A MODULAR COMPUTER PROGRAM FOR THE ACQUISITION AND ANALYSIS OF BIOMAGNETIC SIGNALS USING SQUID MAGNETOMETERS

by

Andrei Irimia

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Computer Science

August, 2004

Nashville, Tennessee

Approved

J. Michael FitzPatrick

Leonard Alan Bradshaw

To my family

ACKNOWLEDGEMENTS

I would like to thank Drs. L. Alan Bradshaw, J. Michael Fitzpatrick and Andrew Pullan for their continuous support and supervision of this research project. I am grateful to Dr. Bradshaw for introducing me to the interesting scientific area of biophysical electrodynamics and for guiding me throughout my studies and intellectual growth at Vanderbilt University. I also appreciate the enthusiastic support of Dr. Fitzpatrick since the beginning of my graduate studies and I express my thanks to him for supervising this master's thesis. I am indebted to Dr. Andrew Pullan of the University of Auckland, New Zealand, for his guidance and sustained contribution to this research project.

I also appreciate the encouragement, interest and support of Dr. William O. Richards of the Department of Surgery at the Vanderbilt University Medical Center, particularly throughout the development phase of the software that constitutes the object of my thesis. I am thankful to Robert Palmer for sharing the source code of his SQUID analysis software with me and for giving me useful indications with respect to the programming aspects of my work. Many thanks to Jared A. Sims for his help with the data acquisition part of the project and for his contribution to our published work. I would also like to thank our New Zealand collaborators at the University of Auckland, Leo Cheng and Martin Buist, for sharing their experience of CMISS with me and for helping me with all the aspects of CMISS input data conversion.

I am thankful to the entire faculty in the Departments of Electrical Engineering, Computer Science, Physics and Astronomy at Vanderbilt University for their friendship, encouragement and collaboration. I am particularly grateful to Drs. Charlotte Froese Fischer of Vanderbilt University and Georgio Tachiev of Florida International University for our interesting collaboration in computational atomic theory. I would also like to thank Drs. Thomas J. Weiler and Thomas W. Kephart for their encouragement and for our scientific collaboration in high-energy particle theory.

I am very grateful to all my family in Romania for their continuous support and encouragement. The completion of this work would have been impossible without the support of my Romanian friend Cristian Onofrei. I am also very grateful to my Albanian friend Denada Duka and to my wonderful Chinese and Peruvian friends from Stanford University, Lu Shan and Daniel Alcazar-Roman. I would like to mention my Indian and Chinese friends in the Medical Image Processing Laboratory—Ramya Balachandran, Pallavi Choudhoury, Pallavi Joshi, Ning Xu—and thank them for our collaboration and memorable times together. I am also thankful to my Chinese friends from Yale University, Yansong Zhao and Xin Xin, for their help and enthusiastic support as well as to my Czech friend Petr Jordan from Harvard University for our long-standing friendship. Many thanks to my American friends in Los Angeles, James Slattery and David Scott, for their continuous support since my arrival to the United States.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

3D	Three-dimensional
AR	auto-regressive
BEM	Boundary element method
BER	Basic electric rhythm
CMISS	Continuum Mechanics, Image analysis, Signal processing and System identification
cpm	Cycle per minute
CT	Computed tomography
DC	Direct current
ECA	Electrical control activity
ERA	Electrical response activity
EENG	Electroenterogram
EGG	Electrogastrogram
GEA	Gastric electrical activity
FEM	Finite element method
FFT	Fast Fourier transform
GI	Gastrointestinal
ICC	Interstitial cells of Cajal
IO	Input and output
IUPS	International Union for Physiological Sciences
GUI	Graphical user interface

MCG	Magnetocardiogram
MEG	Magnetoencephalogram
MENG	Magnetoenterogram
MGG	Magnetogastrogram
MIT	Massachusetts Institute of Technology
MRI	Magnetic resonance imaging
MVT	Mesenteric venous thrombosis
NDE	Nondestructive evaluation
OS	Operating system
NDT	Nondestructive testing
PSD	Power spectral density
SQUID	Superconducting quantum interference Device magnetometer
TFR	Time series representation
VUIRB	Vanderbilt University Institutional Review Board

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

Gastrointestinal Electrical Control Activity

The study of the electrical control activity (ECA) in the stomach is of great interest in the medical field due to the proven possibility to detect various pathological conditions of this organ from the analysis of gastric slow wave signals. Although over 60 million Americans are affected by digestive diseases [20], the electrophysiological mechanisms associated with pathological conditions in the gastrointestinal system are only beginning to be understood. One such condition is gastroparesis, which is characterized by abnormally slow gastric emptying rates, dyspepsia, nausea, discomfort and intermittent vomiting [34, 47]. In the past, numerous investigations were carried out to detect phenomena associated with diseases such as gastroparesis and ischemia using the electrogastrogram (EGG) and magnetogastrogram (MGG) [16]. Given that the mortality rate of patients suffering from acute mesenteric ischemia—including mesenteric venous thrombosis (MVT)—at least 50%, the importance of such studies cannot be understated. The EGG and MGG rely on the fact that in the stomach and small bowel, electric currents due to the presence of transmembrane potentials exhibit propagation patterns, which are also present throughout the rest of the gastrointestinal (GI) tract. This phenomenon consists of depolarization and repolarization waves that advance along the corpus of the stomach as a result of the presence of coupled cells in the smooth muscle of the tract. In the case of the stomach, the structure of gastric tissues allows the organ to behave like an electric syncytium, which gives rise to electric currents. The peristaltic waves that arise in the muscle tissue layers of the GI tract are produced by the interaction of the enteric nervous system, the interstitial cells of Cajal (ICC) and the smooth muscle cells [49]. The ICC are located within the Auerbachs plexus, a group of ganglion cells between the circular and longitudinal layers of the *muscularis externa* in the digestive tract. Approximately every 20 seconds, the ICC spontaneously depolarize to initiate the slow wave. Once initiated by the ICC, electrical activity is conducted from fiber to fiber throughout the body of the stomach; muscular contractions are associated with the electrical response activity (ERA), which is triggered by the depolarization of the cell membrane above threshold [23]. Because of volume-conductor smoothing of intracellular potentials and sequential phase shifts between adjacent cells, the waveform of the cutaneous waveform resembles a sinusoid.

The transmembrane potentials produced by coupled cells in smooth muscle give rise to dipole moments that vary in frequency and phase according to their anatomical position and configuration, reflecting the fact that gastric slow wave frequencies are location dependent along the GI tract [53, 60]. These slow wave variations in their own turn produce magnetic fields, which can be recorded experimentally. Studies have shown that analyzing these magnetic fields can be useful in detecting pathological states of the stomach in human patients. Moreover, the experimental and computational task of identifying sources of electric current in the GI tract is important because it can allow such conditions to be identified noninvasively.

The gastric slow wave, first recorded in 1921 by Alvarez [1], is a biophysical phenomenon that originates in the corpus of the stomach and propagates across this organ in the form of a de-polarization / re-polarization wave front which moves in the direction of the pylorus at a frequency of 3-4 cycles per minute (cpm) in human subjects. Electric propagation is made possible by the smooth gastric muscle, which consists of several layers of coupled cells that allow polarization waves to advance across the stomach until they reach the pylorus. At that point, propagation resets to the corpus region before a new cycle begins. A recently developed theoretical model of the human stomach whose realism and usefulness in studying the GEA has been confirmed by computational and simulation studies [25-27] suggests that the GEA can be modeled by a ring of current dipoles of approximately equal magnitude and positioned homogeneously at equal distances from each other in the cylindrical body of the stomach. It has been shown both theoretically and experimentally [3-6] that this dipole ring configuration can be well approximated by a single dipole located at the center of the circumferential ring and whose magnitude is equal to the sum of magnitudes of the smaller dipoles.

Modeling the GEA with accuracy and realism is an interesting theoretical and computational problem. Although a number of models have been proposed to simulate this phenomenon since the time when it was first detected by Alvarez [1], it can be difficult to ascertain which of them captures the characteristics of gastric activity most vividly. For example, coupled relaxation oscillator models are also suitable for studying the time evolution of smooth muscle propagation, while dipole models are more appropriate for the investigation of electrical source uncoupling. In the past, modeling the GEA using electric current dipoles led to two important predictions, namely that the natural gastric slow wave frequency gradient may be detected noninvasively using magnetometers and also that gastric wave propagation in gastric musculature results in propagating patterns of magnetic fields. In a previous study, GI models were also compared statistically to allow a direct assessment of how many GI electric sources are required to model the GEA realistically [30].

Throughout a typical propagation cycle, the amplitude of the electric potential recorded from a healthy human subject at the surface of the abdomen increases by more than one order of magnitude, while the propagation velocity is thought by some researchers [38] to reach around 6.8 mm/s in the pylorus from the approximate value of 3 mm/s [21] at the beginning of the cycle. The gastric slow wave frequency, on the other hand, varies from 3 cpm in the stomach [55] to approximately 12 cpm in the intestine and 8 cpm in the terminal ileum [18, 46]. Lately, consistent efforts in the direction of modeling the complex phenomenon of gastric electrical activity have led to the development of various analytical and numerical electric wave propagation models that can reproduce EGG recordings [38-43]. It has thus been found that the frequency of slow wave cycles varies greatly not only depending upon the portion of the GI tract analyzed, but also upon the health state of the organ and other tissue characteristics.

SQUID Magnetometry

Because abnormalities in the propagation of GI bioelectric currents are associated with disease states, a great deal of investigative effort has been invested in the direction of detecting them noninvasively. Experimentally, bioelectric currents have been detected and investigated using the EGG; in recent years, however, a number of inherent difficulties associated with this method—such as the dependence of electric recordings upon tissue conductivity, which attenuates the EGG signal—have suggested the use of the magnetogastrogram (MGG) instead, because the magnetic—and not electric—field of the stomach is measured with the latter procedure. This is advantageous because magnetic fields are dependent on tissue permeability, which is nearly equal to that of free space. Investigating the GEA from magnetic field recordings is also encouraging due to the finding that, although magnetic field strengths decrease rapidly with distance from their sources, they do reveal the characteristics of these sources in a more accurate manner [3, 4].

Due to the fact that gastric biomagnetic fields are very weak—having strengths of the order of 10^{-12} T—a highly sensitive measurement apparatus is required for experimental data collection. Such an instrument is the Superconducting QUantum Interference Device (SQUID) biomagnetometer, which remains to this date the most sensitive device for the detection and measurement of extremely low-magnitude magnetic fields. In particular, the Tristan 637i biomagnetometer, produced by Tristan Technologies (San Diego, CA), is a highly sensitive, multi-channel SQUID magnetometer system that can detect the bioelectromagnetic activity in the human stomach and intestine. Among others, the SQUID magnetometer has been shown to possess the ability of detecting intestinal ischemia, a disease that is difficult to diagnose and usually fatal. SQUID sensors are able to detect electrical signals resulting from the basic electrical rhythm (BER) of the intestine, whose frequency changes under ischemia. Thus, the potential for the use of SQUIDs in clinical diagnosis is significant.

Clinically, the ability to detect gastric activity noninvasively via a test that can be administered rapidly and efficiently would revolutionize functional gastroenterology. For the first time, reasons exist to hope that the MGG offers the possibility to fulfill this goal. In several important studies conducted in the Biomagnetism Laboratory at Vanderbilt University, recordings from normal and divided pig stomachs were used to perform a comparative study of how propagation is affected in the latter case by deliberate damage of the smooth muscle tissue. It was then shown that SQUID magnetometers are capable of detecting abnormal characteristics of the GEA and of obtaining new and meaningful information that may help in the early diagnosis of gastroparesis. Because SQUID measurements are performed noninvasively, abnormal changes in the electrical signal of the

GI tract can be detected without surgical intervention and the patient diagnosis process can be carried out rapidly, avoiding extensive patient preparation [58].

Motivation and Purpose

For a number of years, SQUID magnetometers have been used very successfully to detect and study GI diseases. Nevertheless, although the hardware components of this instrument are available commercially as an integrated unit, the same cannot be said about the potentially numerous software tools that are often required for processing and analyzing SQUID data. Such tools are nevertheless indispensable to a large variety of clinicians, engineers and biophysicists whose work involves the analysis of GI biomagnetic signals. In spite of this acute need, an integrative, user-friendly software tool for acquiring and analyzing GI SQUID data has not yet appeared in the scientific literature. In order to fulfill the need for such a computer-based system, a modular, integrative and multipurpose software program has been designed, implemented and tested on the SQUID 637i biomagnetometer produced by Tristan Technologies (San Diego, CA). The purpose of this thesis is to describe this integrative software and to demonstrate its suitability and usefulness to clinical research. Moreover, one goal of the work described in what follows is to set the standard for SQUID data processing software and to significantly improve the state of the art in the important and critical research area of biomagnetic instrumentation and analysis.

In Chapter 2, *Instrumentation and Experimental Setup*, the functioning principles of the SQUID magnetometer are presented. In particular, the hardware specifications of the Tristan 637i biomagnetometer are described in detail, especially in those respects that are most relevant to the design and capabilities of the modular software program described in this thesis. Finally, a description of a typical MGG experiment is given in order to familiarize the reader with the manner in which biomagnetic signals are recorded from the human stomach and intestine.

The three major functional units of the data acquisition and analysis software are described in Chapter 3, *Data Acquisition, Processing and Analysis*. Each of these units—the data acquisition module, data processing and analysis module, and CMISS visualization module—are described in detail with respect to both their functionality and their design and implementation. Finally, Chapter 4, *Conclusions and Future Research*, outlines the most important uses of our software and draws attention to several directions of future progress and possible improvement.

CHAPTER II

INSTRUMENTATION AND EXPERIMENTAL SETUP

The Superconducting QUantum Interference Device Biomagnetometer

SQUIDs have a long history of use for nondestructive evaluation (NDE) and testing (NDT), in which technical methods are used to examine materials or components in a manner that does not impair their future usefulness and serviceability [61]. In biomagnetism, these instruments were first used in 1970 by Edgar Edelsack, Jim Zimmerman and David Cohen, who recorded the magnetocardiogram (MCG) from the heart of a subject at the National Magnet Laboratory at the Massachusetts Institute of Technology (MIT). Later on, Cohen used SQUIDs to develop a novel technique called the magnetoencephalogram (MEG); his accomplishments prompted numerous other investigators to record signals due to a variety of other sources in the human body [14-16]. The use of SQUIDs for the investigation of GI magnetic fields was pioneered in the Living State Physics Laboratories at Vanderbilt University in the 1990s. There, it was first shown that the biomagnetic fields of the stomach and intestine could be used to detect pathological conditions of these organs noninvasively. Since then, a great deal of research has been conducted in order to quantify and understand changes in the BER that are associated with abnormal conditions of the human GI tract.

A SQUID—or Superconducting QUantum Interference Device—is a magnetometer that contains one or more Josephson junctions. A Josephson junction is a weak insulative layer between two superconductors that is able to support a supercurrent below a critical temperature I_c [61]. In a direct current (DC) SQUID—which contains two Josephson

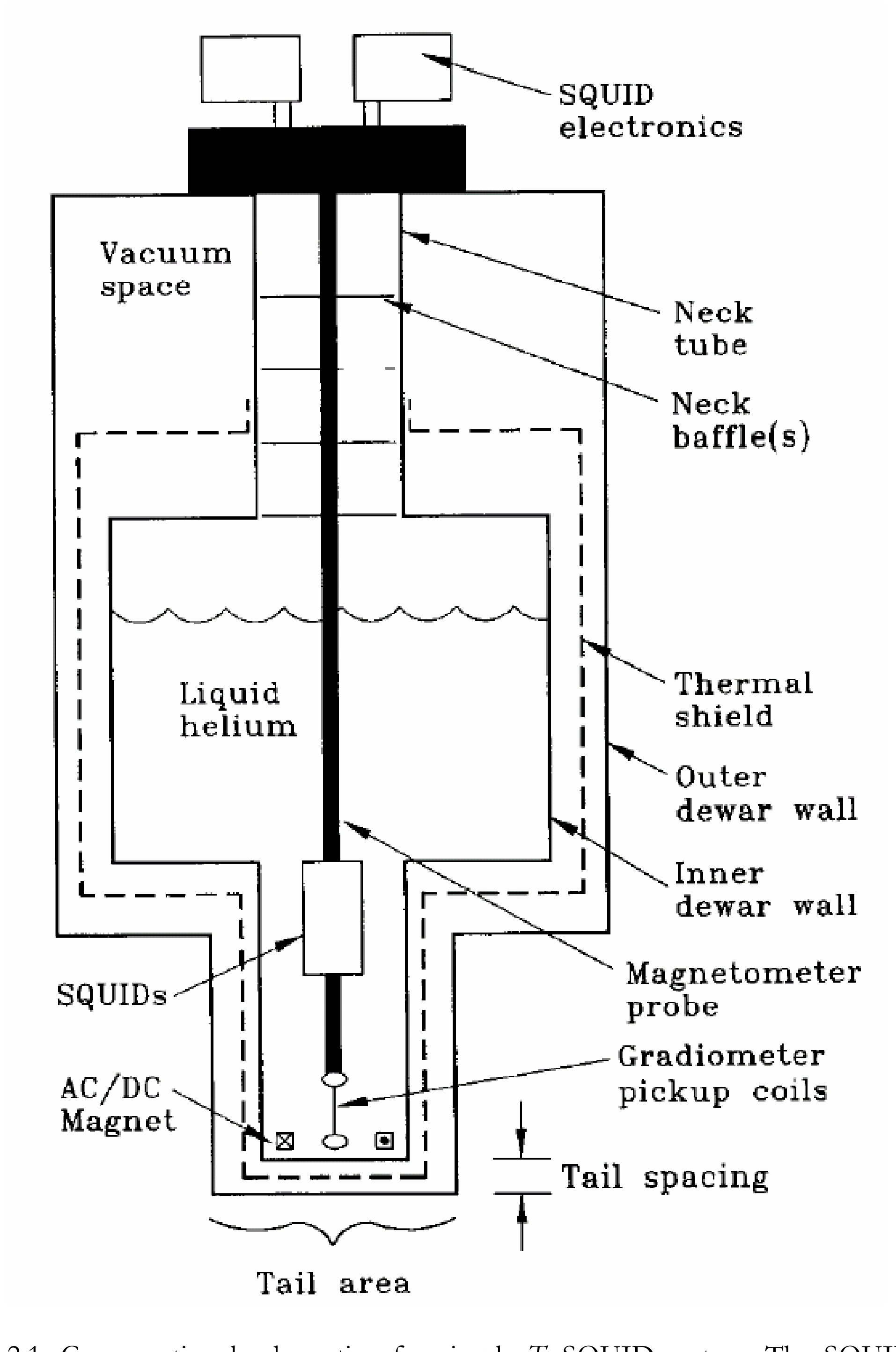


Figure 2.1. Cross-sectional schematic of a simple T_c SQUID system. The SQUID and input coils are located within a superconducting niobium cylinder. A flux-locked loop is housed in a SQUID electronics box positioned above the dewar and linked to the SQUID via a magnetometer probe. When the dewar is filled with liquid helium, the SQUID, niobium canister and pickup coils are cooled to a temperature below the boiling point of He (4.2 K). From [32], reproduced with permission.

junctions—a superconducting loop with a pair of Josephson junctions is applied in order to measure the loop impedance. Due to several special properties of the Josephson junction, the impedance is a periodic function of the magnetic flux threading the SQUID. Using this setup, a modulation signal applied to the bias current and a lock-in detector are employed to measure the impedance and to linearize the voltage-to-flux relationship. Thus the SQUID functions as a flux-to-voltage converter of extremely high sensitivity. In biomagnetism, where the strength of the measured field is very weak and a substantial amount of information is present at low frequencies, the SQUID magnetometer is excellently suited as a measurement tool.

A typical modern SQUID is located inside a small, cylindrical, superconducting magnetic shield within a liquid helium dewar. A number of superconducting pickup coils located at the bottom of the dewar are configured as gradiometers and are able to detect the difference in one component of the field between two points. When cooled below a critical temperature T_c, the superconductor exhibits zero resistance to DC currents up to a critical current value I_c. Due to a number of properties of superconductors, the superconducting ring of a SQUID can only enclose an amount of magnetic flux Φ that is an integer multiple of the flux quantum $\Phi_0 = 2.07 \times 10^{-15}$ Wb [32]. The superconducting components of the SQUID are immersed in a helium reservoir, which is thermally insulated by a vacuum jacket containing radiation shields.

Instead of exposing the actual SQUID to the external magnetic field to be measured, most modern instruments use a multi-turn pickup coil that is linked inductively to the SQUID. Thus, the pickup coils have the capability to sense the ambient field while the input coils of the SQUID are shielded from the external field within a superconducting niobium canister.

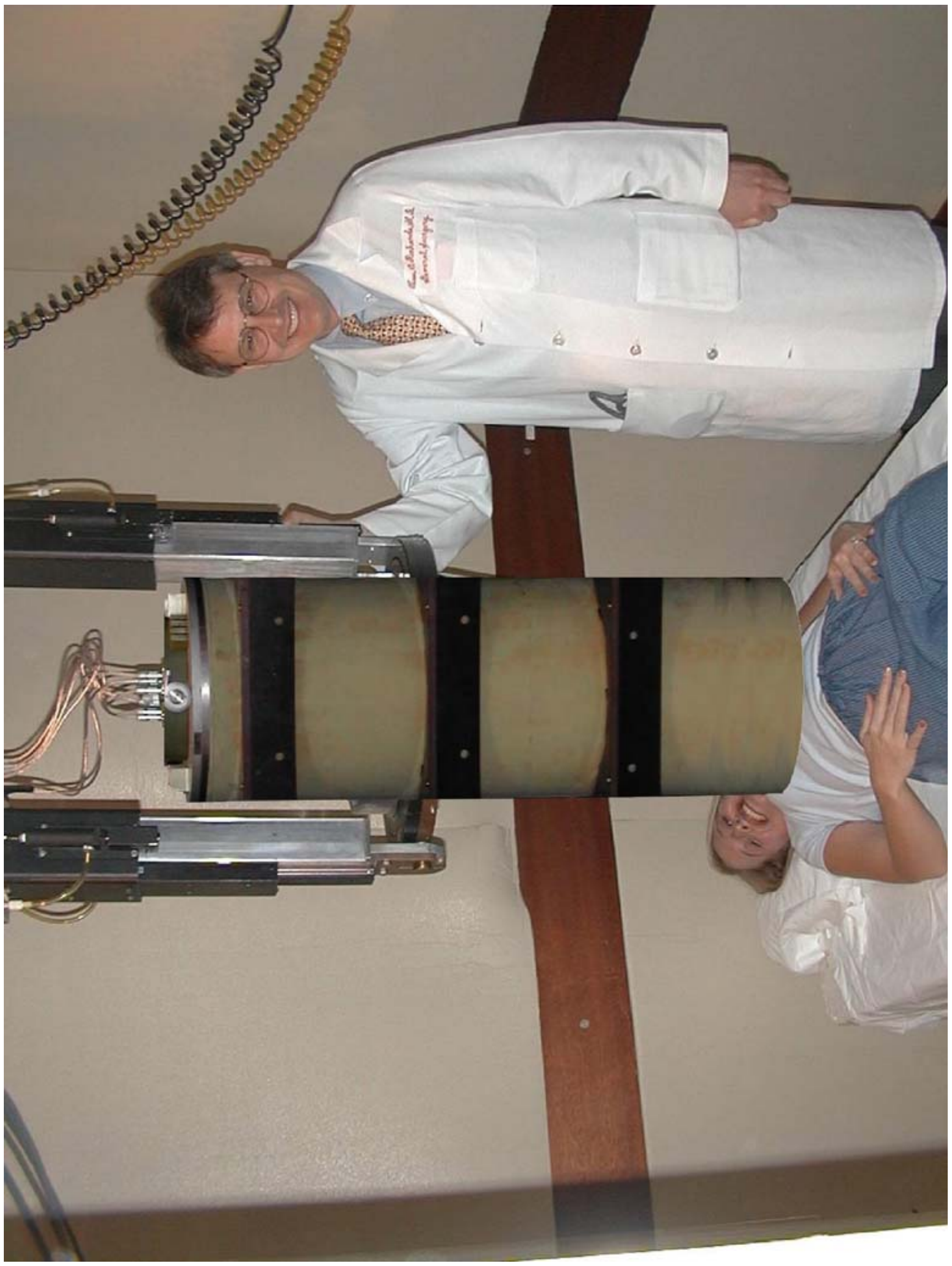


Figure 2.2. The Tristan 637i SQUID magnetometer in the Biomagnetism Laboratory at Vanderbilt University. The magnetometer is located in a magnetically shielded room where measurements are taken noninvasively

Hardware Specifications of the Tristan 637i SQUID Magnetometer

The Tristan 637i magnetometer has 5 pickup coils that record the magnetic field gradient in the $\hat{\mathbf{x}}$ direction and 5 coils for the $\hat{\mathbf{y}}$ direction. In addition, 19 axial coils are available for measuring gradients in the $\hat{\mathbf{z}}$ direction. The 29 superconducting pickup coils of the 637i SQUID magnetometer coils are distributed in the shape of two concentric, coplanar hexagons, as in Figure 2.3, and can record the quantity $\Delta\mathbf{B}/\Delta z$. Ten other such input channels are also positioned in the same plane as the other channels, five of each being used to record $\Delta\mathbf{B}/\Delta x$ and $\Delta\mathbf{B}/\Delta y$, respectively.

