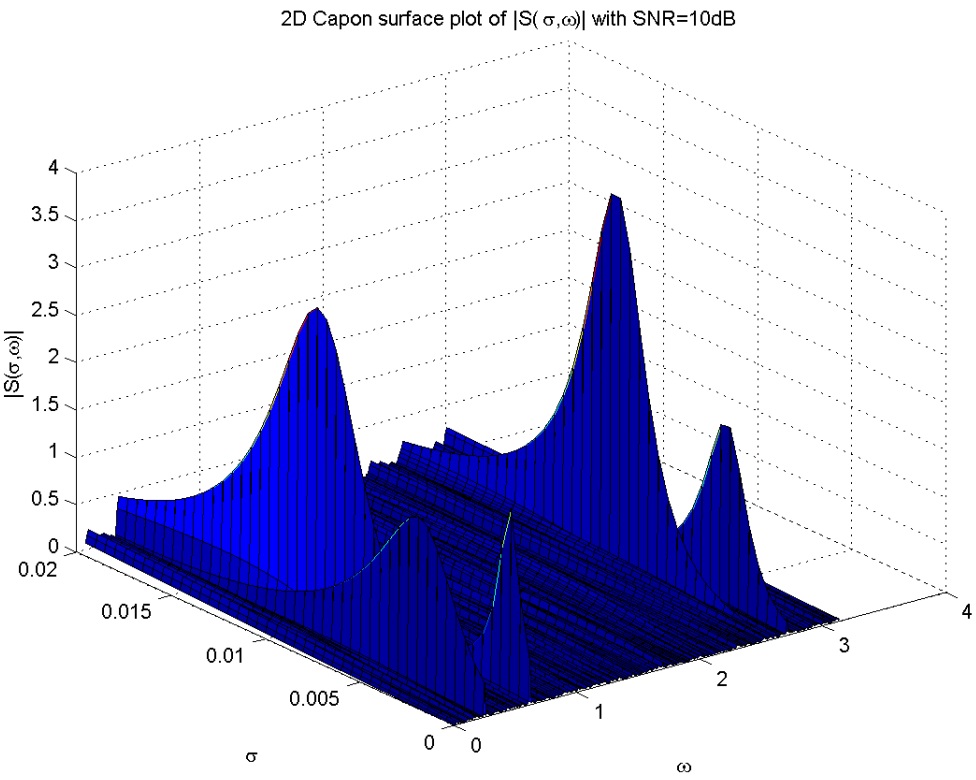

Fig. 36. Raw 2D Capon spectrum surface.

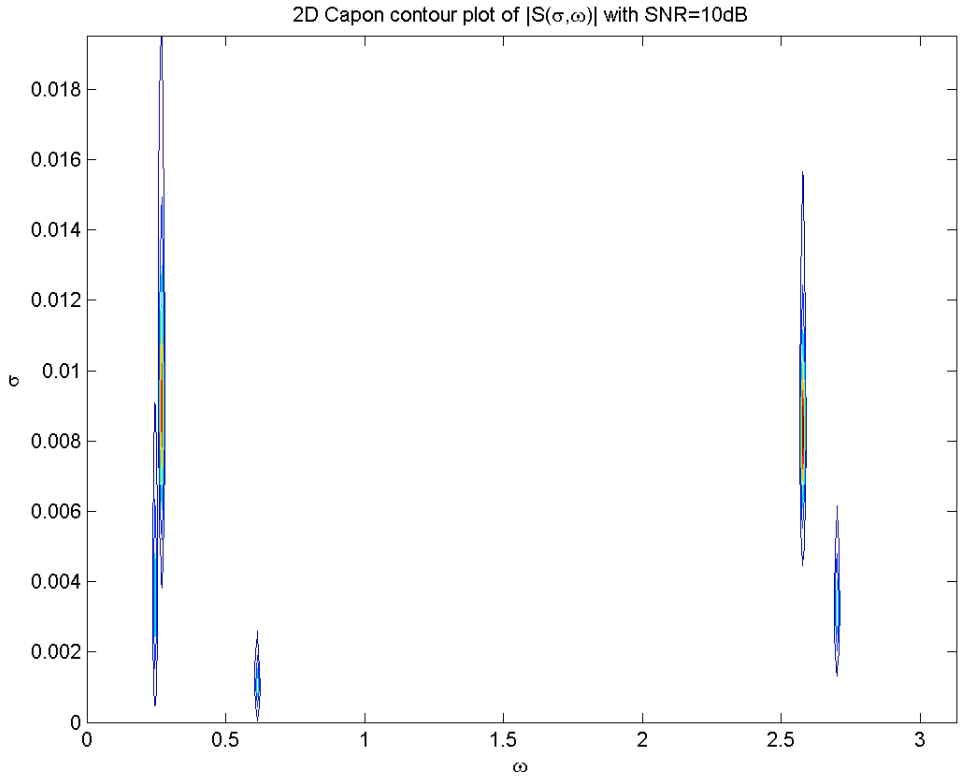

Fig. 37. Raw 2D Capon spectrum contour.

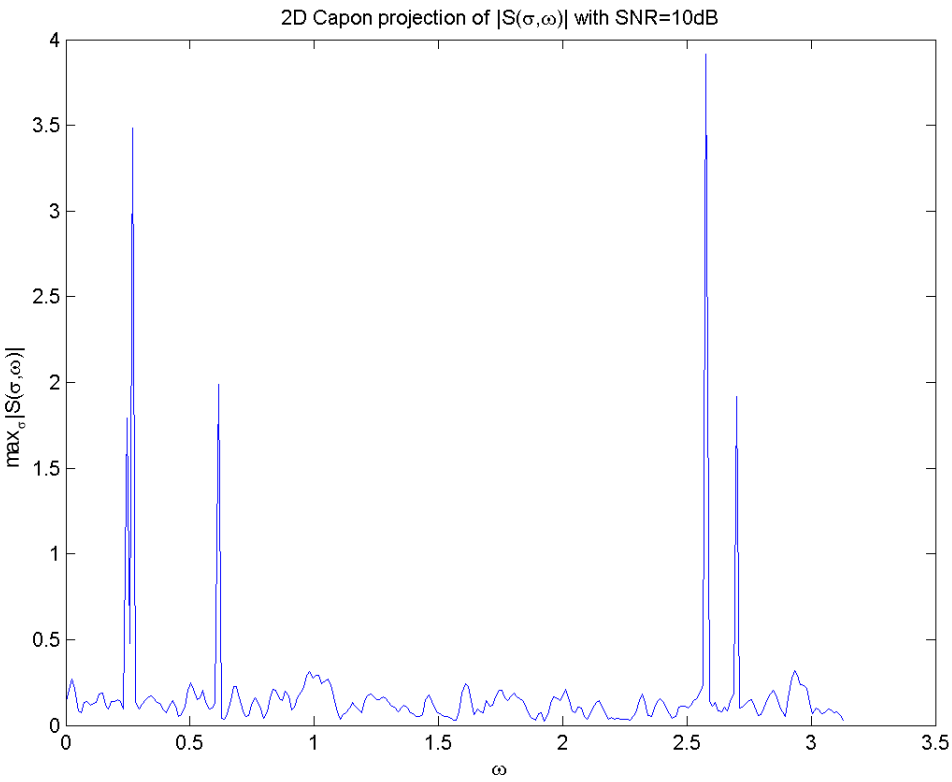

Fig. 38. Raw 2D Capon spectrum projection.

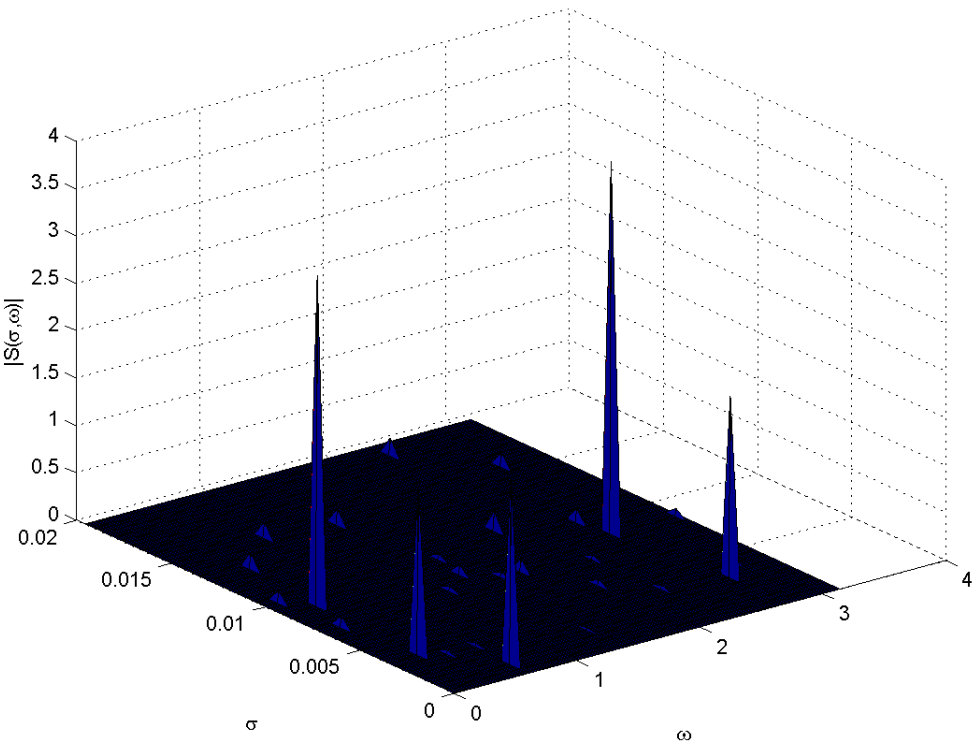

Fig. 39. Peak-enhanced 2D Capon spectrum surface.

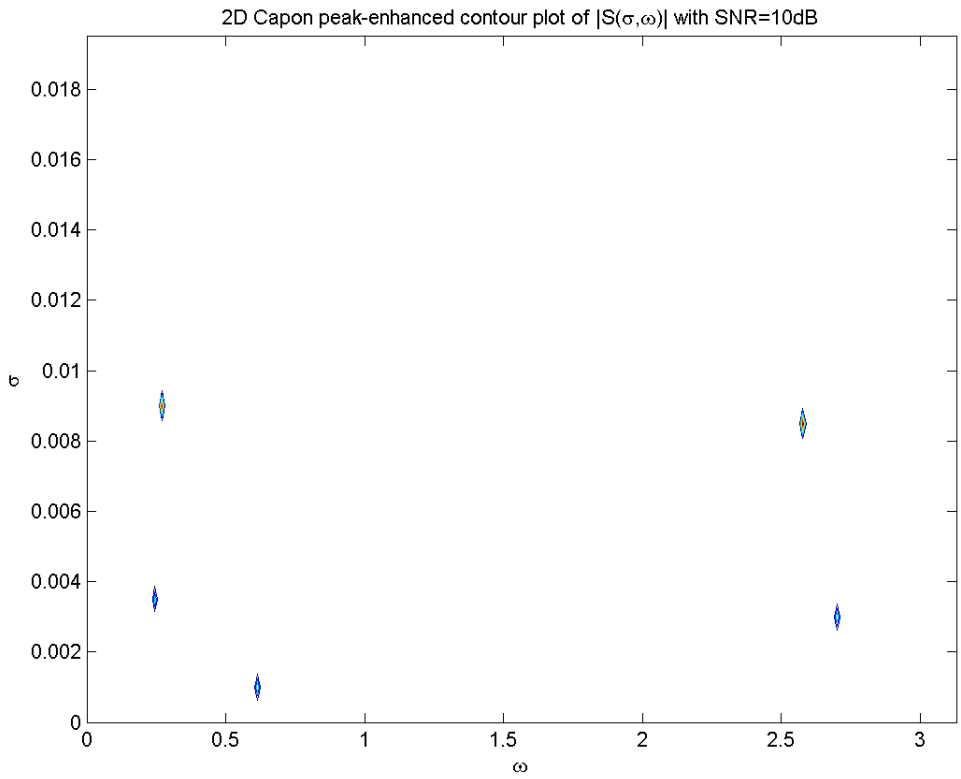

Fig. 40. Peak-enhanced 2D Capon spectrum contour.

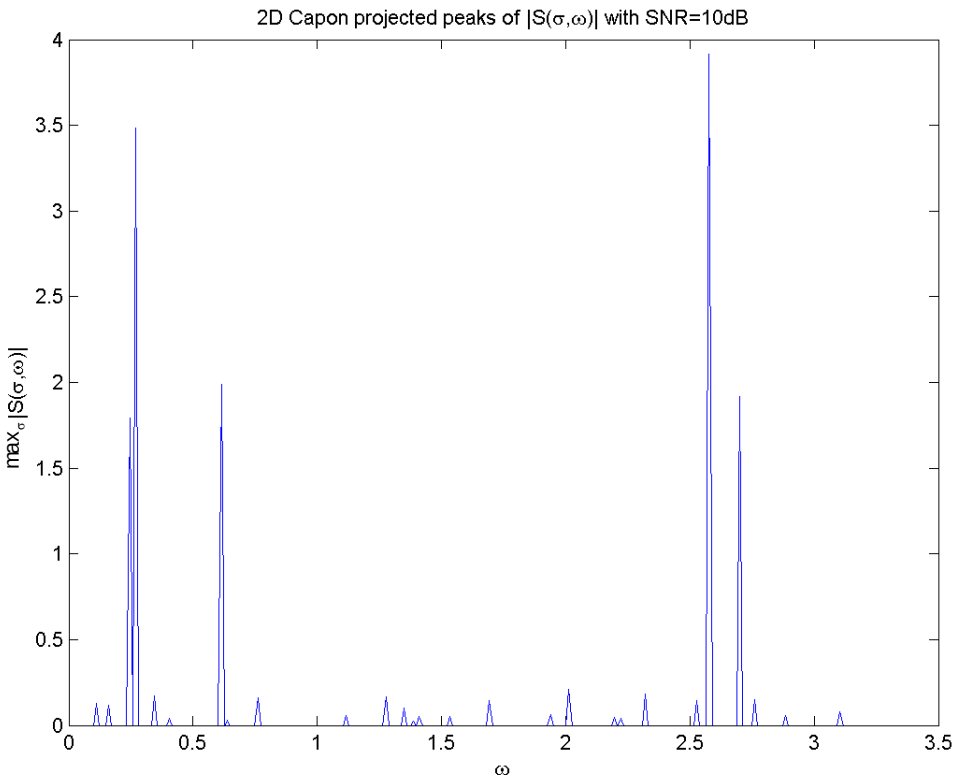

Fig. 41. Peak-enhanced 2D Capon spectrum projection.

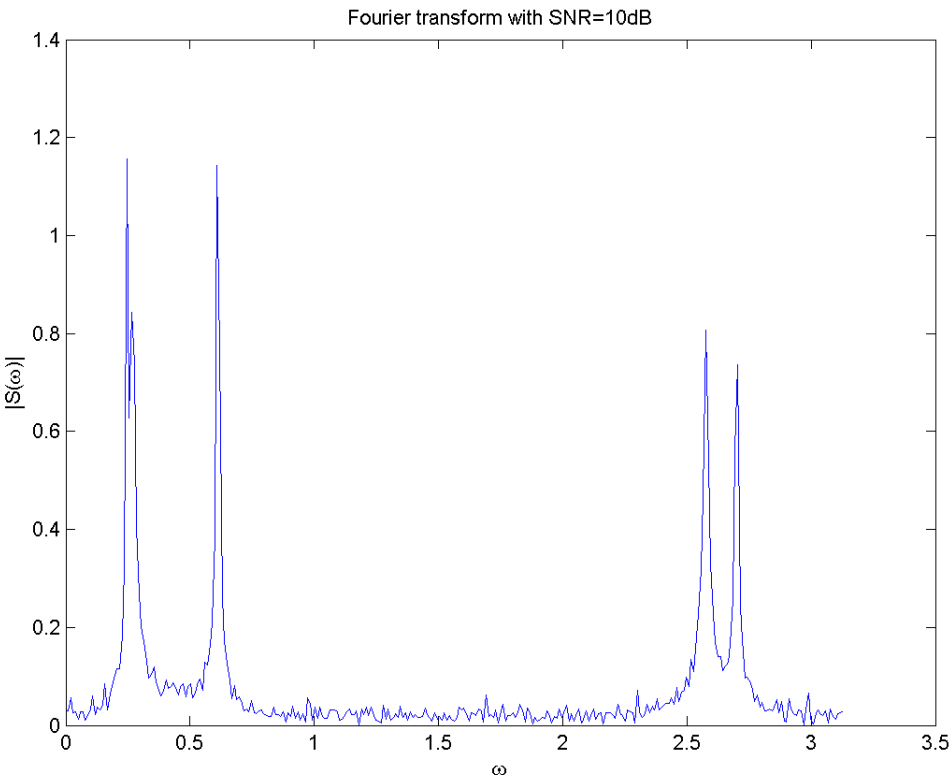

Fig. 42. Fourier transform.

6.3 Simulation Procedure

All simulations are implemented using MATLAB® version 6.5 on standard Microsoft Windows® based personal computers (PCs). Each resulting data set is analyzed to determine how many expected peaks were detected, and also how many false peaks were detected. If an expected peak is detected, the root-mean-square (RMS) error of the magnitude of this peak and the RMS error of the location of the peak in the (σ, ω) plane are computed.

6.3.1 Peak Analysis

An important aspect of the simulation strategy is to define a mechanism that can be used systematically to analyze the results, and quantify with accuracy and precision the performance of the particular algorithm being tested. A robust procedure is required to determine the location of peaks in the (σ, ω) plane. Therefore, the following peak detection method is proposed. The peak identification process determines that a peak *exists* if its estimated magnitude is greater than *or equal to* the estimated magnitude at all of the 3, 5, or 8 adjacent grid points.

In the simulations we will refer to those peaks known to exist in the simulated signal as *expected peaks*. These have known (expected) magnitude A, phase θ, damping σ, and frequency ω. All peaks identified by the estimation technique and peak picking process will be referred to as *detected peaks*. A detected peak is considered as a *true peak* if its ω matches that of some expected peak (i.e., lies on the same ω grid line). All remaining detected peaks are classified as *false peaks*.

For simplicity, we evaluate only one peak from each ω grid line. In the case of multiple peaks on the same ω grid line, we use the peak with the maximum value, or in the case of two or more equal peaks on the same ω grid line, we use the peak closest to $\sigma = 0$. Furthermore, "peaks" detected on the maximum σ grid line are ignored since it cannot be assured that these are indeed peaks.

6.3.2 Figures of Merit

Each simulation is evaluated to quantify the accuracy and effectiveness of the spectral estimation technique in finding the expected peaks and estimating their characteristics. The four figures of merit used for this evaluation are listed as follows.

6.3.2.1 Percent Missed Peaks

$$\%\ Missed\ \ Peaks = \left(\frac{\#\ of\ expected\ \ peaks - \#\ of\ \ true\ \ peaks}{\#\ of\ expected\ \ peaks} \right) \times 100\% \qquad (6.1)$$

6.3.2.2 Percent False Peaks

$$\%\ False\ \ Peaks = \left(\frac{\#\ of\ \ false\ \ peaks}{\#\ of\ \ possible\ \ peaks} \right) \times 100\% \qquad (6.2)$$

where $\#\ of\ \ false\ \ peaks = \#\ of\ detected\ \ peaks - \#\ of\ true\ peaks$ and

$\#\ of\ \ possible\ peaks = \#\ of\ \omega\ grid\ lines$.

6.3.2.3 Relative RMS Magnitude Error

$$Relative\ RMS\ Magnitude\ Error = \sqrt{\frac{1}{\#\ of\ true\ peaks} \sum \frac{(\Delta\ magnitude)^2}{(expected\ magnitude)^2}} \qquad (6.3)$$

where $\Delta\ magnitude = estimated\ magnitude - expected\ magnitude$

and where the sum is taken over all true peaks.

6.3.2.4 Relative RMS Damping (σ) Error

$$Relative\,RMS\,Damping\,Error = \sqrt{\frac{1}{\#\,of\ true\ peaks}\sum\frac{\left(\Delta\,damping\right)^2}{\left(maximum\,damping\right)^2}} \qquad (6.4)$$

where $\Delta\,damping = estimated\,damping - expected\,damping$,

$maximum\,damping = value\,of\ maximum\,\sigma\ grid\ line$,

and where the sum is taken over all true peaks.

6.3.3 Test Signals

Three signals comprised of various known frequency and damping components will be used for the simulations in this chapter. These were chosen to provide a variety with regard to the number of signal components, closeness of peaks, peak amplitude ratios, etc. Parameters for these test signals are provided in Table 3-Table 5 and the test signals are shown in Fig. 43-Fig. 45.

6.3.3.1 Test Signal I

$$\begin{aligned} x[n] = {} & e^{j\left(\pi/4\right)}e^{-0.002n}e^{j\left(63\pi/256\right)n} + \\ & 0.75e^{-j\left(\pi/2\right)}e^{-0.006n}e^{j\left(127\pi/256\right)n} + \\ & 0.25e^{-0.004n}e^{j\left(191\pi/256\right)n} \end{aligned} \qquad (6.5)$$

ω	σ	A	θ
$63\pi/256$	-0.002	1.0	$\pi/4$
$127\pi/256$	-0.006	0.75	$-\pi/2$
$191\pi/256$	-0.004	0.25	0

Table 3. Parameters for test signal I.

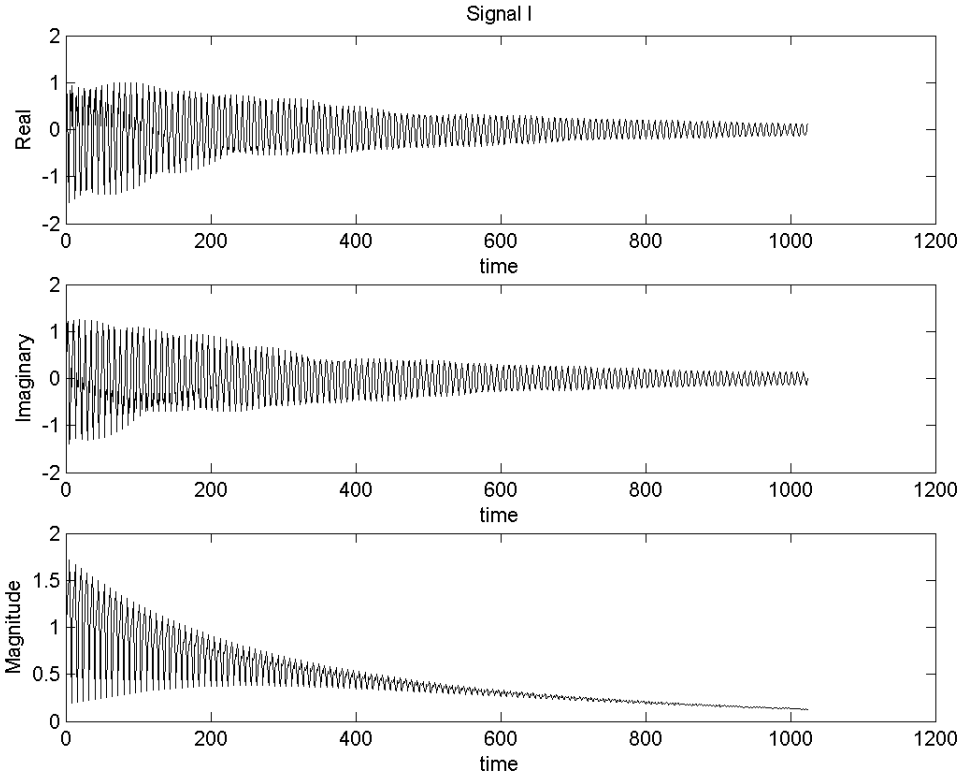

Fig. 43. Test signal I.

6.3.3.2 Test Signal II

$$
\begin{aligned}
x[n] = {} & e^{j(\pi/4)} e^{-0.001n} e^{j(30\pi/256)n} + \\
& 25 e^{-0.002n} e^{j(40\pi/256)n} + \\
& 0.75 e^{j(-\pi/4)} e^{-0.004n} e^{j(50\pi/256)n} + \\
& 0.25 e^{-0.003n} e^{j(100\pi/256)n} + \\
& e^{-0.001n} e^{j(120\pi/256)n} + \\
& 10 e^{j(\pi/2)} e^{-0.002n} e^{j(125\pi/256)n} + \\
& 0.5 e^{j(-\pi/2)} e^{-0.002n} e^{j(195\pi/256)n} + \\
& e^{-0.003n} e^{j(225\pi/256)n}
\end{aligned}
\tag{6.6}
$$

ω	σ	A	θ
$30\pi/256$	-0.001	1.0	$\pi/4$
$40\pi/256$	-0.002	25.0	0
$50\pi/256$	-0.004	0.75	$-\pi/4$
$100\pi/256$	-0.003	0.25	0
$120\pi/256$	-0.001	1.0	0
$125\pi/256$	-0.002	10.0	$\pi/2$
$195\pi/256$	-0.002	0.5	$-\pi/2$
$225\pi/256$	-0.003	1.0	0

Table 4. Parameters for test signal II.

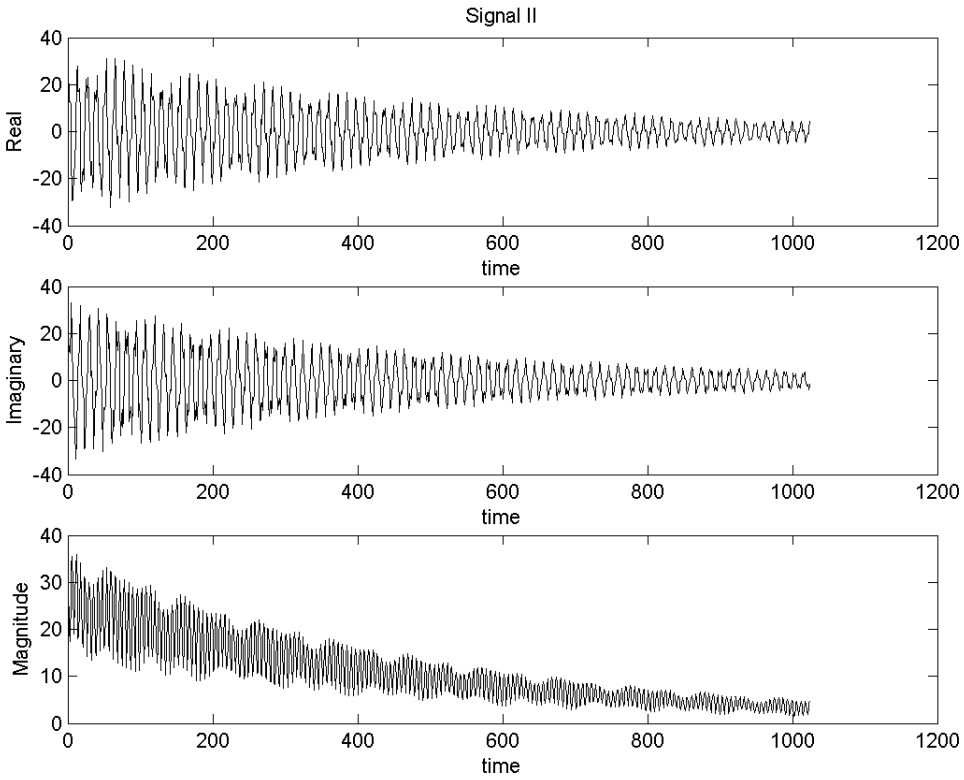

Fig. 44. Test signal II.

6.3.3.3 Test Signal III

$$
\begin{aligned}
x[n] = {} & 0.5e^{-0.004n} + \\
& 0.75e^{j\left(-\pi/2\right)}e^{-0.002n}e^{j\left(10\pi/256\right)n} + \\
& e^{j\left(\pi/4\right)}e^{-0.003n}e^{j\left(31\pi/256\right)n} + \\
& 0.75e^{-0.003n}e^{j\left(34\pi/256\right)n} + \\
& 0.5e^{j\left(-\pi/2\right)}e^{-0.005n}e^{j\left(55\pi/256\right)n} + \\
& e^{j\left(\pi/2\right)}e^{-0.001n}e^{j\left(63\pi/256\right)n} + \\
& e^{j\left(3\pi/4\right)}e^{-0.005n}e^{j\left(66\pi/256\right)n} + \\
& 0.25e^{-0.004n}e^{j\left(73\pi/256\right)n} + \\
& 0.5e^{j\left(\pi/2\right)}e^{j\left(95\pi/256\right)n} + \\
& 0.5e^{j\left(-\pi/2\right)}e^{-0.006n}e^{j\left(104\pi/256\right)n}
\end{aligned}
\tag{6.7}
$$

ω	σ	A	θ
0	-0.004	0.50	0
$10\pi/256$	-0.002	0.75	$-\pi/2$
$31\pi/256$	-0.003	1.0	$\pi/4$
$34\pi/256$	-0.003	0.75	0
$55\pi/256$	-0.005	0.50	$-\pi/2$
$63\pi/256$	-0.001	1.0	$\pi/2$
$66\pi/256$	-0.005	1.0	$3\pi/4$
$73\pi/256$	-0.004	0.25	0
$95\pi/256$	0	0.50	$\pi/2$
$104\pi/256$	-0.006	0.50	$-\pi/2$

Table 5. Parameters for test signal III.

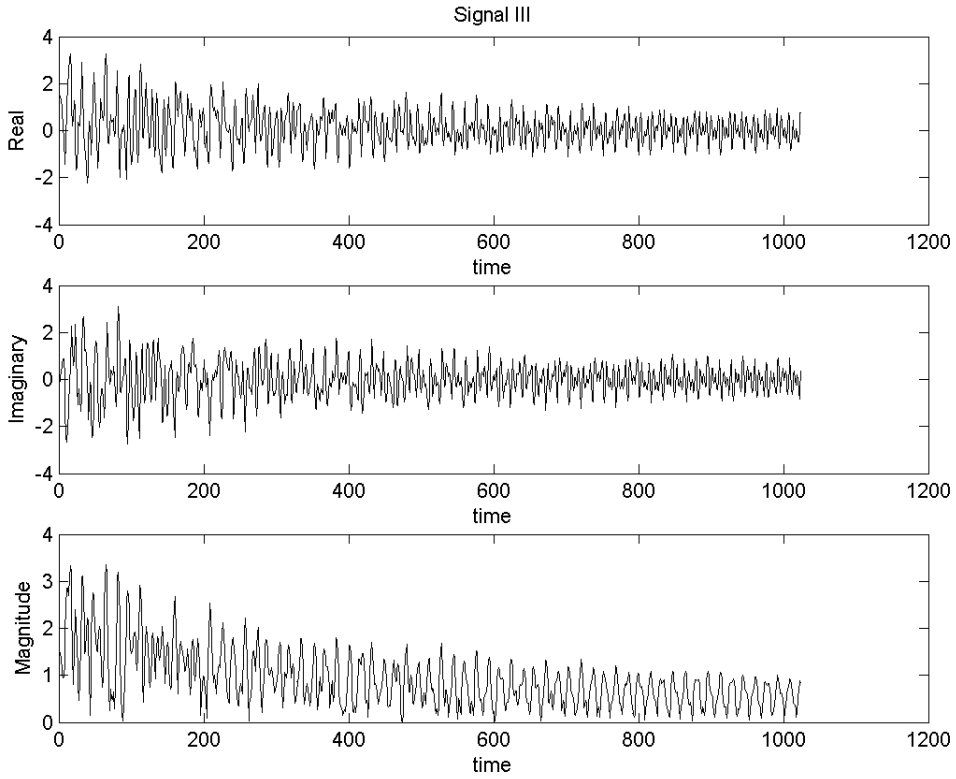

Fig. 45. Test signal III.

6.3.4 SNR Considerations

An appropriate amount of noise is added to each of the test signals to achieve the SNR desired for the particular simulation. The additive noise is white, Gaussian, and zero-mean. The details of how SNR is computed for these simulations are provided in the Appendix in Sec. A.3. For multiple-channel simulations, it is possible that the channels being simulated may contain different SNR values. For the analysis of these

simulations, we will consider the maximum SNR of all channels as the parameter of interest.

6.3.5 Comment on Grid Points

Empirical evidence suggests that simulated signal components not occurring at a precise ω grid point may be difficult if not impossible to detect. Therefore, all simulations being evaluated contain signal components that occur at frequencies corresponding to grid point values. Fortunately, MRS data sets do not appear to have similar limitations.

6.3.6 Simulations

A series of 14 different simulation sets was carried out. The results of each set were stored to a file, and plotted for ease of interpretation. These simulations are related to measuring the accuracy and efficiency of peak identification using the new techniques introduced in Chapter 4 and Chapter 5. For each simulation $N = 512$, the number of frequency (ω) grid points is $N_\omega = 256$, the number of damping (σ) grid points is $N_\sigma = 40$, and the threshold below which all peaks are ignored is $peak\ threshold = 0.025$ (10% of the *smallest* signal component in each of the signals studied). In every case the SNR was calculated based on $N_f = 768$, regardless of the length M of the filter (see Appendix Sec. A.3).

6.3.6.1 Single-Channel Weighted 2D Spectral Estimation

Nine simulations are related to measuring the accuracy and efficiency of peak identification using the weighted 2D Capon method, the weighted 2D APES method, and the combined weighted 2D APES/2D Capon method introduced in Chapter 4. Each of these simulations used the three test signals, with 10 similar but different noise signals added at the appropriate *SNR*. Each simulation pass thus consisted of 30 separate spectrum determinations using the particular method being studied. For each of the 30 runs the figures of merit as discussed in Sec. 6.3.2 were calculated and then averaged together to form a combined representation which is then plotted.

6.3.6.2 Multiple-Channel Weighted 2D Spectral Estimation

Five simulation sets are related to measuring the accuracy and efficiency of peak identification using the weighted signal averaging method and the weighted spectrum averaging method introduced in Chapter 5 . Each of these simulations used the third test signal, defined in Eqn. (6.7), with 10 similar but different noise signals added at the appropriate *SNR*. Each simulation pass thus consisted of 10 separate spectrum determinations. In each case the figures of merit as discussed in Sec. 6.3.2 were calculated and then averaged together to form a combined representation which is then plotted.

6.4 Simulation Results

The results of the simulations carried out for single-channel weighted 2D spectral estimation are shown in Fig. 46 - Fig. 55. The results of the simulations carried out for

multiple-channel weighted 2D spectral estimation are shown in Fig. 56 -Fig. 60. In some situations it may not be possible to compute RMS error terms as specified by Eqn. (6.3) and Eqn. (6.4) since no peaks were detected. In this case, the data points are not represented on the plots.

In all cases the four figures of merit are plotted versus *SNR* where *SNR* varies from –18 dB to 48 dB in increments of 6 dB. For convenience, the parameters α and β are denoted as $\alpha = \alpha_1\sigma + \alpha_2$ and $\beta = \beta_1\sigma + \beta_2$. In the figure legends and captions any of the parameter values α_1, α_2, β_1, and β_2 not shown are zero. The specifications for the single-channel simulations and corresponding figures are given in Table 6.

Fig.	Method	*K*	*M*	α	β	γ
Fig. 46	weighted 2D Capon	*N*	variable	0	0	
Fig. 47	weighted 2D Capon	*N*	variable	$-\sigma/2$	0	
Fig. 48	weighted 2D Capon	1	variable		0	
Fig. 49	weighted 2D Capon	*N*	128	variable	variable	
Fig. 50	weighted 2D Capon	*N*	128	variable	variable	
Fig. 51	weighted 2D Capon	1	128		variable	
Fig. 52	weighted 2D APES		variable		0	
Fig. 53	variable	variable	128	variable	0	
Fig. 54	combined weighted 2D APES / 2D Capon		128		0	variable
Fig. 55	weighted 2D APES		128		variable	

Table 6. Single-channel simulation specifications.

It is recalled that weighted 2D Capon with $K = N$, $\alpha = 0$, and $\beta = 0$ reduces to conventional 2D Capon and weighted 2D APES with $\beta = 0$ reduces to conventional 2D APES.

All of the multiple-channel simulations use conventional 2D Capon with $M = 128$. Both signal averaging and spectrum averaging are studied for four channels $(C = 4)$. For comparison, the corresponding single-channel case $(C = 1)$ is also shown. The specifications for the multiple-channel simulations and the corresponding figures are given in Table 7.

Fig.	*SNR* Channel 1	*SNR* Channels 2-4	Channel Gains
Fig. 56	value shown	value shown	ideal
Fig. 57	value shown	value shown – 6 dB	ideal
Fig. 58	value shown	value shown – 12 dB	ideal
Fig. 59	value shown	value shown – 18 dB	ideal
Fig. 60	value shown	value shown	ideal and estimated

Table 7. Multiple-channel simulation specifications.

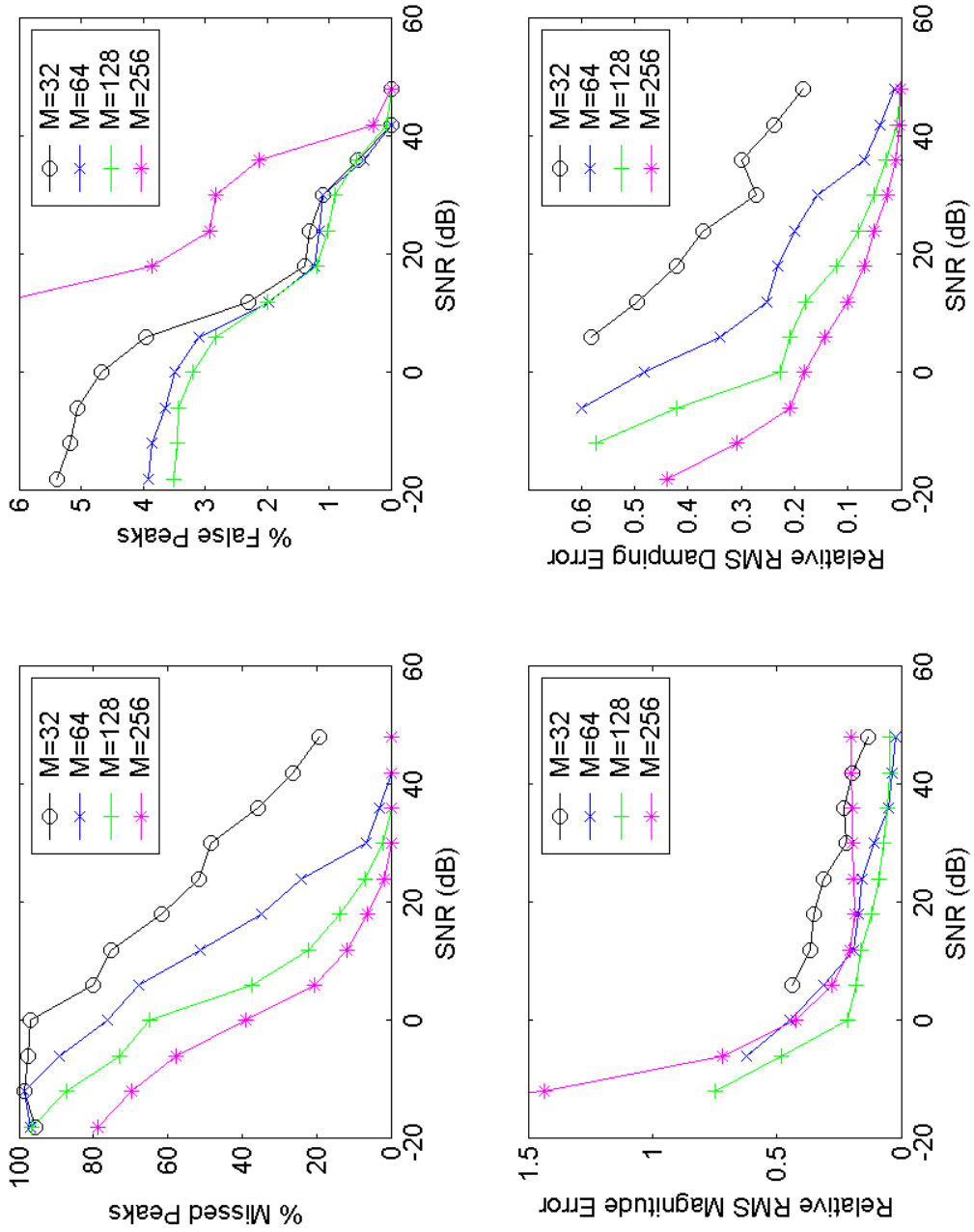

Fig. 46. Weighted 2D Capon $(K = N)$ for various M's (conventional 2D Capon).

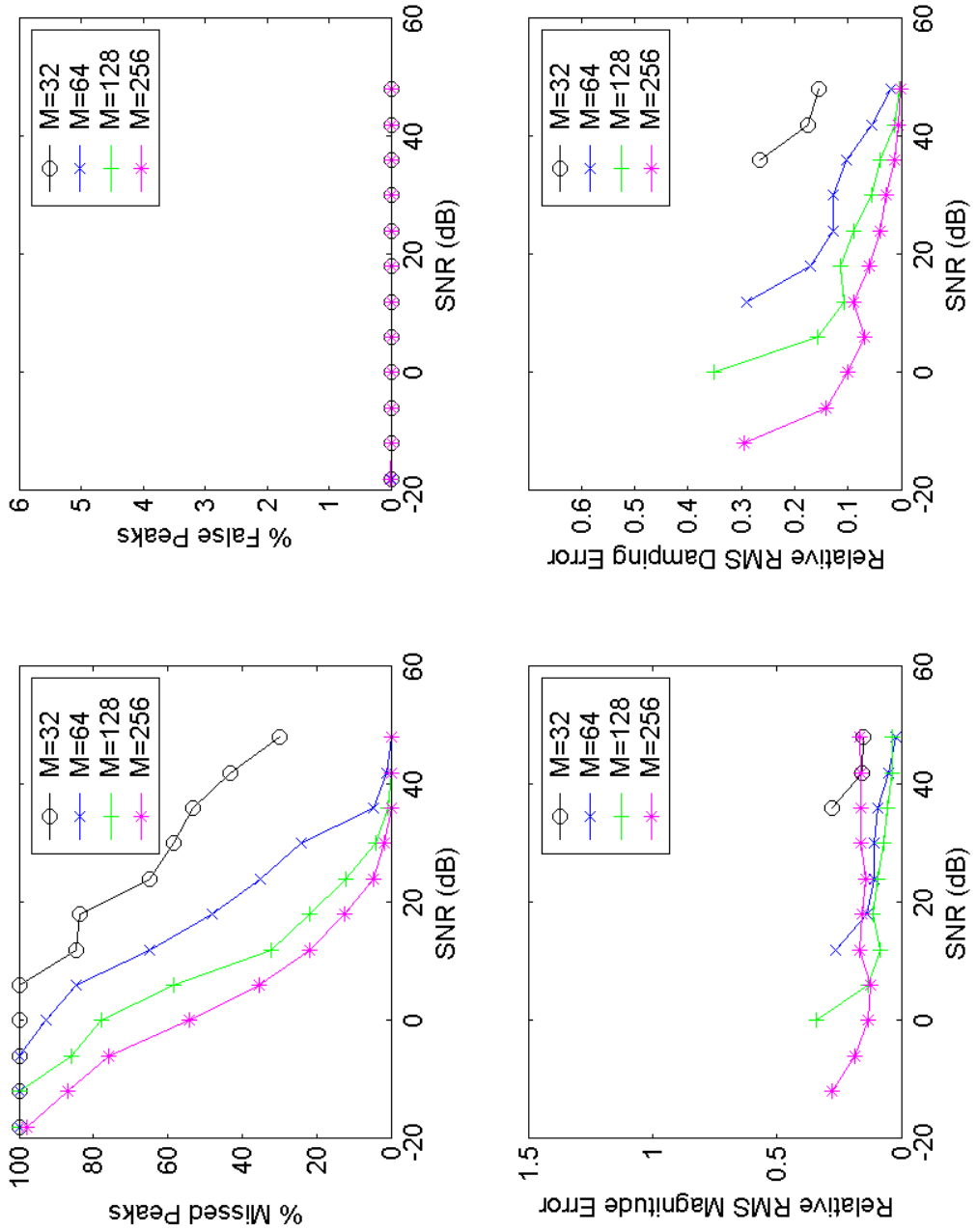

Fig. 47. Weighted 2D Capon $\left(K = N,\ \alpha = -\sigma_2^2\right)$ for various M's.

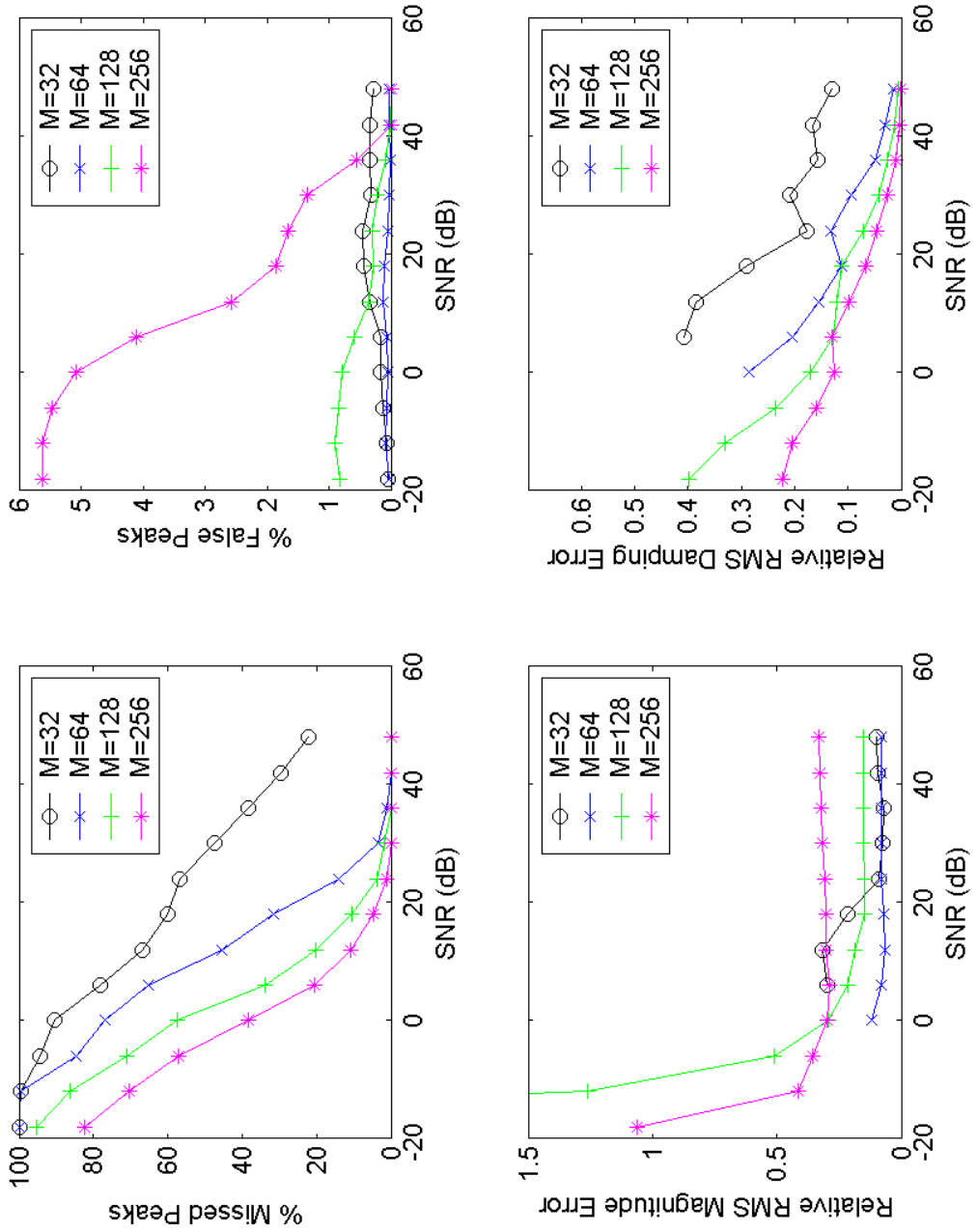

Fig. 48. Weighted 2D Capon $(K=1)$ for various M's.

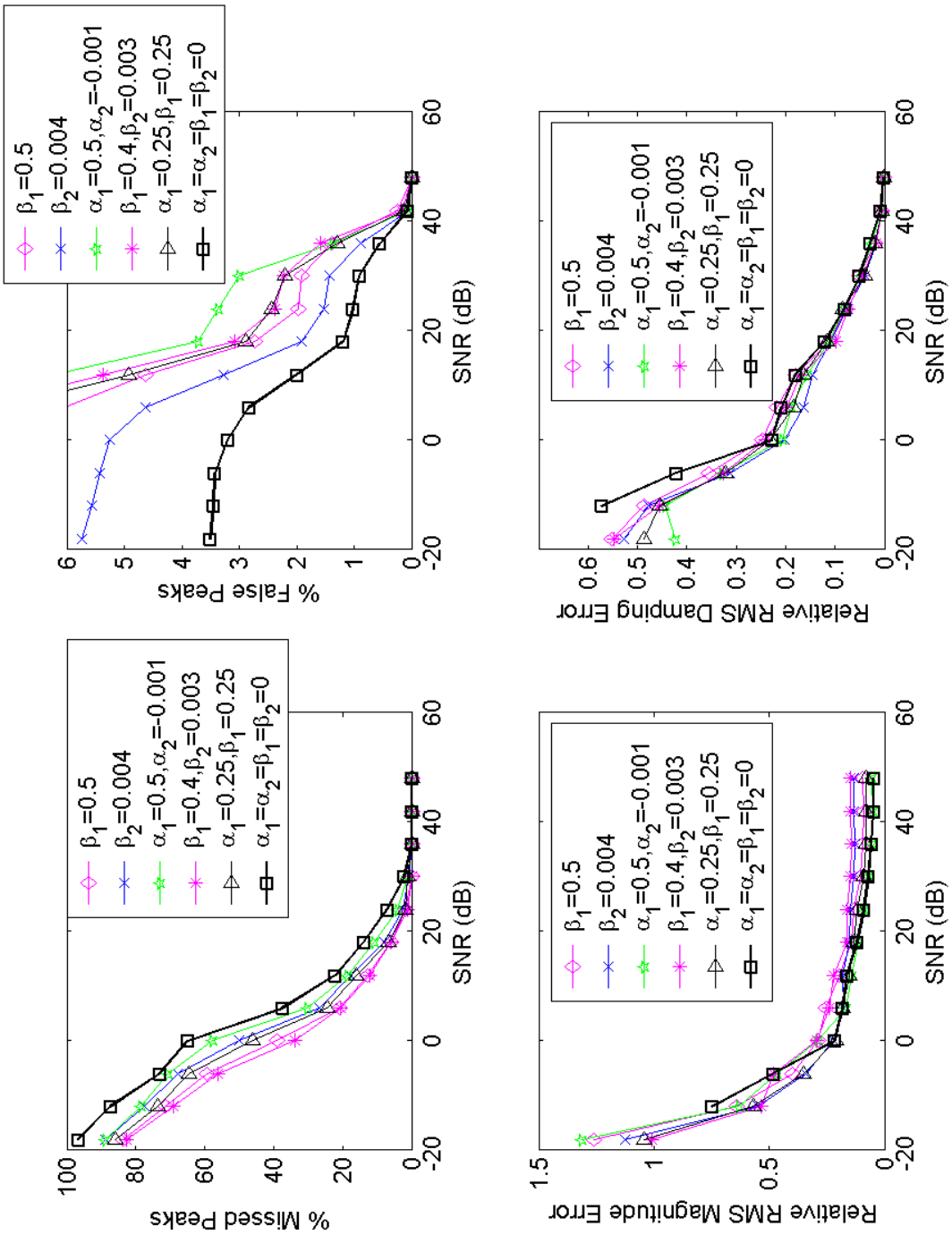

Fig. 49. Weighted 2D Capon $(K = N, M = 128)$ for various α's and β's.

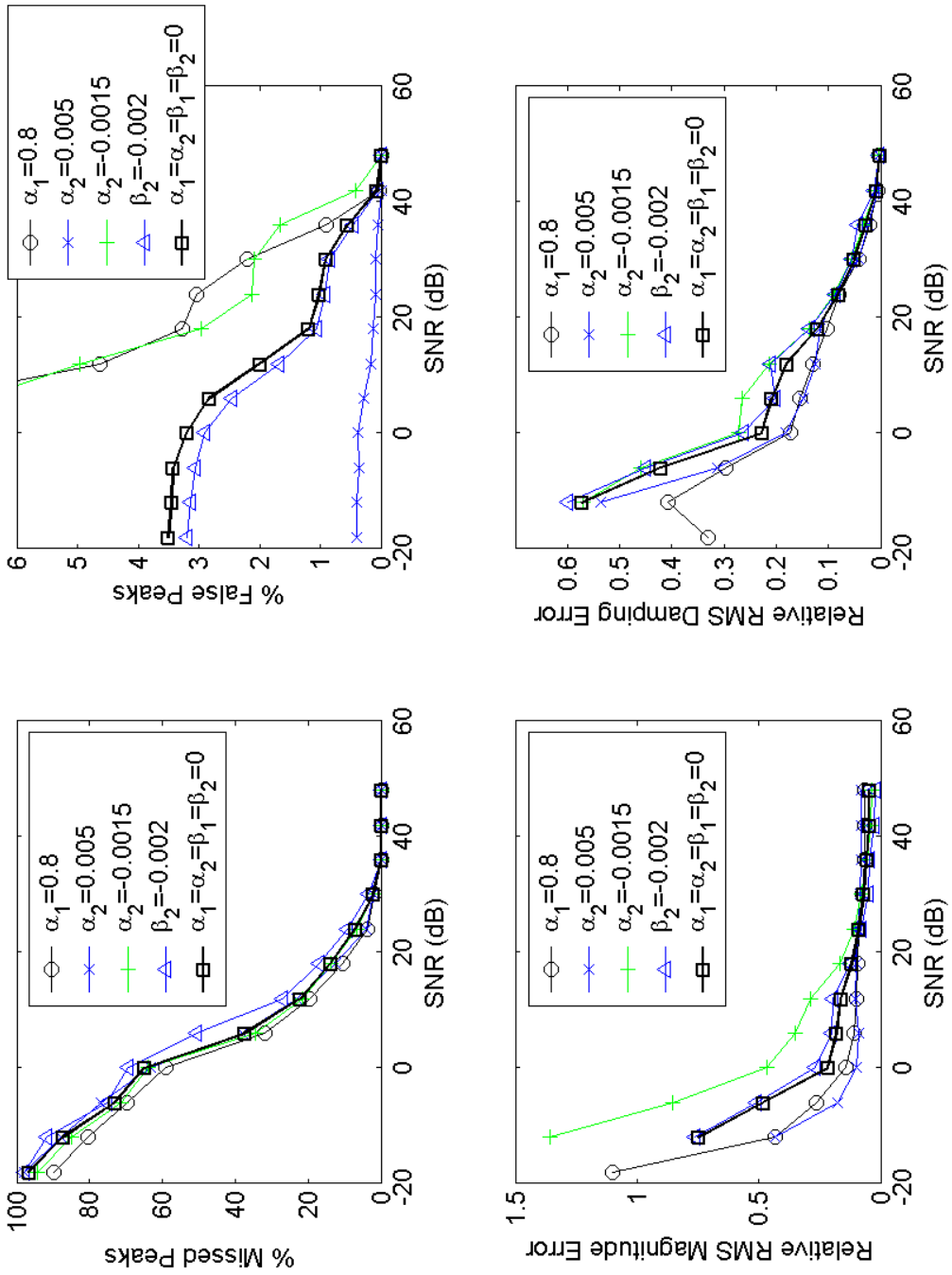

Fig. 50. Weighted 2D Capon $(K = N, M = 128)$ for various α's and β's.

