4

A DEVICE USER INTERFACE FOR THE GUIDED ABLATIVE THERAPY OF CARDIAC ARRHYTHMIAS

by

Maya Barley

B.S., Electrical Engineering
Rice University, 2001

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the degree of
Master of Science in Electrical Engineering

at the

Massachusetts Institute of Technology

[September 2003]
August 2003

Signature of Author _____

Department of Electrical Engineering and Computer Science
August 1, 2003

Certified by _____

Richard J. Cohen, M.D., PhD. Whitaker Professor in Biomedical Engineering,
Harvard-MIT Division of Health Sciences and Technology
Thesis Supervisor

Accepted by _____

Arthur C. Smith
Chairman, Department Committee on Graduate Studies

A Device User Interface for the Guided Ablative Therapy of Cardiac Arrhythmias

by

Maya Barley

Submitted to the Department of Electrical Engineering and Computer Science on August 1, 2003
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

ABSTRACT

Radio Frequency Ablation (RFA) of cardiac arrhythmias involves the guidance of an ablation catheter to the site of the arrhythmia and the administration of a high-intensity radio-frequency current to the tissue. The current technique used to locate the arrhythmic site suffers from a number of drawbacks. Ablation is a trial-and-error procedure and may require many hours, during which the arrhythmia is ongoing. Patients with hemodynamically unstable VT are therefore excluded, as are those with more complex arrhythmias, accounting for an estimated 90% of patients. Furthermore, the technique is only successful in 71% to 76% of the cases to which it is applied.

A new algorithm was recently identified that allows the non-invasive and rapid detection of the origin of an arrhythmia from body-surface ECG signals, making the RFA procedure accessible to many patients hitherto excluded. Software implementing this algorithm, and providing a multi-layer graphical user-interface to operate in conjunction with an RFA device, has been designed and implemented. If used in tandem with commercially available ECG and ablation catheter devices, this software will allow cardiologists to deliver ablating currents much more precisely and more quickly than is currently possible, and reach a far wider group of patients.

Thesis Supervisor: Richard J. Cohen, MD, PhD.

Title: Professor of Health Sciences and Technology

Acknowledgements

First, I would like to thank Dr. Cohen for his support, help and patience through all this work.

I also want to thank Dr. Bill Stevenson for kindly allowing me to sit in on an ablation procedure and answering all my questions.

Thanks also to Dr. Stephen Burns for helping to get me started on this project, to Antonis for patiently putting up with all my questions about the inverse algorithm while he was trying to work, and to Grace and Tamara, two wonderful lab-mates who have given me advice and guidance.

I want to thank my parents for all their love and never-ending support (and highly useful advice).

And Adrien for always being there for me, always encouraging me to do more than I thought possible, and making my life wonderfully happy.

Table of Contents

1. Clinical Background

1.1 Cardiac Arrhythmias

Cardiac arrhythmias – changes in the regular beat of the heart – are one of the most prominent causes of morbidity in the developed world. It is estimated that 200,000 people a year are treated for ventricular arrhythmias. Furthermore, of the 700,000 deaths each year caused by heart disease in the United States, 60% to 65% are sudden and presumed to be caused by tachyarrhythmias. [1]

1.1.1 Causes

There are several underlying causes of tachycardia. Non-ischemic dilated cardiomyopathy, hypertrophic cardiomyopathy, valvular disease, regional autonomic dysfunction [2] , congenital heart disease and primary electrophysiological abnormalities (such as Wolff-Parkinson-White Syndrome) are all contexts for ventricular tachyarrhythmias. However, eighty percent of sudden cardiac deaths attributed to arrhythmias are due to the after-effects of a myocardial infarction.[3] The electrical properties of infarcted tissue can cause the formation of a reentrant circuit and the precipitation of a lethal ventricular tachycarrhythmia. These regions can also become points of abnormal initiation of impulse activity. Transient myocardial ischemia, perhaps caused by coronary spasm or unstable platelet thrombi, can lead to death via the same mechanisms. [4]

1.1.2 Re-entrant Circuit Formation

As stated above, the conduction properties of infarcted tissue can result in the formation of a re-entrant circuit, a form of abnormal impulse conduction.[5] Figure 1.1 shows a theoretic reentry circuit originating from a chronic infarct. A normal sinus rhythm beat, sweeping through the myocardium to depolarize the entire ventricle, encounters the entrance site of the reentrant circuit. If the circuit 'captures' during this QRS complex, the entrance to the scar region is also activated and a wavefront propagates slowly through the scar tissue. At the *exit point*, it emerges from the infarct region into normal tissue. If the conduction time from entrance to exit within the scar exceeds the refractory period of the functioning myocardium, this will cause the onset of a second QRS complex. To complete the circuit, the wave of depolarization propagates either along the border of the scar (as in Figure 1.1) or through the scar itself, until it once again reaches the exit

site. If the speed of conduction is very fast, a rapidly circulating wavefront or 'circus rhythm' is initiated, and ventricular tachycardia results.

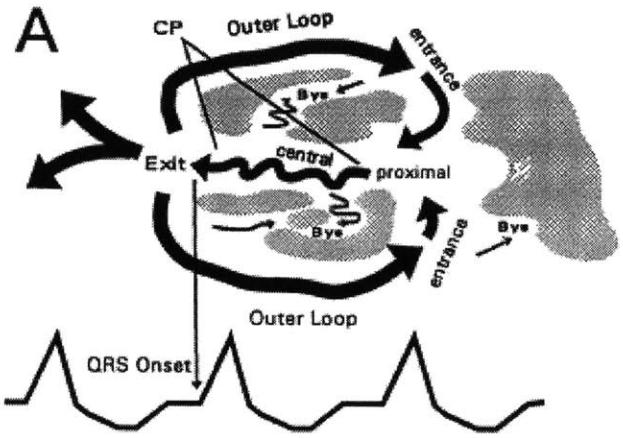


***Figure 1.1**: Reentrant Circuit around an infarct scar. From Stevenson at al. [21]*

1.2 Current Treatment for Arrhythmias

1.2.1 Antiarrhythmic drugs

Antiarrhythmic drugs are frequently prescribed because they alter the electrophysiological properties of the reentrant circuit and suppress potential triggers for the development of VT.[6] However, within 2 years, > 40 % of patients being treated for sustained VT will experience recurrences. [7] Also, an antiarrhythmic drug can create an abnormality on which a transient risk event, such as ischemia, interacts to produce a lethal ventricular arrhythmia. [8]

This was clearly demonstrated in the Cardiac Arrhythmia Suppression Trial (CAST). CAST showed that despite the increased risk of sudden cardiac death associated with ventricular ectopic beats, in post-infarction patients and older age groups, suppression of these arrhythmias with class I drugs such as encainide and moricizine, calcium antagonists, and class III drugs conferred either an increased risk of death or no improvement on survival. [9] It is hypothesized that the negative dromotrophic actions of these drugs exacerbated ischemia-induced conduction slowing. Thus, a transient change in the myocardial substrate could cause anti-arrhythmic drugs to actually become a risk factor for sudden cardiac death.

Only Amiodarone and beta-blockers have been shown to generally decrease the risk of sudden death in the myocardial infarction patient. However, amiodarone produces side effects in 75% of patients within 5 years, including hypothyroidism, tremor and neurological toxicity. Clearly, a more effective means of treating ventricular arrhythmias is needed.

1.2.2 Implantable Cardioverter-Defibrillators

Three recent studies confirm the superiority of implantable cardioverter defibrillators (ICDs) over antiarrhythmic drug therapy for preventing sudden death in patients at risk for hemodynamically unstable ventricular tachyarrhythmias. [10], [11], [12] These devices can administer anti-tachycardia pacing that will terminate most monomorphic VTs, and electrical cardioversion if necessary. During cardioversion, a single high-energy pulse of current is administered to the heart, 'resetting' the cycle of electrical activity; it is hoped that once the tissue is uniformly de-polarized, normal sinus rhythm will resume.

Approximately one third of patients will experience some adverse effect, including inappropriate shocks, lead problems, and infection. [13] The 'shock' of cardioversion felt by ICD patients is substantial and there are many instances of patients either requesting that the device be removed or refusing implantation. [14] Often, amiodarone is used in conjunction with an ICD since it reduces the frequency of cardiac events necessitating defibrillation. However, antiarrhythmic drugs can lower the rate of VT to just below the ICD's threshold of detection or raise the energy required for defibrillation, reducing the probability that the ICD will be effective. Furthermore, ICD implantation and maintenance is expensive; the MADIT study found the average cost to be $27,000 per patient. [15]

Although ICDs extend survival, they only treat arrhythmias when they occur and do not prevent recurrences. [16] A form of preventive treatment is therefore desirable for cases in which the VT is persistent. The most recently developed procedure for the treatment of arrhythmias is radio-frequency ablation (RFA). It involves the guidance of an ablation catheter to the exit site of the reentrant circuit and the administration of high-intensity radio-frequency current to the tissue. If the site has been accurately identified and ablated, the necrotic tissue that remains will transect the isthmus of the circuit. The arrhythmia will then be non-inducible. In stable ventricular tachycardia patients without structural disease, this treatment is often used to lessen the number of ICD therapies that must be administered to patients. In one study [17], the total number of ICD

therapies in 21 patients with any VT decreased from 59.3 ± 79.7 per month before ablation, to 0.6 ± 1.1 per month after successful ablation; this represents a 99.8% reduction in defibrillator therapies. These patients reported a significant improvement in quality-of-life after successful ablation.

1.3 Radio-Frequency Ablation

At the beginning of the procedure, ablation and electrode catheters are inserted into either a femoral artery or vein, depending on the location of the arrhythmic site suggested by the morphology of the VT, and advanced into the heart. The catheter positions are monitored with biplanar fluoroscopy (a 2-D imaging technique allowing 'real-time' x-ray), with 30° right anterior oblique and 60° left anterior oblique projections. To remove positional changes due to heart motion, the fluoroscopic images are usually R-wave gated. A detailed map of endocardial electrical activity is produced using techniques defined below. Throughout the mapping procedure, the patient is administered 1000 units/hour of intravenous heparin to prevent coagulation and thrombus formation around the catheters.

Once the site of the arrhythmia is determined from this map, fluoroscopy is used to guide the ablation catheter to the same site and the ablating current is delivered. The frequency of the current used is between 300 and 750 kHz, with a power output of 20 to 50W over a 30 to 90 second period. Irreversible tissue destruction requires a temperature of approximately 50 °C; therefore the power output of the radio-frequency generator is automatically or manually titrated to achieve a temperature between 60 and 75 °C. If the temperature at the catheter tip exceeds 100°C, coagulated plasma and dessicated tissue will clump on the electrode tip. This predisposes the patient to thromboembolic complications and requires that the catheter be removed so that the tip can be wiped clean. A sudden rise in electrode impedance is indicative of coagulation at the tip, and energy delivery is ceased.

Typical ablation catheters are 2.2 mm in diameter and create a lesion 5-6 mm in diameter and 2-3 mm deep. The lesion consists of a central core of necrotic tissue surrounded by a zone of hemorrhage and inflammation. The inflammation may resolve without residual necrosis, therefore if the reentrant exit site lies on the periphery of the lesion the arrhythmia could recur even several weeks after ablation. [18] Consequently, the ablation must be accurate to within one to two millimeters of the arrhythmic site to ensure success.

1.4 Current Endocardial Mapping Techniques

1.4.1 Mapping of Hemodynamically Stable Ventricular Tachycardia

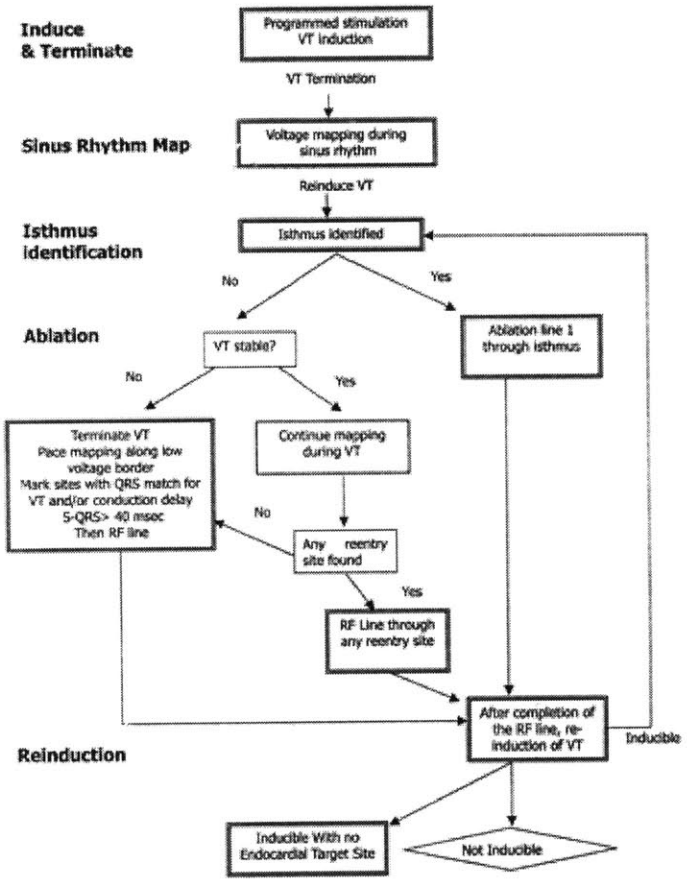


***Figure 1.2**: Flow diagram of approach to mapping and ablation of VT, Soejima et al. [21]*

Initial VT induction

The sequence of mapping and ablation for both stable and unstable VT is shown in Figure 1.2. VT is first induced by rapid pacing to determine its surface QRS morphology. The arrhythmia is then terminated using cardioversion or burst pacing.

Sinus Mapping

Next, the electrical characteristics of the endocardial surface of the heart are mapped during sinus rhythm. This is done by consecutively stimulating the endocardium at over 500 sites using an intracardiac electrode and recording the local bipolar electrograms. Observations of animal MI models suggest that reentry circuit isthmuses can be best defined by delineating infarcted areas of dense, fibrous scar, since these are potentially arrhythmogenic. [19] The sinus mapping procedure identifies scar tissue by stimulating the endocardium sub-threshold while sinus rhythm is ongoing. The resulting local electrograms are measured a few millimeters from the pacing electrode. Scar regions are characterized by low-voltage (< 1.5 mV) multiphasic electrograms. Slow conduction, suggested by a long delay between a supra-threshold pacing stimulus and surface QRS onset, is also indicative. Therefore, scar regions are termed *zones of fragmentation.*

An example of an electroanatomical map produced by sinus mapping is shown in Figure 1.3.

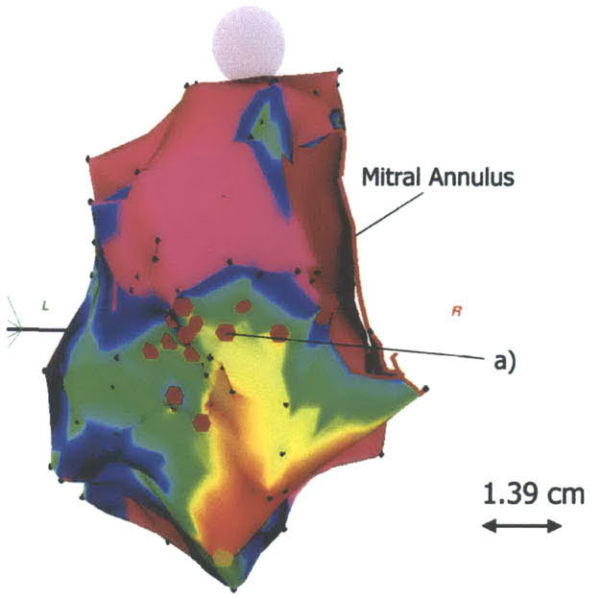


Figure 1.3: Sinus Mapping. *Colors indicate sinus rhythm electrogram amplitude , with lowest-amplitude areas of red, increasing to yellow, green amd blue. Normal voltage electrogram regions (> 1.5 mV) generated by 'healthy' endocardium are shown in purple. From Soejima et al. [21]*

Pace Mapping

Pace-mapping to mimic the VT is then carried out. This technique is based on the principle that supra-threshold pacing at the site of arrhythmic origin would result in an identical surface ECG morphology to that of the clinical VT. [20] By comparing and finally matching the QRS morphologies, the cardiologist can localize the site of reentry. For scar-based reentrant circuits, the mapping takes place along the border of the scar region to identify the exit point of the arrhythmia. Movement of the catheter from the exit site into the scar tissue itself demonstrates abnormal low-voltage electrograms and a supra-threshold pacing morphology similar to the VT morphology (often with a significant delay between stimulus and QRS).

Entrainment mapping

Finally, *entrainment mapping* is the gold standard for ablating hemodynamically stable reentrant circuits. It is carried out after pace-mapping has localized the site of the arrhythmia, and is used to identify the exact location. Figure 1.4 illustrates the principles of entrainment.

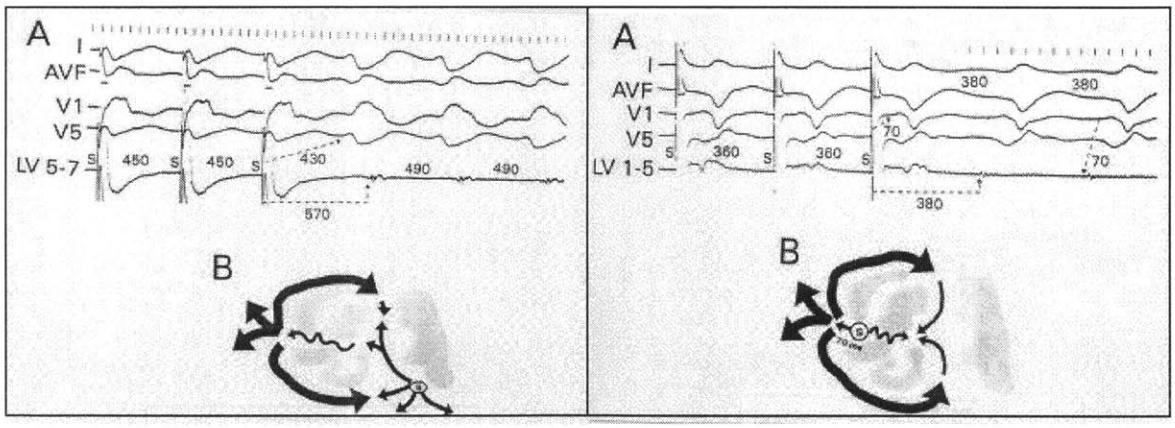


Figure 1.4: Entrainment Mapping. *The left illustration shows the ECG resulting from pacing outside the reentrant circuit; note the long post-pacing interval (570 ms) as compared to the VT cycle length (490 ms), and the difference in QRS morphology during and after pacing. The right illustration shows pacing at the exit site of the reentrant circuit. The post-pacing interval (380 ms) is equal to the VT cycle length, and the QRS morphologies during and after pacing are identical.Stevenson et al. [21]*

While the VT is ongoing, an endocardial site is paced at a cycle length 40 to 100 ms shorter than that of the VT. Unless the site is remote to the reentrant circuit, this results in acceleration of the VT to the pacing cycle length. If the site is not within the reentrant circuit, the *post-pacing interval* – the time from the last pacing stimulus after pacing is stopped until the subsequent depolarization *at the pacing site* due to 'return' of the activation wavefront – will be more than 30 ms greater than the VT cycle length, as seen in Figure 1.4. This difference results from the addition of conduction time from pacing site to reentrant circuit and back to the conduction time through the reentrant circuit itself.

If the pacing site lies within the reentrant circuit, there is no significant conduction time between pacing site and reentrant circuit. However, pacing at a cycle length significantly shorter than the VT transiently alters the intracellular ionic composition of the myocytes. This changes the conduction properties of the cardiac tissue; therefore, a post-pacing interval within 30 ms of the VT cycle length is accepted.

When pacing occurs within the reentrant circuit, the collision of the pace-stimulated antidromic wavefront and the orthodromic wavefront (returning through the circuit) occurs near the pacing site. Since the antidromic wavefront depolarizes only a small section of tissue, this is not detectable on the surface ECG and the fusion of the two wavefronts is concealed [21]. This is called *entrainment with concealed fusion.* Ablation at such a site will result in termination of the tachycardia.

1.4.2 Mapping of Hemodynamically Unstable VT

It is impossible to sustain patients with more severe, hemodynamically unstable arrhythmias for the 3 to 6 hours in which VT is ongoing during entrainment mapping. [17] Instead after initial VT induction, the arrhythmia is quickly terminated. Sinus mapping is used to identify scar regions, and pace-mapping distinguishes sites at which the paced QRS morphology matches that of the unstable VT. Since pace-mapping is far less accurate than entrainment mapping, ablation in patients with unstable VT is generally unsuccessful and therefore rarely attempted.

1.4.3 Ablation

During ablation, a line of RF lesions is created in an attempt to transect the isthmus through which abnormal beats are transmitted to the functioning myocardium. In patients with reentrant circuits originating from MI scar regions, ablation lines follow the contours of the scar. The procedure is considered successful if the original VT is no longer inducible.

1.4.4 Probability of Success

The current probability of RFA success in a patient with purely monomorphic, stable VT is 71 to 76%. However, it is likely that this represents <10% of the total population of patients with VT. [22] Although Furniss et al. [23] have conducted a limited study indicating that RFA can be used to successfully treat patients with hemodynamically unstable arrhythmias, these patients are not generally treated unless their arrhythmia is incessant. In these cases, ICD implantation is considered the most appropriate option.

The extended duration of the procedure also exposes cardiologists and patients to undesirable levels of x-ray radiation. Furthermore, since fluoroscopic imaging of the heart yields a two-dimensional projection of a 3-D structure, it is difficult for the cardiologist to accurately gauge the catheters' positions in three-dimensional space. The LocaLisa system is a newly developed, non-fluoroscopic technique to localize the 3-dimensional position of ablation electrodes. It has been shown to yield reproducible results with an accuracy to less than 1.4 +/- 1.1 mm [24]. However, this system does not map the arrhythmia itself, therefore the overall accuracy of the ablation procedure is still limited.

Lastly, it is theoretically desirable to limit the number of ablation sites to the minimum required for success, minimizing the risk of damage to functioning myocardium and the creation of potentially thrombogenic endocardial lesions. [25] Current treatment may create upward of thirty lesions, and achieves only limited success.

Consequently, catheter ablation is currently used to improve a VT patient's quality of life but not to cure [26]. However, novel mapping technologies have been developed that could improve the success rate of the procedure and reduce the rate of recurrence of the arrhythmia.

1.5 New Mapping Technologies

1.5.1 CARTO

Two types of activation-mapping technology currently exist for use in radio-catheter ablation. 'CARTO' uses a special catheter to generate 3-D electroanatomic cardiac maps, and is now widely used in RFA procedures. A device external to the patient's body emits a very low magnetic field that is detected at the tip of the mapping and ablation catheter, and is used to sense its location and orientation. The catheter tip simultaneously stimulates the cardiac tissue and records the resulting local electrocardiograms. The amplitude of the local electrograms during sinus mapping, and the site at which they were recorded, are displayed in a 3-D electro-anatomical map (as shown in Figure 3) that clearly delineates scar tissue. During pace and entrainment mapping, CARTO allows the cardiologist to mark sites on the map that demonstrate concealed fusion or a QRS morphology matching that of the VT. The catheter tip is also displayed; the display is R-wave gated so that movement due to heart motion is cancelled.

As with the multiple lead imaging technique, CARTO has several drawbacks. Mapping is time-intensive therefore hemodynamically unstable patients are still untreatable. Also, the degree of resolution of the endocardial map is limited by the time available to acquire data points (upwards of 550 electrograms are required during ventricular mapping). The 3D map is not provided in real-time therefore new maps must be generated to detect a change in the arrhythmia or to fully-visualize multiple VTs. Lastly, multiple-point acquisition has to be performed with care, ensuring that contact with the endocardium is adequate and that fibrous, low-voltage structures such as the mitral valve annulus are appropriately delineated. Otherwise, scar-regions can be falsely exaggerated. [20]

1.5.2 Ensite 3000

Another recent development is the Ensite 3000 basket catheter, a mapping system that uses a single, non-contact intra-cavity multielectrode array to sense the voltage field produced by endocardial activation. The 64-electrode braid array computes virtual electrograms simultaneously from more than 3000 ventricular sites using a boundary element inverse solution. This information is then used to reconstruct the entire chamber's endocardial activity, which is displayed as a dynamic three-dimensional isopotential color map.

This is undoubtedly the best currently-developed mapping technique for complex or hemodynamically unstable arrhythmias, and has shown success in several studies. [27] However, the overall accuracy of reconstructed electrograms decreases with distance from the electrode array affecting the validity of the map. [28] Also, the endocardial geometry that is used to calculate the boundary element inverse solution during VT is acquired during baseline rhythm; the assumption that the geometry does not change may be an important limitation if the heart is vigorously contracting. Furthermore, aggressive anticoagulation measures must be taken which can lead to serious bleeding complications. The Ensite 3000 is not widely used within the medical community.

1.5.3 Body Surface Potential Mapping

This is the only non-invasive mapping technology currently available, and is not widely used. An array of 64, 128 or 256 electrodes is placed on the torso, and the body surface potentials are recorded. The potentials on the heart surface are then computed by solving Laplace's equation within the torso volume, assuming a specific geometric relationship between the epicardial and torso surfaces. A dynamic isopotential map is then created which shows the spread of activation across the ventricles. By pacing at different sites within the ventricle, a database of isopotential activation maps is generated. By comparing the VT isopotential map to those in the database, the computer guides the catheter to the correct site for ablation.

Several studies have shown that BSP mapping is effective at imaging reentry pathways and their key components. [29] By improving the accuracy of reentry site identification, this mapping technique can also reduce the number of ablations required to terminate the VT. [30] Furthermore, this system can reconstruct epicardial electrophysiological information, unlike CARTO and Ensite 3000. Since the sub-epicardium plays an important role in the maintenance of reentry circuits in up to 33% of patients, this is an important advantage.

However, in a study done by SippensGroenewegen [31], BSPM was found to localize the site of the arrhythmia only to within 2 cm of a pace-mapping site. Since an ablation lesion is only 5 to 6 mm in diameter, this degree of localization is insufficient. Also, BSPM has a limited ability to identify and resolve multiple simultaneous cardiac events, since the potentials at the body surface are a weighted sum of electrical contributions from the entire heart. Consequently it has been

suggested that BSPM be used for screening purposes or to examine the effect of certain medications on the arrhythmic substrate, rather than as a mapping tool in RF ablation procedures.