The hardware associated with the Tristan magnetometer includes a cryogenic flux-to-voltage converter, superconducting pickup coils serving as a direct current flux transformer, connecting wires and a cryostat. Pickup coils are arranged in a gradiometer configuration with one pickup coil located a certain distance above a lower coil. These coils are wound in opposite directions so that the flux in the upper coil is subtracted from the flux in the lower coil, thus canceling steady ambient magnetic fields such as that of the Earth.

During a SQUID experiment, an analog-to-digital device is employed to convert the magnetic data recorded by these channels into binary file format. The data are stored as a matrix with 29 columns corresponding to each input channel of the SQUID magnetometer and a number of rows equal to the number of discrete sampling times at which data values are recorded. Before the data are stored in digital format, a number of analog filters and down-sampling operations are also carried out. In our case, the effective sampling frequency used for storing SQUID data is user selectable and with a commonly used value of 300 Hz, although the actual sampling frequency of the SQUID magnetometer is higher.

Two types of coil assemblies are present among the 29 signal channels: an axial assembly and a three-channel vector assembly. A number of 14 axial assemblies and 5 vector assemblies are present in the sensor; signal channels are first-order gradiometers with a baseline of 5 cm while the normal channel is an axial gradiometer. In the vector assemblies, two transverse channels are planar gradiometers.

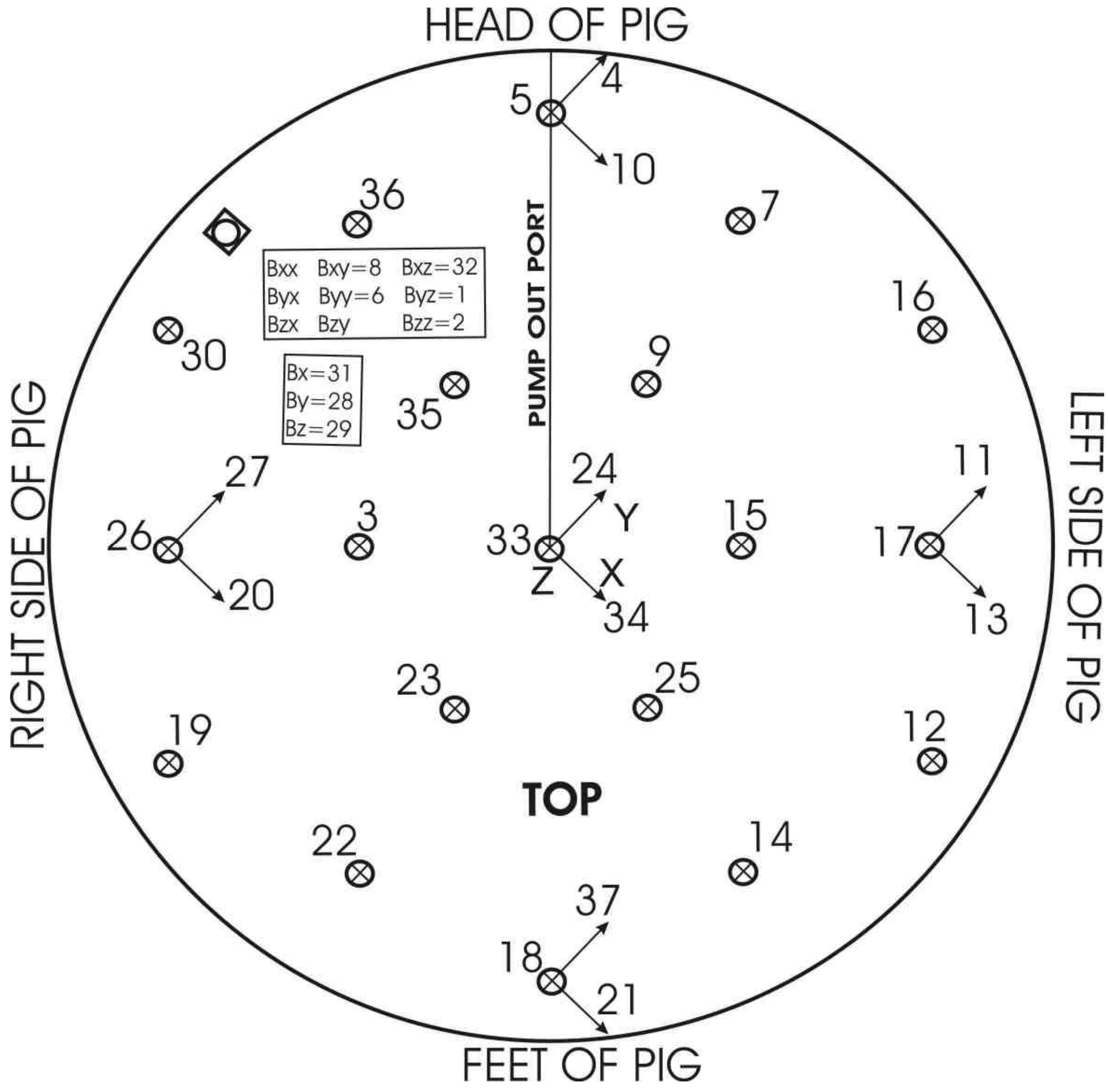


Figure 2.3. Spatial distribution of the co-planar input coils for the 637i SQUID biomagnetometer produced by Tristan Technologies (San Diego, CA). The magnetometer possesses a total of 29 pickup coils that are distributed within two concentric hexagons around a 5-cm baseline. 19 of these coils record changes in the magnetic field in the $\hat{\mathbf{z}}$ direction (i.e. $\Delta\mathbf{B}_x/\Delta z$), while 5 coils record $\Delta\mathbf{B}_y/\Delta x$ and 5 record $\Delta\mathbf{B}_x/\Delta x$. The x and y channels are located at the center and extremities of the baseline and their positions are marked in the figure by short arrows inscriptioned with x or y, as appropriate. In addition to these 29 channels recording absolute changes in B_x, B_y and B_z, a number of noise channels and respiration channels for monitoring and recording breathholds are also provided.

The gradiometers of the SQUID are designed in such fashion as to reduce environmental magnetic noise, which is due to distant sources and couples equal but opposite signals into the two gradiometer coils. Because biological signals are located close to the input array, their sources will couple strongly into the closest gradiometer coil, while the distant coil causes little signal loss. For this reason, the gradiometer design of the SQUID impacts only signals when dealing with distant environmental sources of noise. For sources in the stomach and intestine, the signal coils are therefore essentially magnetometers.

The 29 z channels of the SQUID measure only the **B** field component that is normal to the outer surface of the sensor and, because of the experimental setup for GI recordings, normal to the body of the subject. All three components of the field are measured at only five locations, thus allowing one to determine the direction of the field and its magnitude at those positions.

In addition to the 29 channels that record magnetic field data, a number of 8 magnetic sensing channels are provided in a tensor array to monitor environmental magnetic noise. This array is located high enough above the input coils to avoid detection of the signal of interest but close enough to the subject to record environmental noise that is representative of the noise in the signal coils. The total of 37 SQUID channels are mounted in a 27 liter liquid helium dewar, although the average helium consumption is less than 6 liters per day, allowing the system to remain usable for several days in a row after the cooling of the SQUID.

MGG Experiment Description

MGG and MENG experiments are conducted at Vanderbilt University in the magnetically shielded room of the Biomagnetism Laboratory. Informed consent is obtained from every volunteer and the Vanderbilt University Institutional Review Board (VUIRB) must approve each study. In a typical experiment of this type, the subject is positioned

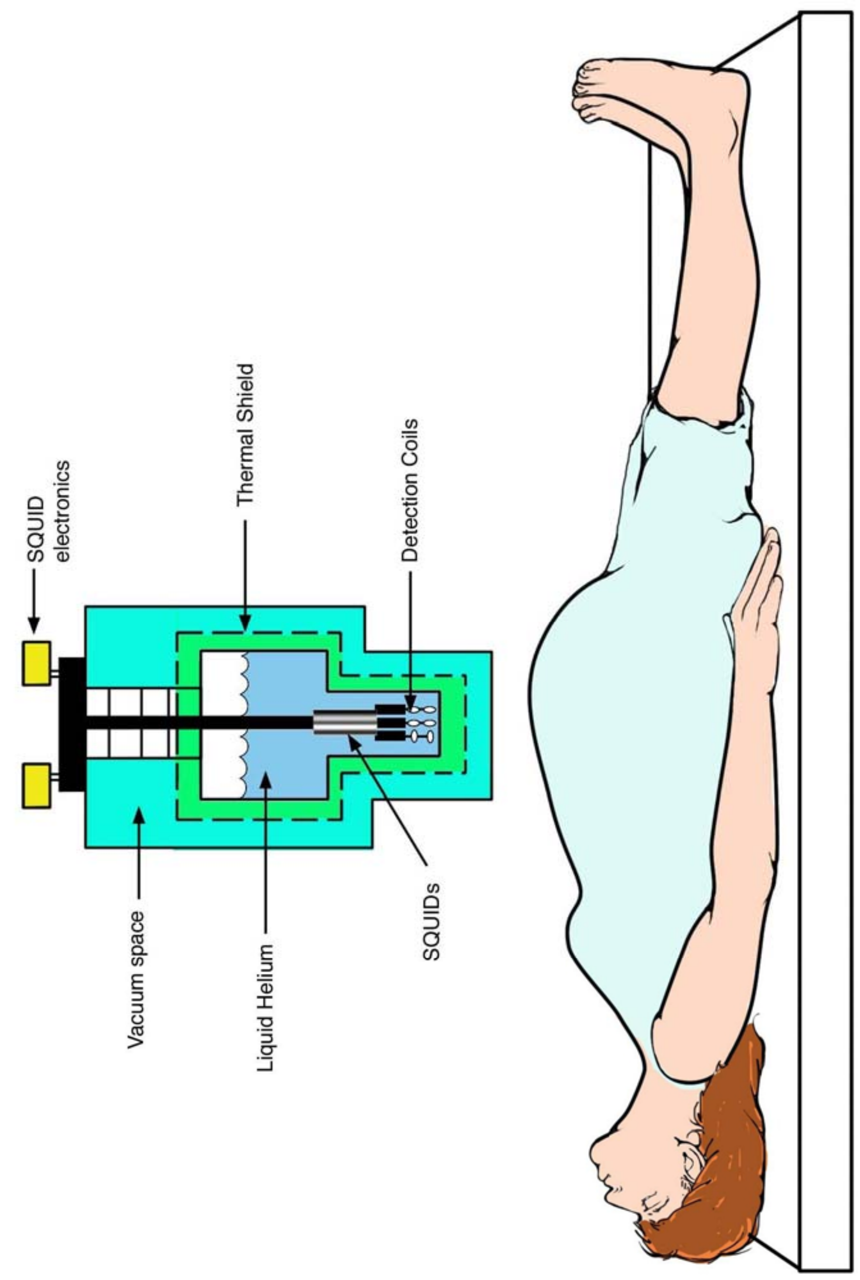


Figure 2.4. Experimental setup for data acquisition in MGG. The subject is positioned horizontally directly below the pickup coils of the SQUID gradiometer.

horizontally below the magnetometer coils and is then asked to suspend respiration and lie quietly for a period of at least one minute during each recording. A drawing of the experimental setup in such cases is provided in Figure 2.4. In some cases, fasting is required of the subject prior to the experiment. Very often, biomagnetic data are acquired both

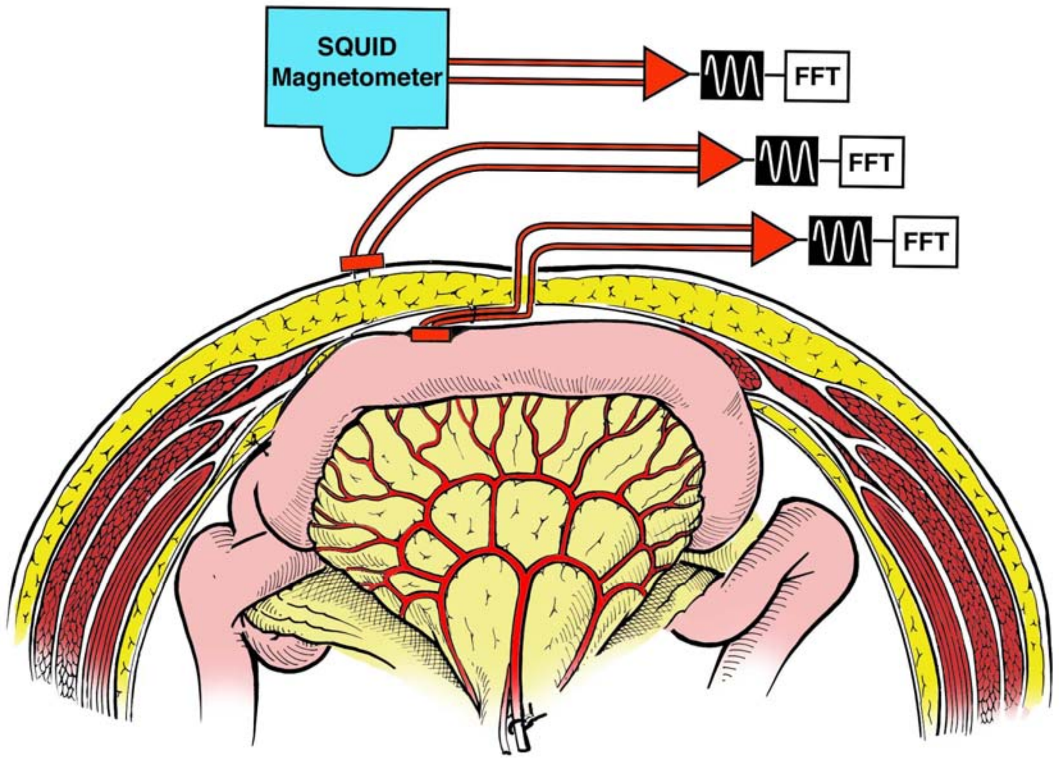


Figure 2.5. Experimental setup of simultaneous MENG/EENG data acquisition. The intestine of most common mammals is located under several layers of skin, fat and other tissues, which all attenuate the EENG signal. For this reason, the EENG is invasive because it requires the placement of electrodes directly on the external gastric wall surface, while the MGG is non-invasive because magnetic fields are recorded in the latter procedure. Because the permittivity of biological tissues is nearly equal to that of free space, the MGG signal is attenuated far less than the EGG signal.

before and after the consumption of a meal to study the electrical activity of the stomach in both of these two conditions. For each recording, the magnetometer is oriented such that the coils measuring the x and y components of the signal tangential to the body surface are oriented in the saggital and horizontal planes. On the other hand, the coil measuring the z component normal to the body surface is oriented in the frontal plane [6].

A variety of clinical investigations and data measurements can be done with the Tristan SQUID magnetometer, as shown in Figure 2.5 and Figure 2.6. Very often, measurements for both the MGG/MENG and the EGG/EENG are taken at the same time

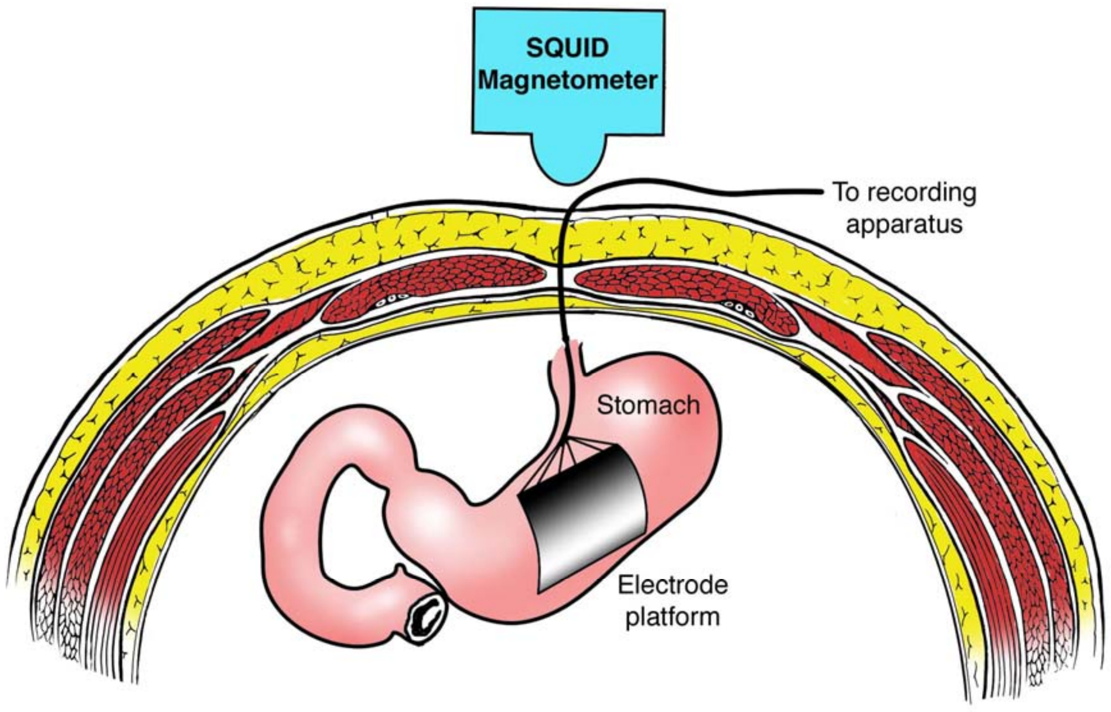


Figure 2.6. Experimental setup of simultaneous MGG/EGG data acquisition. The EGG electrode platform is placed invasively at the external surface of the stomach while the SQUID magnetometer is positioned above the body.

not only for practical reasons but also to have both signals available for investigation, e.g. in a correlation study of the MGG and EGG signals. Because the EGG/EENG signal is attenuated dramatically from the internal source to the external environment due to the presence of skin and fat, EGG electrodes are often placed invasively directly on the outer surface of the stomach. The MGG signal, however, depends on the permittivity of tissues, which is nearly equal to μ_0. Thus MGG signals can be acquired noninvasively.

CHAPTER III

DATA ACQUISITION, PROCESSING AND ANALYSIS

A variety of hardware and software tools are required to record, decimate, filter and store SQUID data. Likewise, the analysis of GI electromagnetic phenomena from these recordings is often complex and time-consuming. Quite often, a large number of computational and visual tools are required to study and understand the gastric slow wave from magnetic field recordings. Because of this, any acquisition and analysis software for SQUID data must be flexible and adaptable; moreover, the nature and purpose of the analysis tools required for gastric slow wave analysis can differ quite significantly from study to study because their investigative goals can be very different. For all these reasons, a modular approach was adopted for the design of the acquisition and analysis program described in this thesis. Every module was designed separately with its own interface and visual features, while providing a unique control switchboard from which all modules can be easily accessed.

The first distinct unit of the program, the data acquisition module, is responsible for controlling data acquisition, analog filtering and storage in digital binary format. This module was designed in the LabView visual programming language and it possesses a friendly GUI that allows data to be acquired easily and efficiently. The second unit, the data processing and analysis module, gives access to a number of important computational and visual tools whose use is very common in the analysis of GI magnetic signals. Such tools include data display and filtering, time frequency representations (TFRs) of power series distributions

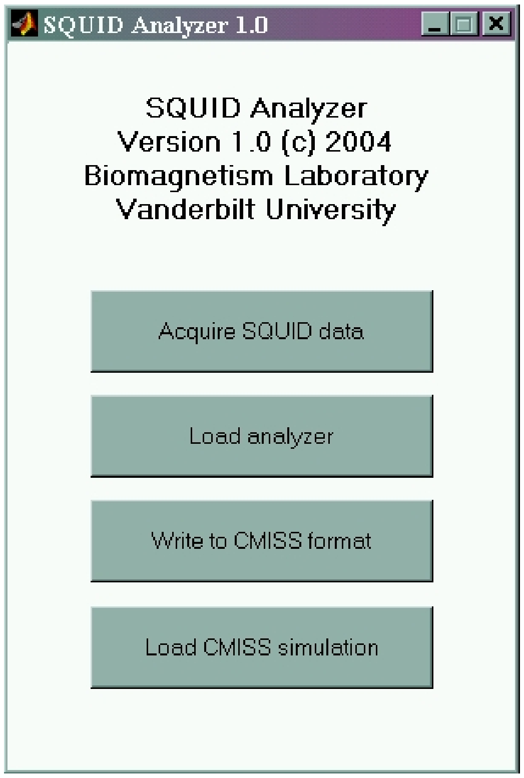


Figure 3.1. Software control switchboard for the acquisition and analysis program. The first button loads the LabView data acquisition module, while the second one is used for the data analysis and processing module. SQUID data can be converted into CMISS format using the third button, whereafter the CMISS visualization can be created.

(PSDs) in GI signals and SQUID data frequency maps. This module was implemented in the MatLab programming language. The third and last unit of the program contains a three-dimensional (3D) realistic model of the human torso and internal organs. Within this model, magnetic field components measured by the SQUID are visualized in 3D at the locations where they were recorded, with the reconstructed body of the stomach rendered directly below the magnetometer channel array. This last module was designed using the CMISS (Continuum Mechanics, Image analysis, Signal processing and System identification) simulation program of the Bioengineering Institute at the University of Auckland, New Zealand. Throughout the remainder of this chapter, each of these modules is described in detail and examples of data processed using the program are given.

Software Control Switchboard

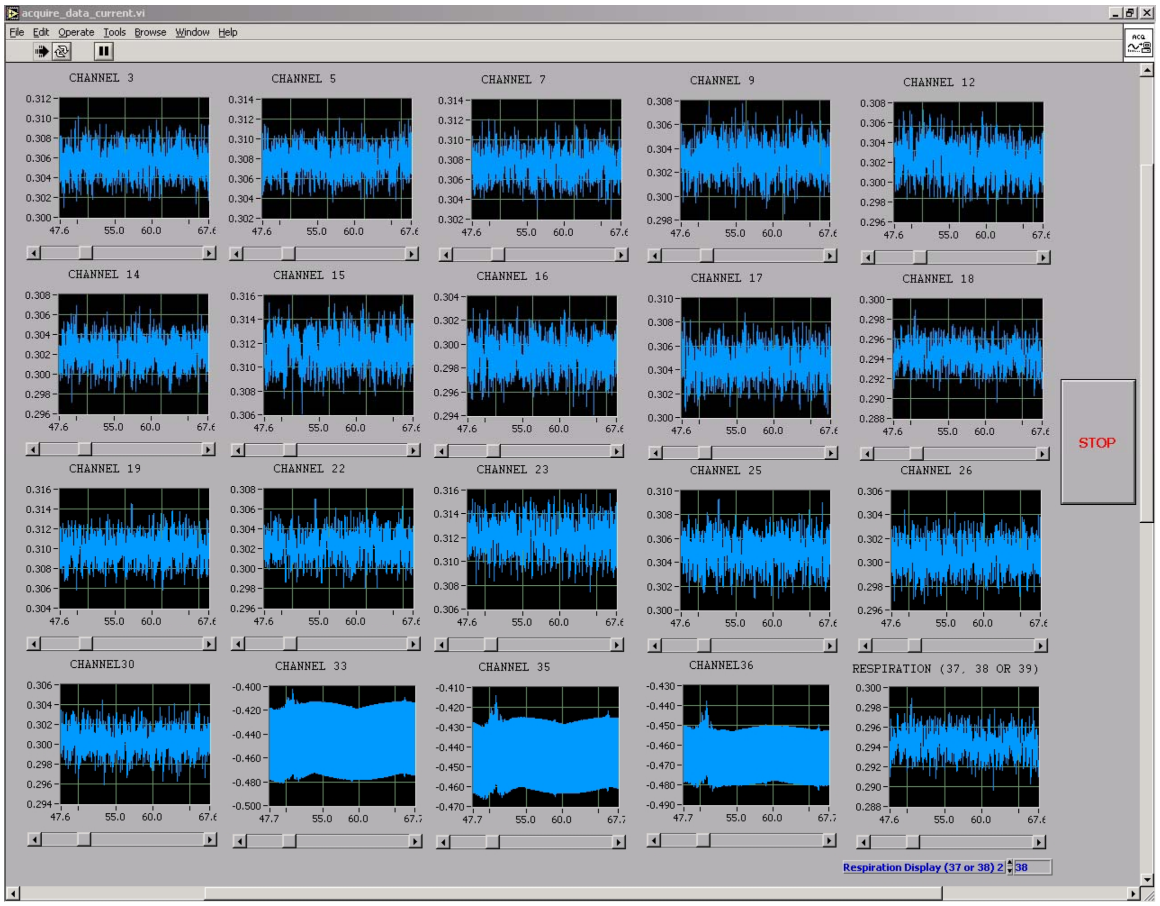
Each module of the acquisition and analysis program can be accessed from the software control switchboard. The graphical user interface (GUI) of the switchboard is presented in Figure 3.1. It includes a button for loading the acquisition program, one for loading the data analysis program and another one for generating a CMISS visualization. Because the format for CMISS input files is very different from that in which data are written to disk by the SQUID acquisition module, an intermediate step must be carried out in which an appropriate format conversion is made. Pressing the third button visible on the switchboard can perform this action.