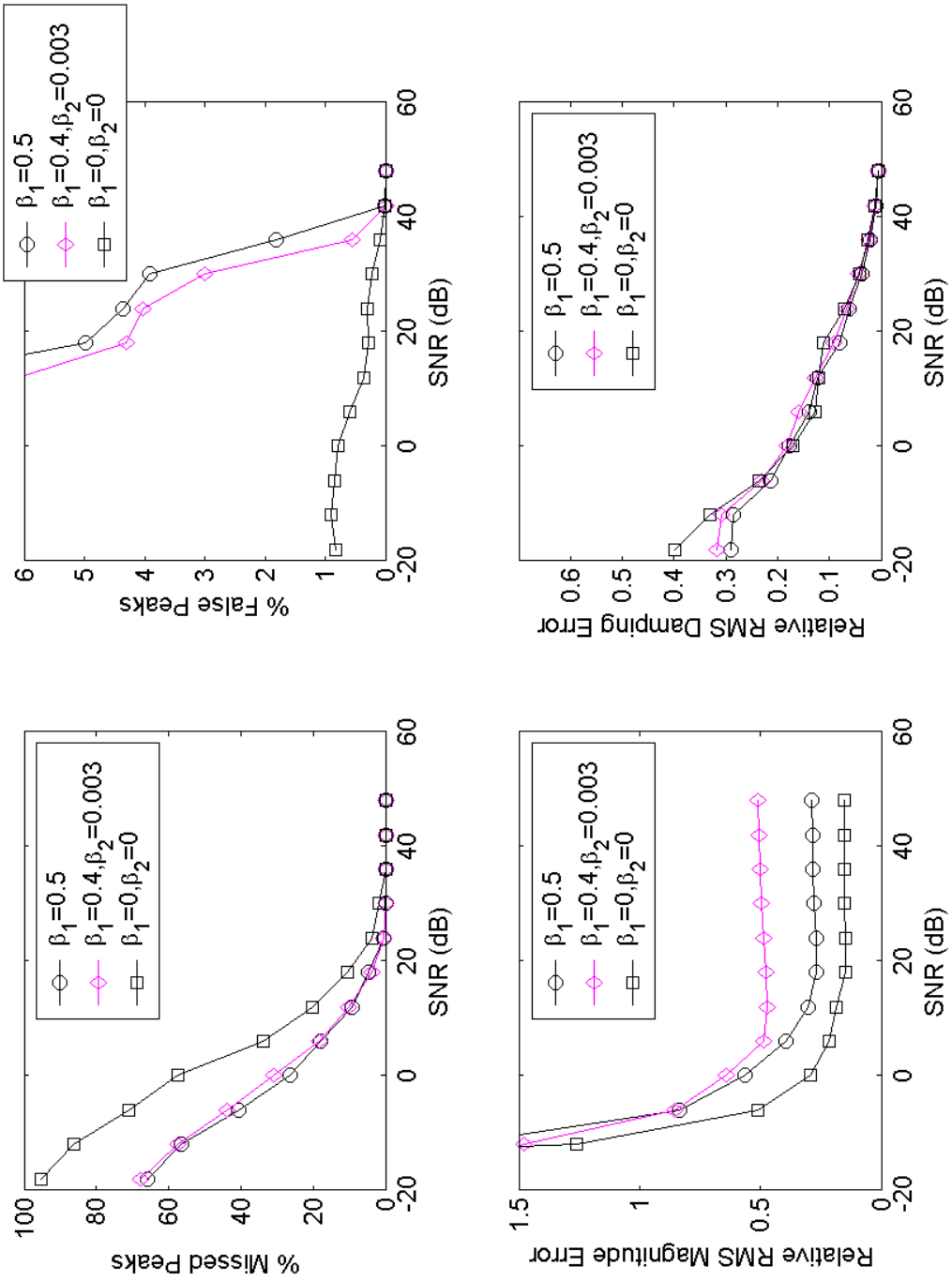

Fig. 51. Weighted 2D Capon $(K=1, M=128)$ for various β's.

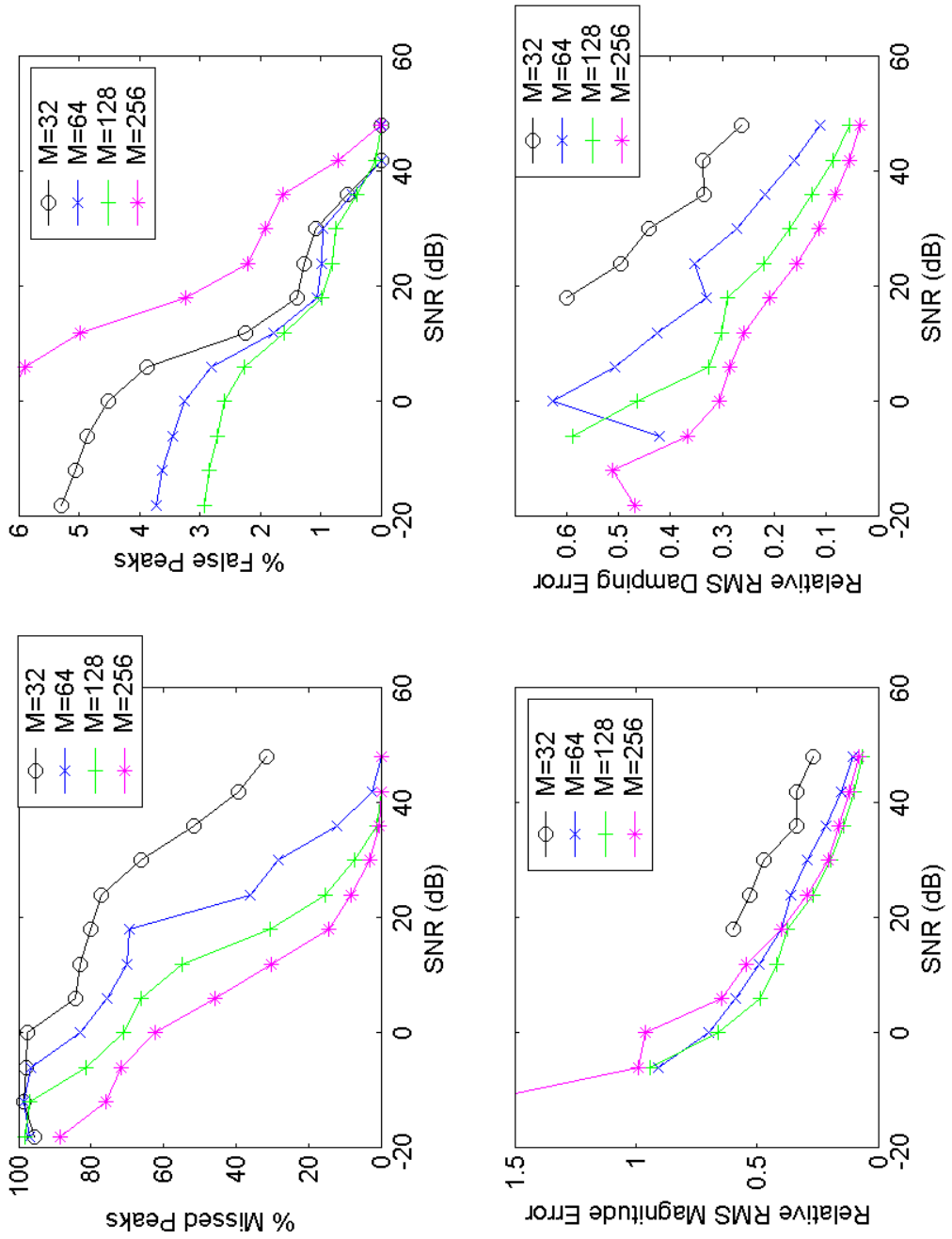

Fig. 52. Weighted 2D APES for various M's (conventional 2D APES).

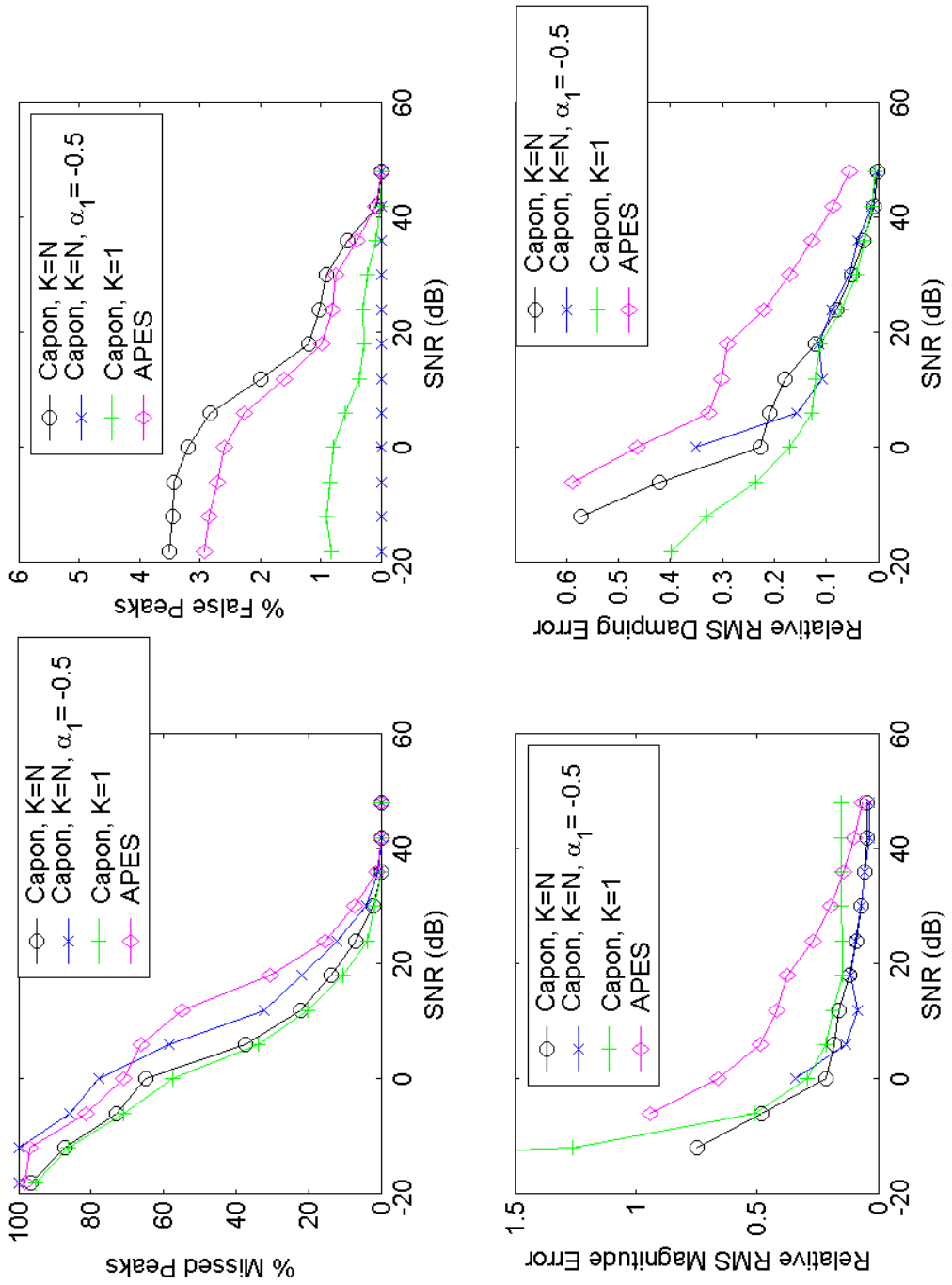

Fig. 53. Various weighted 2D spectral estimators for $M = 128$.

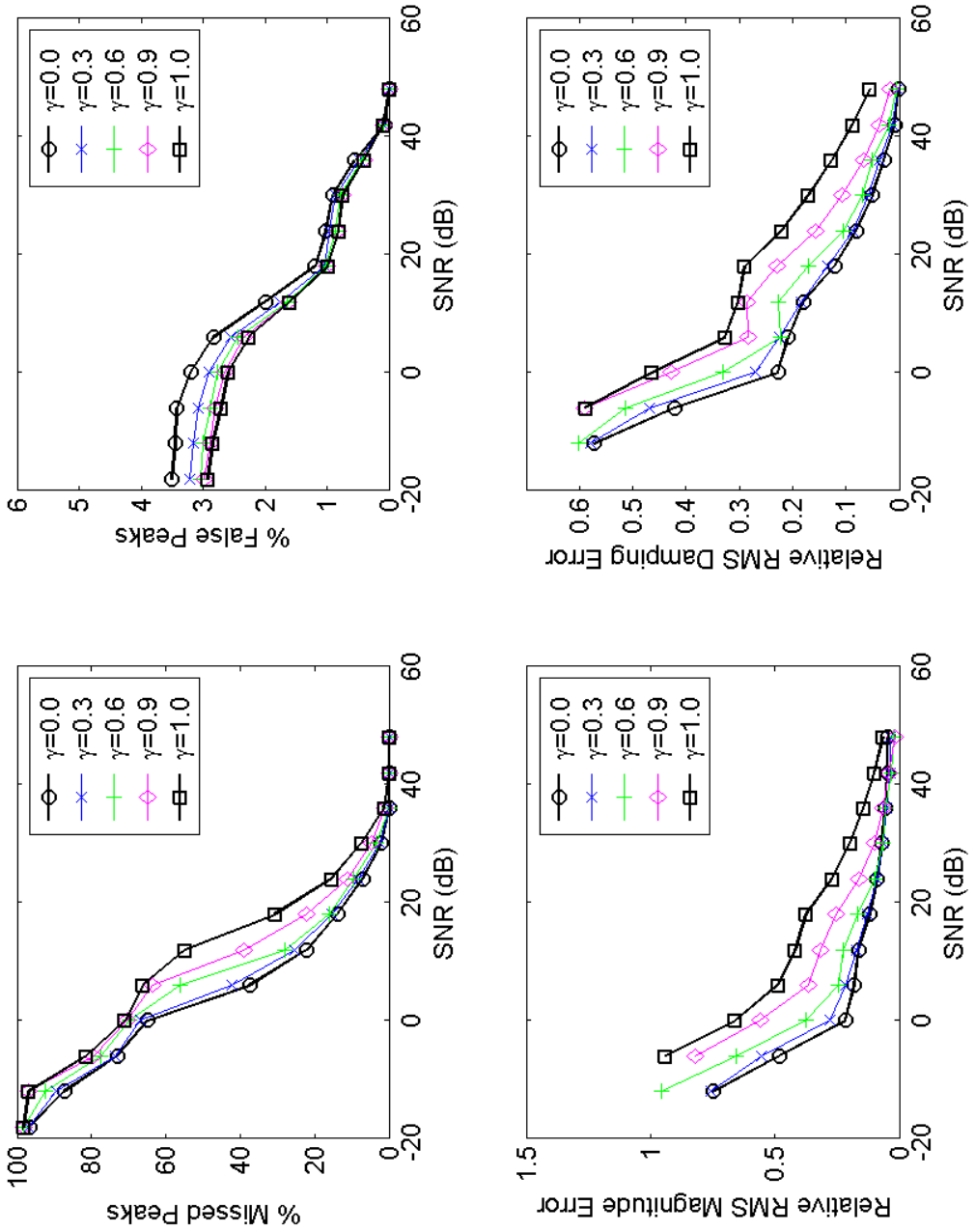

Fig. 54. Combined weighted 2D APES / 2D Capon $(M = 128)$ for various γ's.

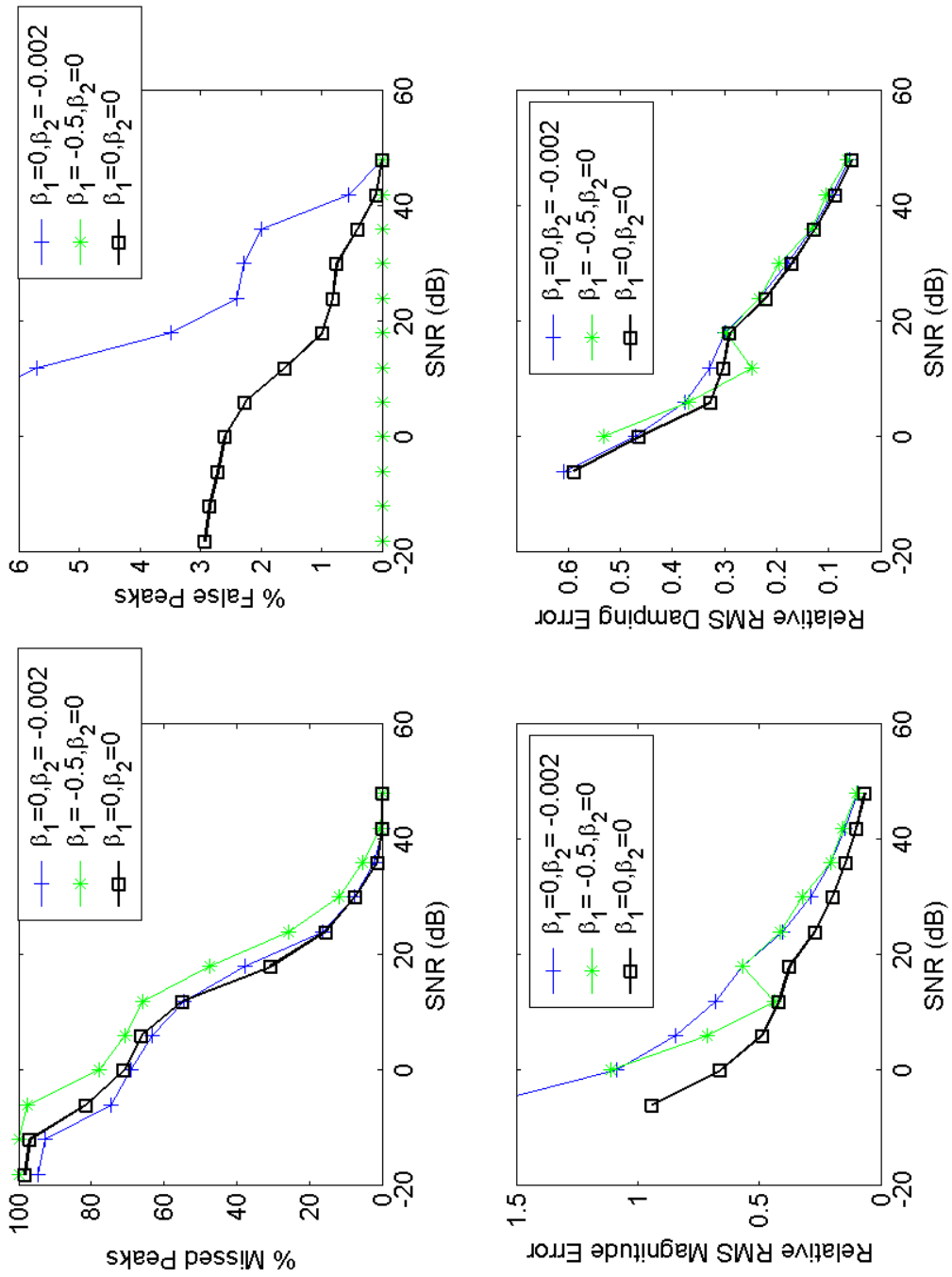

Fig. 55. Weighted 2D APES $(M=128)$ for various β's .

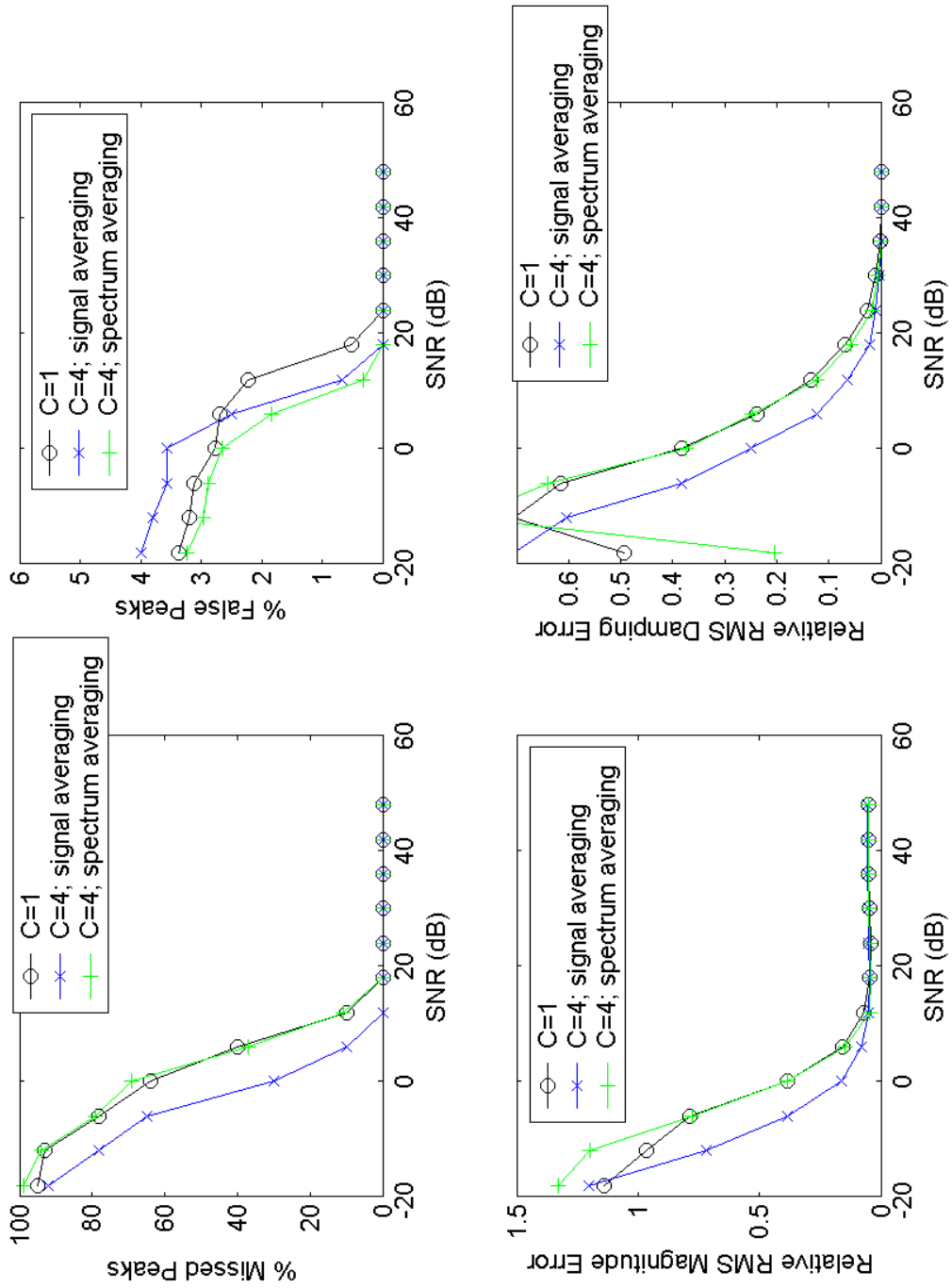

Fig. 56. Multiple-channel conventional 2D Capon (equal *SNR*'s).

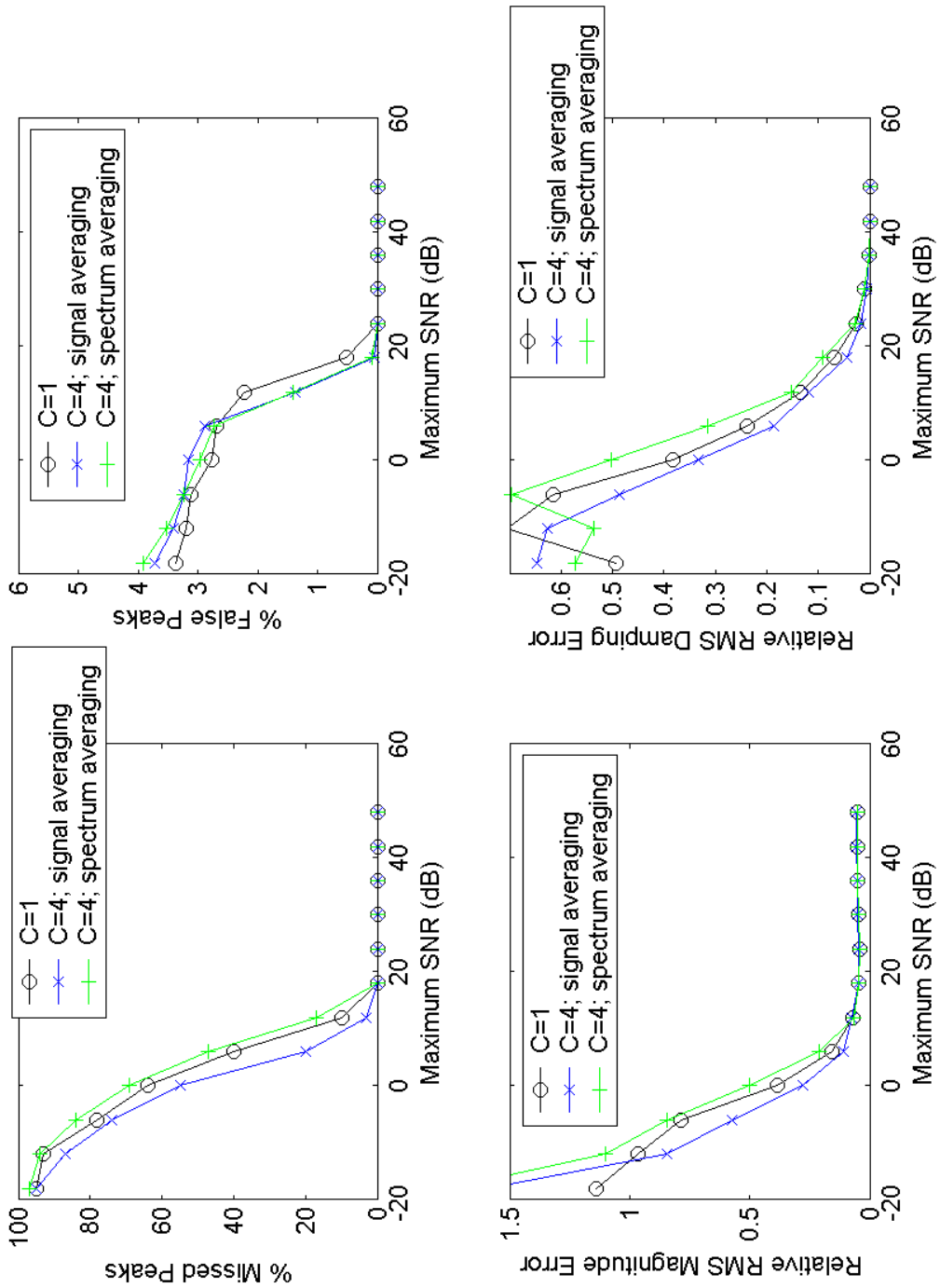

Fig. 57. Multiple-channel conventional 2D Capon (*SNR*'s reduced by 6 dB for three channels).

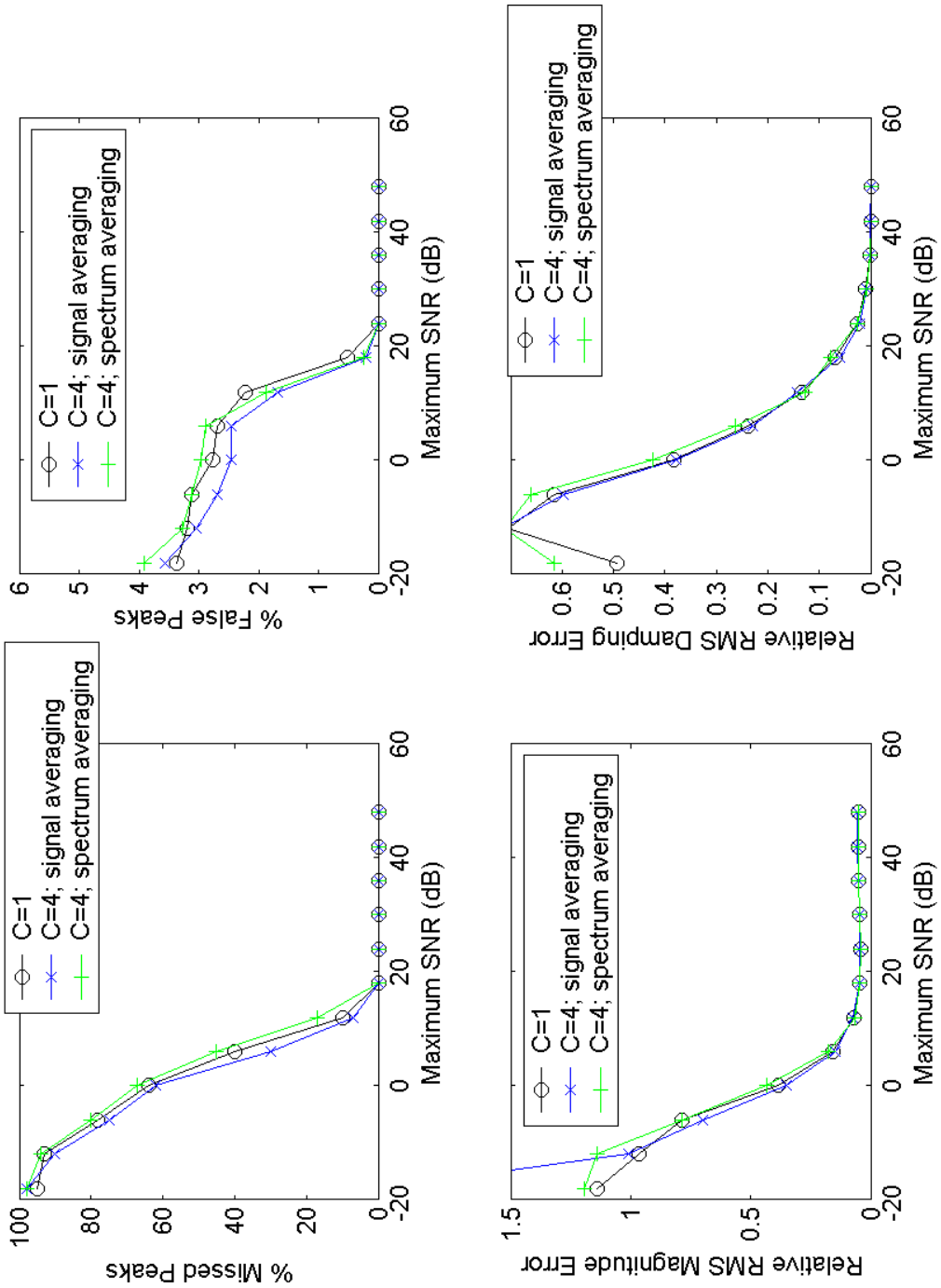

Fig. 58. Multiple-channel conventional 2D Capon (*SNR*'s reduced by 12 dB for three channels).

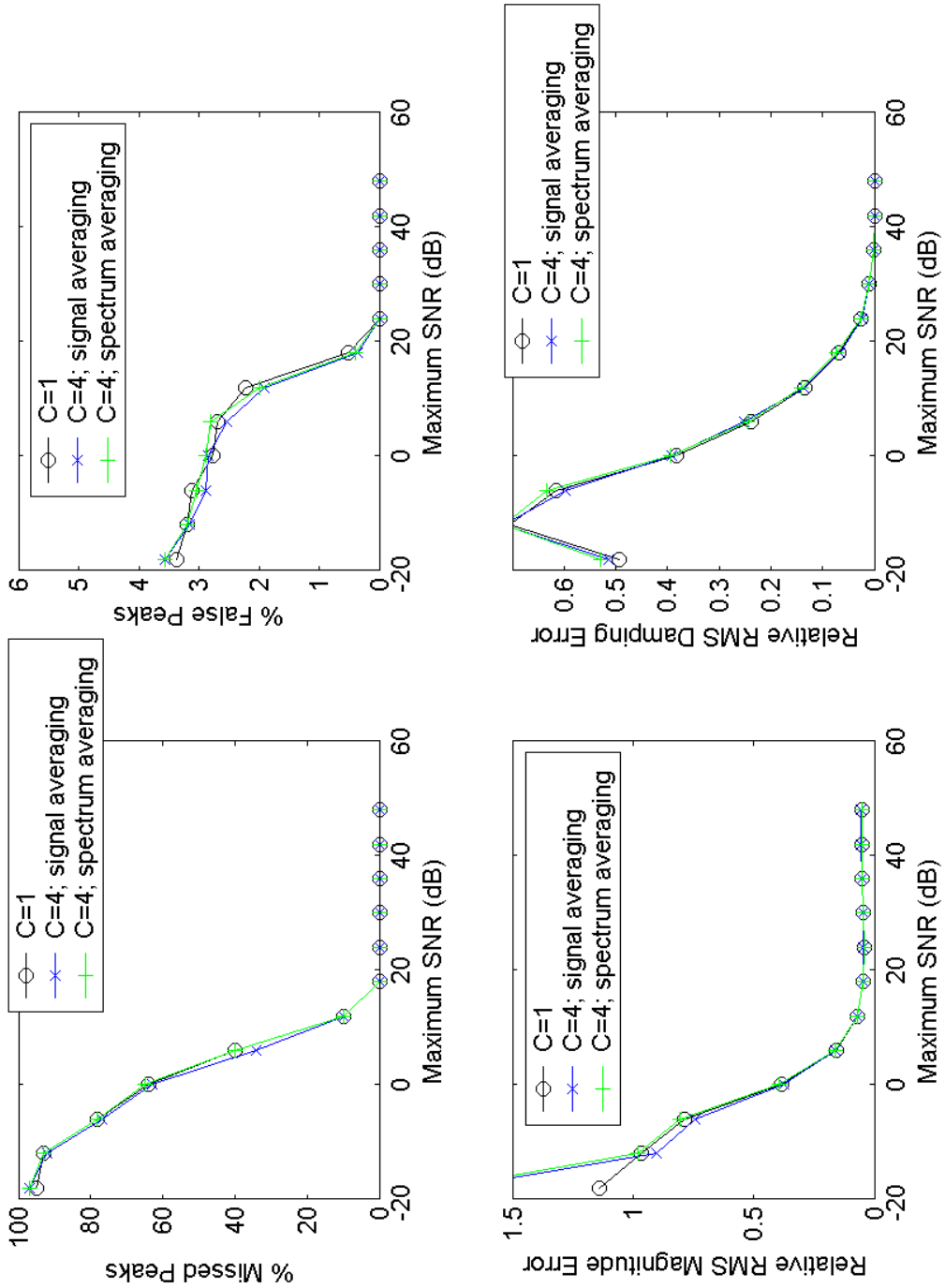

Fig. 59. Multiple-channel conventional 2D Capon (*SNR*'s reduced by 18 dB for three channels).

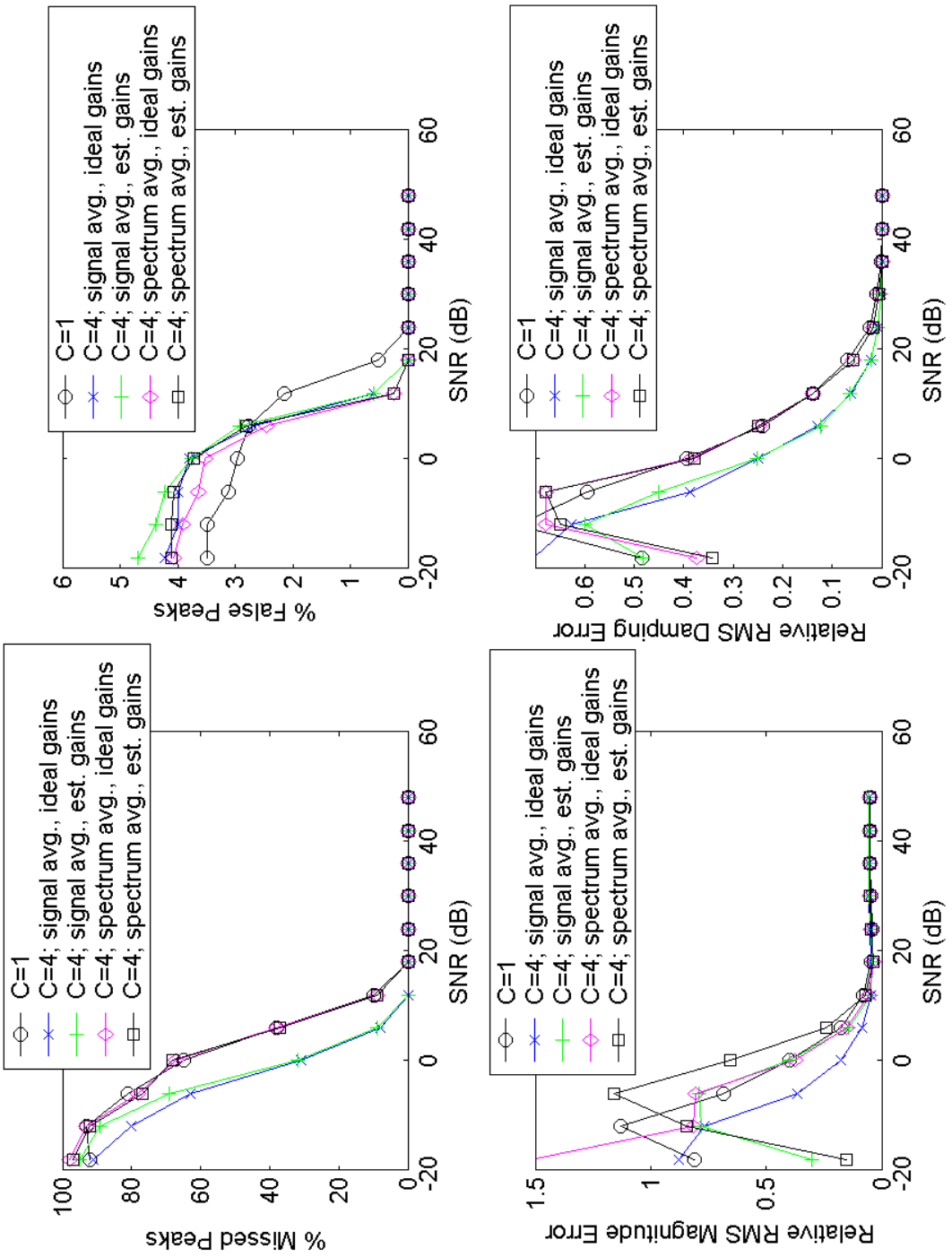

Fig. 60. Multiple-channel conventional 2D Capon (equal *SNR*'s – ideal gains and estimated gains).

6.5 Peak Identification Quality Measure

The simulations shown in Fig. 46-Fig. 60 demonstrate how certain parameter choices affect the figures of merit selected for study. In some cases, it is observed that parameter choices that yield a decrease in % missed peaks will cause an increase in % false peaks. To better understand this influence, and to measure the quality of a particular set of parameter choices a peak identification quality measure (PIQM) is defined:

$$PIQM = \left(\frac{(100\% - \%\, missed\ peaks)}{100\%} \right) \times \left(\frac{(100\% - \%\ false\ peaks)}{100\%} \right) \qquad (6.8)$$

This represents the fraction of known peaks detected multiplied by the fraction of known "non-peaks" not detected. The value of $PIQM$ ranges from 0 to 1, with $PIQM = 1$ indicating perfect peak identification with no missed peaks and no false peaks. Conversely, for the case of 100 % missed peaks and 100 % false peaks, $PIQM = 0$.

It is useful to plot $PIQM$ vs. SNR to compare the performance of various parameter choices in terms of their peak identification capabilities. It is also useful to compare the total area under the $PIQM$ curve to gauge the performance over the entire range of SNRs. A plot of $PIQM$ vs. SNR for selected simulations shown in Fig. 49-Fig. 51 is shown in Fig. 61.

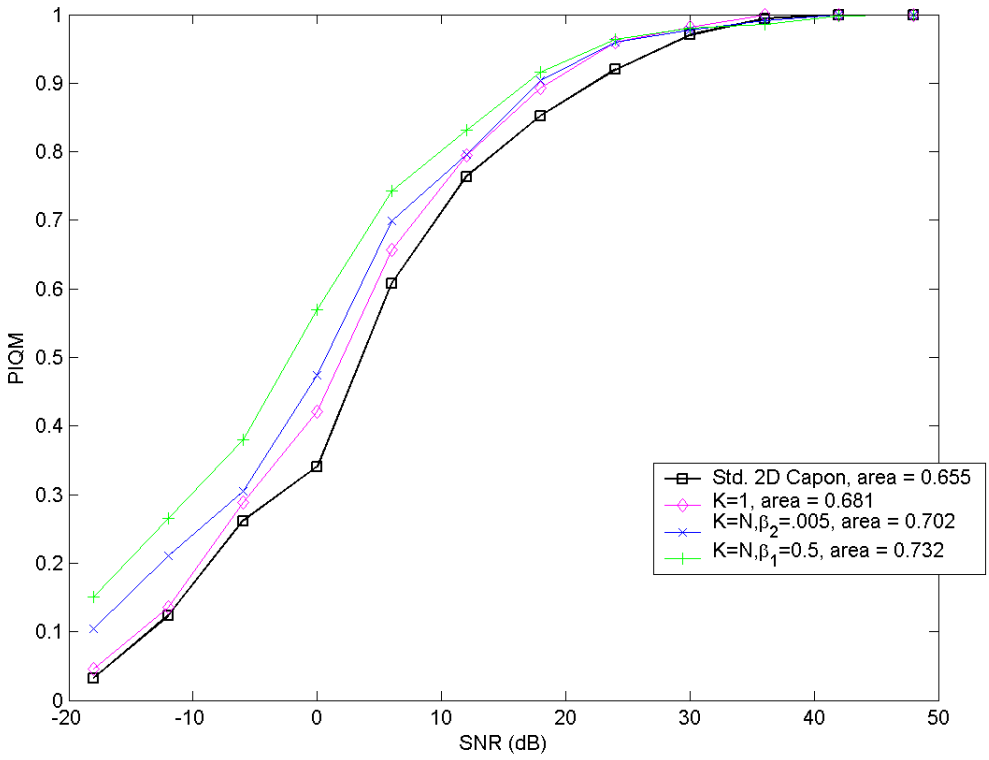

Fig. 61. *PIQM* plot vs. SNR for 2D Capon and selected weighted 2D Capon methods with $M = 128$.

6.6 Computation Time Results

We will now extend our evaluation to consider the timing performance of the new two-dimensional spectral estimation techniques, since this is also a consideration when deciding how to best implement two-dimensional spectral estimation for MRS. The goal of the timing simulations will be to determine the relative timing performance of the various weighted 2D Capon and the weighted 2D APES techniques as a function of filter size (M).

For the timing simulations a Dell Inspiron® 1100 laptop computer, with an Intel Celeron® CPU with a clock rate of 2 GHz, running Microsoft Windows XP® version 2002, configured with 384 Mbytes of RAM was used. The algorithms were implemented in MATAB® 6.5, release 13. We used the test signal defined by Eqn. (6.7) with $SNR = 18$ dB and 5 noise instances for filter sizes $M = \{32, 64, 128, 256\}$ and the algorithm parameters shown in Table 8. Results were averaged over 5 runs and then plotted in Fig. 62 and Fig. 63.

Algorithm	α_1	α_2	β_1	β_2
Weighted 2D Capon ($K = N$)	0	0	0	0
Weighted 2D Capon ($K = N$)	0	0	0.001	0
Weighted 2D Capon ($K = N$)	-0.5	0	0	0
Weighted 2D Capon ($K = N$)	-0.5	0	0.001	0
Weighted 2D Capon ($K = 1$)			0	0
Weighted 2D Capon ($K = 1$)			0.001	0
Combined weighted 2D APES / 2D Capon ($\gamma = 0.5$)			0	0
Combined weighted 2D APES / 2D Capon ($\gamma = 0.5$)			0.001	0
Combined weighted 2D APES / 2D Capon ($\gamma = 0.5$)			-0.5	0

Table 8. Parameters for timing simulation.

It should be noted that the particular choice of parameter values does not influence computation time (except in those cases where one of the values is –0.5). Rather, the results depend on whether the parameters are constant or functions of σ (see Table 1 in Chapter 4). It is also noted that in Fig. 62 the case $K = N$, $\alpha \neq -\sigma/2$,

$\beta = constant$ corresponds, in terms of computation time, to conventional 2D Capon.

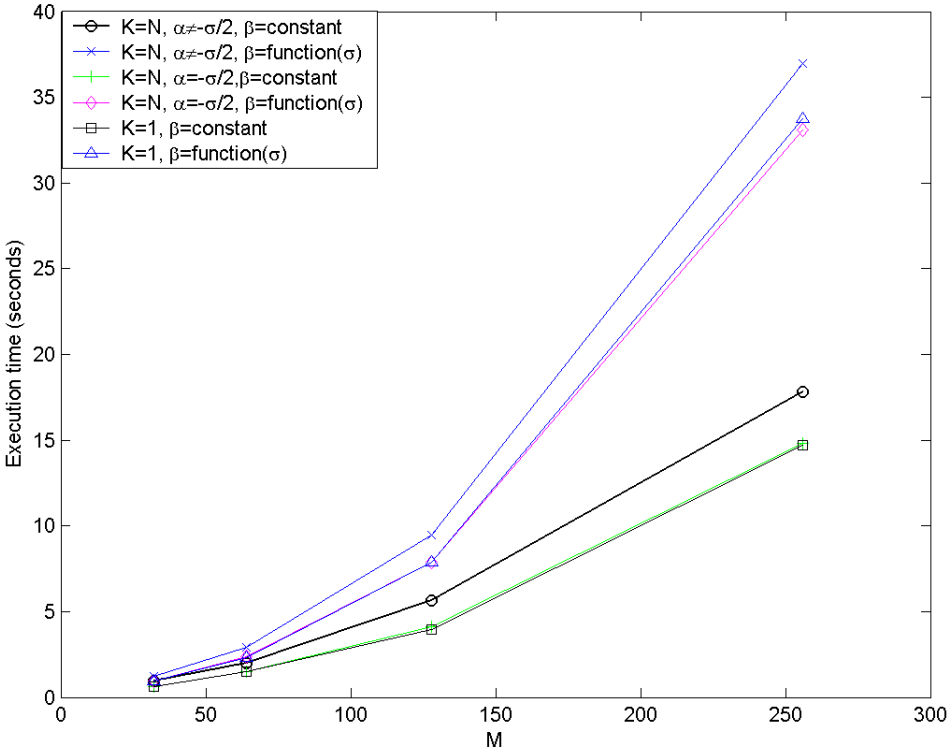

Fig. 62. Weighted 2D Capon timing vs. filter length, M .

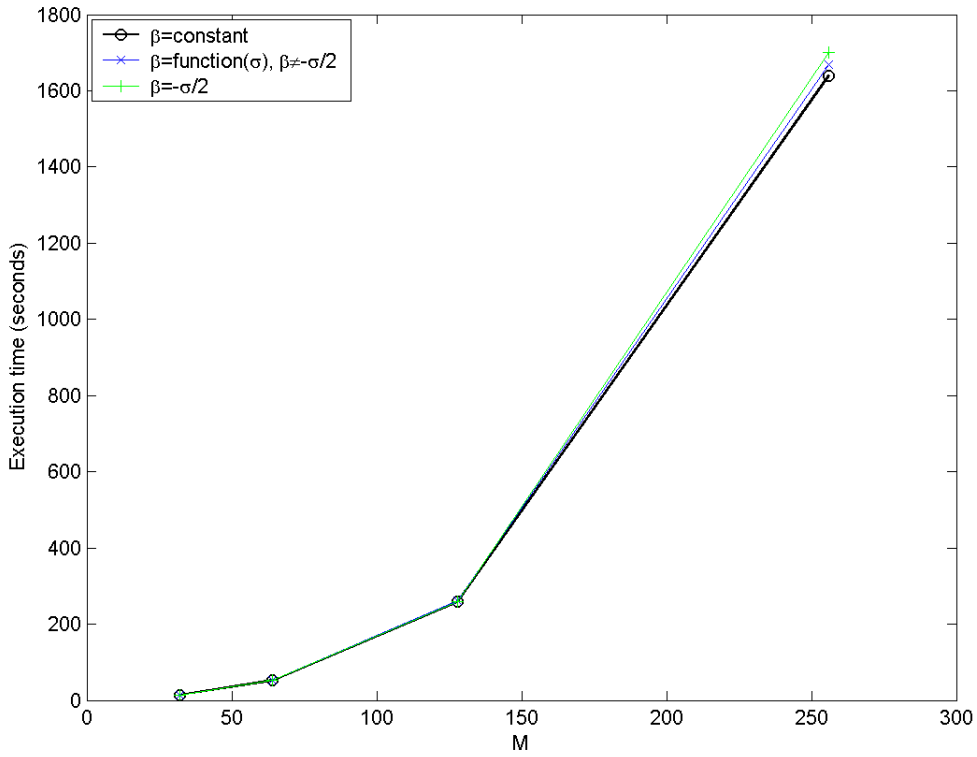

Fig. 63. Combined weighted 2D APES / 2D Capon timing vs. filter length, M .

In analyzing the data for the weighted 2D Capon method, it has been determined that the two new techniques designed with computational time as a consideration ($K=1$ and $K=N, \alpha=-\sigma/2$) produce a reduction in computation time, relative to conventional 2D Capon, ranging from about 35% for $M=32$ to about 17% for $M=256$. On the other hand, the remaining three new techniques require an increase in computation time ranging from about 0-30% for $M=32$ to about 86-107% for $M=256$. For the combined weighted 2D APES / 2D Capon method (including conventional 2D APES) all techniques require roughly the same computation time.

6.7 Discussion of Results

The simulations just completed provide some truly interesting findings. The signals used for the simulations, with appropriately scaled noise provide a controlled environment from which we can measure how well the new proposed algorithms perform. We wish to infer from the results of the simulations how these algorithms might perform on MRS data sets; therefore the analysis of these simulations holds great significance. We will now briefly review the simulations represented by Fig. 46 - Fig. 63.

6.7.1 Observations from Single-Channel Simulations

Certain conclusions can be drawn from the simulations. These relate to the various performance trade-offs with respect to the figures of merit considered (see Eqn. (6.1) – Eqn.(6.4)): % missed peaks, % false peaks, relative RMS magnitude error, relative RMS damping error, and execution time.

Fig. 46, Fig. 47, Fig. 48, and Fig. 52 indicate the effect of filter length, M , on performance:

- Fig. 46. Weighted 2D Capon $(K = N)$ for various M's (conventional 2D Capon).
- Fig. 47. Weighted 2D Capon $\left(K = N,\ \alpha = -\sigma/2\right)$ for various M's.
- Fig. 48. Weighted 2D Capon $(K = 1)$ for various M's.
- Fig. 52. Weighted 2D APES for various M's (conventional 2D APES).

For all four of these methods, in general:

1. As M increases, % missed peaks and relative RMS damping error decrease.

2. As M increases, % false peaks and relative RMS magnitude error first decrease and then increase, suggesting an *optimal* mid-range value for M with respect to these measures.

It is also observed that when using weighted 2D Capon $(K = N)$, $\alpha = -\sigma/2, \beta = 0$, false peaks are virtually completely eliminated. In addition, for conventional 2D APES, other studies [2] have shown that relative RMS magnitude error may be significantly reduced if better estimates for damping are available from some other method, i.e., 2D Capon.

Fig. 53 compares these four methods for $M = 128$. The results here should be considered in conjunction with the computation time results.

The three 2D Capon methods generally perform better than the 2D APES method, and require significantly less computation time. However, in light of the previous comment regarding the use of 2D APES along with better damping estimates, 2D APES may be appropriate if reduced relative RMS magnitude error is critical. The two new weighted 2D Capon methods require less computation time than conventional 2D Capon.

In addition these methods produce considerably fewer false peaks. With regard to the other measures they perform either slightly better, slightly worse, or comparable to the 2D Capon method.

Fig. 49 and Fig. 50 indicate the effect of the parameters of α and β on the performance of the weighted 2D Capon $(K = N)$ method for $M = 128$. Fig. 49 shows that certain parameter choices can result in a significant reduction in % missed peaks, but at the expense of increasing % false peaks and relative RMS magnitude error. Fig. 50 shows that certain parameter choices can lead to one or more of the following: reduced % false peaks, reduced relative RMS magnitude error for low *SNR*s, reduced relative RMS magnitude error for high SNRs, and reduced relative RMS damping error for low *SNR*s.