2. A New Method: The Inverse Technique

2.1 Solving the Inverse Problem

The Inverse Problem in electrocardiography involves the mapping of two-dimensional body-surface potentials to the three-dimensional cardiac excitation pattern that created them. The endocardial excitation pattern at any instant can be visualized as a wavefront of single electric dipoles. This is approximated as a belt source with a constant dipole moment per unit length directed parallel to the propagation direction. By summing the individual dipoles, the source can be approximated as a cumulative single equivalent moving dipole (SEMD). Therefore, the solution to the inverse problem is the SEMD that best reproduces the electrode potentials at a given set of electrodes at a given time instant.

In general, the inverse problem has no unique solution. Furthermore, it has been shown that a single equivalent point dipole source cannot adequately represent cardiac electrical activity throughout the cardiac cycle. [32] However if the source is well-localized, as is the case when the wavefront initially exits from a narrow reentry isthmus, the approximation is valid.

An algorithm has been identified that would allow an SEMD's location, strength and orientation to be reliably calculated from one sample of a non-invasive, 64-lead body surface ECG. [33] Using this algorithm, a *sequence* of dipoles can be calculated from the body surface potentials generated over a single beat of VT. From this dipole trajectory, the one that best represents the reentry-circuit exit site can be identified. The same inverse algorithm may then be used to determine the location and orientation of the catheter tip. The algorithm assumes a homogenous volume conductor; however, in reality blood, lungs, bone, muscle, fat and fluid have very different conductivities. Although studies have shown that torso inhomogeneities have a minor effect on BSPM patterns [34], if the cardiac and catheter dipoles are matched both in location and orientation, it is hoped that any spatial inaccuracies due to tissue inhomogeneities will cancel.

2.2 Mathematical Basis for the Inverse Algorithm

The torso is assumed to be a cylinder of radius r_torso, and length l_torso. The torso volume is filled with cubes 1.5 cm on a side, and a dipole is assumed to lie at the center of each cube. The dipole moment $\boldsymbol{p}$ for each dipole is then found using 3 plus 3 parameter optimization. [33] The estimated potential in each lead, $\phi^i(\boldsymbol{r}, \boldsymbol{p})$, is then found using an infinite volume conductor estimation:

$$\phi = \frac{1}{4\pi g} \frac{\boldsymbol{p} \bullet (\boldsymbol{r}' - \boldsymbol{r})}{|\boldsymbol{r}' - \boldsymbol{r}|^3}$$

where $\boldsymbol{r}'$ represents the electrode location and $\boldsymbol{r}$ the dipole location. The dipole that best 'fits' the measured potentials is then selected by finding the dipole that minimizes an objective function, χ^2 per degree of freedom:

$$\psi(\boldsymbol{r}, \boldsymbol{p}) = \chi^2/dof = \frac{1}{dof} \sum_{i=1}^{I} \left(\frac{\phi^i(\boldsymbol{r}, \boldsymbol{p}) - \phi_m^i}{\sigma^i} \right)^2$$

where $\phi^i(\boldsymbol{r}, \boldsymbol{p})$ is the estimated potential at the ith electrode due to a dipole $\boldsymbol{r}$ with moment vector $\boldsymbol{p}$. ϕ_m^i is the actual measured potential at the ith electrode. σ^i is the noise measurement in lead i, dof = number of electrodes – number of fit parameters, and I is the number of electrodes.

The cube containing the best-fit dipole and its neighboring cubes are then filled with cubes 0.5 cm on a side and the χ^2-minimization procedure is repeated to find the best-fit dipole. This continues until the resolution of the cubes is 1 mm to a side. The dipole at this resolution whose body surface potentials best approximate the measured voltages is chosen as the SEMD representation for that time sample.

2.3 Benefits of using the Inverse Algorithm as a Mapping Tool

The current rate of success for VT ablation therapy is less than 70%. With a resolution on the order of millimeters, we believe that the Inverse Technique will allow cardiologists to deliver ablating currents much more precisely and also less invasively than mapping technologies such as

CARTO. Furthermore, exposure to fluoroscopy radiation will also be much reduced for both patient and cardiologist. This will decrease the risk of cancer and genetic defects in the patient's offspring [35], and make the procedure more applicable for use in children.

Since the algorithm requires the body-surface potentials from *only a single beat of VT* to calculate the trajectory of the equivalent dipole over the cardiac cycle, only a few seconds of data need be recorded for the re-entrant site to be localized. Therefore, complex or hemodynamically unstable arrhythmias – occurring in greater than 90% of patients, who were hitherto excluded from RFA therapy – can now be localized and treated. Consequently, the number of deaths that result from treatment with anti-arrhythmic drugs, and the number of patients requiring ICD implantation, could be significantly reduced. This will have significant economic impact and dramatically improve quality-of-life for those with ventricular arrhythmias.

3. Design of the RFA Device

3.1 Overall RFA system design

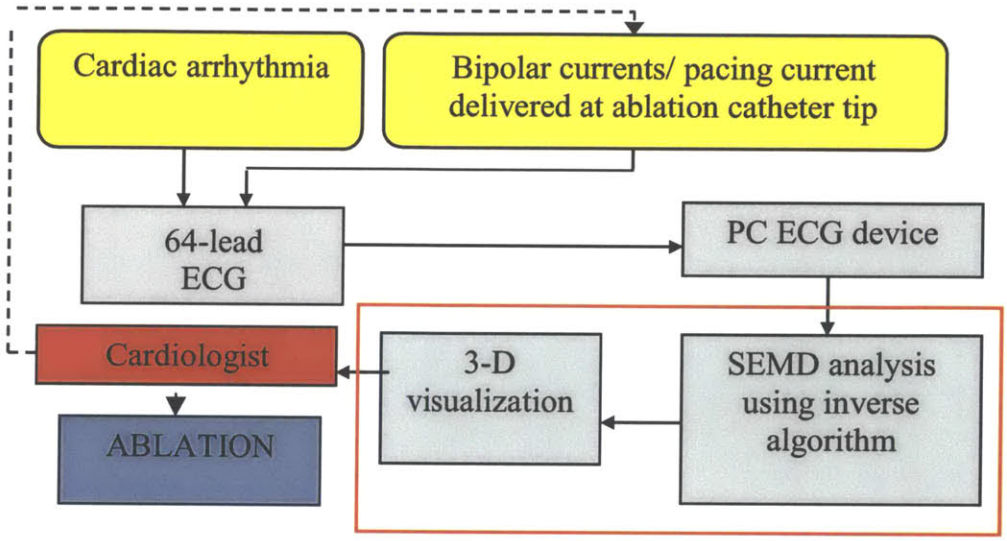


Figure 3.1: *RFA device flow chart*

3.1.1 Locating the re-entrant circuit

The complete radio-frequency ablation device would be designed as shown in Figure 3.1. Potentials recorded by the 64-lead body-surface ECG are transmitted to a laptop computer, which simultaneously displays the input in real-time and stores the data in memory. The user interface to the device allows the cardiologist to examine the ECG recordings and select the data segment of interest in the VT waveform. This is processed by the SEMD inverse algorithm software, which outputs the dipole parameters for every sample in the selected data segment. The dipoles are then displayed in a 3-dimensional graphical interface. The cardiologist can manipulate this environment and individually view each dipole's parameters. The dipole that best represents the ablation site is then chosen based on various factors (see section 8).

3.1.2 Guiding the Ablation Catheter

Once the dipole that best represents the ablation site is chosen, the cardiologist must align the ablation catheter tip and the dipole. The implementation of the ablation catheter is beyond the scope of this project; however, I will mention two design possibilities that would allow the cardiologist to do this.

If bipolar currents are delivered from the tip of a specially-designed catheter, at a frequency above the range of bio-potentials (1 to 250Hz), this can be detected by the body-surface electrodes. Once the signal is band-pass filtered at the frequency of these bipolar currents, the same inverse algorithm can be used to calculate the location of the catheter-tip 'dipole'. If the catheter-tip and reentry-site dipoles are matched both in location and orientation, it is hoped that tissue inhomegenities and systematic error will have no effect on the accuracy of the match.

An alternative approach is to pace with the ablation catheter tip at the same cycle length as the tachycardia and record the resulting ECG potentials. By sampling the electrode potentials that occur within a few milliseconds of each pacing stimulus, and finding the inverse solution to these potentials, we can locate the dipole closest to the catheter tip (and hence the tip itself, if good endocardial contact is maintained). As with the bipolar current method, tissue inhomogeneities would theoretically have no effect since only relative distance between catheter tip and re-entry site is important.

This second approach has numerous advantages. Firstly, the ablation catheter need not be custom designed, since commercially available catheters have the ability to both pace and ablate. Furthermore, if the VT is fast, electrical activity from the previous beat is still present when the next wave of depolarization leaves the reentry circuit. Since an SEMD is the vector sum of all cardiac excitation this could introduce significant error into the dipole approximation. However, if the catheter tip were used to pace *at the same rate* as the VT, the excitation overlap would theoretically be the same and the relative error between the cardiac and catheter dipole approximations would be zero.

Lastly, the pacing option is more financially viable. Most commercially available ECG systems have a frequency cutoff around 250 Hz. If the bipolar current method were used, the bandwidth of

the ECG device would need to extend to at least 500 Hz to prevent bio-potential interference with the catheter-tip signal. High-bandwidth systems do exist, however they are far more expensive.

3.2 Biomedical Safety Considerations

Commercially available medical devices must conform to several safety guidelines, most importantly AAMI EC11, IEC601-1, IEC601-1-1, IEC601-1-2 and IEC601-2-25. These guidelines relate mainly to hardware, and impose maximum values on lead leakage, ground-lead resistance, etc. However, some are applicable to the current software design project.

Electric shock is a major hazard in the medical environment. The patient is often connected to several monitoring devices simultaneously; if one of these develops an electric fault and the lowest-impedance connection to ground exists though the patient, he/she will be exposed to potentially lethal currents. If the ECG device develops a fault or provides this connection to ground, the position of the electrodes directly above the heart increases the risk of death. A current of only 75 to 400 mA in this region will cause ventricular defibrillation, while 1- 6 amps will cause a sustained myocardial contraction.

To provide patient isolation from the mains source, the software should be run on a battery-powered laptop. Secondly, to isolate the patient from ground, it is vital that the electrode connections to the ECG device have an input impedance of greater than 100 MOhm at frequencies around 60 Hz. Also, since a defibrillator may need to be used if the patient develops a dangerous tachycardia, the ECG device should be protected to 4kV (360 Joules). This level of isolation can be provided if an isolation amplifier is used to filter and amplify the electrode signals, and these are streamed in real-time to the laptop-based ECG system via a fiber-optic cable or remote connection.

3.3 Choice of Operating System and Software

Safety considerations also affect the choice of operating system (OS). The OS must be able to perform in real-time, in a critical-care environment where loss of data due to a 'crash' could result in failure of the procedure. Since physicians may not have much computer training and little time to learn, the OS must have a simple user-interface. Also, since this application requires many graphical windows to be run simultaneously, features such as off-screen memory support

are a significant advantage. One such operating system, the QNX OS, is currently used in several critical-care medical applications.

However, since this project was aimed more at 'proof of concept' than commercial-viability, it was decided to implement the system in Windows 2000/ME/XP in spite of its poor reputation for fault tolerance. Unlike UNIX, Windows has a physician-friendly interface. The Mac OS is also a possibility, although there are many more commercially available platforms that can support a Windows environment.

4. The Graphical User Interface

4.1 User Interface Structure

The flowchart in Figure 4.1 shows the broad structure of the user interface. For each step, a list of specifications was compiled and approved as described in this section.

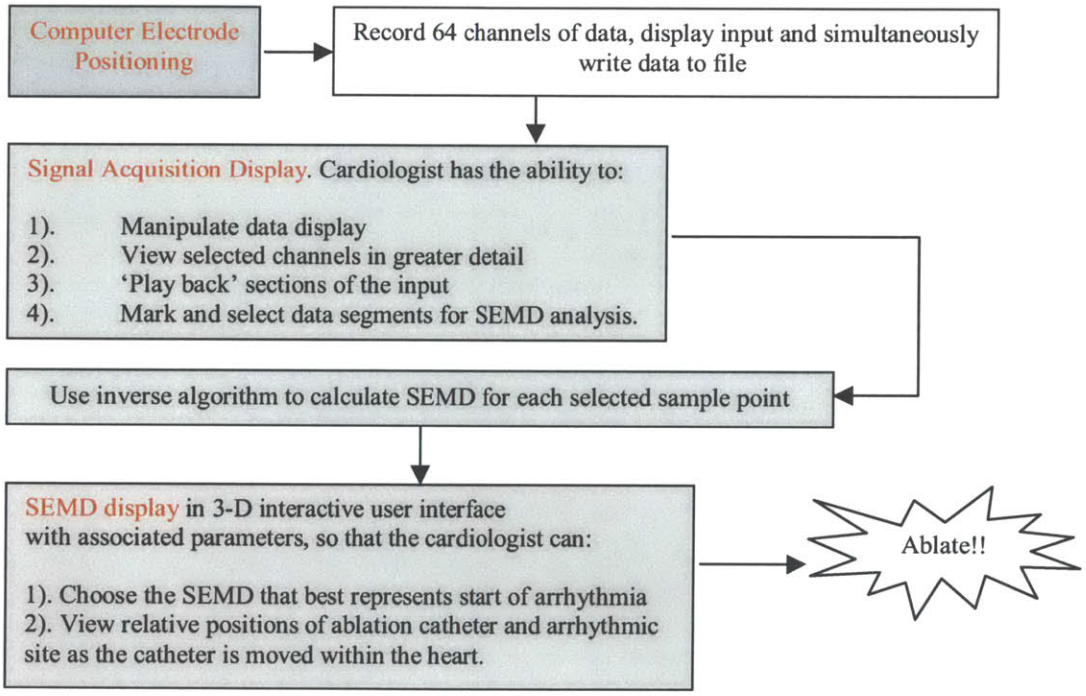


***Figure 4.1**: Flow Chart for User Interface*

4.2 User Interface Specifications

The user-visible software consists of three graphical interfaces displayed in three different windows. All three windows may be displayed simultaneously, although the operator has the option of minimizing windows or moving them to alternative desktops.

I. Computer Electrode positioning

II. Signal acquisition display

III. Single Equivalent Moving Dipole (SEMD) analysis display

I. Computer Electrode Positioning

The cardiologist must place the electrodes in the position assumed by the inverse dipole algorithm. The first graphical interface should permit the operator to view electrode position information and input patient information:

- Operator can select and display the appropriate torso model, with pre-defined electrode positions. This option will help the operator properly align the patient electrodes with the torso model electrodes.
- The torso is presented as a 3D-wireframe model.

II. Data Acquisition display

Once the cardiologist has entered the patient data and correctly positioned the electrodes, the computer will begin recording data. By default, signals are displayed in real-time. Two types of signals are displayed:

1. Standard ECG signals from catheters and body surface leads to use as part of the standard mapping protocol.
2. The individual electrode signals, to verify the quality of the electrode connection

After recording has finished, the operator is able to manually scroll backwards in time through the signals, or view the data as a 'movie'. 10 minutes of data should be accessible in RAM, and the remainder will be stored on the disk. The signal acquisition interface should have the following capabilities:

- Operator is able to manually adjust the display gain for individual channels
- Operator is able to mark intervals between events with the mouse
- Operator is able to move the positions of individual channel displays on the screen to allow him/her to organize signals in a manner relevant to the catheter's position
- Operator is able to select (and print out) a data segment for SEMD analysis

This data segment is passed to the inverse algorithm. The analysis is not done in real time. The SEMD parameters are calculated for each sample in the chosen segment and are then displayed.

III. Single Equivalent Moving Dipole (SEMD) display

There are two phases of operation. Initially, only the sequence of SEMDs related to cardiac excitation is displayed. Once the re-entrant dipole has been chosen and the ablation catheter inserted, the SEMD related to the catheter tip must be displayed also. In both cases, an individual dipole is represented by a three-dimensional ellipsoid (dipole ellipsoid), in conjunction with a dipole vector.

Individual SEMDs are displayed as follows:

- The **position** of the dipole ellipsoid center (and the start of the dipole vector) is the 3-dimensional location of the dipole.
- The **dimensions** of the dipole ellipsoid reflect the three-dimensional positional uncertainty of the dipole – a large diameter ellipsoid is poorly localized, while a small ellipsoid is highly localized.
- The **color** of the dipole ellipsoid indicates the magnitude of the dipole, using a two-color sliding scale – red indicates large magnitude, blue is small magnitude.
- The **orientation** of the dipole vector reflects the orientation of the dipole
- *The length of the dipole vector is constant.* The ellipsoid is semi-transparent, so that the dipole vector can be seen even if the dipole is poorly localized.

All dipoles from the data segment selected in the "Signal Acquisition" interface are initially presented in a 3-D rotatable display that demonstrates the trajectory of the cardiac dipole over that period.

- The data segment is presented as an 'arc' of colored dots, each dot indicating the estimated spatial location of the dipole at that instant. Each dot is color-coded corresponding to the magnitude of the dipole as above (red is strong, blue is weak).
- The cardiologist can step through this trajectory using a button on the interface. Each consecutive dot will expand into the corresponding dipole-ellipsoid and dipole vector, superimposed on the dipole trajectory. This will give the cardiologist an indication of the orientation and positional uncertainty of the dipole. If desirable, the ellipsoid can be hidden leaving only the vector.

- Parameters of the selected dipole are shown numerically on the side of the display.
- An ECG signal from a standard surface lead over the selected data segment is displayed. The time of the currently selected dipole is marked.
- The operator has the option of displaying the surrounding torso model selected during the "Computer Electrode Positioning" phase. (This is only of physiologic use if the correct dimensions of the patient's heart are known. A CT scan of the patient's torso could generate a 3-D image of the heart, which would then be used by this program.)
- The operator can "zoom in" on a section of the trajectory to view dipoles more closely, and return to the normal field of view.

The cardiologist will use the dipole trajectory and ellipsoid/vector display to determine the dipole that best represents the arrhythmic origin (for further discussion, see Section 8). This dipole is then selected and 'saved'.

The second stage of analysis involves the superposition of a "real time" display of the catheter tip with the single cardiac dipole selected above. The update rate of the display may only be once every few seconds (decided by the time required to calculate the catheter dipole parameters), but this should be sufficient since the catheter is moved slowly. The catheter tip dipole is displayed as a small dipole ball/vector.

4.3 Choice of Programming Environment

There were several important requirements concerning software environment:

- Must provide excellent data acquisition support, and wide compatibility with commercially available DAQ devices.
- Must have good graphical capabilities, in 3-D and in real-time.
- Must handle complex numerical calculations, and be able to handle large data matrices.
- Must be programmatically able to provide a simple user interface for the physician.

LabVIEW (National Instruments) interfaced with Matlab (Mathworks) suits this purpose. LabVIEW is a graphical programming language that is perfectly suited to DAQ system user-interface design. Although numerical computation and 3-D visualization are poorly supported, it

is easily interfaced with Matlab, a language specifically designed to handle large matrices and mathematical equations. Matlab also has excellent 3-D image rendering capabilities.

Consequently, the Computer Electrode Positioning and Data Acquisition displays are implemented with LabVIEW. Matlab is then used to calculate the dipole parameters for the selected data segment and display these in a 3-D graphical user interface.

4.4 Data Sources

Since the goal of this project was to implement only the software component of the RFA device, pre-collected data was used to test the software. This data was obtained from several sources.

60-channel swine ventricular pacing data was pre-recorded at a sample-rate of 500 Hz by Dr. Antonis Armoundas, and saved as a series of '.d' files.

63-channel BSPM data, recorded at 250 Hz, was kindly donated by the research group of Dr. Didier Klug from the Sacre-Coeur Hospital in Montreal. This data is single-beat, slow ventricular tachycardia data, recorded in a series of TIMI (Thrombolysis in Myocardial Infarction) studies. The data is saved in ASCII '.asc' format.

Lastly, to test signal-parameter estimation techniques (see Section 6), single-channel ventricular tachycardia and sinus rhythm data was obtained from the MIT-BIH Arrhythmia Database and the Creighton University Ventricular Tachyarrhythmia Database, both available through Physionet. These sources provided examples of signals with different rates and morphologies of VT, excessive baseline wander, and high-frequency noise. This data was recorded at a sample rate of 250 Hz, and saved in text '.txt' format.

5. Signal Acquisition and Display

5.1 Patient Information Entry

The initial display shown in Figure 8 allows the cardiologist to view the correct electrode positioning and enter pertinent patient information. The cardiologist first selects the patient's sex/type (male, female, or pig for research purposes) from the pulldown menu. The program uses this information to decide which wireframe model to display when the cardiologist clicks the button "View Electrode Positioning". Figure 5.1 shows the male BSPM electrode configuration.

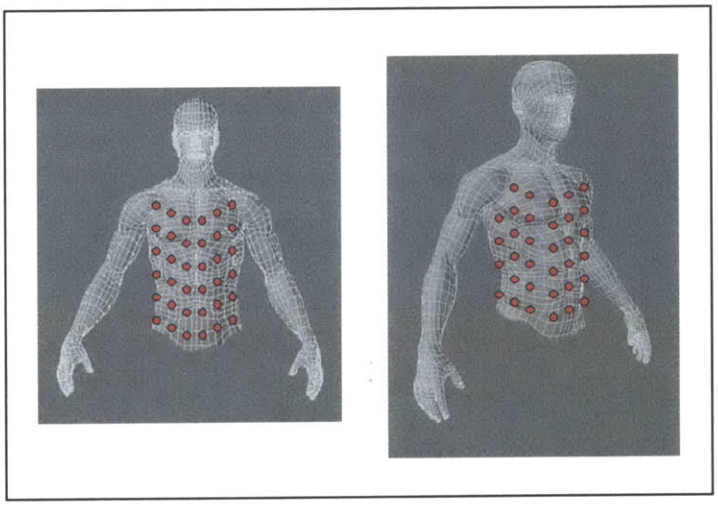


***Figure 5.1**: Male Electrode Positioning Wireframe Torso Model*

As shown in Figure 5.2, patient data entered in the two upper text boxes appears, along with the patient type, in the text box labeled "Final Subject Information". Once the cardiologist has finalized and checked the patient's information, and has positioned the electrodes as indicated by the wireframe model, he/she presses the button marked "OK" to begin data acquisition.

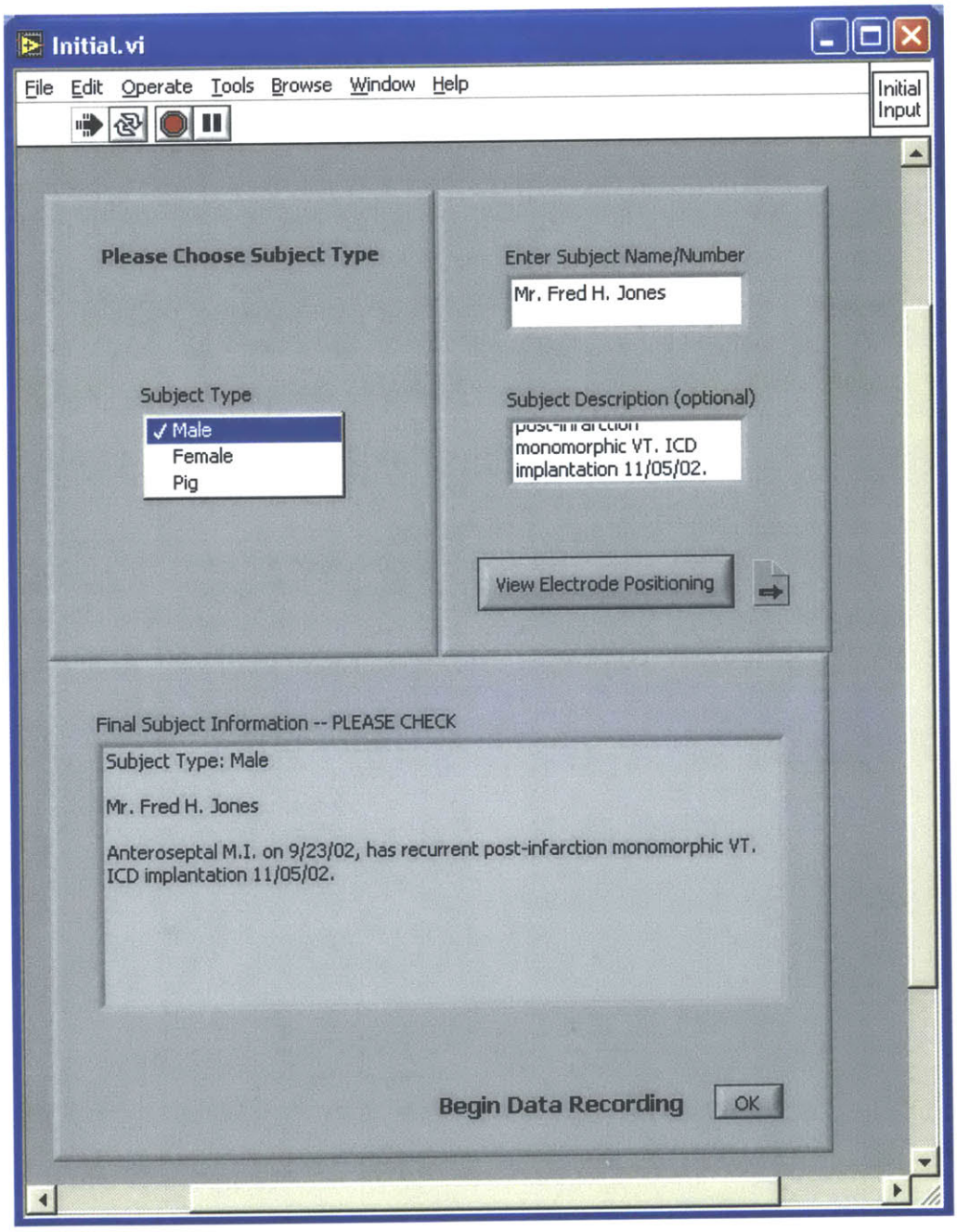


***Figure 5.2**: Patient Data Entry entry*

5.2 The Data Acquisition Interface

5.2.1 Signal Acquisition and Real-Time Display

The cardiologist is initially presented with a prompt to enter the name of the file in which the data should be stored, the number of electrodes that are being used and the sampling frequency (see Figure 5.3). After he/she clicks OK, the device begins recording to the specified file name. A program to interface with the ECG device has not been developed as part of this project, since its design is dependent upon the final hardware chosen; also, it is often provided with commercially available devices. However, LabVIEW has excellent data acquisition capabilities therefore the design of such a program in the future would not be complicated.