Data Acquisition Module

The GUI used throughout the data acquisition process was developed using the LabView visual programming language; a sample screenshot is displayed in Figure 3.2. As data is being acquired with the SQUID magnetometer, the signal recorded in each channel is plotted as a function of time on the corresponding graph labeled with the number of the appropriate channel. This allows one to observe and study SQUID data in real time as they are being acquired.

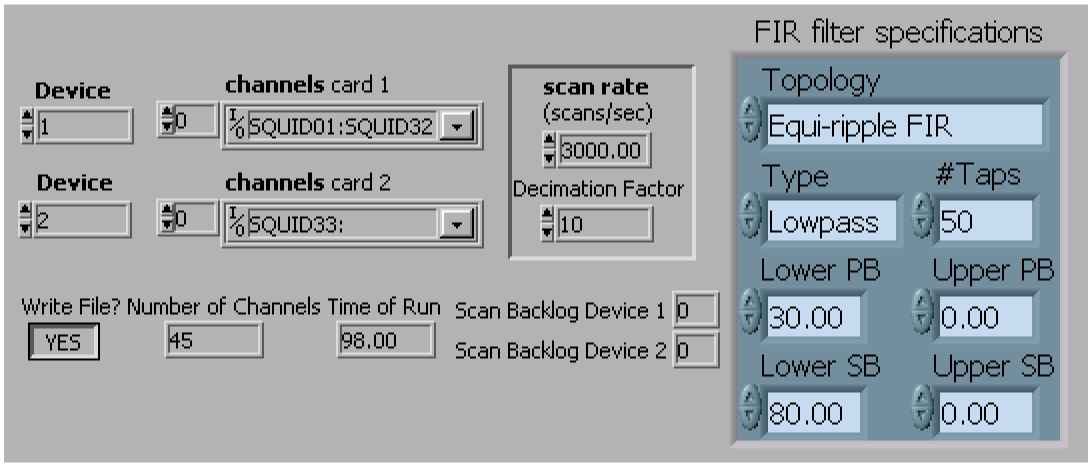
The SQUID magnetometer records analog signals at a DC frequency of 1 MHz. The original signal is then amplified and decimated to a frequency of 3 kHz post-acquisition. Lower frequency noise sources that are due to the motion artifacts of metal objects can be seen in the noise reference channels of the magnetometer and canceled adaptively or filtered digitally. Once acquired, the 3 kHz data are stored digitally in binary format for further processing.

A number of other acquisition controls and options are also made available in the



Respiration Display (37 or 38) 2 38

Figure 3.2. SQUID Analyzer GUI for the data acquisition module. The signal in every z channel is plotted against time using an adaptive technique where the interval range of the abscissa is varied with time according to the amplitude of the signal.



Device 1 0 channels card 1 SQUID01:SQUID32

Device 2 0 channels card 2 SQUID33:

scan rate (scans/sec) 3000.00

Decimation Factor 10

Write File? YES

Number of Channels 45

Time of Run 98.00

Scan Backlog Device 1 0

Scan Backlog Device 2 0

FIR filter specifications

Topology Equi-ripple FIR

Type Lowpass

#Taps 50

Lower PB 30.00

Upper PB 0.00

Lower SB 80.00

Upper SB 0.00

Figure 3.3. Control options for the data acquisition module of the SQUID Analyzer GUI. The data acquisition rate (scan rate), data decimation factor, FIR filter specifications and other options can be input here.

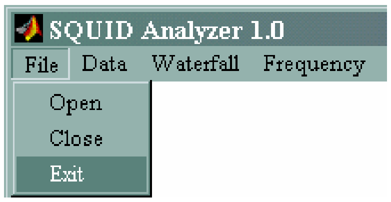


Figure 3.4. SQUID Analyzer 1.0 File menu. Software users can open SQUID magnetometer files for analysis, close files and exit the analyzer program.

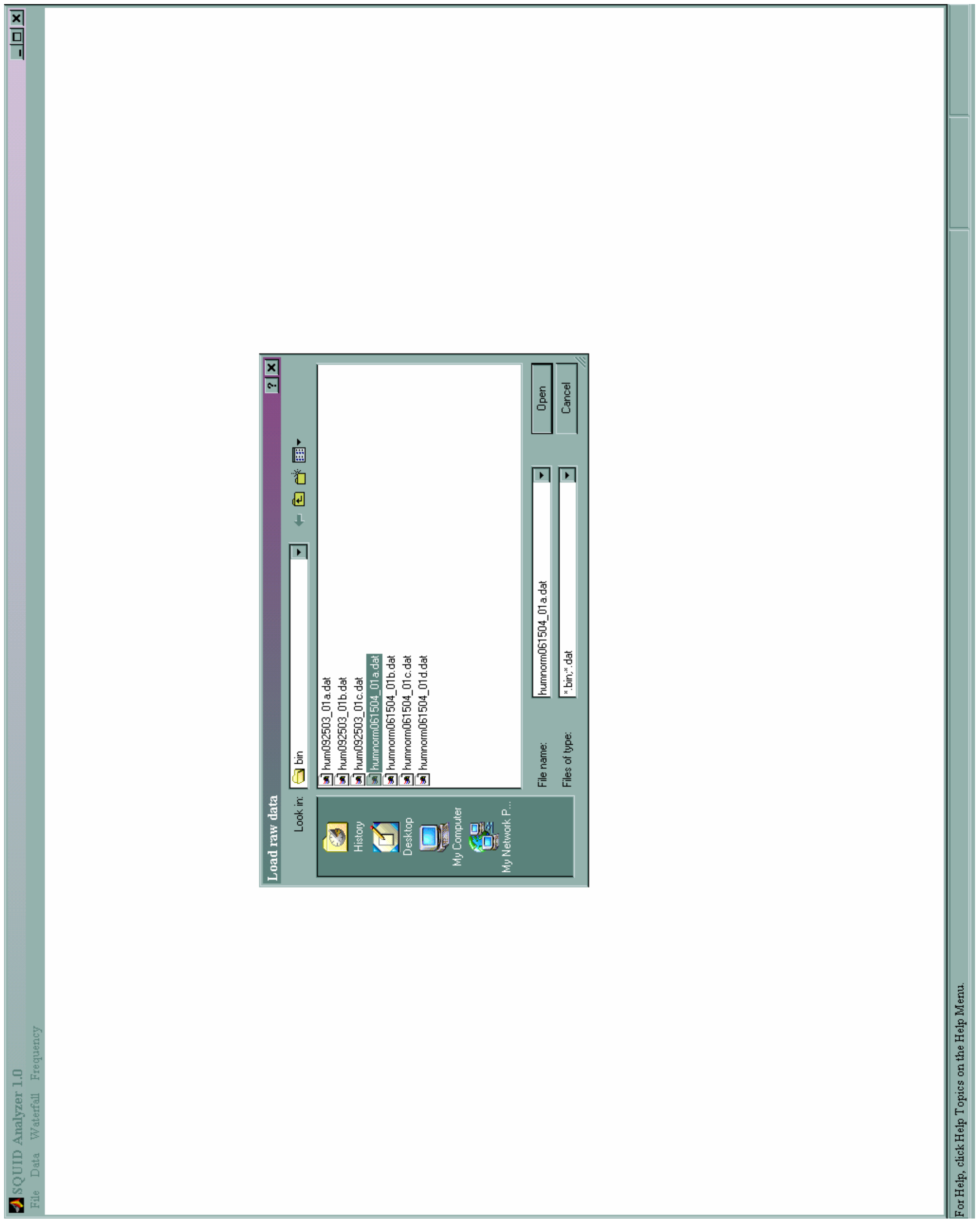
GUI of the data acquisition module, as shown in Figure 3.3. These include an input box for specifying the sampling rate to be used for writing data to the output file, as well as a control box for applying analog filtering to this data before writing it to the storage medium.

Data Processing and Analysis Module

A GUI was designed in MatLab for the data analysis module, dubbed SQUID Analyzer 1.0. The menu options of the GUI, `File`, `Data`, `Waterfal` and `Frequency`, allow the user to control the most important functions of the software. Throughout the remainder of this section, each of these functions is described in detail.

As shown in Figure 3.4, the first option from the `File` menu allows the user to open a raw data file created as a result of recording magnetic signals using the SQUID magnetometer. When accessing this menu feature, a dialog box opens as shown in Figure 3.5, where the user can browse for the desired file and then click the `OK` button.

Immediately after a file has been selected for analysis, the user is requested to specify the SQUID channels for which data should be loaded. As shown in Figure 3.6, the required information can be specified by the user in the input box that appears on the screen; if only the letters `x`, `y` or `z` are specified in the box, the program automatically loads all channels that record B_x, B_y or B_z data, respectively. However, the user also has the option of loading



Look in: bin

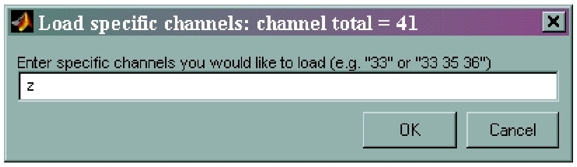
hum092503_01a.dat
hum092503_01b.dat
hum092503_01c.dat
humnorm061504_01a.dat
humnorm061504_01b.dat
humnorm061504_01c.dat
humnorm061504_01d.dat

File name: humnorm061504_01a.dat
Files of type: *.bin;*.dat

Figure 3.5. Sample input window for browsing and selecting SQUID magnetometer input files.

only a smaller subset of channel data by specifying the numbers of the channels to be loaded as a space-delimited string.

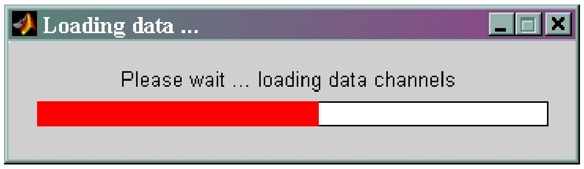
Throughout the process of loading and decimating the data, a wait bar is displayed as in Figure 3.7 while the requested loading operation takes place. The SQUID Analyzer allows


Load specific channels: channel total = 41

Enter specific channels you would like to load (e.g. "33" or "33 35 36")

z

Figure 3.6. Input box for specifying SQUID channel data to be loaded into memory. The user can specify 'x', 'y', 'z' (to load all channels containing B_x, B_y, or B_z data, respectively) or the numbers of individual channels separated by spaces can be input separately.


Loading data ...

Please wait ... loading data channels

Figure 3.7. Wait bar displayed throughout the process of loading SQUID magnetometer channel data.

one to specify the decimation factor to be applied when loading the data; this can be an integer greater than or equal to 1; the MatLab `decimate` function is employed to decimate the data.

SQUID Channel Data Visualization

As shown in Figure 3.8 and Figure 3.9, several display options are made available from the `Data` menu. From the `Plot raw data` menu item, the user is given access to the option of plotting raw data using one of the options `Numerical channel order` or `Spatial channel arrangement`. These two display modes are also available from the second menu item, namely `Plot filtered data`. The third menu item, `Adjust data time range`, allows the user to focus only on a particular time segment within a data set, which is very useful because breathhold time intervals within a large data set can be processed and analyzed more adequately.

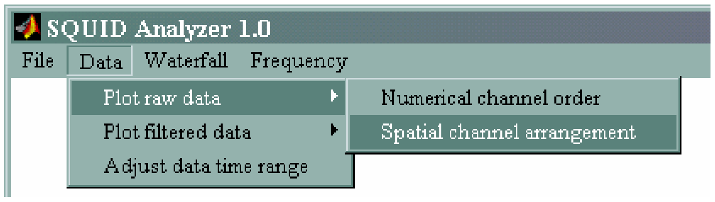


Figure 3.8. SQUID Analyzer 1.0 Data menu (I). The software user can plot raw or filtered data in either numerical channel order or using the SQUID channel arrangement. The time range of the data being plotted and analyzed can be changed using the third menu item.

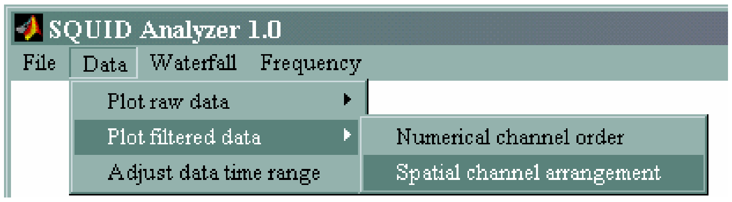
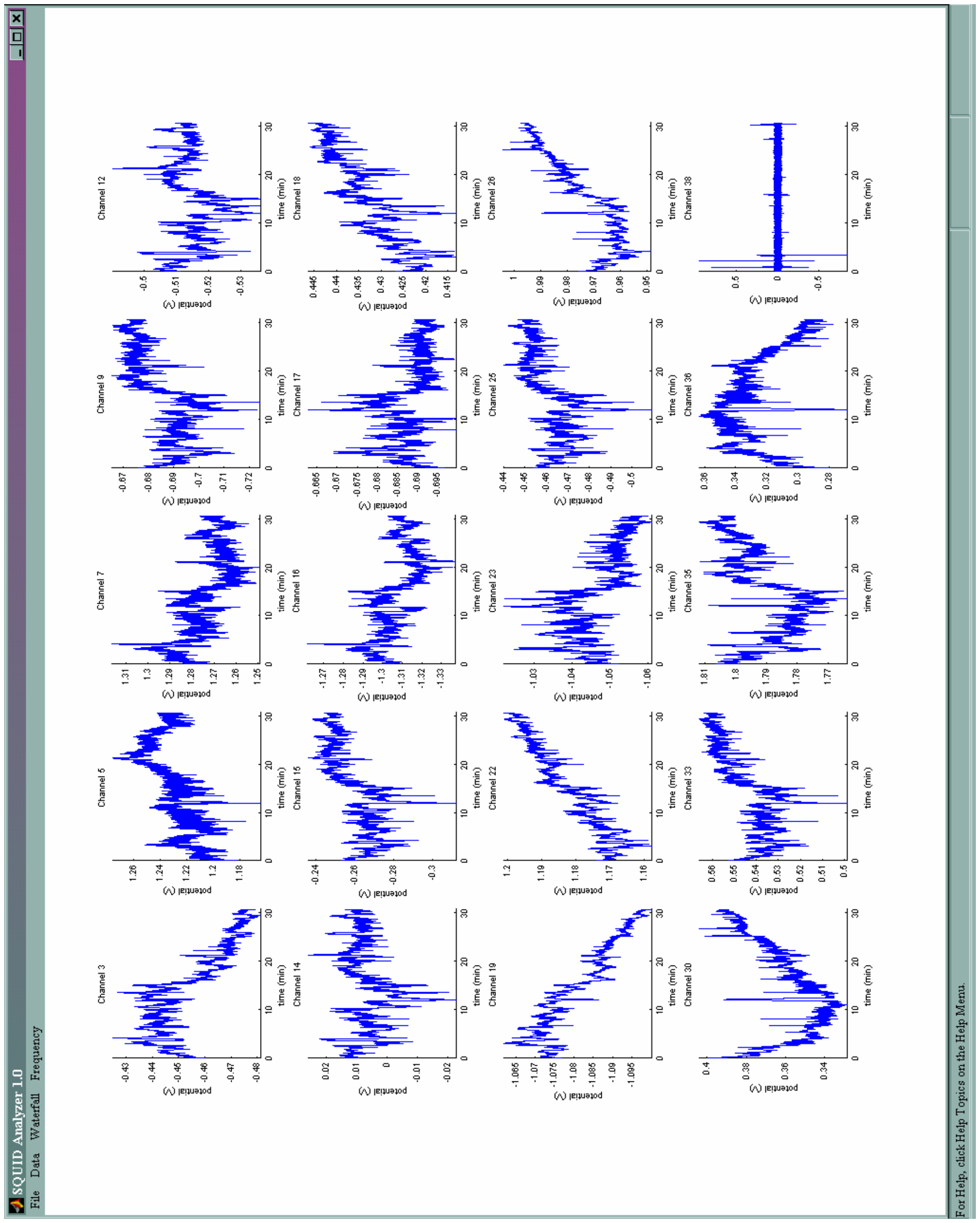


Figure 3.9. SQUID Analyzer 1.0 Data menu (II). The options of the second menu item are shown.

Two different display options are available for visualizing data acquired from the SQUID magnetometer. In the first of these, as presented in Figure 3.10, plots of electric potential or magnetic field data are shown in the numerical order of each channel. This view has two primary advantages. First of all, it provides an ordered display of data that is made available in a conventional manner. Secondly, it allows one to easily identify and analyze the data in a channel of interest simply from the numerical ordering of the channels.

The second data view provided attempts to reproduce the spatial arrangement of each input coil of the gradiometer (see Figure 3.11). This view has the advantage of providing a way to associate each data channel with a particular anatomic region of interest. It also allows one to identify areas where particular frequencies in the electrical activity signal are present. This can be useful because, very often, a specific frequency value can allow one to associate the region where that frequency component is strongest with a particular organ,



For Help, click Help Topics on the Help Menu.

Figure 3.10. Sample raw SQUID z channel data displayed in the numerical order of magnetometer z channels.

e.g. with a particular segment of the intestine. Moreover, it may be possible to observe and analyze the frequency gradient of slow waves along the GI tract from the visual analysis of SQUID data being displayed in this fashion. The spatial arrangement mode is adapted to the configuration of the Tristan 637i biomagnetometer, but the program can be easily adapted

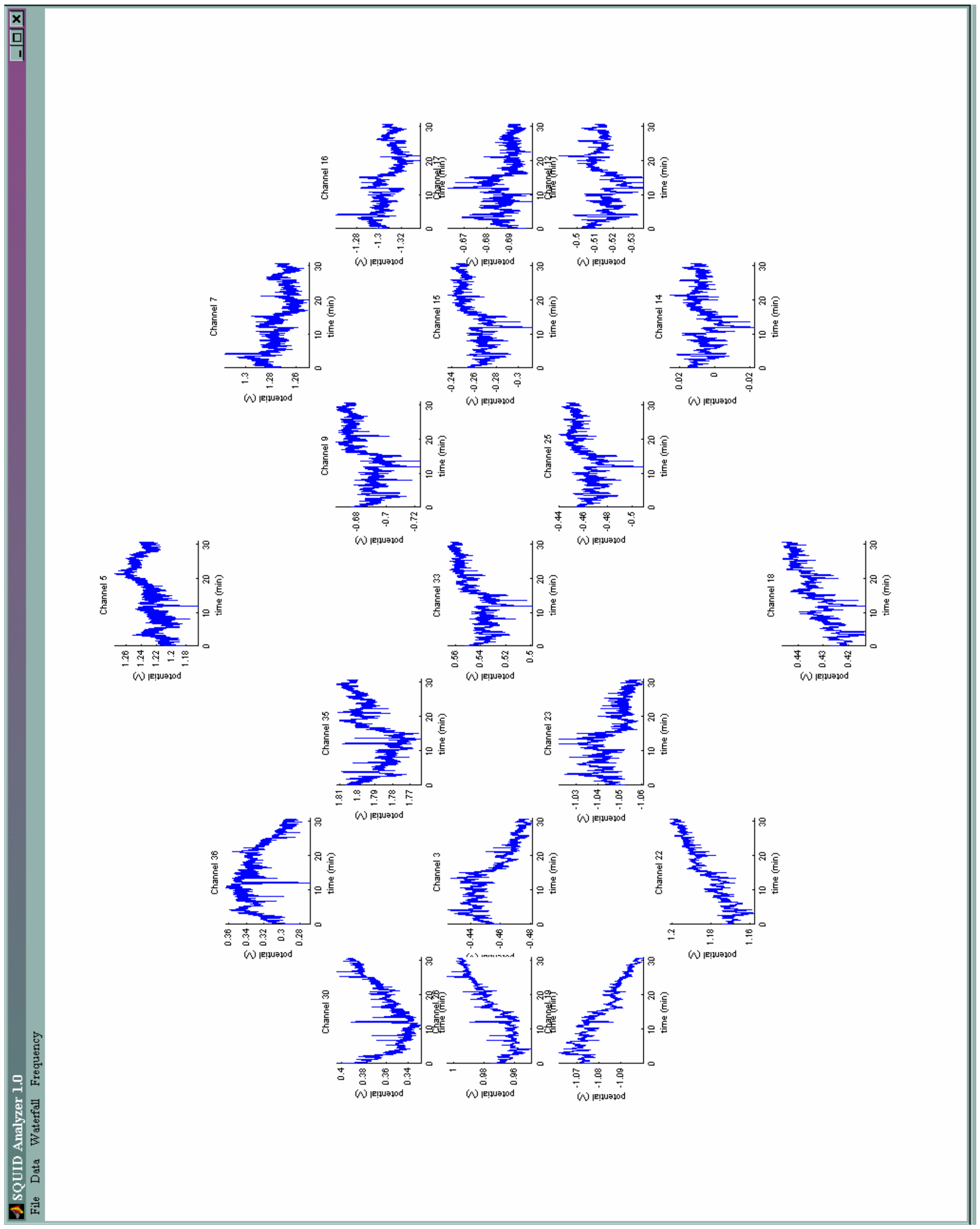
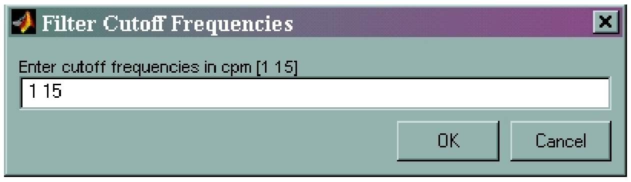


Figure 3.11. Sample raw SQUID z channel data displayed using the spatial arrangement of the magnetometer's z channels.

for different coil positions.

The second item in the `Data` menu gives the user access to the data filtering features of the program. The first data processing step involves detrending the data using the MatLab function `detrend`; this is done to eliminate undesired, short-lived trends in the


Filter Cutoff Frequencies

Enter cutoff frequencies in cpm [1 15]

1 15

Figure 3.12. Input box for specifying Butterworth filter cutoff parameters. The upper and lower limits of the filter separated by a space must be specified.

data. The `detrend` function removes the mean value or linear trend from a vector or matrix in view of FFT processing. It computes the least-squares fit of a straight line (or composite line for piecewise linear trends) to the data and subtracts the resulting function from the data.

Noise with a frequency lower than 1.5 cpm is then subtracted by the software using polynomial subtraction of variable order, in which a polynomial is fit to short-lived trends in the data and then subtracted from it to remove these trends. The degree of the polynomial is user-selectable. The third data processing step involves the design of a bandpass, second-order Butterworth filter using the MatLab function `butter`. The Butterworth filter is maximally flat in the pass band and monotonic overall, which reduces the effect of pass band ripples in the signal to a minimum. The `butter` function createz z-transform coefficients for a Butterworth filter up to a user-defined order. Butterworth filters sacrifice rolloff steepness for monotonicity in the pass- and stopbands. The function `butter` designs an order *n* lowpass digital Butterworth filter with a user-specified cutoff frequency. It returns the filter coefficients in two row vectors *b* and *a* of length *n*+1, with coefficients in descending powers of *z*:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b(1) + b(2)\frac{1}{z} + \ldots + b(n+1)\frac{1}{z^n}}{1 + a(2)\frac{1}{z} + \ldots + a(n+1)\frac{1}{z^n}}$$

These coefficients are used by the MatLab function `filtfilt` to filter the data in the forward and reverse directions for zero-phase filtering. This process is applied using the MatLab function `filtfilt`, which implements zero-phase digital filtering. It does this by processing the input data in both the forward and reverse directions [48]. After filtering in the forward direction, it reverses the filtered sequence and runs it back through the filter. The resulting sequence has precisely zero-phase distortion and double the filter order. Zero-phase distortion is necessary in our case because it avoids the distortion of magnetic field propagation characteristics. In addition to the forward-reverse filtering, it attempts to minimize startup transients by adjusting initial conditions to match the DC component of the signal and by prepending several filter lengths of a flipped, reflected copy of the input signal [37].