Fig. 51 indicates the effect of the parameter β on the performance of the weighted 2D Capon $(K = 1)$ method for $M = 128$. Here, it is shown that certain parameter choices can result in significant reduction in % missed peaks, but at the expense of increased % false peaks and greater relative RMS magnitude error.

Fig. 54 indicates the effect of the parameter γ on the performance of the combined weighted 2D APES / 2D Capon method for $B = 0$ and $M = 128$. As expected, as γ varies from 0 to 1, the performance measures generally show a gradual transition from those of the conventional 2D Capon method to those of the conventional 2D APES

method. However, with respect to relative RMS magnitude error at high SNR's, intermediate γ's outperform both $\gamma = 0$ and $\gamma = 1$, suggesting an *optimal* mid-range value for γ in this regard.

Fig. 55 indicates the effect of parameter β on the performance of the weighted 2D APES method for $M = 128$. It is noted that certain parameter choices can result in a significant reduction in % false peaks and/or relative RMS magnitude error.

Fig. 61 shows that for the weighted 2D Capon method with *K*=*N* certain parameter combinations lead to improved peak identification as measured by the *PIQM*. In addition, weighted 2D Capon with K=1 outperforms conventional 2D Capon, again as measured by the *PIQM*.

In summary, it is seen that no one method is best in all respects. The choice of which method to use in a specific application is thus dictated by the most critical aspect of that application, for example, the ability to find as many true peaks as possible, the accuracy of the magnitude estimates of those peaks, or computation time.

6.7.2 Observations from Multiple-Channel Simulations

With respect to computation time, signal averaging requires essentially the same amount of time as would be required by the same 2D estimation method for one channel. On the other hand, spectrum averaging requires C times as much computation time as signal averaging.

Fig. 56 (equal noise variances on all channels) shows that signal averaging outperforms single-channel processing with respect to all measures, as would be expected. On the other hand, spectrum averaging is no better than single-channel processing, with one exception: with respect to % false peaks, spectrum averaging outperforms both signal averaging and single-channel processing. Thus, because of its poor performance and greater computation time, spectrum averaging would not be appropriate unless the elimination of false peaks was the most critical requirement of the given application.

Fig. 57 - Fig. 59 show that for unequal channel noise variances (with one "dominant" channel having a larger *SNR* than the rest) signal averaging outperforms spectrum averaging with respect to all measures. Furthermore, as the dominant channel becomes "more dominant," the performance of both signal averaging and spectrum averaging gradually converge to the performance of the single channel processing.

Fig. 60 shows that the use of estimated channel gains instead of ideal channel gains leads to virtually no performance degradation, with one exception: for low SNR's (below about 10 dB) the relative RMS magnitude error is greater using estimated channel gains.

6.7.3 Observations from Computational Time Simulations

Computational time has been examined in Fig. 62 and Fig. 63. For all proposed methods examined, the filter size, M, is quite obviously the single most important

parameter affecting execution time. Another major point to note is that the weighted 2D Capon technique is much faster than the weighted 2D APES technique or the combined weighted 2D APES / 2D Capon technique. For the weighted 2D Capon technique, several computational efficiencies have been measured, and based on performance criteria established for the particular 2D spectral estimation data set, these techniques may provide acceptable performance with increased speed.

6.8 Summary

In this chapter we have performed a series of extensive simulations on test signals consisting of mixtures of damped sinusoids with different SNRs. In this way we were able to provide precisely controlled inputs, and evaluate the accuracy and efficiency of the new 2D spectral estimation techniques that have been proposed in Chapters 4 and 5. We have shown through these simulations that the proposed techniques provide improved performance when compared to standard 2D Capon and 2D APES under certain conditions. In the next chapter we will apply these new techniques to MRS signal processing.

Chapter 7

2D Spectral Estimation Methods Applied to MRS Data

Having completed a thorough introduction in Chapter 4 of the weighted 2D Capon, weighted 2D APES, and combined weighted 2D APES / 2D Capon methods; and the introduction in Chapter 5 of the new multiple-channel 2D spectral estimation techniques utilizing spectrum averaging and signal averaging; followed by an extensive set of simulations in Chapter 6 demonstrating the effectiveness of these new methods, we now apply these techniques to MRS signal processing. In this chapter we provide several examples of the new processing techniques applied to MRS data acquired from phantoms containing solutions of known concentrations of metabolites, and to a limited set of *in vivo* data sets acquired from human volunteers. Through these examples, we provide an evaluation of the performance of various 2D spectral estimation techniques from a variety of viewpoints.

MRS provides a noninvasive means of determining chemical information from a region of interest, often located in the human brain. It has continued to grow as a successful clinical application[78]-[85] and has become vitally important for the diagnosis, detection and effective management of a number of diseases. The combination of MRI and MRS has almost eliminated the need to perform exploratory surgery as a means of clinical diagnosis. The techniques proposed in Chapter 4 and Chapter 5 hold great promise to further improve MRS as a clinical application.

The reproducibility and accuracy of clinical MRS examinations has been the subject of several studies[86]-[88]. One parameter of great interest to the clinician is T_2^*, the effective decay rate of transverse magnetization. Changes in T_2^* provide useful diagnostic information[89]-[92] for a number of diseases and also provide some indication of functional activity.

7.1 Conventional 2D Capon and 2D APES Methods Applied to MRS Phantom Data

Conventional 2D Capon and conventional 2D APES provide improved spectral estimates compared to those provided from conventional MRS absorption spectra via Fourier transformation. Spectral peaks that are close in frequency are more easily separated using conventional 2D Capon and conventional 2D APES. The accuracy of these spectral estimates is improved as well since, unlike one-dimensional methods based on the Fourier transform, true amplitudes are not masked by the effect of damping.

The damping information provided by conventional 2D Capon and conventional 2D APES may be of significant clinical utility. Hurd, et al., have studied T_2^* characteristics for a number of metabolites in the human brain[86]. The spectral estimates provided by conventional 2D Capon and conventional 2D APES provide a means of estimating T_2^*, the effective decay rate of transverse magnetization, for several metabolites of interest[74].

Empirical results have shown that conventional 2D Capon analysis provides a more accurate estimate of frequency and damping since it provides fine resolution to

resolve closely spaced peaks, while conventional 2D APES provides a more accurate estimate of the amplitude for a given frequency and damping factor [58]. Stoica and Sundin propose that using conventional 2D Capon first to find the peaks, then using conventional 2D APES to estimate the amplitudes provides benefits compared to using just one technique or the other[2]. In Sec. 4.7 we proposed a hybrid technique that combines the two methods.

Conventional 2D Capon and conventional 2D APES can be used to compute two-dimensional spectral estimates on the same raw data acquired to generate the conventional MRS absorption plot shown in Fig. 29. Fig. 65-Fig. 68 compare various plots of a conventional 2D Capon spectrum and a conventional 2D APES spectrum computed using $N = 1792$, $N_\omega = N/2$, $N_\sigma = 40$, *peak threshold* $= 0.0$ and $M = 256$ from the GE MRS phantom.

The MRS experiment for this comparison was performed using a GE 1.5T MR scanner and a single-channel head coil with PRESS on an 8 cc volume, with a TE of 35msec and a TR of 1500msec, 2 NEX, 8 reference frames and 16 water-suppressed (CHESS) frames. The total scan time was 1 minute and 18 seconds. The conventional MRS absorption spectrum from the GE MRS phantom for this scan is shown in Fig. 64.

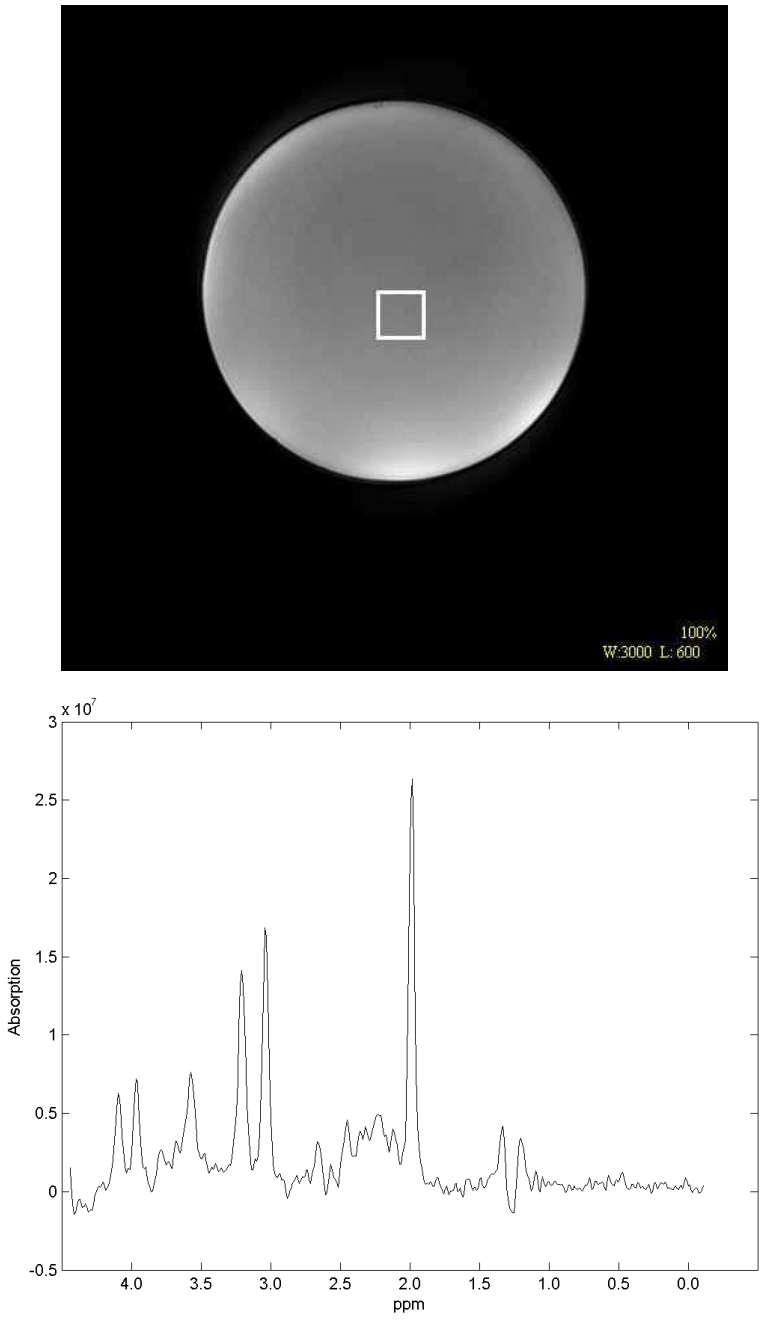

Fig. 64. MRS absorption spectrum from GE MRS phantom using single-channel head coil.

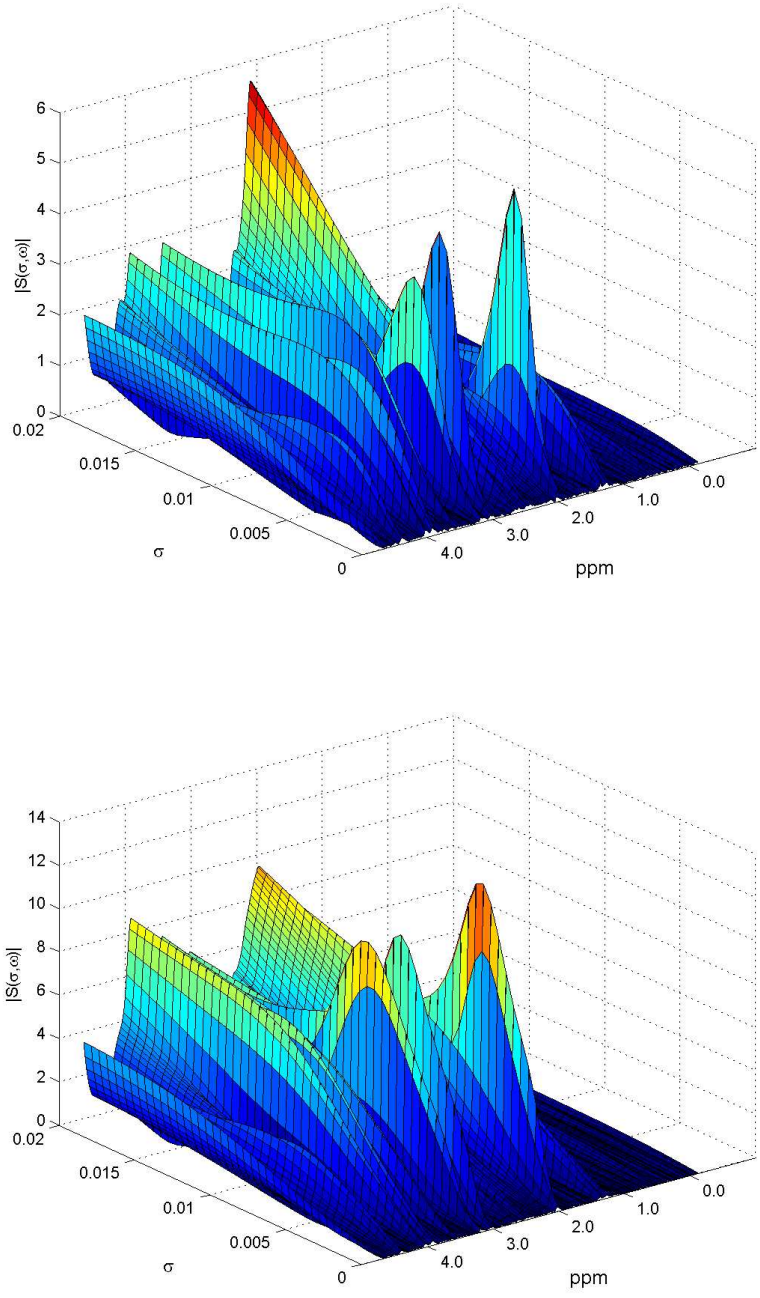

Fig. 65. Conventional 2D Capon (top) and conventional 2D APES (bottom) spectra obtained from the GE MRS phantom.

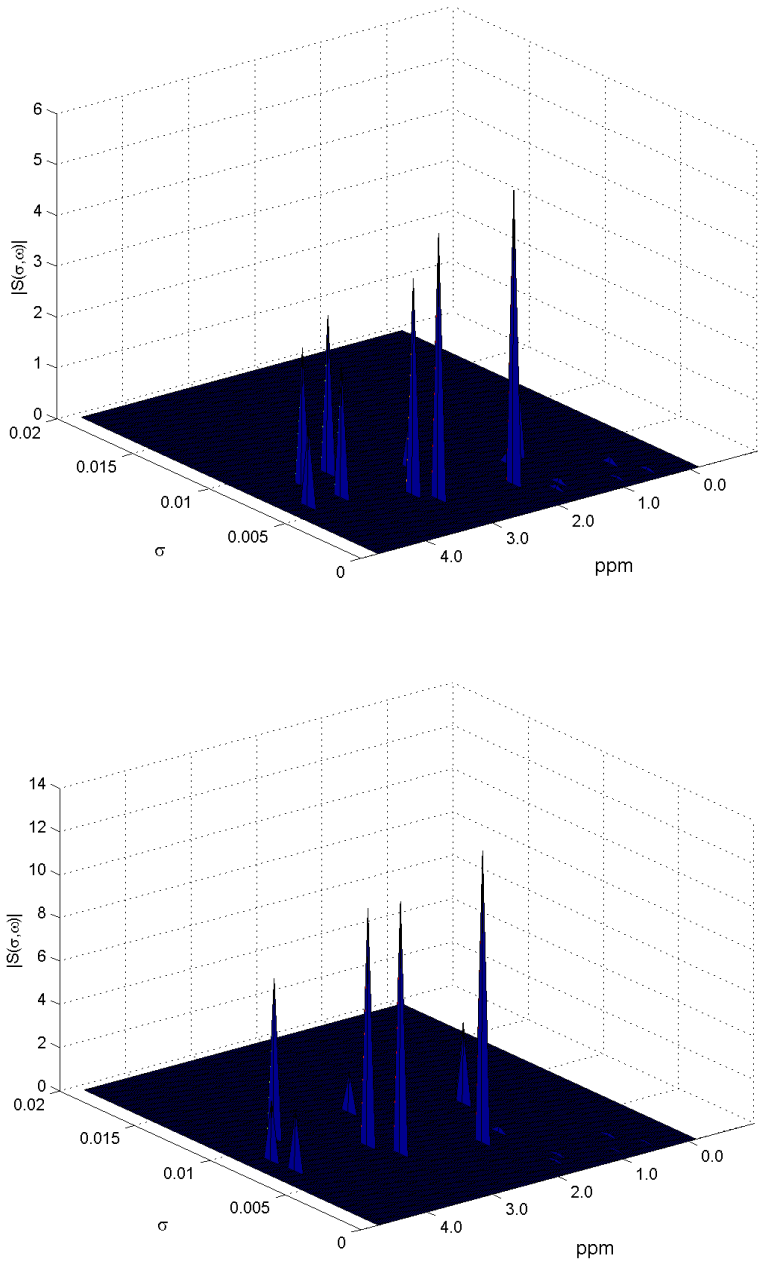

Fig. 66. Conventional 2D Capon (top) and conventional 2D APES (bottom) peak-enhanced spectra obtained from the GE MRS phantom.

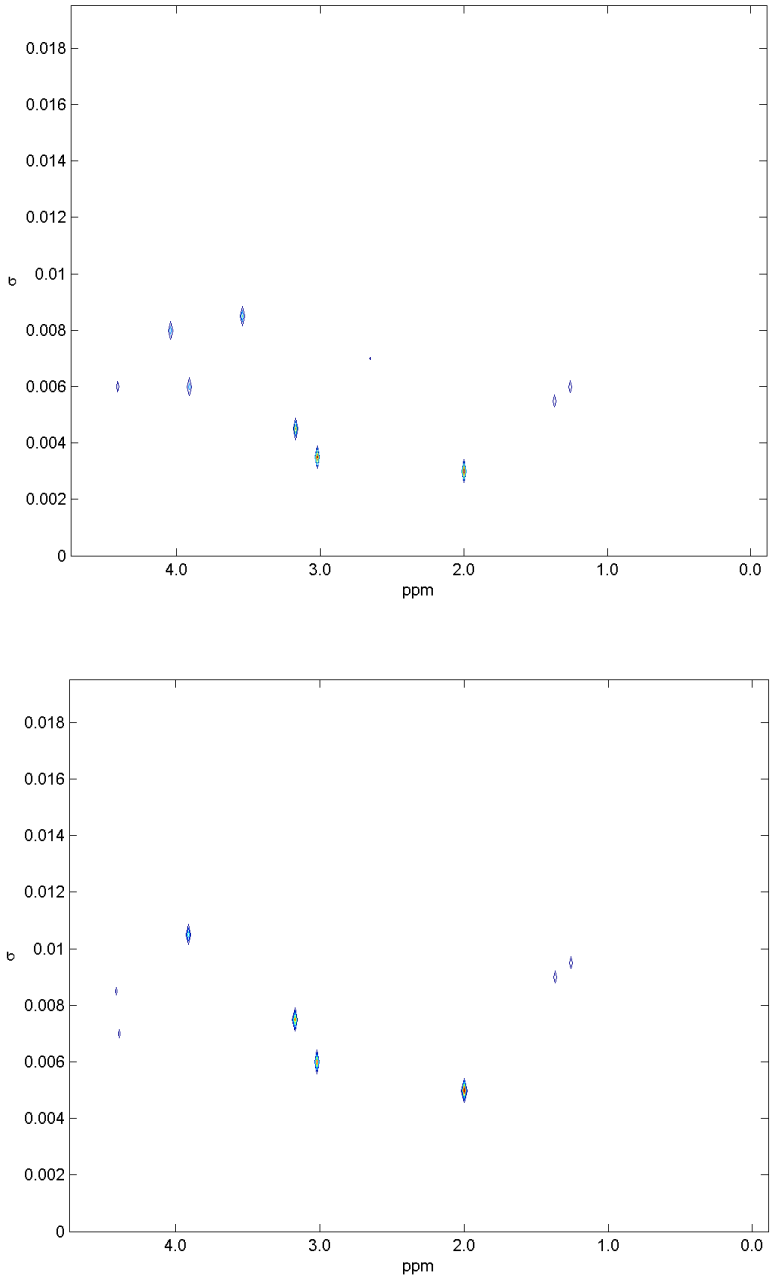

Fig. 67. Conventional 2D Capon (top) and conventional 2D APES (bottom) contour plots obtained from the GE MRS phantom.

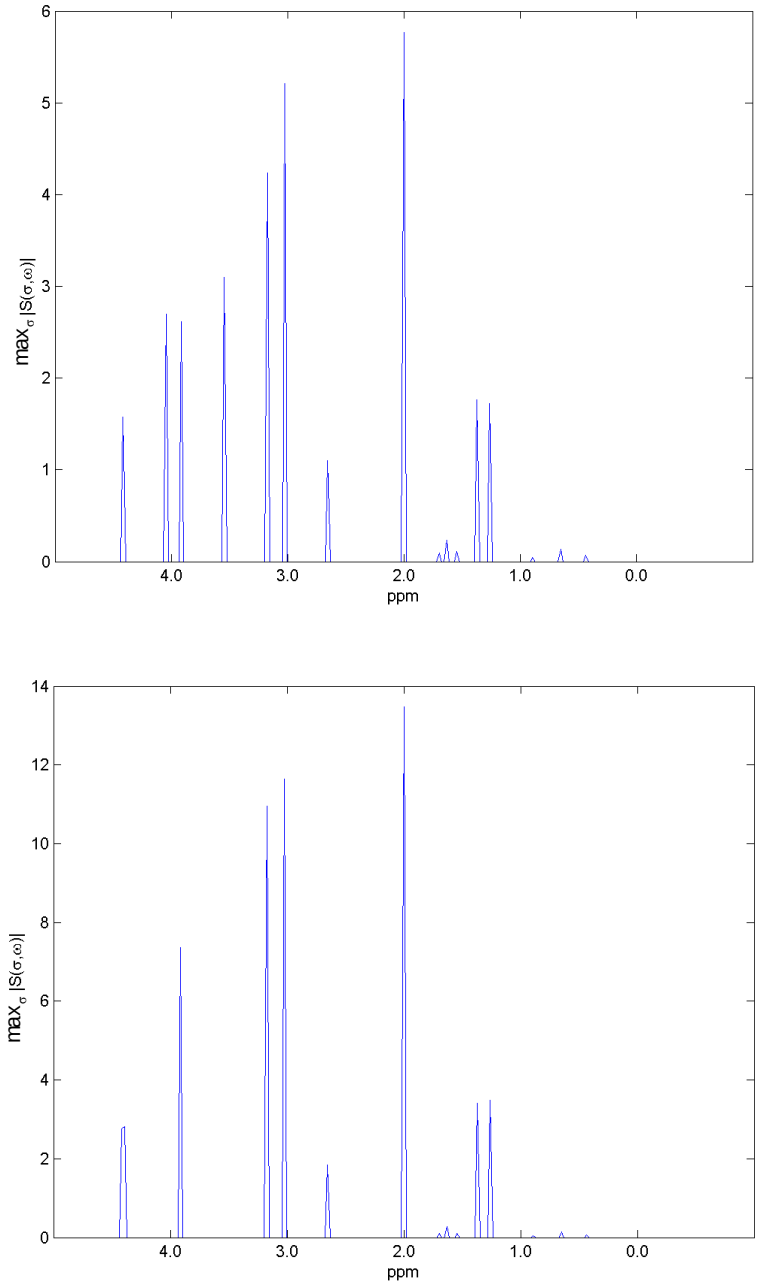

Fig. 68. Conventional 2D Capon (top) and conventional 2D APES (bottom) maximum peak projections obtained from the GE MRS phantom.

It can be seen clearly from Fig. 68 that the conventional 2D Capon method finds two peaks that are missed by the conventional 2D APES method. In addition, the 2D Capon method requires considerably less computation time.

7.2 Weighted 2D Capon Method

7.2.1 MRS Phantom Data

Simulations from Chapter 6 showed that weighted 2D Capon analysis performs better at finding peaks than standard 2D Capon analysis in certain cases. In this example, we use the phase-corrected, water-suppressed MRS signal with residual water removed obtained during an MRS experiment from the GE MRS phantom to compare weighted 2D Capon analysis with $\beta = -\sigma/2$ to weighted 2D Capon analysis with $\beta = 0.004$, both methods using $N = 1792$, $N_\omega = N/2$, $N_\sigma = 40$, $peak\ threshold = 0.0$ and $M = 256$. The MRS experiment for this comparison was the same as that shown in Fig. 64 and described in Sec. 7.1. Fig. 69-Fig. 72 show comparisons of how effective these two weighted 2D Capon techniques are on the data acquired from this scan.

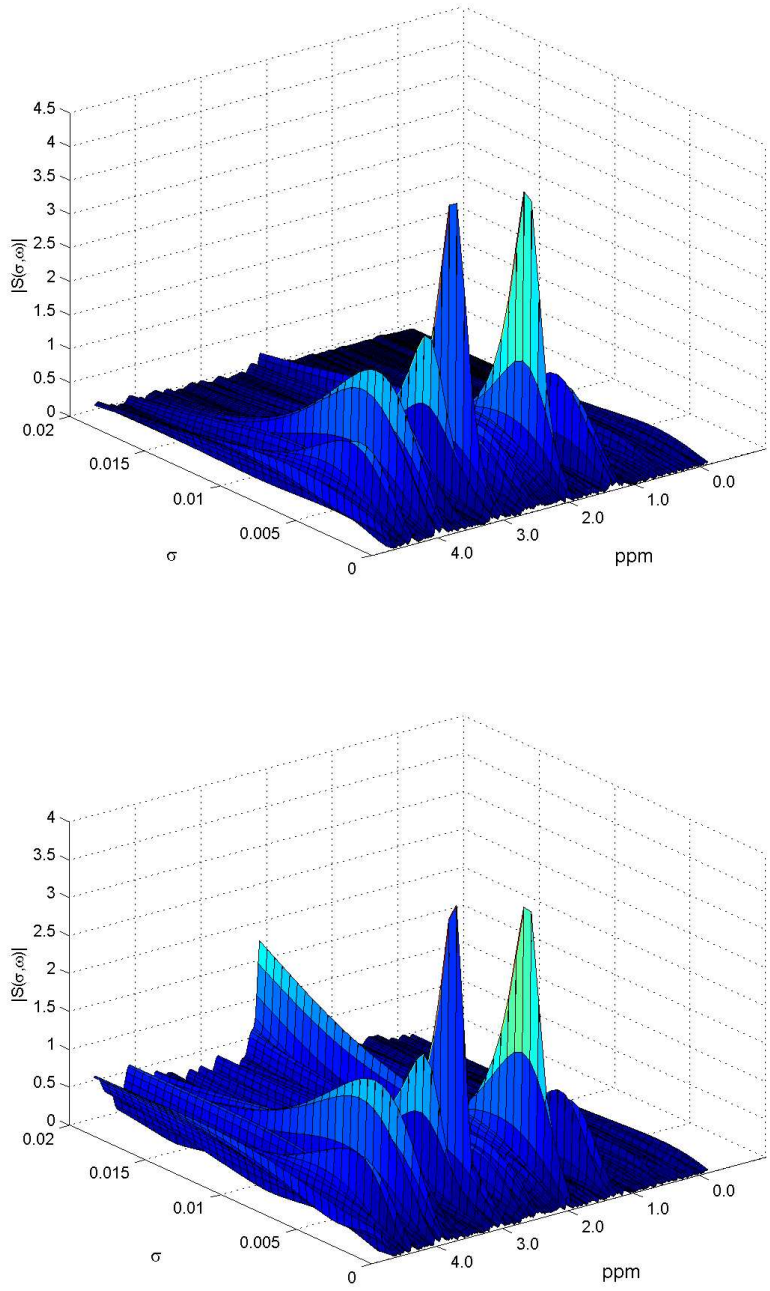

Fig. 69. Weighted 2D Capon with $\beta = -\sigma/2$ (top) and weighted 2D Capon with $\beta = 0.004$ (bottom) spectra obtained from the GE MRS phantom.

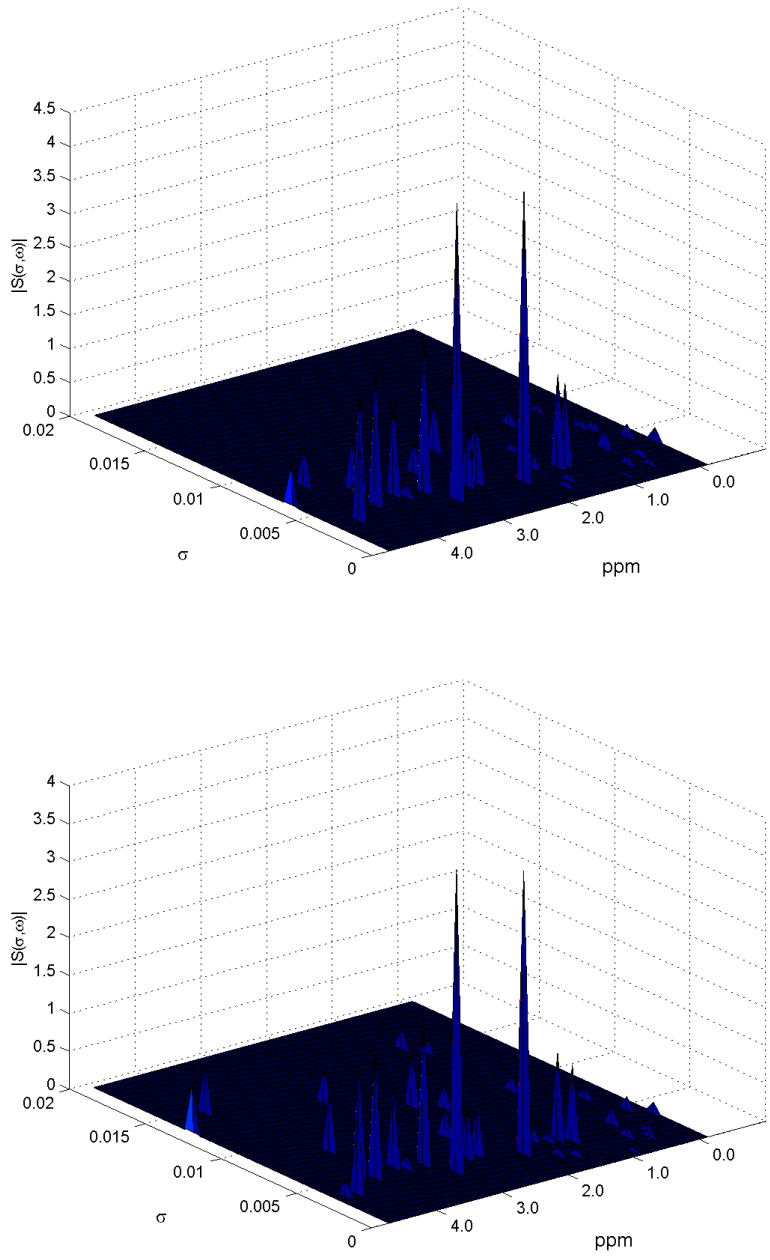

Fig. 70. Weighted 2D Capon with $\beta = -\sigma/2$ (top) and weighted 2D Capon with $\beta = 0.004$ (bottom) peak-enhanced spectra obtained from the GE MRS phantom.

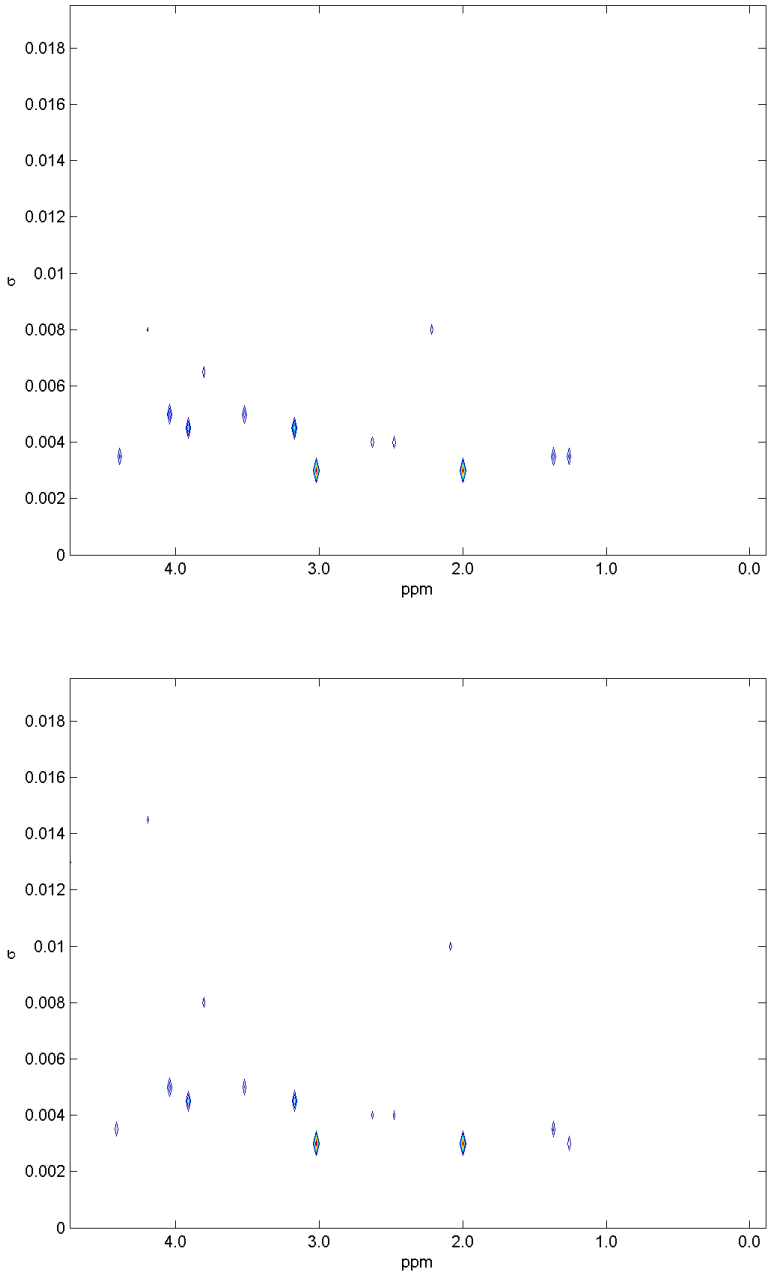

Fig. 71. Weighted 2D Capon with $\beta = -\sigma/2$ (top) and weighted 2D Capon with $\beta = 0.004$ (bottom) contour plots obtained from the GE MRS phantom.

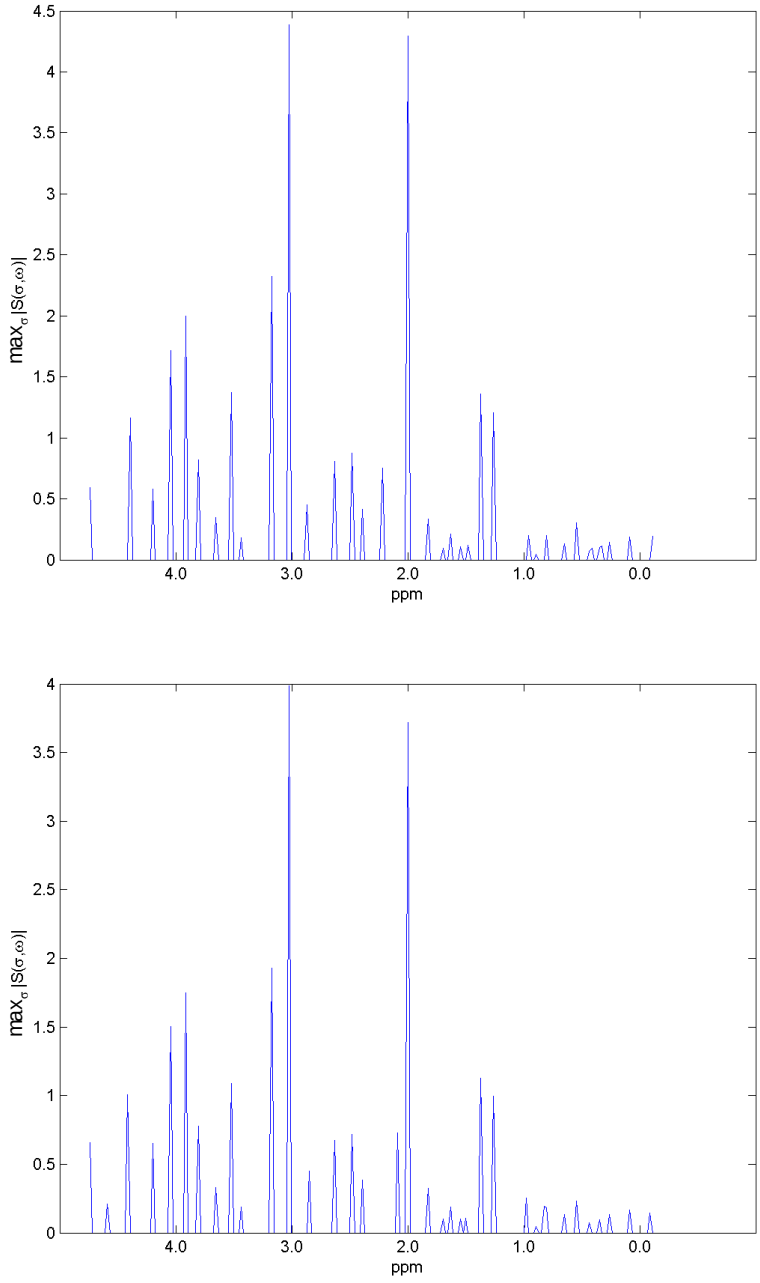

Fig. 72. Weighted 2D Capon with $\beta = -\sigma/2$ (top) and weighted 2D Capon with $\beta = 0.004$ (bottom) maximum peak projections obtained from the GE MRS phantom.

It is seen that in terms of identifying peaks both of these methods perform in a similar fashion. Using $\beta = -\sigma/2$ requires more than double the computation time than using $\beta = 0.004$, which requires the same computation time as conventional 2D Capon. Comparing the results of conventional 2D Capon and 2D APES from Fig. 68 with the weighted 2D Capon methods shown in Fig. 72 it is seen that the two new weighted 2D Capon methods significantly outperform conventional 2D Capon and 2D APES in terms of identifying peaks.

7.2.2 *In Vivo* Data

We now compare weighted 2D Capon analysis with $\beta = -\sigma/2$ to conventional 2D Capon analysis for an *in vivo* MRS data set obtained from the brain of a healthy human volunteer. The MRS experiment for this comparison was performed using a GE 1.5T MR scanner and a single-channel head coil with PRESS on an 8 cc volume, with a TE of 35msec and a TR of 1500msec, 2 NEX, 8 reference frames and 64 water-suppressed (CHESS) frames. The total scan time was 3 minutes and 42 seconds. For the comparison, we selected $N = 1792$, $N_\omega = N/2$, $N_\sigma = 40$, $peak\ threshold = 0.0$ and $M = 256$ for both the weighted 2D Capon analysis and conventional 2D Capon analysis. The data set from which spectral estimates were created was the phase-corrected, water-suppressed MRS signal with residual water removed. The conventional MRS absorption spectrum from the brain of a human volunteer is shown in Fig. 73. Fig. 74-Fig. 77 show

comparisons of how effective these two weighted 2D Capon techniques are on the data acquired from this scan.

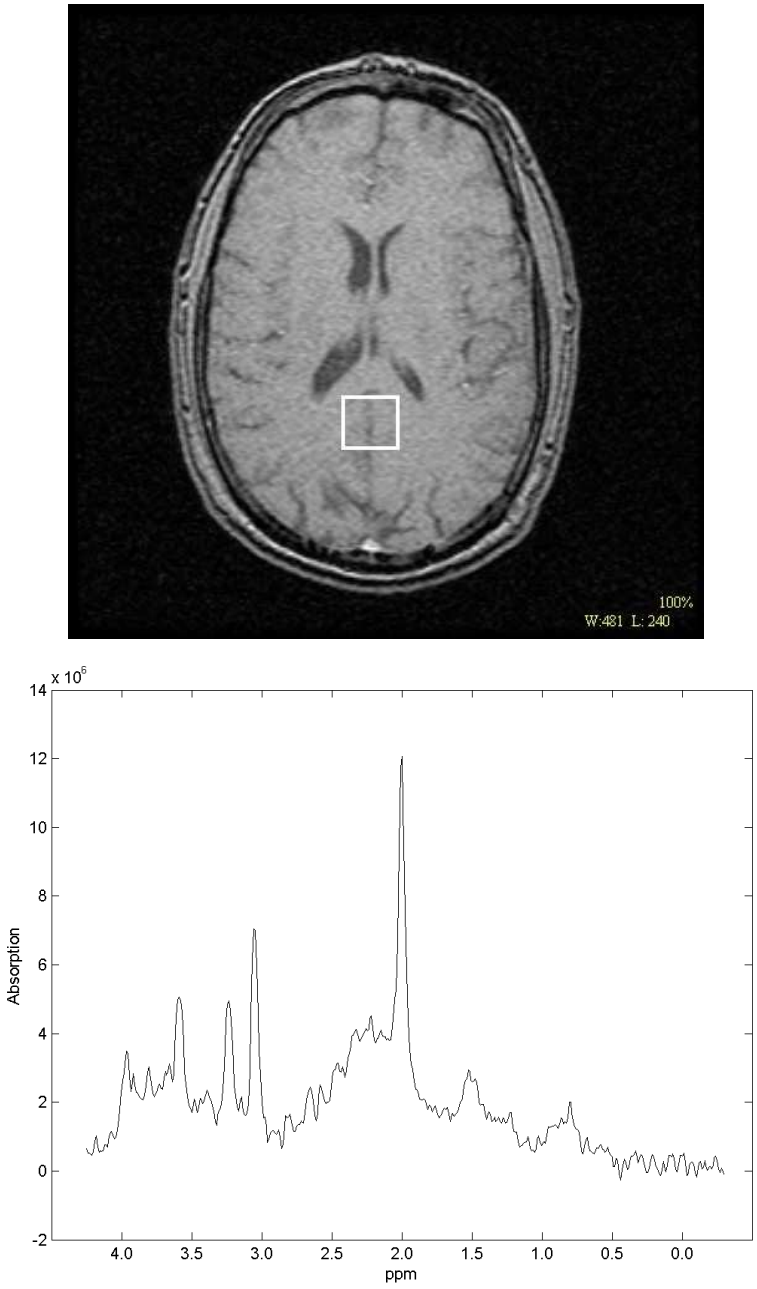

Fig. 73. MRS absorption spectrum from human volunteer using single-channel head coil.

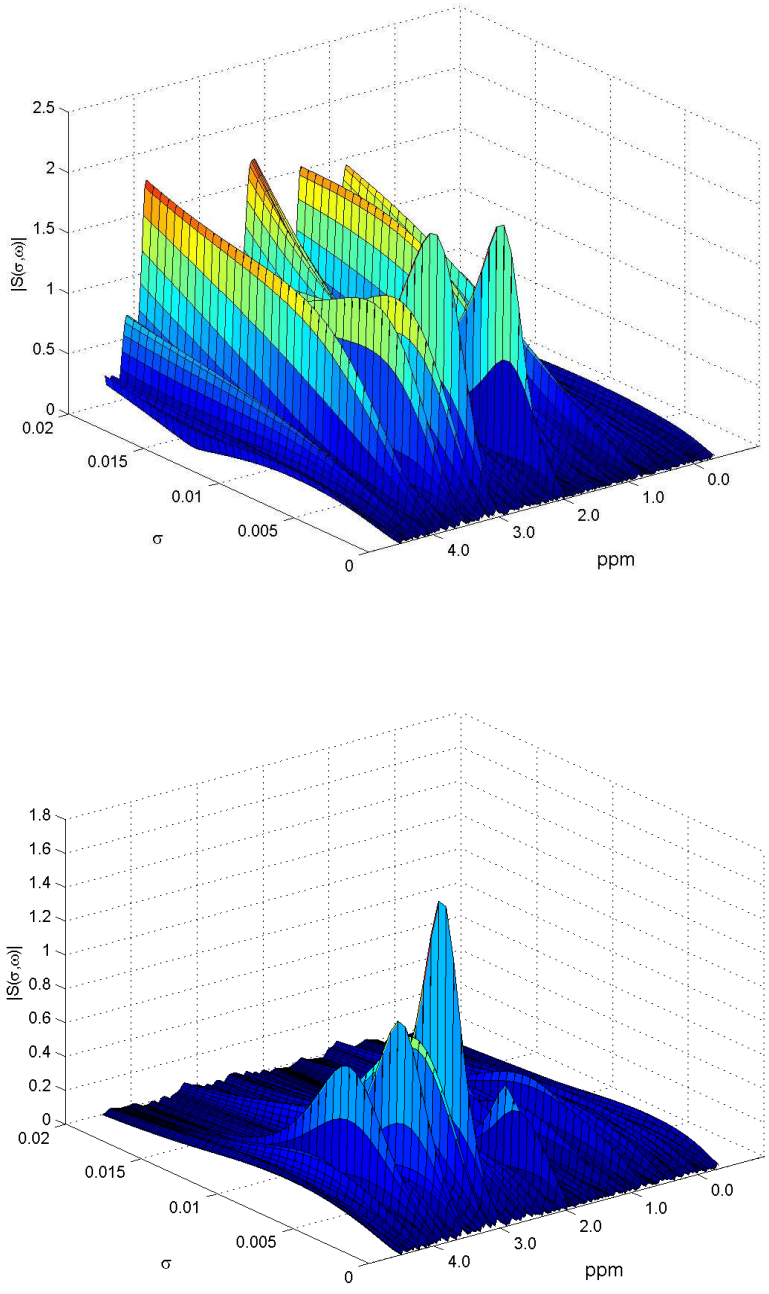

Fig. 74. Conventional 2D Capon (top) and weighted 2D Capon with $\beta = -\sigma/2$ (bottom) spectra obtained from the brain of a human volunteer.

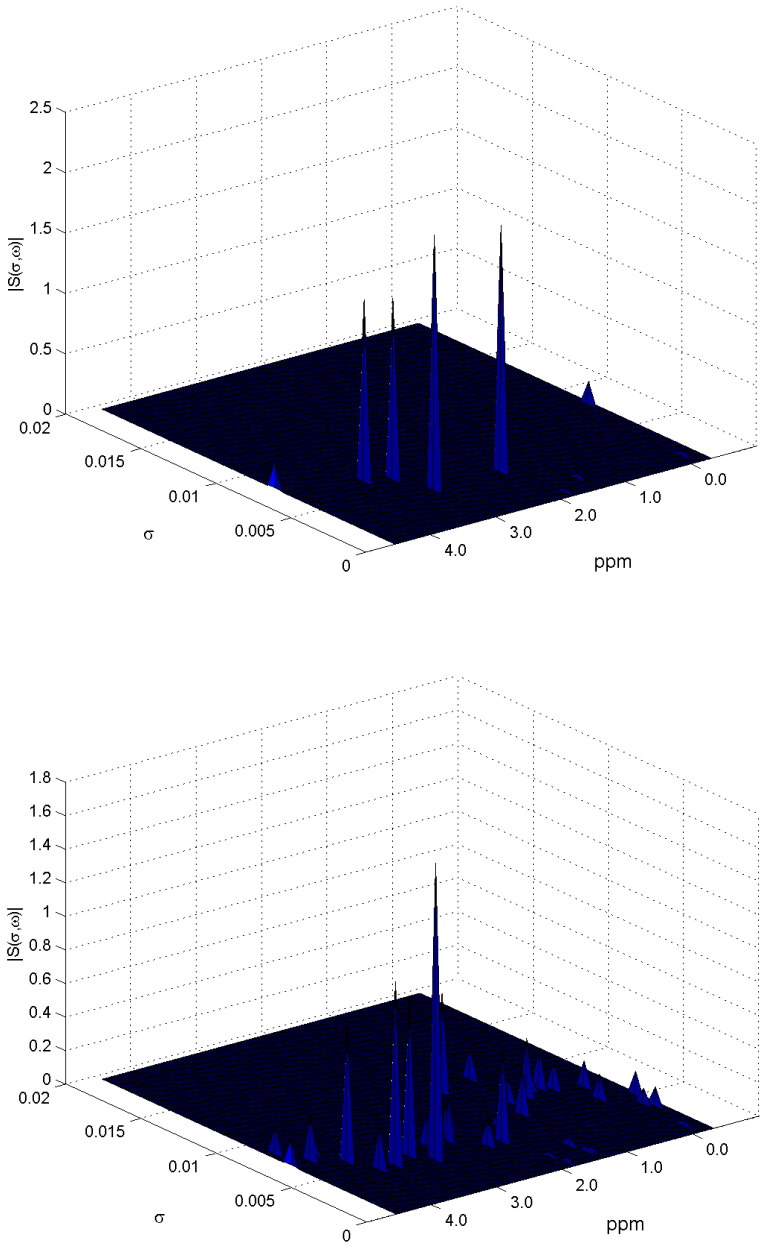

Fig. 75. Conventional 2D Capon (top) and weighted 2D Capon with $\beta = -\sigma/2$ (bottom) peak-enhanced spectra obtained from the brain of a human volunteer.

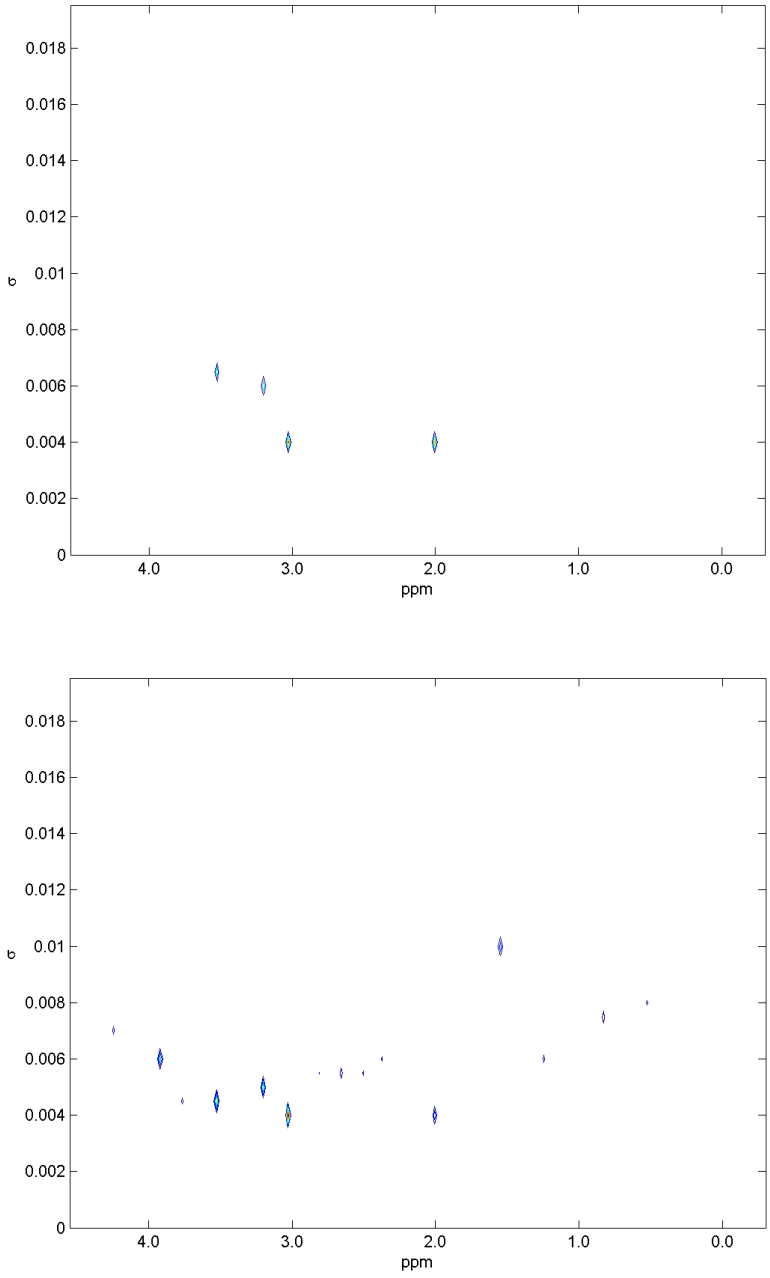

Fig. 76. Conventional 2D Capon (top) and weighted 2D Capon with $\beta = -\sigma/2$ (bottom) contour plots obtained from the brain of a human volunteer.

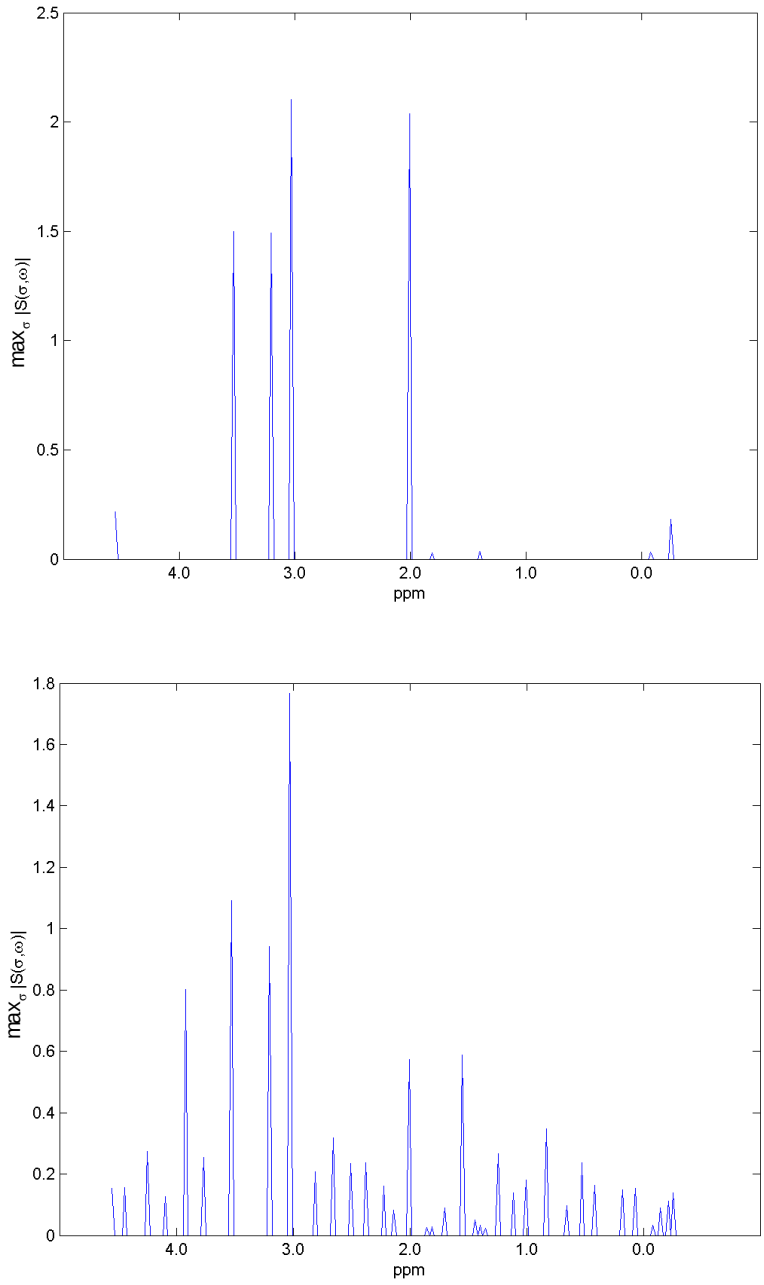

Fig. 77. Conventional 2D Capon (top) and weighted 2D Capon with $\beta = -\sigma/2$ (bottom) maximum peak projections obtained from the brain of a human volunteer.