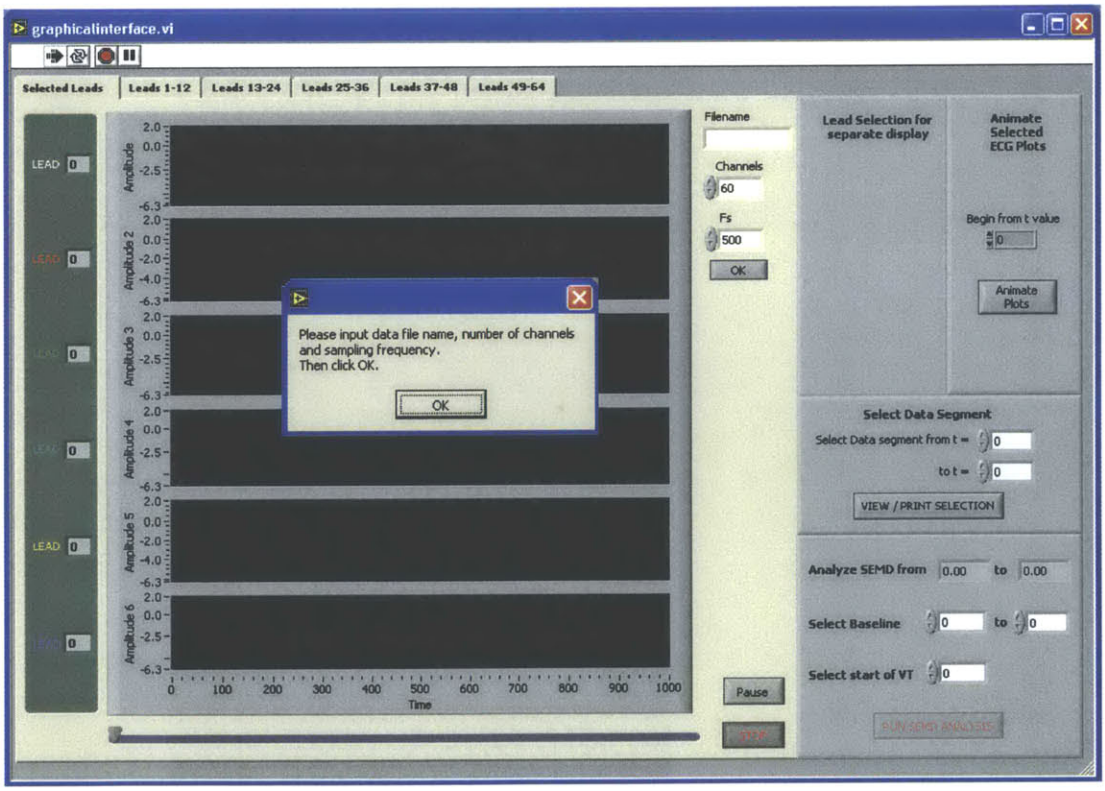


***Figure 5.3**: User prompt and initial screen before recording begins.*

Signal data from all 64 channels is simultaneously written to memory and displayed. Since pre-recorded data is used in this project, scrolling the data across the display as it is read from the stored file mimics this process. As shown in Figure 5.4, the data is displayed in a series of tabbed pages; this arrangement works best for simultaneously displaying many sets of high-resolution data in a manageable way.

While recording, the data is displayed in real-time on tabbed-pages 2 through 6, divided according to the heading at the top of the page (e.g. 'Leads 1-12'). The limits of the Y-axes (in mV) on each page are adapted to the current data set, and are coupled to the maximum and minimum electrode voltage displayed on that page. The markings on the X-axes indicate the sample number. The program also dynamically sets the X-axis limits, so that two seconds of data are always displayed on the screen. For instance, for a sampling frequency of 250 Hz, 500 consecutive samples can be seen on the screen at a time. For ease of use, the signal legend and the signal itself are color-coded to match.

This tabbed display system allows the software to be easily extended if additional input channels are required. The inverse problem estimation improves with the number of inputs (the chi-square value of the estimated dipole decreases with addition of low-noise channels). Therefore, ease of extension will be a valuable feature when testing the completed ablation device.

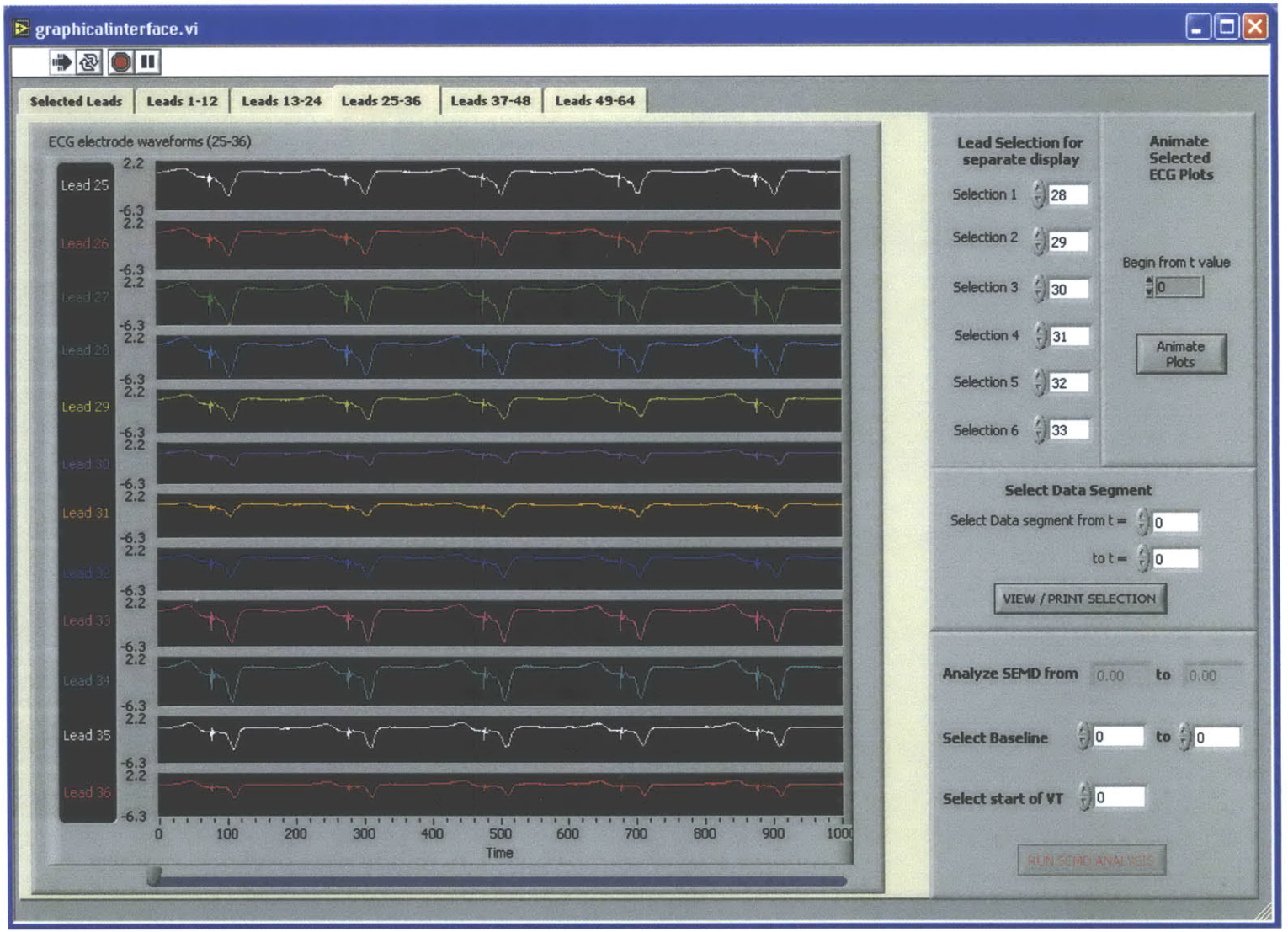


***Figure 5.4**: Recording and simultaneous display of signal data. Leads 25-36 are shown.*

5.2.2 Selecting a data segment

Once the data has finished recording, the user can scroll through the data using the scroll bars at the bottom of each page. The first tabbed page is used to display signals selected by the user that he/she would like to view at a higher display gain; for instance those with important characteristics, or that have a low signal-to-noise ratio. This page is initialized to display the strongest signals out of the 64, which are invariably found in electrodes towards the middle of the torso. Using the digital controls in the top right panel of the interface, the user can change the selection and order of the channels as required (see Figure 5.5). The number of each selected channel is displayed next to its graph, and the scroll bar at the bottom can be used to scroll backwards and forwards in time.

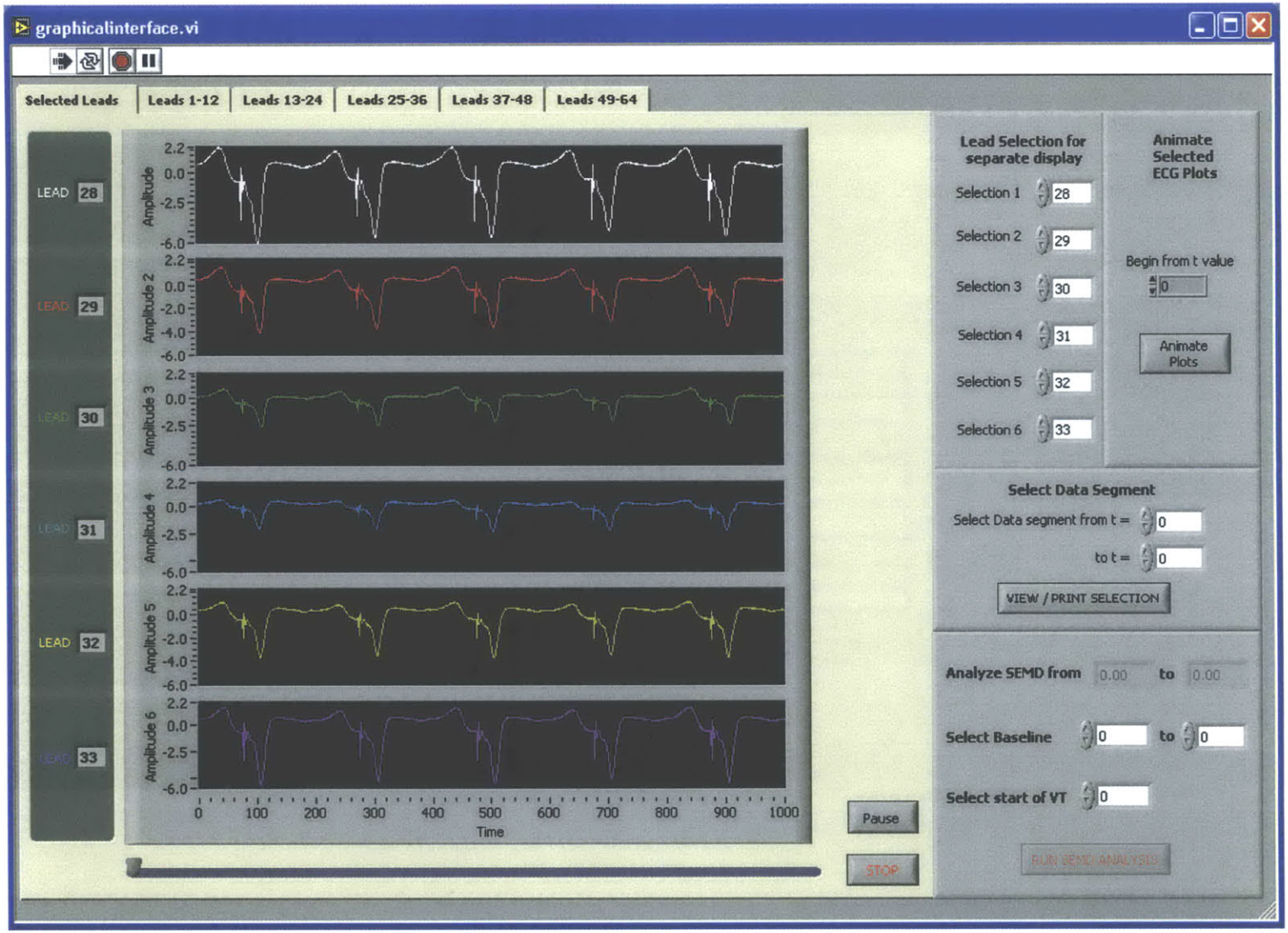


***Figure 5.5**: Display of selected leads*

The user can also view these signals as a movie. This is a useful feature if the cardiologist needs to inspect long data segments, since the signals will scroll automatically until paused. To do this, the cardiologist enters the time at which he/she wishes to begin the animation, and clicks on the 'Animate Plots' button on the right. The Animation Display is shown in Figure 5.6.

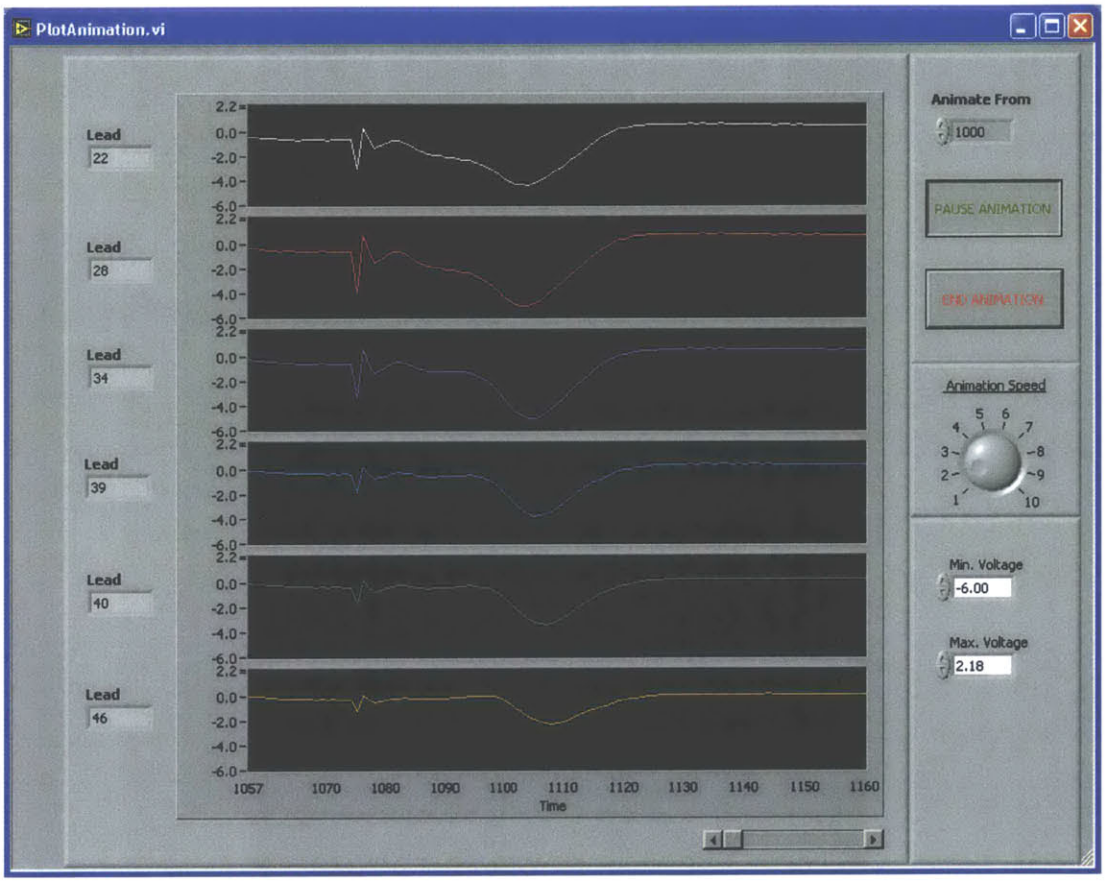


***Figure 5.6**: Animation of selected leads*

The signals are displayed with their electrode number and scroll from left to right at a speed determined by the setting of the 'Animation Speed' dial. Control of the scroll speed is useful if, for instance, the patient is defibrillated after a period of tachycardia and the VT is then re-induced several minutes later; the cardiologist can scan quickly through the intervening period of sinus rhythm. To pause the animation and examine an interesting feature, the cardiologist presses the 'Pause Animation' button, and to resume the button is pressed again. The 'End Animation' button will close the window automatically and return the user to the main display.

By scrolling manually and using the animation feature the cardiologist will identify an arrhythmic beat, or a segment whose characteristics require closer examination. To view this section of the data more closely, print it, or to select a segment for SEMD analysis, the cardiologist enters the data range of interest into the text boxes in the panel marked 'Select Data Segment'. Clicking the button 'View/Print Segment' then opens up a new window displaying only the range of interest, as shown in Figure 5.7.

The signals shown in Figure 5.7 are from pre-recorded swine paced data. The features seen include the tail-end of the T-wave from the last beat (until approximately t = 1862), a pacing spike artifact (at t = 1876), and the QRS complex due to the pacing stimulus (starting at around t = 1895). We wish to image the dipole during, and especially at the beginning of, the QRS complex. Earliest activation of the ventricle is associated with a downward-turn of the lead potentials. Also, after the peak of the QRS the depolarization wavefront is too diffuse for its dipole representation to have any physical meaning. Therefore we choose the limits of SEMD analysis just before and just after the down-turn and peak of the QRS respectively.

To delineate the desired data range, the cardiologist first presses the button marked 'Display Cursors'. Two cursors appear whose numerical values are displayed in the text boxes marked 'Cursor 1' and 'Cursor 2'. To print the window, the cardiologist clicks the 'Print Display' button. To select the range demarcated by the two cursors for SEMD analysis the user presses 'Final Values Chosen'. This returns the user to the main display and passes the cursor values to the "Analyze SEMD" panel on the lower-right, where they are displayed in the text boxes. If the user did not select a data segment in the View/Print display, these boxes are inactive and appear blank (as in Figure 5.5). When the values shown are final, the cardiologist clicks the button "Run SEMD Analysis" to calculate dipole parameters for all samples within the data range (see Figure 5.8).

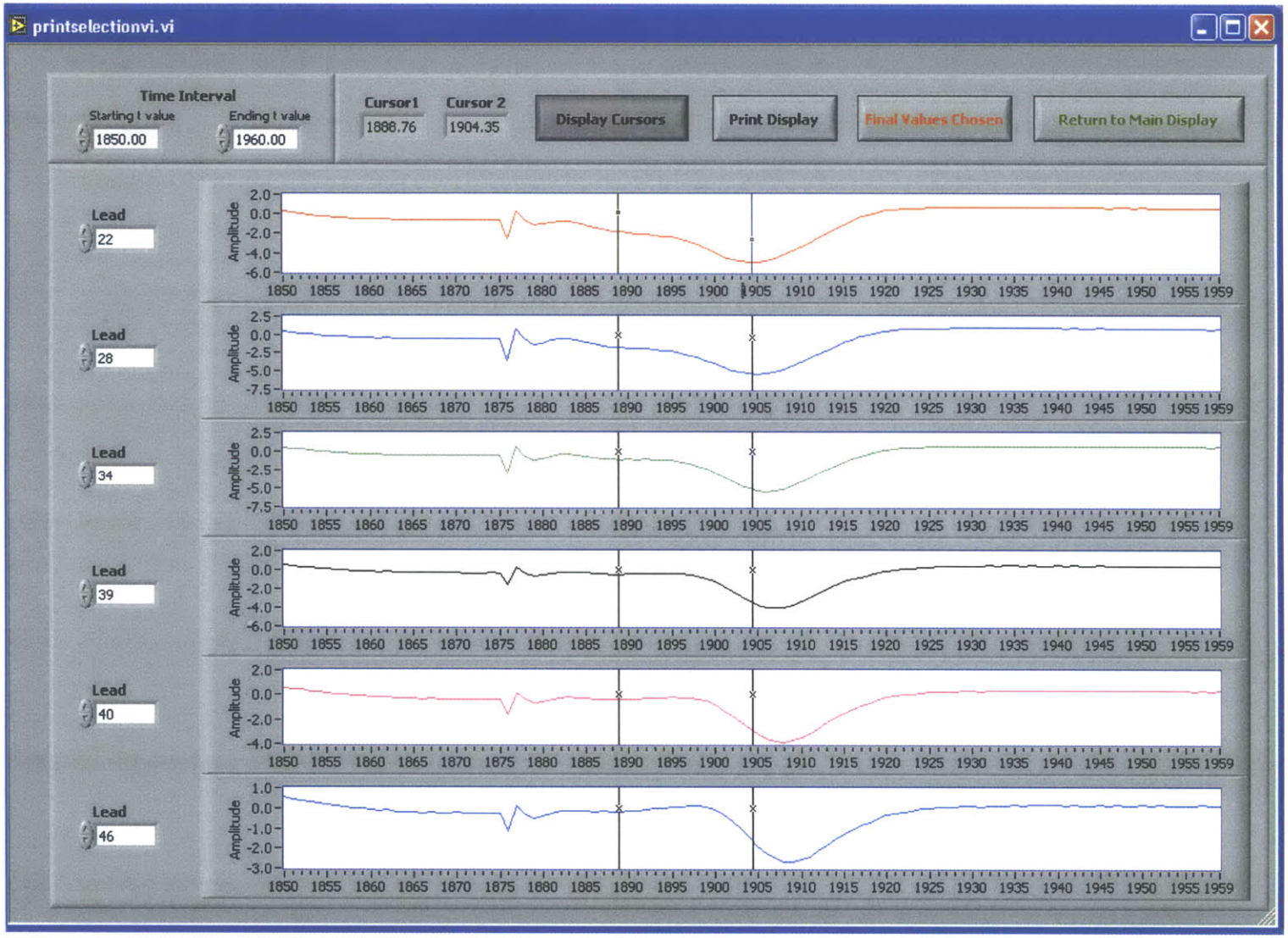


***Figure 5.7**: View/Print Data Segment, and select a segment for SEMD analysis*

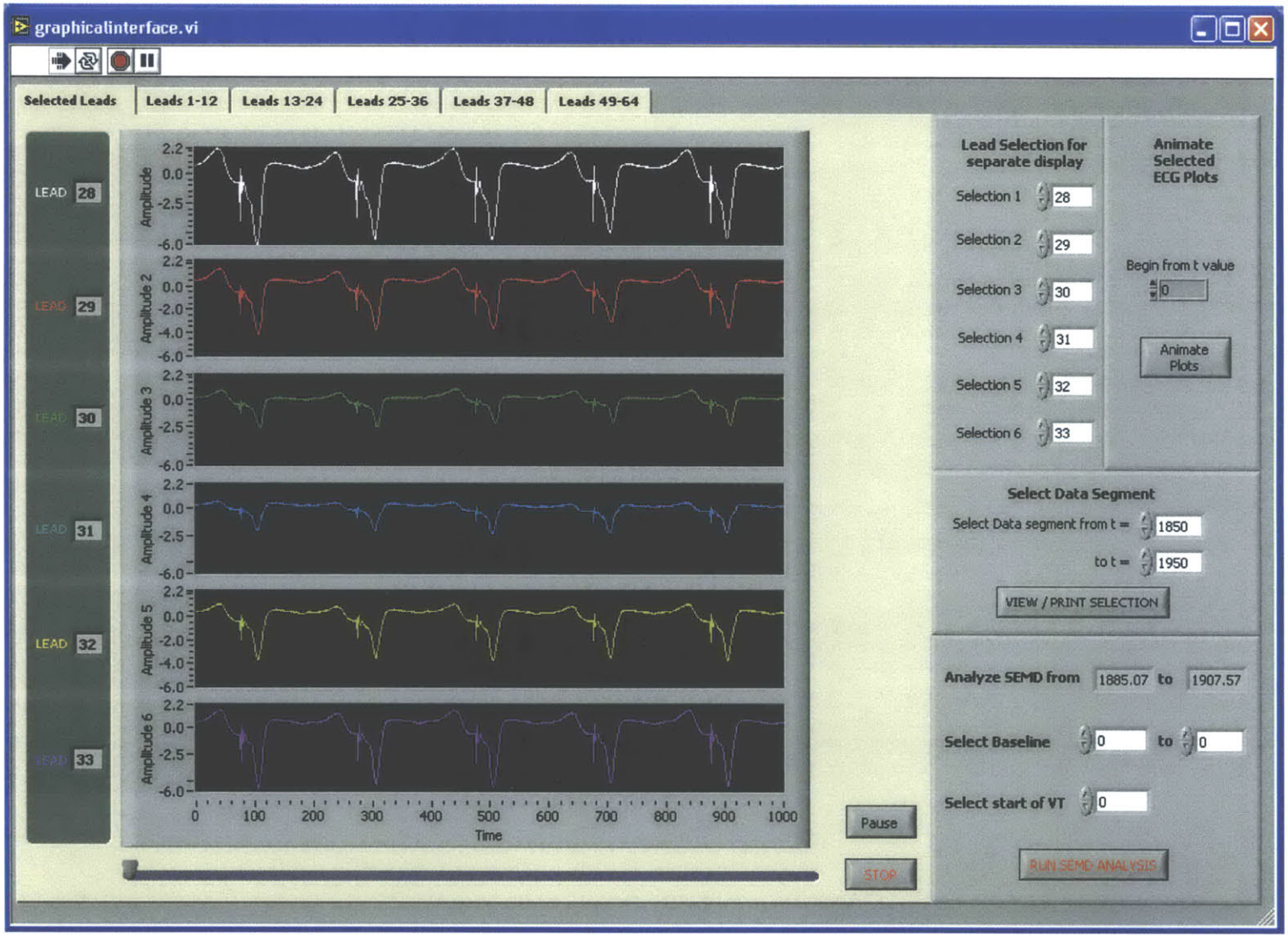


***Figure 5.8**: After selection of a final data range, the values are displayed and the user can select to run SEMD analysis on this segment.*