The default filter cutoff frequencies specified by the SQUID Analyzer software program are 1.8 and 18 cpm, corresponding to 0.03 and 0.3 Hz, respectively. While the GEA has a frequency of approximately 3 cpm, intestinal activity exhibits a frequency gradient along the GI tract and its frequencies range between approximately 4 and 18 cpm. Thus, the default parameters of the program are adequate for capturing information contained in the signal that refers to these two phenomena.

In clinical GI research, changes in the frequency components of the signal are of interest. To study those, power spectral density (PSD) estimates for each of several epochs in the signal must be computed to create a time-frequency representation (TFR) of the data [4, 5]. Two methods for carrying out this task are most common: the classical fast Fourier transform (FFT) and modern autoregressive (AR) spectral analysis. The Fourier power spectrum is very suitable for stationary signals that are not expected to change over the duration of the sample. Because FFT spectral resolution increases with the length of the

Filtering data at0.016667 and0.25 Hz
Please wait...filtering data

Figure 3.13. Wait bar displayed throughout the process of filtering SQUID magnetometer channel data in view of future display or computation of frequency waterfall plots.

sample—thus requiring long sample durations for better resolution—this method is not optimal for studying ischemic signals, which are prone to change over short periods of time. For this reason, AR spectral analysis is preferable for short time intervals (1 min).

Although efficient and accurate, AR spectral analysis also suffers from a number of drawbacks. For a sinusoidal wave, the area under the AR spectral curve depends on the signal power in a linear fashion, but peaks in the AR PSD are proportional to the square of the power [36]. Deviations of real signals from sinusoids also make the problem of estimating amplitudes more problematic [5]. For this reason, the square root of the AR PSD is often reported. Thus AR analysis is an inappropriate tool for determining the power at given frequencies, although it is certainly useful for locating dominant frequencies in the sample. For further information and comparison of AR and FFT analyses, we direct the reader to [56].

In light of these two alternatives, both AR analysis and the FFT are included in our software. The frequency spectra of the analyzed signal can be computed by the SQUID Analyzer using the fast Fourier transform (FFT) to obtain power spectral density (PSD) values for a specific frequency range. First, segments of data each containing one minute of SQUID recordings are created. Every segment is then windowed using a Hanning window in order to reduce FFT leakage error and zero-padded to increase frequency sampling in the

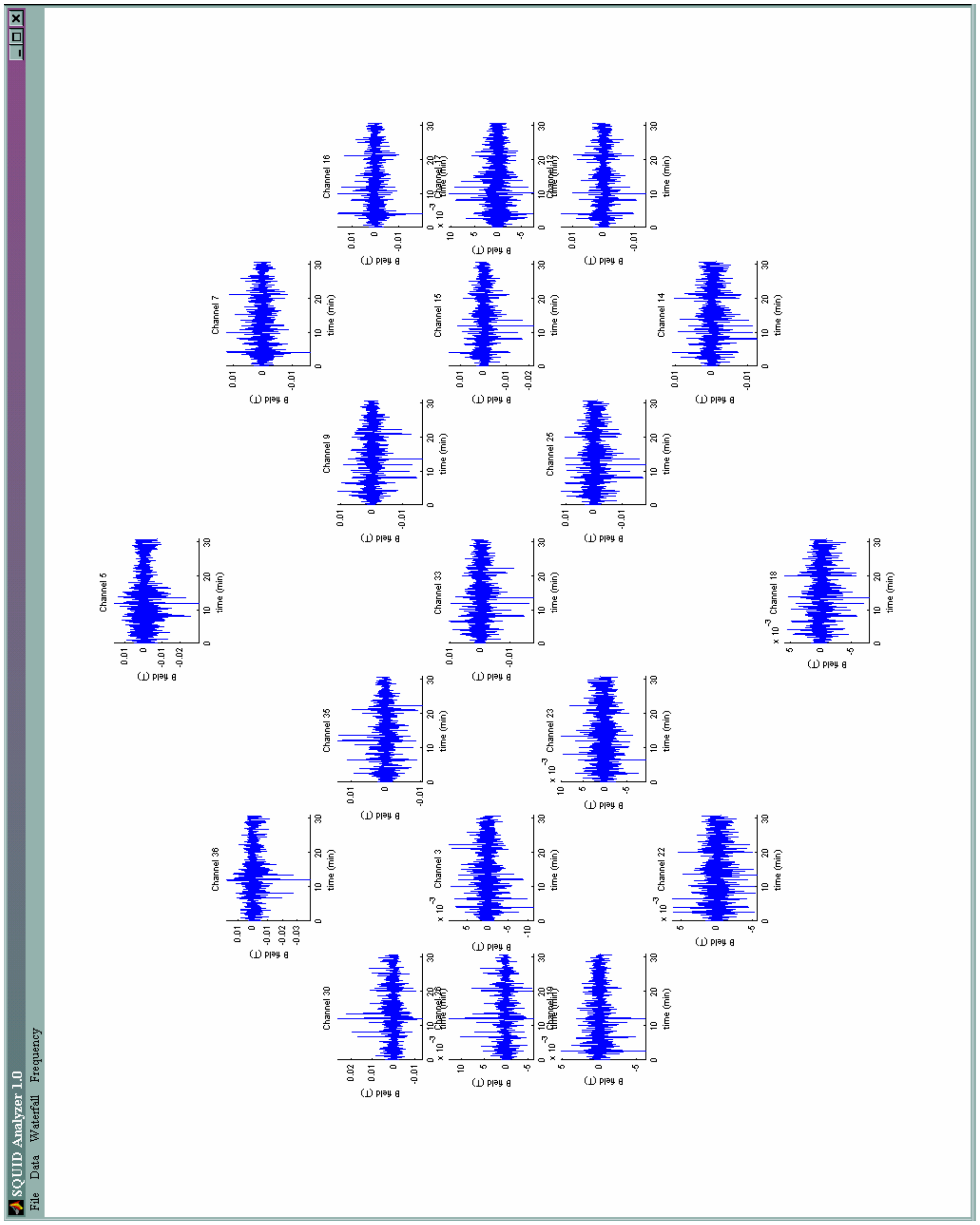
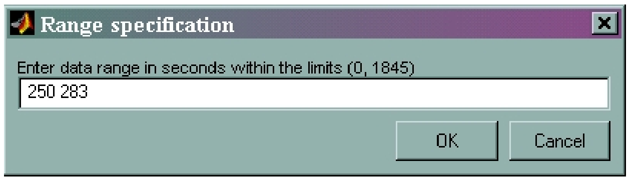


Figure 3.14. Sample filtered SQUID *z* channel data displayed using the spatial arrangement of the magnetometer's *z* channels. Over 30 minutes of data are displayed.

spectrum. The MatLab `fft` function is then employed to compute the FFT. The `fft` function computes a one-dimensional Fourier transform for vectors of length *n*:

$$X(k) = \sum_{j=1}^{n} x(j)\omega_n^{(j-1)(k-1)}$$



Enter data range in seconds within the limits (0, 1845)
250 283

Figure 3.15. Input box for specifying the time interval of SQUID data to be displayed and analyzed. The upper and lower limits of the interval separated by a space must be specified.

where

$$\omega_n = e^{-2\pi i/n}$$

is an *n*-th root of unity. The resulting frequency spectra are then assembled for all available data segments and displayed in the form of a three-dimensional (3D) frequency spectra waterfall plot, where each plot corresponds to one minute of SQUID data. As in the case of data loading, a wait bar is displayed to the screen while this process is applied to the loaded data (see Figure 3.13).

A sample input box where Butterworth filter parameters are specified is shown in Figure 3.12. As in the case of raw data display, a numerical order and a spatial arrangement display mode are made available for waterfall plots. When accessing either one of these, the input box prompting the user to specify the cutoff frequencies to be used as parameters to the filter is displayed.

A sample display of filtered data using the spatial channel arrangement display mode is shown in Figure 3.16. As one can see from this figure, data trends due to extraneous causes have been removed by detrending. Nevertheless, it can often be difficult to analyze and understand such data due to its high time variability and to the large time interval used for display on the ordinate axis. To correct this inconvenience and to provide a more reliable tool for analysis, the `Adjust data time range` option was made available from the

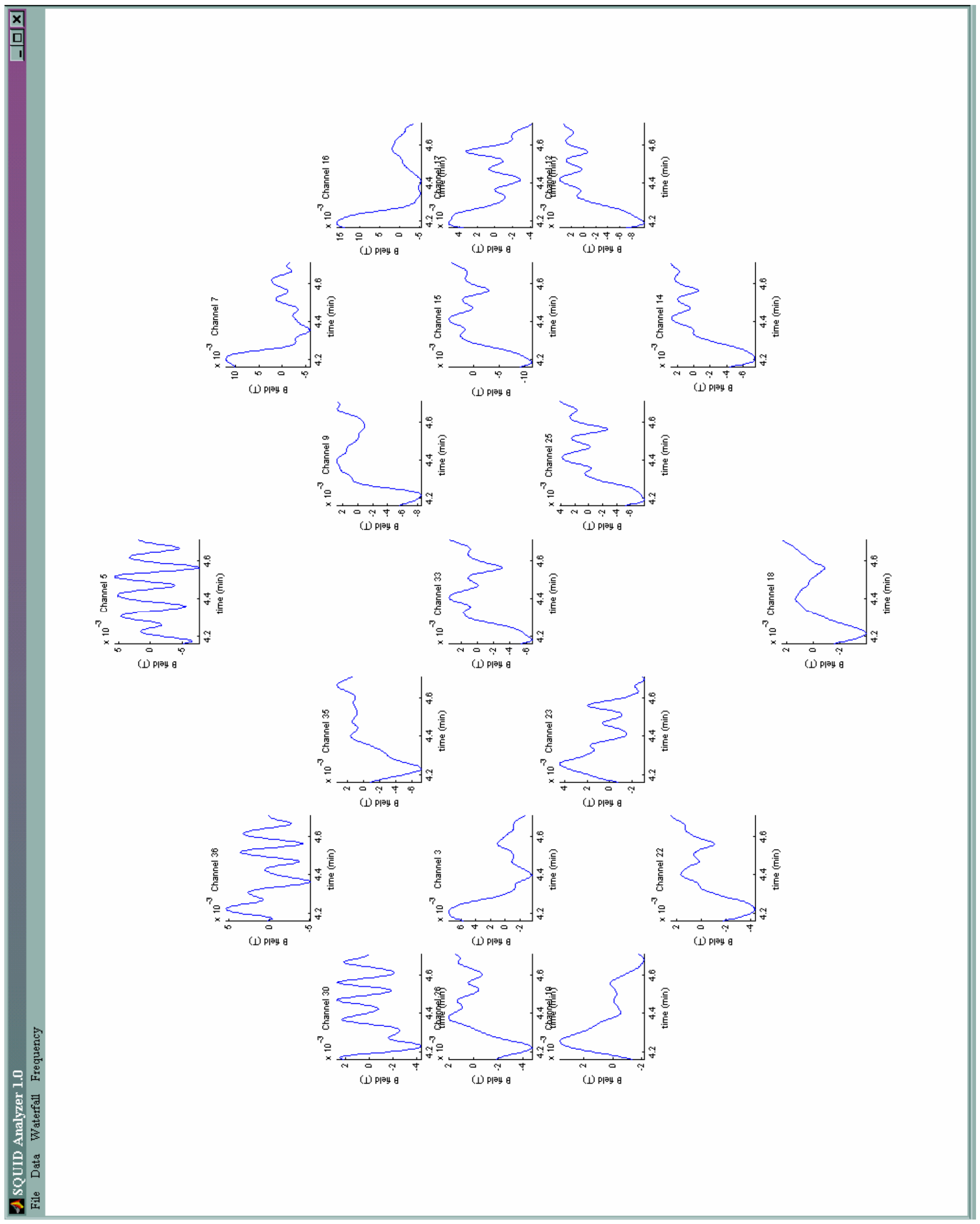


Figure 3.16. Sample filtered SQUID z channel data displayed using the spatial arrangement of the magnetometer's z channels as a result of adjusting the time interval for display as shown in the previous figure. As can be seen, the gastric and intestinal waveforms are easier to distinguish using these display options.

`Data` menu. When accessing this option, an input box is displayed to the screen as in Figure 3.15. The user can then input the time interval in seconds for which data is to be displayed. This feature is useful because, very often, for example, waveforms that are of biophysical interest are easier to view and analyze on a minute-by-minute basis as opposed to the alternative of viewing the entire dataset at once. An illustration of this is provided in Figure

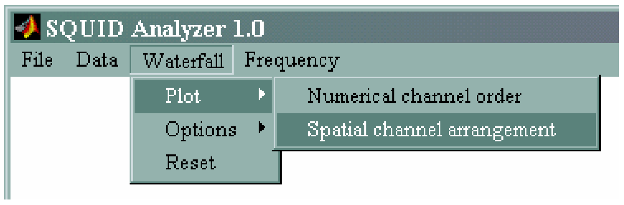


Figure 3.17. SQUID Analyzer 1.0 Waterfall menu (I). The options of the first menu item are shown.

3.16. As is apparent from this figure, all the waveforms are much easier to analyze and interpret when a smaller time interval is used for display.

EGG Waterfall Plot Generation

In addition to displaying raw and filtered SQUID channel data, the SQUID Analyzer is equipped with the ability to create waterfall plots of GI signal frequency spectra. As explained previously, these spectra are computed using the fast Fourier transform (FFT).

The display options available for frequency waterfall plots are similar to those for visualizing raw or filtered data. These options are all shown in Figures 3.17 and 3.18. Each plot can be viewed either in the numerical order of the magnetometer channels (Figure 3.20) or using the spatial arrangement setup in which every waterfall plot is at the approximate location of the corresponding SQUID input coil (Figure 3.23). In addition, a number of options are made available to manipulate the waterfall plots three-dimensionally. By pressing the ↑ and ↓ keys, the elevation of the camera viewpoint for each waterfall plot can be increased or decreased, respectively, by increments of 5^o in the range -90^o to 90^o. Similarly, pressing the ← or → keys changes the azimuth of the viewpoint by decrements or increments of 5^o, respectively, in the range 0^o to 360^o.

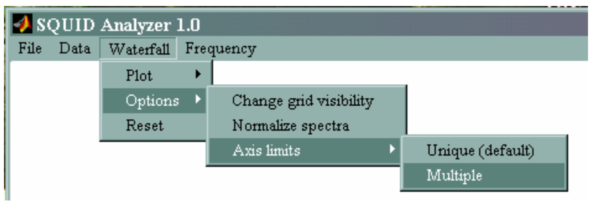
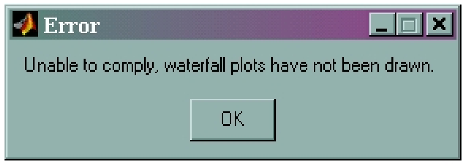


Figure 3.18. SQUID Analyzer 1.0 Waterfall menu (II). The options of the second menu item are shown. From this menu option, the user can change grid visibility, normalize waterfall spectra and set the axis limits for the waterfall plots to be displayed.


Error

Unable to comply, waterfall plots have not been drawn.

Figure 3.19. Sample error message displayed by the SQUID Analyzer when the user attempts to modify waterfall plot properties before these are drawn.

Other display options associated with the frequency waterfall plots include spectra normalization, changing grid visibility and the modification of the $\hat{\mathbf{z}}$ axis range displayed in each waterfall plot. When the spectra normalization option is used, every point in each 3D waterfall plot is normalized to the absolute maximum in that plot and this process is repeated for all the waterfall plots displayed. Changing grid visibility, on the other hand, removes or redraws the coordinate grid of each plot, as appropriate. The use of these two options is shown in Figure 3.20 and Figure 3.21.

Frequency Map Creation

The third important functionality of the SQUID Analyzer is to produce SQUID data frequency plots. The ultimate purpose of this utility is to associate the frequency spectrum data presented in waterfall plots with the spatial locations of the SQUID input coils. Spatial maps of frequency content allow one to localize electrical activities with different

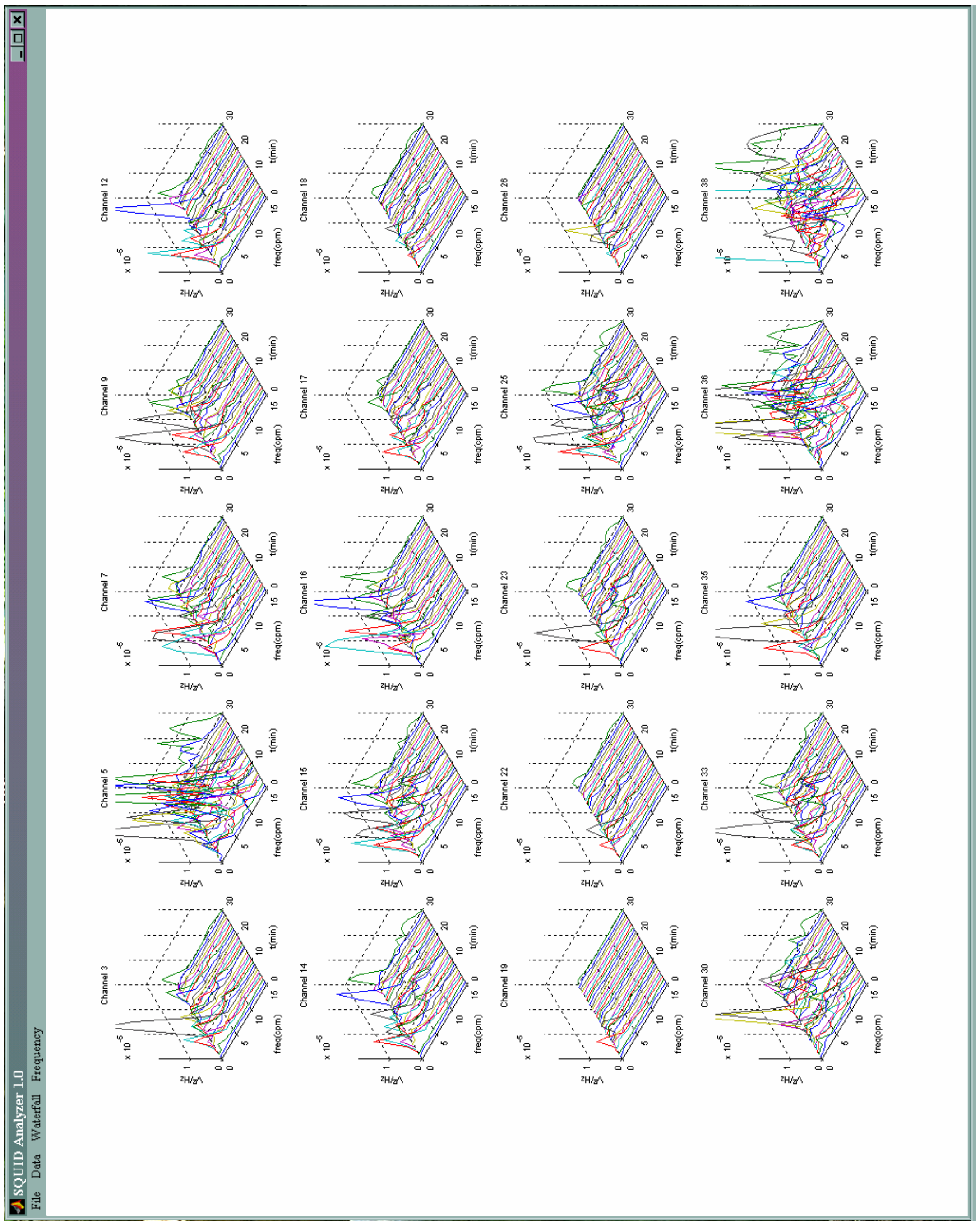


Figure 3.20. Sample waterfall plots generated using the numerical channel order display mode.

frequencies, such as the higher frequencies of small bowel electrical activity or brady- and tachygastrias.

The creation of frequency plots relies on the filtering and FFT frequency spectra computation processes that have already been described. To generate these plots, the

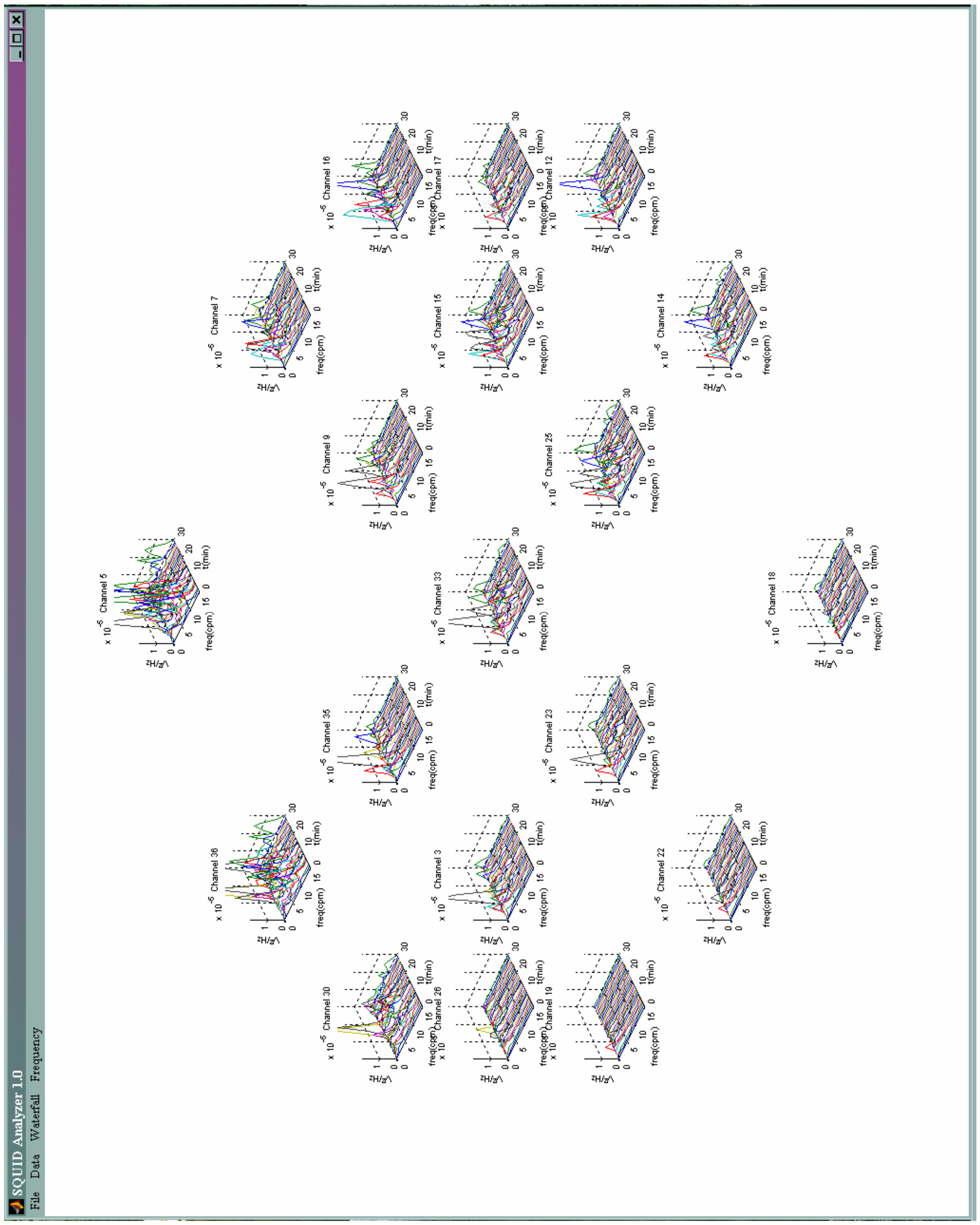


Figure 3.21. Sample waterfall plots generated using the spatial channel arrangement display mode.