Using weighted 2D Capon with $\beta = -\sigma/2$ finds many more peaks than conventional 2D Capon, however weighted 2D Capon with $\beta = -\sigma/2$ requires more than double the computation time.

7.2.3 Weighted 2D Capon Execution Time Considerations for MRS Phantom Data

One aspect of the clinical utility of MRS is related to the time it takes to acquire data from a patient, and then to properly analyze this data. Because the scan TR for most spectroscopy scans is long, anything that can potentially shorten scan time to reduce the length of the scan is of vital importance[93]. Additionally, any MRS processing time reductions may also provide clinical benefits, especially in the case of an emergency when the speed of obtaining an accurate diagnosis may provide a life-saving benefit.

To study the relationship between processing time and the ability to detect spectral peaks, we acquired MRS data from a GE MRS phantom using a single-channel head coil with PRESS on an 8 cc volume, with a TE of 35msec and a TR of 1500msec, 2 NEX, 8 reference frames and 16 water-suppressed (CHESS) frames. The total scan time was 1 minute and 18 seconds. The MRS experiment for this comparison was the same as that shown in Fig. 64 and described in Sec. 7.1. Using the phase-corrected water-suppressed MRS data with residual water removed we tested three techniques for filter length M varying from 100 to 400 in increments of 50. These were conventional 2D Capon, weighted 2D Capon with $K = 1, \beta = 0.008$, and weighted 2D Capon with

$K = N, \beta = 0.008$. In all cases we used $N = 1024, N_\omega = 2048, N_\sigma = 40$, and a *peak threshold* equal to 5% of the maximum peak detected.

For the timing measurements, we used a personal computer manufactured by Milwaukee PC (Milwaukee, WI, USA) configured with an Intel Pentium III® CPU with a clock rate of 1 GHz, running Microsoft Windows ME® version 4.90.3000, configured with 128 Mbytes of RAM. The algorithms were implemented in MATAB® 5.3, release 11.1. Fig. 78 summarizes the results, showing the number of known peaks[2] detected versus execution time. It is seen that both of the new weighted 2D Capon methods outperform the conventional 2D Capon method. Specifically, for a fixed execution time the new methods identify more known peaks. Alternatively, to find a fixed number of known peaks the new methods require less execution time.

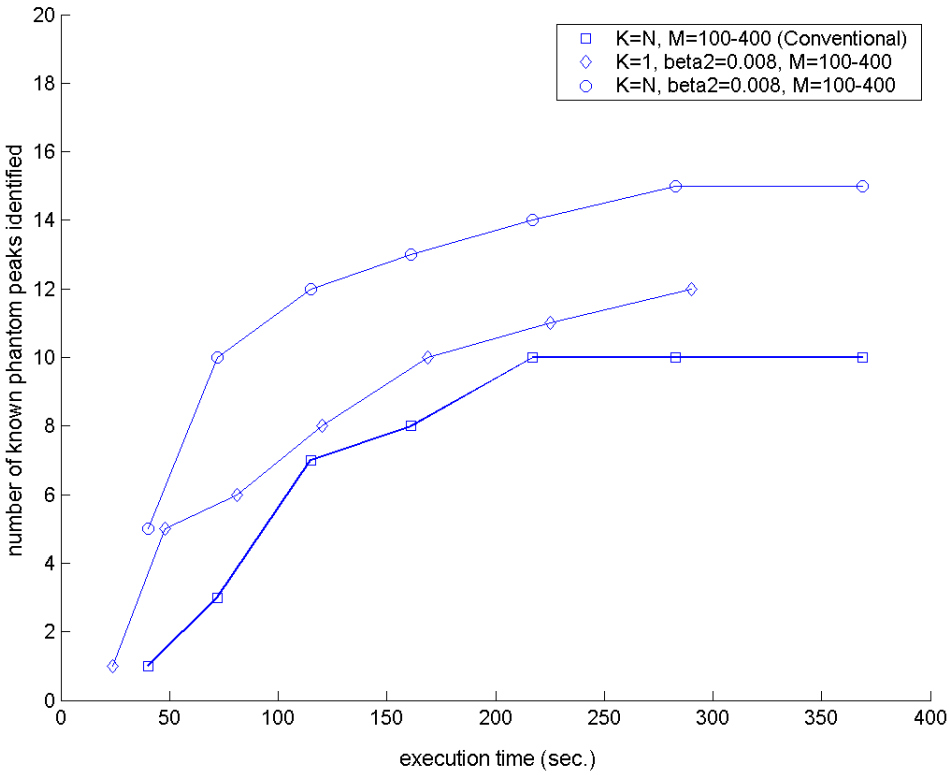

Fig. 78. Effectiveness of peak detection using weighted 2D Capon analysis.

7.2.4 Performing 2D Spectral Estimation on MRS Phantom Data with no Phase Correction

We will now examine how conventional 2D Capon analysis works on MRS data sets that have not been phase-corrected and for which residual water has not been removed. In this example, we acquired MRS data from the GE MRS phantom and used conventional 2D Capon analysis with $N = 1792$, $N_\omega = N/2$, $N_\sigma = 40$, $peak\ threshold = 0.0$ and $M = 256$. The MRS experiment for this comparison was

performed using a 1.5T GE MR scanner and a single-channel head coil with PRESS on an 8 cc volume, with a TE of 35msec and a TR of 1500msec, 2 NEX, 8 reference frames and 16 water-suppressed (CHESS) frames. The total scan time was 1 minute and 18 seconds. The MRS experiment for this comparison was the same as that shown in Fig. 64 and described in Sec. 7.1.

In Fig. 79 we compare the magnitude of Fourier transformed water-suppressed MRS data with and without phase-correction and residual water removal. It can be seen in this example that the presence of residual water interferes substantially with the ability to resolve peaks when using the Fourier transform. In Fig. 80-Fig. 83 we use the conventional 2D Capon techniques on the same data sets yielding the results shown in Fig. 79 to observe how effective the conventional 2D Capon method is in resolving metabolite peaks in the presence of residual water. These figures show that for metabolites of interest the conventional 2D Capon method is able to resolve metabolite peaks from the data set that has not been phase corrected and from which residual water has not been removed. However, on closer examination of Fig. 83 we note a slight shift in the frequency of these peaks for data without phase-correction and residual water removal.

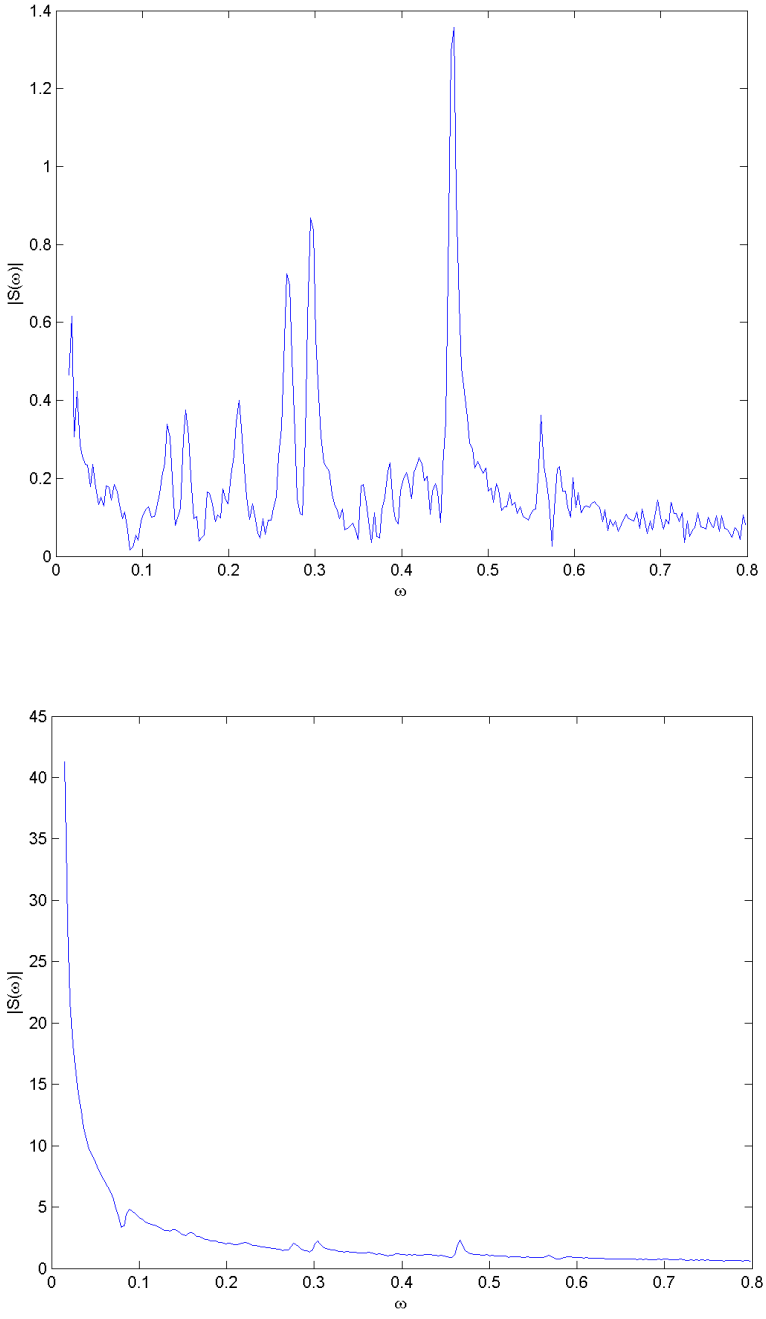

Fig. 79. Fourier transform of water-suppressed MRS data with (top) and without (bottom) phase-correction and residual water removal.

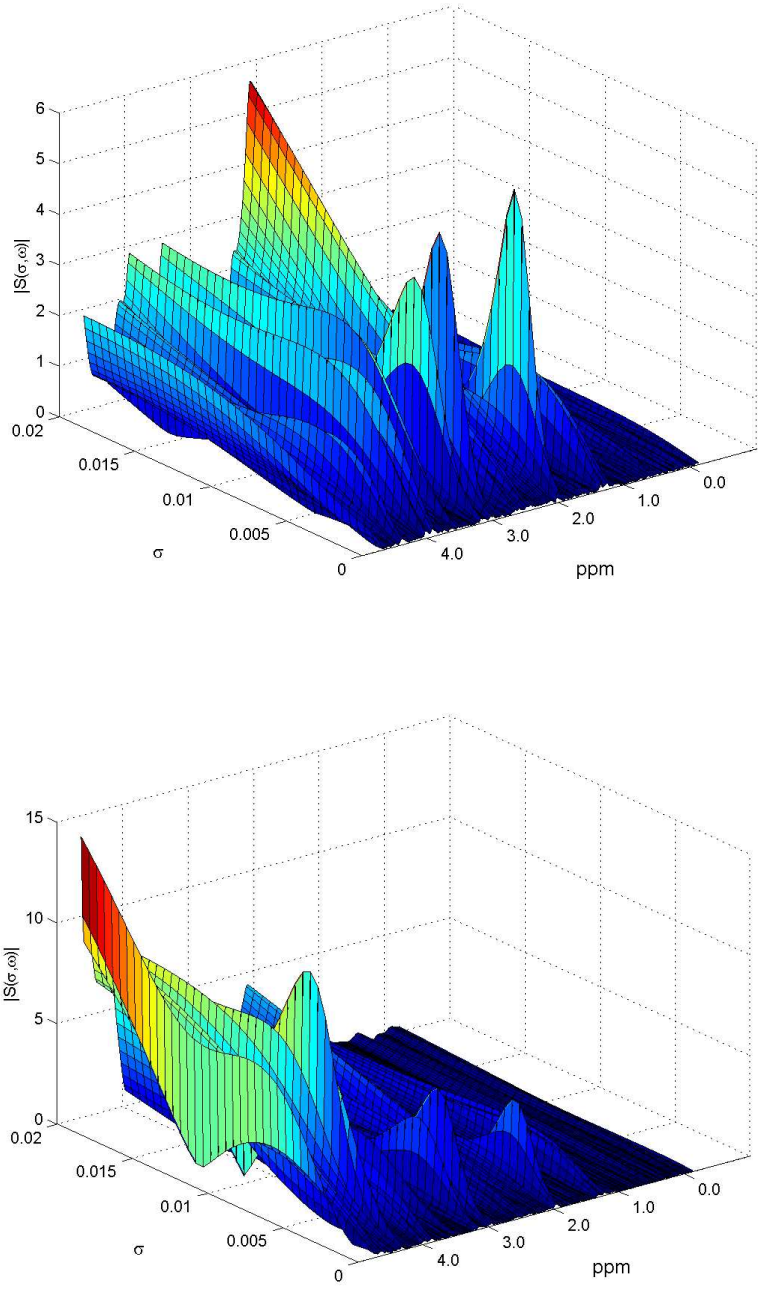

Fig. 80. Conventional 2D Capon analysis of water-suppressed MRS data with (top) and without (bottom) phase-correction and residual water removal.

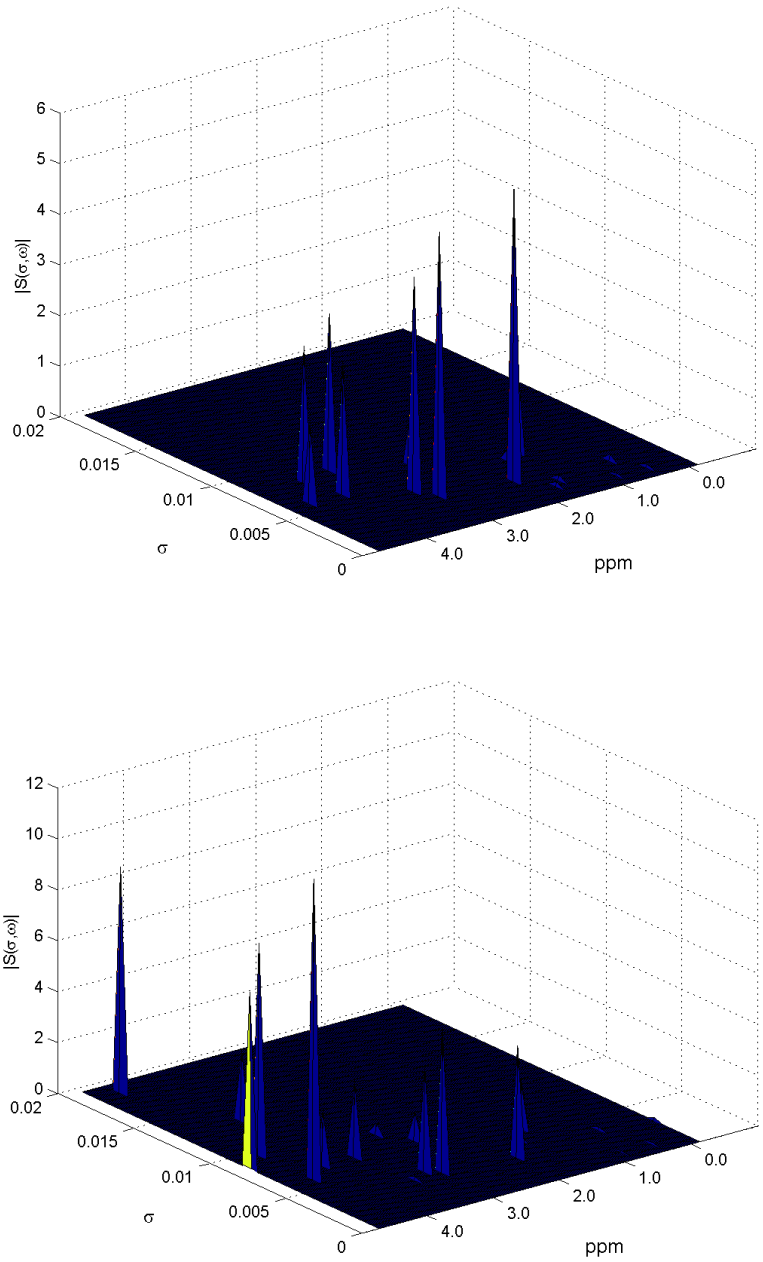

Fig. 81. Conventional 2D Capon peak-enhanced spectra of water-suppressed MRS data with (top) and without (bottom) phase-correction and residual water removal.

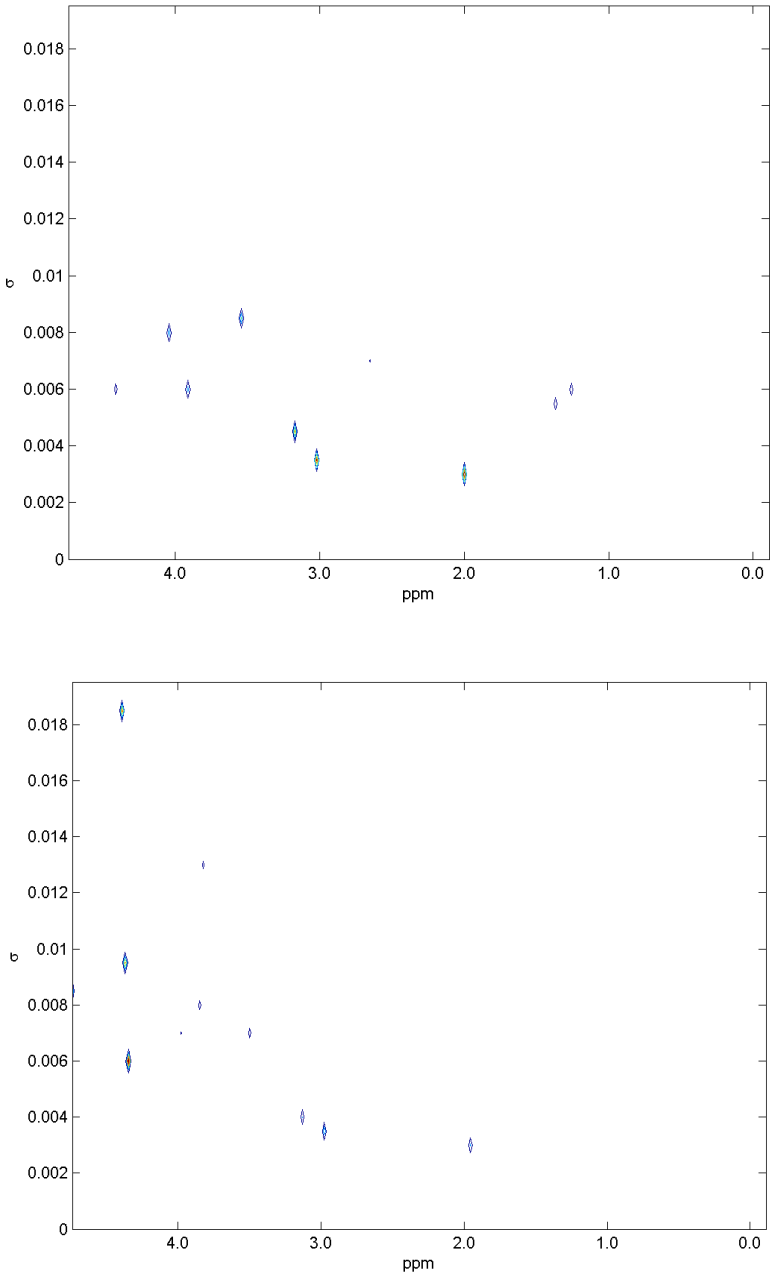

Fig. 82. Conventional 2D Capon contour plots with (top) and without (bottom) phase-correction and residual water removal.

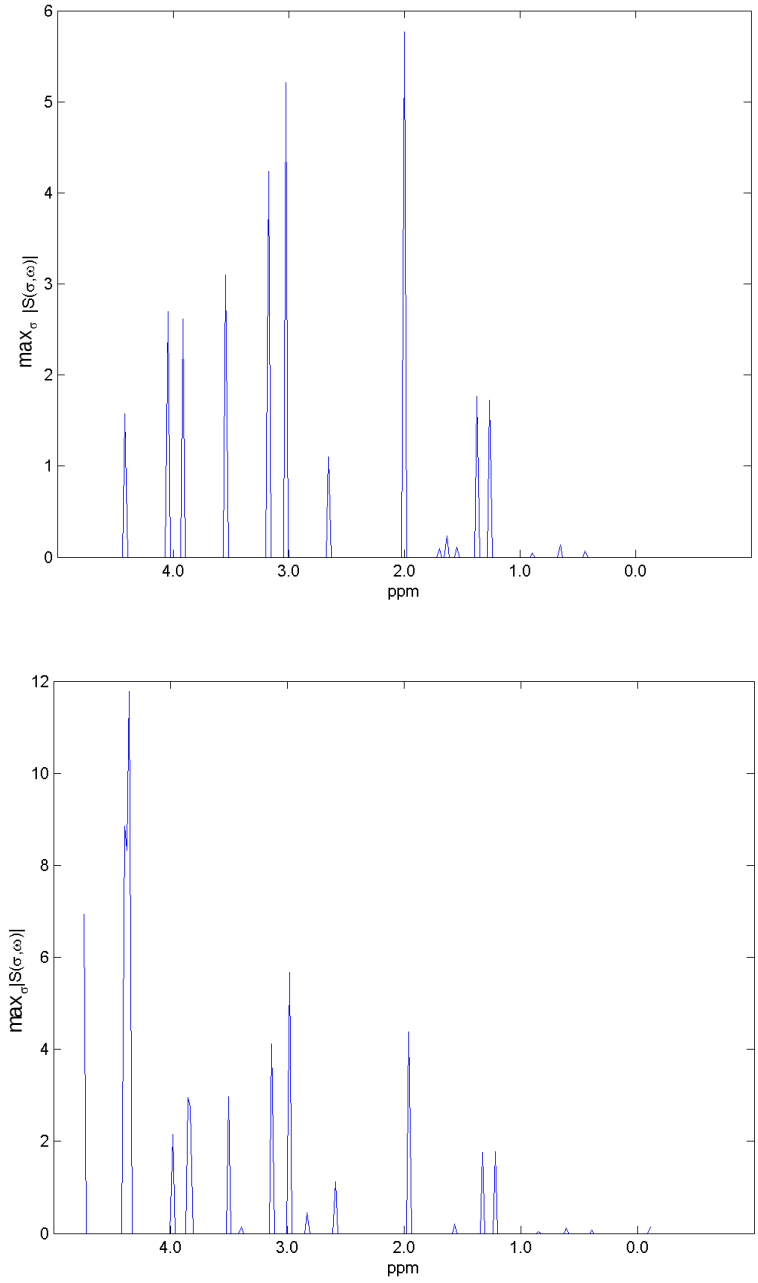

Fig. 83. Conventional 2D Capon maximum peak projections of water-suppressed MRS data with (top) and without (bottom) phase-correction and residual water removal.

7.3 Multiple-Channel 2D Spectral Estimation

We have proposed two main techniques for multiple-channel 2D spectrum estimation of MRS data: signal averaging and spectrum averaging. These techniques have been examined in Chapter 6 through extensive simulations (see Fig. 56 - Fig. 60). We would now like to compare and contrast how these two techniques work on data acquired from MRS experiments. In our first set of experiments, we study the techniques applied to data acquired from an MRS phantom. We then apply spectrum averaging and signal averaging to *in vivo* MRS data acquired from the brain of a human volunteer.

It should be pointed out that the 2D spectral estimation techniques used in this section used estimated channel gains (g's) from observed data as described in Sec. 5.4. Also, the noise variances (ρ's) as described in Sec. 5.5 were estimated from the last 248 samples of each observed signal where the length of the signal is: $N_f = 2048$.

7.3.1 Conventional 2D Capon Analysis of MRS Phantom Data

Simulations from Chapter 6 showed that two-dimensional spectral estimation techniques based on spectrum averaging and signal averaging are both effective although signal averaging usually appeared to be the method of choice. We now compare spectrum averaging and signal averaging techniques used on MRS data acquired using an eight-channel domed head coil, manufactured by MRI Devices, Inc. (Waukesha, WI, USA) from a GE MRS phantom. The scan was performed using a GE 1.5T MR scanner and a PRESS sequence on an 8 cc volume, with a TE of 35msec and a TR of 1500msec, 2

NEX, 8 reference frames and 16 water-suppressed (CHESS) frames. The total scan time was 1 minute and 18 seconds. The conventional MRS absorption spectrum from the GE MRS phantom for this scan is generated by the method described by Frigo, Heinen, et al.,[3],[4] and is shown in Fig. 84. An MRS absorption spectrum is generated for each of the 8 channels and shown in Fig. 85.

Each channel was processed independently to apply the appropriate phase-correction and residual water removal on the water-suppressed MRS data obtained during the MRS experiment. Conventional 2D Capon analysis with $N = 1792$, $N_\omega = N/2$, $N_\sigma = 40$, $peak\ threshold = 0.0$ and $M = 256$ is then used with spectrum averaging and signal averaging to compute the results. Fig. 85-Fig. 89 show comparisons between using spectrum averaging and signal averaging techniques to compute 2D spectral estimates on the data acquired from this scan.

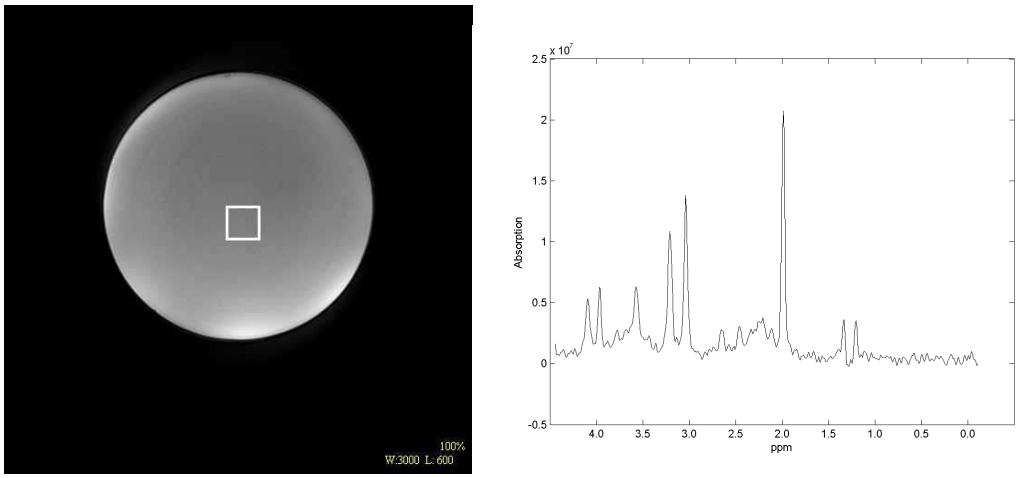

Fig. 84. MRS absorption spectrum from GE MRS phantom using 8-channel head coil.

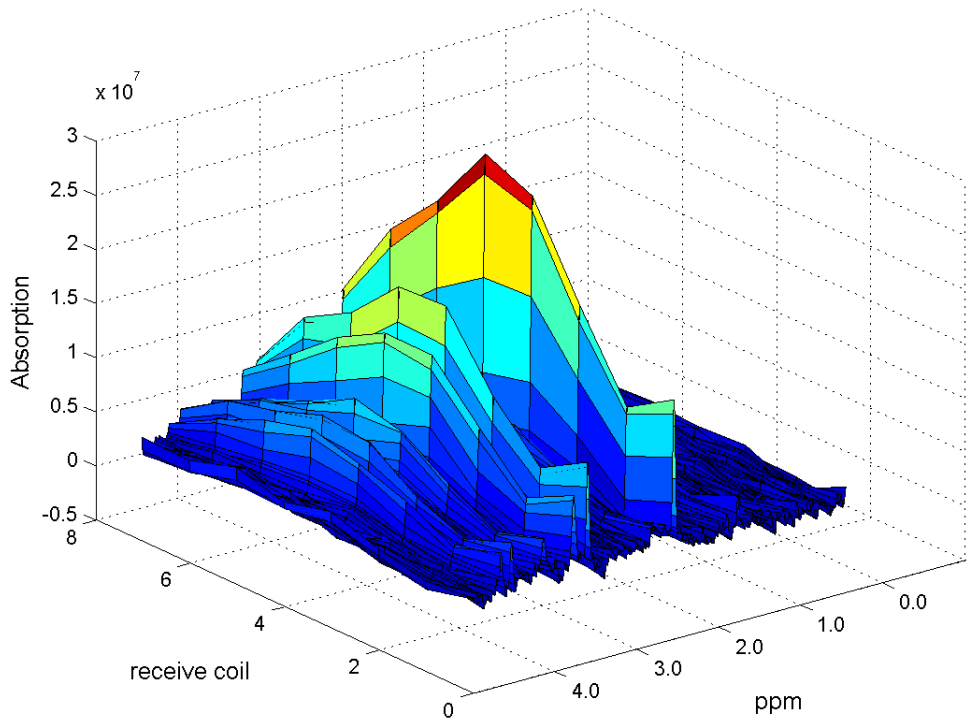

Fig. 85. “Stacked” MRS absorption spectra from each receive coil for GE MRS phantom.

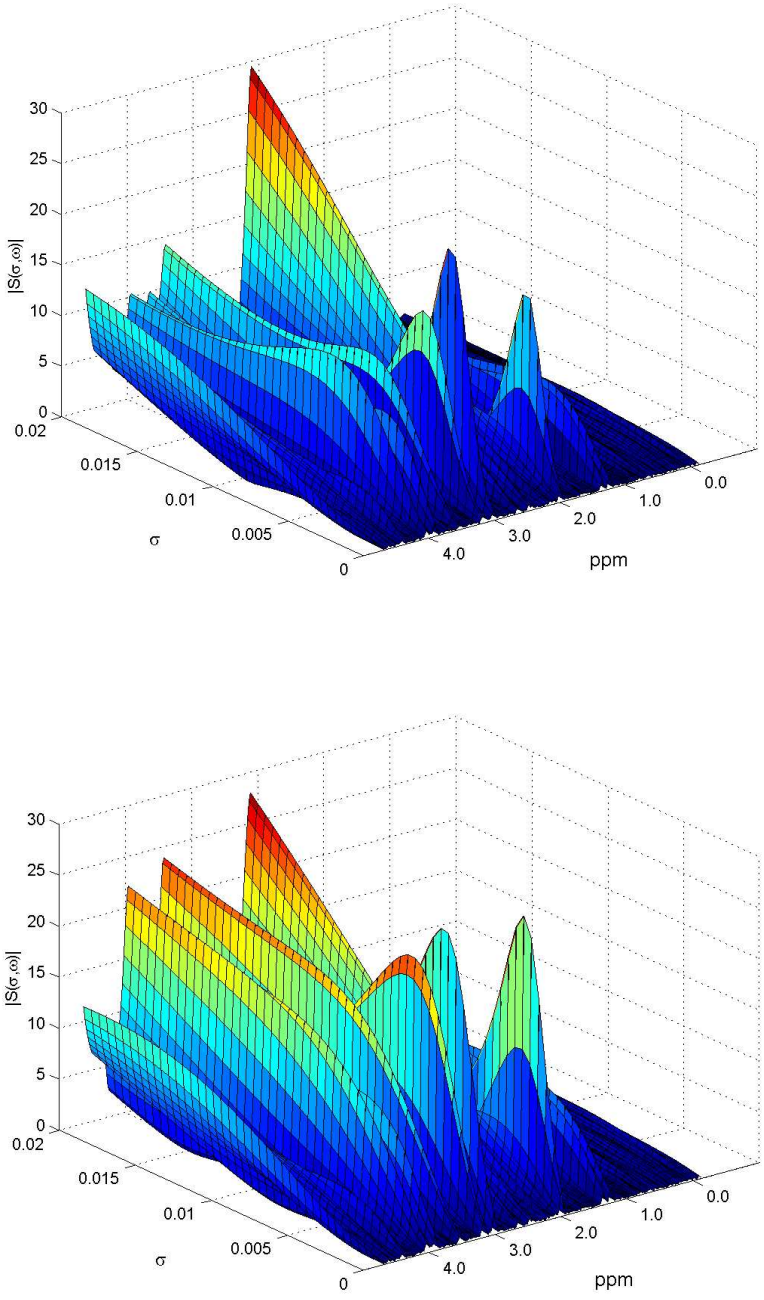

Fig. 86. Signal averaging (top) and spectrum averaging (bottom) used with conventional 2D Capon analysis on a GE MRS phantom.

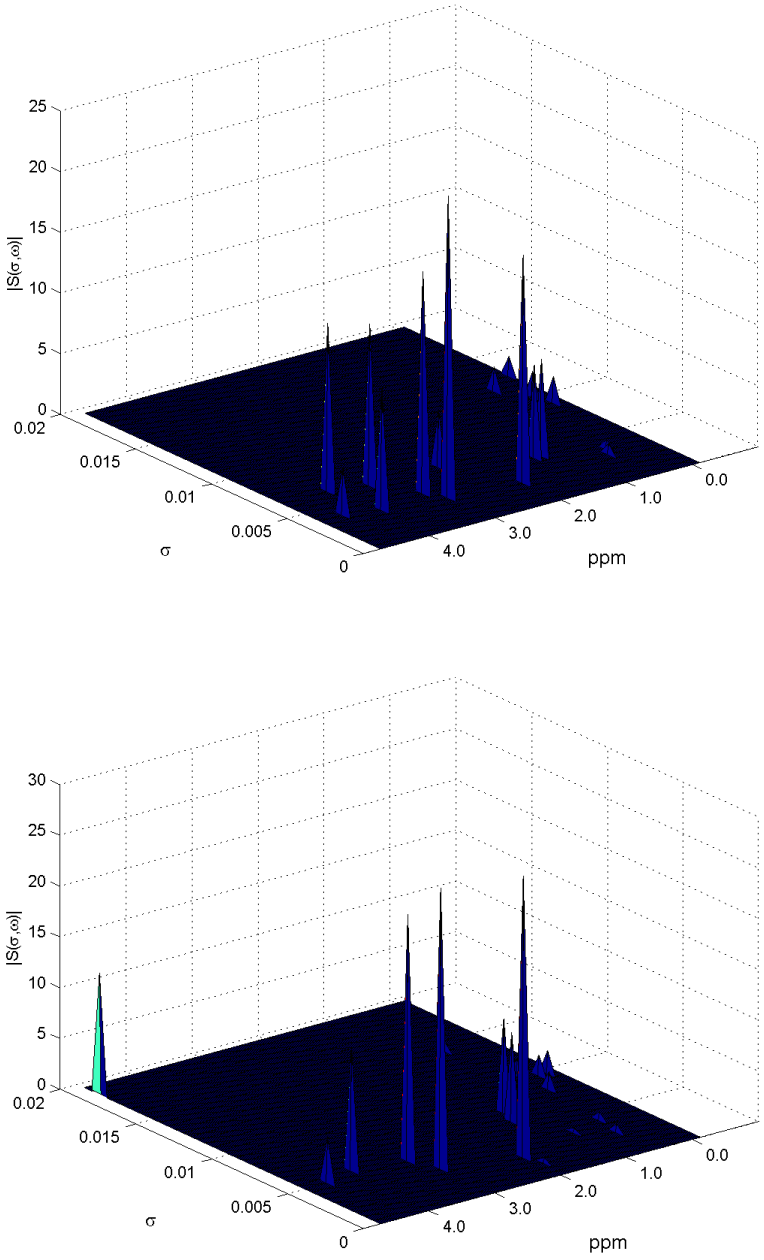

Fig. 87. Peak-enhanced spectra obtained by signal averaging (top) and spectrum averaging (bottom) with conventional 2D Capon analysis on a GE MRS phantom.

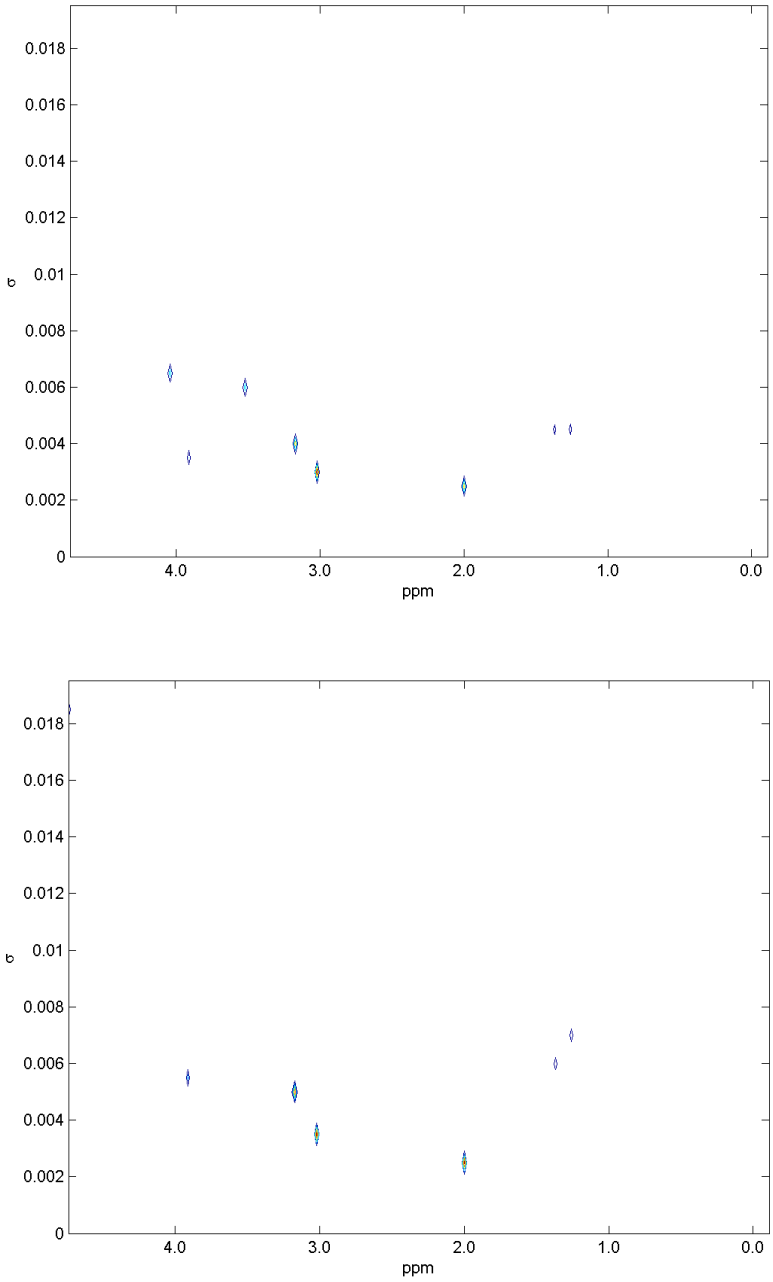

Fig. 88. Contour plots obtained by signal averaging (top) and spectrum averaging (bottom) with conventional 2D Capon analysis on a GE MRS phantom.

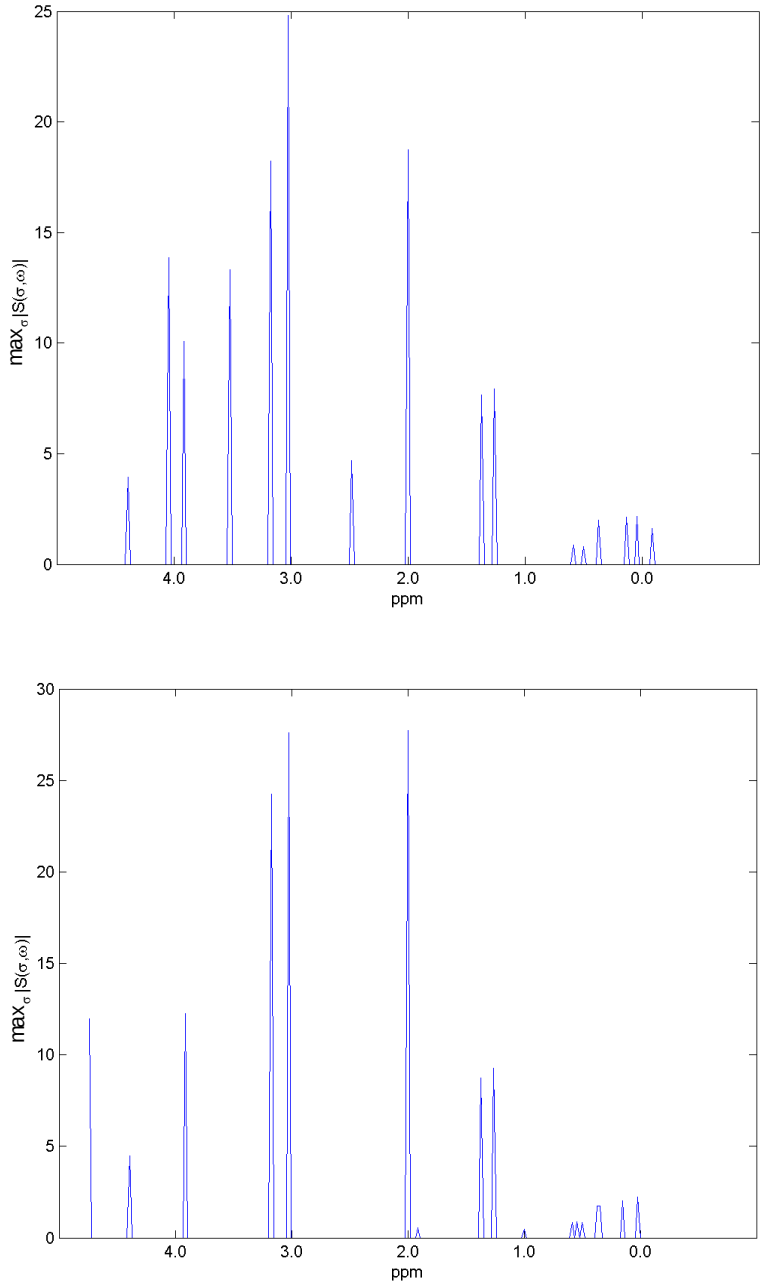

Fig. 89. Maximum peak projections obtained by signal averaging (top) and spectrum averaging (bottom) with conventional 2D Capon analysis on a GE MRS phantom.

It is observed from these comparisons that signal averaging and spectrum averaging both provide suitable 2D spectral estimates for the data obtained from the GE MRS phantom. The signal averaging technique finds more peaks and is approximately 8 times faster.

7.3.2 Conventional 2D Capon Analysis of *In Vivo* Data

We now compare spectrum averaging and signal averaging techniques used on *in vivo* MRS data acquired from the brain of a human volunteer with an eight-channel domed head coil, manufactured by MRI Devices, Inc. (Waukesha, WI, USA). The scan was performed using a GE 1.5T MR scanner and a PRESS sequence on an 8 cc volume, with a TE of 144msec and a TR of 1500msec, 8 NEX, 2 reference frames and 16 water-suppressed (CHESS) frames. The total scan time was 3 minutes and 48 seconds. The conventional MRS absorption spectrum for this scan is generated by the method described by Frigo, Heinen, et al.,[3],[4] and is shown in Fig. 90. An MRS absorption spectrum is generated for each of the 8 channels and shown in Fig. 91.

Each channel was processed independently to apply the appropriate phase-correction and residual water removal on the water-suppressed MRS data obtained during the MRS experiment. Conventional 2D Capon analysis with $N = 1792$, $N_\omega = N/2$, $N_\sigma = 40$, $peak\ threshold = 0.0$ and $M = 256$ is then used with spectrum averaging and signal averaging to compute the results. Fig. 92 - Fig. 95 show comparisons between

using spectrum averaging and signal averaging techniques to compute 2D spectral estimates on the data acquired from this scan.

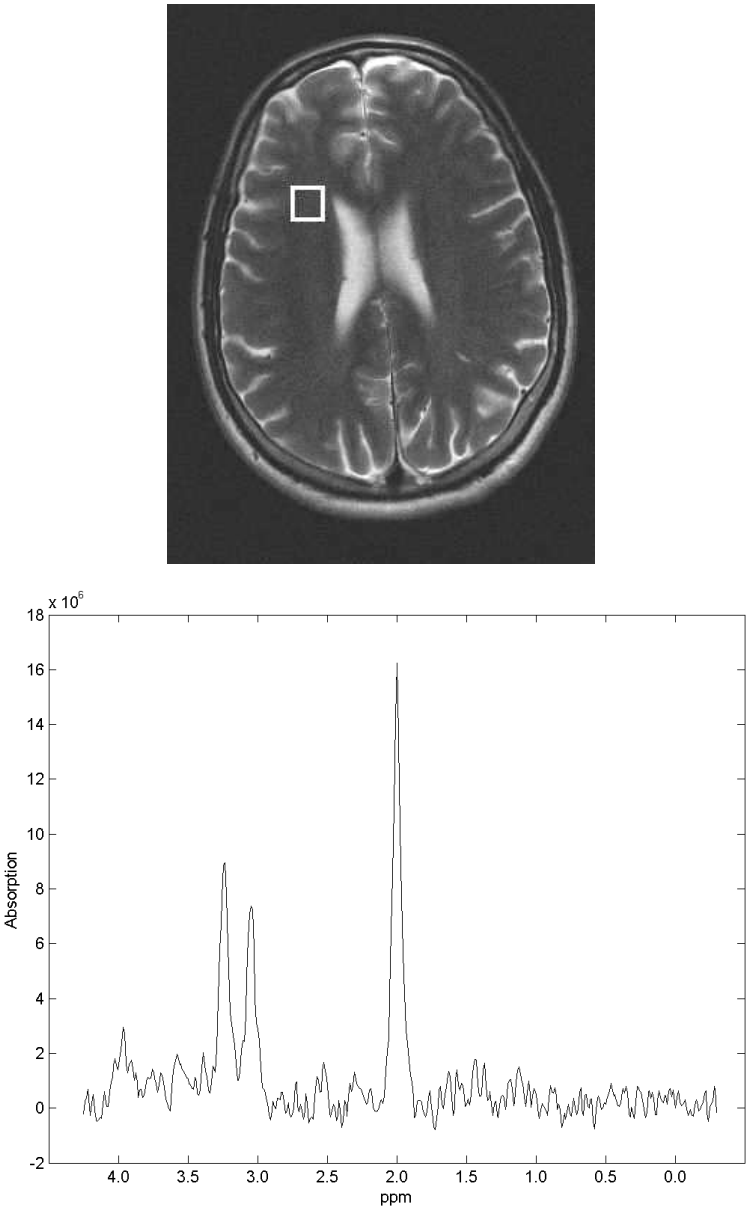

Fig. 90. MRS absorption spectrum from human volunteer using eight-channel head coil.

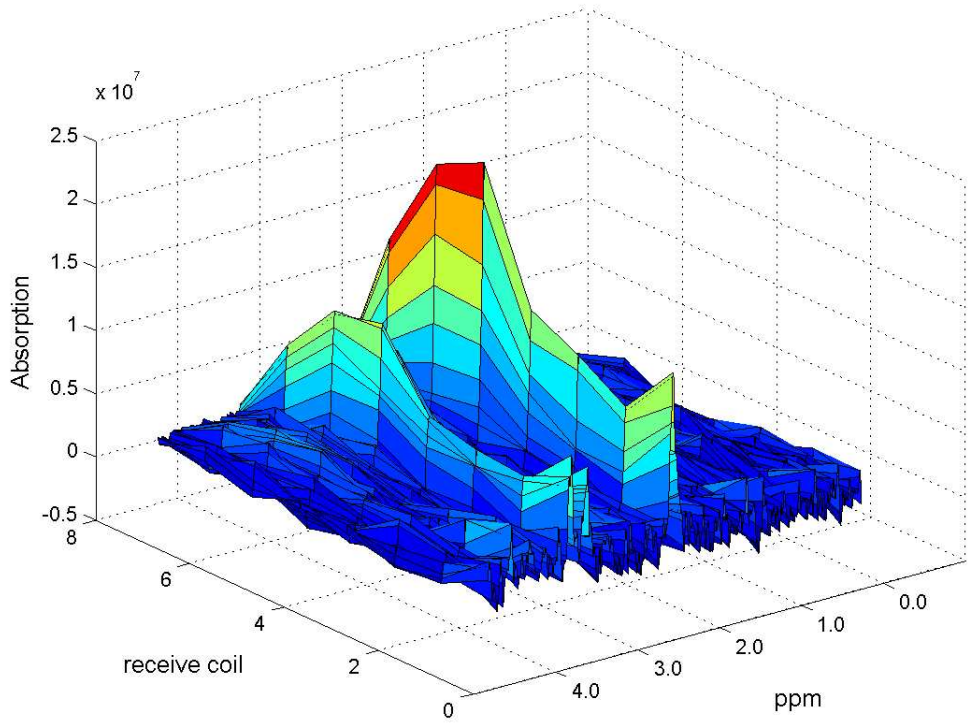

Fig. 91. “Stacked” MRS absorption spectra from each receive coil from brain of human volunteer.

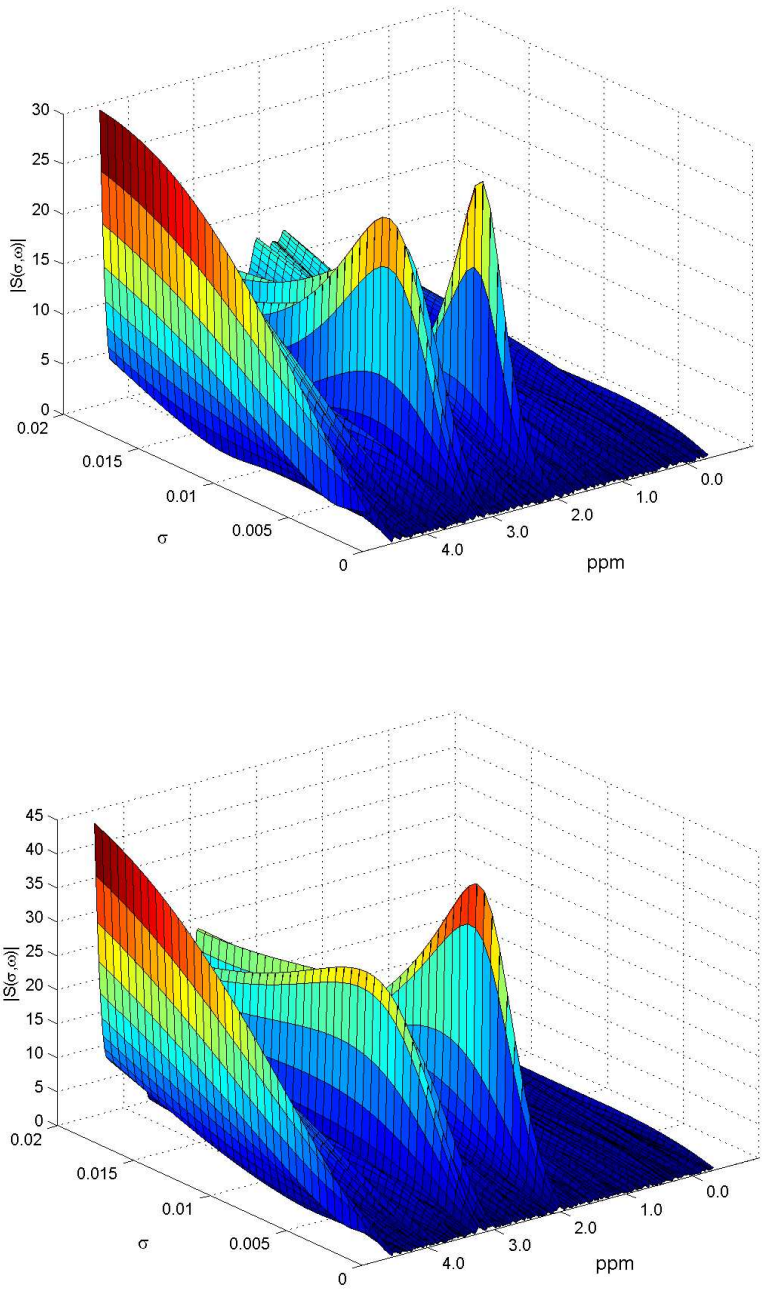

Fig. 92. Signal averaging (top) and spectrum averaging (bottom) used with conventional 2D Capon analysis on the brain of a human volunteer.