6. Signal Parameter and Dipole Estimation

6.1 Matlab Program Structure for Dipole Parameter Calculation

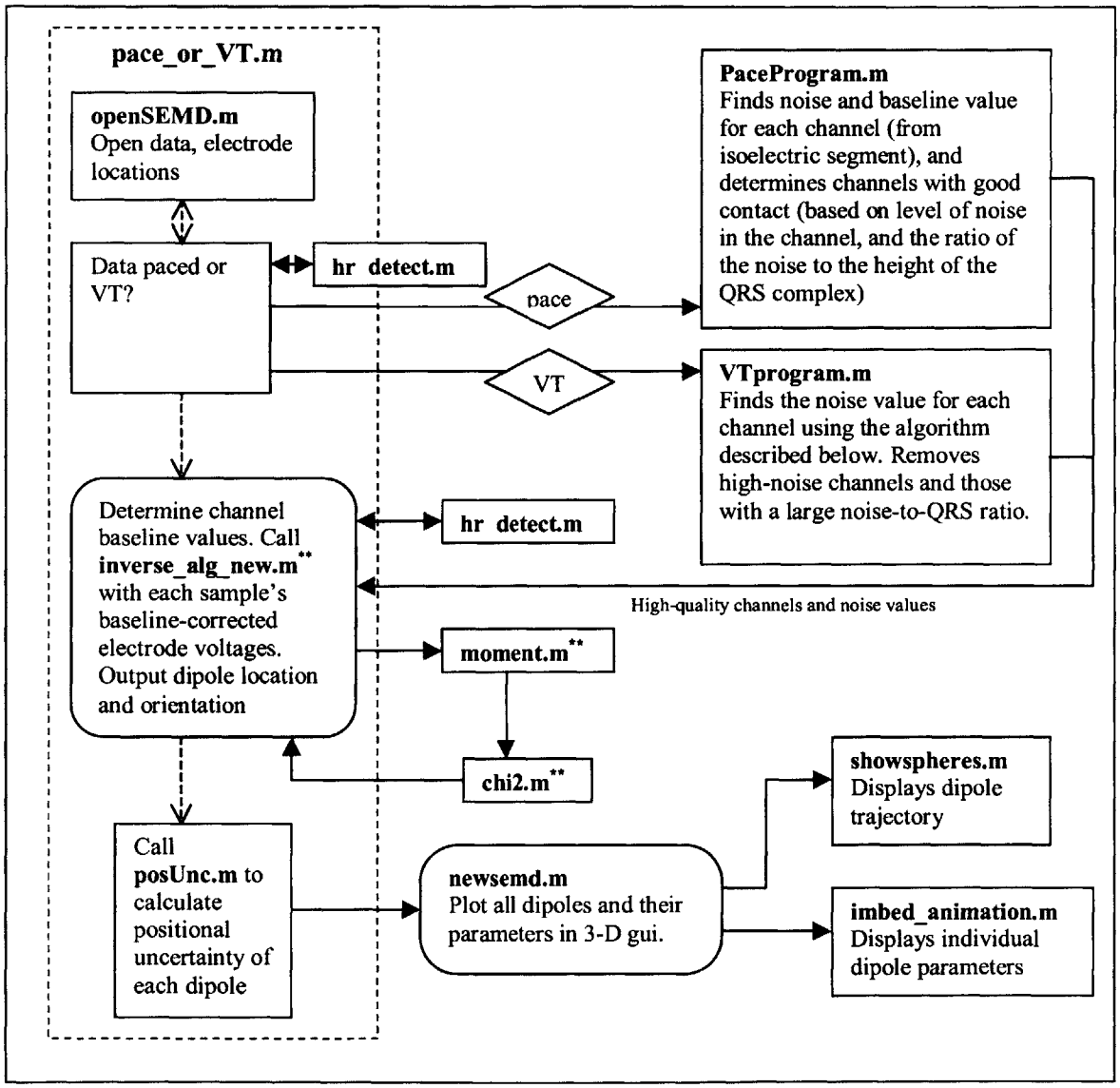


Figure 6.1: *Flow Chart of dipole parameter estimation and display. All program code is displayed in the Appendix.*

***Programs written by Tamara Williams.*

Using LabView's built-in Matlab Script Box, the values and data range specified by the user in the Data Acquisition Interface are passed to the program *pace_or_VT.m* (see Appendix). This initiates the cascade of programs shown in Figure 6.1 that first calculate and then display the dipoles for each sample in the selected data segment. All Matlab programs have been included in the Appendix. Tamara Williams, a PhD candidate in Electrical Engineering at MIT, implemented the basic inverse algorithm as part of her thesis work (in Figure 6.1, these programs are indicated by an asterisk). Her program code is included in the Appendix with her permission.

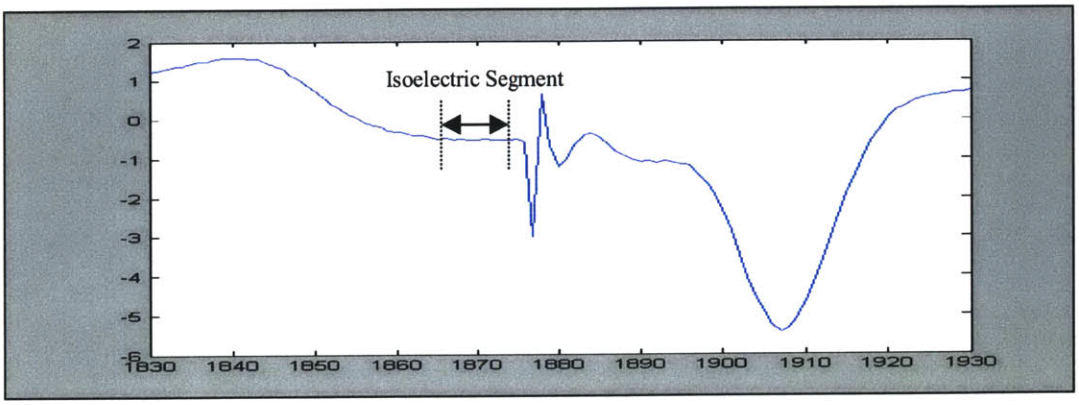
6.2 Signal Parameter Estimation

6.2.1 Baseline Estimation

Baseline drift in an ECG lead is usually caused by respiration or movement of the subject.[36] The severity of baseline wander is different in each lead, and if two measurements are made far apart in time each lead's baseline may have changed significantly. Therefore baseline drift can significantly alter the dipole parameters found by the inverse algorithm. This is an important problem to consider if we are to match the catheter tip and reentrant circuit. Only by accurately baseline-correcting the lead values in the calculation of both dipoles will the inverse algorithm calculate the same dipole parameters.

If the baseline is assumed to be continuously changing its estimation is an inherently ill-posed problem, since 64 measurements are made while 6 dipole parameters plus 64 baselines must be found. There will always be six more variables than equations describing them; therefore it is impossible to devise an algorithm that will accurately estimate all parameters. If the baseline is assumed to be constant over a short time period, the problem becomes solvable. However, algorithms estimating this constant baseline by minimizing chi-square or RNMSE over any time period failed to find the correct value for data in which the baseline was known. Therefore, two approaches were devised, one for paced data in which the baseline can be observed empirically from particular time segments (see below), and another for VT data in which a signal-processing method must be employed.

- ***Paced Data***



The ECG tracing above is taken from paced data and clearly shows the isoelectric period prior to the pacing spike. The T-wave has finished, the heart is repolarized, and cardiac electrical activity is essentially zero. If there were no baseline offset, the potential during this time segment would be zero. Therefore, the DC value of the signal during the isoelectric period is an excellent approximation of the current baseline. By calculating local gradients, the algorithm (a subsection of *Paceprogram.m*) finds the most recent isoelectric period before the selected data segment. The mean isoelectric signal value in each lead is then calculated. The baseline is assumed to remain approximately constant over the timescale of a beat.

- ***VT data***

The isoelectric segment does not exist during ventricular tachycardia, since the next depolarization begins before electrical activity from the prior beat has dissipated. Several signal-filtering approaches have been described to detect baseline. Morphological filtering, a technique based on set operations, appears to be very effective in sinus rhythm [37] while wavelet transforms are highly suited to identifying regular patterns within a signal and therefore removing baseline wander [38]. Adaptive signal filtering has also been employed. The American Heart Association (AHA) recommends a cutoff frequency of 0.67 Hz for detecting DC and slowly-varying baseline wander during sinus rhythm (0.8 Hz). Cut-off frequencies above this value may cause distortion by interfering with the low-frequency S-T segments, especially at slow heart rates. However, the AHA also states that if linear phase is preserved the cut-off frequency can be chosen taking the fundamental frequency of the heart rate or lower. [39]

- *Filtering the cardiac signal to detect baseline wander*

A 4th-order lowpass Butterworth filter is a good choice for baseline detection, since its maximally flat passband will ensure minimal distortion of the baseline signal. However, this filter has a slow decay, therefore the signal may be distorted if the cutoff frequency is not sufficiently low. [40] The inbuilt Matlab function *filtfilt.m* will use pre-defined 4th-order Butterworth filter parameters to filter in both the forward and reverse directions, causing zero phase distortion and producing an 8th order roll-off.

To determine the appropriate cutoff frequency for baseline detection, the power spectra of several single-channel VT ECG signals with significant baseline wander were plotted. As shown in Figure 6.2, the power spectra characteristically show two low-frequency 'spikes'. The higher-frequency spike corresponds to the heart rate. The lower spike (below 1.5Hz) is due to baseline wander. For this data set, a cutoff frequency around 1.5Hz will capture the baseline frequency components.

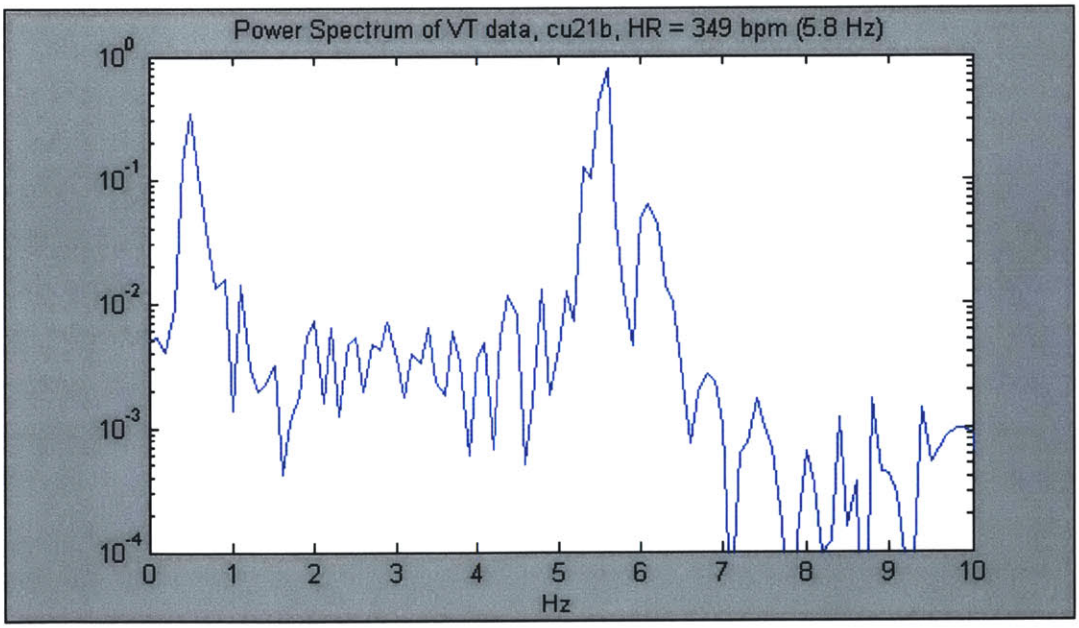


***Figure 6.2**: Power Spectrum of VT data, showing two significant low-frequency components: the first 'spike' at 0.5 Hz is due to baseline wander, while the second at 5.8 Hz corresponds to the heart rate.*

As the heart rate decreases, the cutoff frequency must be reduced to ensure that low-frequency cardiac signal components are not removed. The appropriate cutoff frequencies were determined for many data sets, with heart rates ranging from 169 to 340 beats per minute. An equation for the linear relationship between heart rate and cutoff frequency was then found using a least-squares approximation.

- *Correcting the DC value of the estimated baseline*

Low-pass filtering removes not only the DC offset and low-frequency baseline components but also the average value of the signal being filtered. Since there is no guarantee that the average value in a lead over one cycle equals zero, the DC value of the baseline estimate is distorted. For instance, if the filtering approach is used on sinus rhythm or paced data, the estimated baseline is found to accurately *track* changes in the isoelectric segment but with an offset from the true value. This offset can be calculated by comparing the value of the estimated and actual baselines during an isoelectric segment, or by calculating the average value of a sinus beat. The true baseline is then equal to the estimate obtained by filtering minus the offset.

The same problem occurs during VT, although in this case there is no isoelectric segment and the average value of the signal is unknown. Therefore the true VT baseline cannot be directly calculated. To solve this problem, we must make several assumptions. Firstly, we may assume that the patient begins the procedure in sinus rhythm and that several beats of sinus rhythm will be recorded before VT is induced. Secondly, baseline drift is a property of the electrodes and patient movement and is independent of the cardiac signal itself. Therefore, we may assume that a cutoff frequency that satisfactorily captures baseline wander during sinus rhythm will likewise capture baseline wander during VT. Lastly, since changes in the baseline occur over time periods significantly longer than one heart beat, it can be approximated as constant for intervals of a few seconds.

With these assumptions, the VT baseline estimate may be corrected. The cardiologist records several cycles of sinus rhythm or paced data, and induces VT using rapid pacing. The electrode recordings are examined as before in the 'Data Acquisition' interface, and a segment selected for SEMD analysis. In addition, the cardiologist enters the time segment of the last isoelectric segment prior to VT induction, and the time at which the VT morphology appears. With this information, the computer compares the value of the estimated baseline at the start of VT with the

value of the actual baseline during the last isoelectric segment. Given the above assumptions, the difference can be assumed to be the offset due to the average of the VT signal. This offset is then subtracted from the estimate obtained by filtering to yield an accurate baseline estimation.

- Implementing the VT baseline algorithm

After choosing a VT data segment in the Data Acquisition Interface, the cardiologist enters the times marking the beginning and end of the last isoelectric segment before VT induction, and the time of the start of VT (see Figure 13). These values are passed to the Matlab program *pace_or_VT.m.*

The program *hr_detect.m* (see Appendix) provides an accurate measurement of the heart rate prior to VT induction. *pace_or_VT.m* uses the least-squares equation described above to select a suitable cutoff frequency based on this heart rate. It then implements the appropriate low-pass fourth-order Butterworth filter and retrieves a slow-varying signal that mimics the baseline but with an incorrect DC offset. The program corrects this baseline estimate using the algorithm described above and the times entered in the Data Acquisition Interface.

- Reducing Edge Effects

The baseline is calculated for the entire signal rather than the selected data segment only, to reduce the impact of "edge effects" within the period of interest. Edge effects are a well-documented effect of low-pass filtering. The convolution kernel 'assumes' that the beginning and end of the signal represent sudden increases from zero; these sharp edges have a high frequency component that, when removed by a low-pass filter, distort the edges of the signal over many samples. To minimize these effects, ECG data can be windowed using an extended cosine bell function. [41] In *pace_or_VT.m*, this is done by multiplying the initial and final hundred samples by a quarter-cycle of raised cosine.

- Accuracy of the Algorithm

Single-channel sinus rhythm and VT data was obtained from the Creighton University Ventricular Arrhythmia Database. The above algorithm was implemented on all data sets and the results were observed to determine if the algorithm was:

1). Flexible. The algorithm does not need to be modified by the user for different data sets, since it accurately selects the required cutoff frequency from the calculated heart rate (no components of the cardiac signal are detectable in the baseline, for any data set).

2). Accurate. ECG data was found in the CU database with an isoelectric segment detectable before a segment of ventricular tachycardia. The VT spontaneously resolves with a isoelectic period just prior to the resumption of single ectopic beats. The algorithm was implemented on this segment of VT, using the isoelectric segment prior to the start of VT (indicated by the first set of arrows) to correct the baseline estimate. The resulting baseline is shown in Figure 6.3a. The isoelectric segment occuring at the cessation of VT (indicated by the second set of arrows) is shown in close-up in Figure 6.3b with its associated baseline approximation. Although the signal is extremely noisy in this region (the standard deviation of the signal is 0.0442 mV), the baseline estimate provides an excellent approximation: the mean value of the signal between samples 9300 and 9390 is 0.0966 mV, while the mean estimated baseline is 0.0766 mV.

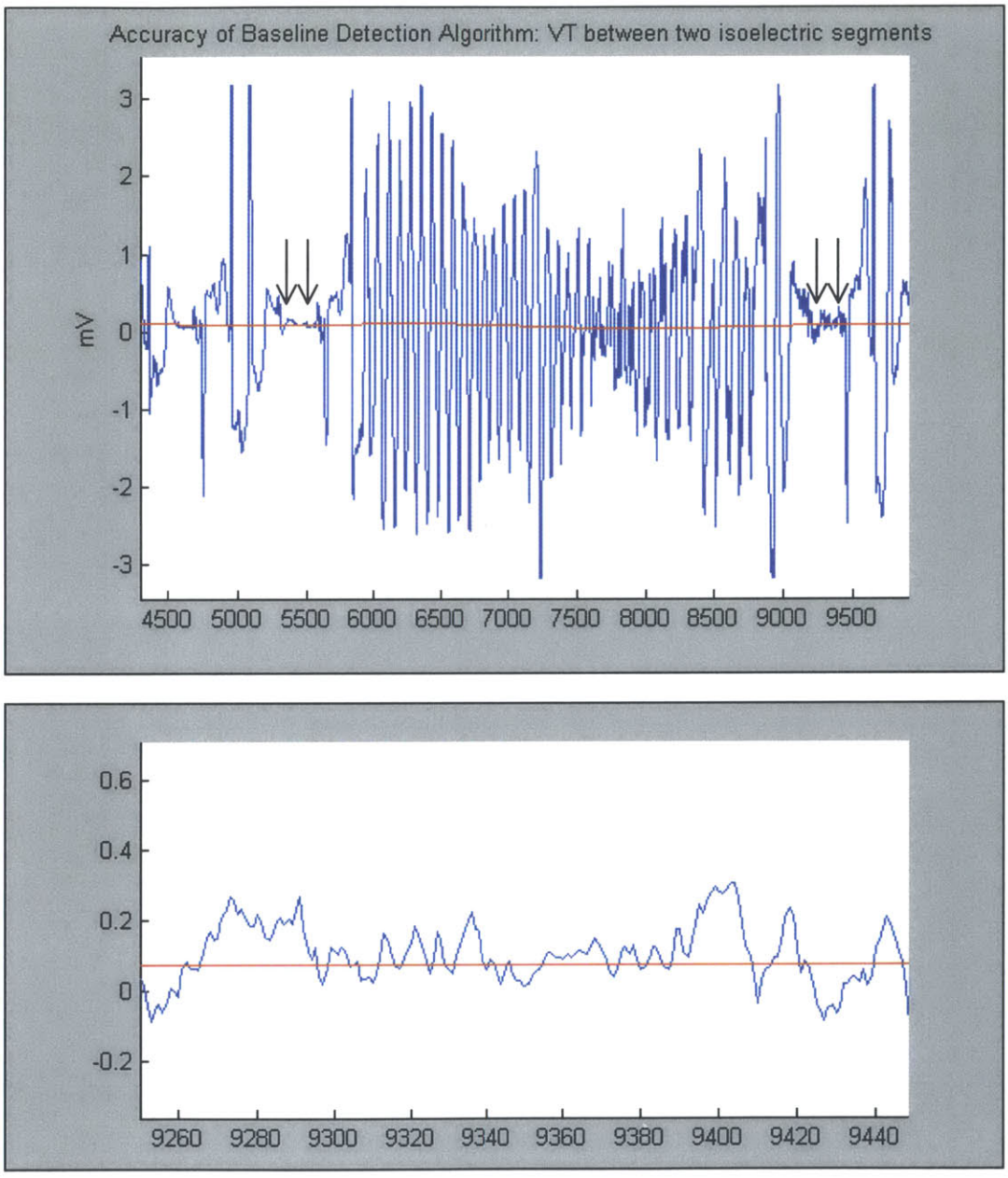


***Figure 6.3**: a) Baseline Estimation (in red) for segment of ventricular tachycardia between two isoelectric segments. b). Close-up of post-VT isoelectric segment*

6.2.2 High-frequency Noise Estimation

The calculation of chi-square requires the standard deviation of the noise in each channel, σ^i (see Section 2.2). High-frequency noise sources include: powerline interference (60 Hz) which is usually filtered by the ECG amplifier itself using a notch filter; EMG artifact from muscle contraction in the μV-range (modeled as random noise); instrumentation noise in the 30 μV range also modeled as random noise, and radio electrosurgical noise (100 kHz – 1 MHz). VT and Paced data require two different noise-estimation algorithms, as described below.

- **Paced Data**

A pacing spike 'impulse' has a very short duration (approximately 8 ms), and therefore has a very broad spectrum in the frequency domain. Therefore high-pass filtering of paced data fails to separate the signal from high-frequency noise. However, during the isoelectric period small perturbations in the cardiac signal are due to noise. The algorithm, a subsection of *Paceprogram.m*, calculates the standard deviation of the isoelectric segment to determine the value of σ^i for each channel.

- **VT data**

The isoelectric segment does not exist during ventricular tachycardia, since the next depolarization wave begins before the electrical activity from the prior beat has dissipated. However, with no pacing artifact present, a noise estimate may now be obtained using a high-pass filter. A plot of the power spectrum of single-beat VT data (Figure 6.4a) shows no significant signal above approximately 100 Hz. A high-pass filter with a cutoff frequency above this value yields the noise-signal shown in Figure 6.4b. *VTprogram.m* implements this filter and calculates the standard deviation of the remaining signal over the selected time segment. The noise estimate for the channel shown in Figure 6.4b is 0.0087 mV, within the range expected for an ECG conducted in a medical setting (0.001 mV – 0.1 mV). Noise estimates of other VT data also yield values within the expected range.

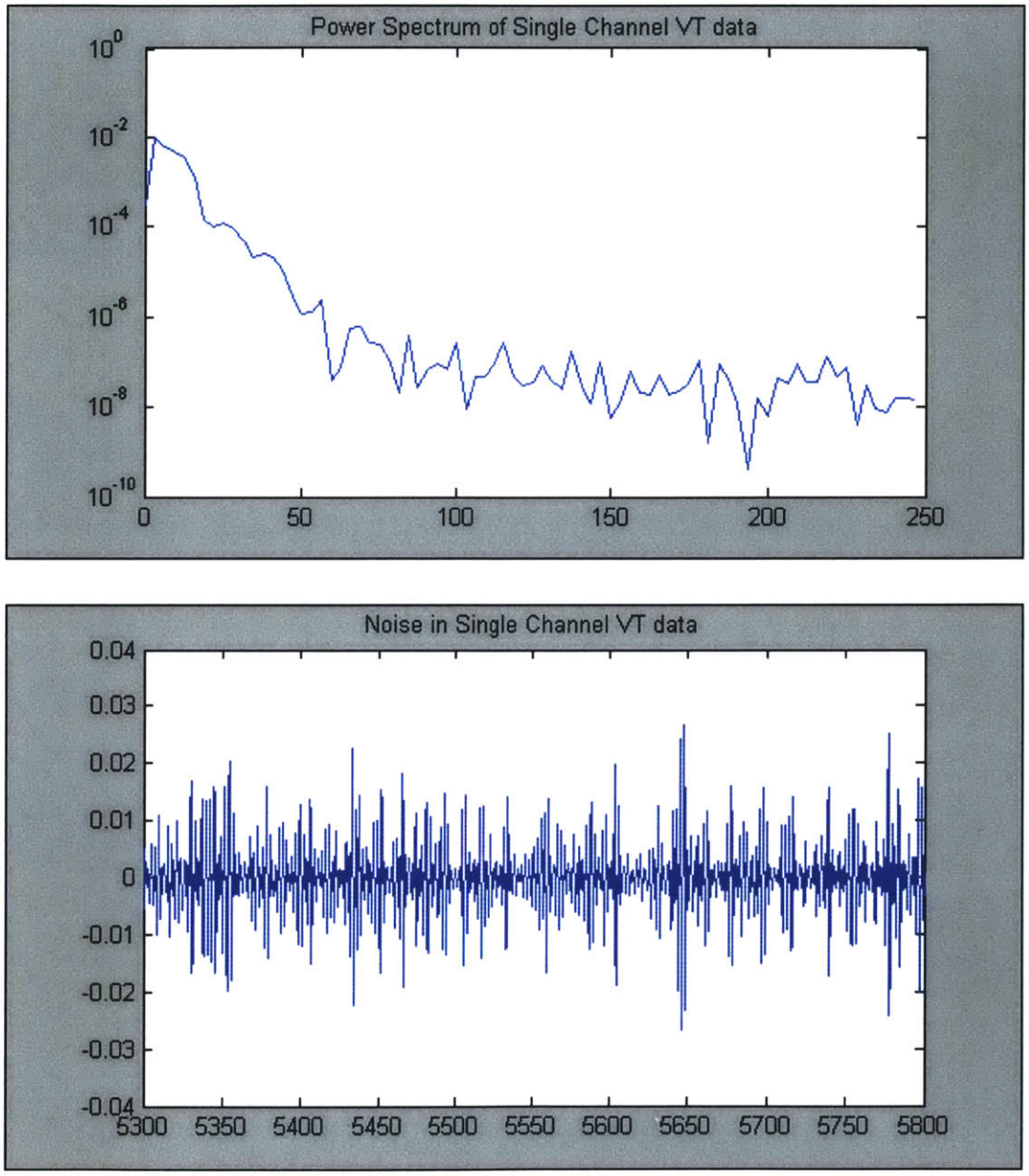


***Figure 6.4**: a). Power Spectrum of single-channel VT data, b). Result after high-pass filtering the same VT data, leaving the noisy part of the signal*

6.2.3 Selecting high-quality channels

Leads with poor electrode-patient contact or hardware problems are characterized by high noise values and poor signal-to-noise ratios (see Figure B1 in the Appendix). The programs *Paceprogram.m* and *VTprogram.m* eliminate channels with σ_i greater than a value σ_C, determined by analyzing the effect on χ^2. A very low value of σ_C eradicates many channels, resulting in too few measurements for accurate dipole localization. On the other hand, a very high σ_C includes signals with significant noise in the dipole estimation, increasing the error in the calculation.