frequency spectra are divided into segments of variable length spanning the frequency interval specified by the cutoff frequencies of the Butterworth filter. For instance, consider a spectrum generated as a result of applying the FFT to a dataset that has been filtered using a Butterworth filter with cutoff frequencies of 1 and 15 cpm. Such a spectrum can be divided,

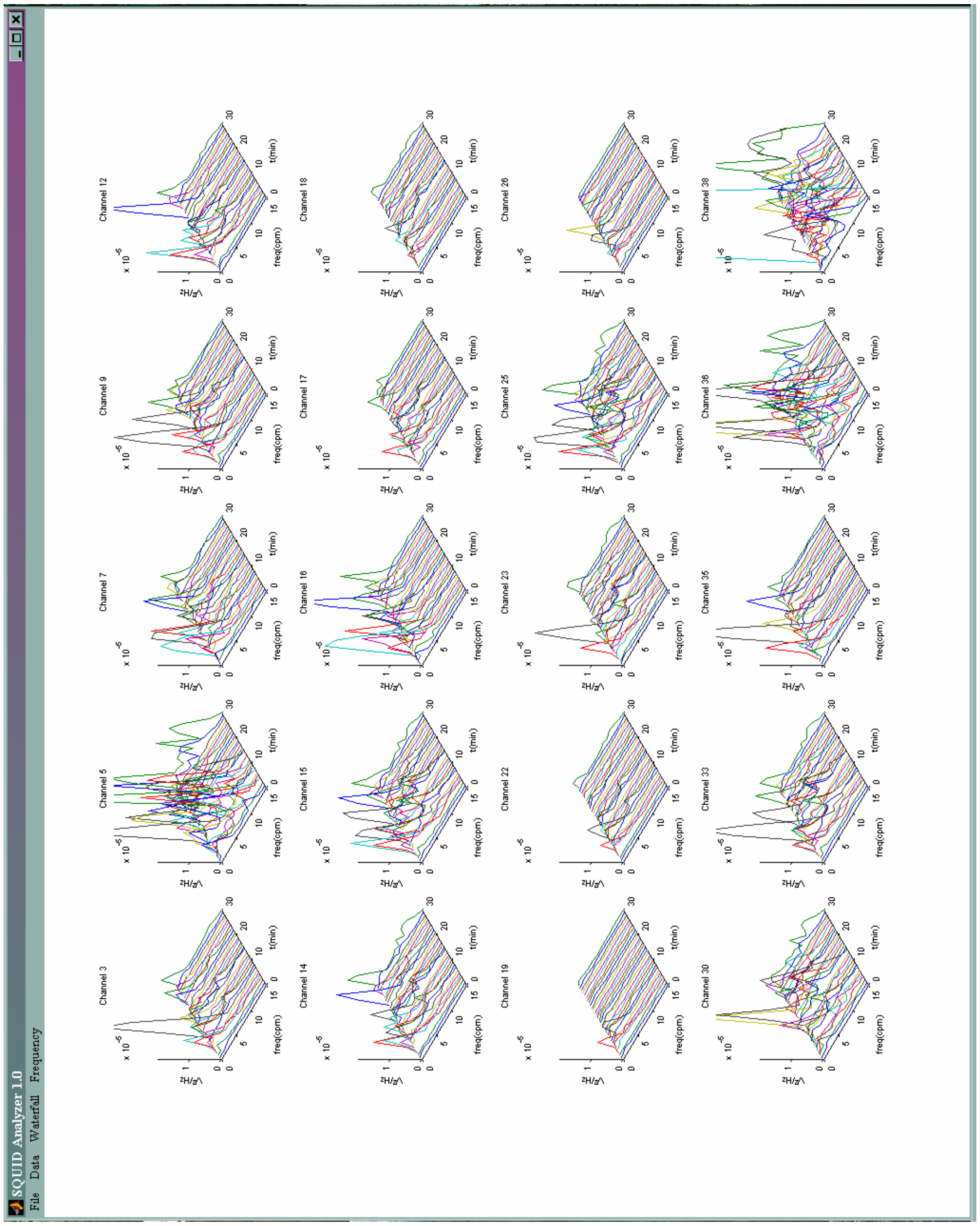


Figure 3.22. Sample waterfall plots generated using the numerical channel order display mode. The grid visibility feature was turned off so as to display solely the waterfall plots and the coordinate axes.

for instance, into segments spanning one cpm, i.e. 1-2 cpm, 2-3 cpm, ... , 14-15 cpm. For each such interval of the form [t_{i-1} , t_i] among the entire set of frequency spectrum intervals [t_0, t_1], [t_1, t_2], ... , [t_{n-1}, t_n], one can compute the definite integral

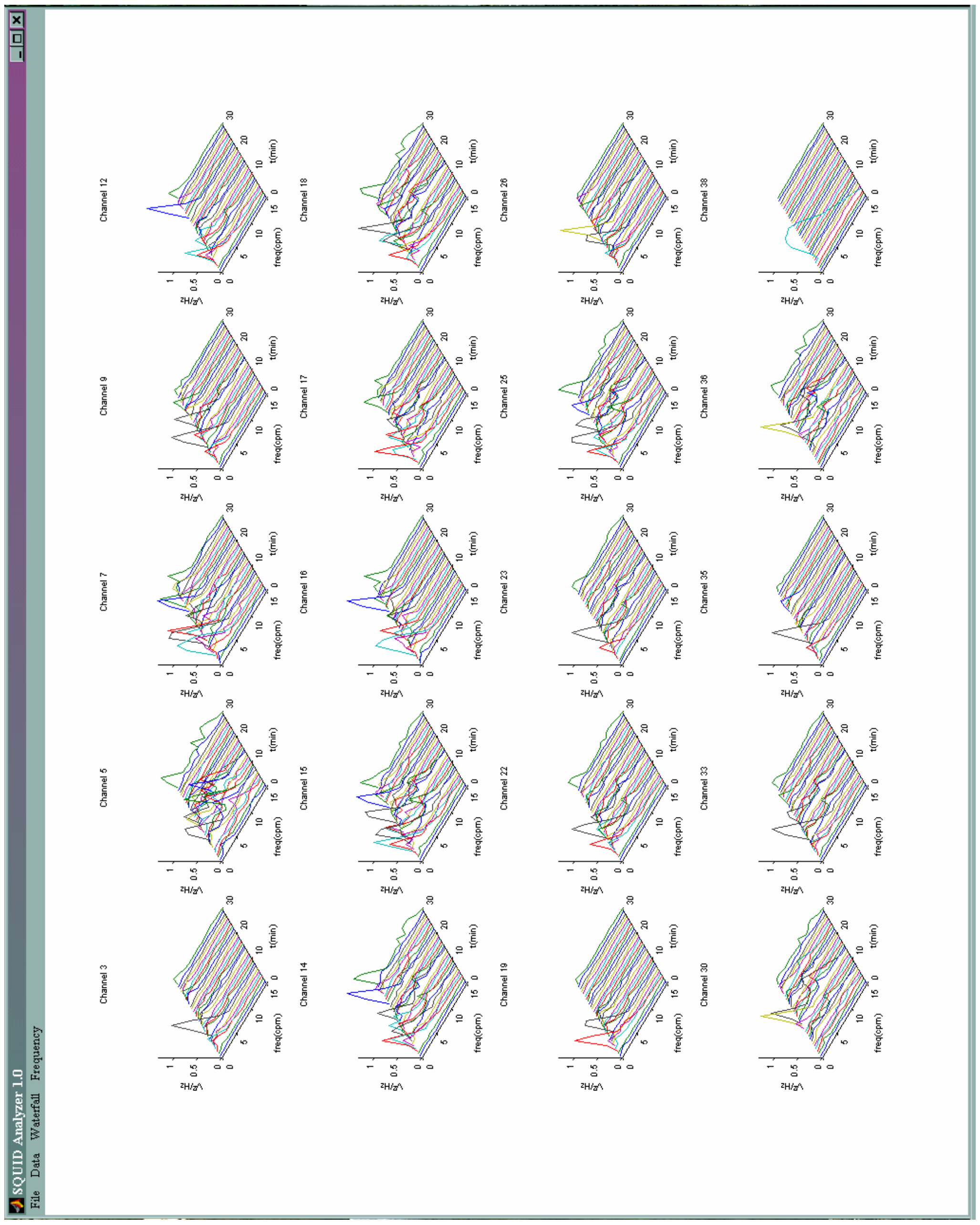


Figure 3.23. Sample waterfall plots generated using the spatial channel arrangement display mode. In addition to the spectra normalization option, the grid visibility feature was turned off so as to display solely the waterfall plots and the coordinate axes.

$$\int_{t_{i-1}}^{t_i} f(x)dx$$

where *f(x)* is the PSD function. This process can be repeated for each SQUID input channel and the value of the integral can be associated with the spatial location of the respective channel.

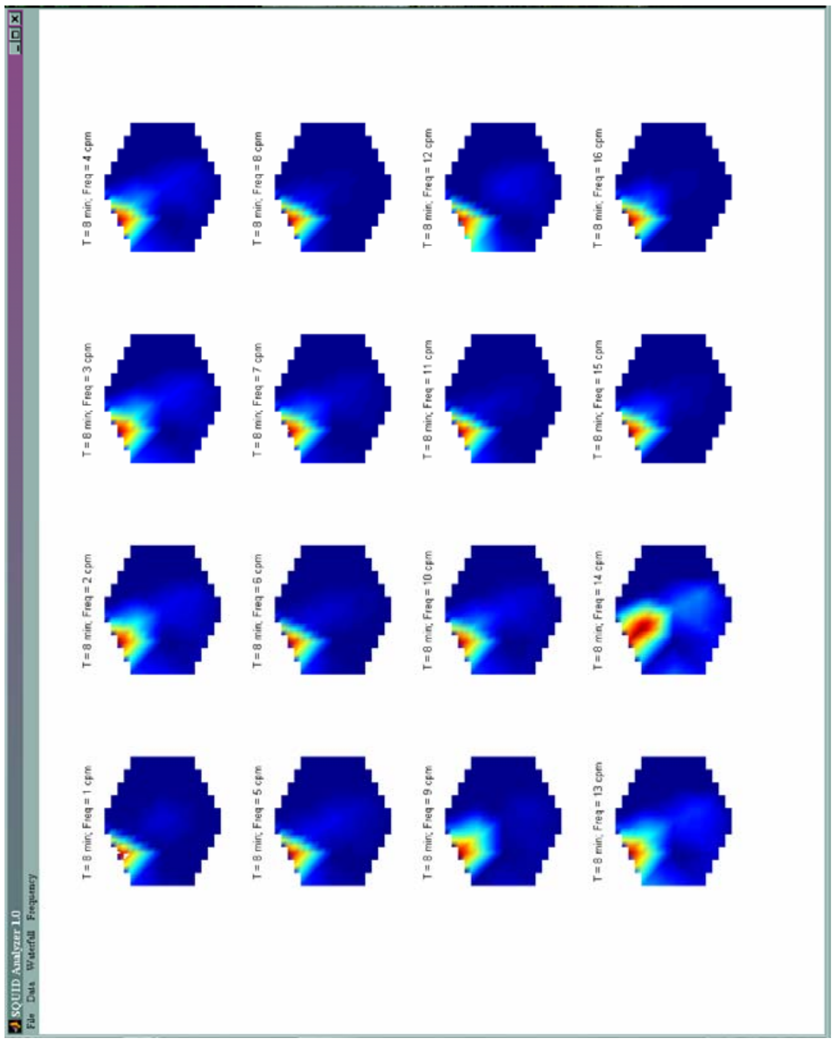


Figure 3.24. Sample frequency maps generated using SQUID Analyzer 1.0. Frequency data within the spatial extent of the biomagnetometer input coils is represented continuously using a data grid interpolation algorithm provided by MatLab. The frequency spectrum is divided into intervals of one cycle per minute each and the corresponding frequency data is used to generate the frequency maps. This is done for the time data points provided in the input file such that one set of frequency maps corresponds to one minute of GI electrical activity contained in the SQUID input data file.

The MatLab function `griddata` can then be used to interpolate these values on a 2D grid within the area delimited by the SQUID input coil perimeter. Thus, one can generate a 3D surface that represents a mapping of frequencies within the SQUID coil perimeter. Such a surface is in effect a function of two coordinate variables (x and y) that allows one to associate a particular frequency with a specific spatial location. In this manner, anatomical regions of the body can be associated with certain frequencies and regions where a particular frequency is dominant can be easily identified from the frequency map [58]. The

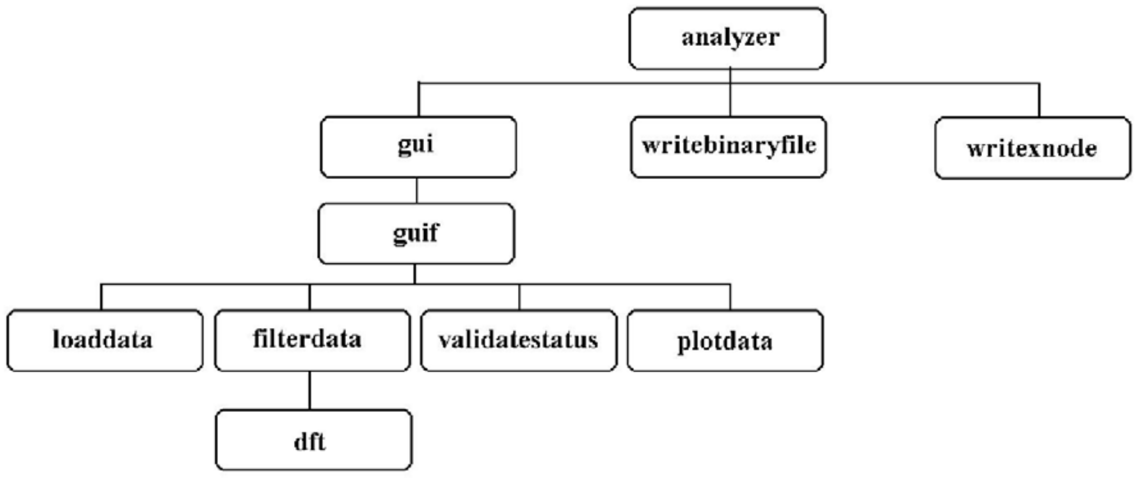


Figure 3.25. SQUID Analyzer function call chart.

process described above can be repeated for each time interval of SQUID data being analyzed; an example of the results obtained for one such time interval is presented in Figure 3.24.

In summary, a list of the SQUID Analyzer MatLab functions is provided in Table 1 in addition to their call chart from Figure 3.26. The `analyzer` function loads the switchboard GUI while `gui` loads the SQUID Analyzer module for data analysis. Most of the other functions are used for data processing and display within the data analysis module with the exception of two functions, `writebinaryfile` and `writexnode`, whose purpose is to perform data conversion from SQUID acquisition format to CMISS visualization format.

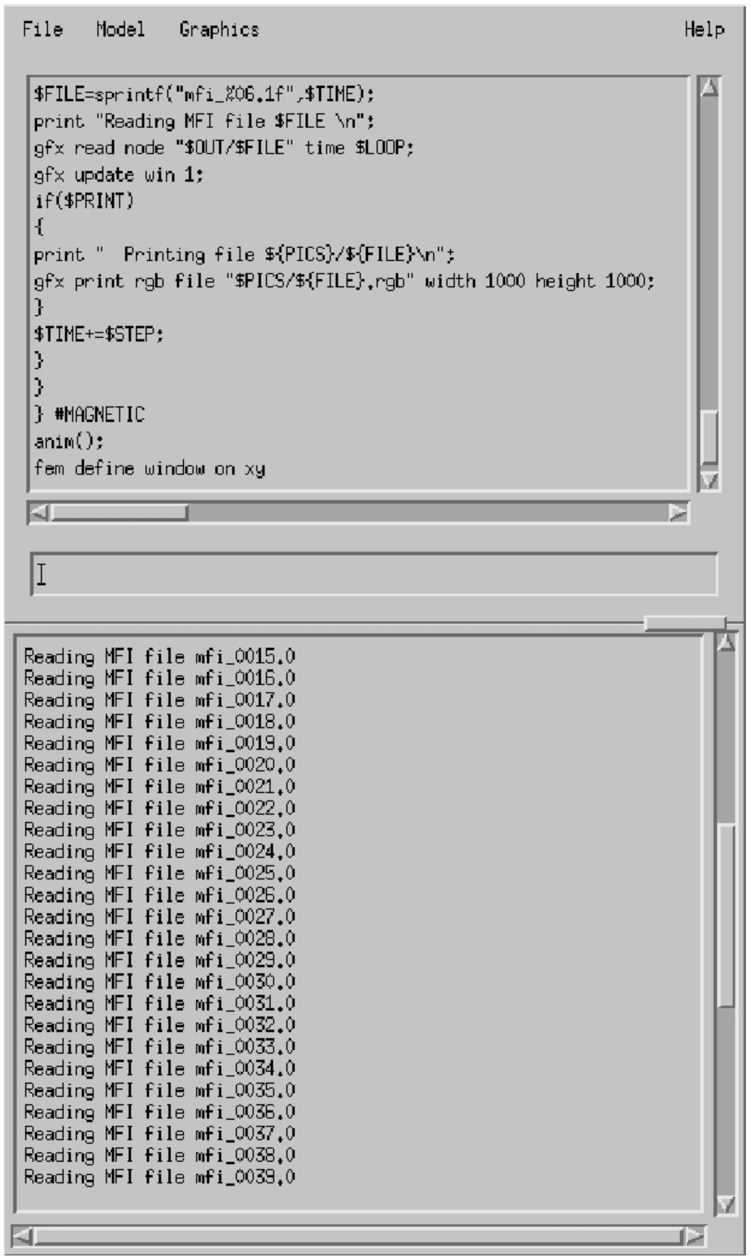
CMISS Visualization Module

The third module of the SQUID Analyzer program was implemented in CMISS. CMISS is a 3D modeling environment developed over the past twenty years in the Bioengineering Institute at the University of Auckland (Auckland, New Zealand). The

Table 1. SQUID Analyzer MatLab functions and their purposes.

analyzer.m	Loads switchboard GUI and handles user input
dft.m	Accepts a SQUID data matrix as input, filters it using a Butterworth filter, and calculates the corresponding power spectrum or power spectral density.
filterdata.m	Accepts a SQUID data matrix as input, applies polynomial subtraction to it, splits the data in time segments and invokes the `dft` function to filter the data time segments.
freqmap.m	Creates and displays a time sequence of frequency maps using the power spectral density matrix accepted as input.
gui.m	Loads the SQUID Analyzer GUI and handles user input from it. All such input is passed to the `guif` function.
guif.m	Handles all functionality of the SQUID Analyzer GUI. Based on the nature of user input, appropriate functions are called to load, display and filter data, etc. Function calls are made to `waterfall` and `freqmap` if waterfall plots or frequency maps must be created.
loaddata.m	Prompts the user to specify a SQUID data input file, which it decimates and loads into memory.
plotdata.m	Takes a matrix of SQUID data and plots it using the display option specified by the user.
validatestatus.m	Validates user input or events, returning `TRUE` or `FALSE`.
waterfallplot.m	Accepts a power spectral density matrix as input and uses it to generate waterfall plots of SQUID data, which it displays to the screen as specified by the user.
writebinaryfile.m	Accepts a SQUID data matrix as input and writes it to a file in CMISS binary format required for numerical calculations.
writexnode.m	Accepts a SQUID data matrix as input and writes it to a file in CMISS `exnode` format required for generating graphics from `cmgui`.

purpose of this software is to provide a set of tools for applying the boundary element method (BEM) and finite element method (FEM) to a variety of problems in bioengineering, including, among others, cardiac and GI biopotential problems. It consists of a number of modules including a graphical front end with advanced 3D display and modeling capabilities, and a computational back end that can be executed remotely on powerful workstations or supercomputers [12]. An example of the CMISS interface for creating and manipulating visual output (called `cmgui`) is presented in Figure 3.26. It consists of an input box where CMISS commands can be entered, a panel where the content of these commands is stored



```
$FILE=sprintf("mfi_%06.1f",$TIME);
print "Reading MFI file $FILE \n";
gfx read node "$OUT/$FILE" time $LOOP;
gfx update win 1;
if($PRINT)
{
print "  Printing file ${PICS}/${FILE}\n";
gfx print rgb file "$PICS/${FILE}.rgb" width 1000 height 1000;
}
$TIME+=$STEP;
}
}
} #MAGNETIC
anim();
fem define window on xy
```

```
Reading MFI file mfi_0015.0
Reading MFI file mfi_0016.0
Reading MFI file mfi_0017.0
Reading MFI file mfi_0018.0
Reading MFI file mfi_0019.0
Reading MFI file mfi_0020.0
Reading MFI file mfi_0021.0
Reading MFI file mfi_0022.0
Reading MFI file mfi_0023.0
Reading MFI file mfi_0024.0
Reading MFI file mfi_0025.0
Reading MFI file mfi_0026.0
Reading MFI file mfi_0027.0
Reading MFI file mfi_0028.0
Reading MFI file mfi_0029.0
Reading MFI file mfi_0030.0
Reading MFI file mfi_0031.0
Reading MFI file mfi_0032.0
Reading MFI file mfi_0033.0
Reading MFI file mfi_0034.0
Reading MFI file mfi_0035.0
Reading MFI file mfi_0036.0
Reading MFI file mfi_0037.0
Reading MFI file mfi_0038.0
Reading MFI file mfi_0039.0
```

Figure 3.26. The CMISS `cmgui` interface. Commands typed in the input box are stored in the first panel and execution feedback is provided by CMISS in the second panel.

throughout each user session, and a second panel where feedback is provided to the user by CMISS with respect to the execution of the commands that were entered. The interface also provides a series of menus for loading commands from input files with the `.com` extension

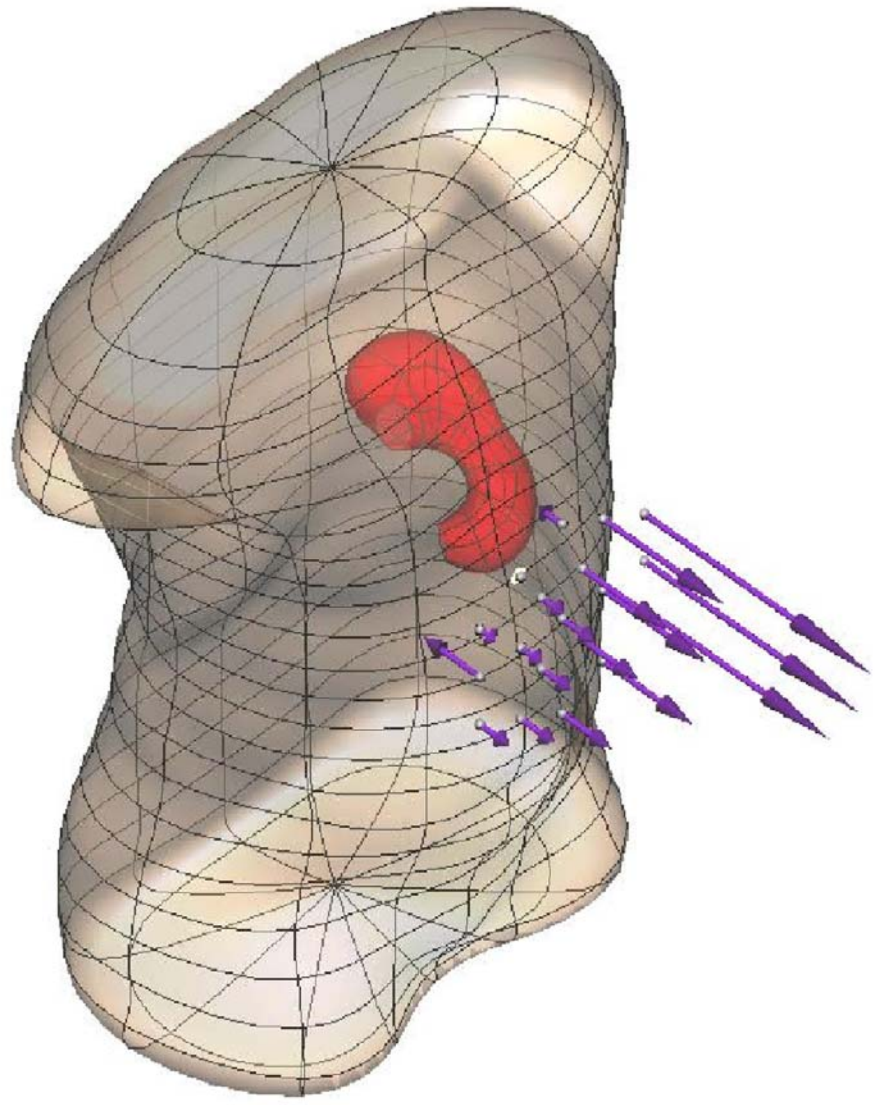


Figure 3.27. CMISS simulation of the stomach within the torso. Above the abdomen, each location of the SQUID magnetometer input coils is also represented by a small dot. For each corresponding input channel, a purple vector arrow is drawn to represent the quantity B_z recorded in that respective SQUID channel.

and for performing a number of various graphical and computational tasks.