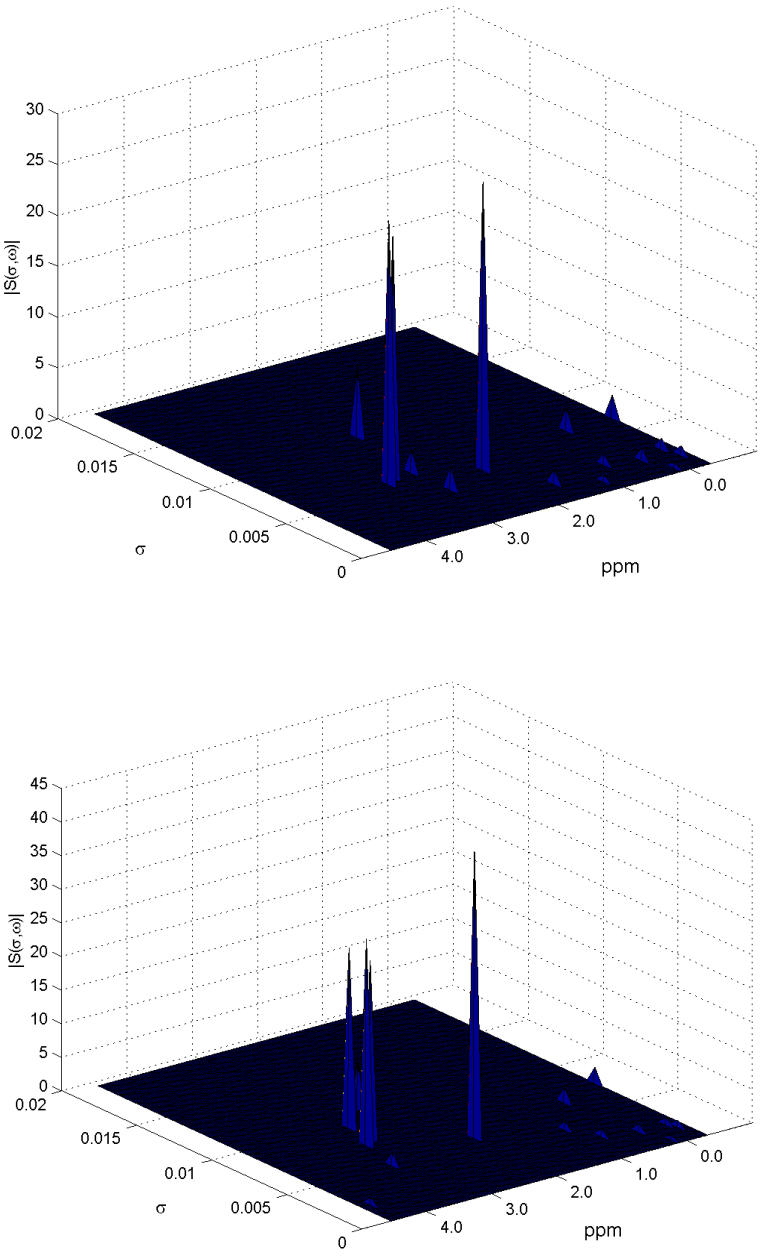

Fig. 93. Peak-enhanced spectra obtained by signal averaging (top) and spectrum averaging (bottom) with conventional 2D Capon analysis on the brain of a human volunteer.

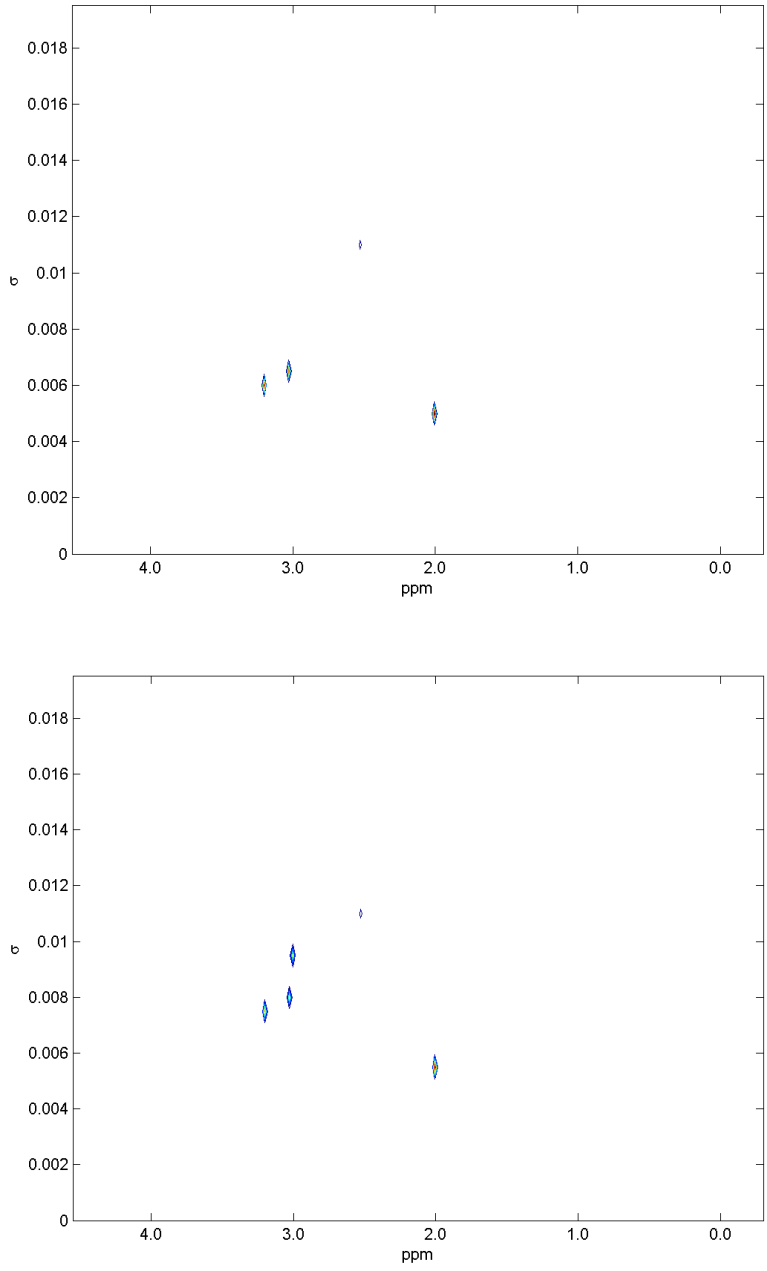

Fig. 94. Contour plots obtained by signal averaging (top) and spectrum averaging (bottom) with conventional 2D Capon analysis on the brain of a human volunteer.

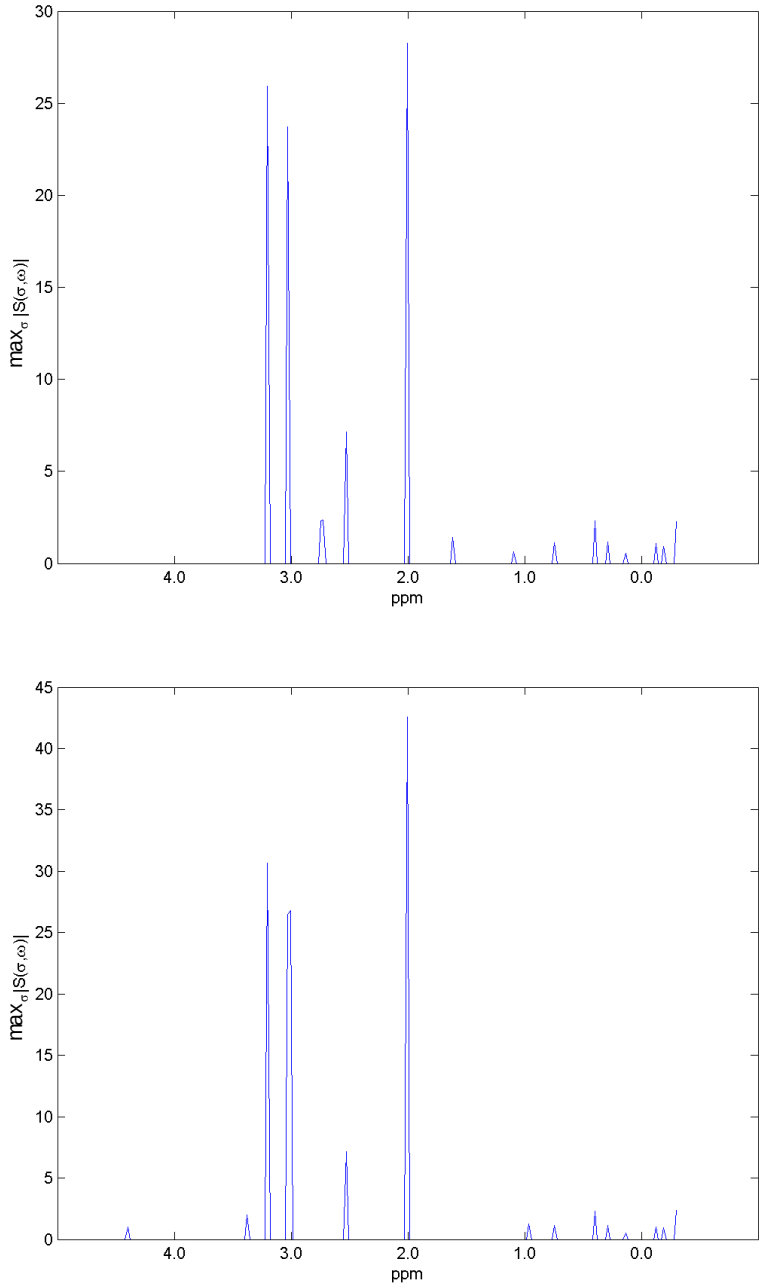

Fig. 95. Maximum peak projections obtained by signal averaging (top) and spectrum averaging (bottom) with conventional 2D Capon analysis on the brain of a human volunteer.

As was observed from our multiple-channel studies using the GE MRS phantom in Sec. 7.3.1 we observe from these comparisons that the signal averaging technique finds more peaks than the spectrum averaging technique and has the additional benefit that it is approximately 8 times faster when using an 8-channel receive coil.

7.3.3 Weighted 2D Capon Analysis of MRS Phantom Data

We now compare using multiple-channel conventional 2D Capon analysis with multiple-channel weighted 2D Capon analysis. In both cases, signal averaging will be used on MRS data acquired using an eight-channel domed head coil, manufactured by MRI Devices, Inc. (Waukesha, WI, USA) from a GE MRS phantom. The scan was performed using a GE 1.5T MR scanner and a PRESS sequence on an 8 cc volume, with a TE of 35msec and a TR of 1500msec, 2 NEX, 8 reference frames and 16 water-suppressed (CHESS) frames. The total scan time was 1 minute and 18 seconds. The MRS experiment for this comparison was the same as that shown in Fig. 84 and described in Sec. 7.3.1.

Conventional 2D Capon analysis with $N = 1792$, $N_\omega = N/2$, $N_\sigma = 40$, $peak\ threshold = 0.0$ and $M = 256$ is compared to weighted 2D Capon analysis with $N = 1792$, $N_\omega = N/2$, $N_\sigma = 40$, $peak\ threshold = 0.0$, $\beta = -\sigma/2$, and $M = 256$. Fig. 96 - Fig. 99 show comparisons between the conventional 2D Capon technique and the weighted 2D Capon technique for the multiple-channel data acquired from this scan.

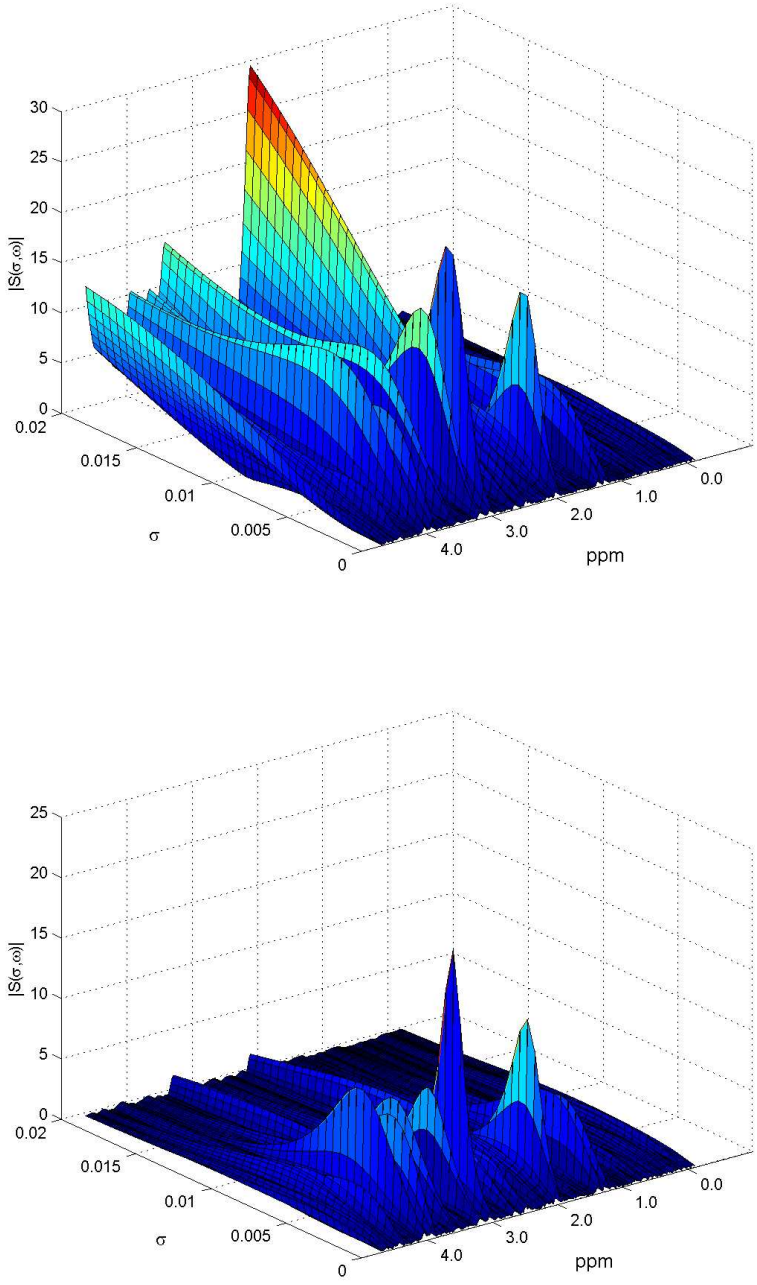

Fig. 96. 8-channel signal averaging for conventional 2D Capon (top) and weighted 2D Capon with $\beta = -\sigma/2$ (bottom) on the GE MRS phantom.

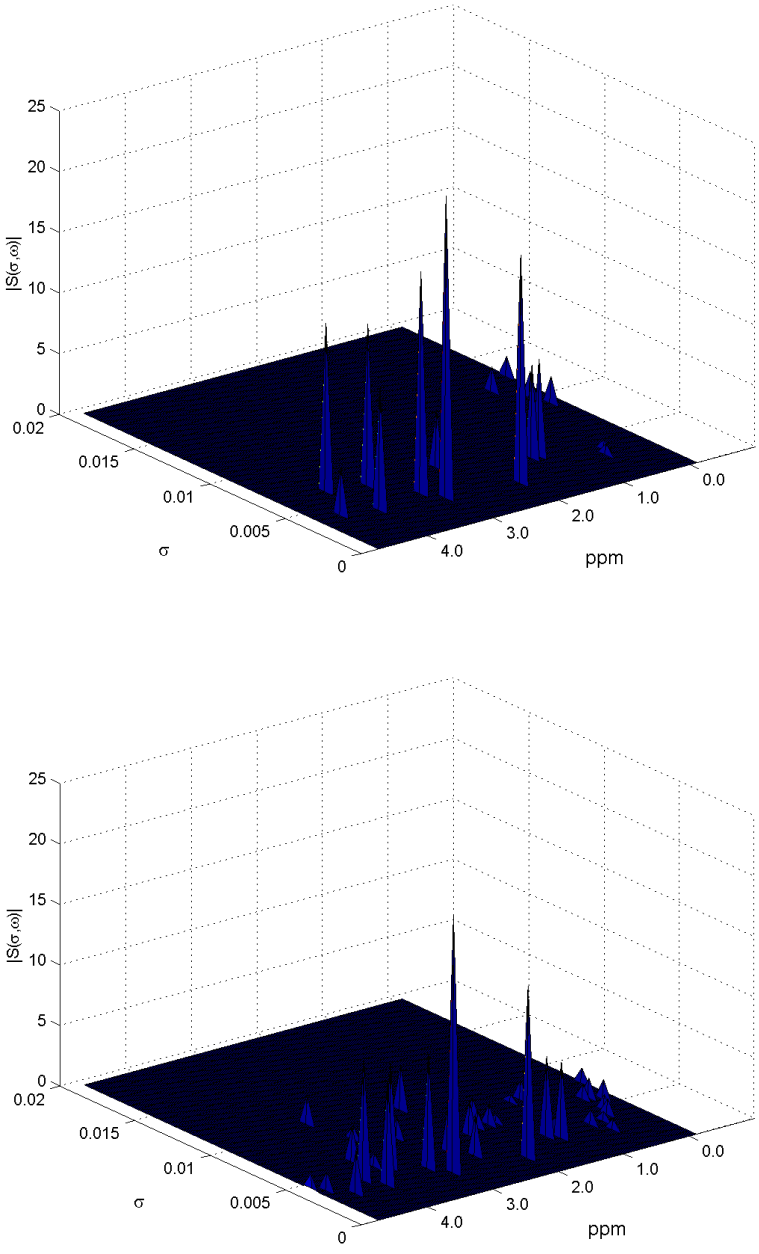

Fig. 97. Peak-enhanced spectra from 8-channel signal averaging for conventional 2D Capon (top) and weighted 2D Capon with $\beta = -\sigma/2$ (bottom) on the GE MRS phantom.

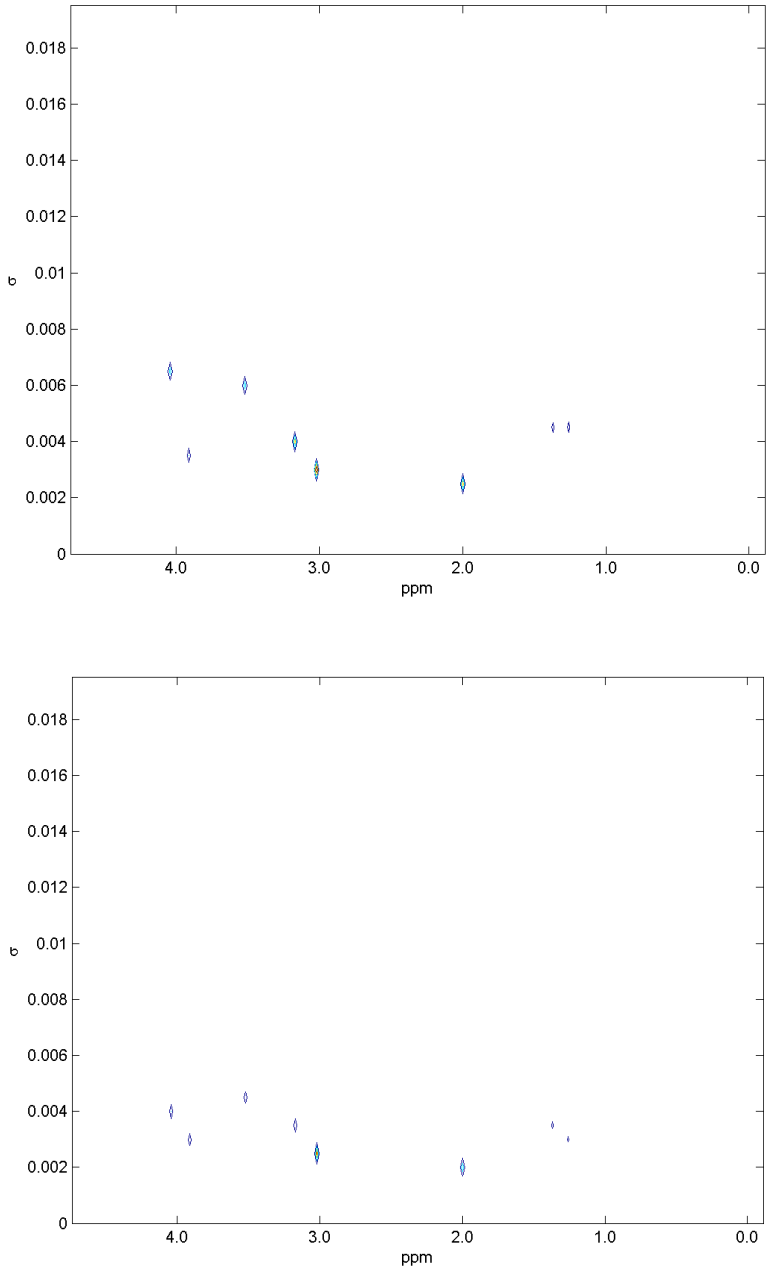

Fig. 98. Contour plots from 8-channel signal averaging for conventional 2D Capon (top) and weighted 2D Capon with $\beta = -\sigma/2$ (bottom) on the GE MRS phantom.

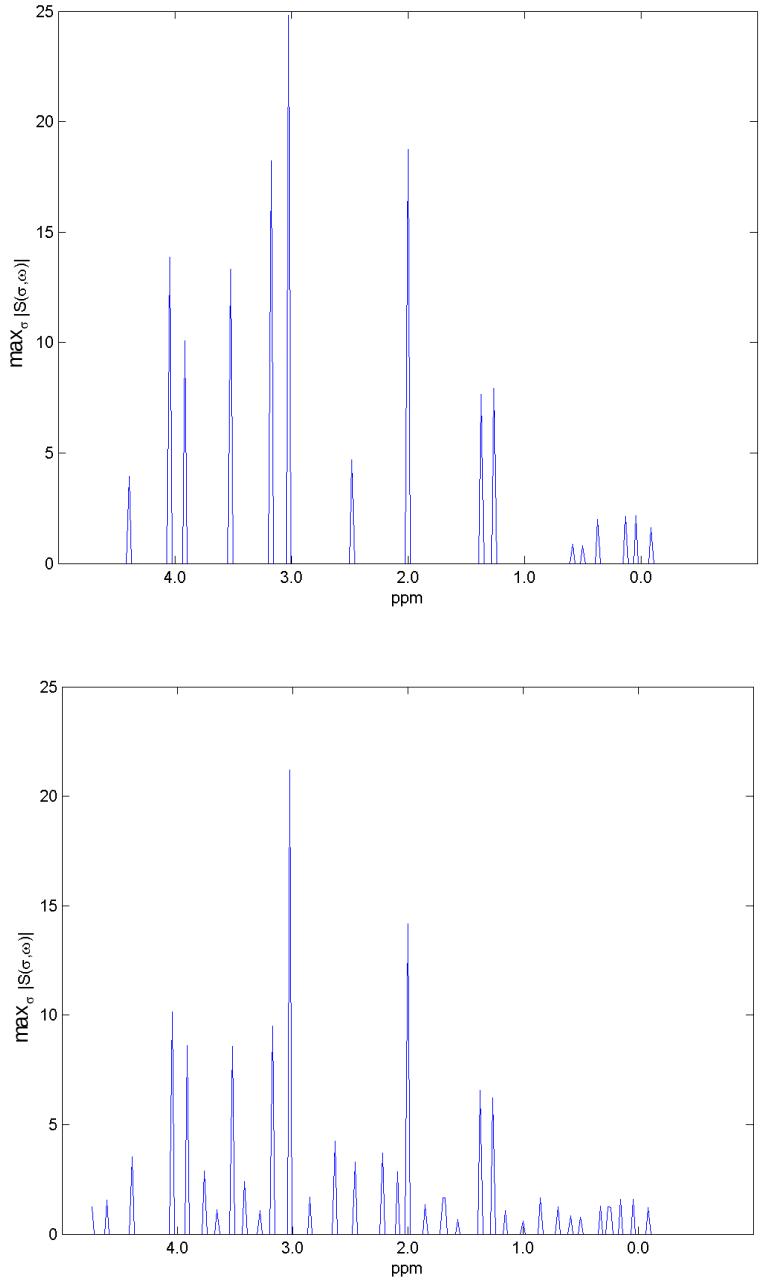

Fig. 99. Maximum peak projections from 8-channel signal averaging for conventional 2D Capon (top) and weighted 2D Capon with $\beta = -\sigma/2$ (bottom) on the GE MRS phantom.

From Fig. 99 we observe that the 2D weighted Capon technique identifies more peaks from this brain study than the conventional 2D Capon technique. It should be pointed out, however, that the 2D weighted Capon technique chosen for this evaluation is more than twice as slow in terms of execution speed.

7.4 Summary

In this chapter we have examined a number of examples of applying 2D spectral estimation techniques to MRS data processing. First, we compared conventional 2D Capon analysis to conventional 2D APES analysis. Then, we examined how weighted 2D Capon analysis performed on MRS data acquired using a single-channel head coil from the GE MRS phantom and from the brain of a healthy volunteer. We further extended our investigation to demonstrate how certain weighted 2D Capon techniques can identify more peaks in less execution time than conventional 2D Capon analysis. We also investigated how well conventional 2D Capon analysis performs on MRS data sets that have not been phase-corrected and from which residual water was not removed.

For multiple-channel MRS data sets, we compared signal averaging and spectrum averaging using conventional 2D Capon analysis on MRS data acquired using an eight-channel head coil. Both techniques seem to work well on data from the GE MRS phantom and from the brain of a healthy human volunteer. Signal averaging seems to identify more peaks and requires less computational processing. We extended our investigation to demonstrate how multiple-channel weighted 2D Capon techniques work with signal averaging.

The application of the new 2D spectral estimation techniques to data acquired during MRS scans seems to offer the same benefits as determined from the simulations performed in Chapter 6, namely, increased peak detection capabilities and/or reduced execution time for certain cases. We have shown through the examples in this chapter that these new 2D spectral estimation techniques can be used effectively for MRS applications and show great promise for future integration into common clinical practice.

Chapter 8

Summary and Conclusions

The motivation for the research discussed in this dissertation was initially to find improved signal processing techniques for MRS to enable the non-invasive measurement of blood glucose *in vivo* for improved diabetes disease management and control. The 2D spectral estimation techniques that evolved out of this research, however, apply to typical clinical MRS applications as well, so the results of this research have a much broader impact than was originally intended. At this time, we provide a brief summary and offer conclusions related to the new signal processing techniques introduced for 2D spectral estimation and for their application to MRS data.

8.1 Summary

We began our discussion with a brief review of the principles of nuclear magnetic resonance in Chapter 2. The remarkable scientific discoveries that were made long ago have provided a foundation upon which MR scanners have been created, and the impact of these medical imaging machines to the practice of modern medicine has been extraordinary.

In Chapter 3 we examined SVQ proton MRS which is typically used in a clinical setting to quantify metabolites in the human brain. We provided a comprehensive review of the signal processing algorithms that are typical for MRS SVQ scans and carefully

documented each processing step. A significant contribution of our work has been to implement efficient algorithms in MATLAB to create a conventional MRS absorption spectrum from data acquired during an SVQ study for both single-channel and multiple-channel SVQ. These algorithms are included in the Appendix, Sec.A.4.

In Chapter 4 we introduced several *new* nonparametric two-dimensional spectral estimation techniques: the weighted 2D Capon method, the weighted 2D APES method, and the combined weighted 2D APES / 2D Capon method. Each of these techniques provides spectral estimates of damping as well as frequency, making them useful for MRS data analysis. We show through extensive simulations carried out in Chapter 6 that these new techniques offer improved performance in terms of peak identification , estimation accuracy and/or computation time over conventional 2D Capon and 2D APES in certain cases. MATLAB implementations of these algorithms are included in the Appendix, Sec. A.5.

In Chapter 5 we introduced two *new* algorithms for multiple-channel 2D spectral estimation: signal averaging and spectrum averaging. Each of these techniques provide a means of creating a 2D spectral estimate from multiple-channel MRS data. We also introduced a method to optimally estimate the relative channel gains from observed data, which provides great benefit when the ideal gains for each channel are not known. This technique is potentially applicable to a variety of other problems as well. We show through extensive simulations carried out in Chapter 6 the merits of these new techniques. MATLAB code for the simulations is included in the Appendix, Sec. A.6.

In Chapter 6, we evaluated the proposed new techniques introduced in Chapter 4 and Chapter 5 by performing an extensive set of simulations. The motivation for performing these simulations was to provide a controlled set of conditions, and then to compare and contrast the performance of these new algorithms for several different scenarios. In this chapter we also introduced the concepts of peak spectrum and projected peak spectrum. While these are rather simple ideas, they greatly enhance the user's ability to interpret the data.

In Chapter 7 we applied the new techniques proposed in Chapters 4 and 5, and simulated in Chapter 6, to data collected during MRS experiments. Examples were provided demonstrating how these techniques work in MRS studies involving phantoms with known concentrations of metabolites, as well as in a limited number of *in vivo* MRS studies involving human volunteers.

8.2 Conclusions

In the case of single-channel 2D spectral estimation, extensive simulations have led to the following conclusions for the situations studied:

1.) For the weighted 2D Capon and weighted 2D APES methods, and their conventional counterparts, the filter length M should generally be made as large as possible to reduce missed peaks and false peaks. On the other hand, intermediate values of M lead to reduced magnitude and damping estimation errors.

2.) The weighted 2D Capon method for $K = 1$ and for $K = N$, $\alpha = -\sigma/2$ provides computation time savings of 17-35% as compared to the conventional 2D Capon method for the cases studied.

3.) The parameter choices in the weighted 2D Capon and weighted 2D APES methods offer various trade-offs in the four performance measures studied. By proper choice of these parameters, one may obtain improvement in any one of the performance measures, as compared to the conventional methods. However, this is usually accompanied by degradation in one or more of the remaining measures. In some of the cases studied the new methods required no additional computation time as compared to the corresponding conventional methods, while in other cases the computation time is more than doubled.

4.) As the parameter γ varies from 0 to 1, the performance of the combined 2D APES / 2D Capon method gradually transitions from that of the weighted 2D Capon method to that of the weighted 2D APES method. In every case the computation time equals that of the conventional 2D APES method.

5.) It is clear that no one method is best for all situations. In a given application it is necessary to decide what performance measures are most critical and how much computation time is acceptable and then to choose

the method and associated parameters accordingly. Additional simulations may be needed if the nature of the data is significantly different from that studied here.

In the case of multiple-channel 2D spectral estimation, the simulations have led to the following conclusions:

1.) In general, signal averaging is preferred over spectrum averaging because of its significantly better performance and reduced execution time (its execution time is roughly equal to that of the single-channel case).

2.) For the case of equal channel noise variances, the multiple-channel signal averaging techniques significantly outperform the single-channel techniques applied to any of the channels. On the other hand, if one of the channels has a much smaller noise variance than any of the others, the multiple-channel techniques produce essentially the same results as would be obtained using the corresponding single-channel technique on the least noisy channel.

3.) Using estimated channel gains leads to virtually no performance degradation as compared to using ideal (known) gains.

The primary goal of this research was to enhance the clinical efficiency and utility of MRS through improved 2D spectral estimation techniques. We have examined these new techniques in great detail, and have shown how they may be applied to MRS signal

processing. Two potential clinical benefits stemming from this research are: increased accuracy for MRS diagnosis and increased patient throughput through reduced scan time and/or faster analysis techniques.

As a result of this research, a related algorithm for generating a single MRS absorption spectrum for multiple-channel SVQ has been implemented [3],[4] and has been received favorably by a number of clinical sites around the world. 2D spectral estimation for MRS has yet to gain widespread clinical acceptance. In part, this is due to the lack of simple clinical tools allowing clinicians and spectroscopists to analyze the results of these techniques, but also it is due to the fact that 2D spectral estimation for MRS has only recently been introduced [2],[74], and the clinical benefits of this approach are still being evaluated.

Based on the results of extensive simulations and a limited number of MRS experiments, we are confident that the new techniques introduced in this dissertation have the potential to add clinical value and improve the overall quality of spectral analysis for MRS studies.

8.3 Suggestions for Future Research

The new 2D spectral estimation techniques proposed in this dissertation warrant further investigation on several fronts. First, the simulations carried out in this study, while quite extensive, have not considered the myriad of possible situations. For example, the quantities N, N_ω, and N_σ were held constant in the studies to make the

analysis tractable. In addition, the values of the parameters α and β used in the weighted 2D Capon and weighted 2D APES simulations were limited. Further research considering a more thorough and systematic collection of these parameters would be of value. Further studies should also take into account different test signals that might be appropriate in different applications such as speech processing or synthetic aperture radar (SAR) imaging.

Brief mention was made in Chapter 6 of the fact that "off-grid" peaks (those not occurring on ω grid lines) can be very difficult to identify, particularly for high SNR data. This has not been addressed, or even mentioned, in the literature. Because it was decided that this problem was beyond the scope of this study, all peaks were located on ω grid lines in the simulations. One simple remedy for this problem is to establish a finer grid in the ω direction, i.e., to increase N_ω. However, this increases computation time. Further studies concerning ways to improve "off-grid" peak detection are warranted.

Because certain 2D spectral estimation techniques perform better in some respects than others, it may be possible to improve performance by using a combination of techniques. One might consider taking the results from one technique and providing them to a second or even third technique for additional processing. For example, a method that performs well in terms of minimizing missed peaks or false peaks might be used to identify the values of ω associated with peaks. A second method, which

performs well in estimating damping values, could be used at these ω's to estimate the σ's. Finally, a third method, which performs well on magnitude estimation, could be applied at the (σ,ω) locations identified by the earlier methods to estimate the magnitudes. Stoica and Sundin[2] have, in fact, employed such a procedure with some success, which they refer to as the 2D CAPES method. In 2D CAPES, the peak locations are determined using the 2D Capon method, and then the 2D APES method is applied only at these peak locations to estimate the magnitudes. Further studies in this area could lead to overall performance improvements.

The method introduced in Chapter 5 for optimally estimating relative channel gains from observed data would seem to be of significant interest in its own merits. Other possible applications might be in estimating gains for microphone arrays used in speech processing and for antenna arrays used for communications or SAR imaging. In fact, the estimated gains could also be used to establish weights for signal averaging using conventional FFT processing of multiple-channel MRS data and to implement multiple-channel CSI. These applications would seem to be worthy of further consideration.

In this work the application of 2D spectral estimation to MRS phantom and *in vivo* data was quite limited. There is a need to perform substantive clinical evaluations of the new 2D spectral estimation techniques for MRS applications proposed in this dissertation. The application of these techniques to a limited number of MRS data sets

in Chapter 7 offers proof of concept that these techniques may have clinical merit. However, the efficacy of using these and other 2D spectral estimation techniques in a clinical setting must be determined.

References

[1] M. W. Weiner, "MRI/MRS of Neurodegenerative Diseases and Epilepsy," *Proc. of ISMRM*, vol. 11, p. 2, July 2003.

[2] P. Stoica and T. Sundin, "Nonparametric NMR Spectroscopy," *Journal of Magnetic Resonance*, vol. 152, pp. 57-69, 2001.

[3] F. J. Frigo, J. A. Heinen, T. E. Raidy, J. A. Hopkins and S. G. Tan, "Magnetic Resonance Spectroscopy Using Multiple Receive Coils," *Proc. of ISMRM*, vol. 12, p. 2260, 2004.

[4] F. J. Frigo, J. A. Heinen, T. E. Raidy and J. A. Hopkins, "Method and System of Generating MRS Spectra from Multiple Receiver Data," *US Patent Application 10/615,714*, July 9, 2003.

[5] W. Pauli, "Theory of Relativity," *Enzyklopaedie der Mathematischen Wissenschaften*, vol. 5, part 2, 1920.

[6] I. I. Rabi, J. R. Zacharias, S. Millman and P. Kusch., "A New Method of Measuring Nuclear Magnetic Moments," *Physics Review*, vol. 53, p. 318, 1938.

[7] N. F. Ramsey, "The Legacy of I. I. Rabi", *Proc. of ISMRM – 7th Annual Lauterbur Lecture*, 2003.

[8] F. Bloch, W. W. Hensen and M. E. Packard, "The Nuclear Induction Experiment," *Physical Review*, vol. 70, pp. 474-485, 1946.

[9] E. M. Purcell, H. Torrey and R. Pound, "Resonance Absorption by Nuclear Magnetic Moments in a Solid," *Physical Review*, vol. 69, pp. 37-38, 1946.

[10] R. V. Damadian, "Tumor Detection by Nuclear Magnetic Resonance," *Science*, vol. 171, p. 1151, 1971.

[11] P. C. Lauterbur, "Image Formation by Induced Local Interactions: Examples of Employing Nuclear Magnetic Resonance," *Nature*, vol. 242, pp.190-191, 1973.

[12] P. Mansfield and P. K. Grannell, "'Diffraction' and Microscopy in Solids and Liquids by NMR," *Physical Review B*, vol. 12, no. 9, pp. 3618-3634, Nov. 1975.

[13] D. G. Nishimura, *Principles of Magnetic Resonance Imaging*, Stanford University, Palo Alto, CA, 1996.

[14] E. M. Haacke, R. W. Brown, M. R. Thompson and R. Venkatesan, *Magnetic Resonance Imaging – Physical Principles and Sequence Design*, Wiley-Liss, New York, NY, 1999.

[15] M. T. Vlaardingerbroek and J. A. Den Boer, *Magnetic Resonance Imaging-Theory and Practice*, Springer, New York, NY, 1999.

[16] M. A. Brown and R. C. Semelka, *MRI: Basic Principles and Applications*, Wiley-Liss, New York, NY, 1995.

[17] A. D. Elster, *Questions and Answers in Magnetic Resonance Imaging*, Mosby, St. Louis, MO, 1994.

[18] M. A. Bernstein, K. F. King and Z. J. Zhou, *Handbook of MRI Pulse Sequences*, Academic Press, New York, NY, 2004.

[19] A. Abragam, *The Principles of Nuclear Magnetism*, Oxford Press, New York, NY, 1985.

[20] C. P. Slichter, *Principles of Magnetic Resonance*, Springer-Verlag, New York, NY, 1992.

[21] R. T. Morrison and R. N. Boyd, *Organic Chemistry - Third Edition*, Allyn and Bacon, Inc., Boston, MA, 1978.

[22] N. Salibi and M. A. Brown, *Clinical MR Spectroscopy*, Wiley-Liss, New York, NY, 1998.

[23] H. Friebolin, *Basic One- and Two-Dimensional NMR Spectroscopy - Third Edition*, Wiley-VCH, New York, NY,1998.

[24] M. Rudin, R. de Beer, et al., *In-Vivo Magnetic Resonance Spectroscopy I; Probeheads and Radio frequency Pulses, Spectrum Analysis*, Springer-Verlag, New York, NY, 1992.

[25] D. M. Spielman, E. Adalsteinsson and K. O. Lim, "Quantitative Assessment of Improved Homogeneity Using Higher-order Shims for Spectroscopic Imaging of the Brain," *Magnetic Resonance in Medicine*, vol. 40, pp. 376-382, 1998.

[26] F. A. Bovey, L. Jelinski and P. A. Mirau, *Nuclear Magnetic Resonance Spectroscopy*, Academic Press, Inc., New York, NY, 1988.

[27] A. Rahman, *One and Two Dimensional NMR Spectroscopy*, Elsevier, New York, NY, 1989.

[28] A. Rahman and M. I. Choudhary, *Solving Problems with NMR Spectroscopy*, Academic Press, Inc., San Diego, CA, 1996.

[29] P. B. Roemer, W. A. Edelstein, C. E. Hayes, S. P. Souza and O. M. Mueller, "The NMR Phased Array," *Magnetic Resonance in Medicine*, vol.16, p. 192, 1990.

[30] S. E. Moyher, D. B. Vigeron and S. J. Nelson, "Surface Coil MR Imaging of the Human Brain with an Analytic Reception Profile Correction," *Journal of Magnetic Resonance*, vol. 5, pp. 139-144, 1995.

[31] P. R. P. Hoole, *Smart Antennas and Signal Processing for Communications, Biomedical and Radar Systems*, WIT Press, Boston, MA, 2001.

[32] R. F. Lee, R. O. Giaquinto and C. J. Hardy, "Coupling and Decoupling Theory and Its Application to the MRI Phased Array," *Magnetic Resonance in Medicine*, vol. 48, pp. 203-213, 2002.

[33] R. Yan, D. Erdogmus, E. G. Larsson, J. C. Principe and J. R. Fitzsimmons, "Image Combination for High-Field Phased-Array MRI," *IEEE International Conf. on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 1-4, April 2003.

[34] R. C. Hansen, *Phased Array Antennas*, John Wiley & Sons, Inc., New York, NY, 1998.

[35] S. Stergiopoulos, *Advanced Signal Processing Handbook: Theory and Implementation for Radar, Sonar, and Medical Imaging Real-Time Systems*, CRC Press, Inc., New York, NY, 2001.

[36] J. Jin, *Electromagnetic Analysis and Design in Magnetic Resonance Imaging*, CRC Press, Inc., New York, NY, 1999.

[37] J. Chen, S. Jeng, F. Lin and W. Kuan, "Quantitative Analysis of Magnetic Resonance Radio-Frequency Coils Based on Method of Moment," *IEEE Trans. on Magnetics,* vol. 35, no. 3, pp. 2118-2127, May 1999.

[38] F. Lin, W. Kuan, S. Jeng and J. Chen, "Quantitative Spectral/Spatial Analysis of Phased Array Coil in Magnetic Resonance Imaging Based on Method of Moment," *IEEE Trans. on Medical Imaging*, vol. 18, no. 12, pp.1129-1137, Dec. 1999.

[39] L. L. Wald, S. E. Moyner, M. R. Day, S. J. Nelson and D. B. Vigneron, "Proton Spectroscopic Imaging of the Human Brain Using Phased Array Detectors," *Magnetic Resonance in Medicine* , vol. 34, pp. 440-445, 1995.

[40] S. G. Tan, W. Song, A. Jesmanowicz, J. S. Hyde, T. E. Raidy and S. J. Li, "Multi-Channel Magnetic Resonance Spectroscopy," *Proc. SMRM 12th Annual Meeting*, p. 370, 1993.

[41] T. Shirmer, S. Kohler, D. Gultekin and T. E. Raidy, "A Simple Absolute Scaling for Spectroscopic Data Acquired with Phased-Array Coils at 1.5T," *Proc. of ISMRM*, vol. 12, 2004.

[42] P. B. Baker, J. Gillen, P. C. van Zihl, and X. Golay, "Phased-array Multi-Slice Proton MR Spectroscopic Imaging at 3 Tesla," *Proc. of ISMRM*, vol. 11, p. 1133, 2003.

[43] S. W. Provencher, "Estimation of Metabolite Concentrations from Localized in Vivo Proton NMR Spectra," *Magnetic Resonance in Medicine*, vol. 30, pp. 672-679, 1993.

[44] R. E. Hurd and M. G. Boucher, "Gradient Enhanced NMR Correlation Spectroscopy," *US Patent 5,077,524*, Dec. 31, 1991.

[45] B. J. Soher, R. E. Hurd, N. Sailasuta and P. B. Barker, "Quantitation of Automated Single-Voxel Proton MRS using Cerebral Water as Internal Reference," *Magnetic Resonance in Medicine*, vol. 36, pp. 335-339, 1996.

[46] E. O. Brigham, *The Fast Fourier Transform and its Applications*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[47] C. B. Ahn and Z. H. Cho, "A New Phase Correction Method in NMR Imaging Based on Autocorrelation and Histogram Analysis," *IEEE Trans. on Medical Imaging*, vol. MI-6, no. 1, pp. 32-36, March 1987.

[48] C. B. Ahn, S. Y. Lee, O. Nalcioglu and Z. H. Cho, "Spectroscopic Imaging by Quadrature Modulated Echo Time Shifting," *Magnetic Resonance Imaging*, vol. 4, pp. 110-111, 1986.

[49] C. B. Ahn, et al., "Linear Phase Correction," *IEEE Trans. Med Imaging*, vol. MI-5, no. 1, 1986.

[50] A. E. Derome, *Modern NMR Techniques for Chemistry Research*, Pergamon Press, New York, NY, 1987.

[51] F. Abildgaard, H. Gesmar and J. J. Led, "Quantitative Analysis of Complicated Nonideal Fourier Transform NMR Spectra," *Journal of Magnetic Resonance*, vol. 79, pp. 78-99, 1988.

[52] C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, Berlin, Germany, 1978.

[53] D. S. G. Pollock, *A Handbook of Time-Series Analysis, Signal Processing and Dynamics,* Academic Press, New York, NY, 1999.

[54] R. E. Hurd, "Magnetic Resonance Spectroscopic Imaging Having Reduced Parasitic Sidebands," *US Patent 6,069,478*, May 30, 2000.

[55] G. A. Morris, "Compensation of Instrumental Imperfections by Deconvolution Using and Internal Reference Signal," *Journal of Magnetic Resonance*, vol. 80, pp. 547-552, 1988.

[56] D. B. Clayton, E. Adalsteinsson and D. M. Spielman, "Quantitation of 3D Chemical Shift Data: Nonlinear Least-Squares Spectral Estimation Using a Water Reference and A Priori Knowledge," *Proc. of ISMRM*, vol. 11, p. 1158, 2003.

[57] J. Capon, "High Resolution Frequency Wave Number Spectrum Analysis," *Proc. IEEE*, vol. 57, no. 8, pp. 1408-1418, 1969.

[58] T. Sundin, *Spectral Analysis and Magnetic Resonance Spectroscopy*, Uppsala University, Uppsala, Sweden, 2001.

[59] J. Li and P. Stoica, "Adaptive Filtering Approach to Spectral Estimation and SAR Imaging," *IEEE Trans. Signal Processing*, vol. 44, no. 6, pp. 1469-1484, 1996.

[60] P. Stoica and R. L. Moses, *Introduction to Spectral Analysis*, Prentice-Hall, Upper Saddle River, NJ, 1997.

[61] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*, John Wiley and Sons, Inc., New York, NY, 1996.

[62] S. J. Orfanidis, *Optimum Signal Processing: An Introduction*, McGraw-Hill, Inc., New York, NY, 1988.

[63] Z. Liu, H. Li and J. Li, "Efficient Implementation of Capon and APES for Spectral Estimation," *IEEE Trans. Aerospace and Electronic Systems*, vol. 34, no. 4, pp. 1314-1319, Oct. 1998.

[64] P. Stoica, H. Li, and J. Li, "A New Derivation of the APES Filter," *IEEE Trans. on Signal Processing*, vol. 6, no. 8, pp. 205-206, Aug. 1999.

[65] A. Jakobsson, *Model-Based and Matched-Filterbank Signal Analysis*, Uppsala University, Uppsala, Sweden, 2000.

[66] F. Gini and F. Lombardini, "Multilook APES for Multibaseline SAR Interferometry," *IEEE Trans. on Signal Processing*, vol. 50, no. 7, pp. 1800-1803, July 2002.

[67] E. G. Larsson, P. Stoica and J. Li, "Amplitude Spectrum Estimation for Two-Dimensional Gapped Data," *IEEE Trans. on Signal Processing*, vol. 50, no. 5, pp. 1314-1319, June 2002.

[68] A. Jakobsson, S. L. Marple, Jr. and P. Stoica, "Computationally Efficient Two-Dimensional Capon Spectrum Analysis," *IEEE Trans. on Signal Processing*, vol. 48, no. 9, pp. 2651-2661, Sept. 2000.

[69] F. Lombardini, M. Montanari and F. Gini, "Reflectivity Estimation for Multibaseline Interferometric Radar Imaging of Layover Extended Sources," *IEEE Trans. on Signal Processing*, vol. 51, no. 6, pp. 1508-1519, June 2003.

[70] E. G. Larsson and J. Li, "Spectral Analysis of Periodically Gapped Data," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 39, no. 3, pp 1089-1097, July 2003.

[71] H. Li, J. Li and P. Stoica, "Performance Analysis of Forward-Backward Matched-Filterbank Spectral Estimators," *IEEE Trans. on Signal Processing*, vol. 46, no. 7, pp. 1954-1966, July 1998.

[72] M. R. Palsetia and J. Li, "Using APES for Interferometric SAR Imaging," *IEEE Trans. on Image Processing*, vol. 7, no. 9, pp. 1340-1353, Sept. 1998.

[73] E. G. Larsson and P. Stoica, "Fast Implementation of Two-Dimensional APES and Capon Spectral Estimators," *IEEE Conf. on Acoustics, Speech and Signal Processing*, vol. 5, pp. 3069-3072, May 2001.

[74] F. J. Frigo, J. A. Heinen, J. A. Hopkins, T. Niendorf and B. J. Mock, "Using Peak-Enhanced 2D-Capon Analysis with Single Voxel Proton Magnetic Resonance Spectroscopy to Estimate T2* for Metabolites," *Proc. of ISMRM*, vol. 12, p. 2437, 2004.

[75] C. A. Balanis, *Advanced Engineering Electromagnetics*, John Wiley & Sons, New York, NY, 1989.

[76] D. Spielman, P. Webb and A. Macovski, "A Statistical Framework for in Vivo Spectroscopic Imaging," *Journal of Magnetic Resonance*, vol. 79, pp. 66-77, 1988.

[77] T. V. Sreenivas and R. J. Niederjohn, "Zero-Crossing Based Spectral Analysis and SVD Spectral Analysis for Formant Frequency Estimation in Noise," *IEEE Trans. on Signal Processing*, vol. 40, no. 2, pp. 282-293, Feb. 1992.

[78] D. D. Stark, W. G. Bradley, G. B. Matson, M. W. Weiner, et al., *Magnetic Resonance Imaging, Volume I – third edition*, Mosby, Chicago, IL, pp. 181-214, 1999.

[79] M. W. Weiner, "Clinical Assessment of Ischemia and Malignancy with Magnetic Resonance Spectroscopy," *IEEE Conf. on Engineering in Medicine and Biology*, p. 315, 1988.

[80] I. Mader, J. Karitsky, et al., "Proton MRS in Kennedy Disease: Absolute Metabolite and Macromolecular Concentrations," *Journal of Magnetic Resonance Imaging*, vol. 16, pp. 160-167, 2002.

[81] K. Ugurbil, et al., "Magnetic Resonance Imaging of Brain Function and Neurochemistry," *Proc. of IEEE*, vol. 89, no. 7, pp. 1093-1106, July 2001.

[82] N. Binesh, K. Yue, L. Fairbanks and M. A. Thomas, "Reproducibility of Localized 2D Correlated MR Spectroscopy," *Magnetic Resonance in Medicine*, vol. 48, pp. 942-948, 2002.

[83] M. A. McLean, A. L. Busza, L. L. Wald, R. J. Simister, G. J. Barker and S. R. Williams, "In Vivo GABA+ Measurement at 1.5T Using a PRESS-Localized Double Quantum Filter," *Magnetic Resonance in Medicine*, vol. 48, pp. 233-241, 2002.

[84] I. Asllani, E. Shankland, T. Pratum and M. Kushmerick, "Double Quantum Filtered ^{1}H NMR Spectroscopy Enables Quantification of Lactate in Muscle," *Journal of Magnetic Resonance*, vol. 152, pp. 195-202, 2001.

[85] N. Weiskopf, U. Klose, N. Birbaurmer and K. Mathiak, "Single Line Imaging Spectroscopy (SLIMS): Exploring the fMRI Signal," *Proc. of ISMRM*, vol. 11, p. 1760, 2003.

[86] R. E. Hurd, N. Sailasuta, R. Srinivansan, D. B. Vigneron and S. Nelson, "3T Brain Spectroscopy: Repeatability and Inter-subject Variability," *Proc. of ISMRM*, vol. 11, p. 1145, 2003.

[87] P. G. Mullins, L. Rowland, J. Bustillo, E. J. Bedrick, J. Lauriello and W. M. Brooks, "Reproducibility of ^{1}H-MRS Measurements in Schizophrenic Patients," *Magnetic Resonance in Medicine*, vol. 50, pp. 704-707, 2003.

[88] B. S. Y. Li, J. S. Babb, B. J. Soher, A. A. Maudsley and O. Gonen, "Reproducibility of 3D Proton Spectroscopy in the Human Brain," *Magnetic Resonance in Medicine*, vol. 47, pp. 437-446, 2002.

[89] B. G. Goodyear, A. M. Demchuk and R. Frayne,"T_2^* Heterogeneity in Cerebral Ischemia: Implications for fMRI Interpretation," *Proc. of ISMRM*, vol. 11, p. 1807, 2003.

[90] S. Michaeli, et al., "Proton T2 Relaxation Study of Water, N-acetylaspartate, and Creatine in Human Brain using Hahn and Carr-Purcell Spin Echoes at 4T and 7T," *Magnetic Resonance in Medicine*, vol. 47, pp. 629-633, 2002.

[91] K. S. Opstad, J. R. Griffiths, B. A. Bell and F. A. Howe, "In Vivo Lipid T2 Relaxation Time Measurements in High-Grade Tumors: Differentiation of Glioblastomas and Metastases," *Proc. of ISMRM*, vol. 11, p. 754, 2003.

[92] M. Singh, P. Patel and D. Khosla, "Estimation of T2* in Functional Spectroscopy during Visual Stimulation," *IEEE Trans. on Nuclear Science*, vol. 43, no. 3, June 1996.

[93] X. Golay, J. Gillen, P. C. M van Zilj and P. B. Barker, "Scan Time Reduction in Proton Magnetic Resonance Spectroscopic Imaging of the Human Brain," *Magnetic Resonance in Medicine*, vol. 47, pp. 384-387, 2002.

[94] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*, John Wiley & Sons, Inc., New York, NY, 1996.

[95] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[96] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1975.

[97] S. L. Marple, Jr., *Digital Spectral Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[98] S. S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[99] J. J. K. Ó Ruanaidh, and W. J. Fitzgerald, *Numerical Bayesian Methods Applied to Signal Processing*, Springer, New York, NY, 1996.

[100] E. R. Davies, *Electronics, Noise and Signal Recovery*, Academics Press, New York, NY, 1993

[101] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston, MA, 1992.

[102] M. Bellanger, *Digital Processing of Signals*, John Wiley & Sons, New York, NY, 1989.

Appendix

A.1 Quadratic Minimization

The following result is adapted from a development in Stoica and Moses[60] (see also [94]-[102]).