The optimal value of σ_C was determined by examining the effect of σ_C on the value of χ^2. The value of σ_C that minimized χ^2 was calculated to be approximately 0.12. This value was found to remove all channels that a cardiologist would manually remove by visual inspection and retain those that had defined signal morphology. The paced data has a far higher level of noise than the VT data sets that were used in this project (perhaps because the former was experimental rather than taken in a medical setting), therefore eradication of noisy channels was far more important for the paced data set.

An estimate of SNR was also found for each channel by comparing the peak-to-peak QRS signal with the noise estimation. A minimum SNR of 2 was chosen as the condition for a channel to be retained; from visual inspection, channels with no discernible signal had an SNR much smaller than 2, while all 'useful' channels had an SNR far greater than 2.

6.3 Dipole Estimation

For each sample in the selected data segment, the program *inverse_alg_new.m* is provided with the baseline-corrected lead voltages, noise values, electrode locations and torso dimensions. The parameters for the dipole at that time point are returned, including the location and moment in three dimensions, and the 'quality of fit' (the chi-square value for that dipole).

6.4 Dipole Positional Uncertainty (See Appendix, function *posUnc.m*)

Also of interest to the cardiologist is the 'positional uncertainty' of the dipole. This reflects the standard deviation of the estimated dipole location given the standard deviation of the error in the

in each lead (proportional to the square of the measured voltage minus the estimated voltage). If the magnitude of the cardiac signal were of the same order as the noise, or the wave of depolarization too diffuse, we would expect the standard deviation of the error in each lead to be significant and the dipole to have a very high positional uncertainty. A dipole with a high positional uncertainty is of limited use.

To find a dipole's positional uncertainty, an algorithm must be found to relate the standard deviation of the error in each lead, σ_v, to the standard deviation of the dipole parameters. An unbiased estimate of σ^2_v (s^2) is given by the error or residual mean square:

$$s^2 = \frac{SSE}{n-k-1} \quad where\ SSE = \sum_{i=1}^{n} e_i^{\,2} = \sum_{i=1}^{n} (V^i{}_m - V^i{}_e)^2$$

Next, the set of six dipole parameters (x, y, z, p_x, p_y, p_z) are represented as a vector $\mathbf{x}$; therefore, the body-surface electrode voltages can be expressed as $\mathbf{v} = f(\mathbf{x})$. To find the value of $\mathbf{v}$ as a result of small perturbations of $\mathbf{x}$ around a value $\mathbf{x}_0$, we take the Taylor expansion of this expression:

$$\mathbf{v} = \mathbf{v}_0 + \Delta\mathbf{v} = f(\mathbf{x})|_{x0} + \nabla f(\mathbf{x})|_{x0} * \Delta\mathbf{x}$$

The addition of noise introduces an error vector $\boldsymbol{\varepsilon}$. Therefore:

$$\Delta\mathbf{v} = (\nabla f(\mathbf{x})|_{x0} * \Delta\mathbf{x}) + \boldsymbol{\varepsilon} = (\mathbf{A}*\Delta\mathbf{x}) + \boldsymbol{\varepsilon}$$

To find σ_x, the standard deviation in the location of the dipole, the matrix $\mathbf{A}$ that describes the mapping from $\Delta\mathbf{x}$ to $\Delta\mathbf{v}$ must be developed. Since the unbounded forward model that describes the analytical relationship between dipole parameters and lead voltages is known, we take the gradient of this function. The resulting vectors for each dipole parameter, $\delta\mathbf{v}_i/\delta\mathbf{x}_j$, are inserted into $\mathbf{A}$ as column vectors. To invert the matrix $\mathbf{A}$ and map instead from $\Delta\mathbf{v}_i$ to $\Delta\mathbf{x}$ requires a least-squares approximation:

$$\Delta\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\Delta\mathbf{v}_i$$

Since $(\mathbf{A}^T\mathbf{A})^{-1}$ represents the variance-covariance matrix of $\mathbf{x}$, the variances of the lead voltages, σ^2_v, and the variances of the dipole parameters, σ^2_x, are related by:

$$\sigma^2_x = \text{diag}(\mathbf{A}^T\mathbf{A})^{-1} * \sigma^2_v$$

Using these expressions, σ^2_x can be calculated. The square-roots of the first three elements of this vector, σ_x, σ_y and σ_z, represent the positional uncertainty of the dipole.

7. Dipole Visualization

7.1 Structure of dipole visualization software

The dipole parameter calculations in Section 6 are made without intervention by the cardiologist. However, each dipole calculation takes at least 2.5 seconds, therefore there is considerable delay before the Dipole Display Interface appears. To reduce this delay, when the button marked 'Run SEMD Analysis' is clicked, the execution of the Data Acquisition Interface is paused in run mode until user activity is detected. Since the interface is embedded in a continuous loop, continually polling the interface to detect changes, this frees a significant portion of CPU time. The speed of the inverse calculations and the ease with which the user can rotate and interact with the Dipole Display are significantly improved.

Upon completion of dipole parameter calculations for the selected data segment, *pace_or_VT.m* stores these parameters in matrix form in the file 'plotm.txt'. This provides a level of protection in the event of a recoverable hardware malfunction, and is also a simple way of allowing many GUI functions to access the same data. The function *newsemd.m* controls user interaction with the Cardiac Dipole Display Interface. *newsemd* consists of several 'callback' functions, which are activated when the cardiologist presses the associated button on the interface. Two separate functions, *showspheres.m* and *imbed_animation.m*, are used to plot the dipole trajectory and individual dipole characteristics respectively.

Figure 7.1 shows the program flowchart for the Cardiac Dipole Display Interface.

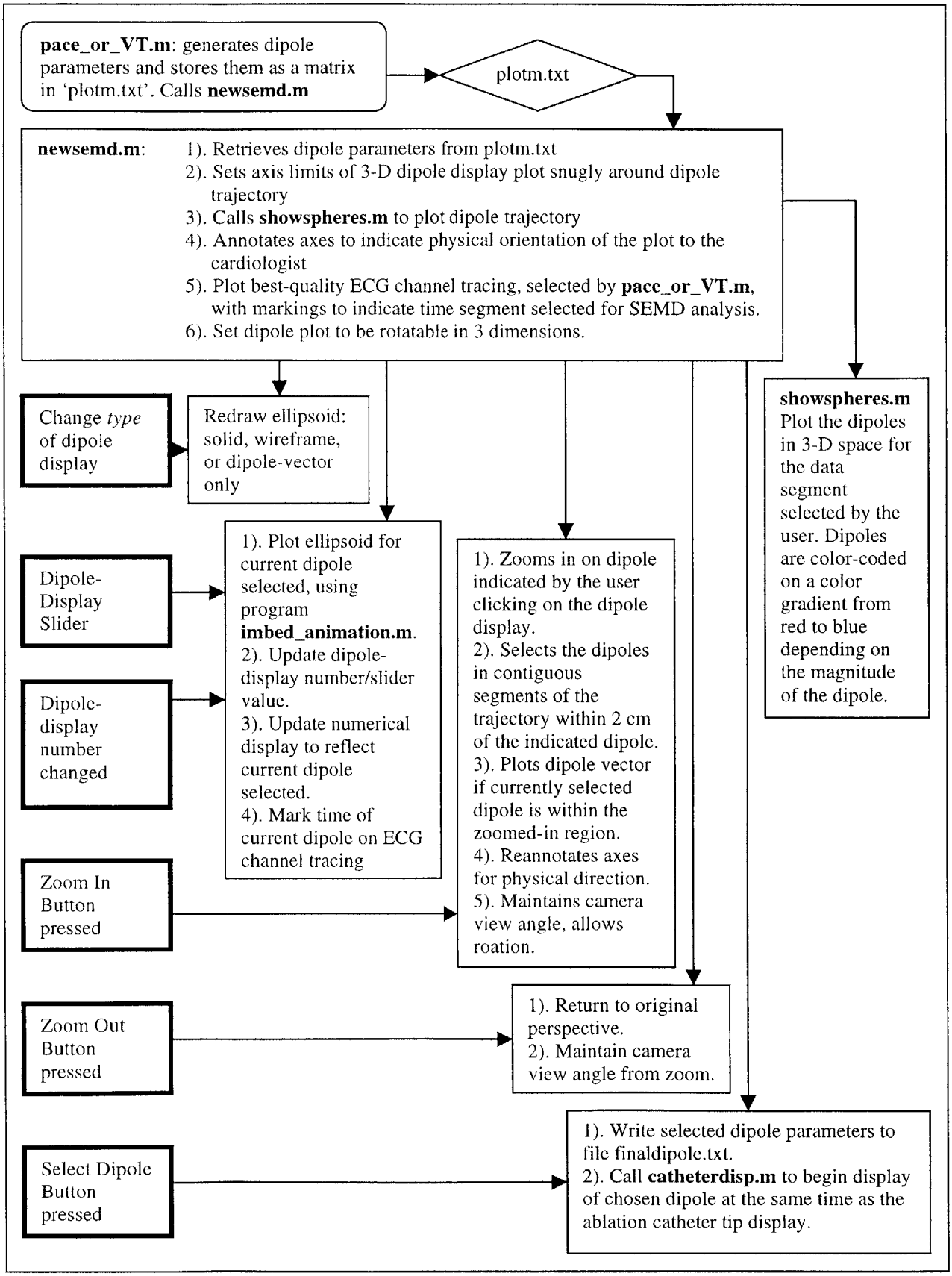


***Figure 7.1**: Program structure flowchart for Cardiac Dipole Display Interface*

7.2 The Cardiac Dipole Display Interface

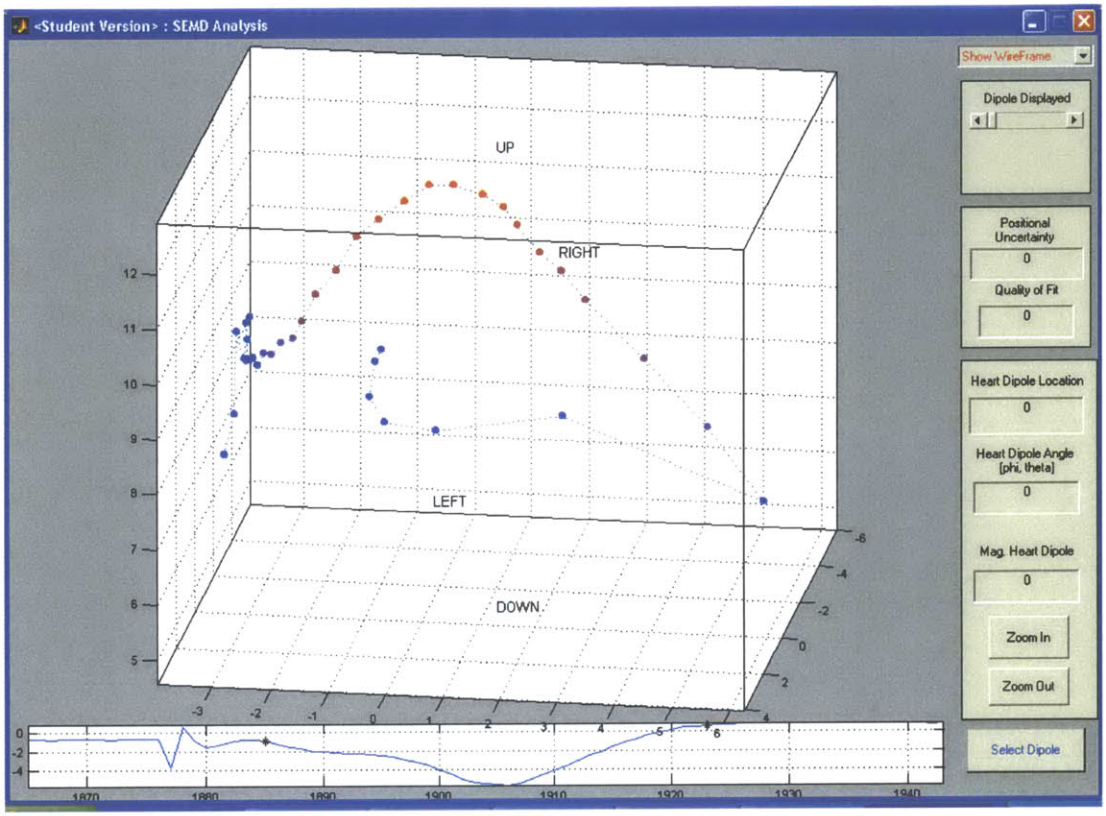


***Figure 7.2**: The Cardiac Dipole Display Interface*

7.2.1 The dipole trajectory

Figure 7.2 shows the Cardiac Dipole Display Interface and a dipole trajectory resulting from paced data. As described in the specifications, the trajectory is initially displayed as a series of 'dipole dots'. They are connected by a dotted line for the cardiologist to more easily gauge the dipole's continuity in time, and are color-coded on a scale from blue to red indicating increasing dipole magnitude. Also, the dipole display is rotatable in three-dimensions. The numbers on the axes indicate the physical dimensions of the trajectory and its position within the torso, while text indicating the top, bottom, left and right of the torso assist in orientation.

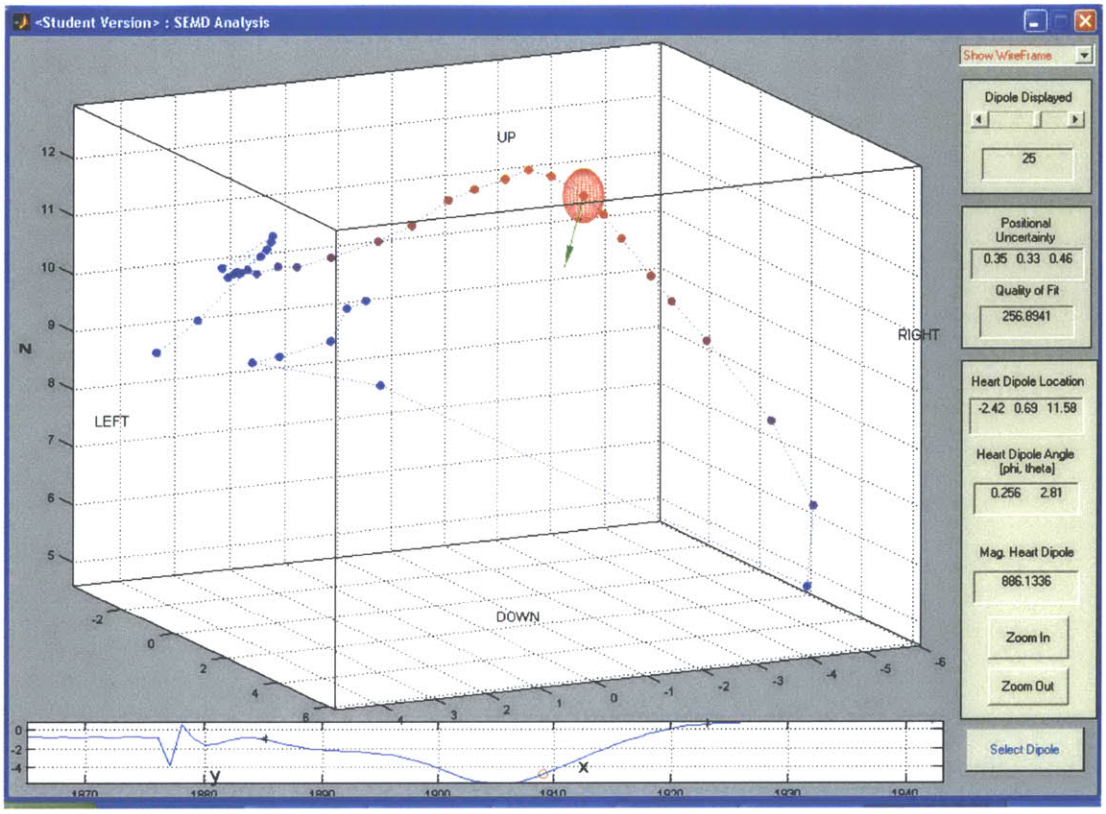
The plot at the bottom of the GUI displays a high-quality ECG channel tracing chosen by the program *pace_or_VT.m*. Black markers on the waveform (seen here at the beginning and end of the QRS complex) demarcate the segment selected for SEMD analysis.

To view a segment of the trajectory in greater detail, the cardiologist presses the button marked 'Zoom In' at the bottom-right of the interface window. If the user then mouse-clicks on the display axes above the dipole of interest, the dipole and the contiguous segment of trajectory within a four-centimeter cube are automatically selected and enlarged. The camera view angle is maintained. The zoomed-in section is rotatable and annotated to show physical dimension, location within the torso and orientation. To return to the original display dimensions while maintaining the present viewing angle the cardiologist clicks the button marked 'Zoom Out'.

7.2.2 Individual Dipole Parameter Display

To view the parameters of individual dipoles, the cardiologist can view each dipole in turn using the slider at the top right of the Cardiac Dipole Display interface. This will update the dipole number displayed below the slider, and the numerical displays of dipole location, orientation, magnitude, positional uncertainty, and quality of fit. On the 3-D trajectory display, the 'dot' corresponding to the currently selected dipole expands into an ellipsoid whose dimensions reflect the positional uncertainty. Selecting a different choice (solid, wireframe, or dipole vector only) from the popup menu at the top-right will change the appearance of the ellipsoid. The time of the current dipole displayed is marked on the ECG tracing with a red circle.

Figure 7.3a shows the Dipole Display GUI with a selected dipole. The red marker on the ECG tracing indicates that this dipole occurs just after the peak of the QRS. If the cardiologist zooms in on the currently selected dipole, the vector associated with that dipole is also displayed. This is shown in Figure 7.3b.



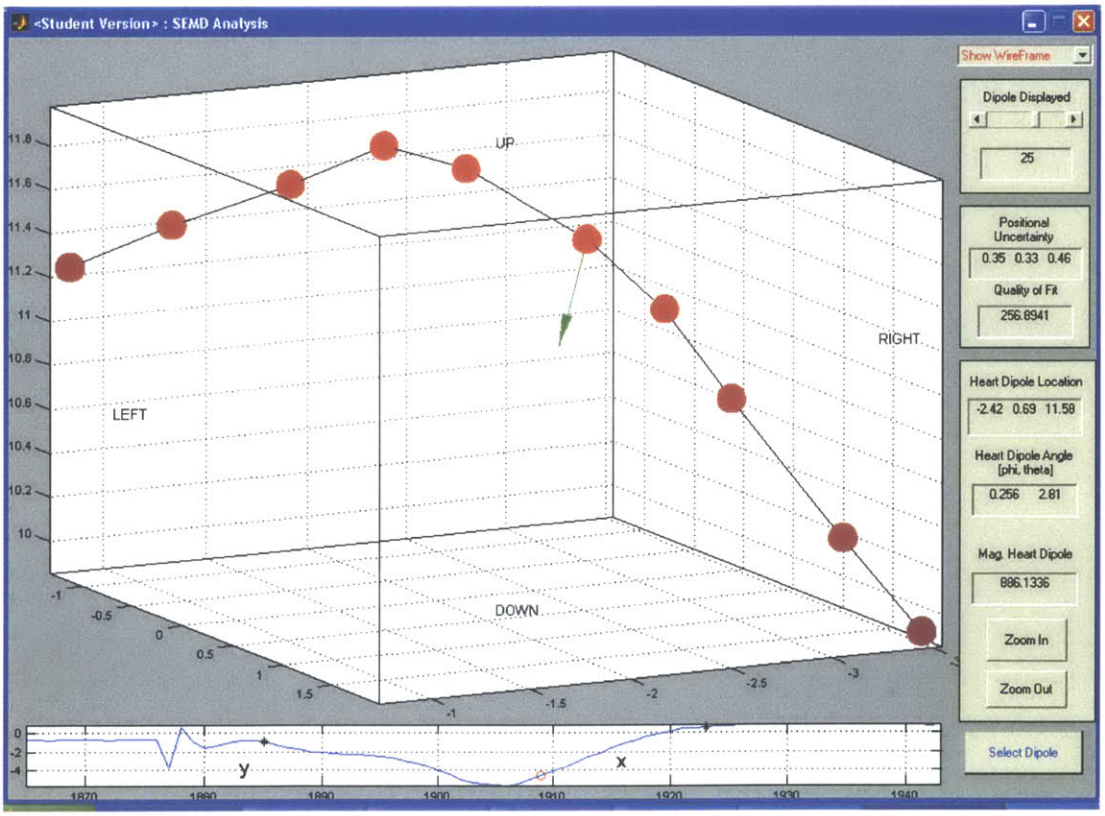


***Figure 7.3**: a). Dipole trajectory with dipole at time t = 1909 currently selected b). A closer view.*

7.3 Selecting the dipole of interest

Using the parameters displayed in the Cardiac Dipole Display Interface, the cardiologist must choose the dipole that he/she feels best represents the exit site of the reentrant circuit. This occurs early in the cardiac cycle and dipole trajectory, and is the point closest to the site of earliest ventricular activation. To assist the cardiologist, an additional window is automatically presented in conjunction with the dipole display showing the change in several dipole parameters (magnitude, chi-square, etc.) over the course of the selected time segment. An example of this window for a segment of paced data is shown in Figure 7.4.

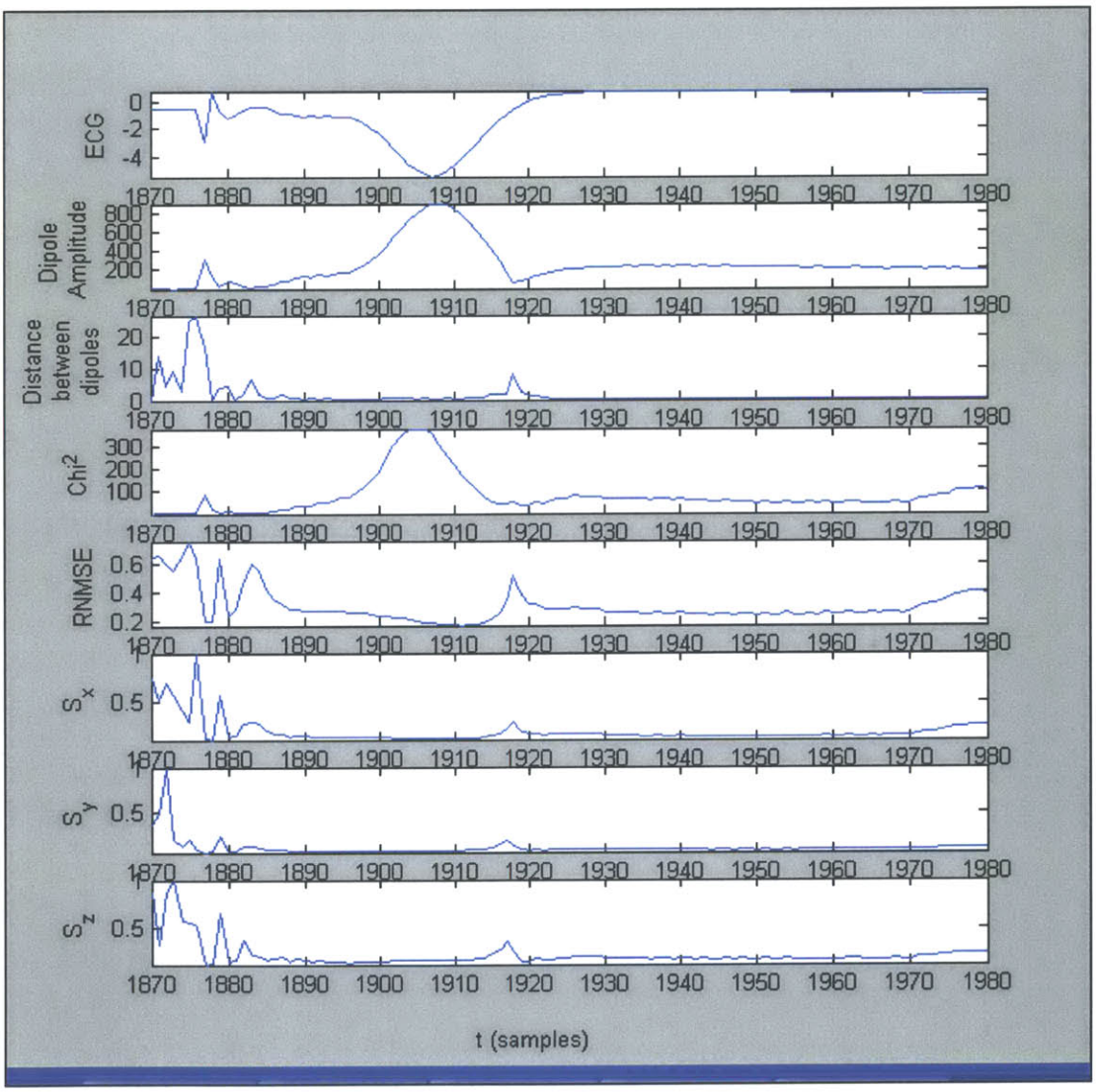


***Figure 7.4**: Window displayed alongside the Dipole Display Interface to aid the cardiologist in choosing an ablation site.*

An analysis of the dipole trajectory and the use of available information to select an ablation site is discussed in Section 8. To choose the dipole currently displayed, the cardiologist clicks the button marked 'Select Dipole'. This will write the current dipole parameters to 'finaldipole.txt', close the Cardiac Dipole Display GUI, and call *catheterdisp.m.*

7.4 The Ablation Interface: Simultaneous display of dipole and ablation catheter

The program *catheterdisp.m* controls the callback functions for a Matlab GUI designed to simultaneously display the selected cardiac dipole and the dipole due to the ablation catheter tip. Figure 7.5 shows the Ablation Interface. The catheter is represented as a small dark-blue sphere, while the cardiac dipole is a lighter ellipsoid with dimensions equal to its positional uncertainty. The legend at the top right also assists in differentiating the two dipoles. As for the previous interface, the display axes are annotated to indicate direction and location within the torso.