After significant contributions to the Cardiome project, the Bioengineering Group at the University of Auckland laid the foundation of the Gut Physiome project, which involves the development of a modeling framework that integrates physiological, anatomical and medical knowledge of the GI system. Equations derived from physical conservation laws are solved in CMISS to predict the integrative behavior of an organ from knowledge of the anatomical structure and tissue properties. The tissue properties used in these organ level simulations can also incorporate tissue structure and cellular processes such as ion movement through membrane channels, signal transduction pathways and metabolic processes, together with the spatial variation of the parameters characterizing these processes. CMISS has facilities for fitting models to geometric data from imaging modalities such as magnetic resonance imaging (MRI) or computed tomography (CT) and has a rich set of tools for graphical interaction with the models and the display of simulation results. Control of the program is via GUIs or scripting languages such as Perl and Python [49].

The main academic goal of CMISS is to support the Physiome project of the International Union for Physiological Sciences (IUPS) [30]. This requires the ability to model biological structure and function at all scales from the molecular structure of proteins (nm scale) to 3D cell models (μm) to tissue models (mm) to whole organ models (m), and the ability to relate models at widely different temporal scales. The computational engine encompasses a number of computational algorithms designed to handle the particular problems of modeling biological structures and systems. The computational and graphical engines can be invoked either together or separately so large computational tasks can be performed on supercomputers and the visualization of results can be done on a graphics workstation. The code base is designed to have a minimum of architecture specific routines allowing CMISS to run on multiple UNIX based platforms including Linux on Intel and AMD processors, AIX on IBM processors and IRIX on Silicon Graphics processors. Parallel processing directives are included in the code base through OpenMP. The visualizations described in this section are the results of work done by the research group of

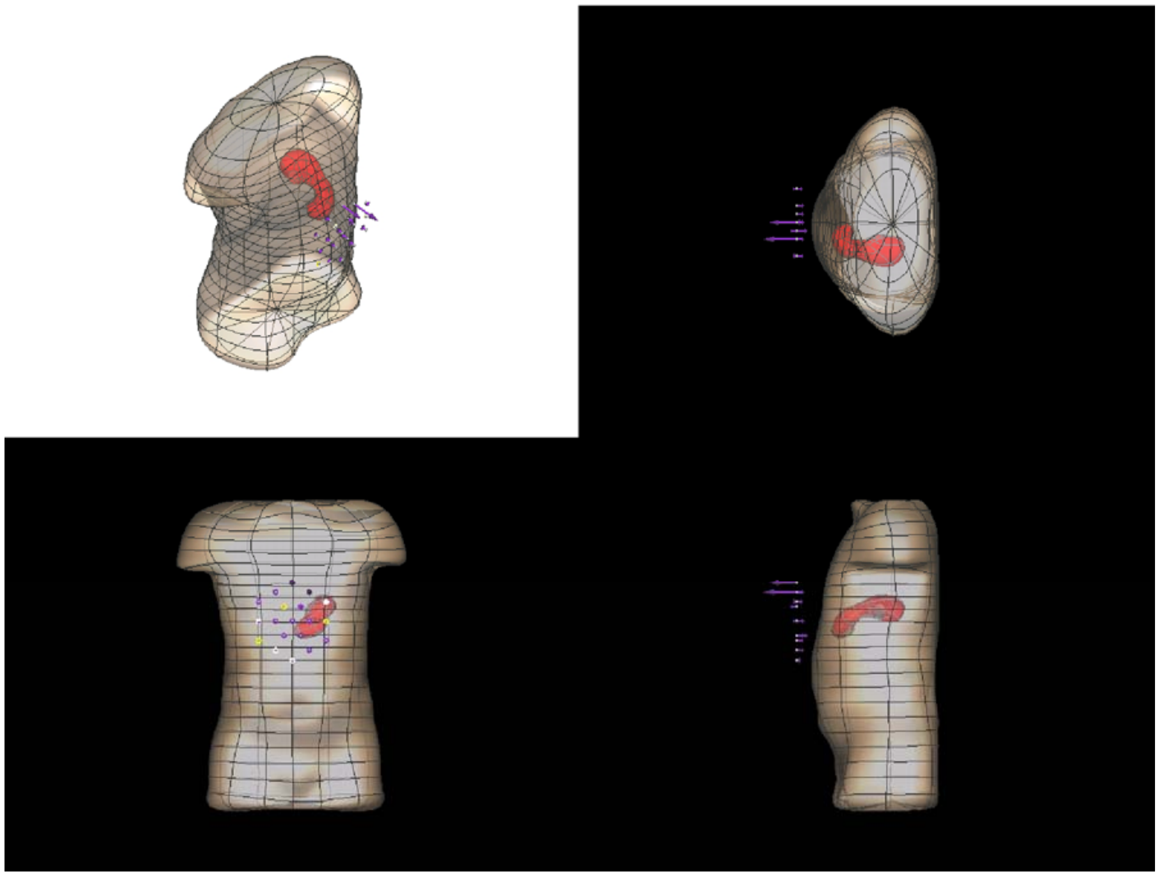


Figure 3.28. Perspective viewpoints available from the CMISS simulation. In the top-left white rectangle, the viewpoint of the camera can be changed by dragging the mouse over the screen. The viewpoint in the other three rectangles cannot be changed, allowing the user to have a frontal, lateral and upper view of the torso and SQUID input locations at the same time.

Andrew Pullan of the Bioengineering Laboratory at the University of Auckland, New Zealand. The ultimate goal of modeling GI electromagnetic phenomena using these visualizations is to provide an integrative framework for describing the physiological, anatomical and clinical knowledge of the GI system at the cellular (continuum) and macroscopic level.

The anatomical data source that was used for the construction of the initial GI model is the visible human project [55]. The image set provided by this project was used to extract data points which were then used to fit an initial generic model to the data via an

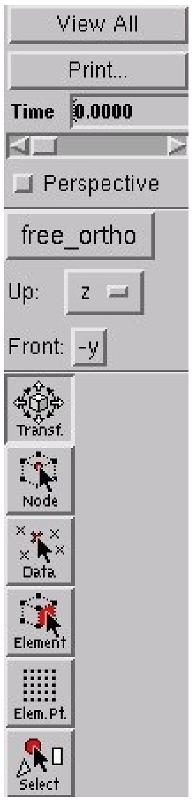


Figure 3.29. CMISS simulation GUI. Various options are provided for controlling the perspective and orientation of the camera viewpoint.

optimization algorithm in which distances between points in the data and fitted models was minimized [48]. From this initial data, a bilinear surface mesh was created and refitted using the iterative fitting procedure of Bradley *et al.* [2] to generate a bicubic Hermite C^1 mesh. This procedure was used for all internal organs, including the small and large intestines. For the stomach, the muscular microstructure was added because the directions of electrical propagation and contraction are often preferential [16, 20, 46]. The thickness of organ walls was added to each digestive organ by internal projection of the outer surface towards the mesh centerline. For the stomach, five layers were included in the model: a longitudinal layer, a thin ICC layer, a circular muscle layer, a second thin ICC layer and a second circular muscle layer [49]. The assumption of transverse anisotropy was made and a microstructural fiber direction was described in each smooth muscle layer.

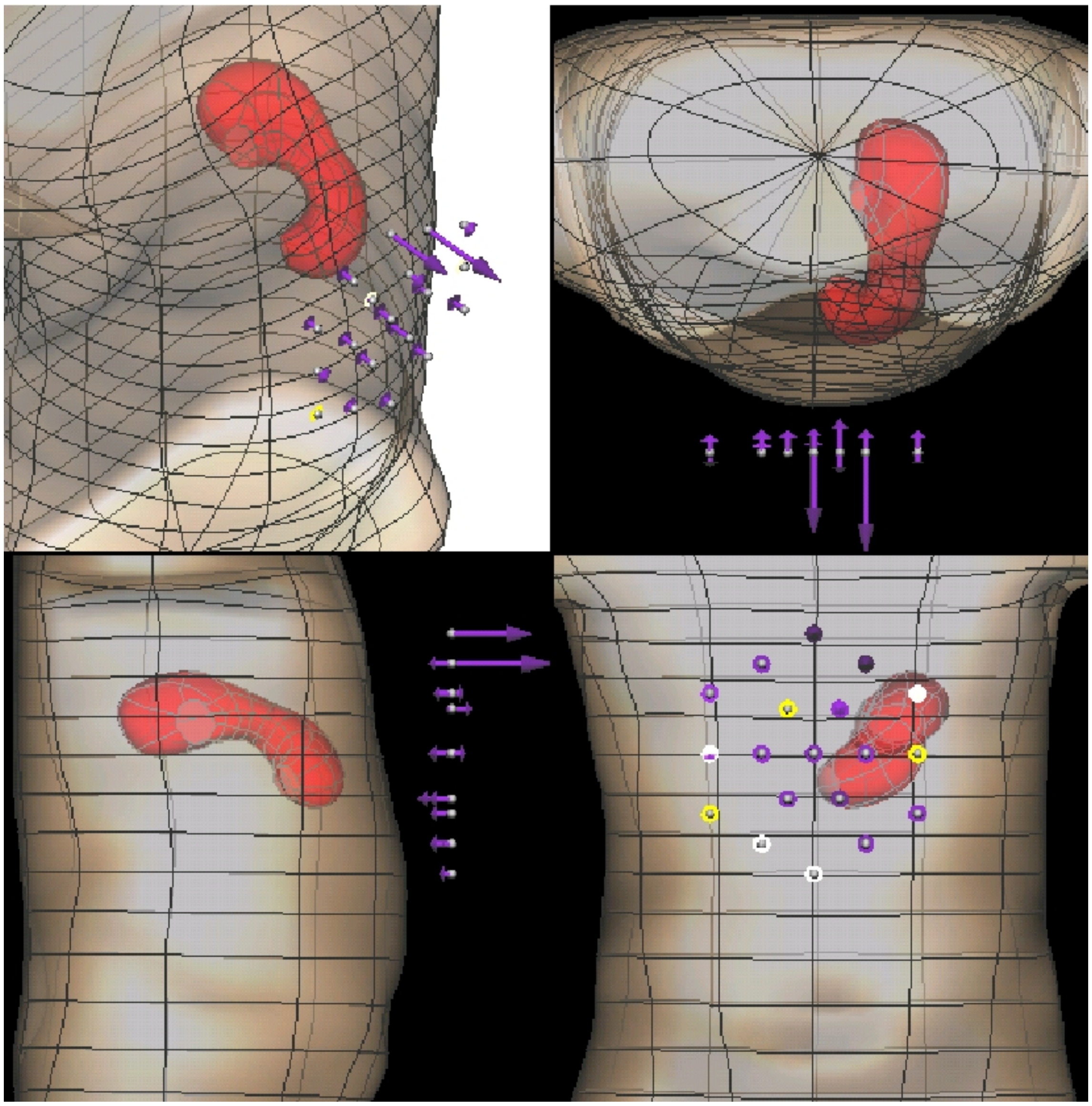


Figure 3.30. CMISS orthographic view of the torso, stomach and magnetic field input locations.

Because the spatial scale of the finite elements fitted from the digitized data is too coarse to solve the bidomain equations, a finite element finite difference method or a structured finite element method was employed [11, 21]. To create a 3D animation of magnetic field vectors, SQUID signal data output by the magnetometer and written to the storage medium must be converted into a format compatible with CMISS. This can be done from the software control switchboard, as previously explained.

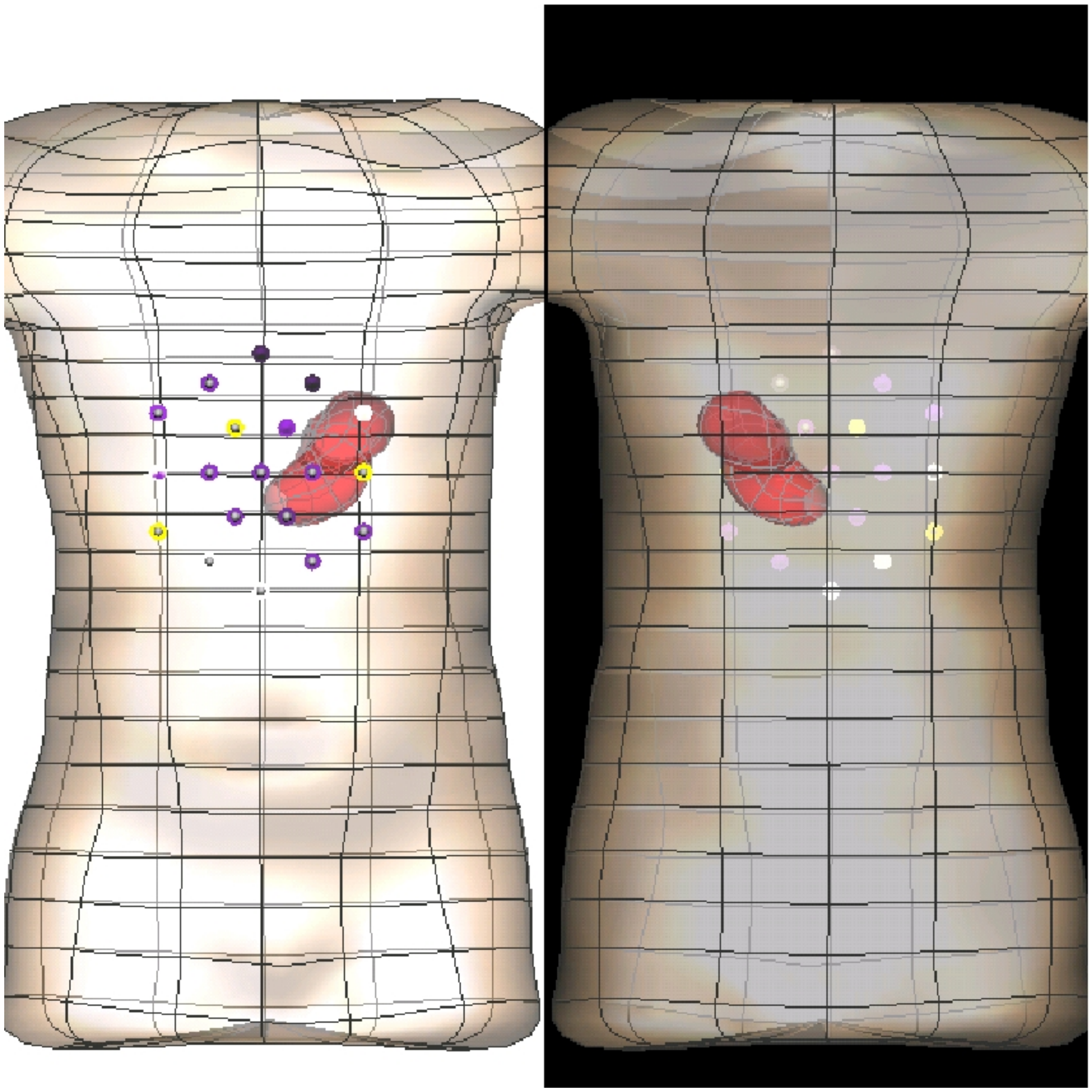


Figure 3.31. CMISS front and back view of the torso, stomach and magnetic field input locations.

Examples of the visualizations produced by CMISS are presented in Figure 3.27 and Figure 3.28. In Figure 3.27, the body of the stomach is simulated within the torso. Above the abdomen, each location of the SQUID magnetometer input coils is represented by a small dot. For each corresponding input channel, a purple vector arrow is drawn to represent the quantity B_z recorded in that respective SQUID channel.

Several views are available for this simulation, as presented in Figure 3.28. In the top-left white rectangle, the viewpoint of the camera can be changed by dragging the mouse over the screen. The viewpoint in the other three rectangles cannot be changed, allowing the user to have a frontal, lateral and upper view of the torso and SQUID input locations at the same time.

Magnetic field data acquired with the SQUID can be related to a model of the patient, or overlain on a generic torso. To incorporate a patient specific model, appropriate anatomical images (CT/MRI) must firstly be acquired, together with appropriate information with respect to the location of the patient with respect to the SQUID sensors. A model of the patient (complete with representations of the stomach and small intestine, if desired) can then be constructed using the methods outlined in [49]. For the information displayed here, a generic torso model, based on the visible human project, was used.

The GUI available in CMISS for controlling the parameters of the simulation is shown in Figure 3.29. By pressing the `View All` button, all four views described in Figure 3.28 are made visible. The `Print ...` button allows the user to take a screen shoot of the CMISS visualization window and save the resulting image to any storage medium. Because one frame is generated for each minute of data, a set of tools are provided to access different frames. One of these is a slidebar, which can be used to visualize different display frames of the simulation. Alternatively, the entire movie can be played in an infinite loop. A perspective image of the torso can be obtained by pressing the radio button `Perspective` and seven different views can selected from the drop-down menu that has the default option `simple`. These views are: `simple`, `free orthographic`, `front_back`, `front_side`, `orthographic` and `pseudo3D`. Examples of the ortographic and front-back views are provided in Figures 3.30 and 3.31, respectively.

CHAPTER IV

DISCUSSION AND CONCLUSIONS

Answering the numerous questions that are of interest in GI research potentially requires a great deal of experimental effort and investigative ingenuity. Undoubtedly, of prime importance in this process is the issue of how experimental data is acquired and analyzed so as to justify realistic and well-grounded scientific conclusions in this respect. With the advent of SQUID magnetometers, the achievement of many research goals that had been heretofore impossible or extremely difficult paved the way towards a solid scientific understanding of physiological mechanisms in the GI system with the potential to revolutionize clinical diagnosis methods. Nevertheless, to convert this potential into an active force of scientific progress and development in the biomedical field, a robust theoretical and computational basis for investigation and analysis is a *sine qua non* condition. Furthermore, in addition to the need of understanding the functional principles of GI bioelectrodynamics, an adequate set of data analysis tools must be readily available to the biomedical researcher. These tools should be specialized enough to allow various individual aspects of GI biophysics to be investigated, but general enough to be useful in a variety of studies that could potentially focus on very distinct phenomenological aspects GI physiology.

As in virtually all other areas of science, the reliability of data analysis tools is crucial to the process of making appropriate interpretations of experimental data and to the ultimate success or failure of any scientific investigation. This is why the development of the data acquisition and analysis software package described in this thesis marks a particularly important step in fulfilling the long-term goal of improving the state of the art in GI clinical diagnosis. Our modular program not only sets the standard for SQUID data analysis software, but also contributes to the establishment of the MGG as a superior, more powerful and authoritative investigative method in clinical research. The modular approach adopted for the development of the SQUID Analyzer software program implies the potential of our integrated analysis tool to undertake continuous development as new theoretical and experimental methods for GI research become important.

Thus far, the use of waterfall plots and frequency maps for GI data analysis has become a standard in many physiological studies [9]. Our CMISS visualization tool, on the other hand, is novel to the modeling of the GI system, although it borrows heavily in ideas and character from similar methodologies that are available for cardiac modeling and simulation. This exchange of techniques between the two areas is salutatory in light of the remarkable progress and extensive experience that already exist in the cardiac field; this historical background is very important because much of the electrodynamics that is now only beginning to be investigated in GI research already benefits from over twenty years of study in cardiac modeling.

Much future research and software development can be done in the direction that has been followed in this thesis. Although many signal analysis concepts that are potentially useful in GI electrodynamics are not discussed here, the modular structure of the SQUID Analyzer program allows much more to be implemented. For example, an important area whose history of investigation in cardiac research is extensive concerns the biomagnetic inverse problem and the information that inverse solutions can yield concerning the patterns of (ab)normal current propagation in the GI tract. While this and many other directions of scientific inquiry remain to be followed, the field of biophysical electrodynamics remains an exciting area of investigation, whose progress must go hand in hand with the development of robust software that can address a wide range of scientific goals and help researchers make continuous contributions to biophysics.

BIBLIOGRAPHY

[1]. W.C. Alvarez (1921) The electrogastrogram and what it shows. *Journal of the American Medical Association*, **78**, 1116-1119.

[2]. L.A. Bradshaw, J.P. Wikswo (1995) Autoregressive and eigenfrequency spectral analysis of magnetoenterographic signals. *Proceedings of the 17th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* CD-ROM.

[3]. C.P. Bradley, A.J. Pullan and P.J. Hunter (1997) Geometric modeling of the human torso using cubic Hermite elements. *Annals of Biomedical Engineering*, **25**, 96-111.

[4]. L.A. Bradshaw, S.H. Allos, J.P. Wikswo Jr. and W.O. Richards (1997) Correlation and comparison of magnetic and electric detection of small intestinal electrical activity *American Journal of Physiology: Gastrointestinal and Liver Physiology*, **272**, G1159-1167.

[5]. S.H. Allos, D.J. Staton, L.A. Bradshaw, S. Halter, J.P. Wikswo Jr., W.O. Richards (1997) Superconducting quantum interference device magnetometer for diagnosis of ischemia caused by mesenteric venous thrombosis *World Journal of Surgery* **21**, 173-178.

[6]. L.A. Bradshaw, J.K. Ladipo, D.J. Staton, J.P. Wikswo, Jr. and W.O. Richards (1999) The human vector magnetogastrogram and magnetoenterogram. *IEEE Transactions of Biomedical Engineering* **46**, 959-971.

[7]. L.A. Bradshaw, W.O. Richards, and J.P. Wikswo Jr. (2001) Volume conductor effects on the spatial resolution of magnetic fields and electric potentials from gastrointestinal electrical activity *Medical and Biological Engineering and Computing* **39**, 35-43.

[8]. L.A. Bradshaw, R.S. Wijesinghe, and J.P. Wikswo Jr. (2001) Spatial filter approach for comparison of the forward and inverse problems of electroencephalography and magnetoencephalography *Annals of Biomedical Engineering* **29**, 214-226.

[9]. L.A. Bradshaw, A. Myers, J.P. Wikswo Jr. and W.O. Richards (2003) A spatio-temporal dipole simulation of gastrointestinal magnetic fields *IEEE Transactions of Biomedical Engineering* **50** 836-847.

[10]. L.A. Bradshaw, A.G. Myers, A. Redmond, J.P. Wikswo Jr. and W.O. Richards (2003) Biomagnetic detection of gastric electrical activity in normal and vagotomized rabbits. *Neurogastroenterology and Motility*, **15**, 475-482.

[11]. L.A. Bradshaw, S.H. Allos, J.P. Wikswo, Jr. and W.O. Richards (1997) Correlation and comparison of magnetic and electric detection of small intestinal electrical activity *American Journal of Physiology*, **272**, G1159-1167.

[12]. M.L. Buist, G.B. Sands, P.J. Hunter and A.J. Pullan (2002) A Deformable finite element derived finite difference method for cardiac activation problems. *Annals of Biomedical Engineering* **31**, 577-588.

[13]. CMISS: An interactive computer program for Continuum Mechanics, Image analysis, Signal processing and System Identification (2004) www.cmiss.org. Accessed July 13, 2004.

[14]. Cohen D., Edelsack E.A. and Zimmerman J.E. (1970) Magnetocardiograms taken inside a shielded room with a superconducting point-contact magnetometer. *Applied Physics Letters*, **16**, 278-280.

[15]. Cohen D. (1972) Magnetoencephalography: Detection of the brain's electrical activity with a superconducting magnetometer *Science*, **175**, 664-666.

[16]. Cohen D. (1968) Magnetoencephalography: Evidence of magnetic fields produced by alpha-rhythm currents. *Science*, **161**, 784-786.

[17]. E.E. Daniel and Y.F. Wang (1999) Gap junctions in intestinal smooth muscle and interstitial cells of Cajal *Microscopic Research and Technology*, **47**, 309-320.

[18]. N.E. Diamant and A. Bartoff (1969) Nature of the intestinal slow wave frequency gradient *American Journal of Physiology*, **216**, 301-307.

[19]. R.G. Duran, C. Padra and R. Rodriguez (2003) A posteriori error estimates for the finite element approximation of eigenvalue problems *Mathematical Models Methods in the Applied Sciences* **13**, 1219-1229.