A.1.1 Theorem

Assume that R is a complex Hermitian (i.e., $R^H = R$) positive definite $M \times M$ matrix and $\bar{h}$ and $\bar{s}$ are complex $M \times 1$ vectors. Then the unique vector $\bar{h}_0$ that minimizes

$$J(\bar{h}) = \bar{h}^H R \bar{h} \tag{A.1}$$

over $\bar{h}$, subject to the condition

$$\bar{h}^H \bar{s} = 1 \tag{A.2}$$

is given by

$$\bar{h}_0 = \frac{R^{-1}\bar{s}}{\bar{s}^H R^{-1}\bar{s}} \tag{A.3}$$

A.1.2 Proof

We first note that

$$\bar{h}_0^H \bar{s} = \frac{\bar{s}^H R^{-1} \bar{s}}{\bar{s}^H R^{-1} \bar{s}} = 1 \tag{A.4}$$

verifying that $\bar{h}_0$ satisfies Eqn. (A.2). Now, for an arbitrary vector, $\bar{h}$, we may write

$$\bar{h} = \bar{h}_0 + \bar{\delta} \tag{A.5}$$

For our purposes we only need to consider vectors $\bar{h}$ that satisfy Eqn. (A.2). Thus, since

$$\bar{h}^H \bar{s} = \bar{h}_0^H \bar{s} + \bar{\delta}^H \bar{s} = 1 \tag{A.6}$$

it follows from Eqn. (A.4) that

$$\bar{\delta}^H \bar{s} = 0 \tag{A.7}$$

Now, from Eqn. (A.1) and Eqn. (A.5),

$$\begin{aligned} J\left(\bar{h}\right) &= \left(\bar{h}_0^H + \bar{\delta}^H\right) R \left(\bar{h}_0 + \bar{\delta}\right) \\ &= \bar{h}_0^H R \bar{h}_0 + \bar{\delta}^H R \bar{h}_0 + \bar{h}_0^H R \bar{\delta} + \bar{\delta}^H R \bar{\delta} \\ &= J\left(\bar{h}_0\right) + 2\bar{\delta}^H R \bar{h}_0 + \bar{\delta}^H R \bar{\delta} \\ &= J\left(\bar{h}_0\right) + 2\bar{\delta}^H R \frac{R^{-1}\bar{s}}{\bar{s}^H R^{-1} \bar{s}} + \bar{\delta}^H R \bar{\delta} \\ &= J\left(\bar{h}_0\right) + 2\frac{\bar{\delta}^H \bar{s}}{\bar{s}^H R^{-1} \bar{s}} + \bar{\delta}^H R \bar{\delta} \end{aligned} \tag{A.8}$$

From Eqn. (A.7), the center term in Eqn. (A.8) is zero, resulting in

$$J\left(\bar{h}\right) = J\left(\bar{h}_0\right) + \bar{\delta}^H R \bar{\delta} \tag{A.9}$$

Since R is positive definite, the second term in Eqn. (A.9) is always greater than or equal to 0, and equal to 0 if and only if $\bar{\delta} = 0$. That is, using Eqn. (A.5), we have that Eqn. (A.9) is a minimum if and only if $\bar{h} = \bar{h}_0$, thus completing the proof.

A.2 Matrix Inversion Lemma

The following result is adapted from a development in Stoica and Moses[60] (see also [94]-[102]).

A.2.1 Theorem

Assume R is a non-singular complex $M \times M$ matrix, $\bar{X}$ is a complex $M \times 1$ vector, and a is a non-zero complex scalar. Then, if

$$Q = R - \frac{1}{a}\bar{X}\bar{X}^H \tag{A.10}$$

the inverse of Q may be computed as

$$Q^{-1} = R^{-1} + \frac{R^{-1}\bar{X}\bar{X}^H R^{-1}}{a - \bar{X}^H R^{-1}\bar{X}} \tag{A.11}$$

A.2.2 Proof

From Eqn. (A.10) and Eqn. (A.11) we have

$$
\begin{aligned}
QQ^{-1} &= \left(R - \frac{1}{a}\bar{X}\bar{X}^{H} \right)\left(R^{-1} + \frac{R^{-1}\bar{X}\bar{X}^{H}R^{-1}}{a - \bar{X}^{H}R^{-1}\bar{X}} \right) \\
&= RR^{-1} - \frac{1}{a}\bar{X}\bar{X}^{H}R^{-1} + \frac{RR^{-1}\bar{X}\bar{X}^{H}R^{-1}}{a - \bar{X}^{H}R^{-1}\bar{X}} - \frac{1}{a}\frac{\bar{X}\bar{X}^{H}R^{-1}\bar{X}\bar{X}^{H}R^{-1}}{a - \bar{X}^{H}R^{-1}\bar{X}} \\
&= I - \frac{1}{a}\bar{X}\bar{X}^{H}R^{-1} + \frac{1}{a}\frac{a\bar{X}\bar{X}^{H}R^{-1}}{a - \bar{X}^{H}R^{-1}\bar{X}} - \frac{1}{a}\frac{\bar{X}\bar{X}^{H}R^{-1}\bar{X}\bar{X}^{H}R^{-1}}{a - \bar{X}^{H}R^{-1}\bar{X}} \\
&= I - \frac{1}{a}\bar{X}\bar{X}^{H}R^{-1} + \frac{1}{a}\frac{\bar{X}a\bar{X}^{H}R^{-1} - \bar{X}\left(\bar{X}^{H}R^{-1}\bar{X}\right)\bar{X}^{H}R^{-1}}{a - \bar{X}^{H}R^{-1}\bar{X}} \qquad \text{(A.12)} \\
&= I - \frac{1}{a}\bar{X}\bar{X}^{H}R^{-1} + \frac{1}{a}\frac{\bar{X}\left(a - \bar{X}^{H}R^{-1}\bar{X}\right)\bar{X}^{H}R^{-1}}{\left(a - \bar{X}^{H}R^{-1}\bar{X}\right)} \\
&= I - \frac{1}{a}\bar{X}\bar{X}^{H}R^{-1} + \frac{1}{a}\bar{X}\bar{X}^{H}R^{-1} \\
&= I
\end{aligned}
$$

Thus Q^{-1} as given by Eqn. (A.11) is indeed the inverse of Q given in Eqn. (A.10), completing the proof. Note that if R^{-1} is known then the computational task of determining the inverse of Q is replaced by several matrix multiplications and one scalar division.

A.3 Calculation of SNR

An important consideration when evaluating the fidelity of a sampled signal is its *signal-to-noise ratio* (SNR)[60], [94]- [102]. For clarity, we provide the details of how we define and compute SNR in our simulations.

Given a *clean signal,* $s[n]$, where $n = 0,1,\cdots,N_f - 1$, and a corresponding *base noise signal,* $n_0[n]$, where $n = 0,1,\cdots,N_f - 1$ we find it convenient to multiply $n_0[n]$ by a scalar weighting factor, w, to adjust $n_0[n]$ in a manner that provides the desired SNR.

$$n_a[n] = w \cdot n_0[n] \qquad \text{for } n = 0,1,\cdots,N_f - 1 \tag{A.13}$$

The signal energy may be defined as

$$E_s = \sum_{n=0}^{N_f-1} \left| s^2[n] \right| \tag{A.14}$$

and the base noise energy may be defined as

$$E_{n_0} = \sum_{n=0}^{N_f-1} \left| {n_0}^2[n] \right| \tag{A.15}$$

Also, the actual noise energy may be defined as

$$E_{n_a} = \sum_{n=0}^{N_f-1} \left| {n_a}^2[n] \right| \tag{A.16}$$

The SNR (in dB) is defined as

$$SNR = 10\log_{10}\left(\frac{E_s}{E_{n_a}}\right) \tag{A.17}$$

Combining Eqn. (A.13) and Eqn. (A.16) we have

$$E_{n_a} = \sum_{n=0}^{N_f-1}\left|{n_a}^2[n]\right| = \sum_{n=0}^{N_f-1}\left|w^2 n_0^2[n]\right| = w^2\sum_{n=0}^{N_f-1}\left|{n_0}^2[n]\right| = w^2 E_{n_0} \tag{A.18}$$

Then using Eqn. (A.18) and Eqn. (A.17) we find w as follows:

$$SNR = 10\log_{10}\left(\frac{E_s}{w^2 E_{n_0}}\right) \tag{A.19}$$

$$\frac{SNR}{10} = \log_{10}\left(\frac{E_s}{w^2 E_{n_0}}\right) \tag{A.20}$$

$$10^{\left(SNR/10\right)} = \frac{E_s}{w^2 E_{n_0}} \tag{A.21}$$

$$w^2 = \frac{E_s}{E_{n_0}10^{\left(SNR/10\right)}} \tag{A.22}$$

$$w = \sqrt{\frac{E_s}{E_{n_0}10^{\left(SNR/10\right)}}} \tag{A.23}$$

Eqn. (A.23) thus provides the value of w needed to produce the desired SNR.

A.4 MATLAB Code for Computing MRS Absorption Spectra

A.4.1 phascor.m

```
% Spectroscopy - Generate Phase correction coefficients
% Marquette University,   Milwaukee, WI  USA
% Copyright 2001, 2002, 2003, 2004 - All rights reserved.
% Fred J. Frigo
%
% Feb 15, 2001  - Original
% June 26, 2001 - Accept Pfile name as argument
% Sept 3, 2001  - Use Equiripple FIR filter for linear phase correction
% Oct 15, 2001  - Open Pfile directly instead of intermediate file.
%                 add ability to return an array of vectors.
% Dec 8, 2001   - Added DeBoor spline smoothing.
% Feb 11, 2002  - Added multi-channel support
% Jul 30, 2002  - Return Max reference value for multi-channel scaling
% Jun 16, 2003  - Added support for Pfile format for MGD2 / 11.0
% Mar 5, 2004   - Plot enhancements
%
%

function [pcor_vector, ref_vector, ref_scale] =  phascor( pfilename, channel_num )
i = sqrt(-1);

% set flag to 1 to plot intermediate results
save_plot = 0;

% Open Pfile to read reference scan data.
fid = fopen(pfilename,'r', 'ieee-be');
if fid == -1
    err_msg = sprintf('Unable to locate Pfile %s', pfile)
    return;
end

% Determine size of Pfile header based on Rev number
status = fseek(fid, 0, 'bof');
[f_hdr_value, count] = fread(fid, 1, 'real*4');
rdbm_rev_num = f_hdr_value(1);
if( rdbm_rev_num == 7.0 )
    pfile_header_size = 39984;  % LX
    bandwidth_index = 9839;
elseif ( rdbm_rev_num == 8.0 )
    pfile_header_size = 60464;  % Cardiac / MGD
    bandwidth_index = 14959;
elseif ( rdbm_rev_num == 5.0 )
    pfile_header_size = 39940;  % Signa 5.5
    bandwidth_index = 9839;    % ??
else
    % In 11.0 (ME2) the header and data are stored as little-endian
    fclose(fid);
    fid = fopen(pfilename,'r', 'ieee-le');
    status = fseek(fid, 0, 'bof');
    [f_hdr_value, count] = fread(fid, 1, 'real*4');
    if (f_hdr_value == 9.0)
        pfile_header_size= 61464;
        bandwidth_index = 14959;
    else
        err_msg = sprintf('Invalid Pfile header revision: %f', f_hdr_value )
        return;
    end
end
```

```
status = fseek(fid, 0, 'bof');
[hdr_value, count] = fread(fid, 52, 'integer*2');
nex = hdr_value(37);
nframes = hdr_value(38);
da_xres = hdr_value(52);

% Read 'user19' CV  - number of reference frames
status = fseek(fid, 0, 'bof');
[f_hdr_value, count] = fread(fid, 74, 'real*4');
num_ref_frames = f_hdr_value(74);

% Read the reference frames of data
frame_size = 2*da_xres*4;
baseline_size = frame_size;
channel_size = (nframes + 1)*frame_size;
data_offset = pfile_header_size + (channel_size*(channel_num - 1)) + baseline_size;
status = fseek(fid, data_offset, 'bof');

ref_data_elements = 2*da_xres*num_ref_frames;
[raw_data, count] = fread(fid, ref_data_elements, 'integer*4');
fclose(fid);

% Store the reference frames in the ref_frames array
vector_size = da_xres;
ref_frames=[];
vtmp = [1:vector_size];

for j = 1:num_ref_frames
    vector_offset = vector_size*2*(j-1);
    for k = 1:vector_size
        vtmp(k) = raw_data((vector_offset + k*2)-1)+ (raw_data(vector_offset + k*2)*i);
    end
    ref_frames(j,:)=vtmp;
end

ref_size = da_xres;
vtmp = 0.0;
% Average the reference data frames
for j = 1:num_ref_frames
   vtmp = vtmp + ref_frames(j,:);
end
ref = vtmp / num_ref_frames;

% return the averaged reference vector
ref_vector = ref;

% Plot Input Reference data
if (save_plot == 1)
   plot_complex( 'Averaged reference data', ref); % fig 8
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% normalize ref data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ref_scale = max( abs(ref) );
ref_norm = ref / ref_scale;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DC mixing
%   Find the largest frequency component
%   Create a sinusoid of same frequency and opposite phase?
%   Multiply by the sinusoid to cancel out this LARGE freq?
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Take FFT of Average Ref Scan data
```

```
ref_ft=fft(ref_norm);

% Find Max of FFT'd data
%   We dont really want to get the last half of points though.
%   Perhaps we can multiply by an alteration vector to look at
%   just the center points?
%
[refmax, index] = max(ref_ft(1:ref_size-8));
max_index = sprintf('Max freq weight in ref scan frames is %d', index);

% Generate ramp vector, muliply by index of max value
dc = linspace(0.0, (-2.0*pi) , ref_size);
dc = dc.*index;

% Create sinusoid with pure frequency
cos_dc = cos(dc);
sin_dc = sin(dc);

% DC mixing
corr = cos_dc + sin_dc*i;
ref_raw = ref_norm.*corr;

% Plot corrected Reference data,  and phase correction vector
if (save_plot == 1)
    plot_complex('Phase correction vector after DC mixing', corr); % fig 9
    plot_complex('Reference data after DC mixing', ref_raw); % fig 10
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Zero phasing
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
zeroterm =  ref_raw(1);
% Complex conjugate of the first point in the DC corrected ref frame
zeroterm = real(zeroterm) - imag(zeroterm)*i;

% The phase angle is now zero for the first point in the ref frame
ref_raw = ref_raw * zeroterm;
corr = corr * zeroterm;

% Plot corrected Reference data,  and phase correction vector
if (save_plot == 1)
    plot_complex('Phase correction vector after zero phase adjustment', corr); % fig 12
    plot_complex('Reference data after zero phase adjustment', ref_raw); % fig 11
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Linear phase correction factor
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calculate phase of ref vector
ref_ang = angle(ref_raw);

% Unwrap Phase of reference frame
unwr_ref = unwrap( ref_ang);

% Plot phase of reference data, and unwrapped phase
if (save_plot == 1)
    plot_2_real('Phase \phi_z_p[n] (radians)', ref_ang, 'Unwrapped phase \phi_z_p[n]
(radians)', unwr_ref); % fig 13
end

% Add up how many periods are present in the ref frame
pscale = unwr_ref(ref_size);
```

```
% Generate linear phase vector, muliply by unwrapped phase
ramp = linspace(0.0, -1 , ref_size);
lin_phas = pscale.*ramp;

cos_linp = cos( lin_phas);
sin_linp = sin( lin_phas);

lp_corr = cos_linp + (i*sin_linp);

% Apply linear phase vector to reference frame and to phase corr vector
ref_raw = lp_corr.*ref_raw;
corr = lp_corr.*corr;

% Plot corrected Reference data,  and phase correction vector
if (save_plot == 1)
    plot_complex('Linear phase correction vector', lp_corr); % fig 14
    plot_complex('Phase correction vector after linear phase correction', corr); % fig 16
    plot_complex('Reference data after linear phase correction', ref_raw); % fig 15
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Smooth the phase of the Ref Frame
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ref_phas = angle( ref_raw );
uref_phas = unwrap(ref_phas);

% compute e**( -0.25*ln(abs(ref_raw))
mag_raw = abs(ref_raw);
ln_raw = -0.25*log(mag_raw);
dy = exp(ln_raw);

smooth_factor = 0.9999;

% Spline smoothing (DeBoor)
filt_phas = smooth_spline( uref_phas, dy, ref_size, smooth_factor);

% Plot unwrapped phase of phase corrected reference data, and smoothed phase
if (save_plot == 1)
    plot_2_real('Unwrapped phase \phi_lp[n] (radians)', uref_phas, 'Unwrapped phase
\phi_s[n] (radians)', filt_phas); % fig 17
end

% Generate sinusoidal waveform based on smoothed phase
filt_phas = -1.0*filt_phas;
cos_fphs = cos( filt_phas);
sin_fphs = sin( filt_phas);
fphs = cos_fphs + i.*sin_fphs;

% Final step.. Multiply by corr vector and by ref vector
ref_raw = ref_raw.*fphs;
corr = corr.*fphs;

% Plot corrected Reference data,  and phase correction vector
if (save_plot == 1)
    plot_complex('Phase correction vector', corr); % fig 19
    plot_complex('Reference data after phase correction', ref_raw); % fig 18
end

% return the phase correction vector
pcor_vector = corr;
```

A.4.2 plot_2_real.m

```
% plot_2_real -  Plots 2 real valued vectors
% Marquette University,   Milwaukee, WI  USA
% Copyright 2004 - All rights reserved.
% Fred J. Frigo
%
% Mar 5, 2004   - Original
%

function  plot_2_real( plot1_label, input_data1, plot2_label, input_data2 )

   % get size of vector
   vector_size = max(size(input_data1));
   x = [ 1:vector_size];

   figure;
   subplot(2,1,1);
   plot(x,input_data1,'k');
   ylabel(plot1_label);
   xlabel('time');

   if (nargin > 2)
      subplot(2,1,2);
      plot(x, input_data2, 'k');
      ylabel(plot2_label);
      xlabel('time');
   end
```

A.4.3 plot_complex.m

```
% plot_complex -  Plots real, imag and magnitude values of complex vector
% Marquette University,   Milwaukee, WI  USA
% Copyright 2004 - All rights reserved.
% Fred J. Frigo
%
% Mar 5, 2004   - Original
%
%

function  plot_complex( plot_title, input_data )

   % get size of vector
   vector_size = max(size(input_data));
   x = [ 1:vector_size];

   figure;
   subplot(3,1,1);
   plot(x,real(input_data),'k');
   title(plot_title);
   ylabel('Real');
   xlabel('time');

   subplot(3,1,2);
   plot(x, imag(input_data), 'k');
   ylabel('Imaginary');
   xlabel('time');

   subplot(3,1,3);
   plot(x, abs(input_data), 'k');
   ylabel('Magnitude');
   xlabel('time');
```

```
   % Entire phase correction vector has magnitude of 1.0
   % Check for entire vector with magnitude = 1.0
   if(( max(abs(input_data) < 1.0000001 )) && ...
      ( min(abs(input_data) > 0.999999  )))
      my_axis=axis;
      my_axis(3)=0.0;  % ymin
      my_axis(4)=1.5;  % ymax
      axis(my_axis);
   end
```

A.4.4 plotp.m

```
% plotp.m  -  plot raw data from Pfile
%
% Marquette University,   Milwaukee, WI  USA
% Copyright 2002, 2003 - All rights reserved.
% Fred Frigo
%
% Date:  Jan 21, 2002
%
%   - this is based on MATLAB code from David Zhu - checkRaw.m
%   - updated May 14, 2003 to support 11.0 (little endian format)
%

function plotp( pfile )

i = sqrt(-1);


if(nargin == 0)
    [fname, pname] = uigetfile('*.*', 'Select Pfile');

    pfile = strcat(pname, fname);
end

% Open Pfile to read reference scan data.
fid = fopen(pfile,'r', 'ieee-be');
if fid == -1
    err_msg = sprintf('Unable to locate Pfile %s', pfile)
    return;
end

% Determine size of Pfile header based on Rev number
status = fseek(fid, 0, 'bof');
[f_hdr_value, count] = fread(fid, 1, 'real*4');
rdbm_rev_num = f_hdr_value(1);
if( rdbm_rev_num == 7.0 )
    pfile_header_size = 39984;  % LX
elseif ( rdbm_rev_num == 8.0 )
    pfile_header_size = 60464;  % Cardiac / MGD
elseif (( rdbm_rev_num > 5.0 ) && (rdbm_rev_num < 6.0))
    pfile_header_size = 39940;  % Signa 5.5
else
    % In 11.0 (ME2) the header and data are stored as little-endian
    fclose(fid);
    fid = fopen(pfile,'r', 'ieee-le');
    status = fseek(fid, 0, 'bof');
    [f_hdr_value, count] = fread(fid, 1, 'real*4');
    if (f_hdr_value == 9.0)
        pfile_header_size= 61464;
    else
```

```
        err_msg = sprintf('Invalid Pfile header revision: %f', f_hdr_value )
        return;
    end
end

% Read header information
status = fseek(fid, 0, 'bof');
[hdr_value, count] = fread(fid, 102, 'integer*2');
npasses = hdr_value(33);
nslices = hdr_value(35);
nechoes = hdr_value(36);
nframes = hdr_value(38);
point_size = hdr_value(42);
da_xres = hdr_value(52);
da_yres = hdr_value(53);
rc_xres = hdr_value(54);
rc_yres = hdr_value(55);
start_recv = hdr_value(101);
stop_recv = hdr_value(102);
nreceivers = (stop_recv - start_recv) + 1;

% Determine number of slices in this Pfile:  this does not work for all cases.
slices_in_pass = nslices/npasses;

% Compute size (in bytes) of each frame, echo and slice
data_elements = da_xres*2;
frame_size = data_elements*point_size;
echo_size = frame_size*da_yres;
slice_size = echo_size*nechoes;
mslice_size = slice_size*slices_in_pass;

for k = 500:1000  % give a large number 1000 to loop forever
  % Enter slice number to plot
  if ( slices_in_pass > 1 )
      slice_msg = sprintf('Enter the slice number: [1..%d]',slices_in_pass);
      my_slice = input(slice_msg);
      if (my_slice > slices_in_pass)
          err_msg = sprintf('Invalid number of slices. Slice number set to 1.')
          my_slice = 1;
      end
  else
      my_slice = 1;
  end

  % Enter echo number to plot
  if ( nechoes > 1 )
      echo_msg = sprintf('Enter the echo number: [1..%d]',nechoes);
      my_echo = input(echo_msg);
      if (my_echo > nechoes )
          err_msg = sprintf('Invalid echo number. Echo number set to 1.')
          my_echo = 1;
      end
  else
      my_echo = 1;
  end

  % Enter receiver number to plot
  if ( nreceivers > 1 )
      recv_msg = sprintf('Enter the receiver number: [1..%d]',nreceivers);
      my_receiver = input(recv_msg);
      if (my_receiver > nreceivers)
          err_msg = sprintf('Invalid receiver number. Receiver number set to 1.')
          my_receiver = 1;
      end
  else
      my_receiver = 1;
```

```
  end

  % Enter the view number
  view_msg = sprintf('Enter the frame number (1 is baseline): [1..%d]',da_yres);
  my_frame = input(view_msg);
  if (my_frame > da_yres)
      err_msg = sprintf('Invalid frame number. Frame number set to 1.')
      my_frame = 1;
  end

  % Compute offset in bytes to start of frame.
  file_offset = pfile_header_size + ((my_slice - 1)*slice_size) + ...
                      + ((my_receiver -1)*mslice_size) + ...
                      + ((my_echo-1)*echo_size) + ...
                      + ((my_frame-1)*frame_size);

  status = fseek(fid, file_offset, 'bof');

  % read data: point_size = 2 means 16 bit data, point_size = 4 means EDR )
  if (point_size == 2 )
     [raw_data, count] = fread(fid, data_elements, 'integer*2');
  else
      [raw_data, count] = fread(fid, data_elements, 'integer*4');
  end

  for m = 1:da_xres
     frame_data(m) = raw_data((2*m)-1) + i*raw_data(2*m);
  end

  figure(k);
  subplot(3,1,1);
  plot(real(frame_data));
  title(sprintf('%s, slice %d, recv %d, echo %d, frame %d', fname, my_slice, my_receiver,
my_echo, my_frame));
  %title('Reference data');
  xlabel('time');
  ylabel('Real');
  subplot(3,1,2);
  plot(imag(frame_data));
  %title(sprintf('Imaginary Data'));
  xlabel('time');
  ylabel('Imaginary');
  subplot(3,1,3);
  plot(abs(frame_data));
  %title(sprintf('Magnitude Data'));
  xlabel('time');
  ylabel('Magnitude');


  % check to see if we should quit
  quit_answer = input('Press Enter to continue, "q" to quit:', 's');
  if ( size( quit_answer ) > 0 )
     if (quit_answer == 'q')
         break;
     end
  end

end
fclose(fid);
```

A.4.5 smooth_spline.m

```
% Spline smoothing  (DeBoor's algorithm)
% Marquette University,   Milwaukee, WI  USA
% Copyright 2001 - All rights reserved.
% Fred Frigo
%
% Dec 8, 2001
%
% Adapted to MATLAB from the following Fortran source file
%    found at http://www.psc.edu/~burkardt/src/splpak/splpak.f90

function spline_sig = smooth_spline( y, dx, npoint, smooth_factor)

p=smooth_factor;
a=[npoint:4];
v=[npoint:7];
a= 0.0;
v= 0.0;


%qty=[npoint:1];
%qu=[npoint:1];
%u=[npoint:1];

x = linspace(0.0, (npoint-1.0)/npoint , npoint);

% setupq
  v(1,4) = x(2)-x(1);

  for i = 2:npoint-1
    v(i,4) = x(i+1)-x(i);
    v(i,1) = dx(i-1)/v(i-1,4);
    v(i,2) = ((-1.0.*dx(i))/v(i,4)) - (dx(i)/v(i-1,4));
    v(i,3) = dx(i+1)/v(i,4);
  end


  v(npoint,1) = 0.0;
  for i = 2:npoint-1
    v(i,5) = (v(i,1)*v(i,1)) + (v(i,2)*v(i,2)) + (v(i,3)*v(i,3));
  end

  for i = 3:npoint-1
    v(i-1,6) = (v(i-1,2)*v(i,1)) + (v(i-1,3)*v(i,2));
  end

  v(npoint-1,6) = 0.0;

  for i = 4: npoint-1
    v(i-2,7) = v(i-2,3)*v(i,1);
  end

  v(npoint-2,7) = 0.0;
  v(npoint-1,7) = 0.0;
%!
%!  Construct  q-transp. * y  in  qty.
%!
  prev = (y(2)-y(1))/v(1,4);
  for i= 2:npoint-1
    diff = (y(i+1)-y(i))/v(i,4);
    %qty(i) = diff-prev;
    a(i,4) = diff - prev;
    prev = diff;
```

```
  end

% end setupq

%chol1d

%!
%!  Construct 6*(1-p)*q-transp.*(d**2)*q + p*r
%!
  six1mp = 6.0.*(1.0-p);
  twop = 2.0.*p;

  for i = 2: npoint-1
    v(i,1) = (six1mp.*v(i,5)) + (twop.*(v(i-1,4)) + v(i,4));
    v(i,2) = (six1mp.*v(i,6)) +( p.*v(i,4));
    v(i,3) = six1mp.*v(i,7);
  end

  if ( npoint < 4 )
    u(1) = 0.0;
    u(2) = a(2,4)/v(2,1);
    u(3) = 0.0;
%!
%!  Factorization
%!
  else
    for i = 2: npoint-2;
      ratio = v(i,2)/v(i,1);
      v(i+1,1) = v(i+1,1)-(ratio.*v(i,2));
      v(i+1,2) = v(i+1,2)-(ratio.*v(i,3));
      v(i,2) = ratio;
      ratio = v(i,3)./v(i,1);
      v(i+2,1) = v(i+2,1)-(ratio.*v(i,3));
      v(i,3) = ratio;
    end
%!
%!  Forward substitution
%!
    a(1,3) = 0.0;
    v(1,3) = 0.0;
    a(2,3) = a(2,4);
    for i = 2: npoint-2
      a(i+1,3) = a(i+1,4) - (v(i,2)*a(i,3)) - (v(i-1,3)*a(i-1,3));
    end
%!
%!  Back substitution.
%!
    a(npoint,3) = 0.0;
    a(npoint-1,3) = a(npoint-1,3) / v(npoint-1,1);

    for i = npoint-2:-1:2
      a(i,3) = (a(i,3)/v(i,1)) - (a(i+1,3)*v(i,2)) - (a(i+2,3)*v(i,3));
    end

  end
%!
%!  Construct Q*U.
%!
  prev = 0.0;
  for i = 2: npoint
    a(i,1) = (a(i,3)-a(i-1,3))/v(i-1,4);
    a(i-1,1) = a(i,1)-prev;
    prev = a(i,1);
  end

  a(npoint,1) = -1.0.*a(npoint,1);
```

```
%end chol1d

  for i = 1: npoint
    spline_sig(i) = y(i)-(6.0.*(1.0-p).*dx(i).*dx(i).*a(i,1));
  end

%  for i = 1: npoint
%    a(i,3) = 6.0*a(i,3)*p;
%  end

%  for i = 1: npoint-1
%    a(i,4) = (a(i+1,3)-a(i,3))/v(i,4);
%    a(i,2) = (a(i+1,1)-a(i,1))/v(i,4)-(a(i,3)+a(i,4)/3.*v(i,4))/2.*v(i,4);
%  end
```

A.4.6 spectro.m

```
% Spectroscopy Mainline
% Marquette University,   Milwaukee, WI  USA
% Copyright 2001, 2002, 2003 - All rights reserved.
% Fred J. Frigo
%
%
% This function calls other MR spectroscopy related functions
% to compute results from the given raw data file (Pfile).
%
% Oct 15, 2001 - Original
% Feb 11, 2002 - Multi-channel spectro
% July 30, 2002 - Combine multi-channel results using reference weighting
% Jan 1, 2003   - Label PPM axis
% June 16, 2003 - Pfile updates for MGD2 / 11.0

function spectro( pfilename )

% Check to see if pfile name was passed in
if ( nargin == 0 )
   % Enter name of Pfile
   [fname, pname] = uigetfile('*.*', 'Select Pfile');
   pfilename = strcat(pname, fname);
end

% Open Pfile to read reference scan data.
fid = fopen(pfilename,'r', 'ieee-be');
if fid == -1
    err_msg = sprintf('Unable to locate Pfile %s', pfile)
    return;
end

% Determine size of Pfile header based on Rev number
status = fseek(fid, 0, 'bof');
[f_hdr_value, count] = fread(fid, 1, 'real*4');
rdbm_rev_num = f_hdr_value(1);
if( rdbm_rev_num == 7.0 )
    pfile_header_size = 39984;  % LX
elseif ( rdbm_rev_num == 8.0 )
    pfile_header_size = 60464;  % Cardiac / MGD
elseif (( rdbm_rev_num > 5.0 ) && (rdbm_rev_num < 6.0))
    pfile_header_size = 39940;  % Signa 5.5
else
    % In 11.0 (ME2) the header and data are stored as little-endian
```

```
    fclose(fid);
    fid = fopen(pfilename,'r', 'ieee-le');
    status = fseek(fid, 0, 'bof');
    [f_hdr_value, count] = fread(fid, 1, 'real*4');
    if (f_hdr_value == 9.0)
        pfile_header_size= 61464;
    else
        err_msg = sprintf('Invalid Pfile header revision: %f', f_hdr_value )
        return;
    end
end

% Read header to determine number of channels
status = fseek(fid, 0, 'bof');
[hdr_value, count] = fread(fid, 102, 'integer*2');
da_xres = hdr_value(52);
start_recv = hdr_value(101);
stop_recv = hdr_value(102);
nreceivers = (stop_recv - start_recv) + 1;

% some other useful scan parameters.
nex = hdr_value(37);

% Read 'user07' CV  - temperature in degree C
status = fseek(fid, 0, 'bof');
[f_hdr_value, count] = fread(fid, 62, 'real*4');
tempC = f_hdr_value(62)
fclose(fid);


% Index for results  (Right Hand Side of water peak)
start = round(da_xres*0.025);
stop = round(da_xres*0.25);

% Create PPM axis:   Must shift spectrum for temperature
ppm_start_37C = -4.25;
ppm_stop_37C = 0.30;
ppm_per_degree_C=0.01;
ppm_offset = (tempC-37)*ppm_per_degree_C;
ppm_start = ppm_start_37C + ppm_offset;
ppm_stop = ppm_stop_37C + ppm_offset;
combine_x = linspace(ppm_start,ppm_stop,(stop-start+1));

% Loop to compute phase correction vector for each receiver
for loop = 1:nreceivers
   % Compute phase correction results
   [pcor_vector, ref_vector, receiver_weight(loop)] = phascor( pfilename , loop);

   % Compute spectroscopy results
   results(loop,:) = spectro_proc( pcor_vector, ref_vector, pfilename, loop );

   % Plot results for each channel
   figure(100);
   subplot(nreceivers,1,loop);
   plot( combine_x, real( results(loop,start:stop)), 'k' );
   mesh_results(loop,:)=real(results(loop,start:stop));

   if( loop == 1)
       my_string= sprintf('Spectro results for: %s ',fname);
       %title( my_string);
       xlabel('ppm');
       if (nreceivers == 1)
          ylabel('Absorption');
       end
       set(gca,'XTick',-4.0:0.5:0.0);
       set(gca,'XTickLabel',{'4.0','3.5','3.0','2.5','2.0','1.5','1.0','0.5','0.0'});
```

```
   end

end

% Calculate combined results if more than one receiver
if nreceivers > 1
  % Find receiver with strongest signal (using Max reference value)
  [max_weight, strongest_receiver] = max(receiver_weight);

  % Dont use channels whose Max is lower than the threshold
  receiver_threshold = 0.05*max_weight;   % 0.45 default
  combined_weight = 0.0;
  receivers_to_use = 0;
  for loop = 1:nreceivers
    if receiver_weight(loop) >receiver_threshold
       receiver_to_use = receivers_to_use + 1;
       combined_weight = combined_weight + receiver_weight(loop);
    end
  end

  % Linear weighted combination
  accum_results = zeros(size(results(1,:)));
  for loop = 1:nreceivers
    if receiver_weight(loop) >receiver_threshold
       weight = receiver_weight(loop) / combined_weight;
       accum_results = accum_results + (weight.*real(results(loop,:)));
    else
       weight = 0.0;
    end
    weight
  end
  combined_results = accum_results;

  % Plot combined results
  figure;
  plot( combine_x, real( combined_results(start:stop)), 'k' );
  my_string= sprintf('Combined spectro results for: %s ',fname);
  title( my_string);
  xlabel('ppm');
  ylabel('Absorption');
  set(gca,'XTick',-4.0:0.5:0.0);
  set(gca,'XTickLabel',{'4.0','3.5','3.0','2.5','2.0','1.5','1.0','0.5','0.0'});


  % Plot multiple channel results
  figure;
  surf(combine_x, 1:1:8, mesh_results);
  set(gca,'XTick',-4.0:1.0:0.0);
  set(gca,'XTickLabel',{'4.0','3.0','2.0','1.0','0.0'});
  xlabel('ppm');
  zlabel('Absorption');
  ylabel('receive coil');

end
```

A.4.7 spectro_proc.m

```
% Spectroscopy - Phase correct, water subtract and Fourier Transform
% Marquette University,   Milwaukee, WI  USA
% Copyright 2000, 2001, 2002, 2003, 2004 - All rights reserved.
% Fred Frigo
%
```

```
% This function generates MR spectroscopy results from:
% the phase correction vector, pc, (computed by the phascor function)
% the averaged reference data, ref, (computed by the phascor funtion)
% the name (Pfile) of the raw data file containing water suppressed data.
% and the receiver number, channel_num.
%
%
% Dec 28, 2000 - Original
% May 2, 2001  - updates to plot intermediate results
% June 26, 2001 - updates to accept Pfile name as arg
% Oct 15, 2001 - updates to read from Pfile directly, plus new args
% Feb 11, 2002 - modified for multi-channel spectro
% Nov 2, 2003  - Updates to read 5.X and 11.0 format files
% Mar 8, 2004  - Plot enhancements
%

function results = spectro_proc ( pc, ref, pfilename, channel_num)


i = sqrt(-1);

% flag set to 1 for plots of intermediate results
save_plot = 0;

% Open Pfile to read reference scan data.
fid = fopen(pfilename,'r', 'ieee-be');
if fid == -1
    err_msg = sprintf('Unable to locate Pfile %s', pfilename)
    return;
end

% Determine size of Pfile header based on Rev number
status = fseek(fid, 0, 'bof');
[f_hdr_value, count] = fread(fid, 1, 'real*4');
rdbm_rev_num = f_hdr_value(1);
if( rdbm_rev_num == 7.0 )
    pfile_header_size = 39984;  % LX
elseif ( rdbm_rev_num == 8.0 )
    pfile_header_size = 60464;  % Cardiac / MGD
elseif ( rdbm_rev_num == 5.0 )
    pfile_header_size = 39940;  % Signa 5.5
else
    % In 11.0 (ME2) the header and data are stored as little-endian
    fclose(fid);
    fid = fopen(pfilename,'r', 'ieee-le');
    status = fseek(fid, 0, 'bof');
    [f_hdr_value, count] = fread(fid, 1, 'real*4');
    if (f_hdr_value == 9.0)
        pfile_header_size= 61464;
    else
        err_msg = sprintf('Invalid Pfile header revision: %f', f_hdr_value )
        return;
    end
end


status = fseek(fid, 0, 'bof');
[hdr_value, count] = fread(fid, 52, 'integer*2');
nex = hdr_value(37);
nframes = hdr_value(38);
da_xres = hdr_value(52);

% Read 'user19' CV  - number of reference frames
status = fseek(fid, 0, 'bof');
[f_hdr_value, count] = fread(fid, 74, 'real*4');
num_ref_frames = f_hdr_value(74);
```

```
% Read Pfile to get water supressed signal
ref_offset = 2*da_xres*(num_ref_frames+1)*4;
frame_size = 2*da_xres*4;
channel_size = (nframes + 1)*frame_size;
data_offset = pfile_header_size + (channel_size*(channel_num - 1)) + ref_offset;

status = fseek(fid, data_offset, 'bof');
num_sig_frames = nframes - num_ref_frames;
data_elements = 2*da_xres*num_sig_frames;
[raw_data, count] = fread(fid, data_elements, 'integer*4');
fclose(fid);

% Store the reference frames in the ref_frames array
vector_size = da_xres;
sig_frames=[];
vtmp = [1:vector_size];

for j = 1:num_sig_frames
    vector_offset = vector_size*2*(j-1);
    for k = 1:vector_size
        vtmp(k) = raw_data((vector_offset + k*2)-1)+ (raw_data(vector_offset + k*2)*i);
    end
    sig_frames(j,:)=vtmp;
end

vtmp = 0.0;
% Average the reference data frames
for j = 1:num_sig_frames
   vtmp = vtmp + sig_frames(j,:);
end

% Create averaged water suppressed signal vector.
sig = vtmp / num_sig_frames;

% For debug: store water-suppressed signal to file
recv_string = sprintf('%d', channel_num);
signal_file = strcat( pfilename, '.recv', recv_string,'.raw.dat');
fidref = fopen(signal_file, 'w+b');
for findex=1:da_xres
    fwrite(fidref, real(sig(findex)), 'real*4');
    fwrite(fidref, imag(sig(findex)), 'real*4');
end
fclose(fidref);


x=[1:da_xres];
% Plot input signal
if (save_plot == 1)
   plot_complex('Band-limited MR spectroscopy signal', sig);  % fig20
end

% Phase Correct Water supressed signal (sig) and Water signal (ref)
sig = pc.*sig;
ref = pc.*ref;

% Plot phase corrected signal and ref
if (save_plot == 1)
   plot_complex('Phase-corrected water-supressed data', sig);  %fig 21
   plot_complex('Phase corrected non-water-suppressed reference data', ref); %fig 22
end

% Subtract to 'signal' from 'water' obtain 'pure water'
pure_water = ref - sig;

% Plot pure water signal
```

```
if (save_plot == 1)
   plot_complex('Pure water', pure_water);  %fig23
end

% Negate every other element (this shifts the water peak to the center)
a_pure_wat = pure_water;
a_sig = sig;
for n = 1:(da_xres/2)
   a_pure_wat(2*n)= -1.0*a_pure_wat(2*n);
   a_sig(2*n) = -1.0*a_sig(2*n);
end

% Apodization Window
hanning_size = da_xres/1.6;
half_han_size = hanning_size/2;
win=hanning(hanning_size);
apod = linspace(0.0, 0.0, da_xres);
apod(1:half_han_size) = win((half_han_size+1):hanning_size);

% Plot apodization window
if (save_plot == 1)
   plot_2_real('w_1[n]',apod);  % fig24
end

% Apply apodization window to water and signal vectors
w_pure_wat = apod.*a_pure_wat;
w_sig = apod.*a_sig;

% Fourier Transform the apodized, alternated water and signal vectors
ft_wat = fft( w_pure_wat );
ft_sig = fft( w_sig);

% Plot Fourier Transform of Pure water and Signal
if (save_plot == 1)
   plot_2_real('Fourier transform of pure water, S_w[k]', abs(ft_wat), ...
               'Fourier transform of signal, S_s[k]', abs(ft_sig));  %fig 25
end

% Scale the pure water.
%    Assume water in signal and reference is the same.
%    Use 'real' coefficients in 16 element band near center
min_xres=(da_xres/2)-(da_xres/128);
max_xres=(da_xres/2)+(da_xres/128);
mag_wat = ft_wat(min_xres:max_xres);
mag_sig = ft_sig(min_xres:max_xres);
mag_wat = abs( real( mag_wat) );
mag_sig = abs( real( mag_sig) );
water_max = max( mag_wat );
sig_max = max( mag_sig );

% Scale the pure water so it can be subtracted off
scale = sig_max / water_max;
pure_water = scale.*pure_water;

% Subtract "scaled" pure water from signal.
pure_sig = sig - pure_water;

% Plot Pure Signal
if (save_plot == 1)
   plot_complex('Water-subtracted pure signal', pure_sig);  %fig 26
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% For debug: save phase-corrected, water-subtracted signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
recv_string = sprintf('%d', channel_num);
```

```
signal_file = strcat( pfilename, '.recv', recv_string,'.signal.dat');
fidsig = fopen(signal_file, 'w+b');
for findex=1:da_xres
    fwrite(fidsig, real(pure_sig(findex)), 'real*4');
    fwrite(fidsig, imag(pure_sig(findex)), 'real*4');
end
fclose(fidsig);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% For debug: save reference signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
recv_string = sprintf('%d', channel_num);
signal_file = strcat( pfilename, '.recv', recv_string,'.ref.dat');
fidref = fopen(signal_file, 'w+b');
for findex=1:da_xres
    fwrite(fidref, real(ref(findex)), 'real*4');
    fwrite(fidref, imag(ref(findex)), 'real*4');
end
fclose(fidref);


% Window for the "pure signal"
% win=hanning(1024);
win=hanning((da_xres*2));
awin = linspace(0.0, 0.0, da_xres);
% awin(1:512) = win(513:1024);
awin(1:da_xres) = win((da_xres+1):(da_xres*2));

if (save_plot == 1)
   plot_2_real('w_2[n]', awin);  % fig 27
end

% Apodize and zero pad the "pure signal" prior to the Fourier transform
zero_pad = 1;
a_pure_sig = linspace( 0.0, 0.0, (da_xres*2)*zero_pad);
a_pure_sig(1:da_xres) = awin.*pure_sig;
nmr_spect = fft( a_pure_sig );

if (save_plot == 1)  %fig 28
   plot_complex('Phase-corrected, apodized, water-suppressed signal with residual water
removed',a_pure_sig);
end

results = nmr_spect;
```

A.4.8 sraw_image.m

```
% sraw_image.m  -  plot raw data for a single echo from a Pfile
%
% Marquette University,   Milwaukee, WI  USA
% Copyright 2002, 2003 - All rights reserved.
% Fred Frigo
%
% Date:  Jan 21, 2002     - based  raw_image.m
%
%  - create 2 images, one for reference data, one for water supressed data
%

function sraw_image( pfile )

% Check to see if pfile name was passed in
if ( nargin == 0 )
   % Enter name of Pfile
```

```
   [fname, pname] = uigetfile('*.*', 'Select Pfile');
   pfile = strcat(pname, fname);
end

i = sqrt(-1);

% Open Pfile to read reference scan data.
fid = fopen(pfile,'r', 'ieee-be');
if fid == -1
    err_msg = sprintf('Unable to locate Pfile %s', pfile)
    return;
end

% Determine size of Pfile header based on Rev number
status = fseek(fid, 0, 'bof');
[f_hdr_value, count] = fread(fid, 1, 'real*4');
rdbm_rev_num = f_hdr_value(1);
if( rdbm_rev_num == 7.0 )
    pfile_header_size = 39984;  % LX
elseif ( rdbm_rev_num == 8.0 )
    pfile_header_size = 60464;  % Cardiac / MGD
elseif (( rdbm_rev_num > 5.0 ) && (rdbm_rev_num < 6.0))
    pfile_header_size = 39940;  % Signa 5.5
else
    % In 11.0 (ME2) the header and data are stored as little-endian
    fclose(fid);
    fid = fopen(pfile,'r', 'ieee-le');
    status = fseek(fid, 0, 'bof');
    [f_hdr_value, count] = fread(fid, 1, 'real*4');
    if (f_hdr_value == 9.0)
        pfile_header_size= 61464;
    else
        err_msg = sprintf('Invalid Pfile header revision: %f', f_hdr_value )
        return;
    end
end

status = fseek(fid, 0, 'bof');

% Read header information
[hdr_value, count] = fread(fid, 5122, 'integer*2');
nechoes = hdr_value(36);
nframes = hdr_value(38);
point_size = hdr_value(42);
da_xres = hdr_value(52);
da_yres = hdr_value(53);
rc_xres = hdr_value(54);
rc_yres = hdr_value(55);
start_recv = hdr_value(101);
stop_recv = hdr_value(102);
nreceivers = (stop_recv - start_recv) + 1;
slices_in_pass = hdr_value(5122);

% Read 'user19' CV  - number of reference frames
status = fseek(fid, 0, 'bof');
[f_hdr_value, count] = fread(fid, 74, 'real*4');
rdbm_rev_num = f_hdr_value(1);
num_ref_frames = f_hdr_value(74);


% Compute size (in bytes) of each frame, echo and slice
data_elements = da_xres*2*(da_yres-1);
frame_size = da_xres*2*point_size;
echo_size = frame_size*da_yres;
slice_size = echo_size*nechoes;
mslice_size = slice_size*nreceivers;
```

```
for k = 1:1000  % give a large number 1000 to loop forever

  % Enter slice number to plot
  my_slice = 1;
  if ( slices_in_pass > 1 )
      slice_msg = sprintf('Enter the slice number: [1..%d]',slices_in_pass);
      my_slice = input(slice_msg);
      if (my_slice > slices_in_pass)
          err_msg = sprintf('Invalid number of slices. Slice number set to 1.')
          my_slice = 1;
      end
  end

  % Enter echo number to plot
  my_echo = 1;
  if ( nechoes > 1 )
      echo_msg = sprintf('Enter the echo number: [1..%d]',nechoes);
      my_echo = input(echo_msg);
      if (my_echo > nechoes )
          err_msg = sprintf('Invalid echo number. Echo number set to 1.')
          my_echo = 1;
      end
  end

  % Enter receiver number to plot
  my_receiver = 1;
  if ( nreceivers > 1 )
      recv_msg = sprintf('Enter the receiver number: [1..%d]',nreceivers);
      my_receiver = input(recv_msg);
      if (my_receiver > nreceivers)
          err_msg = sprintf('Invalid receiver number. Receiver number set to 1.')
          my_receiver = 1;
      end
  end


  % Compute offset in bytes to start of frame.  (skip baseline view)
  file_offset = pfile_header_size + ((my_slice - 1)*mslice_size) + ...
                      + ((my_receiver -1)*slice_size) + ...
                      + ((my_echo-1)*echo_size) + ...
                      + (frame_size);

  status = fseek(fid, file_offset, 'bof');

  % read data: point_size = 2 means 16 bit data, point_size = 4 means EDR )
  if (point_size == 2 )
      [raw_data, count] = fread(fid, data_elements, 'integer*2');
  else
      [raw_data, count] = fread(fid, data_elements, 'integer*4');
  end

  %frame_data = zeros(da_xres);
  for j = 1:(da_yres -1)
     row_offset = (j-1)*da_xres*2;
     for m = 1:da_xres
        frame_data(j,m) = raw_data( ((2*m)-1) + row_offset) + i*raw_data((2*m) +
row_offset);
     end
  end

  figure(k);
  subplot(2,1,1);
  imagesc( abs(frame_data([1:num_ref_frames],:) ));
```

```
  %title(sprintf('Magnitude of Raw Reference Spectro Data, slice %d, recv %d, echo %d',
my_slice, my_receiver, my_echo));
  title('Magnitude of Raw Reference Data');
  xlabel('time');
  ylabel('frame number');
  subplot(2,1,2);
  imagesc( abs(frame_data([num_ref_frames+1:nframes-1],:)));
  title('Magnitude of Raw Water Suppressed Data');
  xlabel('time');
  ylabel('frame number');

  % check to see if we should quit
  quit_answer = input('Press Enter to continue, "q" to quit:', 's');
  if ( size( quit_answer ) > 0 )
     if (quit_answer == 'q')
         break;
     end
  end

end
fclose(fid);
```

A.5 MATLAB Code for 2D Spectral Estimation

A.5.1 APESCapon2D.m

```
% APESCapon2D.m  - weighted 2D APES / weighted 2D Capon
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
%  Fred J. Frigo, James A. Heinen
%
% Nw must be greater than or equal to N/2.
% k1 and k2 must satisfy 0<=k1<k2<=Nw-1.
% If bet1=0, only one matrix inversion is used.
% If bet1=-0.5 and bet2=0, only one fft is used.
% 0<gam<=1.  gam=1 reduces to 2D APES.  gam=0 would reduce to 2D Capon.
%

function S=APESCapon2D(x,N,M,Nsig,Nw,k1,k2,delsig,bet1,bet2,gam,C,g)

delw=pi/Nw;
x1=x(1,:);
xbb=x1(ones(M,1)*[1:N]+[0:M-1]'*ones(1,N));
if C>1
   for i=2:C
      xi=x(i,:);
      xbb=[xbb
         xi(ones(M,1)*[1:N]+[0:M-1]'*ones(1,N))];
   end
end
if bet1==0
   bet=bet2;
   Rinv=fRinv(xbb,N,M,bet,C);
end
if bet1==-0.5&bet2==0
   Xbb=fXbb(xbb,N,M,Nw,0,C);
end
for m=0:Nsig-1
   sig=m*delsig;
   if bet1~=0
      bet=bet1*sig+bet2;
      Rinv=fRinv(xbb,N,M,bet,C);
   end
   if bet1==-0.5&bet2==0
      L=fL(N,0.5*sig);
   else
      bet=bet1*sig+bet2;
      Xbb=fXbb(xbb,N,M,Nw,sig+2*bet,C);
      L=fL(N,sig+bet);
   end
   sbb0=exp(-sig*[0:M-1]'*ones(1,k2-k1+1)+j*delw*[0:M-1]'*[k1:k2]);
   sbb=g(1)*sbb0;
   if C>1
      for i=2:C
         sbb=[sbb
            g(i)*sbb0];
      end
   end
   for k=k1:k2
      Qinv=Rinv+gam*Rinv*Xbb(:,k+1)*Xbb(:,k+1)'*Rinv/...
         (L-gam*Xbb(:,k+1)'*Rinv*Xbb(:,k+1));
```

```
      S(m+1,k-k1+1)=sbb(:,k-k1+1)'*Qinv*Xbb(:,k+1)/...
         (L*sbb(:,k-k1+1)'*Qinv*sbb(:,k-k1+1));
   end
end
return

function Rinv=fRinv(xbb,N,M,bet,C)
e=repmat(exp(-bet*[0:N-1]),M*C,1);
xbbe=xbb.*e;
R=xbbe*xbbe';
Rinv=inv(R);
return

function Xbb=fXbb(xbb,N,M,Nw,sig,C)
e=repmat(exp(-sig*[0:N-1]),M*C,1);
xbbe=xbb.*e;
Xbb=fft(xbbe.',2*Nw).';
return

function L=fL(N,sig)
if sig==0
   L=N;
else
   L=(1-exp(-2*sig*N))/(1-exp(-2*sig));
end
return
```

A.5.2 Capon2D.m

```
% Capon2D.m - weighted 2D Capon
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
%  Fred J. Frigo, James A. Heinen
%
% K must be 1 or N.  alp1 and alp2 are irrelevant if K=1.
% Nw must be greater than or equal to N/2.
% k1 and k2 must satisfy 0<=k1<k2<=Nw-1.
% If bet1=0, only one matrix inversion is used.
% If alp1=-0.5 and alp2=0, only one fft is used.
% If K=1, no fft's are used.
%
function S=Capon2D(x,N,M,K,Nsig,Nw,k1,k2,delsig,alp1,alp2,bet1,bet2,C,g)

delw=pi/Nw;
x1=x(1,:);
xbb=x1(ones(M,1)*[1:N]+[0:M-1]'*ones(1,N));
if C>1
   for i=2:C
      xi=x(i,:);
      xbb=[xbb
         xi(ones(M,1)*[1:N]+[0:M-1]'*ones(1,N))];
   end
end
if bet1==0
   bet=bet2;
   Rinv=fRinv(xbb,N,M,bet,C);
end
if K==1
   Xbb=repmat(x(1,1:M).',1,Nw);
   if C>1
      for i=2:C
```

```
         Xbb=[Xbb
            repmat(x(i,1:M).',1,Nw)];
      end
   end
elseif (K~=1)&(alp1==-0.5&alp2==0)
   Xbb=fXbb(xbb,N,M,Nw,0,C);
end
for m=0:Nsig-1
   sig=m*delsig;
   if bet1~=0
      bet=bet1*sig+bet2;
      Rinv=fRinv(xbb,N,M,bet,C);
   end
   if K==1
      L=1;
   elseif (K~=1)&(alp1==-0.5&alp2==0)
      L=fL(N,0.5*sig);
   else
      alp=alp1*sig+alp2;
      Xbb=fXbb(xbb,N,M,Nw,sig+2*alp,C);
      L=fL(N,sig+alp);
   end
   sbb0=exp(-sig*[0:M-1]'*ones(1,k2-k1+1)+j*delw*[0:M-1]'*[k1:k2]);
   sbb=g(1)*sbb0;
   if C>1
      for i=2:C
         sbb=[sbb
            g(i)*sbb0];
      end
   end
   S(m+1,:)=(ones(1,M*C)*(conj(sbb).*(Rinv*Xbb(:,k1+1:k2+1)))./...
      (L*ones(1,M*C)*(conj(sbb).*(Rinv*sbb)));
end
return

function Rinv=fRinv(xbb,N,M,bet,C)
e=repmat(exp(-bet*[0:N-1]),M*C,1);
xbbe=xbb.*e;
R=xbbe*xbbe';
Rinv=inv(R);
return

function Xbb=fXbb(xbb,N,M,Nw,sig,C)
e=repmat(exp(-sig*[0:N-1]),M*C,1);
xbbe=xbb.*e;
Xbb=fft(xbbe.',2*Nw).';
return

function L=fL(N,sig)
if sig==0
   L=N;
else
   L=(1-exp(-2*sig*N))/(1-exp(-2*sig));
end
return
```