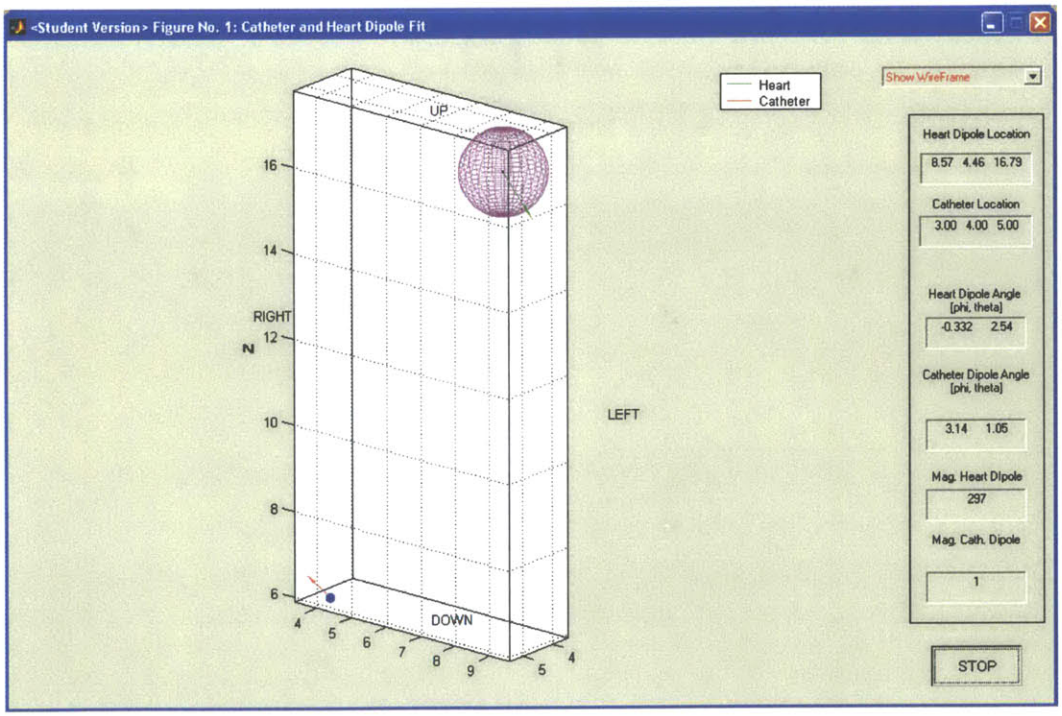


***Figure 7.5**: The Ablation Interface, in which both the cardiac and ablation catheter dipoles are displayed.*

Since an ablation catheter has yet to be built, the program simulates the movement of the catheter using a 'for' loop in which the (x, y, z) values of the catheter location are incremented every 2.5 seconds, the time the inverse algorithm takes to compute a dipole location. The parameters of the cardiac dipole and the current catheter dipole are displayed numerically on the interface. To give the cardiologist a better sense of spatial direction and distance the display axes are rotatable in three dimensions.

As before, the user can choose the type of dipole display from the pulldown menu at the top right. This becomes an especially useful feature as the catheter tip and cardiac dipole are brought into proximity, since the cardiologist will wish to view *only* the exact location and orientation of the dipole. Furthermore, the axes are programmed to fit tightly around the dipoles. Therefore, as their locations converge, the program automatically increases the spatial resolution for greater accuracy of alignment and ablation.

Once the two dipoles are aligned both in location and orientation, the ablation current is delivered. If VT is still inducible with the same morphology, the cardiologist can return to the same Dipole Display Interface and choose a different cardiac dipole. Alternatively, if the VT morphology has changed – indicating the formation of a new reentrant circuit – the entire procedure can be easily repeated by again collecting a few seconds of data preceeded by sinus rhythm, selecting a characteristic data segment, choosing a representative dipole and aligning it with the ablation catheter.

8. The Choice of Ablation Site

The cardiologist wishes to deliver ablation current to the exit point of the reentrant circuit, the site of earliest systolic activation during VT. The dipole representing this point is highly localized, since the activation wavefront is not diffuse; therefore it has a small magnitude, and is well-estimated by the inverse algorithm. Figures 8.1 and 8.2 show the change in several important dipole characteristics over the cardiac cycle, for paced and VT data respectively. These figures are identical to those that appear in conjunction with the Dipole Display Interface, therefore the goal of this section is to describe how the cardiologist would use this information in his or her choice of ablation site.

The first plot in each figure shows the ECG tracing from the highest quality channel (selected by *pace_or_VT.m*). The dipole parameters corresponding to these time samples are plotted below them. These parameters are: dipole vector amplitude, the distance between each consecutive dipole, chi-square, root normalized mean square error (RNMSE), and S_x, S_y and S_z (the positional uncertainties in x, y and z). Each parameter is discussed in detail below.

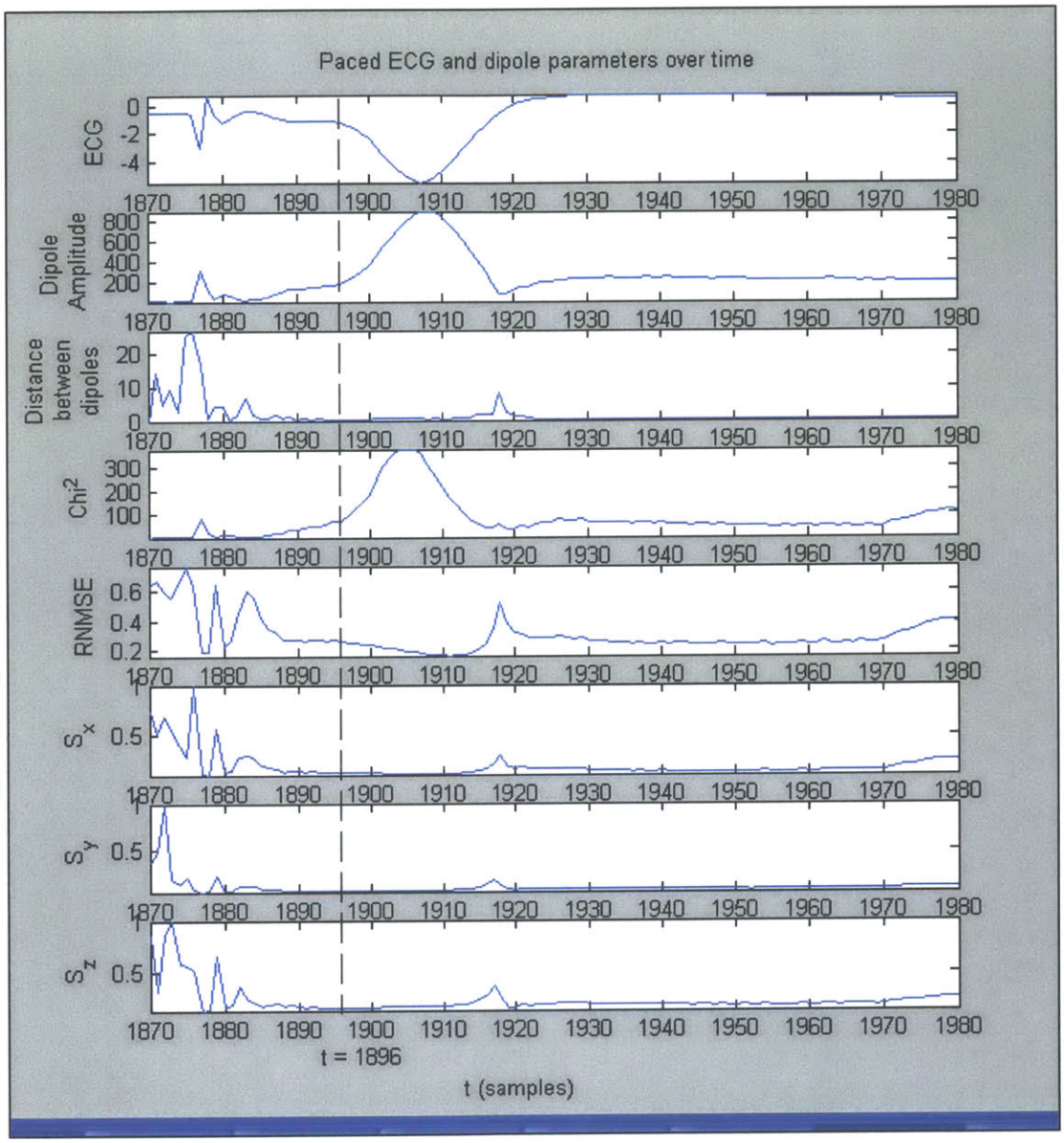


***Figure 8.1**: The change in several dipole characteristics over time between the spike artifact and the end of the QRS complex in paced data.*

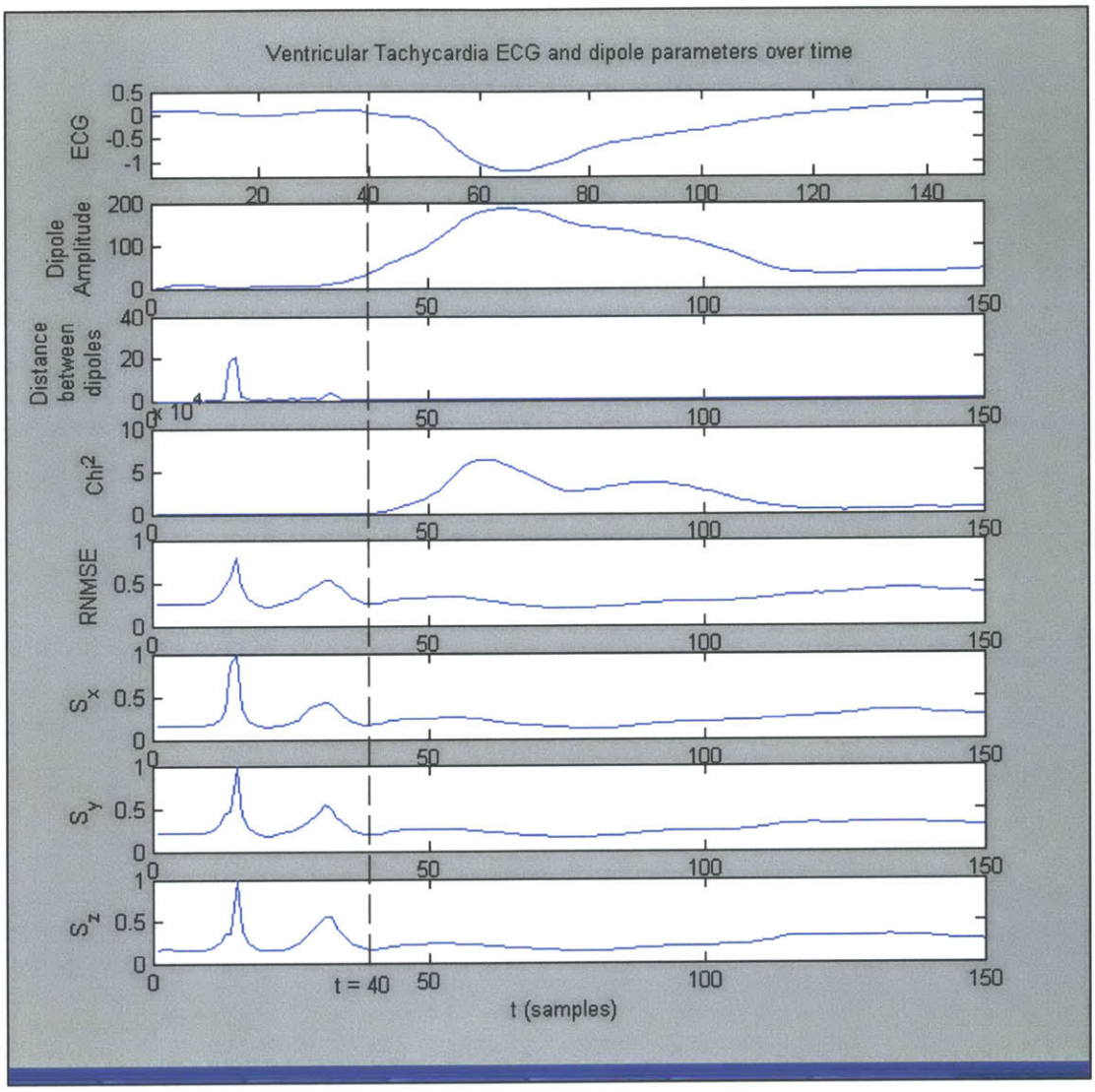


***Figure 8.2**: The change in several dipole parameters over a cardiac cycle of single-beat VT data.*

8.1 Position in ECG waveform and within the dipole trajectory

The dipole trajectory display and ECG waveforms alone contain significant information. Since the reentrant circuit exit point is the site from which excitation spreads to the rest of the myocardium, the corresponding dipole will occur early in the QRS complex and the dipole trajectory. Also, as shown in Figure 8.3, the dipole locations are highly uncorrelated before and during the pacing spike. However, after the initial down-turn of the QRS, the dipoles follow a clear trajectory. At the down-turn itself (circled in the figure), the dipoles are closely grouped since the area of activation is minimal and the speed of depolarization slow. This is the region in which the reentrant dipole will be found.

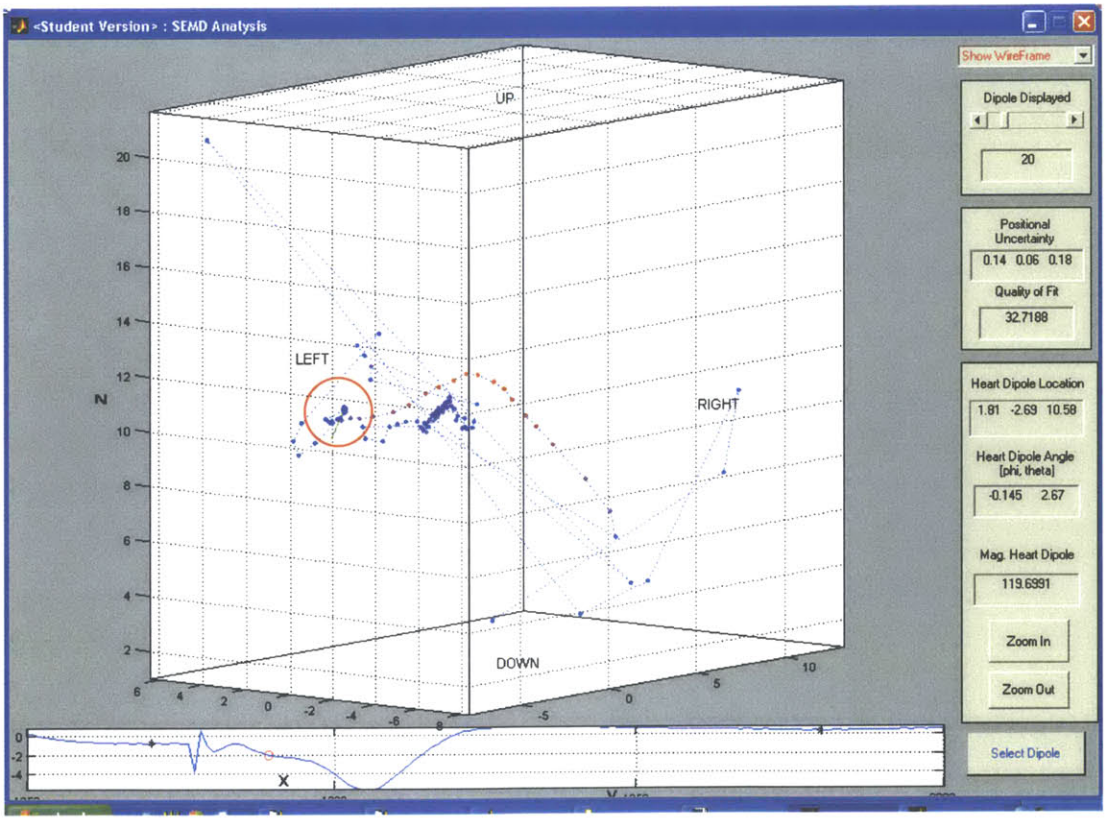


***Figure 8.3**: Dipole trajectory for paced data, for the region before the pacing spike to the end of the QRS. The uncorrelated section of the dipole trajectory occurs before, during and slightly after the pacing spike. The point marked in red on the ECG corresponds to the time at which the dipoles highlighted by a circle on the display occur. Here, the dipoles become closely grouped. This point is the start of the QRS complex, and these dipoles are close to the exit point of the reentrant circuit.*

8.2 Dipole Amplitude

Since the dipole is highly localized when it leaves the reentrant circuit, we are interested in the region in which the dipole amplitude is small. This region is found near the beginning of the QRS.

8.3 Distance between consecutive dipoles

As noted above, the dipole positions before and during the pacing spike are uncorrelated because of inaccurate estimation, and therefore the distances between them are large. The inverse algorithm can only approximate the dipole more accurately when the SNR increases at the start of the QRS. Here, the dipoles are closely spaced since the area of activation is small and the speed of depolarization slow. The plot of distance between adjacent dipoles helps the cardiologist find this dipole cluster at the beginning of the QRS. The dipole chosen as the site of ablation should occur soon after the distance between consecutive dipoles reaches a minimum.

8.4 Chi-square and RNMSE

The chi-square value is a comparison between the residual of the dipole fit and the noise, or a measure of "goodness-of-fit". In an ideal system with a highly localized dipole, chi-square would be approximately equal to one since the error in the model would be entirely due to random noise. In reality, due to the presence of systematic error (as a result of tissue inhomogeneities, non-uniform electrode distribution over the body surface, etc.) chi-square will never have unit value even for a good dipole estimate.

If cardiac electrical activity is well-localized, the SEMD representation will be physically meaningful and the estimated voltages should lie close to the measured values. As the wavefront becomes more diffuse, the SEMD representation becomes essentially meaningless and the chi-square value increases. This can be clearly seen in Figure 8.1; the maximum chi-square lies near the peak of the QRS, where the voltages are largest and the wavefront is becoming highly diffuse. Therefore, the ablation site dipole should be characterized by a low chi-square value to ensure that the dipole estimation is good.

The RNMSE (root normalized mean square error) represents the normalized root-mean-square difference between the measured and estimated electrode voltages due to a dipole at location (x, y,z) with moment (p_x, p_y, p_z). It indicates how much potential in the measured signal cannot be attributed to the SEMD:

$$RNMSE = \sqrt{\frac{\sum_{i=1}^{I}(V^i{}_e - V^i{}_m)^2}{\sum_{i=1}^{I}(V^i{}_m)^2}}$$

where $V^i{}_e$ is the voltage at electrode estimated by the forward algorithm, $V^i{}_m$ is the measured voltage at the same electrode, and I is the total number of channels.

RNMSE is a normalized measure of error; therefore its value is large (close to 1) if the measured potentials and noise are on the same order. For both paced and VT data, high RNMSE values are found in the low-voltage period before the start of the QRS complex. A dipole estimated from samples in this region would likely be inaccurate. As the SNR increases with increasing cardiac activity, the dipole estimate initially becomes more accurate and the RNMSE falls. However, RNMSE divides the error in the estimate by the absolute value of the measured voltages, therefore it may decrease even though the SEMD approximation is less accurate near the peak of the QRS. Consequently, the dipole chosen as the ablation point should occur soon after the initial peaks in RNMSE (see Figures 23 and 24) although not at an absolute minimum.

8.5 The Positional Uncertainties in x, y and z

The positional uncertainty is the standard deviation of the position of the estimated dipole given the standard deviation of the error in each lead. A low value assures us that high noise or model inaccuracies have not significantly distorted the dipole location, and that under the same noise conditions and using the same inverse algorithm, the ablation catheter can be accurately aligned with the dipole. Before the start of the QRS complex, the positional uncertainty is high since the standard deviation of the noise is comparable to the measured potentials; therefore slight fluctuations in the lead voltages due to noise could lead to large perturbations in the dipole location. Later in the QRS, positional uncertainty is a balance between the increasing error in the estimation as the wavefront becomes more diffuse, and a higher signal-to-noise ratio that means

perturbations in the signal due to noise have less effect. Therefore, the selected dipole should have a minimal positional uncertainty and occur soon after S_x, S_y and S_z decrease near the beginning of the QRS.

8.6 Which dipole best represents the exit point?

Based on the arguments above, the time point that best approximates the initiation of activity is marked with a dashed line on Figures 8.1 and 8.2. The dipoles that correspond to these time points are shown in Figure 8.4 and 8.5 below.

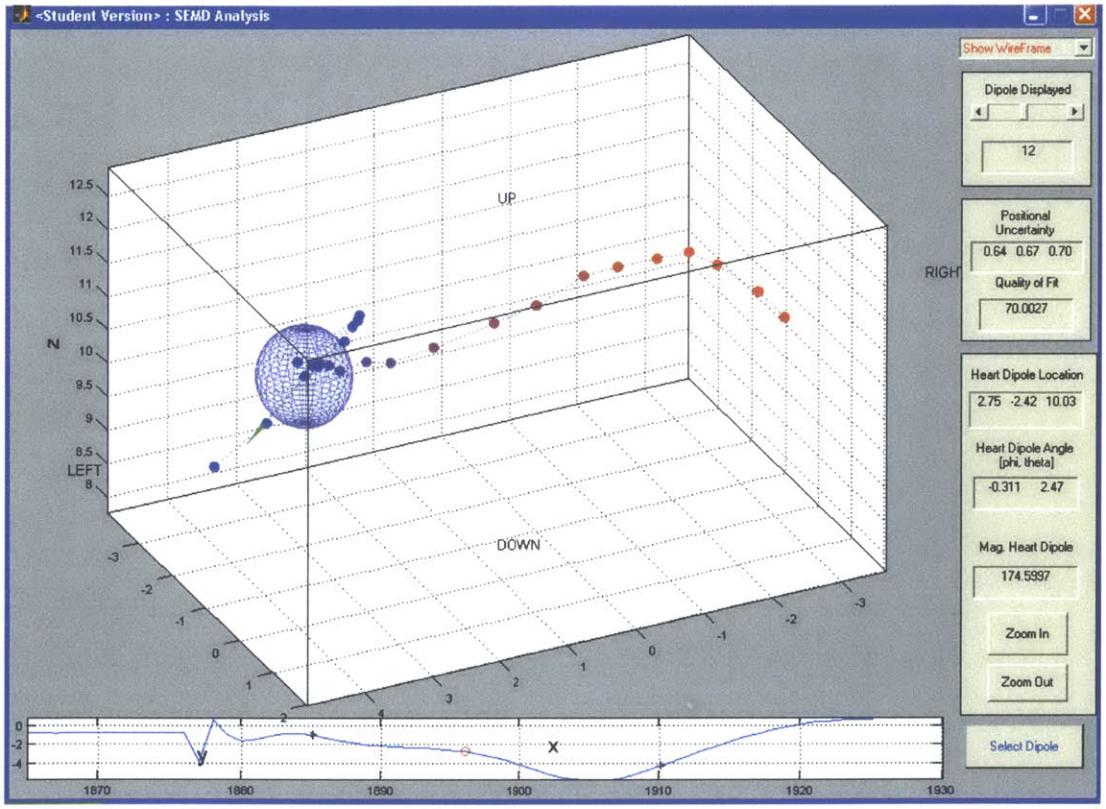


***Figure 8.4**: Dipole that best represents the pacing site ('exit of the reentrant circuit') for the paced data, corresponding to the time marked by a dashed line in Figure 8.1.*

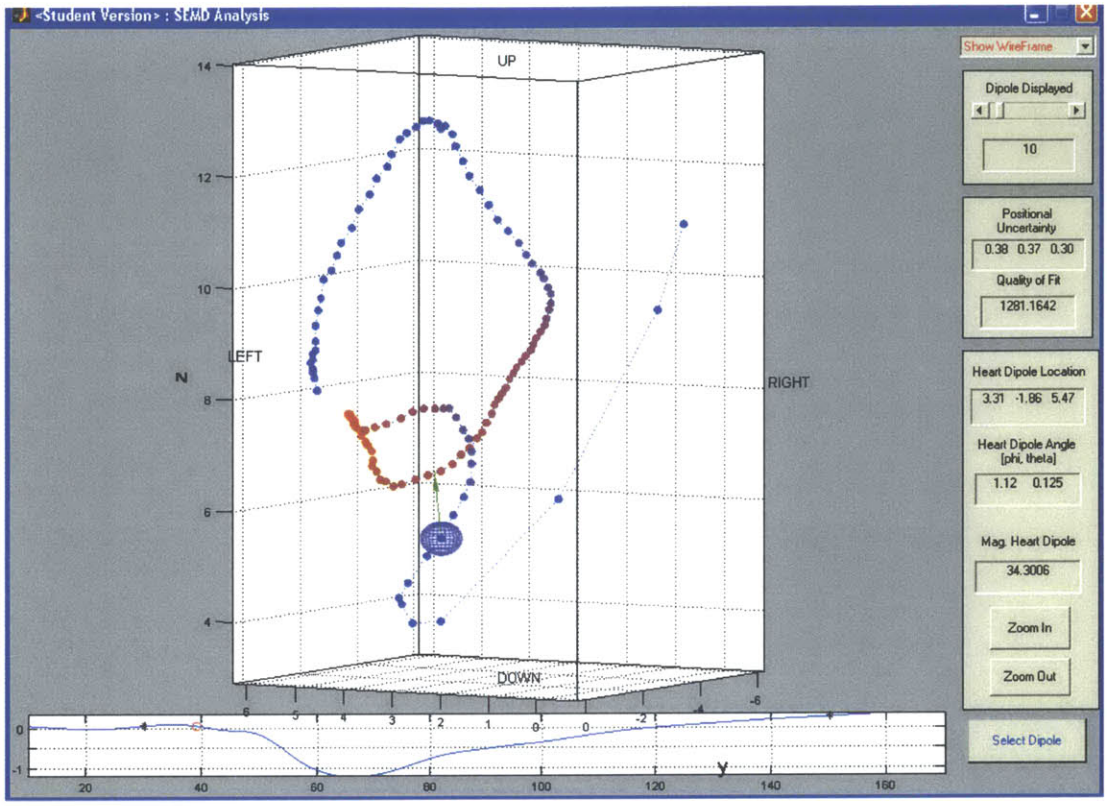


***Figure 8.5**: Dipole that best represents the exit of the reentrant circuit for the slow VT BSPM data, corresponding to the time marked by a dashed line in Figure 8.2.*

9. Conclusions and Future Work

The Inverse Technique offers a novel method for the radio-frequency ablation of arrhythmias, providing a rapid and more accurate method of identifying the exit site of a reentrant circuit. The 90% of ventricular tachycardia patients who are currently untreatable could become candidates for ablation, dramatically improving their quality-of-life and life expectancy. Furthermore, treating VT patients with a goal to *cure* rather than ameliorate their condition will have significant economic impact, since ICD implantation and maintenance will no longer be necessary.

The software described in this thesis provides an effective and seamless RFA user interface, with all the tools a cardiologist would require to localize and ablate an arrhythmic site. The format is straight-forward and does not require lengthy training before it can be used. The software is also perfectly suited to the current RFA setting, in which multiple screens are used to display all information simultaneously; in this case, the Data Acquisition, Cardiac Dipole Display and Ablation interfaces could be viewed concurrently. Once an ECG system has been chosen, and a driver and data acquisition program implemented in LabVIEW, the software can be run 'as is'.