[20]. J.E. Everhart (1994) Digestive diseases in the United States: epidemiology and impact. U.S. Department of Heath and Human Services, National Institutes of Health, National Institute of Diabetes and Digestive and Kidney Diseases. U.S. Government Printing Office, Washington D.C.

[21]. G.D.S. Hirst (2001) An additional role for ICC in the control of gastrointestinal motility? *Journal of Physiology* **537**, 1.

[22]. D.A. Hooks (2001) Three-dimensional mapping of electrical propagation in the heart: experimental and mathematical model based analysis. Ph.D. Thesis, The University of Auckland, New Zealand, 26-33.

[23]. J.D. Huiziga (2001) Physiology and pathophysiology of the interstitial cells of Cajal: from bench to bedside II. Gastric motility: lessons from mutant mice on slow waves and innervation. *American Journal of Physiology: Gastrointestinal and Liver Physiology*, 281, G1129-G1134.

[24]. R.J. Ilmoniemi, M.S. Hamalainen and J. Knuutila (1985) *Biomagnetism: Applications and Theory*, Oxford: Pergamon, 278-282.

[25]. Irimia and L.A. Bradshaw (2002) Theoretical models and biomathematical algorithms for identifying multiple gastrointestinal dipoles *Proceedings of the IASTED International Conference on Applied Modeling and Simulation*, ACTA Press, Anaheim, CA, 56-60.

[26]. Irimia and L.A. Bradshaw (2002) Recursively-applied scanning algorithms for inverse biomagnetic analyses of gastrointestinal biomagnetic fields *Proceedings of the IASTED International Conference on Applied Modeling and Simulation*, 2002, ACTA Press, Anaheim, CA, 51-55.

[27]. Irimia (2002) Three-dimensional simulations of the gastric biomagnetic field using a recursive dipole localization algorithm *Proceedings of the IASTED International Conference on Applied Modeling and Simulation, 2002*, ACTA Press, Anaheim, CA, 45-50.

[28]. Irimia and L.A. Bradshaw (2003) Theoretical ellipsoidal model of gastric electrical control activity propagation *Physical Review E: Statistical, Nonlinear and Soft Matter Physics*, **68**, 051905.

[29]. Irimia and L.A. Bradshaw (2004) Theoretical and computational methods for the noninvasive detection of gastric electrical source coupling *Physical Review E: Statistical, Nonlinear and Soft Matter Physics* **69**, 051920.

[30]. Irimia, J.J. Beauchamp and L.A. Bradshaw (2004) Theoretical and computational multiple regression study of gastric electrical activity using dipole tracing from magnetic field measurements. *Journal of biological physics*, accepted.

[31]. The International Union of Physiological Sciences (2004) www.iups.org. Accessed July 15 2004.

[32]. Jenks W.G., Sadeghi S.S.H and Wikswo Jr., J.P. (1997) SQUIDS for nondestructive evaluation. *Journal of Physics D: Applied Physics* **30**, 293-323.

[33]. Jenks W.G., Thomas I.M. and Wikswo Jr., J.P. (1997) SQUIDS in *Encyclopedia of Applied Physics*, **19**, 457-468.

[34]. K.L. Koch (1999) Diabetic gastropathy: gastric neuromuscular dysfunction in diabetes mellitus. A review of symptoms, pathophysiology and treatment. *Digestive Diseases and Sciences* **44**, 1061-1075.

[35]. Kothapali (1992) Origin of changes in the epigastric impedance signal as determined by a three-dimensional model *IEEE Transactions on Biomedical Engineering*, **39**, 1005-1010.

[36]. S.L. Marple (1987) *Digital Spectral Analysis with Applications* Prentice Hall, Englewood Cliffs, New Jersey.

[37]. *MatLab 6.0 Help Manual version 12.1* (2003) Mathworks Inc., Nantucket, Massachusetts.

[38]. M.P. Mintchev and K.L. Bowes (1998) Computer simulation of the effect of changing abdominal thickness on the electrogastrogram *Medical Engineering & Physic}*, **20**, 177-181.

[39]. M.P. Mintchev, A. Stickel, S.J. Otto and K.L. Bowes (1997) Reliability of percent distribution of power of the electrogastrogram in recognizing gastric electrical uncoupling *IEEE Transactions on Biomedical Engineering* **44**, 1288-1291.

[40]. M.P. Mintchev, A. Girard, and K.L. Bowes (2000) Nonlinear adaptive noise compensation in electrogastrograms recorded from healthy dogs *IEEE Transactions on Biomedical Engineering* **47**, 239-248.

[41]. M.P. Mintchev, K.L. Bowes (1998) Computer simulation of the impact of different dimensions of the stomach on the validity of electrogastrograms *Medical & Biological Engineering and Computing*}, **36**, 95-100.

[42]. M.P. Mintchev, S.J. Otto and K.L. Bowes (1997) Electrogastrography can recognize gastric electrical uncoupling in dogs *Gastroenterology* **112**, 2006-2011.

[43]. N. Mirizzi, R. Stella and U. Scafoglieri (1986) Model to simulate the gastric electrical control and response activity on the stomach wall and on the abdominal surface *Medical & Biological Engineering and Computing*, **24**, 157-163.

[44]. J.C. Mosher, R.M. Leahy and P.S. Lewis (1995) Matrix kernels for MEG and EEG source localization and imaging, *Proceedings of the International Conference on Acoustics, Speech and Signal Processing, 1995*, 2943.

[45]. J.C. Mosher and R.M. Leahy (1999) Source localization using recursively applied and projected (RAP) MUSIC *IEEE Transactions on Signal Processing* **47**, 332-340.

[46]. T.S. Nelsen and J.C. Becker (1968) Simulation of the electrical and mechanical gradient of the small intestine *American Journal of Physiology* **214** 749-757.

[47]. O'Donovan, C. Feinle-Bisset, K. JOnes and M. Horowitz. Idiopathic and diabetic gastroparesis. *Current Treatment Options in Gastroenterology* **6** 299-309.

[48]. A.V. Oppenheim and R.W. Schafer (1989) *Discrete-Time Signal Processing* Prentice-Hall, 311-312.

[49]. A. Pullan, L. Cheng, R. Yassi and M. Buist (2004) Modeling gastrointestinal bioelectric activity. *Progress in Biophysics and Molecular Biology*, **85**, 523-550.

[50]. A. Pullan (2004) private communication.

[51]. P.Z. Rashev, M.P. Mintchev, and K.L. Bowes (2000) Application of an object-oriented programming paradigm in three-dimensional computer modeling of mechanically active gastrointestinal tissues *IEEE Transactions on Information Technology in Biomedicine*, **4**, 247-258.

[52]. P.Z. Rashev, K.L. Bowes, and M.P. Mintchev (2002) Three-dimensional object-oriented modeling of the stomach for the purpose of microprocessor-controlled functional stimulation *IEEE Transactions on Information Technology in Biomedicine*, **6**, 296-309.

[53]. S.K. Sarna, K.L. Bowes and E.E. Daniel (1976) Gastric pacemakers *Gastroenterology* **70** 226-231.

[54]. J. Sarvas (1987) Basic mathematical and electromagnetic concepts of the biomagnetic inverse problem *Physics in Medicine & Biology*, **32**, 11-22.

[55]. A.J.P.M. Smout, E.J. van der Scee and J.L. Grashuis (1980) What is measured in electrogastrography? *Digestive Diseases and Sciences* **25** 179-187.

[56]. V. Spitzer, M.J. Ackerman, A.L. Scherzinger and R.M. Whitlock (1996) The visible human male: a technical report. *Journal of the American Medical Informatics Association*, **3**, 118-130.

[57]. G.K. Turnbull, S.P. Ritcey, G. Stroink, B. Brandts and P. van Leeuwen (1999) Spatial and temporal variations in the magnetic fields produced by human gastrointestinal activity *Medical and Biological Engineering and Computing* **37** 549-554.

[58]. Tristan Technologies, Inc. (2004) *Intestinal ischemia system - Model 637*. http:// www.tristantech.com/prod_gut.html, accessed on July 10, 2004.

[59]. Z.S. Wang, J.Y. Cheung, S.K. Gao and J.D.Z. Chen (2000) Spatio-temporal nonlinear modeling of gastric myoelectrical activity *Methods of Information in Medicine*, **39**, 186-190.

[60]. N. Weisbrodt (1987) Motility of the small intestine, *Physiology of the Gastrointestinal tract*, Raven Press, New York.

[61]. Wikswo, Jr. J.P. (1995) SQUID magnetometers for biomagnetism and nondestructive testing: important questions and initial answers. *IEEE Transactions on Applied Superconductivity* **5**, 74-80.

APPENDIX

SQUID Analyzer 1.0 MatLab function code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Andrei Irimia
% Living State Physics Laboratories
% Vanderbilt University
% Function name:     analyzer
% Input argument(s): varargin
% Output argument(s):varargount
% Purpose:           The analyzer function has the sole purpose of serving as a
%                    binding function between the GUI and the functions that are
%                    executed when a user event occurs.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = analyzer (varargin)

% GUI Application M-file for gui.fig
% FIG = GUI launch gui GUI.
% GUI('callback_name', ...) invoke the named callback.

if nargin == 0

    % LAUNCH GUI
    fig = openfig(mfilename,'reuse');

    % Use system color scheme for figure:
    set(fig,'Color','white');

    % Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);

    if nargout > 0
        varargout{1} = fig;
    end

elseif ischar(varargin{1})

    % INVOKE NAMED SUBFUNCTION OR CALLBACK
    try
        % FEVAL switchyard
        [varargout{1:nargout}] = feval(varargin{:});
    catch
        disp(lasterr);
    end