A.5.3 fgest.m

```
% fgest.m - gain estimation for multiple-channel signals for FID signal
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
%  Fred J. Frigo, James A. Heinen
```

```
%
% xlong is the matrix containing signals from C channels
%
function gest=fgest(xlong,C)
one=ones(C,1);
R=xlong*xlong';
Rg=R-trace(R)*eye(C);
RgRginv=inv(Rg'*Rg);
gest=RgRginv*one/(one'*RgRginv*one);
return
```

A.5.4 frhsqest.m

```
% frhosqest.m  - multiple-channel rho-sqaured (noise variance) estimation for FID signal
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
%  Fred J. Frigo, James A. Heinen
%
% xlong is the matrix containing signals from C channels
%
% Ns = length of signal (high SRN region of FID signal)
% Nf = total length of FID signal.
%
% note:  rho^2 is estimated from "noisy" end of FID signal
%

function rhosqest=frhosqest(xlong,Ns,Nf,C)
xrho=xlong(:,Ns+1:Nf);
for i=1:C
   rhosqest(i)=xrho(i,:)*xrho(i,:)';
end
rhosqest=rhosqest.'/(Nf-Ns);
return
```

A.5.5 getx.m

```
% getx.m - get simulated test signal (5 different damped sinusoid components)
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
%  Fred J. Frigo, James A. Heinen
%
% note: if C > 1, a set of test signals is created, each with a different
%       SNR and gain as specified by gideal, dbSNRideal
%
function [xlong,xclean,rhosqideal]=getx(Nf,delw,C,gideal,dbSNRideal)
n=[0:Nf-1];
t=n;
xclean=2*exp(j*pi/3)*exp((-0.003+j*20*delw)*t);
xclean=xclean+4*exp(-j*pi/6)*exp((-0.008+j*25*delw)*t);
xclean=xclean+2*exp(j*pi)*exp((-0.001+j*50*delw)*t);
xclean=xclean+4*exp(j*pi/6)*exp((-0.008+j*210*delw)*t);
xclean=xclean+2*exp(-j*pi/3)*exp((-0.003+j*220*delw)*t);
xeng=xclean*xclean';
for i=1:C
   noise=(randn(size(xclean)))+j*(randn(size(xclean)));
   neng=noise*noise';
   noisegain(i)=sqrt((abs(gideal(i))^2*xeng)/(neng*10^(dbSNRideal(i)/10)));
```

```
   noisei=noisegain(i)*noise;
   rhosqideal(i)=noisei*noisei'/Nf;
   xi=gideal(i)*xclean+noisei;
   if i==1
      xlong=xi;
   else
      xlong=[xlong
         xi];
   end
end
rhosqideal=rhosqideal.';
return
```

A.5.6 getxmrs.m

```
% getxmrs.m - read phase-corrected, water-suppressed MRS signal
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
% Fred J. Frigo, James A. Heinen
%
% Note: If C > 1, a file for each receive channel will be obtained.
%
function [xlong,dbSNRideal,fname]=getxmrs(Nf,C);
for i=1:C
   [fname, pname] = uigetfile('*.*', 'Select spectroscopy raw data file');
   afile = strcat(pname, fname);
   fid = fopen(afile,'r', 'ieee-le');
   [raw_data, count] = fread(fid, inf, 'real*4');
   fclose(fid);
   for m = 1:(count/2)
        frame_data(m) = raw_data((2*m)-1) + j*raw_data(2*m);
   end
   xi=frame_data(1:Nf)*0.0001;
   xeng(i)=xi*xi';
   if i==1
      xlong=xi;
   else
      xlong=[xlong
         xi];
   end
end
dbSNRideal=NaN;
return
```

A.5.7 peak.m

```
% peak.m - peak-enhancement, plot peaks as Dirac delta functions
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
%  Fred J. Frigo, James A. Heinen
%
% S - Input is 2D (real or complex) surface.
% thresh - threshold; peaks must have magnitude greater than this
% suppreslastrow - if set to 1, ignore peaks from last row (recommended)
% Sp - Output representing 2D peak-enhanced surface
%
function Sp=peak(S,thresh,suppresslastrow)
S=abs(S);
[Nsig,Nw]=size(S);
```

```
Sb=zeros(Nsig+2,Nw+2);
Sbp=zeros(Nsig+2,Nw+2);
for m=1:Nsig
   for k=1:Nw
      Sb(m+1,k+1)=S(m,k);
   end
end
for m=2:Nsig+1
   for k=2:Nw+1
      if Sb(m,k)>=max([Sb(m+1,k-1) Sb(m+1,k) Sb(m+1,k+1) Sb(m,k-1) ...
            Sb(m,k+1) Sb(m-1,k-1) Sb(m-1,k) Sb(m-1,k+1)])&Sb(m,k)>=thresh
       Sbp(m,k)=Sb(m,k);
    end
   end
end
for m=1:Nsig
   for k=1:Nw
      Sp(m,k)=Sbp(m+1,k+1);
   end
end
if suppresslastrow==1
   for k=1:Nw
      Sp(Nsig,k)=0;
   end
end
return
```

A.5.8 peakproj.m

```
% peakproj.m - Create peak projection plot of a 2D surface
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
%  Fred J. Frigo, James A. Heinen
%
function Spp=peakproj(S,thresh)
S=abs(S);
[Nsig,Nw]=size(S);
for k=1:Nw
   Spp(k)=max(S(:,k));
   if Spp(k)<=thresh
      Spp(k)=0;
   end
end
return
```

A.5.9 spectrum2D.m

```
% spectrum2D.m - mainline function to evaluate 2D Capon / 2D APES
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
% Fred J. Frigo, James A. Heinen
%
% This is the main MATLAB code used to analyze:
%   weighted 2D Capon
%   weighted 2D APES
%   combined weighted 2D APES / 2D APES
%   2D multiple-channel spectral estimation using signal averaging
%   2D multiple-channel spectral estimation using spectrum averaging
%
```

```
% Use datatype=1 to use simulated test signal or
%     datatype=2 to use phase-corrected, water-suppressed MRS data
%
% Note:
%   Standard 2D Capon:   alp1=alp2=bet1=bet2=gam=0;   spectrumtype=1
%   Standard 2D APES: alp1=alp2=bet1=bet2=0; gam=1;   spectrumtype=2
%
clear all
close all
N=1792;  %512   % Note: for MRS data N+M should = 2048 for optimum results!
M=256;   %128
Ns=1800;   %1800
Nf=2048;%0<Ns<Nf.  Nf>=N+M-1.
K=N;%K must be 1 or N.  alp1 and alp2 are irrelevant if K=1.
Nsig=40;
Nw=N/2;%Nw must be greater than or equal to N/2.
k1=0;
k2=Nw-1;%0<=k1<k2<=Nw-1.
delsig=0.0005;
alp1=0;
alp2=0;
bet1=0;  % 0 = default - set this to 0.5 to increase peak picking sensitivity
bet2=0;
gam=0;%0<gam<=1.  gam=1 reduces to 2D APES.  gam=0 would reduce to 2D Capon.
C=1;

datatype=2;%=1 for simulated data, 2 for mrs data.
spectrumtype=1;%=1 for 2D Capon, 2 for 2D APES/Capon.

if C>1
   combotype=1;%=1 for signal averaging, 2 for spectrum averaging, 3 for composite
spectrum.
   gtype=2;%=1 for ideal g's, 2 for estimated g's.
           %ideal can be used only for simulated data.
   rhosqtype=2;%=1 for ideal rhosq's, 2 for estimated rhosq's.
               %ideal can be used only for simulated data.
end

delw=pi/Nw;
mrs_name='x(t)';
if datatype==1
   gideal=[1];%[0.5*j+0.4 0.2 0.15+j*3 0.15-j*3].';%length(gideal)=C.  If C=1, then
gideal=1.
   dbSNRideal=[10];%[40 40 40 40].';%length(dbSNRideal)=C.
   [xlong,xclean,rhosqideal]=getx(Nf,delw,C,gideal,dbSNRideal);
elseif datatype==2
   [xlong,dbSNRideal,mrs_name]=getxmrs(Nf,C);
   tempC=37  %37
   %[xlong,dbSNRideal,mrs_name]=getxmrs_sphere(Nf,C);
   k1=floor(0.02*Nw/pi);
   k2=floor(0.8*Nw/pi);
end

x=xlong(:,1:N+M-1);

if C>1

   if gtype==1&datatype==1
      g=gideal;
   elseif gtype==2
      g=fgest(xlong,C);
      %g=fgest(x,C);
   end

   if rhosqtype==1&datatype==1
      rhosq=rhosqideal;
```

```
   elseif rhosqtype==2
      rhosq=frhosqest(xlong,Ns,Nf,C);
   end

   Rw=diag(rhosq);
   Rwinv=inv(Rw);
   w=Rwinv*g/(g'*Rwinv*g);
   xest=w'*x;

else % if C == 1, then g = 1
    g=1;
end

tic

if (C==1)|(C>1&combotype==3)
   if spectrumtype==1
      S=Capon2D(x,N,M,K,Nsig,Nw,k1,k2,delsig,alp1,alp2,bet1,bet2,C,g);
   elseif spectrumtype==2
      S=APESCapon2D(x,N,M,Nsig,Nw,k1,k2,delsig,bet1,bet2,gam,C,g);
   end
end

if C>1&combotype==1
   if spectrumtype==1
      S=Capon2D(xest,N,M,K,Nsig,Nw,k1,k2,delsig,alp1,alp2,bet1,bet2,1,1);
   elseif spectrumtype==2
      S=APESCapon2D(xest,N,M,Nsig,Nw,k1,k2,delsig,bet1,bet2,gam,1,1);
   end
end

if C>1&combotype==2
   S=zeros(Nsig,k2-k1+1);
   for i=1:C
      if spectrumtype==1
         Si=Capon2D(x(i,:),N,M,K,Nsig,Nw,k1,k2,delsig,alp1,alp2,bet1,bet2,1,1);
      elseif spectrumtype==2
         Si=APESCapon2D(x(i,:),N,M,Nsig,Nw,k1,k2,delsig,bet1,bet2,gam,1,1);
      end
      S=S+conj(w(i))*Si;
   end
end

toc

if spectrumtype==1
   paras=['N = ',num2str(N),', M = ',num2str(M),', K = ',num2str(K),...
          ', alp1 = ',num2str(alp1),', alp2 = ',num2str(alp2),...
          ', bet1 = ',num2str(bet1),', bet2 = ',num2str(bet2),...
          ', SNR = ',num2str(dbSNRideal'),' db'];
elseif spectrumtype==2
   paras=['N = ',num2str(N),', M = ',num2str(M),...
          ', bet1 = ',num2str(bet1),', bet2 = ',num2str(bet2),...
          ', gam = ',num2str(gam),', SNR = ',num2str(dbSNRideal'),' db'];
end

sigset=delsig*[0:Nsig-1];
wset=delw*[k1:k2];
n=[0:N+M-2];

Sp=peak(S,0,1);
Spp=peakproj(Sp,0);

% create projection of the non-peak enhanced spectrum (fig38 phD)
rawSpp=peakproj(abs(S), 0);
```

```
% Plots
if( spectrumtype == 1)
  analysis_string = '2D Capon ';
else
  analysis_string = '2D Capon/APES ';
end

figure
title_string = strcat([analysis_string, 'surface plot of |S(\sigma,\omega)| for ',
mrs_name]);
surf(wset,sigset,abs(S)),title(title_string), xlabel('\omega'),
ylabel('\sigma'),zlabel('|S(\sigma,\omega)|');

figure
title_string = strcat([analysis_string, 'contour plot of |S(\sigma,\omega)| for ',
mrs_name]);
contour(wset,sigset,abs(S)),title(title_string),xlabel('\omega'),ylabel('\sigma');

figure
title_string = strcat([analysis_string, 'peak-enhanced surface plot of |S(\sigma,\omega)|
for ', mrs_name]);
surf(wset,sigset,Sp),title(title_string),xlabel('\omega'),ylabel('\sigma'),zlabel('|S(\si
gma,\omega)|');

figure
title_string = strcat([analysis_string, 'peak-enhanced contour plot of |S(\sigma,\omega)|
for ', mrs_name]);
contour(wset,sigset,Sp),title(title_string),xlabel('\omega'),ylabel('\sigma');
% using sign(Sp) causes all peaks to be shown, no matter how small
% contour(wset,sigset,sign(Sp)),title(title_string),xlabel('\omega'),ylabel('\sigma');

figure
title_string = strcat([analysis_string, 'projected peaks of |S(\sigma,\omega)| for ',
mrs_name]);
plot(wset,Spp),title(title_string),xlabel('\omega'),ylabel('|S(\sigma,\omega)|');

figure
title_string = strcat([analysis_string, 'projection of |S(\sigma,\omega)| for ',
mrs_name]);
plot(wset,rawSpp),title(title_string),xlabel('\omega'),ylabel('|S(\sigma,\omega)|');

X=fft(x.').';
[Cft,Nft]=size(abs(X));
k1ft=floor(Nft*k1/(2*Nw));
k2ft=floor(Nft*k2/(2*Nw));
wfset=[k1ft:k2ft]*2*pi/Nft;
figure
plot(wfset,abs(X(:,k1ft+1:k2ft+1))/Nft),title(strcat(['Fourier transform of
',mrs_name])),xlabel('\omega'),ylabel('|S(\omega)|');
%plot(wfset,abs(X(:,k1ft+1:k2ft+1))/Nft),title('abs(FFT)s of observed
signal(s)'),xlabel(paras)
if C>1
   Xest=fft(xest.').';
   figure
   plot(wfset,abs(Xest(:,k1ft+1:k2ft+1))/Nft),title('average abs(FFT)'),xlabel(paras)
end
figure
plot(n,abs(x).'),title(strcat(['Input signal ',mrs_name])), xlabel('t'),ylabel('x(t)');
%plot(n,abs(x).'),title('observed signal(s)'),xlabel(paras)

% Plots identical to above, but scaled to ppm axis.
% PPM axis  ( calibrated for 37C )
% tempC must be set for proper ppm axis scaling. (tempC=37) for in vivo
ppm_start_37C = -4.55;
ppm_stop_37C = 0.30;
ppm_per_degree_C=0.01;
```

```
ppm_offset = (tempC-37)*ppm_per_degree_C;
ppm_start = ppm_start_37C + ppm_offset;
ppm_stop = ppm_stop_37C + ppm_offset;
ppm_x = linspace(ppm_start,ppm_stop,(k2-k1+1));


figure
title_string = strcat([analysis_string, 'surface plot of |S(\sigma,\omega)| for ',
mrs_name]);
surf(ppm_x,sigset,abs(S)), xlabel('ppm'), ylabel('\sigma'),zlabel('|S(\sigma,\omega)|');
set(gca,'XTick',-4.0:1.0:0.0);
set(gca,'XTickLabel',{'4.0','3.0','2.0','1.0','0.0'});

figure
title_string = strcat([analysis_string, 'contour plot of |S(\sigma,\omega)| for ',
mrs_name]);
contour(ppm_x,sigset,abs(S)),xlabel('ppm'),ylabel('\sigma');
set(gca,'XTick',-4.0:1.0:0.0);
set(gca,'XTickLabel',{'4.0','3.0','2.0','1.0','0.0'});

figure
title_string = strcat([analysis_string, 'peak enhanced surface plot of |S(\sigma,\omega)|
for ', mrs_name]);
surf(ppm_x,sigset,Sp),xlabel('ppm'),ylabel('\sigma'),zlabel('|S(\sigma,\omega)|');
set(gca,'XTick',-4.0:1.0:0.0);
set(gca,'XTickLabel',{'4.0','3.0','2.0','1.0','0.0'});

figure
title_string = strcat([analysis_string, 'peak enhanced contour plot of |S(\sigma,\omega)|
for ', mrs_name]);
contour(ppm_x,sigset,Sp),xlabel('ppm'),ylabel('\sigma');
set(gca,'XTick',-4.0:1.0:0.0);
set(gca,'XTickLabel',{'4.0','3.0','2.0','1.0','0.0'});

figure
title_string = strcat([analysis_string, 'projected peaks of |S(\sigma,\omega)| for ',
mrs_name]);
plot(ppm_x,Spp),xlabel('ppm'),ylabel('|S(\sigma,\omega)|');
set(gca,'XTick',-4.0:1.0:0.0);
set(gca,'XTickLabel',{'4.0','3.0','2.0','1.0','0.0'});
```

A.6 MATLAB Code for Simulations

A.6.1 check_for_nan.m

```
% check_for_nan.m - Check for NaN and set all prior points to NaN if found.
% Marquette University,   Milwaukee, WI  USA
% Copyright 2004 - All rights reserved.
% Fred J. Frigo
% Feb 10, 2004  - original
%
%

function out_data=check_for_nan(in_data)

array_size = max(size(in_data));

% Check for NaN and set all prior points to NaN if found.
for m=array_size:-1:1
    if( isnan(in_data(m)) == 1 )
        for k=m:-1:1
            out_data(k)=NaN;
        end
        break;
    else
        out_data(m)=in_data(m);
    end
end
return;
```

A.6.2 count_peaks.m

```
% count_peaks.m - Count the number of peaks that exceed a threshold
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003 - All rights reserved.
% Fred J. Frigo
% Dec 23, 2003
%
%

function peaks_found=count_peaks(S,thresh)
peaks_found = 0;
S=abs(S);
[Nsig,Nw]=size(S);
for k=1:Nw
   Spp(k)=max(S(:,k));
   if Spp(k)>thresh
      peaks_found = peaks_found + 1;
   end
end
return
```

A.6.3 create_noise.m

```
% create_noise.m  -  create 10 noise instances and save to files
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
% Fred Frigo
% Dec 10, 2003
```

```
% Feb 20, 2004  - added extra noise files for multi-channel testing

clear all
close all

% size of desired noise frame
N=1024;

% number of noise frames to create
num_noise_frames = 40;
filename=[
    'noise_0.dat';
    'noise_1.dat';
    'noise_2.dat';
    'noise_3.dat';
    'noise_4.dat';
    'noise_5.dat';
    'noise_6.dat';
    'noise_7.dat';
    'noise_8.dat';
    'noise_9.dat';
    'noise10.dat';
    'noise11.dat';
    'noise12.dat';
    'noise13.dat';
    'noise14.dat';
    'noise15.dat';
    'noise16.dat';
    'noise17.dat';
    'noise18.dat';
    'noise19.dat';
    'noise20.dat';
    'noise21.dat';
    'noise22.dat';
    'noise23.dat';
    'noise24.dat';
    'noise25.dat';
    'noise26.dat';
    'noise27.dat';
    'noise28.dat';
    'noise29.dat';
    'noise30.dat';
    'noise31.dat';
    'noise32.dat';
    'noise33.dat';
    'noise34.dat';
    'noise35.dat';
    'noise36.dat';
    'noise37.dat';
    'noise38.dat';
    'noise39.dat'];

% debug flag for plotting
do_plot=0;

% Loop to create complex noise frames and plot them
for index=1:num_noise_frames

   % Create complex noise
   noise=(randn(N) + j*(randn(N)))./sqrt(2.0);

   if (index ==1)
       figure;
       subplot(2,1,1);
       plot(1:N, abs(noise), 'k' );
       if (index == num_noise_frames)
```

```
            xlabel('Magnitude of noise');
        end

        subplot(2,1,2);
        plot(1:N, angle(noise), 'k' );
        if (index == num_noise_frames)
            xlabel('Phase of noise');
        end
    end

    % Save to file
    fidsig = fopen(filename(index,:), 'w+b');
    for findex=1:N
       fwrite(fidsig, real(noise(findex)), 'real*4');
       fwrite(fidsig, imag(noise(findex)), 'real*4');
    end
    fclose(fidsig);

end
```

A.6.4 create_signals.m

```
% create_signals.m  -  create 3 simulation signals and save to files
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
% Fred Frigo
% Dec 10, 2003
% Mar 14, 2004 - plot enhancements

clear all
close all

% size of desired signals
Nf=1024;
N=512;
Nw=N/2;
delw=pi/Nw;
t=[0:Nf-1];

% signal I
signal_1=1.0*exp(j*pi/4.0)*exp((-0.002+j*(63.0)*delw)*t);
signal_1=signal_1+(0.75)*exp(j*(-pi/2.0))*exp((-0.006+j*(127.0)*delw)*t);
signal_1=signal_1+(0.25)*exp(j*0.0)*exp((-0.004+j*(191.0)*delw)*t);

%------------------------------
% plot of signal I
%------------------------------
plot_complex('Signal I', signal_1);

% Save to file
fidsig = fopen('signal_1.dat', 'w+b');
for findex=1:Nf
   fwrite(fidsig, real(signal_1(findex)), 'real*4');
   fwrite(fidsig, imag(signal_1(findex)), 'real*4');
end
fclose(fidsig);

%------------------------------
% signal II
%------------------------------
signal_2=1.0*exp(j*pi/4.0)*exp((-0.001+j*(30.0)*delw)*t);
signal_2=signal_2+(25.0)*exp(j*0.0)*exp((-0.002+j*(40.0)*delw)*t);
signal_2=signal_2+(0.75)*exp(j*(-pi/4.0))*exp((-0.004+j*(50.0)*delw)*t);
```

```
signal_2=signal_2+(0.25)*exp(j*0.0)*exp((-0.003+j*(100.0)*delw)*t);
signal_2=signal_2+(1.0)*exp(j*0.0)*exp((-0.001+j*(120.0)*delw)*t);
signal_2=signal_2+(10.0)*exp(j*(pi/2.0))*exp((-0.002+j*(125.0)*delw)*t);
signal_2=signal_2+(0.5)*exp(j*(-pi/2.0))*exp((-0.002+j*(195.0)*delw)*t);
signal_2=signal_2+(1.0)*exp(j*0.0)*exp((-0.003+j*(225.0)*delw)*t);

% plot of signal II
plot_complex('Signal II', signal_2);

% Save to file
fidsig = fopen('signal_2.dat', 'w+b');
for findex=1:Nf
   fwrite(fidsig, real(signal_2(findex)), 'real*4');
   fwrite(fidsig, imag(signal_2(findex)), 'real*4');
end
fclose(fidsig);

%------------------------------
% signal III
%------------------------------
signal_3=0.5*exp(j*0.0)*exp((-0.004+j*(0.0)*delw)*t);
signal_3=signal_3+(0.75)*exp(j*(-pi/2.0))*exp((-0.002+j*(10.0)*delw)*t);
signal_3=signal_3+(1.0)*exp(j*(pi/4.0))*exp((-0.003+j*(31.0)*delw)*t);
signal_3=signal_3+(0.75)*exp(j*(0.0))*exp((-0.003+j*(34.0)*delw)*t);
signal_3=signal_3+(0.5)*exp(j*(-pi/2.0))*exp((-0.005+j*(55.0)*delw)*t);
signal_3=signal_3+(1.0)*exp(j*(pi/2.0))*exp((-0.001+j*(63.0)*delw)*t);
signal_3=signal_3+(1.0)*exp(j*(-3.0*pi/4.0))*exp((-0.005+j*(66.0)*delw)*t);
signal_3=signal_3+(0.25)*exp(j*(0.0))*exp((-0.004+j*(73.0)*delw)*t);
signal_3=signal_3+(0.5)*exp(j*(pi/2.0))*exp((-0.000+j*(95.0)*delw)*t);
signal_3=signal_3+(0.5)*exp(j*(-pi/2.0))*exp((-0.006+j*(104.0)*delw)*t);

% plot of signal III
plot_complex('Signal III', signal_3);

% Save to file
fidsig = fopen('signal_3.dat', 'w+b');
for findex=1:Nf
   fwrite(fidsig, real(signal_3(findex)), 'real*4');
   fwrite(fidsig, imag(signal_3(findex)), 'real*4');
end
fclose(fidsig);
```

A.6.5 find_peak.m

```
% find_peak.m - Find peak and compute squared error for amplitude, phase
%               and damping
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
% Fred J. Frigo
% Dec 23, 2003 - original
% Jan 20, 2004 - Added bias error computation
%
%
%    Inputs:
%           S = input data signal
%           component_info = contains info about each peak in the signal
%                       [ A, theta, sigma, omega_index, next_omega_flag]
%                       A = expected magnitude
%                       theta = expected phase
%                       sigma = expected sigma (damping)
%                       omega_index = omega index for expected peak
%                       next_omega_flag = 1 if OK to use next omega index
```

```
%           delw = step size for omega (freq)
%           delsig = step size for sigma (damping)
%           threshold = magnitude required to be classified as a peak
%
%    Outputs:
%           rms_mag_err = rms error for magnitude
%           rms_phase_err = rms error for phase
%           rms_sigma_err = rms error for sigma (damping)
%           pct_mag_err = percent error for magnitude
%           pct_phase_err = percent error for phase
%           pct_sigma_err = percent error for sigma (damping)
%           bias_mag_err = bias error for magnitude
%           bias_phase_err = bias error for phase
%           bias_sigma_err = bias error for sigma (damping)


function [found_peak_flag, rms_mag_err, rms_phase_err, rms_sigma_err, ...
                            pct_mag_err, pct_phase_err, pct_sigma_err, ...
                            bias_mag_err, bias_phase_err, bias_sigma_err] = find_peak(
S, component_info, delw, delsig, threshold )

rms_mag_err = 0.0;
rms_phase_err = 0.0;
rms_sigma_err = 0.0;
pct_mag_err = 0.0;
pct_phase_err = 0.0;
pct_sigma_err = 0.0;
bias_mag_err = 0.0;
bias_phase_err = 0.0;
bias_sigma_err = 0.0;


expected_magnitude = component_info(1);
expected_theta = component_info(2);
expected_sigma = component_info(3);
omega_start = component_info(4)+1;
next_omega_flag = component_info(5);

% Obtain num of sigma values
num_sigma = min(size( S(:,:)));  % Nsig

% Check to see if we need to include two values of omega for search (boundary condition)
if (next_omega_flag == 1)
    omega_end = omega_start + 1;
else
    omega_end = omega_start;
end

found_peak_flag = 0;
found_mag = 0.0;
found_phase = 0.0;
found_sigma = 0.0;

for omega = omega_start:omega_end
    [max_amp sigma_index] = max( abs(S(:,omega)) );
    if (max_amp > threshold)
       found_mag = abs(S(sigma_index, omega));
       S(sigma_index, omega);
       found_phase = angle(S(sigma_index, omega));
       found_sigma = delsig*(sigma_index-1);
       threshold = max_amp;
       found_peak_flag = 1;
    end
end

% Compute error terms if peak was found
```

```
if (found_peak_flag==1 )
    mag_diff = found_mag - expected_magnitude;
    rms_mag_err = (mag_diff*mag_diff)/(expected_magnitude*expected_magnitude);
    pct_mag_err = abs(mag_diff)/abs(expected_magnitude);
    bias_mag_err = mag_diff/expected_magnitude;

    phase_diff = found_phase - expected_theta;
    % -pi < phase_diff < pi
    if (phase_diff > pi )
        phase_diff = phase_diff - (2.0*pi);
    elseif ( -pi > phase_diff )
        phase_diff = phase_diff + (2.0*pi);
    end
    rms_phase_err = (phase_diff*phase_diff)/((2.0*pi)*(2.0*pi));
    pct_phase_err = abs(phase_diff)/(2.0*pi);
    bias_phase_err = phase_diff/(2.0*pi);

    sigma_diff = found_sigma - expected_sigma;
    max_sigma = (num_sigma-1)*delsig;
    rms_sigma_err = (sigma_diff*sigma_diff)/(max_sigma*max_sigma);
    pct_sigma_err = abs(sigma_diff)/abs(max_sigma);
    bias_sigma_err = sigma_diff/max_sigma;
end

return
```

A.6.6 mplot_1.m

```
% mplot_1.m - Mainline script to generate multichannel plot #1
% Marquette University,   Milwaukee, WI  USA
% Copyright 2004 - All rights reserved.
% Fred J. Frigo
% Feb 29, 2004  - original
%
% Plots are obtained from msim_1_results.txt created by
%  the msim_1.m MATLAB file.
%
% Standard Capon with number of channels, C = 1
% Standard Capon with signal averaging, C = 4
% Standard Capon with spectrum averaging, C=4
% 3 curves

clear all
close all

% File to read
results_file = 'msim_1_results.txt';


% SNR values
num_snr_values = 12;
snr_value=[ -18.0, -12.0, -6.0, 0.0, 6.0, 12.0, 18.0, 24.0, 30.0, 36.0, 42.0, 48.0];

% Open results file
fid=fopen(results_file, 'rt');

% First three lines contain labels to discard
line1=fgets(fid);
line2=fgets(fid);
line3=fgets(fid);

% Loop through report
num_sigs =1;  % only one signal for this report
```

```
num_C_values=3;  % 3 different tests
for sig_id=1:num_sigs
   for cloop = 1: num_C_values % num_C_values
       % SNR loop
       for snrloop = 1: num_snr_values  % num_snr_values
           % read the block of data
           for field = 1: 12

               text_line=fgets(fid);
               % The 'mean' field occurs in the sub-string starting at 60
               %      this depends on the results file being used
               result_mean = sscanf(text_line(60:71),'%f');
               results(sig_id, cloop, snrloop, field)=result_mean;

            end  % field loop
        end % SNR loop
    end % C loop
end % signal loop
fclose(fid);

% Average results for all signals
for cloop = 1: num_C_values  % num_m_values

       % SNR loop
       for snrloop = 1: num_snr_values  % num_snr_values

              avg_missed_peaks( cloop, snrloop)= mean(
results(1:num_sigs,cloop,snrloop,1));
              avg_false_peaks( cloop, snrloop)= mean(
results(1:num_sigs,cloop,snrloop,2));
              avg_mag_rms_error( cloop, snrloop)= mean(
results(1:num_sigs,cloop,snrloop,3));
              avg_sigma_rms_error( cloop, snrloop)= mean(
results(1:num_sigs,cloop,snrloop,5));

       end
       % Check for NaN and set all prior points to NaN if found.
       avg_mag_rms_error( cloop,:) = check_for_nan( avg_mag_rms_error( cloop,:));
       avg_sigma_rms_error( cloop,:) = check_for_nan( avg_sigma_rms_error( cloop,:));

end %  C loop

% Missed peaks plot
figure;
plot( snr_value, squeeze(avg_missed_peaks(1,:)), 'ko-', ...
      snr_value, squeeze(avg_missed_peaks(2,:)), 'bx-', ...
      snr_value, squeeze(avg_missed_peaks(3,:)), 'g+-');
legend('C=1','C=4; signal averaging','C=4; spectrum averaging');
xlabel('SNR (dB)');
ylabel('% Missed Peaks');
title('Capon, K=N');

% False peaks
figure;
plot( snr_value, squeeze(avg_false_peaks(1,:)), 'ko-', ...
      snr_value, squeeze(avg_false_peaks(2,:)), 'bx-', ...
      snr_value, squeeze(avg_false_peaks(3,:)), 'g+-');
legend('C=1','C=4; signal averaging','C=4; spectrum averaging');
xlabel('SNR (dB)');
ylabel('% False Peaks');
title('Capon, K=N');

% Magnitude RMS errors
figure;
plot( snr_value, squeeze(avg_mag_rms_error(1,:)), 'ko-', ...
      snr_value, squeeze(avg_mag_rms_error(2,:)), 'bx-', ...
```

```
      snr_value, squeeze(avg_mag_rms_error(3,:)), 'g+-');
legend('C=1','C=4; signal averaging','C=4; spectrum averaging');
xlabel('SNR (dB)');
ylabel('Relative RMS Magnitude Error');
title('Capon, K=N');

% Sigma RMS errors
figure;
plot( snr_value, squeeze(avg_sigma_rms_error(1,:)), 'ko-', ...
      snr_value, squeeze(avg_sigma_rms_error(2,:)), 'bx-', ...
      snr_value, squeeze(avg_sigma_rms_error(3,:)), 'g+-');
legend('C=1','C=4; signal averaging','C=4; spectrum averaging');
xlabel('SNR (dB)');
ylabel('Relative RMS Damping Error');
title('Capon, K=N');

% ------------------------------------------------------
% Combined plot
% ------------------------------------------------------

figure;

% Missed peaks plot
subplot(2,2,1);
plot( snr_value, squeeze(avg_missed_peaks(1,:)), 'ko-', ...
      snr_value, squeeze(avg_missed_peaks(2,:)), 'bx-', ...
      snr_value, squeeze(avg_missed_peaks(3,:)), 'g+-');
legend('C=1','C=4; signal averaging','C=4; spectrum averaging');
xlabel('SNR (dB)');
ylabel('% Missed Peaks');

% False peaks
subplot(2,2,2);
plot( snr_value, squeeze(avg_false_peaks(1,:)), 'ko-', ...
      snr_value, squeeze(avg_false_peaks(2,:)), 'bx-', ...
      snr_value, squeeze(avg_false_peaks(3,:)), 'g+-');
legend('C=1','C=4; signal averaging','C=4; spectrum averaging');
xlabel('SNR (dB)');
ylabel('% False Peaks');

% Magnitude RMS errors
subplot(2,2,3);
plot( snr_value, squeeze(avg_mag_rms_error(1,:)), 'ko-', ...
      snr_value, squeeze(avg_mag_rms_error(2,:)), 'bx-', ...
      snr_value, squeeze(avg_mag_rms_error(3,:)), 'g+-');
legend('C=1','C=4; signal averaging','C=4; spectrum averaging');
xlabel('SNR (dB)');
ylabel('Relative RMS Magnitude Error');

% Sigma RMS errors
subplot(2,2,4);
plot( snr_value, squeeze(avg_sigma_rms_error(1,:)), 'ko-', ...
      snr_value, squeeze(avg_sigma_rms_error(2,:)), 'bx-', ...
      snr_value, squeeze(avg_sigma_rms_error(3,:)), 'g+-');
legend('C=1','C=4; signal averaging','C=4; spectrum averaging');
xlabel('SNR (dB)');
ylabel('Relative RMS Damping Error');
```

A.6.7 msim_1.m

```
% msim_1.m - Mainline script to run multiple channel simulation #1
% Marquette University,   Milwaukee, WI  USA
```

```
% Copyright 2004 - All rights reserved.
% Fred J. Frigo
% Feb 22, 2004  - original


clear all
close all

results_file = 'msim_1_results.txt';
simulation_id = 9;

% Noise files
num_noise_frames = 10;
noise_files=[
    'noise_0.dat';
    'noise_1.dat';
    'noise_2.dat';
    'noise_3.dat';
    'noise_4.dat';
    'noise_5.dat';
    'noise_6.dat';
    'noise_7.dat';
    'noise_8.dat';
    'noise_9.dat';
    'noise10.dat';
    'noise11.dat';
    'noise12.dat';
    'noise13.dat';
    'noise14.dat';
    'noise15.dat';
    'noise16.dat';
    'noise17.dat';
    'noise18.dat';
    'noise19.dat';
    'noise20.dat';
    'noise21.dat';
    'noise22.dat';
    'noise23.dat';
    'noise24.dat';
    'noise25.dat';
    'noise26.dat';
    'noise27.dat';
    'noise28.dat';
    'noise29.dat';
    'noise30.dat';
    'noise31.dat';
    'noise32.dat';
    'noise33.dat';
    'noise34.dat';
    'noise35.dat';
    'noise36.dat';
    'noise37.dat';
    'noise38.dat';
    'noise39.dat'];



% Signal files
num_signals = 3;
signal_file=[
    'signal_1.dat';
    'signal_2.dat';
    'signal_3.dat'];

% SNR per channel
num_snr_values = 12;
```

```
snr_values=[ -18.0 ,  -18.0,  -18.0,  -18.0 ;
            -12.0 ,  -12.0,  -12.0,  -12.0 ;
             -6.0,    -6.0,   -6.0,   -6.0 ;
              0.0,     0.0,    0.0,    0.0 ;
              6.0,     6.0,    6.0,    6.0 ;
             12.0,    12.0,   12.0,   12.0 ;
             18.0,    18.0,   18.0,   18.0 ;
             24.0,    24.0,   24.0,   24.0 ;
             30.0,    30.0,   30.0,   30.0 ;
             36.0,    36.0,   36.0,   36.0 ;
             42.0,    42.0,   42.0,   42.0 ;
             48.0,    48.0,   48.0,   48.0 ];

% Ideal gain per channel
gideal = [1.0  1.0  1.0  1.0].';


% Signal Components: (these match signal definition in create_signals.m)
%    [ A, theta, sigma, omega_index, next_omega_flag]
%    A = magnitude
%    theta = phase
%    sigma = sigma (damping)
%    omega_index = frequency index for peak
%    next_omega_flag = 1 if OK to use next omega index (for peaks the are
%                                                          on boundaries)
%
signal_components_1 = [
    1.0,   pi/4.0,    0.002, 63,   0;
    0.75,  -pi/2.0,   0.006, 127,  0;
    0.25,  0.0,       0.004, 191,  0];

signal_components_2 = [
    1.00,   pi/4.0,   0.001,  30,  0;
   25.00,   0.0,      0.002,  40,  0;
    0.75,  -pi/4.0,   0.004,  50,  0;
    0.25,   0.0,      0.003, 100,  0;
    1.00,   0.0,      0.001, 120,  0;
   10.00,   pi/2.0,   0.002, 125,  0;
    0.50,  -pi/2.0,   0.002, 195,  0;
    1.00,   0.0,      0.003, 225,  0];

signal_components_3 = [
    0.50,   0.0,      0.004, 0,    0;
    0.75,  -pi/2.0,   0.002, 10,   0;
    1.00,   pi/4.0,   0.003, 31,   0;
    0.75,   0.0,      0.003, 34,   0;
    0.50,  -pi/2.0,   0.005, 55,   0;
    1.00,   pi/2.0,   0.001, 63,   0;
    1.00,   0.75*pi,  0.005, 66,   0;
    0.25,   0.0,      0.004, 73,   0;
    0.50,   pi/2.0,   0.000, 95,   0;
    0.50,  -pi/2.0,   0.006, 104,  0];

num_C_values = 3;  % number of simulation loops
C_values = [1, 4, 4]; % number of channels for each loop
combotype_values = [ 1, 1, 2];  % 1= signal averaging, 2= spectrum averaging
gtype_values = [1, 1, 1];  % 1= ideal (known) gains per chan, =2 estimate from data

N=512;

% Frame size  (max is 1024)
frame_size = N + 256;

% Get time and date
ctime = clock;
```

```
% Open results file
fid=fopen(results_file, 'wt+');
fprintf(fid,'Simulation: %2d\n', simulation_id);
fprintf(fid,'started: %2.2d/%2.2d/%4.4d  %2.2d:%2.2d:%2.2d -
\n',...
                                                ctime(2), ctime(3), ctime(1),ctime(4),
ctime(5),round(ctime(6)));
fprintf(fid,'           Parameter  signal  SNR     C  combo  gtype   N     mean
max          min         std        variance \n');
fclose(fid);

% Signal loop
for sig_id = 3:3 % num_signals

   % Get data
   fidsig = fopen(signal_file(sig_id,:), 'r+b');
   [raw_data, count] = fread(fidsig, frame_size*2, 'real*4');
   fclose(fidsig);
   for m = 1:(count/2)
      sig_data(m) = raw_data((2*m)-1) + j*raw_data(2*m);
   end

   % Compute signal energy for mixing SNR
   sig_energy=sum(abs(sig_data.*sig_data));

   % Select structure with signal component info
   if (sig_id == 1)
      peak_info = signal_components_1;
   elseif (sig_id == 2)
      peak_info = signal_components_2;
   elseif (sig_id == 3)
      peak_info = signal_components_3;
   end

   % C loop
   for cloop = 1: num_C_values  % num_C_values
       C = C_values(cloop);
       combotype = combotype_values(cloop);
       gtype = gtype_values(cloop);

       % SNR loop
       for snrloop = 1: num_snr_values  % num_snr_values
            % Compute desired weight for noise to yield desired SNR ratio in dB
            dbSNRideal = snr_values(snrloop,:);

            % Noise loop
            for noise_id = 1: num_noise_frames   % num_noise_frames
               % signal energy
               xeng=sig_data*sig_data';

               for chan_id = 1:C  % channel loop

                   % Get noise for this channel
                   noise_file_index = noise_id + num_noise_frames*(chan_id-1);
                   fidn = fopen(noise_files(noise_file_index,:), 'r+b');
                   [raw_data, count] = fread(fidn, frame_size*2, 'real*4');
                   fclose(fidn);
                   for m = 1:(count/2)
                       noise(m) = raw_data((2*m)-1) + j*raw_data(2*m);
                       noise(m)=0.1*noise(m);
                   end

                   % compute noise energy
                   neng=noise*noise';

                   % combine signal and noise with desired SNR
```

```
noisegain(chan_id)=sqrt((abs(gideal(chan_id))^2*xeng)/(neng*10^(dbSNRideal(chan_id)/10)))
;
                    noise_scaled=noisegain(chan_id)*noise;
                    rhosqideal(chan_id)=noise_scaled*noise_scaled'/frame_size; %
Nf=frame_size
                    sig_and_noise=gideal(chan_id)*sig_data+noise_scaled;

                    % Create input signal matrix
                    if (chan_id == 1)
                       x=sig_and_noise;
                    else
                       x=[x
                           sig_and_noise];
                    end
                end % channel loop
                rhosqideal=rhosqideal.';

                % Call function to do processing on algorithm to test
                [missed_peaks(noise_id), false_peaks(noise_id), mag_err_rms(noise_id),
phase_err_rms(noise_id),...
                     sigma_err_rms(noise_id), mag_err_pct(noise_id),
phase_err_pct(noise_id), ...
                     sigma_err_pct(noise_id), mag_err_bias(noise_id),
phase_err_bias(noise_id), ...
                     sigma_err_bias(noise_id), elapsed_time(noise_id)] = ...
                     msim_proc(x, C, combotype, rhosqideal, gideal, gtype, peak_info);

            end  % noise loop

            % Log results for each parameter for a set of noise instances
            for result_loop=1:12

                if result_loop == 1
                    result_data = missed_peaks;
                    result_string = 'Missed_Peaks';
                elseif result_loop == 2
                    result_data = false_peaks;
                    result_string = 'False_Peaks';
                elseif result_loop == 3
                     result_data = mag_err_rms;
                     result_string = 'Mag_Err_RMS';
                elseif result_loop == 4
                     result_data = phase_err_rms;
                     result_string = 'Phase_Err_RMS';
                elseif result_loop == 5
                     result_data = sigma_err_rms;
                     result_string = 'Sigma_Err_RMS';
                elseif result_loop == 6
                     result_data = mag_err_pct;
                     result_string = 'Mag_Err_percent';
                elseif result_loop == 7
                     result_data = phase_err_pct;
                     result_string = 'Phase_Err_percent';
                elseif result_loop == 8
                     result_data = sigma_err_pct;
                     result_string = 'Sigma_Err_percent';
                elseif result_loop == 9
                     result_data = mag_err_bias;
                     result_string = 'Mag_Err_Bias';
                elseif result_loop == 10
                     result_data = phase_err_bias;
                     result_string = 'Phase_Err_Bias';
                elseif result_loop == 11
                     result_data = sigma_err_bias;
                     result_string = 'Sigma_Err_Bias';
```

```
                elseif result_loop == 12
                     result_data = elapsed_time;
                     result_string = 'Elapsed_time';
                end

                % Discard invalid results
                num_results=0;
                for (rindex = 1:num_noise_frames)
                    % If result is valid number, store it for computation
                    if ( isnan(result_data(rindex)) == 0 )
                        num_results = num_results+1;
                        valid_results(num_results) = result_data(rindex);
                    end
                end
                % Compute mean, std and variance on valid results.
                if ( num_results > 0 )
                    result_mean = mean(valid_results(1:num_results));
                    result_std = std(valid_results(1:num_results));
                    result_var = var(valid_results(1:num_results));
                    result_max = max(valid_results(1:num_results));
                    result_min = min(valid_results(1:num_results));
                else
                    result_mean = NaN;
                    result_std = NaN;
                    result_var = NaN;
                    result_max = NaN;
                    result_min = NaN;
                end

                % Log results to output file:
                % 'Parameter  signal  SNR    C   combo   N   mean     max    min     std
variance'
                fid=fopen(results_file, 'at');
                fprintf(fid,'%20.20s   %2.2d    %5.1f   %2.2d    %2.2d     %2.2d   %2.2d
%9.5f     %9.5f    %9.5f   %9.5f   %9.5f\n', ...
                        result_string, sig_id, dbSNRideal(1), C, combotype, gtype,
num_results, result_mean, result_max, ...
                        result_min, result_std, result_var );
                fclose(fid);

            end  % results loop
        end % SNR loop
    end %combo loop
end % signal loop

% Get time and date to add for "completed timestamp"
ctime = clock;
fid=fopen(results_file, 'rt+');
fseek(fid,47,0); % insert timestamp at beginning of file after "started"
fprintf(fid,' completed: %2.2d/%2.2d/%4.4d  %2.2d:%2.2d:%2.2d',...
                ctime(2), ctime(3), ctime(1),ctime(4), ctime(5),round(ctime(6)));
fclose(fid);
```

A.6.8 msim_proc.m

```
% msim_proc.m - Processing associated with multiple channel simulations
% Marquette University,   Milwaukee, WI  USA
% Copyright 2004 - All rights reserved.
% Fred J. Frigo
% Feb 22, 2004  - original
%
%  Inputs:
```

```
%           xlong = signal+noise
%           C = number of channels
%           combotype = 1 for signal averaging, 2 for spectrucm averaging
%           rhosqideal = rho ^2
%           gideal = ideal gains for each channel
%           gytpe = 1 ideal (known) gains per channel, = 2 estimate gains
%           peak_info = contains info about each peak in the signal
%                       [ A, theta, sigma, omega_index, next_omega_flag]
%                       A = expected magnitude
%                       theta = expected phase
%                       sigma = expected sigma (damping)
%                       omega_index = omega index for expected peak
%                       next_omega_flag = 1 if OK to use next omega index
%
%  Outputs:
%           missed_peaks = percentage of missed peaks
%           false_peaks = percentage of false peaks
%           rms_mag_err = RMS error for magnitude
%           rms_phase_err = RMS error for phase
%           rms_sigma_err = RMS error for sigma (damping)
%           pct_mag_err = RMS error for magnitude
%           pct_phase_err = RMS error for phase
%           pct_sigma_err = RMS error for sigma (damping)
%           elapsed_time = elaspsed time in seconds

function [missed_peaks, false_peaks, rms_mag_err, rms_phase_err, rms_sigma_err, ...
             pct_mag_err, pct_phase_err, pct_sigma_err, ...
             bias_mag_err, bias_phase_err, bias_sigma_err,elapsed_time] = ...
             msim_proc(xlong, C, combotype, rhosqideal, gideal, gtype, peak_info);

matched_peaks = 0;
total_peaks = 0;
missed_peaks = 0.0;
false_peaks = 0.0;
rms_mag_err = NaN;  % If no peaks are found, RMS error = NaN
rms_phase_err = NaN;
rms_sigma_err = NaN;
pct_mag_err = NaN; % If no peaks are found, Percent error = NaN
pct_phase_err = NaN;
pct_sigma_err = NaN;
bias_mag_err = NaN; % If no peaks are found, Bias error = NaN
bias_phase_err = NaN;
bias_sigma_err = NaN;

% Find out how many signal components there are
num_expected_peaks = max( size(peak_info(:,5)));

% set flag = 1 to display all plots
show_plots=0;

N=512;
M=128;
Ns=640;   %1800  ?Try N+M
Nf=768;%0<Ns<Nf.  Nf>=N+M-1. % Frame_size
K=N;%K must be 1 or N.  alp1 and alp2 are irrelevant if K=1.
Nsig=40;
Nw=N/2;%Nw must be greater than or equal to N/2.
k1=0;
k2=Nw-1;%0<=k1<k2<=Nw-1.
delsig=0.0002;
alp1=0;
alp2=0;
bet1=0;  % 0 = default - set this to 0.5 to increase peak picking sensitivity
bet2=0;
gam=0;%0<gam<=1.  gam=1 reduces to 2D APES.  gam=0 would reduce to 2D Capon.
%C=1;
```

```
spectrumtype=1;%=1 for 2D Capon, 2 for 2D APES/Capon.

if C>1
   % combotype=1;%=1 for signal averaging, 2 for spectrum averaging, 3 for composite
spectrum.
   % gtype=1;%=1 for ideal g's, 2 for estimated g's.
           %ideal can be used only for simulated data.
   rhosqtype=1;%=1 for ideal rhosq's, 2 for estimated rhosq's.
               %ideal can be used only for simulated data.
end

delw=pi/Nw;

%  signal plus noise
x=xlong(:,1:N+M-1);

if C>1

   if gtype==1
      g=gideal;
   elseif gtype==2
      g=fgest(xlong,C);
   end

   if rhosqtype==1
      rhosq=rhosqideal;
   elseif rhosqtype==2
      rhosq=frhosqest(xlong,Ns,Nf,C);
   end

   Rw=diag(rhosq);
   Rwinv=inv(Rw);
   w=Rwinv*g/(g'*Rwinv*g);

   xest=w'*x;

else % if C == 1, then g = a single value
    g=gideal(1);
end

tic;

if (C==1)|(C>1&combotype==3)
   if spectrumtype==1
      S=Capon2D(x,N,M,K,Nsig,Nw,k1,k2,delsig,alp1,alp2,bet1,bet2,C,g);
   elseif spectrumtype==2
      S=APESCapon2D(x,N,M,Nsig,Nw,k1,k2,delsig,bet1,bet2,gam,C,g);
   end
end

if C>1&combotype==1
   if spectrumtype==1
      S=Capon2D(xest,N,M,K,Nsig,Nw,k1,k2,delsig,alp1,alp2,bet1,bet2,1,1);
   elseif spectrumtype==2
      S=APESCapon2D(xest,N,M,Nsig,Nw,k1,k2,delsig,bet1,bet2,gam,1,1);
   end
end

if C>1&combotype==2
   S=zeros(Nsig,k2-k1+1);
   for i=1:C
      if spectrumtype==1
         Si=Capon2D(x(i,:),N,M,K,Nsig,Nw,k1,k2,delsig,alp1,alp2,bet1,bet2,1,1);
      elseif spectrumtype==2
         Si=APESCapon2D(x(i,:),N,M,Nsig,Nw,k1,k2,delsig,bet1,bet2,gam,1,1);
```

```
      end
      S=S+conj(w(i))*Si;
   end
end

elapsed_time = toc;

sigset=delsig*[0:Nsig-1];
wset=delw*[k1:k2];
n=[0:N+M-2];

% Create peak enhanced 2D surface where each peak is a dirac delta function
Sp=peak_complex(S,0,1);

% Compute peak projections
Spp=peakproj(Sp,0);

% Set threshold for valid peaks:  (Max component amplitude = 1.0)
%  - Use max projection (for determination of threshold)
% max_peak_value = max(abs(Spp));
% peak_threshold = max_peak_value*0.1;
peak_threshold = 0.025;

% Count number of peaks that exceed threshold
total_peaks = count_peaks(Spp, peak_threshold);

% Search for expected signal components, computing error terms if found
rms_mag_sum = 0.0;
rms_phase_sum = 0.0;
rms_sigma_sum = 0.0;
pct_mag_sum = 0.0;
pct_phase_sum = 0.0;
pct_sigma_sum = 0.0;
bias_mag_sum = 0.0;
bias_phase_sum = 0.0;
bias_sigma_sum = 0.0;
peaks_found = 0;
for peak_component = 1: num_expected_peaks
    component_info = squeeze(peak_info(peak_component, :));
    % Search for expected peak calculating magnitude, phase and damping errors
    [found_peak_flag, sq_mag_err, sq_phase_err, sq_sigma_err, abs_mag_err, abs_phase_err,
...
           abs_sigma_err, diff_mag_err, diff_phase_err, diff_sigma_err] = find_peak( Sp,
component_info, delw, delsig, peak_threshold );
    rms_mag_sum = rms_mag_sum + sq_mag_err;
    rms_phase_sum = rms_phase_sum + sq_phase_err;
    rms_sigma_sum = rms_sigma_sum + sq_sigma_err;
    pct_mag_sum = pct_mag_sum + abs_mag_err;
    pct_phase_sum = pct_phase_sum + abs_phase_err;
    pct_sigma_sum = pct_sigma_sum + abs_sigma_err;
    bias_mag_sum = bias_mag_sum + diff_mag_err;
    bias_phase_sum = bias_phase_sum + diff_phase_err;
    bias_sigma_sum = bias_sigma_sum + diff_sigma_err;


    if (found_peak_flag == 1)
        peaks_found = peaks_found + 1;
    end
end

% Compute error values if we found any peaks
if peaks_found > 0
    rms_mag_err = sqrt( (rms_mag_sum/peaks_found) );
    rms_phase_err = sqrt( (rms_phase_sum/peaks_found) );
    rms_sigma_err = sqrt( (rms_sigma_sum/peaks_found) );
    pct_mag_err = (pct_mag_sum/peaks_found)*100.0;
```

```
    pct_phase_err = (pct_phase_sum/peaks_found)*100.0;
    pct_sigma_err = (rms_sigma_sum/peaks_found)*100.0;
    bias_mag_err = bias_mag_sum/peaks_found;
    bias_phase_err = bias_phase_sum/peaks_found;
    bias_sigma_err = bias_sigma_sum/peaks_found;
end

% False positives ( as a percentage)
false_peaks = ((total_peaks - peaks_found)/Nw)*100.0;

% Missed peaks (as a percentage)
missed_peaks = ((num_expected_peaks - peaks_found)/num_expected_peaks)*100.0;

% Create plots if selected.
if (show_plots == 1)

   mrs_name='x(t)';

   % Plots
   if( spectrumtype == 1)
     analysis_string = '2D-Capon ';
   else
     analysis_string = '2D-APES ';
   end

   figure
   title_string = strcat([analysis_string, 'surface plot of |S(\sigma,\omega)| for ',
mrs_name]);
   surf(wset,sigset,abs(S)),title(title_string), xlabel('\omega'),
ylabel('\sigma'),zlabel('|S(\sigma,\omega)|');

   figure
   title_string = strcat([analysis_string, 'contour plot of |S(\sigma,\omega)| for ',
mrs_name]);
   contour(wset,sigset,abs(S)),title(title_string),xlabel('\omega'),ylabel('\sigma');

   figure
   title_string = strcat([analysis_string, 'peak enhanced surface plot of
|S(\sigma,\omega)| for ', mrs_name]);

surf(wset,sigset,abs(Sp)),title(title_string),xlabel('\omega'),ylabel('\sigma'),zlabel('|
S(\sigma,\omega)|');

   figure
   title_string = strcat([analysis_string, 'peak enhanced contour plot of
|S(\sigma,\omega)| for ', mrs_name]);
   contour(wset,sigset,abs(Sp)),title(title_string),xlabel('\omega'),ylabel('\sigma');

   figure
   title_string = strcat([analysis_string, 'projected peaks of |S(\sigma,\omega)| for ',
mrs_name]);
   plot(wset,Spp),title(title_string),xlabel('\omega'),ylabel('|S(\sigma,\omega)|');

   X=fft(x.').';
   [Cft,Nft]=size(abs(X));
   k1ft=floor(Nft*k1/(2*Nw));
   k2ft=floor(Nft*k2/(2*Nw));
   wfset=[k1ft:k2ft]*2*pi/Nft;
   figure
   plot(wfset,abs(X(:,k1ft+1:k2ft+1))/Nft),title(strcat(['Fourier Transform of
',mrs_name])),xlabel('\omega'),ylabel('|S(\omega)|');
   if C>1
      Xest=fft(xest.').';
      figure
      plot(wfset,abs(Xest(:,k1ft+1:k2ft+1))/Nft),title('average abs(FFT)'),xlabel(paras)
   end
```

```
   figure
   plot(n,abs(x).'),title(strcat(['Input signal ',mrs_name])),
xlabel('t'),ylabel('x(t)');

end % (show_plots == 1)

return
```