Although the visualization software and inverse algorithm have been fully implemented, much work must be done before this technology can be used in a medical setting. Firstly, the hardware must be built. The technique that is chosen to locate the ablation catheter tip will heavily influence how the hardware is designed. Localizing dipole currents at the catheter tip will necessitate not only the design of a custom-made ablation catheter, but also require an ECG system with a bandwidth extending to 1 kHz to avoid the interference of bio-potentials. A few such systems do exist, such as BioSEMI's ActiveTwo device with a bandwidth of 1.6 kHz and 64 channels. These systems are expensive, costing anywhere from $28,000 to $60,000. If instead the tip is localized by pacing the endocardium with the ablation catheter, a standard catheter and ECG system may be used. However, since typical ECG cards allow only 12 leads of input, a method for synchronizing six ECG cards will need to be devised to obtain the required 64 channels.

Once the hardware has been designed and assembled, the algorithm must be thoroughly tested with BSPM data and known dipole locations. This data can be collected in the swine model in the following manner. A pacing catheter is fitted with a simple ink-injector system and inserted into

the heart, and several beats of paced VT are recorded. The location of the stimulation can be marked by injecting a small amount of ink into the tissue underneath the electrode. This data can then be used in two ways. Firstly, the user-interface software can be used to calculate the trajectory of the SEMD during pacing and to find the site at which excitation appears to originate. After the heart is dissected, a comparison of the dipole found by the software and the actual location of the pacing electrode will determine the *absolute* accuracy of the algorithm. Secondly, after withdrawing the pacing electrode, the ablation catheter can be inserted and aligned with the probable pacing site. By burning the tissue at this site and then comparing the locations of the burn and ink marks after dissection of the heart, the precision of the ablation catheter alignment algorithm can be determined (the *relative* accuracy of the software).

This testing can also be used to improve the criteria with which a cardiologist should choose an ablation site. Once these criteria have been clearly defined, software can be written that will automatically find the dipole that best matches these criteria and suggest this site to the cardiologist.

In summary, the software implemented here could provide a successful arrhythmia mapping technique and valuable user interface for a fully assembled RFA device. The hardware must be carefully designed, and the inverse algorithm software extensively tested in animal models, before the technology can be used to treat humans. However, if this is eventually made available to the medical community, it has the potential to be of great benefit to the treatment of ventricular arrhythmias.

10. References

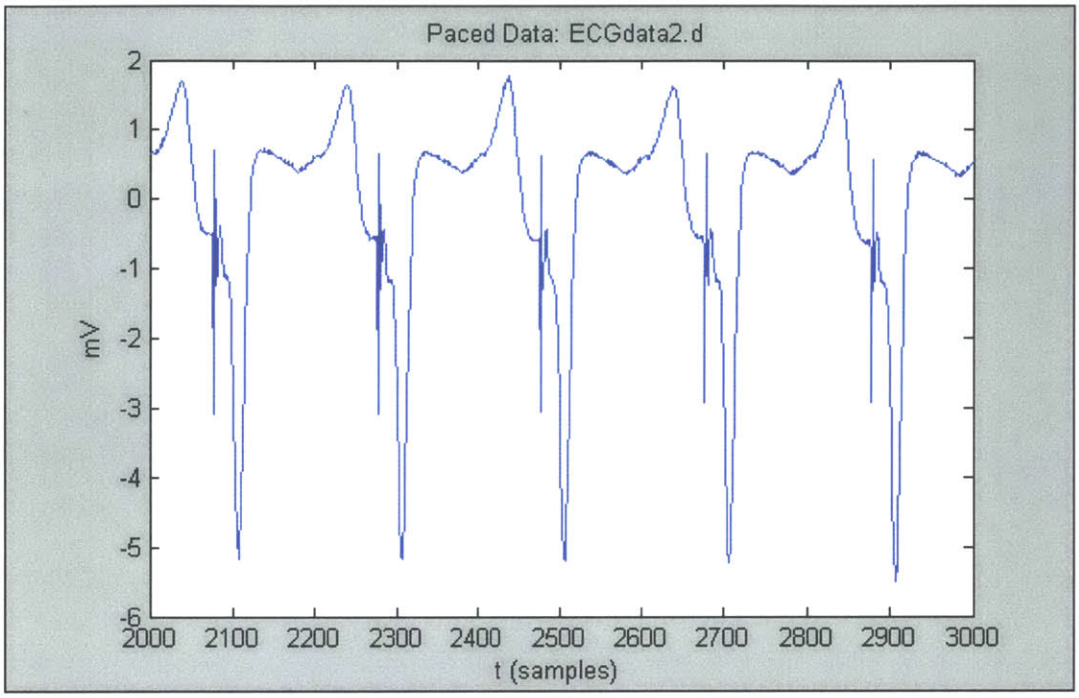
1. Shumway, S.J., et al., *Surgical management of ventricular tachycardia.* Ann Thorac Surg, 1997. **63**(6): p. 1589-91.
2. Mitrani, R.D., et al., *Regional cardiac sympathetic denervation in patients with ventricular tachycardia in the absence of coronary artery disease.* J Am Coll Cardiol, 1993. **22**(5): p. 1344-53.
3. Zipes, D.P. and H.J. Wellens, *Sudden cardiac death.* Circulation, 1998. **98**(21): p. 2334-51.
4. Theroux, P. and V. Fuster, *Acute coronary syndromes: unstable angina and non-Q-wave myocardial infarction.* Circulation, 1998. **97**(12): p. 1195-206.
5. Zipes, D.P., *What have we learned about Cardiac Arrhythmias?* Circulation, 2000. **102**(IV): p. 52-57.
6. Soejima, K. and W.G. Stevenson, *Ventricular tachycardia associated with myocardial infarct scar: a spectrum of therapies for a single patient.* Circulation, 2002. **106**(2): p. 176-9.
7. investigators, T.E., *Determinants of predicted efficacy of antiarrhythmic drugs in the electrophysiologic study versus electrocardiographic monitoring trial. The ESVEM Investigators.* Circulation, 1993. **87**(2): p. 323-9.
8. Nattel, S., D.H. Pedersen, and D.P. Zipes, *Alterations in regional myocardial distribution and arrhythmogenic effects of aprindine produced by coronary artery occlusion in the dog.* Cardiovasc Res, 1981. **15**(2): p. 80-5.
9. Investigators, C.I., *Effect of the antiarrhythmic agent moricizine on survival after myocardial infarction. The Cardiac Arrhythmia Suppression Trial II Investigators.* N Engl J Med, 1992. **327**(4): p. 227-33.
10. investigators, T.A., *A comparison of antiarrhythmic-drug therapy with implantable defibrillators in patients resuscitated from near-fatal ventricular arrhythmias.* N Engl J Med, 1997. **1997**(337): p. 1576-1583.
11. Connolly, S.J., et al., *Canadian implantable defibrillator study (CIDS) : a randomized trial of the implantable cardioverter defibrillator against amiodarone.* Circulation, 2000. **101**(11): p. 1297-302.
12. Kuck, K.H., et al., *Randomized comparison of antiarrhythmic drug therapy with implantable defibrillators in patients resuscitated from cardiac arrest : the Cardiac Arrest Study Hamburg (CASH).* Circulation, 2000. **102**(7): p. 748-54.
13. Stevenson, W.G. and L.M. Epstein, *Predicting sudden death risk for heart failure patients in the implantable cardioverter-defibrillator age.* Circulation, 2003. **107**(4): p. 514-6.
14. Kottkamp, H., et al., *Radiofrequency catheter ablation of sustained ventricular tachycardia in idiopathic dilated cardiomyopathy.* Circulation, 1995. **92**(5): p. 1159-68.
15. Mushlin, A.I., et al., *The cost-effectiveness of automatic implantable cardiac defibrillators: results from MADIT. Multicenter Automatic Defibrillator Implantation Trial.* Circulation, 1998. **97**(21): p. 2129-35.

16. Rosenqvist, M., et al., *Adverse events with transvenous implantable cardioverter-defibrillators: a prospective multicenter study. European 7219 Jewel ICD investigators.* Circulation, 1998. **98**(7): p. 663-70.
17. Strickberger, S.A., et al., *A prospective evaluation of catheter ablation of ventricular tachycardia as adjuvant therapy in patients with coronary artery disease and an implantable cardioverter-defibrillator.* Circulation, 1997. **96**(5): p. 1525-31.
18. Morady, F., *Radio-Frequency Ablation as Treatment for Cardiac Arrhythmias.* N Engl J Med, 1999. **340**(7): p. 534-544.
19. Soejima, K., et al., *Electrically unexcitable scar mapping based on pacing threshold for identification of the reentry circuit isthmus: feasibility for guiding ventricular tachycardia ablation.* Circulation, 2002. **106**(13): p. 1678-83.
20. Dixit, S. and D.J. Callans, *Mapping for ventricular tachycardia.* Card Electrophysiol Rev, 2002. **6**(4): p. 436-41.
21. Stevenson, W.G., et al., *Exploring postinfarction reentrant ventricular tachycardia with entrainment mapping.* J Am Coll Cardiol, 1997. **29**(6): p. 1180-9.
22. Stevenson, W.G., et al., *Radiofrequency catheter ablation of ventricular tachycardia after myocardial infarction.* Circulation, 1998. **98**(4): p. 308-14.
23. Furniss, S., et al., *Radiofrequency ablation of haemodynamically unstable ventricular tachycardia after myocardial infarction.* Heart, 2000. **84**(6): p. 648-52.
24. Wittkampf, F.H., et al., *Accuracy of the LocaLisa system in catheter ablation procedures.* J Electrocardiol, 1999. **32 Suppl**: p. 7-12.
25. Soejima, K., et al., *Catheter ablation in patients with multiple and unstable ventricular tachycardias after myocardial infarction - Short ablation lines guided by reentry circuit isthmuses and sinus rhythm mapping.* Circulation, 2001. **104**(6): p. 664-669.
26. Davies, D.W., *Catheter ablation of ventricular tachycardia: are there limits?* Heart, 2000. **84**(6): p. 585-586.
27. Strickberger, S.A., et al., *Mapping and ablation of ventricular tachycardia guided by virtual electrograms using a noncontact, computerized mapping system.* J Am Coll Cardiol, 2000. **35**(2): p. 414-21.
28. Eldar, M., et al., *Transcutaneous multielectrode basket catheter for endocardial mapping and ablation of ventricular tachycardia in the pig.* Circulation, 1997. **96**(7): p. 2430-7.
29. Burnes, J.E., B. Taccardi, and Y. Rudy, *A noninvasive imaging modality for cardiac arrhythmias.* Circulation, 2000. **102**(17): p. 2152-8.
30. Cohen, T.J., et al., *Development of an interactive computer-guided method for radiofrequency catheter ablation of ventricular tachycardia.* Pacing Clin Electrophysiol, 1996. **19**(4 Pt 1): p. 472-6.
31. Sippensgroenewegen, A., *Localization of the site of origin of postinfarction ventricular tachycardia by endocardial pace mapping: body surface mapping compared with the 12-lead electrogram.* Circulation, 1993. **88**(1): p. 2290-2306.

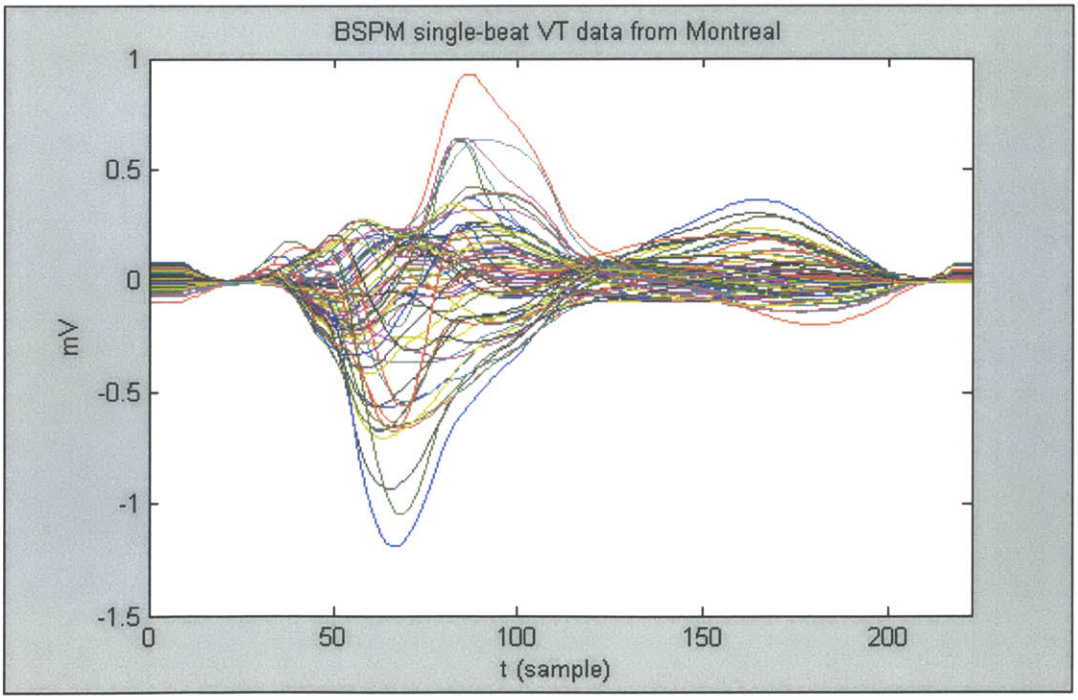
32. Armoundas, A.A., et al., *Applicability of the single equivalent point dipole model to represent a spatially distributed bio-electrical source.* Med Biol Eng Comput, 2001. **39**(5): p. 562-70.
33. Armoundas, A.A., *A Novel Technique for Guiding Ablative Therapy of Cardiac Arrhythmias*, in *Nuclear Engineering*. 1999, MIT: Cambridge. p. 179.
34. Ramanathan, C. and Y. Rudy, *Electrocardiographic imaging: I. Effect of torso inhomogeneities on body surface electrocardiographic potentials.* J Cardiovasc Electrophysiol, 2001. **12**(2): p. 229-40.
35. Calkins, H., et al., *Radiation exposure during radiofrequency catheter ablation of accessory atrioventricular connections.* Circulation, 1991. **84**(6): p. 2376-82.
36. Sun, P., et al., *An improved morphological approach to background normalization of ECG signals.* IEEE Trans Biomed Eng, 2003. **50**(1): p. 117-21.
37. Sun, Y., K. Chan, and S.M. Krishnan, *ECG signal conditioning by morphological filtering.* Comput Biol Med, 2002. **32**(6): p. 465-79.
38. Li, C., C. Zheng, and C. Tai, *Detection of ECG characteristic points using wavelet transforms.* IEEE Trans Biomed Eng, 1995. **42**(1): p. 21-8.
39. Jane, R., *Adaptive Baseline Wander Removal in the ECG: Comparative Analysis with Cubic Spline Technique.* IEEE, 1992.
40. Shusterman, V., et al., *Enhancing the precision of ECG baseline correction: selective filtering and removal of residual error.* Comput Biomed Res, 2000. **33**(2): p. 144-60.
41. Anderson, R.S., D.W. Evans, and L.N. Thibos, *Effect of window size on detection acuity and resolution acuity for sinusoidal gratings in central and peripheral vision.* J Opt Soc Am A, 1996. **13**(4): p. 697-706.

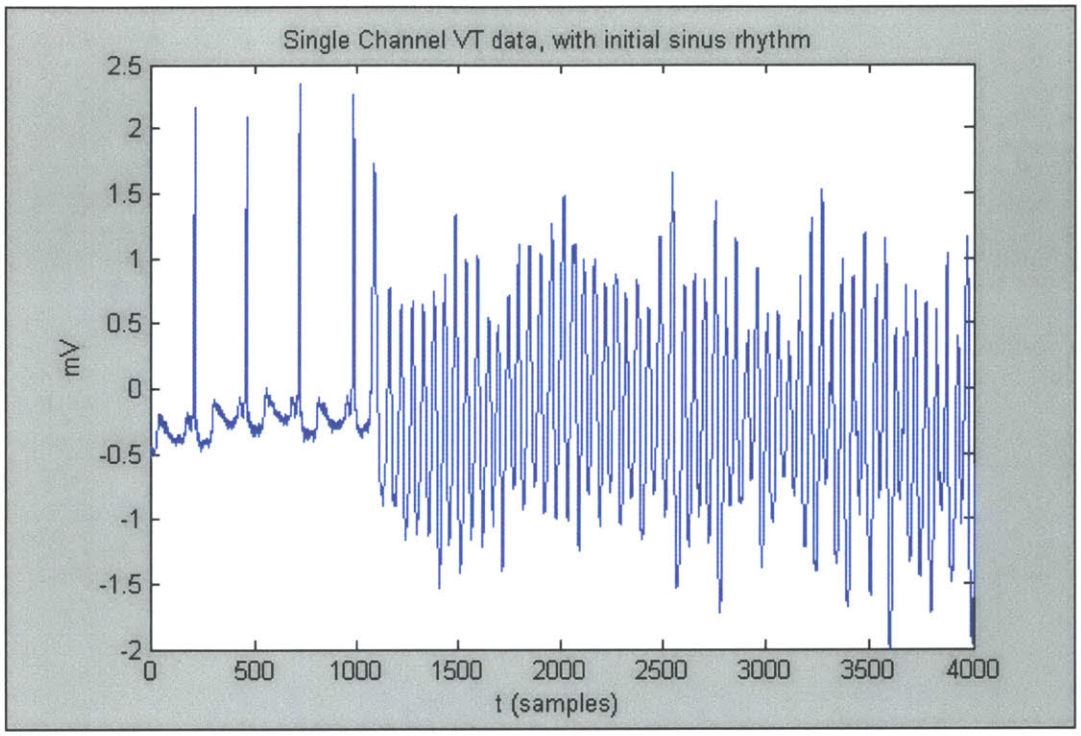
Appendix A.

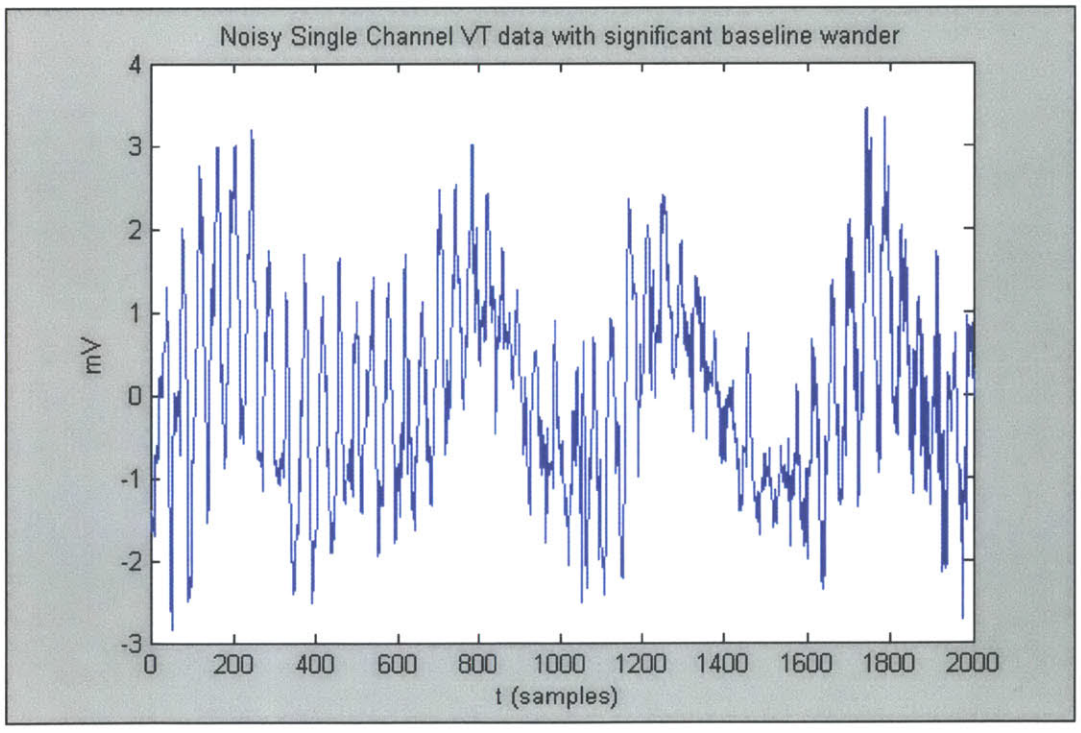
- Example of paced data used in software testing



- Examples of VT data used in software testing



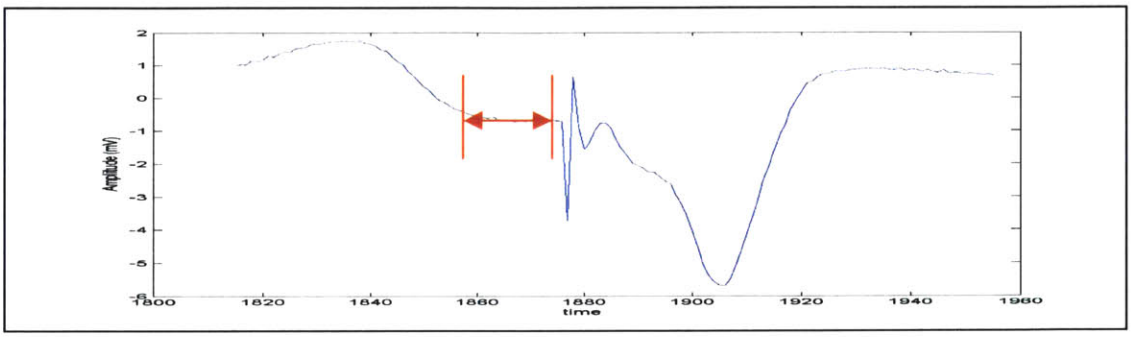

Single Channel VT data, with initial sinus rhythm
mV
t (samples)
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
500
1000
1500
2000
2500
3000
3500
4000


Noisy Single Channel VT data with significant baseline wander
mV
t (samples)
4
3
2
1
0
-1
-2
-3
0
200
400
600
800
1000
1200
1400
1600
1800
2000

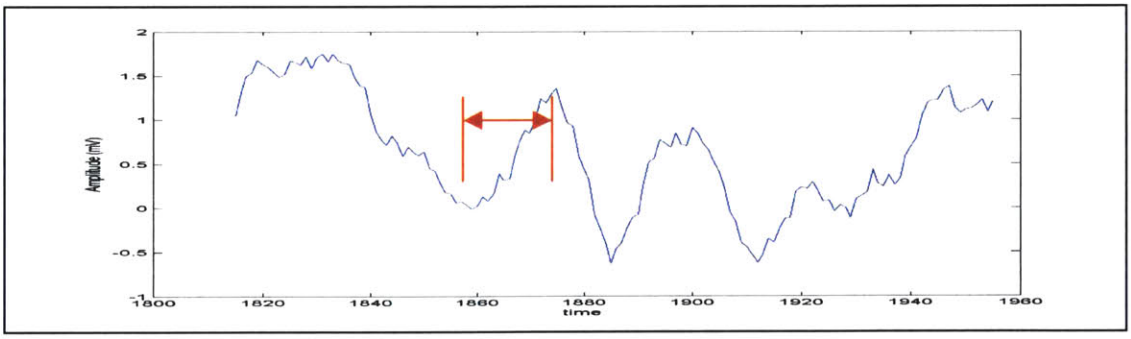
Appendix B.

- Examples of high- and low-noise channels in Paced Data

Sigma $_{noise}$ = 0.0264 (with isoelectric segment indicated)



Sigma $_{noise}$ = 0.379 (with isoelectric segment indicated)



Appendix C.