end


function varargout = analyzerf (h, eventdata, handles, varargin)
```

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Andrei Irimia & Robert Palmer
% Living State Physics Laboratories
% Vanderbilt University
% Function name:     filterdata
% Input  argument(s):dataPxx  - power spectral density (PSD) of data
%                    f        - frequency vector in cpm
%                    datad    - detrended or detrended and filtered data
% Output argument(s):t        - double array of size 1 x N, where N is the number
%                               of time data points
%                    data     - SQUID magnetometer input data, size 45 x N, where
%                               there are 45 channels and N time data points
%                    spec     - text for returning the power spectrum, 'ps', or
%                               the power spectral density, 'psd'
%                    filter   - text of 'filter' if data is also to be filtered,
%                               'default' will use program set parameters, and
%                               'nofilter' does not filter
%                    filt_parm- filter parameters/ cutoff frequencies in the
%                               form [x y] Hz
% Purpose:           The dft function takes a data matrix and performs a
%                    detrended, windowed, fft.
% Example:           [dataPxx, f, datad] = dft (data, t, spec, filter, ...
%                                       filt_parm, sampling_rate, decimate_num);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [dataPxx, f, datad] = dft (data, t, spec, filter, filt_parm, ...
    sampling_rate, decimate_num);

% Default option is psd
if nargin < 3 | isempty(spec),
    spec = 'psd';
end

if nargin < 4 | isempty (filter)
    filter='nofilter';
end

% Get dimensions of data array
[M, N] = size (data);

% Adjust sampling rate
if isempty (decimate_num),
    sampling_rate1 = sampling_rate;
else
    sampling_rate1 = sampling_rate / decimate_num;
end

% Design butterworth filter
Wn        = filt_parm / (sampling_rate1 / 2);
[bh, ah]  = butter (2, Wn (1), 'high');
[bl, al]  = butter (2, Wn (2));

for i = 1:N,
    datad (:,i) = data (:,i);
    % removes edge effects of filtfilt
    datad1(:,i) = [datad(1:round(sampling_rate1*60),i); ...
                  datad(:,i); datad(M-round(sampling_rate1*60) + 1:M,i)];
    % Filter Data with filter coefficients
    datad1(:,i) = filtfilt (bh,ah,datad1(:,i));
    % Filter Data with filter coefficients
    datad1(:,i) = filtfilt (bl,al,datad1(:,i));
    L1          = length (zeros (round (sampling_rate1 * 60), 1));
    L2          = L1 + length (datad (:,i)) - 1;
    datad (:,i) = datad1 (L1:L2,i);
end

window = hann (M);

% Window to reduce leakage error(presence of signal energy at frequencies where
% there is none)
if M <= sampling_rate1 * 70,
    for i = 1:N,
        datadw    (:,i) = datad (:,i). * window;
        % data zero-padded to increase DFT frequency resolution
        datadwpad (:,i) = [datadw(:,i); zeros (length (datad)*3, 1)];
    end
    % Apply the FFT
    D = fft(datadwpad);
else
    for i = 1:N,
        datadw (:,i) = datad (:,i). * window;
    end
    % Apply the FFT
    D = fft (datadw);
end

m = length (D);
switch spec
    case 'ps'
        % Calculation of Power Spectrum
        dataPs_temp = D. * conj (D) / m;

        % calculation of frequency vector in cycles per minute (cpm)
        f = 60 * (0:round (m / 2)) * sampling_rate1 / m;
        dataPs  = dataPs_temp (1:length (f), :);
        dataPxx = dataPs;

    case 'psd'
        % Calculation of Power Spectral Density, could also be calculated by
        % taking DFT of autocorr.
        % Could include process loss in denominator
        dataPsd_temp = (abs (D). ^ 2). / m * (1 / sampling_rate1);

        % calculation of frequency vector in cycles per minute (cpm)
        f = 60 * (0:round (m / 2)) * sampling_rate1 / m;
        dataPsd = dataPsd_temp (1:length (f), :);
        dataPxx = dataPsd;
end
dplot   = datad;
tplot   = t;

return
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Andrei Irimia & Robert Palmer
% Living State Physics Laboratories
% Vanderbilt University
% Function name:     filterdata
% Input  argument(s):handles  - handles of the GUI
%                    decim    - decimation factor
% Input  argument(s):t        - double array of size 1 x N, where N is the number
%                               of time data points
%                    data     - SQUID magnetometer input data, size 45 x N, where
%                               there are 45 channels and N time data points
%                    chansel  - array containing the channels to be displayed
%                    nchan    - total number of channels
%                    npt      - number of points
%                    srate    - sampling rate of the magnetometer
%                    dset     - character specifying which set of coil data to load
%                               can be 'x', 'y', 'z'
%                    rg       - the time interval to be plotted
% Purpose:           The filterdata function first prompts the user to specify
%                    the cutoff frequencies for the butterworth filter to be
%                    designed. Data is then detrended and a polynomial subtraction
%                    is made to eliminate short-lived trends. The filter is then
%                    applied using the dft function and the filtered data is
%                    returned to the calling function.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [spectra, nplots, cutoff, maxima, absmax, fdata] = filterdata (t, ...
    data, chansel, nchan, npt, srate, dset, rg)

% Some default values
cutoff    = 18;
spec      = 'psd';
filter    = 'filter';
filt_parm = [1.8 cutoff]./60;
limit     = 18;

% Request filter information
filt_parm = inputdlg({'Enter cutoff frequencies in cpm [1.8 18]'},...
            'Filter Cutoff Frequencies',1,{'1.8 18'});
filt_parm = str2num(filt_parm{:,:});
cutoff    = filt_parm (1, 2);
filt_parm = filt_parm./60;

% Find the number of plots in the waterfall plot
nplots   = floor   (size (data, 1) / (srate * 60));
nchan    = size    (chansel, 2);
spectra  = zeros   (1, nplots, nchan);

bar_wait = waitbar (0,'Please wait...filtering data','Name',...
strcat('Filtering data at ', num2str (filt_parm(1)), ' and ', ...
        num2str(filt_parm(2)), ' Hz'));

% Before we do anything, we must detrend the data
% Build polynomial for subtraction
step  = (size(data, 1) / (srate * 60)) / size (data, 1);
tplot = (0:step:(size(data, 1) / (srate * 60) - step))';

for k = 1:nchan
    p     = polyfit (tplot, data (:, k), 2);
    % Build data sequence for polynomial subtraction
    f     = polyval (p, tplot);
    % Subtract the polynomial from the actual data
    data (:, k) = data (:, k) - f;
end

% Create filtered data array
fdata = zeros (size (data, 1), size (data, 2));

% We need to call power spectrum function as many times as necessary
for i = 1:nplots
    nbegin = round ((i - 1) * 60 * srate) + 1;
    nend   = round ( i      * 60 * srate);

    if i == nplots
        nend = size (data, 1);
    end

    % Redo polynomial fit, for each separate minute this time
    for k = 1:(nchan - 1)
        tplot = (nbegin / (srate * 60)):step:(nend / (srate * 60));
        tofit = data (nbegin:nend, k)';
        p     = polyfit (tplot, tofit, 2);
        f     = polyval (p,     tplot);
        data (nbegin:nend, k) = (tofit - f)';
    end

    % Now we can process the data and apply the FFT
    [dataPxx,f,datad] = dft (data (nbegin:nend, 1:nchan),...
                             t    (nbegin:nend, 1      ),...
                             spec, filter, filt_parm, srate, '');
    fdata (nbegin:nend, 1:nchan) = datad;

    % We must interpolate to find the correct spectra
    ndx      = find     (abs(f - cutoff) == min (abs (f - cutoff)));

    % When computing the meshgrid for the x2 y2 values, we must increase
    % the cutoff by 1 to avoid NaN in the interpolation function result
    [x1, y1] = meshgrid (f(1,1:ndx),   1:nchan);
    [x2, y2] = meshgrid (0:cutoff + 1, 1:nchan);
    z1       = dataPxx  (1:ndx, 1:nchan)';
    z2       = interp2  (x1, y1, z1, x2, y2);
    z2       = z2';
    spectra (1:cutoff, i, 1:nchan) = z2 (1:cutoff, 1:nchan);
    waitbar (i / nplots);
end
close (bar_wait);

% We must filter the data another time, but this time do it for the entire data set
[dataPxx,f,fdata] = dft (data, t, spec, filter, filt_parm, srate, '');

% Eliminate abnormal NaN's
gg = find (isnan(spectra) == 1);
spectra (gg) = 0;

for i = 1:size (spectra, 3)
    % Find maxima of the spectra
    maxima (1, i) = max (max (spectra (:, :, i)));
end

absmax   = max (max (max (spectra)));
```

```
return
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Andrei Irimia
% Living State Physics Laboratories
% Vanderbilt University
% Function name:     freqmap
% Input argument(s): cutoff  - spectra cutoff for the butterworth filter
%                    chansel - chanels that were selected, e.g. z channels.
%                    spectra - the array containing the spectra to be mapped
% Output argument(s):none
% Purpose:           The freqmap function creates a frequency map using the
%                    spectra array. First, the locations of the magnetometer coils
%                    are determined mathematically. Then the function griddata is
%                    used to interpolate the frequency values on a two-dimensional
%                    mesh.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function freqmap (cutoff, chansel, spectra)

% Locations for z channels
nm = [3 5 7 9 12 14 15 16 17 18 19 22 23 25 26 30 33 35 36];
XX = zeros (19, 1);
YY = XX;

% Generate locations for inner hexagon
inner = [15 9 35 3 23 25];
r     = 0.5;

% Go around the circle and generate the locations
for i = 1:6
    theta  = pi * (i - 1) / 3;
    xcoord = r * cos (theta);
    ycoord = r * sin (theta);

    XX (find (nm == inner (1, i)), 1) = xcoord;
    YY (find (nm == inner (1, i)), 1) = ycoord;
end

val = sqrt (2) / 2;

% Generate locations for outer hexagon
% 6
XX (find (nm == 17)) = 1.0; XX (find (nm == 16)) = 1.0; XX (find (nm == 12)) = 1.0;
XX (find (nm == 26)) =-1.0; XX (find (nm == 30)) =-1.0; XX (find (nm == 19)) =-1.0;
YY (find (nm == 17)) = 0.0; YY (find (nm == 16)) = 0.5; YY (find (nm == 12)) =-0.5;
YY (find (nm == 26)) = 0.0; YY (find (nm == 30)) = 0.5; YY (find (nm == 19)) =-0.5;
% 2
XX (find (nm ==  5)) = 0.0; XX (find (nm == 18)) = 0.0;
YY (find (nm ==  5)) = 1.0; YY (find (nm == 18)) =-1.0;
% 4
XX (find (nm ==  7)) = 0.5; XX (find (nm == 14)) = 0.5;
XX (find (nm == 36)) =-0.5; XX (find (nm == 22)) =-0.5;
YY (find (nm ==  7)) = val; YY (find (nm == 14)) =-val;
YY (find (nm == 36)) = val; YY (find (nm == 22)) =-val;

[x2, y2] = meshgrid (-1:0.1:1, -1:0.1:1);
x1 = XX;
y1 = YY;

maps = zeros (cutoff, size (x2, 1), size (x2, 1));
```

```
length2 = ceil (sqrt (cutoff));
height2 = length2;

% Create a waiting bar
h = waitbar(0,'Please wait ... creating frequency maps','Name','Creating maps ... ');

% For each time point, create a frequency map and display it
for i = 1:size (spectra, 2)
    % For each frequency, figure out the frequency value to display
    % That information is contained in spectra (freq j, time i, channel)
    for j = 1:size (spectra, 1)
        if size (spectra, 3) == 20
            % If we have z channels, we need not plot respiration
            z1 = squeeze (spectra (j, i, 1:(size (spectra, 3) - 1)));
            z1 = z1 -min (z1);
            if size (find (z1 == 0), 1) ~= 0
                z1 = z1./max (z1);
            end
        end
        % Grid the data available
        maps (j, :, :) = griddata (x1,y1,z1,x2,y2);
    end
    waitbar (i / size (spectra, 2), h);
end
close (h);

% Clear the current image
subplot (1, 1, 1);
axis off;

for i = 1:size (spectra, 2)
    for j = 1:size (spectra, 1)
        % Plot what we just obtained
        subplot (length2,height2,j);
        surf(x2,y2,squeeze(maps(j,:,:)));
        grid off;
        shading interp;
        hold on;
        view (0, 90);
        xlabel ('x');
        ylabel ('y');
        title(['t = ' int2str(i) ' min; Freq = ' int2str(j-1) '-' int2str(j) ' cpm']);
        axis equal tight;
        axis off;
        % Add a colorbar only at the end
        colorbar;
        drawnow;
    end
    hold off;
    pause;
end

return
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Andrei Irimia
% Living State Physics Laboratories
% Vanderbilt University
% Function name:     gui
% Input argument(s): varargin
% Output argument(s):varargount
% Purpose:           The gui function has the sole purpose of serving as a binding
%                    function between the GUI and the functions that are executed
%                    when a user event occurs.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = gui(varargin)

% GUI Application M-file for gui.fig
% FIG = GUI launch gui GUI.
% GUI('callback_name', ...) invoke the named callback.

if nargin == 0
    % LAUNCH GUI
    fig = openfig(mfilename,'reuse');

    % Use system color scheme for figure:
    set(fig,'Color','white');

    % Generate a structure of handles to pass to callbacks, and store it.
    handles = guihandles(fig);
    guidata(fig, handles);

    if nargout > 0
        varargout{1} = fig;
    end

elseif ischar(varargin{1})

    % INVOKE NAMED SUBFUNCTION OR CALLBACK
    try
        [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
    catch
        disp(lasterr);
    end

end

function varargout = spatial_Callback(h, eventdata, handles, varargin)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Andrei Irimia
% Living State Physics Laboratories
% Vanderbilt University
% Function name:     guif
% Input argument(s): f - String type. This argument specifies which case to be
%                    executed within a switch statement. Each case corresponds to
%                    some event triggered by the user in the SQUID Analyzer GUI.
% Output argument(s):none
% Purpose:           The guif function is the most important function in the SQUID
%                    analyzer program. From it all other MatLab functions are
%                    called to load, analyze and plot data in various formats.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The dstatus and wstatus variables have a role in the validation part of the
% program. dstatus refers to whether data was loaded or not into memory. wstatus
% refers to the waterfall plots, i.e. it signals whether data has been filtered
% and the spectra array has been created.
% dstatus --> 0 -- no data was loaded yet
%             1 -- data was loaded and plot was drawn in numerical order
%             2 -- plots were drawn in spatial arrangement
% wstatus --> 0 -- no waterfall plot was drawn yet
%             1 -- a waterfall plot was already drawn by user, hence filtering was
%                  applied; also, the plots were made in numerical order
%             2 -- plots were drawn in spatial arrangement
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function guif (f)

switch f
    case 'open'
        % Clear global variables, if existent
        clear t data chansel filename nchan npt srate handles decim dset axv;
        clear gridv azim elev;

        % Create some global variables that are needed throughout the program
        global t data chansel filename nchan npt srate handles decim dset;
        global dstatus wstatus gridv axv azim elev samescale sfactor rg;

        % Recover the handles in the interface
        handles.file_in_use = findobj ('Tag', 'file_in_use');

        % Assign default values to the dstatus and wstatus variables
        dstatus     = 0;
        wstatus     = 0;
        decim       =10;

        % Load data by calling the loaddata function
        [t, data, chansel, filename, nchan, npt, srate, dset] = ...
            loaddata (handles, decim);

        % A series of variables are assigned in view of future data display
        dstatus     = 1;
        axv         = 1;
        gridv       = 1;
        pstatus     = 0;
        azim        = 45;
        elev        = 45;
        samescale   = 1;
        sfactor     = 1;
        rg          = [1 size(data,1)];
```

```
case 'close'
    % Clear all variables and redraw a blank screen
    clear all;
    subplot (1, 1, 1);
    axis off;
    dstatus = 0;
    wstatus = 0;

case 'exit'
    % Closes Matlab and the interface
    close gui;

case 'plotnumerical'
    % Plot channel data in numerical order
    global t data chansel nchan dset dstatus wstatus rg;

    if validatestatus (dstatus, wstatus, 'PLOT')
        plotdata (t, data, chansel, nchan, dset, 'NUM', rg, 0);
        pstatus = 1;
    end

case 'plotspatial'
    % Plot channel data in spatial arrangement order
    global t data chansel nchan dset dstatus wstatus rg;

    if validatestatus (dstatus, wstatus, 'PLOT') == 1
        plotdata (t, data, chansel, nchan, dset, 'SPA', rg, 0);
        pstatus = 2;
    end

case 'plotfiltnum'
    % Global variables
    global t data chansel filename nchan npt srate handles decim dset;
    global spectra nplots cutoff tq absmax dstatus wstatus axv gridv azim;
    global elev maxima samescale sfactor rg;

    if validatestatus (dstatus, wstatus, 'PLOT') == 1
        % No previous filtering was applied, hence it must be done now
        [spectra, nplots, cutoff, maxima, absmax, fdata] = ...
            filterdata (t, data, chansel, nchan, npt, srate, dset, rg);

        plotdata (t, fdata, chansel, nchan, dset, 'NUM', rg, 1);
        pstatus = 1;
    end

case 'plotfiltspa'
    % Global variables
    global t data chansel filename nchan npt srate handles decim dset;
    global spectra nplots cutoff tq absmax dstatus wstatus axv gridv azim;
    global elev maxima samescale sfactor rg;

    if validatestatus (dstatus, wstatus, 'PLOT') == 1
        % No previous filtering was applied
        [spectra, nplots, cutoff, maxima, absmax, fdata] = ...
            filterdata (t, data, chansel, nchan, npt, srate, dset, rg);

        plotdata (t, fdata, chansel, nchan, dset, 'SPA', rg, 1);
        pstatus = 1;
    end
```

```
case 'range'
    % Declare global variables
    global data dstatus wstatus pstatus srate rg;

    if validatestatus (dstatus, wstatus, 'PLOT') == 1
        % Data range is specified here via in an input dialog box
        in = inputdlg({['Enter data range in seconds within the limits (0, ', ...
             int2str(size (data, 1)./srate), ')']},'Range specification',1,{'0'});
        rg = str2num(in{1,1});
        % Convert the range into array index specifications
        if rg (1, 1) == 0
            rg (1, 1) = 1;
        else
            rg (1, 1) = rg (1, 1).*srate;
        end
        rg (1, 2) = floor (rg (1, 2).*srate);
    end

case 'waterfallnumerical'
    % Global variables
    global t data chansel filename nchan npt srate handles decim dset spectra;
    global nplots cutoff absmax dstatus wstatus axv gridv azim elev maxima;
    global samescale sfactor rg;

    if validatestatus (dstatus, wstatus, 'WATR')
        if wstatus == 0
            % No previous filtering was applied
            [spectra, nplots, cutoff, maxima, absmax, fdata] = ...
                filterdata (t, data, chansel, nchan, npt, srate, dset, rg);
        end
        wstatus = 1;

        % Create waterfall plots
        waterfallplot (spectra, nplots, cutoff, nchan, chansel, absmax, ...
            axv, gridv, wstatus, dset, azim, elev, maxima, samescale, sfactor);
    end

case 'waterfallspatial'
    % Global variables
    global t data chansel filename nchan npt srate handles decim dset;
    global spectra nplots cutoff absmax dstatus wstatus axv gridv axv gridv;
    global azim elev maxima samescale sfactor rg;

    if validatestatus (dstatus, wstatus, 'WATR')
        if wstatus == 0
            % No previous filtering was applied
            [spectra, nplots, cutoff, maxima, absmax, fdata] = ...
                filterdata (t, data, chansel, nchan, npt, srate, dset, rg);
        end

        % Create waterfall plots
        wstatus = 2;
        waterfallplot (spectra, nplots, cutoff, nchan, chansel, absmax, ...
            axv, gridv, wstatus, dset, azim, elev, maxima, samescale, sfactor);
    end

case 'gridvisibility'
    % Global variables
    global t data chansel filename nchan npt srate handles decim dset;
```

```
        global spectra nplots cutoff absmax dstatus wstatus axv gridv azim elev;
        global maxima samescale sfactor rg;

        if validatestatus (dstatus, wstatus, 'WTOP')
            % Turn grid on or off
            if gridv == 1
                gridv = 0;
            else
                gridv = 1;
            end

            % Generate waterfall plots
            waterfallplot (spectra, nplots, cutoff, nchan, chansel, absmax, ...
                axv, gridv, wstatus, dset, azim, elev, maxima, samescale, sfactor);
        end

    case 'axisvisibility'
        % Global variables
        global t data chansel filename nchan npt srate handles decim dset;
        global spectra nplots cutoff absmax dstatus wstatus axv gridv axv gridv;
        global azim elev maxima samescale sfactor rg;

        if validatestatus (dstatus, wstatus, 'WTOP')
            % Turn axis on or off
            if axv == 1
                axv = 0;
            else
                axv = 1;
            end

            % Generate waterfall plots
            waterfallplot (spectra, nplots, cutoff, nchan, chansel, absmax, ...
                axv, gridv, wstatus, dset, azim, elev, maxima, samescale, sfactor);
        end

    case 'plotchange'
        % Global variables
        global t data chansel filename nchan npt srate handles decim dset;
        global spectra nplots cutoff absmax dstatus wstatus axv gridv axv gridv;
        global azim elev maxima samescale sfactor rg;

        % See what the user wishes to do by pressing a key
        usrinput = single (get (gcf, 'CurrentCharacter'));

        % In this case, the user wants to view the graph from a different
        % azimuth or elevation
        if usrinput > 27 & usrinput < 32
            if validatestatus (dstatus, wstatus, 'WTOP')
                if usrinput == 28, azim = azim + 10; end
                if usrinput == 29, azim = azim - 10; end
                if usrinput == 30, elev = elev + 10; end
                if usrinput == 31, elev = elev - 10; end

                % Generate waterfall plots
                waterfallplot (spectra, nplots, cutoff, nchan, chansel, ...
                    absmax, axv, gridv, wstatus, dset, azim, elev, maxima, ...
                    samescale, sfactor);
            end
        end
        % In this case, the user wants to adjust the limits for the axes
        if usrinput == 44 | usrinput == 46
            if validatestatus (dstatus, wstatus, 'WTOP')
                % Increment or decrement z range for the waterfall plots
                if usrinput == 44, sfactor = sfactor / 2; end
                if usrinput == 46, sfactor = sfactor * 2; end

                % Generate waterfall plots
                waterfallplot (spectra, nplots, cutoff, nchan, chansel, ...
                    absmax, axv, gridv, wstatus, dset, azim, elev, maxima, ...
                    samescale, sfactor);
            end
        end

    case 'multiplelimits'
        % Global variables
        global t data chansel filename nchan npt srate handles decim dset;
        global spectra nplots cutoff absmax dstatus wstatus axv gridv axv gridv;
        global azim elev maxima samescale sfactor rg;

        if samescale == 1
            samescale = 0;
            if validatestatus (dstatus, wstatus, 'WTOP')
                % Generate waterfall plots
                waterfallplot (spectra, nplots, cutoff, nchan, chansel, ...
                    absmax, axv, gridv, wstatus, dset, azim, elev, maxima, ...
                    samescale, sfactor);
            end
        end

    case 'uniquelimits'
        % Global variables
        global t data chansel filename nchan npt srate handles decim dset;
        global spectra nplots cutoff absmax dstatus wstatus axv gridv axv;
        global gridv azim elev maxima samescale sfactor rg;

        if samescale == 0
            samescale = 1;
            if validatestatus (dstatus, wstatus, 'WTOP')
                % Generate waterfall plots
                waterfallplot (spectra, nplots, cutoff, nchan, chansel, ...
                    absmax, axv, gridv, wstatus, dset, azim, elev, maxima, ...
                    samescale, sfactor);
            end
        end

    case 'normalize'
        % Global variables
        global t data chansel filename nchan npt srate handles decim dset;
        global spectra nplots cutoff absmax dstatus wstatus axv gridv axv;
        global gridv azim elev maxima samescale sfactor rg;

        if validatestatus (dstatus, wstatus, 'WTOP')
            % Normalize components of the spectra matrix
            for i = 1:size (spectra, 3)
                spectra (:, :, i) = spectra (:, :, i)./maxima(1,i);
            end

            % Generate waterfall plots
            waterfallplot (spectra, nplots, cutoff, nchan, chansel, absmax, ...
                axv, gridv, wstatus, dset, azim, elev, maxima, samescale, sfactor);
        end

    case 'resetwaterfall'
        % Global variables
        global wstatus;

        % Clear figure
        subplot (1, 1, 1);
        axis off;

        % Reset waterfall parameter
        wstatus = 0;

    case 'mapnumerical'
        % Some global variables
        global spectra cutoff dstatus wstatus chansel;

        % Generate waterfall plots
        if validatestatus (dstatus, wstatus, 'WTOP')
            size (spectra)
            freqmap (cutoff, chansel, spectra);
        end

    otherwise
        % Dummy statement, more can be added here.
        a = 0;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Andrei Irimia & Robert Palmer
% Living State Physics Laboratories
% Vanderbilt University
% Function name:     loaddata
% Input  argument(s):handles  - handles of the GUI
%                    decim    - decimation factor
% Output argument(s):t        - double array of size 1 x N, where N is the number of
%                               time data points
%                    data     - SQUID magnetometer input data, size 45 x N, where
%                               there are 45 channels and N time data points
%                    chansel  - array containing the channels to be displayed
%                    filename - name of the file where data is located
%                    nchan    - total number of channels
%                    npt      - number of points
%                    srate    - sampling rate of the magnetometer
%                    dset     - character specifying which set of coil data to load
%                               can be 'x', 'y', 'z'
% Purpose:           The loaddata function first opens a window to prompt the user
%                    for the data to be loaded, then decides which channels to
%                    load for further processing. The input file on disk is then
%                    open and binary data are read into memory. A time vector of
%                    points is also created and returned to the user.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [t, data, chansel, filename, nchan, npt, srate, dset] = ...
    loaddata (handles, decim)

% Record current directory
this  = pwd;

% The decimation factor is fixed at 10, but can be changed
if decim < 0
    % MATLAB decimation performs a low-pass Chebyshev filter (forward and reverse)
    % to decimate with a cut-off freq = sampling_rate * 0.8 / r.
    dec = inputdlg({'Enter a number by which to decimate/down sample ',
        '(e.g. 0,2,4,10,20...):'} ,'Decimation/Down Sampling',1,{'0'});
    dec = str2num(dec{1,1});
else
    dec = decim;
end

% Select the data file
[filename,path] = uigetfile('*.bin;*.dat','Load raw data');
cd (path);
file = [path filename];

% Marks file with an identifying number
fid = fopen(filename,'r','b');

% Returns to beginning of file
frewind(fid);

% Reads sampling rate - 1st line of file
srate = fread (fid, 1, 'float32');

if isempty (dec) ~= 0 | dec > 1
    % Recalculates sampling rate based on decimation factor
    srate = srate./dec;
end
```

```
% Reads # of channels - 2nd line of file
nchan = fread (fid, 1, 'float32');

% Reads # of points - 3rd line of file
npt   = fread (fid, 1, 'float32');

% Detect end of file
fseek (fid, 0, 1);
deof  = ftell(fid);

% Calculates # of points
npt   = (deof-12)/(nchan*4);

% Prompt user to enter the channels to be loaded
prompt  = {'Enter specific channels you would like to load (e.g. "33" ', ...
            'or "33 35 36")'};
title   = ['Load specific channels: channel total = ' num2str(nchan)];
lines   = 1;

% Default value is to load 'z' channels
zval = 'z';

if zval == 'z'
    def = {'z'};
else
    def = {['1:' num2str(nchan)]};
end

% Input Dialog for channel selection
chansel  = inputdlg (prompt,title,lines,def);
chansel  = char (chansel);

% Sort and load channels that record x, y, or z data
if chansel == 'z',
    chansel = sort ([5 36 7 30 35 9 16 26 33 15 17 3 23 25 19 12 22 14 18 39]);
    dset    = 'z';
elseif chansel == 'x'
    chansel = sort ([10 13 34 20 21]);
    dset    = 'x';
elseif chansel == 'y'
    chansel = sort ([4 27 24 11 37]);
    dset    = 'y';
else
    chansel = str2num (chansel);
    chansel = sort    (chansel);
end
N = length(chansel);

% Number for skipping each colmun to the next channel point (channels in rows)
skip     = (nchan-1)*4;
bar_wait = waitbar(0, 'Please wait ... loading data channels', 'Name', ...
           'Loading data ... ');

% Algorithm for pulling specific channels
for i=1:N,

    % Iterate through selected channels
    ch_num = chansel(i);
    frewind (fid);
    % Location of first point of selected channel in file (MATLAB uses 4
    % bytes/number)
    ch_sel = 12 + (4*ch_num - 4);

    % Go to location in file
    fseek (fid, ch_sel, 0);

    if isempty (dec) | dec <= 1,
        % Read in channel data
        data_ch (i, :) = fread (fid, [1 npt], 'float32', skip);
    else
        % Read in channel data w/decimation
        data_ch (i, :) = decimate (fread (fid, [1 npt], 'float32', skip), dec);
    end
    waitbar (i/N);
end

% Get only the required data
data = data_ch';

% Create time vector
t = 0:(1/srate):((length(data)-1)/srate);
t = t';

% Close all unrequired windows and return data
fclose (fid);
close (bar_wait);
cd (this);

return
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Andrei Irimia
% Living State Physics Laboratories
% Vanderbilt University
% Function name:     plotdata
% Input  argument(s):t        - double array of size 1 x N, where N is the number of
%                               time data points
%                    data     - SQUID magnetometer input data, size 45 x N, where
%                               there are 45 channels and N time data points
%                    chansel  - array containing the channels to be displayed
%                    flag     - 'NUM' or 'SPA', indicates whether to make a
%                               numerical or spatial arrangement plot of the data
%                    nchan    - total number of channels
%                    rg       - time interval to be plotted
%                    filtflag - flag indicating whether to filter the data or not
%                    dset     - character specifying which set of coil data to load
%                               can be 'x', 'y', 'z'
% Output argument(s):none
% Purpose:           The plotdata function first creates mappings for each array
%                    to be plotted. Then a for loop plots the signal on the time
%                    interval selected by the user.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function plotdata (t, data, chansel, nchan, dset, flag, rg, filtflag)

% Clear current plot if there is any
subplot (1, 1, 1);
axis off;

% The length of the arrays to plot
length2 = ceil (sqrt (size (chansel, 2))) - 1;
height2 = length2 + 1;

if flag == 'NUM'
    for i = 1:size(chansel,2)
        % Plot what we just obtained
        h (i) = subplot (length2,height2,i);
        hold on;

        % Plot data
        tq    = t (rg(1,1):rg(1,2), 1);
        eval   (['plot(tq./60,data(rg(1,1):rg(1,2),i));']);
        set    (gca,'FontSize', 7);
        xlabel ('time (min)');

        % Create an appropriate label for the abcissa
        if filtflag == 0
            ylabel ('potential (V)');
        else
            ylabel ('B field (T)');
        end

        title  (['Channel ' int2str(chansel(1,i))]);
        grid off;
        axis fill tight;
    end
else
    % Create mappings for each channel of the magnetometer
    % Z channels
    mappingz = [ 0  0  0  5  0  0  0; ...
```

```
             0 36  0  0  0  7  0; ...
            30  0 35  0  9  0 16; ...
            26  3  0 33  0 15 17; ...
            19  0 23  0 25  0 12; ...
             0 22  0  0  0 14  0; ...
             0  0  0 18  0  0  0];

% X channels
mappingx = [ 0 10  0; ...
            20 34 13; ...
             0 21  0];

% Y channels
mappingy = [ 0  4  0; ...
            27 24 11; ...
             0 37  0];

for i = 1:size(chansel,2)
    if dset == 'z'
        [r c] = find (mappingz == chansel (1, i));
        % Plot what we just obtained
        if size (r) ~= 0
            h (i) = subplot (7,7,7*(r - 1) + c);
            hold on;
            tq    = t (rg(1,1):rg(1,2), 1);
            eval   (['plot(tq./60,data(rg(1,1):rg(1,2),i));']);
            set    (gca,'FontSize', 7);
            xlabel ('time (min)');

            % Place appropriate label
            if filtflag == 0
                ylabel ('potential (V)');
            else
                ylabel ('B field (T)');
            end

            % Make title
            title  (['Channel ' int2str(chansel(1,i))]);
            grid off;
            axis fill tight;
        end
    elseif dset == 'x' | dset == 'y'
        % Map the x channels
        if dset == 'x'
            [r c] = find (mappingx == chansel (1, i));
        else
            [r c] = find (mappingy == chansel (1, i));
        end

        % Plot what we just obtained
        if size (r) ~= 0
            h (i) = subplot (3,3,3*(r - 1) + c);
            hold on;
            eval   (['plot(t./60,data(rg(1,1):rg(1,2),i));']);
            set    (gca,'FontSize', 7);
            xlabel ('time (min)');

            % Create y axis labels for the potential or magnetic field
            if filtflag == 0
                ylabel ('potential (V)');
                else
                    ylabel ('B field (T)');
                end

                title  (['Channel ' int2str(chansel(1,i))]);
                grid off;
                axis fill tight;
            end
        end
    end
end

return
```

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Andrei Irimia & Robert Palmer
% Living State Physics Laboratories
% Vanderbilt University
% Function name:     validatestatus
% Input  argument(s):dstatus  - data status indicating whether data was loaded
%                    wstatus  - refers to the waterfall plots, i.e. it signals
%                               whether data has been filtered and the spectra
%                               array has been created.
%                    action   - indicates what action to perform
% Output argument(s):none
% Purpose:           The validatestatus function simply checks to see if the
%                    made by the user follows logically, e.g. waterfall plots
%                    cannot be generated before data is loaded.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function valid = validatestatus (dstatus, wstatus, action)

if action == 'PLOT' | action == 'WATR'
    if dstatus == 0
        msgbox ('Unable to comply, no data has been loaded.', 'Error');
        valid = 0;
        return;
    end
end

if action == 'WTOP'
    if dstatus == 0
        msgbox ('Unable to comply, no data has been loaded.', 'Error');
        valid = 0;
        return;
    end

    if wstatus == 0
        msgbox ('Unable to comply, waterfall plots have not been drawn.', 'Error');
        valid = 0;
        return;
    end
end

valid = 1;

return
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Andrei Irimia
% Living State Physics Laboratories
% Vanderbilt University
% Function name:     waterfallplot
% Input  argument(s):t        - double array of size 1 x N, where N is the number of
%                               time data points
%                    data     - SQUID magnetometer input data, size 45 x N, where
%                               there are 45 channels and N time data points
%                    chansel  - array containing the channels to be displayed
%                    flag     - 'NUM' or 'SPA', indicates whether to make a
%                               numerical or spatial arrangement plot of the data
%                    nchan    - total number of channels
%                    rg       - time interval to be plotted
%                    filtflag - flag indicating whether to filter the data or not
%                    dset     - character specifying which set of coil data to load
%                               can be 'x', 'y', 'z'
%                    absmax   - the maximum value in the PSD array
%                    axv      - flag indicating whether to print axes or not
%                    gridv    - flag indicating whether to print the grid or not
%                    dstatus  - data status indicating whether data was loaded
%                    wstatus  - refers to the waterfall plots, i.e. it signals
%                               whether data has been filtered and the spectra
%                               array has been created.
%                    dset     - flag indicating type of data set
%                    azim     - azimuth value
%                    elev     - elevation value
%                    maxima   - array of PSD matrix maxima
%                    samescale- indicates whther to use the same scale or not
%                    sfactor  - scale factor to reduce or increase z range
% Output argument(s):none
% Purpose:           The waterfallplot function first maps each channel of data
%                    to a certain position on the screen. It then plots the data
%                    at that position. Two display options are made available,
%                    namely numerical channel order and spatial arrangement.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function waterfallplot (spectra, nplots, cutoff, nchan, chansel, absmax, axv, ...
    gridv, wstatus, dset, azim, elev, maxima, samescale, sfactor)

% Erase current plot
subplot (1, 1, 1);
axis off;

% Create a meshgrid in view of plotting
[yy, xx] = meshgrid (1:nplots, 1:cutoff);
length2  = ceil (sqrt (size (chansel, 2))) - 1;
height2  = length2 + 1;

if wstatus == 1
    for i = 1:size(chansel,2)
        % Plot what we just obtained
        h (i) = subplot (length2,height2,i);
        hold on;
        eval   (['plot3(xx,yy,spectra(:,:,i));']);
        set    (gca,'FontSize', 7);
        xlabel ('freq(cpm)');
        ylabel ('t(min)');
        zlabel ('V^2/Hz');
        title  (['Channel ' int2str(chansel(1,i))]);
        % Set grid parameter
        if gridv == 1
            grid on;
        else
            grid off;
        end

        % Set axis parameters
        if samescale == 1
            axis([0 cutoff 0 nplots 0 absmax*sfactor]);
        else
            axis([0 cutoff 0 nplots 0 maxima(1,i)*sfactor]);
        end

        if axv == 1
            axis on;
        else
            axis off;
        end

        % Set the camera view
        view (azim, elev);
    end
else
    % Create z mapping
    mappingz= [ 0  0  0  5  0  0  0; ...
                0 36  0  0  0  7  0; ...
               30  0 35  0  9  0 16; ...
               26  3  0 33  0 15 17; ...
               19  0 23  0 25  0 12; ...
                0 22  0  0  0 14  0; ...
                0  0  0 18  0  0  0];

    % Create x mapping
    mappingx = [ 0 10  0; ...
                20 34 13; ...
                 0 21  0];

    % Create y mapping
    mappingy = [ 0  4  0; ...
                27 24 11; ...
                 0 37  0];

    for i = 1:size(chansel,2)
        if dset == 'z'
            [r c] = find (mappingz == chansel (1, i));

            % Plot what we just obtained
            if size (r) ~= 0
                h (i) = subplot (7,7,7*(r - 1) + c);
                hold on;
                eval   (['plot3(xx,yy,spectra(:,:,i));']);
                set    (gca,'FontSize', 7);
                xlabel ('freq(cpm)');
                ylabel ('t(min)');
                zlabel ('V^2/Hz');
                title  (['Channel ' int2str(chansel(1,i))]);

                if gridv == 1
                grid on;
            else
                grid off;
            end

            if samescale == 1
                axis([0 cutoff 0 nplots 0 absmax*sfactor]);
            else
                axis([0 cutoff 0 nplots 0 maxima(1,i)*sfactor]);
            end
            view (azim, elev);
        end
    elseif dset == 'x' | dset == 'y'
        % Find where data should be plotted
        if dset == 'x'
            [r c] = find (mappingx == chansel (1, i));
        else
            [r c] = find (mappingy == chansel (1, i));
        end

        % Plot what we just obtained
        if size (r) ~= 0
            h (i) = subplot (3,3,3*(r - 1) + c);
            hold on;
            eval   (['plot3(xx,yy,spectra(:,:,i));']);
            set    (gca,'FontSize', 7);
            xlabel ('freq(cpm)');
            ylabel ('t(min)');
            zlabel ('V^2/Hz');
            title  (['Channel ' int2str(chansel(1,i))]);

            % Turn grid on or off
            if gridv == 1
                grid on;
            else
                grid off;
            end

            if samescale == 1
                axis([0 cutoff 0 nplots 0 absmax*sfactor]);
            else
                axis([0 cutoff 0 nplots 0 maxima(1,i)*sfactor]);
            end

            view (azim, elev);
        end
    end

    if axv == 1
        axis on;
    else
        axis off;
    end
  end
end

return
```