A.6.9 peak_complex.m

```
% peak_complex.m - Peak Enhancement
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003,2004 - All rights reserved.
%
%    Fred J. Frigo, James A. Heinen
%
%    This function accepts a 2D input (S) and returns a 2D output (Sp)
%    of the same dimension with peaks plotted as Dirac delta functions.
%
%    Changed to return complex valued peak-enhanced 2D output.
%
function Sp=peak_complex(S_input,thresh,suppresslastrow)
S=abs(S_input);
[Nsig,Nw]=size(S);
Sb=zeros(Nsig+2,Nw+2);
Sbp=zeros(Nsig+2,Nw+2);
for m=1:Nsig
   for k=1:Nw
      Sb(m+1,k+1)=S(m,k);
   end
end
for m=2:Nsig+1
   for k=2:Nw+1
      if Sb(m,k)>=max([Sb(m+1,k-1) Sb(m+1,k) Sb(m+1,k+1) Sb(m,k-1) ...
            Sb(m,k+1) Sb(m-1,k-1) Sb(m-1,k) Sb(m-1,k+1)])&Sb(m,k)>=thresh
       Sbp(m,k)=S_input(m-1,k-1);
    end
   end
end
for m=1:Nsig
   for k=1:Nw
      Sp(m,k)=Sbp(m+1,k+1);
   end
end
if suppresslastrow==1
   for k=1:Nw
      Sp(Nsig,k)=0;
   end
end
return
```

A.6.10 plot_3.m

```
% plot_3.m - Mainline script to generate plots #3
% Marquette University,   Milwaukee, WI  USA
% Copyright 2004 - All rights reserved.
% Fred J. Frigo
```

```
% Feb 10, 2004 - original
%
% Plots are obtained from sim_1_results.txt created by
%  the sim_1.m MATLAB file.
%
% Capon K=1
% 4 curves

clear all
close all

% File to read
results_file = 'sim_1_results.txt';

% SNR values
num_snr_values = 12;
snr_value=[ -18.0, -12.0, -6.0, 0.0, 6.0, 12.0, 18.0, 24.0, 30.0, 36.0, 42.0, 48.0];

% Values of M to test
num_m_values = 4;
m_values=[ 32, 64, 128, 256];

% Values of alpha1 to test
num_alp1_values = 1;
alp1_values=[ 0.0];

% Open results file
fid=fopen(results_file, 'rt');

% First three lines contain labels to discard
line1=fgets(fid);
line2=fgets(fid);
line3=fgets(fid);

% Loop through report
for sig_id=1:3
   for mloop = 1: num_m_values  % num_m_values
     for aloop = 1: num_alp1_values   % num_alp1_values
       % SNR loop
       for snrloop = 1: num_snr_values  % num_snr_values
           % read the block of data
           for field = 1: 12

               text_line=fgets(fid);
               % The 'mean' field occurs in the sub-string starting at 57
               %      this depends on the results file being used
               result_mean = sscanf(text_line(50:61),'%f');
               results(sig_id, mloop, aloop, snrloop, field)=result_mean;

            end  % field loop
        end % SNR loop
      end %alpha loop
    end % M loop
end % signal loop
% size(results)
fclose(fid);


% Average results for all signals
for mloop = 1: num_m_values  % num_m_values
    for aloop = 1: num_alp1_values  %num_alp1_values
       % SNR loop
       for snrloop = 1: num_snr_values  % num_snr_values

              avg_missed_peaks( mloop, aloop, snrloop)= mean(
results(1:3,mloop,aloop,snrloop,1));
```

```
              avg_false_peaks( mloop, aloop, snrloop)= mean(
results(1:3,mloop,aloop,snrloop,2));
              avg_mag_rms_error( mloop, aloop, snrloop)= mean(
results(1:3,mloop,aloop,snrloop,3));
              avg_sigma_rms_error( mloop, aloop, snrloop)= mean(
results(1:3,mloop,aloop,snrloop,5));

       end
       % Check for NaN and set all prior points to NaN if found.
       avg_mag_rms_error( mloop, aloop,:) = check_for_nan( avg_mag_rms_error( mloop,
aloop,:));
       avg_sigma_rms_error( mloop, aloop,:) = check_for_nan( avg_sigma_rms_error( mloop,
aloop,:));

   end %alpha loop
end % M loop

% Missed peaks plot
figure;
plot( snr_value, squeeze(avg_missed_peaks(1,1,:)), 'ko-', ...
      snr_value, squeeze(avg_missed_peaks(2,1,:)), 'bx-', ...
      snr_value, squeeze(avg_missed_peaks(3,1,:)), 'g+-', ...
      snr_value, squeeze(avg_missed_peaks(4,1,:)), 'm*-');
legend('M=32','M=64','M=128','M=256');
xlabel('SNR (dB)');
ylabel('% Missed Peaks');
title('Capon, K=1');

% False peaks
figure;
plot( snr_value, squeeze(avg_false_peaks(1,1,:)), 'ko-', ...
      snr_value, squeeze(avg_false_peaks(2,1,:)), 'bx-', ...
      snr_value, squeeze(avg_false_peaks(3,1,:)), 'g+-', ...
      snr_value, squeeze(avg_false_peaks(4,1,:)), 'm*-');
legend('M=32','M=64','M=128','M=256');
xlabel('SNR (dB)');
ylabel('% False Peaks');
title('Capon, K=1');

% Magnitude RMS errors
figure;
plot( snr_value, squeeze(avg_mag_rms_error(1,1,:)), 'ko-', ...
      snr_value, squeeze(avg_mag_rms_error(2,1,:)), 'bx-', ...
      snr_value, squeeze(avg_mag_rms_error(3,1,:)), 'g+-', ...
      snr_value, squeeze(avg_mag_rms_error(4,1,:)), 'm*-');
legend('M=32','M=64','M=128','M=256');
xlabel('SNR (dB)');
ylabel('Relative RMS Magnitude Error');
title('Capon, K=1');

% Sigma RMS errors
figure;
plot( snr_value, squeeze(avg_sigma_rms_error(1,1,:)), 'ko-', ...
      snr_value, squeeze(avg_sigma_rms_error(2,1,:)), 'bx-', ...
      snr_value, squeeze(avg_sigma_rms_error(3,1,:)), 'g+-', ...
      snr_value, squeeze(avg_sigma_rms_error(4,1,:)), 'm*-');
legend('M=32','M=64','M=128','M=256');
xlabel('SNR (dB)');
ylabel('Relative RMS Damping Error');
title('Capon, K=1');

% ------------------------------------------------------
% Combined plot
% ------------------------------------------------------

figure;
```

```
subplot(2,2,1);
plot( snr_value, squeeze(avg_missed_peaks(1,1,:)), 'ko-', ...
      snr_value, squeeze(avg_missed_peaks(2,1,:)), 'bx-', ...
      snr_value, squeeze(avg_missed_peaks(3,1,:)), 'g+-', ...
      snr_value, squeeze(avg_missed_peaks(4,1,:)), 'm*-');
legend('M=32','M=64','M=128','M=256');
xlabel('SNR (dB)');
ylabel('% Missed Peaks');


% False peaks
subplot(2,2,2);
plot( snr_value, squeeze(avg_false_peaks(1,1,:)), 'ko-', ...
      snr_value, squeeze(avg_false_peaks(2,1,:)), 'bx-', ...
      snr_value, squeeze(avg_false_peaks(3,1,:)), 'g+-', ...
      snr_value, squeeze(avg_false_peaks(4,1,:)), 'm*-');
legend('M=32','M=64','M=128','M=256');
xlabel('SNR (dB)');
ylabel('% False Peaks');

% Magnitude RMS errors
subplot(2,2,3);
plot( snr_value, squeeze(avg_mag_rms_error(1,1,:)), 'ko-', ...
      snr_value, squeeze(avg_mag_rms_error(2,1,:)), 'bx-', ...
      snr_value, squeeze(avg_mag_rms_error(3,1,:)), 'g+-', ...
      snr_value, squeeze(avg_mag_rms_error(4,1,:)), 'm*-');
legend('M=32','M=64','M=128','M=256');
xlabel('SNR (dB)');
ylabel('Relative RMS Magnitude Error');

% Adjust scale
myaxis = axis;
myaxis(3)=0;
myaxis(4)=2.0;
axis(myaxis);

% Sigma RMS errors
subplot(2,2,4);
plot( snr_value, squeeze(avg_sigma_rms_error(1,1,:)), 'ko-', ...
      snr_value, squeeze(avg_sigma_rms_error(2,1,:)), 'bx-', ...
      snr_value, squeeze(avg_sigma_rms_error(3,1,:)), 'g+-', ...
      snr_value, squeeze(avg_sigma_rms_error(4,1,:)), 'm*-');
legend('M=32','M=64','M=128','M=256');
xlabel('SNR (dB)');
ylabel('Relative RMS Damping Error');
```

A.6.11 plot_timing.m

```
% plot_timing.m - Mainline script to generate timing plots
% Marquette University,   Milwaukee, WI  USA
% Copyright 2004 - All rights reserved.
% Fred J. Frigo
% Feb 29, 2004  - original
%
% Plots are obtained from timing_sim_results.txt created by
%  the timing_sim_results.m MATLAB file.
%
% For M= 32, 64, 128, 256
% Capon (K=N):
%    alp1=alp2=bet1=bet2=0
%    alp1=alp2=0;   bet1=0.001, bet2=0.0
%    alp1=-0.5, alp2=0.0;  bet1=bet2=0
```

```
%    alp1=-0.5, alp2=0.0;  bet1=0.001, bet2=0.0
% Capon (K=1)
%    alp1=alp2=0.0
%    alp1=0.001, alp2=0.0
% APES/Capon (gamma=0.5)
%    bet1=bet2=0
%    bet1=0.001, bet2=0
%    bet1=-0.5,  bet2=0
% 9 curves

clear all
close all

% File to read
results_file = 'timing_sim_results.txt';

num_snr_values = 1;  % using SNR=18 for this simulation
num_beta_values = 9;

% M values
num_M_values=4;
M_value=[ 32, 64, 128, 256 ];

% Open results file
fid=fopen(results_file, 'rt');

% First three lines contain labels to discard
line1=fgets(fid);
line2=fgets(fid);
line3=fgets(fid);

% Loop through report
num_sigs =1;  % only one signal for this report

for sig_id=1:num_sigs
   for mloop = 1: num_M_values % num_C_values

       for bloop = 1: num_beta_values % num_beta_values

           % read the block of data
           for field = 1: 12

               text_line=fgets(fid);
               % The 'mean' field occurs in the sub-string starting at 60
               %      this depends on the results file being used
               result_mean = sscanf(text_line(71:84),'%f');
               results(sig_id, mloop, bloop, field)=result_mean;

            end  % field loop
        end % beta loop
    end % M loop
end % signal loop
fclose(fid);

% Average results for all signals
for mloop = 1: num_M_values  % num_m_values
     for bloop = 1:num_beta_values
         avg_execution_time( mloop, bloop)= mean( results(1:num_sigs,mloop, bloop,12));
     end
end %  M loop

% Execution time:  Capon
figure;
plot( M_value, squeeze(avg_execution_time(:,1)), 'ko-', ...
      M_value, squeeze(avg_execution_time(:,2)), 'bx-', ...
      M_value, squeeze(avg_execution_time(:,3)), 'g+-', ...
```

```
      M_value, squeeze(avg_execution_time(:,4)), 'md-', ...
      M_value, squeeze(avg_execution_time(:,5)), 'ks-', ...
      M_value, squeeze(avg_execution_time(:,6)), 'b^-');
legend('K=N, \alpha\neq-\sigma/2, \beta=constant', ...
       'K=N, \alpha\neq-\sigma/2, \beta=function(\sigma)',...
       'K=N, \alpha=-\sigma/2,\beta=constant',...
       'K=N, \alpha=-\sigma/2, \beta=function(\sigma)',...
       'K=1, \beta=constant', 'K=1, \beta=function(\sigma)');
xlabel('M');
ylabel('Execution time (seconds)');

% Execution time: Capon/APES
figure;
plot( M_value, squeeze(avg_execution_time(:,7)), 'ko-', ...
      M_value, squeeze(avg_execution_time(:,8)), 'bx-', ...
      M_value, squeeze(avg_execution_time(:,9)), 'g+-');
legend('\beta=constant', '\beta=function(\sigma), \beta\neq-\sigma/2',...
       '\beta=-\sigma/2');
xlabel('M');,
ylabel('Execution time (seconds)');
```

A.6.12 sim_1.m

```
% sim_1.m - Mainline script to run simulation #1
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
% Fred J. Frigo
% Dec 21, 2003  - original
% Jan 7, 2004   - Updates to perform statistics on results
% Jan 20, 2004  - Updates to modify parameters and add bias error

clear all
close all

results_file = 'sim_1_results.txt';
simulation_id = 1;

% Noise files
num_noise_frames = 10;
noise_file=[
    'noise_0.dat';
    'noise_1.dat';
    'noise_2.dat';
    'noise_3.dat';
    'noise_4.dat';
    'noise_5.dat';
    'noise_6.dat';
    'noise_7.dat';
    'noise_8.dat';
    'noise_9.dat'];

% Signal files
num_signals = 3;
signal_file=[
    'signal_1.dat';
    'signal_2.dat';
    'signal_3.dat'];

% SNR values
num_snr_values = 12;
snr_value=[ -18.0, -12.0, -6.0, 0.0, 6.0, 12.0, 18.0, 24.0, 30.0, 36.0, 42.0, 48.0];
```

```
% Signal Components: (these match signal definition in create_signals.m)
%    [ A, theta, sigma, omega_index, next_omega_flag]
%    A = magnitude
%    theta = phase
%    sigma = sigma (damping)
%    omega_index = frequency index for peak
%    next_omega_flag = 1 if OK to use next omega index (for peaks the are
%                                                          on boundaries)
%
signal_components_1 = [
    1.0,   pi/4.0,    0.002, 63,   0;
    0.75,  -pi/2.0,   0.006, 127,  0;
    0.25,  0.0,       0.004, 191,  0];

signal_components_2 = [
    1.00,   pi/4.0,   0.001,  30,  0;
   25.00,   0.0,      0.002,  40,  0;
    0.75,  -pi/4.0,   0.004,  50,  0;
    0.25,   0.0,      0.003, 100,  0;
    1.00,   0.0,      0.001, 120,  0;
   10.00,   pi/2.0,   0.002, 125,  0;
    0.50,  -pi/2.0,   0.002, 195,  0;
    1.00,   0.0,      0.003, 225,  0];

signal_components_3 = [
    0.50,   0.0,      0.004, 0,    0;
    0.75,  -pi/2.0,   0.002, 10,   0;
    1.00,   pi/4.0,   0.003, 31,   0;
    0.75,   0.0,      0.003, 34,   0;
    0.50,  -pi/2.0,   0.005, 55,   0;
    1.00,   pi/2.0,   0.001, 63,   0;
    1.00,   0.75*pi,  0.005, 66,   0;
    0.25,   0.0,      0.004, 73,   0;
    0.50,   pi/2.0,   0.000, 95,   0;
    0.50,  -pi/2.0,   0.006, 104,  0];


% Values of M to test
num_m_values = 4;
m_values=[ 32, 64, 128, 256];

N=512;

% Frame size  (max is 1024)
frame_size = N + 256;

% Get time and date
ctime = clock;

% Open results file
fid=fopen(results_file, 'wt+');
fprintf(fid,'Simulation: %2d\n', simulation_id);
fprintf(fid,'started: %2.2d/%2.2d/%4.4d  %2.2d:%2.2d:%2.2d -
\n',...
                                                ctime(2), ctime(3), ctime(1),ctime(4),
ctime(5),round(ctime(6)));
fprintf(fid,'           Parameter   signal   SNR     M     N     mean          max
min         std        variance \n');
fclose(fid);

% Signal loop
for sig_id = 1: num_signals % num_signals

   % Get data
   fidsig = fopen(signal_file(sig_id,:), 'r+b');
```

```
   [raw_data, count] = fread(fidsig, frame_size*2, 'real*4');
   fclose(fidsig);
   for m = 1:(count/2)
      sig_data(m) = raw_data((2*m)-1) + j*raw_data(2*m);
   end

   % Compute signal energy for mixing SNR
   sig_energy=sum(abs(sig_data.*sig_data));

   % Select structure with signal component info
   if (sig_id == 1)
      peak_info = signal_components_1;
   elseif (sig_id == 2)
      peak_info = signal_components_2;
   elseif (sig_id == 3)
      peak_info = signal_components_3;
   end

   % M loop
   for mloop = 1:num_m_values  % num_m_values
       M_value = m_values(mloop);

       % SNR loop
       for snrloop = 1: num_snr_values  % num_snr_values
            % Compute desired weight for noise to yield desired SNR ratio in dB
            SNR_in_dB = snr_value(snrloop);
            SNR_exponent = SNR_in_dB/10.0;  % numerator = desired SNR ratio in dB

            % Noise loop
            for noise_id = 1:num_noise_frames   % num_noise_frames
               % Get noise
               fidn = fopen(noise_file(noise_id,:), 'r+b');
               [raw_data, count] = fread(fidn, frame_size*2, 'real*4');
               fclose(fidn);
               for m = 1:(count/2)
                   noise(m) = raw_data((2*m)-1) + j*raw_data(2*m);
                   noise(m)=0.1*noise(m);
               end

               % Compute Noise energy for creating desired mix of
               %  signal + noise
               noise_energy=sum(abs(noise.*noise));
               noise_weight = sqrt( (sig_energy/noise_energy)/power(10.0,SNR_exponent));

               % build the signal plus noise
               x= sig_data + (noise_weight*noise);

               % Call function to do processing on algorithm to test
               [missed_peaks(noise_id), false_peaks(noise_id), mag_err_rms(noise_id),
phase_err_rms(noise_id),...
                        sigma_err_rms(noise_id), mag_err_pct(noise_id),
phase_err_pct(noise_id), ...
                        sigma_err_pct(noise_id), mag_err_bias(noise_id),
phase_err_bias(noise_id), ...
                        sigma_err_bias(noise_id), elapsed_time(noise_id)] = sim_1_proc(x,
M_value, peak_info);

            end  % noise loop

            % Log results for each parameter for a set of noise instances
            for result_loop=1:12

                if result_loop == 1
                    result_data = missed_peaks;
                    result_string = 'Missed_Peaks';
                elseif result_loop == 2
```

```
                    result_data = false_peaks;
                    result_string = 'False_Peaks';
                elseif result_loop == 3
                     result_data = mag_err_rms;
                     result_string = 'Mag_Err_RMS';
                elseif result_loop == 4
                     result_data = phase_err_rms;
                     result_string = 'Phase_Err_RMS';
                elseif result_loop == 5
                     result_data = sigma_err_rms;
                     result_string = 'Sigma_Err_RMS';
                elseif result_loop == 6
                     result_data = mag_err_pct;
                     result_string = 'Mag_Err_percent';
                elseif result_loop == 7
                     result_data = phase_err_pct;
                     result_string = 'Phase_Err_percent';
                elseif result_loop == 8
                     result_data = sigma_err_pct;
                     result_string = 'Sigma_Err_percent';
                elseif result_loop == 9
                     result_data = mag_err_bias;
                     result_string = 'Mag_Err_Bias';
                elseif result_loop == 10
                     result_data = phase_err_bias;
                     result_string = 'Phase_Err_Bias';
                elseif result_loop == 11
                     result_data = sigma_err_bias;
                     result_string = 'Sigma_Err_Bias';
                elseif result_loop == 12
                     result_data = elapsed_time;
                     result_string = 'Elapsed_time';
                end

                % Discard invalid results
                num_results=0;
                for (rindex = 1:num_noise_frames)
                    % If result is valid number, store it for computation
                    if ( isnan(result_data(rindex)) == 0 )
                        num_results = num_results+1;
                        valid_results(num_results) = result_data(rindex);
                    end
                end
                % Compute mean, std and variance on valid results.
                if ( num_results > 0 )
                    result_mean = mean(valid_results(1:num_results));
                    result_std = std(valid_results(1:num_results));
                    result_var = var(valid_results(1:num_results));
                    result_max = max(valid_results(1:num_results));
                    result_min = min(valid_results(1:num_results));
                else
                    result_mean = NaN;
                    result_std = NaN;
                    result_var = NaN;
                    result_max = NaN;
                    result_min = NaN;
                end

                % Log results to output file:
                % 'parameter  signal  SNR    M    N    mean     max    min     std
variance'
                fid=fopen(results_file, 'at');
                fprintf(fid,'%20.20s     %2.2d    %5.1f   %3.3d   %2.2d   %9.5f     %9.5f
%9.5f   %9.5f   %9.5f\n', ...
                       result_string, sig_id, SNR_in_dB, M_value, num_results,
result_mean, result_max, ...
```

```
                    result_min, result_std, result_var );
                fclose(fid);

            end  % results loop
        end % SNR loop
    end % M loop
end % signal loop

% Get time and date to add for "completed timestamp"
ctime = clock;
fid=fopen(results_file, 'rt+');
fseek(fid,47,0); % insert timestamp at beginning of file after "started"
fprintf(fid,' completed: %2.2d/%2.2d/%4.4d  %2.2d:%2.2d:%2.2d',...
                ctime(2), ctime(3), ctime(1),ctime(4), ctime(5),round(ctime(6)));
fclose(fid);
```

A.6.13 sim_1_proc.m

```
% sim_1_proc.m - Processing associated with simulation #1
% Marquette University,   Milwaukee, WI  USA
% Copyright 2003, 2004 - All rights reserved.
% Fred J. Frigo
% Dec 21, 2003  - original
% Jan 20, 2004  - updates for calculating errors
%
%  Inputs:
%           input_signal = signal+noise
%           M = filter length
%           peak_info = contains info about each peak in the signal
%                       [ A, theta, sigma, omega_index, next_omega_flag]
%                       A = expected magnitude
%                       theta = expected phase
%                       sigma = expected sigma (damping)
%                       omega_index = omega index for expected peak
%                       next_omega_flag = 1 if OK to use next omega index
%
%  Outputs:
%           missed_peaks = percentage of missed peaks
%           false_peaks = percentage of false peaks
%           rms_mag_err = RMS error for magnitude
%           rms_phase_err = RMS error for phase
%           rms_sigma_err = RMS error for sigma (damping)
%           pct_mag_err = RMS error for magnitude
%           pct_phase_err = RMS error for phase
%           pct_sigma_err = RMS error for sigma (damping)
%           elapsed_time = elaspsed time in seconds

function [missed_peaks, false_peaks, rms_mag_err, rms_phase_err, rms_sigma_err, ...
             pct_mag_err, pct_phase_err, pct_sigma_err, ...
             bias_mag_err, bias_phase_err, bias_sigma_err,elapsed_time] =
sim_1_proc(input_signal, M, peak_info);

matched_peaks = 0;
total_peaks = 0;
missed_peaks = 0.0;
false_peaks = 0.0;
rms_mag_err = NaN;  % If no peaks are found, RMS error = NaN
rms_phase_err = NaN;
rms_sigma_err = NaN;
pct_mag_err = NaN; % If no peaks are found, Percent error = NaN
pct_phase_err = NaN;
```

```
pct_sigma_err = NaN;
bias_mag_err = NaN; % If no peaks are found, Bias error = NaN
bias_phase_err = NaN;
bias_sigma_err = NaN;

% Find out how many signal components there are
num_expected_peaks = max( size(peak_info(:,5)));

% set flag = 1 to display all plots
show_plots=0;

N=512;
%M=256;
Ns=1000;   %1800
Nf=1024;%0<Ns<Nf.  Nf>=N+M-1.
K=1;%K must be 1 or N.  alp1 and alp2 are irrelevant if K=1.
Nsig=40;
Nw=N/2;%Nw must be greater than or equal to N/2.
k1=0;
k2=Nw-1;%0<=k1<k2<=Nw-1.
delsig=0.0002;
alp1=0;
alp2=0;
bet1=0;  % 0 = default - set this to 0.5 to increase peak picking sensitivity
bet2=0;
gam=0;%0<gam<=1.  gam=1 reduces to 2D APES.  gam=0 would reduce to 2D Capon.
C=1;

spectrumtype=1;%=1 for 2D Capon, 2 for 2D APES/Capon.

delw=pi/Nw;

%  signal plus noise
x=input_signal(:,1:N+M-1);

if C>1

   if gtype==1&datatype==1
      g=gideal;
   elseif gtype==2
      g=fgest(xlong,C);
      %g=fgest(x,C);
   end

   if rhosqtype==1&datatype==1
      rhosq=rhosqideal;
   elseif rhosqtype==2
      rhosq=frhosqest(xlong,Ns,Nf,C);
   end

   Rw=diag(rhosq);
   Rwinv=inv(Rw);
   w=Rwinv*g/(g'*Rwinv*g);

   xest=w'*x;

else % if C == 1, then g = 1
    g=1;
end

tic;

if (C==1)|(C>1&combotype==3)
   if spectrumtype==1
      S=Capon2D(x,N,M,K,Nsig,Nw,k1,k2,delsig,alp1,alp2,bet1,bet2,C,g);
   elseif spectrumtype==2
```

```
      S=APESCapon2D(x,N,M,Nsig,Nw,k1,k2,delsig,bet1,bet2,gam,C,g);
   end
end

if C>1&combotype==1
   if spectrumtype==1
      S=Capon2D(xest,N,M,K,Nsig,Nw,k1,k2,delsig,alp1,alp2,bet1,bet2,1,1);
   elseif spectrumtype==2
      S=APESCapon2D(xest,N,M,Nsig,Nw,k1,k2,delsig,bet1,bet2,gam,1,1);
   end
end

if C>1&combotype==2
   S=zeros(Nsig,k2-k1+1);
   for i=1:C
      if spectrumtype==1
         Si=Capon2D(x(i,:),N,M,K,Nsig,Nw,k1,k2,delsig,alp1,alp2,bet1,bet2,1,1);
      elseif spectrumtype==2
         Si=APESCapon2D(x(i,:),N,M,Nsig,Nw,k1,k2,delsig,bet1,bet2,gam,1,1);
      end
      S=S+conj(w(i))*Si;
   end
end

elapsed_time = toc;

sigset=delsig*[0:Nsig-1];
wset=delw*[k1:k2];
n=[0:N+M-2];

% Create peak enhanced 2D surface where each peak is a dirac delta function
Sp=peak_complex(S,0,1);

% Compute peak projections
Spp=peakproj(Sp,0);

% Set threshold for valid peaks:  (Max component amplitude = 1.0)
%  - Use max projection (for determination of threshold)
% max_peak_value = max(abs(Spp));
% peak_threshold = max_peak_value*0.1;
peak_threshold = 0.025;

% Count number of peaks that exceed threshold
total_peaks = count_peaks(Spp, peak_threshold);

% Search for expected signal components, computing error terms if found
rms_mag_sum = 0.0;
rms_phase_sum = 0.0;
rms_sigma_sum = 0.0;
pct_mag_sum = 0.0;
pct_phase_sum = 0.0;
pct_sigma_sum = 0.0;
bias_mag_sum = 0.0;
bias_phase_sum = 0.0;
bias_sigma_sum = 0.0;
peaks_found = 0;
for peak_component = 1: num_expected_peaks
    component_info = squeeze(peak_info(peak_component, :));
    % Search for expected peak calculating magnitude, phase and damping errors
    [found_peak_flag, sq_mag_err, sq_phase_err, sq_sigma_err, abs_mag_err, abs_phase_err,
...
           abs_sigma_err, diff_mag_err, diff_phase_err, diff_sigma_err] = find_peak( Sp,
component_info, delw, delsig, peak_threshold );
    rms_mag_sum = rms_mag_sum + sq_mag_err;
    rms_phase_sum = rms_phase_sum + sq_phase_err;
    rms_sigma_sum = rms_sigma_sum + sq_sigma_err;
```

```
    pct_mag_sum = pct_mag_sum + abs_mag_err;
    pct_phase_sum = pct_phase_sum + abs_phase_err;
    pct_sigma_sum = pct_sigma_sum + abs_sigma_err;
    bias_mag_sum = bias_mag_sum + diff_mag_err;
    bias_phase_sum = bias_phase_sum + diff_phase_err;
    bias_sigma_sum = bias_sigma_sum + diff_sigma_err;


    if (found_peak_flag == 1)
        peaks_found = peaks_found + 1;
    end
end

% Compute error values if we found any peaks
if peaks_found > 0
    rms_mag_err = sqrt( (rms_mag_sum/peaks_found) );
    rms_phase_err = sqrt( (rms_phase_sum/peaks_found) );
    rms_sigma_err = sqrt( (rms_sigma_sum/peaks_found) );
    pct_mag_err = (pct_mag_sum/peaks_found)*100.0;
    pct_phase_err = (pct_phase_sum/peaks_found)*100.0;
    pct_sigma_err = (rms_sigma_sum/peaks_found)*100.0;
    bias_mag_err = bias_mag_sum/peaks_found;
    bias_phase_err = bias_phase_sum/peaks_found;
    bias_sigma_err = bias_sigma_sum/peaks_found;
end

% False positives ( as a percentage)
false_peaks = ((total_peaks - peaks_found)/Nw)*100.0;

% Missed peaks (as a percentage)
missed_peaks = ((num_expected_peaks - peaks_found)/num_expected_peaks)*100.0;

% Create plots if selected.
if (show_plots == 1)

   mrs_name='x(t)';

   % Plots
   if( spectrumtype == 1)
     analysis_string = '2D-Capon ';
   else
     analysis_string = '2D-APES ';
   end

   figure
   title_string = strcat([analysis_string, 'surface plot of |S(\sigma,\omega)| for ',
mrs_name]);
   surf(wset,sigset,abs(S)),title(title_string), xlabel('\omega'),
ylabel('\sigma'),zlabel('|S(\sigma,\omega)|');

   figure
   title_string = strcat([analysis_string, 'contour plot of |S(\sigma,\omega)| for ',
mrs_name]);
   contour(wset,sigset,abs(S)),title(title_string),xlabel('\omega'),ylabel('\sigma');

   figure
   title_string = strcat([analysis_string, 'peak enhanced surface plot of
|S(\sigma,\omega)| for ', mrs_name]);

surf(wset,sigset,abs(Sp)),title(title_string),xlabel('\omega'),ylabel('\sigma'),zlabel('|
S(\sigma,\omega)|');

   figure
   title_string = strcat([analysis_string, 'peak enhanced contour plot of
|S(\sigma,\omega)| for ', mrs_name]);
   contour(wset,sigset,abs(Sp)),title(title_string),xlabel('\omega'),ylabel('\sigma');
```

```
   figure
   title_string = strcat([analysis_string, 'projected peaks of |S(\sigma,\omega)| for ',
mrs_name]);
   plot(wset,Spp),title(title_string),xlabel('\omega'),ylabel('|S(\sigma,\omega)|');

   X=fft(x.').';
   [Cft,Nft]=size(abs(X));
   k1ft=floor(Nft*k1/(2*Nw));
   k2ft=floor(Nft*k2/(2*Nw));
   wfset=[k1ft:k2ft]*2*pi/Nft;
   figure
   plot(wfset,abs(X(:,k1ft+1:k2ft+1))/Nft),title(strcat(['Fourier Transform of
',mrs_name])),xlabel('\omega'),ylabel('|S(\omega)|');
   if C>1
      Xest=fft(xest.').';
      figure
      plot(wfset,abs(Xest(:,k1ft+1:k2ft+1))/Nft),title('average abs(FFT)'),xlabel(paras)
   end
   figure
   plot(n,abs(x).'),title(strcat(['Input signal ',mrs_name])),
xlabel('t'),ylabel('x(t)');

end % (show_plots == 1)

return
```

A.6.14 timing_sim.m

```
% timing_sim.m - Mainline script to run timing simulation
% Marquette University,   Milwaukee, WI  USA
% Copyright 2004 - All rights reserved.
% Fred J. Frigo
% Feb 29, 2004  - original


clear all
close all

results_file = 'timing_sim_results.txt';
simulation_id = 14;

% Noise files
num_noise_frames = 5;  %10
noise_file=[
    'noise_0.dat';
    'noise_1.dat';
    'noise_2.dat';
    'noise_3.dat';
    'noise_4.dat';
    'noise_5.dat';
    'noise_6.dat';
    'noise_7.dat';
    'noise_8.dat';
    'noise_9.dat'];

% Signal files
num_signals = 3;
signal_file=[
    'signal_1.dat';
    'signal_2.dat';
    'signal_3.dat'];
```

```
% SNR values
num_snr_values = 1;
snr_value=[ 18.0 ];


% Signal Components: (these match signal definition in create_signals.m)
%    [ A, theta, sigma, omega_index, next_omega_flag]
%    A = magnitude
%    theta = phase
%    sigma = sigma (damping)
%    omega_index = frequency index for peak
%    next_omega_flag = 1 if OK to use next omega index (for peaks the are
%                                                          on boundaries)
%
signal_components_1 = [
    1.0,   pi/4.0,    0.002, 63,   0;
    0.75,  -pi/2.0,   0.006, 127,  0;
    0.25,  0.0,       0.004, 191,  0];

signal_components_2 = [
    1.00,   pi/4.0,   0.001,  30,  0;
   25.00,   0.0,      0.002,  40,  0;
    0.75,  -pi/4.0,   0.004,  50,  0;
    0.25,   0.0,      0.003, 100,  0;
    1.00,   0.0,      0.001, 120,  0;
   10.00,   pi/2.0,   0.002, 125,  0;
    0.50,  -pi/2.0,   0.002, 195,  0;
    1.00,   0.0,      0.003, 225,  0];

signal_components_3 = [
    0.50,   0.0,      0.004, 0,    0;
    0.75,  -pi/2.0,   0.002, 10,   0;
    1.00,   pi/4.0,   0.003, 31,   0;
    0.75,   0.0,      0.003, 34,   0;
    0.50,  -pi/2.0,   0.005, 55,   0;
    1.00,   pi/2.0,   0.001, 63,   0;
    1.00,   0.75*pi,  0.005, 66,   0;
    0.25,   0.0,      0.004, 73,   0;
    0.50,   pi/2.0,   0.000, 95,   0;
    0.50,  -pi/2.0,   0.006, 104,  0];

% Number of M values to test
num_m_values = 4;
m_values=[ 32, 64, 128, 256];

N=512;

% Values of K, alpha1, beta1, spectrum_type
num_beta_values = 9;
K_values =   [   N,    N,       N,     N,     1,      1,      N,     N,       N];
alp1_values= [ 0.0,  0.0,    -0.5,  -0.5,     0.0,    0.0,    0.0,   0.0,     0.0];
bet1_values= [ 0.0,  0.001,   0.0,   0.001,   0.0,    0.001,  0.0,   0.001,  -0.5];
spect_vals = [   1,     1,     1,       1,     1,        1,    2,       2,      2];


% Frame size  (max is 1024)
frame_size = N + 256;

% Get time and date
ctime = clock;

% Open results file
fid=fopen(results_file, 'wt+');
fprintf(fid,'Simulation: %2d\n', simulation_id);
fprintf(fid,'started: %2.2d/%2.2d/%4.4d  %2.2d:%2.2d:%2.2d -
\n',...
```

```
                                                    ctime(2), ctime(3), ctime(1),ctime(4),
ctime(5),round(ctime(6)));
fprintf(fid,'           Parameter   signal   SNR   K    alp1    bet1   M   spec   N
mean          max          min         std        variance \n');
fclose(fid);

% Signal loop
for sig_id = 3: 3 % num_signals

   % Get data
   fidsig = fopen(signal_file(sig_id,:), 'r+b');
   [raw_data, count] = fread(fidsig, frame_size*2, 'real*4');
   fclose(fidsig);
   for m = 1:(count/2)
      sig_data(m) = raw_data((2*m)-1) + j*raw_data(2*m);
   end

   % Compute signal energy for mixing SNR
   sig_energy=sum(abs(sig_data.*sig_data));

   % Select structure with signal component info
   if (sig_id == 1)
      peak_info = signal_components_1;
   elseif (sig_id == 2)
      peak_info = signal_components_2;
   elseif (sig_id == 3)
      peak_info = signal_components_3;
   end

   % M loop
   for mloop = 1: num_m_values  % num_m_values
     M_value = m_values(mloop);

     for bloop = 1: num_beta_values  % num_beta_values
       bet1 = bet1_values( bloop );
       alp1 = alp1_values( bloop );
       spectrum_type = spect_vals( bloop);
       K_value = K_values( bloop );

       % SNR loop
       for snrloop = 1: num_snr_values  % num_snr_values
            % Compute desired weight for noise to yield desired SNR ratio in dB
            SNR_in_dB = snr_value(snrloop);
            SNR_exponent = SNR_in_dB/10.0;  % numerator = desired SNR ratio in dB

            % Noise loop
            for noise_id = 1:num_noise_frames   % num_noise_frames
               % Get noise
               fidn = fopen(noise_file(noise_id,:), 'r+b');
               [raw_data, count] = fread(fidn, frame_size*2, 'real*4');
               fclose(fidn);
               for m = 1:(count/2)
                   noise(m) = raw_data((2*m)-1) + j*raw_data(2*m);
                   noise(m)=0.1*noise(m);
               end

               % Compute Noise energy for creating desired mix of
               %  signal + noise
               noise_energy=sum(abs(noise.*noise));
               noise_weight = sqrt( (sig_energy/noise_energy)/power(10.0,SNR_exponent));

               % build the signal plus noise
               x= sig_data + (noise_weight*noise);

               % Call function to do processing on algorithm to test
```

```
              [missed_peaks(noise_id), false_peaks(noise_id), mag_err_rms(noise_id),
phase_err_rms(noise_id),...
                    sigma_err_rms(noise_id), mag_err_pct(noise_id),
phase_err_pct(noise_id), ...
                    sigma_err_pct(noise_id), mag_err_bias(noise_id),
phase_err_bias(noise_id), ...
                    sigma_err_bias(noise_id), elapsed_time(noise_id)] =
timing_sim_proc(x, K_value, alp1, bet1, M_value, spectrum_type, peak_info);

            end  % noise loop

            % Log results for each parameter for a set of noise instances
            for result_loop=1:12

                if result_loop == 1
                    result_data = missed_peaks;
                    result_string = 'Missed_Peaks';
                elseif result_loop == 2
                    result_data = false_peaks;
                    result_string = 'False_Peaks';
                elseif result_loop == 3
                     result_data = mag_err_rms;
                     result_string = 'Mag_Err_RMS';
                elseif result_loop == 4
                     result_data = phase_err_rms;
                     result_string = 'Phase_Err_RMS';
                elseif result_loop == 5
                     result_data = sigma_err_rms;
                     result_string = 'Sigma_Err_RMS';
                elseif result_loop == 6
                     result_data = mag_err_pct;
                     result_string = 'Mag_Err_percent';
                elseif result_loop == 7
                     result_data = phase_err_pct;
                     result_string = 'Phase_Err_percent';
                elseif result_loop == 8
                     result_data = sigma_err_pct;
                     result_string = 'Sigma_Err_percent';
                elseif result_loop == 9
                     result_data = mag_err_bias;
                     result_string = 'Mag_Err_Bias';
                elseif result_loop == 10
                     result_data = phase_err_bias;
                     result_string = 'Phase_Err_Bias';
                elseif result_loop == 11
                     result_data = sigma_err_bias;
                     result_string = 'Sigma_Err_Bias';
                elseif result_loop == 12
                     result_data = elapsed_time;
                     result_string = 'Elapsed_time';
                end

                % Discard invalid results
                num_results=0;
                for (rindex = 1:num_noise_frames)
                    % If result is valid number, store it for computation
                    if ( isnan(result_data(rindex)) == 0 )
                        num_results = num_results+1;
                        valid_results(num_results) = result_data(rindex);
                    end
                end
                % Compute mean, std and variance on valid results.
                if ( num_results > 0 )
                    result_mean = mean(valid_results(1:num_results));
                    result_std = std(valid_results(1:num_results));
                    result_var = var(valid_results(1:num_results));
```

```
                    result_max = max(valid_results(1:num_results));
                    result_min = min(valid_results(1:num_results));
                else
                    result_mean = NaN;
                    result_std = NaN;
                    result_var = NaN;
                    result_max = NaN;
                    result_min = NaN;
                end

                % Log results to output file:
                % 'Parameter   signal   SNR  K   alp1  bet1   M   spect_type   N    mean
max    min     std     variance'
                fid=fopen(results_file, 'at');
                fprintf(fid,'%20.20s     %1.1d    %4.1f   %3.3d   %4.1f  %6.3f  %3.3d
%1.1d    %2.2d   %9.5f     %9.5f    %9.5f   %9.5f   %9.5f\n', ...
                        result_string, sig_id, SNR_in_dB, K_value, alp1, bet1, M_value,
spectrum_type, num_results, result_mean, result_max, ...
                        result_min, result_std, result_var );
                fclose(fid);

            end  % results loop
        end % SNR loop
      end % beta loop
    end % M loop
end % signal loop

% Get time and date to add for "completed timestamp"
ctime = clock;
fid=fopen(results_file, 'rt+');
fseek(fid,47,0); % insert timestamp at beginning of file after "started"
fprintf(fid,' completed: %2.2d/%2.2d/%4.4d  %2.2d:%2.2d:%2.2d',...
                ctime(2), ctime(3), ctime(1),ctime(4), ctime(5),round(ctime(6)));
fclose(fid);
```

A.6.15 timing_sim_proc.m

```
% timing_sim_proc.m - Processing associated with timing simulation
% Marquette University,   Milwaukee, WI  USA
% Copyright 2004 - All rights reserved.
% Fred J. Frigo
% Feb 29, 2004  - original
%
%  Inputs:
%           input_signal = signal+noise
%           K = estimation length
%           alp1, bet1 = peak sensistivity parameters
%           M = filter length
%           spectrum_type = 1 = Capon; 2 = Capon/APES
%           peak_info = contains info about each peak in the signal
%                       [ A, theta, sigma, omega_index, next_omega_flag]
%                       A = expected magnitude
%                       theta = expected phase
%                       sigma = expected sigma (damping)
%                       omega_index = omega index for expected peak
%                       next_omega_flag = 1 if OK to use next omega index
%
%  Outputs:
%           missed_peaks = percentage of missed peaks
%           false_peaks = percentage of false peaks
%           rms_mag_err = RMS error for magnitude
%           rms_phase_err = RMS error for phase
%           rms_sigma_err = RMS error for sigma (damping)
```

```
%           pct_mag_err = RMS error for magnitude
%           pct_phase_err = RMS error for phase
%           pct_sigma_err = RMS error for sigma (damping)
%           elapsed_time = elaspsed time in seconds

function [missed_peaks, false_peaks, rms_mag_err, rms_phase_err, rms_sigma_err, ...
             pct_mag_err, pct_phase_err, pct_sigma_err, ...
             bias_mag_err, bias_phase_err, bias_sigma_err,elapsed_time] = ...
             timing_sim_proc(input_signal, K, alp1, bet1, M, spectrum_type, peak_info);

matched_peaks = 0;
total_peaks = 0;
missed_peaks = 0.0;
false_peaks = 0.0;
rms_mag_err = NaN;  % If no peaks are found, RMS error = NaN
rms_phase_err = NaN;
rms_sigma_err = NaN;
pct_mag_err = NaN; % If no peaks are found, Percent error = NaN
pct_phase_err = NaN;
pct_sigma_err = NaN;
bias_mag_err = NaN; % If no peaks are found, Bias error = NaN
bias_phase_err = NaN;
bias_sigma_err = NaN;

% Find out how many signal components there are
num_expected_peaks = max( size(peak_info(:,5)));

% set flag = 1 to display all plots
show_plots=0;

N=512;
%M=256;
Ns=1000;   %1800
Nf=1024;%0<Ns<Nf.  Nf>=N+M-1.
% K=N;%K must be 1 or N.  alp1 and alp2 are irrelevant if K=1.
Nsig=40;
Nw=N/2;%Nw must be greater than or equal to N/2.
k1=0;
k2=Nw-1;%0<=k1<k2<=Nw-1.
delsig=0.0002;
%alp1=0;
alp2=0;
%bet1=0;  % 0 = default - set this to 0.5 to increase peak picking sensitivity
bet2=0;
gam=0.5;%0<gam<=1.  gam=1 reduces to 2D APES.  gam=0 would reduce to 2D Capon.
C=1;

spectrumtype=2;%=1 for 2D Capon, 2 for 2D APES/Capon.

delw=pi/Nw;

%  signal plus noise
x=input_signal(:,1:N+M-1);

if C>1

   if gtype==1&datatype==1
      g=gideal;
   elseif gtype==2
      g=fgest(xlong,C);
      %g=fgest(x,C);
   end

   if rhosqtype==1&datatype==1
      rhosq=rhosqideal;
   elseif rhosqtype==2
```

```
      rhosq=frhosqest(xlong,Ns,Nf,C);
   end

   Rw=diag(rhosq);
   Rwinv=inv(Rw);
   w=Rwinv*g/(g'*Rwinv*g);

   xest=w'*x;

else % if C == 1, then g = 1
    g=1;
end

tic;

if (C==1)|(C>1&combotype==3)
   if spectrumtype==1
      S=Capon2D(x,N,M,K,Nsig,Nw,k1,k2,delsig,alp1,alp2,bet1,bet2,C,g);
   elseif spectrumtype==2
      S=APESCapon2D(x,N,M,Nsig,Nw,k1,k2,delsig,bet1,bet2,gam,C,g);
   end
end

if C>1&combotype==1
   if spectrumtype==1
      S=Capon2D(xest,N,M,K,Nsig,Nw,k1,k2,delsig,alp1,alp2,bet1,bet2,1,1);
   elseif spectrumtype==2
      S=APESCapon2D(xest,N,M,Nsig,Nw,k1,k2,delsig,bet1,bet2,gam,1,1);
   end
end

if C>1&combotype==2
   S=zeros(Nsig,k2-k1+1);
   for i=1:C
      if spectrumtype==1
         Si=Capon2D(x(i,:),N,M,K,Nsig,Nw,k1,k2,delsig,alp1,alp2,bet1,bet2,1,1);
      elseif spectrumtype==2
         Si=APESCapon2D(x(i,:),N,M,Nsig,Nw,k1,k2,delsig,bet1,bet2,gam,1,1);
      end
      S=S+conj(w(i))*Si;
   end
end

elapsed_time = toc;

sigset=delsig*[0:Nsig-1];
wset=delw*[k1:k2];
n=[0:N+M-2];

% Create peak enhanced 2D surface where each peak is a dirac delta function
Sp=peak_complex(S,0,1);

% Compute peak projections
Spp=peakproj(Sp,0);

% Set threshold for valid peaks:  (Max component amplitude = 1.0)
%  - Use max projection (for determination of threshold)
% max_peak_value = max(abs(Spp));
% peak_threshold = max_peak_value*0.1;
peak_threshold = 0.025;

% Count number of peaks that exceed threshold
total_peaks = count_peaks(Spp, peak_threshold);

% Search for expected signal components, computing error terms if found
rms_mag_sum = 0.0;
```

```
rms_phase_sum = 0.0;
rms_sigma_sum = 0.0;
pct_mag_sum = 0.0;
pct_phase_sum = 0.0;
pct_sigma_sum = 0.0;
bias_mag_sum = 0.0;
bias_phase_sum = 0.0;
bias_sigma_sum = 0.0;
peaks_found = 0;
for peak_component = 1: num_expected_peaks
    component_info = squeeze(peak_info(peak_component, :));
    % Search for expected peak calculating magnitude, phase and damping errors
    [found_peak_flag, sq_mag_err, sq_phase_err, sq_sigma_err, abs_mag_err, abs_phase_err,
...
           abs_sigma_err, diff_mag_err, diff_phase_err, diff_sigma_err] = find_peak( Sp,
component_info, delw, delsig, peak_threshold );
    rms_mag_sum = rms_mag_sum + sq_mag_err;
    rms_phase_sum = rms_phase_sum + sq_phase_err;
    rms_sigma_sum = rms_sigma_sum + sq_sigma_err;
    pct_mag_sum = pct_mag_sum + abs_mag_err;
    pct_phase_sum = pct_phase_sum + abs_phase_err;
    pct_sigma_sum = pct_sigma_sum + abs_sigma_err;
    bias_mag_sum = bias_mag_sum + diff_mag_err;
    bias_phase_sum = bias_phase_sum + diff_phase_err;
    bias_sigma_sum = bias_sigma_sum + diff_sigma_err;


    if (found_peak_flag == 1)
        peaks_found = peaks_found + 1;
    end
end

% Compute error values if we found any peaks
if peaks_found > 0
    rms_mag_err = sqrt( (rms_mag_sum/peaks_found) );
    rms_phase_err = sqrt( (rms_phase_sum/peaks_found) );
    rms_sigma_err = sqrt( (rms_sigma_sum/peaks_found) );
    pct_mag_err = (pct_mag_sum/peaks_found)*100.0;
    pct_phase_err = (pct_phase_sum/peaks_found)*100.0;
    pct_sigma_err = (rms_sigma_sum/peaks_found)*100.0;
    bias_mag_err = bias_mag_sum/peaks_found;
    bias_phase_err = bias_phase_sum/peaks_found;
    bias_sigma_err = bias_sigma_sum/peaks_found;
end

% False positives ( as a percentage)
false_peaks = ((total_peaks - peaks_found)/Nw)*100.0;

% Missed peaks (as a percentage)
missed_peaks = ((num_expected_peaks - peaks_found)/num_expected_peaks)*100.0;

% Create plots if selected.
if (show_plots == 1)

   mrs_name='x(t)';

   % Plots
   if( spectrumtype == 1)
     analysis_string = '2D-Capon ';
   else
     analysis_string = '2D-APES ';
   end

   figure
   title_string = strcat([analysis_string, 'surface plot of |S(\sigma,\omega)| for ',
mrs_name]);
```

```
   surf(wset,sigset,abs(S)),title(title_string), xlabel('\omega'),
ylabel('\sigma'),zlabel('|S(\sigma,\omega)|');

   figure
   title_string = strcat([analysis_string, 'contour plot of |S(\sigma,\omega)| for ',
mrs_name]);
   contour(wset,sigset,abs(S)),title(title_string),xlabel('\omega'),ylabel('\sigma');

   figure
   title_string = strcat([analysis_string, 'peak enhanced surface plot of
|S(\sigma,\omega)| for ', mrs_name]);

surf(wset,sigset,abs(Sp)),title(title_string),xlabel('\omega'),ylabel('\sigma'),zlabel('|
S(\sigma,\omega)|');

   figure
   title_string = strcat([analysis_string, 'peak enhanced contour plot of
|S(\sigma,\omega)| for ', mrs_name]);
   contour(wset,sigset,abs(Sp)),title(title_string),xlabel('\omega'),ylabel('\sigma');

   figure
   title_string = strcat([analysis_string, 'projected peaks of |S(\sigma,\omega)| for ',
mrs_name]);
   plot(wset,Spp),title(title_string),xlabel('\omega'),ylabel('|S(\sigma,\omega)|');

   X=fft(x.').';
   [Cft,Nft]=size(abs(X));
   k1ft=floor(Nft*k1/(2*Nw));
   k2ft=floor(Nft*k2/(2*Nw));
   wfset=[k1ft:k2ft]*2*pi/Nft;
   figure
   plot(wfset,abs(X(:,k1ft+1:k2ft+1))/Nft),title(strcat(['Fourier Transform of
',mrs_name])),xlabel('\omega'),ylabel('|S(\omega)|');
   if C>1
      Xest=fft(xest.').';
      figure
      plot(wfset,abs(Xest(:,k1ft+1:k2ft+1))/Nft),title('average abs(FFT)'),xlabel(paras)
   end
   figure
   plot(n,abs(x).'),title(strcat(['Input signal ',mrs_name])),
xlabel('t'),ylabel('x(t)');

end % (show_plots == 1)

return
```