- Matlab Program Code

```
function [e_mat, R, len, r_torso, length_torso] = openSEMD(datafile, channels)

%%-------------------------------------------------------------------------%%
%% By looking at the filetype of 'datafile' (text, .d, or ascii), openSEMD opens the
%% specified file containing ECG data and returns the following parameters:
%%
%%          e_mat =    electrode positions in 3 dimensions
%%          R =        matrix of electrode voltages
%%          len =      time length of data in samples
%%          r_torso =  radius of torso used in the measurement
%%          length_torso = length of torso used in measurement
%%
%%-------------------------------------------------------------------------%%

data = char(datafile);
index = find(datafile == 46);

file_type = datafile(index + 1: length(datafile));

%%-------------------------------------------------------------------------%%
%% Filetype = .d

if file_type == [100],

   length_torso = 21;
   r_torso = 12.5;

   m = 1;
   electrodeMatrix = zeros(channels, 3);

   z_min = 0;
   z_max = 21;
   phi_min = -10;
   phi_max = 180;
   lines = 6;
   columns = 10;

   z_step = (z_max - z_min)/(lines - 1);
   phi_step = (phi_max - phi_min)/(columns - 1);

%    for z = 0:z_step: 21,
%       for phi = phi_min:phi_step:phi_max;
%          xval = r_torso * cos((phi - 90)*pi/180);
%          yval = r_torso * sin((phi - 90)*pi/180);
%          electrodeMatrix(m, :) = [xval, yval, z];
%          m = m+1;
%       end
%    end
```

```
   z_step = -z_step;

for phi = phi_min:phi_step:phi_max;
   for z = 21:z_step: 0,
      xval = r_torso * cos((phi - 90)*pi/180);
      yval = r_torso * sin((phi - 90)*pi/180);
      electrodeMatrix(m, :) = [xval, yval, z];
      m = m+1;
   end
end

   e_mat = electrodeMatrix;

   fid = fopen(data);

   [A, COUNT] = fread(fid, inf, 'int16');
   len = COUNT/channels;

   R = zeros(len, channels);

   for j = 1: channels
      for i = 1: len
         choose = ((i-1)*channels) + j;
         R(i, j) = A(choose);

      end
   end

   R = R.* 0.009765625;


%%-------------------------------------------------------------------%%
%% Add a linear slope to R vectors to see how baseline alg handles it!!

%    I = repmat((linspace(1, 6000, 6000) .* 0.001)', 1, channels);
%    R = R + I;


%%-------------------------------------------------------------------%%
%% Filetype = txt

elseif file_type == [116 120 116],

   length_torso = 21;
   r_torso = 12.5;

   e_mat = [1, 1, 1];

   A = load(data);
   no_ch = size(A, 2);
   R = A(:, 2: no_ch);
   len = size(R,1);

%%-------------------------------------------------------------------%%
%% Filetype = .asc
```

```
elseif file_type == [97 115 99],

   length_torso = 20;
   r_torso = 15;

   e_mat = channel_info(channels);

   d = load(data);
   if channels == 63,
      R = d(:, [1:47, 49:64]) .* 0.001;
   else
      R = d;
   end

   len = size(R, 1);

end
```

```
function ch_pos= channel_info(ch_num)

%%-------------------------------------------------------------------------%%
% Returns channel positions for Didier Klug's BSPM data

%% torso3Dmatch = node --> electrode pairings
%% torso3D = node positionings in 3D space
%%-------------------------------------------------------------------------%%

load torso3Dmatch.asc;
load torso3D.asc;
a = torso3Dmatch(:,1);
a = a(130:length(a));

channels = torso3Dmatch(130:size(torso3Dmatch, 1),2);
[channels, i] = sort(channels);

v_mat = torso3D(a, :);

%% Montreal group use different co-ordinate system than Tamara's program
c(:, 1) =  v_mat(:, 1);
c(:, 2) =  v_mat(:, 3);
c(:, 3) =  -1 .* v_mat(:, 2);

%% Convert channel positions from feet to centimeters: 30.48 cm = 1 foot
ch_pos = c(i, :) .* 30.48;

%% Don't need to fill entire torso with cubes -- remove bottom 5 cm, makes
%% huge difference with timing
min_z = min(ch_pos(:, 3));
ch_pos(:, 3) = ch_pos(:, 3) - repmat(min_z + 10, length(a), 1);
```

```
function decide_program = pace_or_VT(C1, C2, channels, datafile, sample_rate, b_input1,
b_input2, VT_start)
```

```
%-------------------------------------------------------------------------%%
% pace_or_VT: Called by LabVIEW interface graphicalinterface.vi to
% implement inverse algorithm on selected data segment C1 to C2.
%
% Finds best channel by finding the channels with the maximum voltage
% swing, and analyzes this to find the heartrate (hr_detect.m). Based on
% heart rate and maximum gradient between two consecutive points, the
% program decides whether data is paced or VT (this is
% clear cut for the data samples used in this program -- a more general
% technique for separating the two might also include an analysis of
% shape). When Paceprogram or VTprogram have returned the values of the
% baselines and noise, program implements the inverse algorithm.
%
% INPUT: time segment of interest, number of channels, filename, sample
% rate, times delineating last isoelectric segment before VT, time of start
% of VT.
%
% OUTPUT: Dipole parameters are written to the file plotm.txt
%-------------------------------------------------------------------------%%

warning off;

% Load electrode positions, voltages recorded for each channel

[Exyz, R, len, r_torso, length_torso] = openSEMD(datafile, channels);
nos = C2 - C1 + 1;

%%-----------------------------------------------------------------------%%
%% Find the 'best channel'

if C1 < 101,
   start_point = 1;
else
   start_point = C1 - 100;
end

if C1 + 100 > len,
   end_point = len;
else
   end_point = C1 + 100;
end

contactVmeas = R([start_point: end_point], :);
p = max(contactVmeas, [], 1) - min(contactVmeas, [], 1);
[r, i] = max(p);
bestECG = R(:, i);

%%-----------------------------------------------------------------------%%
%% Find heart rate

heart_rate = hr_detect(C1, C2, channels, datafile, p(i), i, sample_rate, 0);

if C1 < 153,
   begin = 3;
else
```

```
   begin  = C1 - 153;
end

if C2 + 350 > len - 2,
   finish = len - 2;
else
   finish = C2 + 350;
end

d_val = abs(bestECG(begin - 2: finish - 2) - bestECG(begin + 2: finish + 2));
heart_rate;

%%-------------------------------------------------------------------------%%

%% Get values that depend on whether data is paced or VT

if (heart_rate < 200) & (max(d_val) > 1),
   %% IF DATA IS PACED

   [choosenoise, s_1, num_channels, basevalue] = Paceprogram(C1, C2, channels, datafile,
sample_rate, i);
   basevalue = repmat(basevalue, (C2 - C1 + 1), 1);

else
   %% IF DATA IS VT
   [choosenoise, s_1, num_channels] = VTprogram(C1, C2, channels, datafile, sample_rate, i);

   %% Find baseline using filtering approach
   %% Implement Raised Cosine filter on data to reduce edge effects

   if len < 1000,
      eT = 5;
   else
      eT = 100;
   end

   t = [1:eT];
   R(1:eT, :) = R(1:eT, :) .* repmat((sin(2*pi/(4 * eT) .* t))', 1, channels);
   rev_cos = fliplr(sin(2*pi/(4*eT) .* t));
   R(size(R, 1) - (eT - 1): size(R, 1), :) = R(size(R, 1) - (eT - 1): size(R, 1), :) .* repmat(rev_cos', 1,
channels);

   %% Find Heart rate during sinus rhythm/pacing -- to find maximum cutoff frequency

   preVThr = hr_detect(1, b_input1, channels, datafile, p(i), i, sample_rate, 1)

   %% Find cutoff frequency, depending on this heart rate

   base_mat = zeros(size(R, 1), channels);
   cf = 0.0044.* (preVThr + 0.1086)
   cf = cf/(sample_rate/2)

   %% Implement Butterworth filter, 4th-order

   [b, a] = butter(4, cf);
```

```
  %% Filter all channels
  for ch = 1: channels,

     y = filtfilt(b, a, R(:, ch));

     %% DC correct, using user-input of baseline prior to start of VT
     real_base = mean(R(b_input1:b_input2, ch));
     b_offset = y(VT_start) - real_base;
     y = y - repmat(b_offset, length(y), 1);
     base_mat(:, ch) = y;

  end

  basevalue = base_mat([C1:C2], choosenoise);

end

Vmeas_noisefree = R([C1:C2], choosenoise);
Exyz = Exyz(choosenoise, :);
sigma_Vmeas = sqrt(sum(s_1.^2)./num_channels);

%%-------------------------------------------------------------------------%%
%% INVERSE ALGORITHM IMPLEMENTATION
%% Calculate dipole parameters for each of chosen samples

xmatrix = zeros(1, nos);
ymatrix = zeros(1, nos);
zmatrix = zeros(1, nos);
thetamat = zeros(1, nos);
phimat = zeros(1, nos);
ampmatrix = zeros(1, nos);
Qfmatrix = zeros(1, nos);
Unx = zeros(1, nos);
Uny = zeros(1, nos);
Unz = zeros(1, nos);
Vesto = zeros(nos, num_channels);
Ar = zeros(1, nos);

chi_mat = [];
D_mat = [];

for sampleno = 1:1:nos,

   sampleno
   Vmeas = (Vmeas_noisefree(sampleno,:) - basevalue(sampleno, :));
   Vmeas = Vmeas';

   [Dxyz, mag_dipole, p_est, min_chi, Vest] = inverse_alg_new(r_torso, length_torso, Exyz,
Vmeas, sigma_Vmeas, num_channels);

   %% Root Normalized Mean Square Error
   RNMSE = sqrt (sum((Vest - Vmeas').^2)/sum(Vmeas.^2));

   %% Store dipole parameters in matrix
   Ar(sampleno) = RNMSE;
   D_mat = [D_mat; Dxyz];
```

```
    chi_mat = [chi_mat; min_chi];
    xmatrix(sampleno) = Dxyz(1);
    ymatrix(sampleno) = Dxyz(2);
    zmatrix(sampleno) = Dxyz(3);
    [thetaval, phival, r] = cart2sph(p_est(1), p_est(2), p_est(3));
    thetamat(sampleno) = thetaval;
    phimat(sampleno) = phival;
    ampmatrix(sampleno) = mag_dipole;
    Qfmatrix(sampleno) = min_chi;

    %% Find positional uncertainty of dipole
    [Px, Py, Pz] = posUnc(Dxyz, Exyz, p_est, Vest, Vmeas);

    Unx(sampleno) = Px;
    Uny(sampleno) = Py;
    Unz(sampleno) = Pz;

end

%% Normalize postional uncertainties for easier plotting
Unx = Unx/max(Unx);
Uny = Uny/max(Uny);
Unz = Unz/max(Unz);


%% Include extra information in matrix for GUI to read.

extravec = zeros(1, nos);
extravec(1) = channels;
extravec(2) = C1;
extravec(3) = C2;
extravec(4) = i;

extravec2 = zeros(1, nos);
extravec2(1:size(datafile, 2)) = datafile;

%% Store matrix M in file plotm.txt
M = [xmatrix; ymatrix; zmatrix; thetamat; phimat; ampmatrix; Qfmatrix; extravec; extravec2; Unx;
Uny; Unz; Ar];
fid2 = fopen('plotm.txt', 'w');
fprintf(fid2, '%5.4f \n', M);
fclose(fid2);

%% Call newsemd.m to plot dipoles and their parameters
newsemd
```

```
function heart_rate = hr_detect(C1, C2, channels, datafile, p_i, i, sample_rate, fixed)

%%------------------------------------------------------------------------%%
% hr_detect: finds the heart rate over the interval specified by C1 and C2, for the data in 'datafile'.
%
% Program finds the maximum value within the interval, and incrementally
% decreases a 'bar' set at this value until periodic peaks are found.
% The period between these peaks is found and used to estimate the heart
```

```
% rate based on the sample rate.
%%------------------------------------------------------------------------%%

[electrodeMatrix, R, len, r_torso, l_torso] = openSEMD(datafile, channels);

bestECG = R(:, i);
len = length(bestECG);

% Determine the period in which to calculate the heart rate. If fixed = 1,
% then heart rate is found within the exact period specified. Otherwise,
% heart rate is found for period +/- sample rate to ensure that enough
% peaks occur within the segment to get an accurate estimation.

if (C1 - sample_rate > 0) & (fixed == 0),
   begin = C1 - sample_rate;
elseif (fixed == 1),
   begin = C1;
else
   begin = 1;
end

if (C2 + sample_rate <= len) & (fixed == 0),
   finish = C2 + sample_rate;
elseif (fixed ==1),
   finish = C2;
else
   finish = len;
end

% Use high-quality ECG channel for periodicity approximation. Do not want a
% signal corrupted by noise.

sECG = bestECG(begin: finish);

[v_max, i_max] = max(sECG);

average_diff = inf;

%% Incrementally lower 'bar' from initial maximum value of the ECG, until
%% periodicity is found in the peaks that lie above this bar.

for bar_inc = 0.01:0.04:0.5,

   l_bound = v_max - bar_inc * p_i;
   V = find(sECG > l_bound);
   mo = length(V);

   if length(V) == 1,
      continue;

   else
      if ( (min(V) > (i_max - sample_rate/2)) & (max(V) < (i_max + sample_rate/2))),
         continue;

      else
         diff_V = diff(V);
```

```
        select = find(diff_V > 35);

        if isempty(select),
           continue;

        elseif length(select) < 3,
           continue;

        else
           find_d = diff_V(select);
           if (max(find_d) - min(find_d) < 10),
              average_diff = mean(find_d);
              break;
           else
              if (fixed == 0),
                 no_bins = (max(find_d) - min(find_d))/20;
                 m = 20;
              else
                 no_bins = (max(find_d) - min(find_d))/(sample_rate/2);
                 m = sample_rate/2;
              end

              [n, x] = hist(find_d, no_bins);
              [y, i] = max(n);
              if (y == 1) & (i ~= 1),          %% OR?
                 continue;
              else
                 box = x(i);
                 actual_d = find((diff_V(select) <= (box + m/2)) & (diff_V(select) >= (box - m/2)));
                 find_d(actual_d);
                 average_diff = mean(find_d(actual_d));
                 break;
              end

           end
        end
     end
  end
end

if bar_inc >= 0.48,  %%(average_diff == inf),
   h_period = len;
else
   h_period = average_diff;
end

heart_rate = 60/(h_period/sample_rate);
```

---

function [choosenoise, s_1, num_channels, basevalue] = **Paceprogram**(C1, C2, channels, datafile, sample_rate, i)

```
%%-------------------------------------------------------------------------%%
% Paceprogram: calculates the noise and baseline values for paced data, by
% examining the isoelectric segment that occurred before the selected data
```

```
% segment.
%
% Eradicates channels with a low SNR and/or high noise.
%
% Returns the noise and baseline values (s_1 and basevalue), the
% indices to the high-quality channels that have not been eliminated, and
% the number of channels left.
%%-------------------------------------------------------------------------%

warning off;

[electrodeMatrix, R, length, r_torso, length_torso] = openSEMD(datafile, channels);

% no. of samples chosen
nos = C2 - C1 + 1;

% submatrix of R, with samples vs. channels
Vmeasmatrix = R([C1:C2], :);

%%-------------------------------------------------------------------------%%
%% Find the 'best channel'

bestECG = R(:, i);

b = bestECG(C1-70:C1 + 70);

%% Plot representative data
% figure
% plot([C1-70:C1 + 70], b)
% hold on
%
% figure
% plot([C1-70:C1 + 70], R([C1-70:C1 + 70], 18), 'r')
% plot([C1-70:C1 + 70], R([C1-70:C1 + 70], 16), 'c')
% plot([C1-70:C1 + 70], R([C1-70:C1 + 70], 34), 'g')
% plot([C1-70:C1 + 70], R([C1-70:C1 + 70], 48), 'k')
% plot([C1-70:C1 + 70], R([C1-70:C1 + 70], 50), 'm')
% plot([C1-70:C1 + 70], R([C1-70:C1 + 70], 54), 'y')
% plot([C1-70:C1 + 70], R([C1-70:C1 + 70], 9), 'g')
% plot([C1-70:C1 + 70], R([C1-70:C1 + 70], 10), 'k')

%%-------------------------------------------------------------------------%%

%% Find time at end of T wave

t_spike = C1;
t_endT = C1 - 10;

if C1 < 81,
   start_point = 11;
else
   start_point = C1 - 70;
end

if C2 + 70 > length,
   end_point = length;
```

```
else
   end_point = C2 + 70;
end


for currentime1 = start_point: end_point,

   bit1 = (bestECG(currentime1 + 5) - bestECG(currentime1));
   bit2 = (bestECG(currentime1 - 10) - bestECG(currentime1 - 5));

   if ((abs(bit1) < (0.1 * abs(bit2))) & (bit2 > 0)),
      t_endT = currentime1
      break

   end
end

%%-------------------------------------------------------------------------%%

% Find time of spike -- REMEMBER, matlab and labview have different indexing schemes

for currentime2 = t_endT: t_endT + 100,

   bit1 = abs((bestECG(currentime2 + 1) - bestECG(currentime2)));
   [I,J,V] = find(abs(diff(bestECG(t_endT: currentime2))));
   bit2 = min(V);

   if (abs(bit1) > (10 * bit2)),
      t_spike = currentime2
      break;
   end
end

%%-------------------------------------------------------------------------%%

%% Find time at peak of QRS, t_QRS (approx), and values at maximum, V_QRS

bit_mat = zeros(1, 44);
r = 4;
for currentime3 = t_spike + 10: t_spike + 50,
   bit1 = (bestECG(currentime3 + 1) - bestECG(currentime3));
   bit_mat(r) = bit1;
   if ((sign(bit1) > 0) & (sign(bit_mat(r - 1)) >= 0) & (sign(bit_mat(r - 2)) <= 0) & (sign(bit_mat(r -
3)) <= 0)) ,
      t_QRS = currentime3 - 1
      break;
   end
   r = r+1;
end

V_QRS = min(R(t_QRS - 10: t_QRS + 10, :), [], 1);

%%-------------------------------------------------------------------------%%
%% Find time at BEGINNING of QRS, after pacing spike for best channel,
%% t_startQRS
```

```
diff_QRS = diff(R(t_spike:t_QRS, i));
diff2 = diff(diff_QRS);
currentime4 = t_spike;
R(currentime4 - 2, i);

for currentime4 = t_spike: t_QRS,
   if ((diff_QRS(currentime4 - t_spike + 1) < (mean(diff_QRS)/10)) ...
         & (diff2(currentime4 - t_spike + 1) < 0) ...
         & (R((currentime4 - 2), i) < R((currentime4), i)) ...
         & (R((currentime4 + 2), i) < R((currentime4), i))),
      t_startQRS = currentime4
      break
   end
end

V_startQRS = max(R(t_startQRS - 2: t_startQRS + 2, :), [], 1);

%%-------------------------------------------------------------------------%%

% Find 'good' channels based on amplitude of QRS compared to sigma_noise

voltage_swing = abs(V_QRS - V_startQRS);
sigma_mat = std(R([t_endT + 2:t_spike - 2], :));
[x, index_mat] = find(voltage_swing > (2 .* sigma_mat));
n_channels = size(index_mat, 2);

%%-------------------------------------------------------------------------%%
%% Calculate sigmas for each channel, discard channels with sigma > 0.12

sigma_mat = sigma_mat(index_mat);

choosenoise= [];

for ch = 1: n_channels,
   if sigma_mat(ch) < 0.12,     %% Represents good boundary between low and high noise
      choosenoise = [choosenoise, index_mat(1, ch)];
   end
end

%%-------------------------------------------------------------------------%%
%% FINAL VALUES TO INPUT INTO TAMARA'S PROGRAM

basevalue = mean(R((t_endT + 2):(t_spike - 2), choosenoise))

Vmeas_noisefree = Vmeasmatrix(:, choosenoise);

Exyz = electrodeMatrix(choosenoise, :);
num_channels = size(choosenoise, 2);

s_1 = std(R((t_endT + 2):(t_spike - 2), choosenoise), 0, 1);     % See p. 149 thesis

sigma_Vmeas = sqrt(sum(s_1.^2)./num_channels);
```

```
function [choosenoise, s_1, num_channels] = VTprogram(C1, C2, channels, datafile,
sample_rate, i)

%%-------------------------------------------------------------------------%%
% VTprogram: calculates the standard deviation of noise in VT data
% (in 'datafile'), calculates the SNR of each channel (by comparing the
% voltage swing in the QRS to the noise level) and eliminates channels with
% high noise and/or low SNR.
%
% Returns: the standard deviation in the noise in each channel (s_1), the
% indices of the channels that have not been eliminated (choosenoise), and
% the number of channels left for analysis (num_channels)
%%-------------------------------------------------------------------------%%
[electrodeMatrix, R, len, r_torso, length_torso] = openSEMD(datafile, channels);

bestECG = R(:, i);

Vmeasmatrix = R([C1:C2], :);
nos = C2 - C1 + 1;

%%-------------------------------------------------------------------------%%
%% Calculate the maximum voltage swing (v_swing) in each channel across the QRS
%% complex

if C1 < sample_rate/2,
   start_point = 1;
else
   start_point = C1 - sample_rate/2;
end

if C1 + sample_rate/2 > len,
   end_point = len;
else
   end_point = C1 + sample_rate/2;
end

contactVmeas = R([start_point: end_point], :);
v_swing = max(contactVmeas, [], 1) - min(contactVmeas, [], 1);

%%-------------------------------------------------------------------------%%
%% Calculate standard deviations of noise (sigma) for each channel, by
%% high-pass filtering the data with a fourth-order butterworth filter.
%% filtfilt.m filters in both the forward and backward directions,
%% producing zero phase distortion.

sigma_mat = [];

for ch = 1: channels,

   [b, a] = butter(4, 0.7, 'high');
   y = filtfilt(b, a, R(:, ch));
   sigma_mat = [sigma_mat, std(y(C1: C2))];

end

%%-------------------------------------------------------------------------%
```

```
% Find 'good' channels based on amplitude of QRS compared to sigma_noise
% (SNR). Discard channels with low SNR.

[x, index_mat] = find(v_swing > (2 .* sigma_mat));

n_channels = size(index_mat, 2);


%%-------------------------------------------------------------------------%%
%% Discard channels with high noise

sigma_mat = sigma_mat(index_mat);

choosenoise= [];

for ch = 1: n_channels,
   if sigma_mat(ch) < 0.12,
      choosenoise = [choosenoise, index_mat(1, ch)];
   end
end

%%-------------------------------------------------------------------------%%
%% FINAL VALUES TO INPUT INTO TAMARA'S PROGRAM

choosenoise = [1:channels];
Vmeas_noisefree = Vmeasmatrix(:, choosenoise);

Exyz = electrodeMatrix(choosenoise, :);
num_channels = size(choosenoise, 2);

s_1 = sigma_mat(choosenoise)
```

```
%*******************************************************
% Tamara Williams
% 02/28/03
% For use in Maya's hardware
%
%
% Input:   Exyz: electrode positions (60 x 3), r_i, t_i, p_i
%          Vmeas:  measured potentials (60 x 1)
%          r_torso: radius of pig's torso;
%          n:  # of electrodes
% Output:  Dxyz: dipole position
%          p_est:  moment parameters; orientation
%          Vest:  estimated potential values
%          mag_dipole: magnitude of the estimated dipole
%          min_chi:  minimum of chi^2 function
%*******************************************************
%%% Old version, as saved on April 21st 2003

function [Dxyz, mag_dipole, p_est, min_chi, Vest] = inverse_alg_new(r_torso, length_torso,
Exyz, Vmeas, sigma_Vmeas, n);
%***************Inverse Algorithm*********************
%Thanks Peter Boettcher <boettcher@ll.mit.edu> for mex file used to multiply 3D matrices
```

```
mex ndfun.c C:\matlab_sv13\extern\lib\win32\microsoft\msvc60\libmwlapack.lib
step = 1.5;
argu = 1;
choose_chi = 1;

while argu == 1,
   % Fill a cylinder instead of sphere
   vertices = cylinder_fill_cube(r_torso, length_torso, step(1));
   [p_est, Vest, Vmeas1] = moment(vertices, Exyz, Vmeas, sigma_Vmeas, n);
   [min_chi,index(1)] = chi2(Vest, Vmeas1, sigma_Vmeas, n, choose_chi);
   estimated_dipole(1,:)=vertices(index(1), :);
   CUBE(1,:)=3*[step(1) step(1) step(1)];
   vertex(1,:) = estimated_dipole(1, :);

   for i=2:4

      step(i) = step(i-1)/3;
      CUBE(i,:) = 3*[step(i) step(i) step(i)];

      vertex(i,:) = estimated_dipole(i-1,:);
      clear vertices
      vertices = cube_fill_cube3(vertex(i,:),CUBE(i-1,:), step(i));
      [p_est, Vest, Vmeas1] = moment(vertices, Exyz, Vmeas, sigma_Vmeas, n);
      [min_chi,index(i)] = chi2(Vest, Vmeas1, sigma_Vmeas, n, choose_chi);

      estimated_dipole(i,:)= vertices(index(i), :);
      p_newest = p_est(:,:, index(i));
      v_newest = Vest(:,:, index(i));

   end;

   Dxyz = estimated_dipole(4,:);
   if (( (sqrt( (Dxyz(1))^2 + (Dxyz(2))^2))<= r_torso) & (Dxyz(3) < length_torso))
      argu = 0;
   else
      choose_chi = choose_chi + 1;
   end
end

Vest = v_newest;
p_est = p_newest;
mag_dipole=sqrt(p_est(1)^2 + p_est(2)^2 + p_est(3)^2);
```

```
%*********************************************************************
% Tamara Williams
% 01/14/03
% This program takes a sphere and fills it with cubes whose size has been
% determined by the user
%
%
% Input:  r_torso: torso radius
%         scale: size of cubes filling the sphere
%         step: step size used in filling the sphere in x,y, and z
```

```
%              directions
% Output:  D_est:  represents the lower, right, front vertex of each cube formed
%         cube_center:  The center of each cube
%*********************************************************************

function vertices = cylinder_fill_cube(r_torso,length_torso, step);

% Initialize the output parameters by setting up empty matrices that will
% be filled. Step across the entire diameter of the torso.

ii=[1:step:2*r_torso-step];
ki=[1:step:length_torso-step];
% In case you step by a fraction, matlab's find command finds the step of
% the vector ii
i=find(ii);
k=find(ki);
shift = r_torso - ii(i) - 0.5*step;   %12.5 -> 11.5
shiftk=length_torso - ki(k) -  0.5*step;
[shiftx,shifty,shiftz]=meshgrid(shift, shift, shiftk);

% Only analyze cubes found inside the torso

vertices=[shiftx(:) shifty(:) shiftz(:)];

z_vertices=vertices(:,3);
ind=find(z_vertices <= length_torso);
vertices2=[vertices(:,1) vertices(:,2)];
mag_vert=sqrt(sum(vertices2.^2,2));
[ind2]=find(mag_vert < r_torso);

vertices=vertices(ind2,:);
```

```
function [p_est, Vest, Vmeas1] = moment(vertices, Exyz, Vmeas, sigma_Vmeas,n);

%%-------------------------------------------------------------------------%%
% Tamara Williams
% 01/14/03
% Use 3 plus 3 parameter optimization to find dipole moments and
% compute new potential.  Use least squares to find the dipole moment

%%-------------------------------------------------------------------------%%

Exyz=reshape(repmat(Exyz',length(vertices),1)',n,3,[]); % m x 3 x n

%Repeat each dipole vector n times then make it the size of Exyz
vertices=vertices';

%Use this to match with Exyz
verticesx=reshape(repmat(vertices(:),1,n)',n,3,[]);

E2=permute(Exyz,[2,1,3]);
D2=permute(verticesx,[2,1,3]);

mag3=(sum((Exyz-verticesx).*(Exyz-verticesx),2)).^(-1.5);
```

```
mag33=[mag3 mag3 mag3];

%Produces the best solution from the entire matrix

const=1/(sigma_Vmeas(1)^2*4*pi*1.2e-3);
M1=const*(E2-D2); % 3 x 60 x m
M2=(Exyz-verticesx).*mag33.*mag33;

M = ndfun('mult', M1,M2);

Vmeas=repmat(Vmeas,length(vertices),1);  % 1 x 60 x m
Vmeas1=reshape(Vmeas,1,n,[]);

%Scale by 100 to make dipole more realistic

B0=const*(Exyz-verticesx).*mag33;
B=ndfun('mult', Vmeas1, B0);

Minv = ndfun('inv', M);
p_est = ndfun('mult', B, Minv);

mag332=permute(mag33,[2 1 3]);

A2=(E2-D2).*mag332;
Vest=ndfun('mult', p_est,A2); % 1 x n x m
```

```
%**************************************************************************
% Tamara Williams
% 06/20/01
%
% Minimize the objective function and compute the dipole location
% parameters using the Brute Force method
%
% Input:  dipole_params: the outputs of the random seed search for the true dipole location
%         Ex, Ey, Ez:  electrodes in CARTESIAN coordinates
%         Vmeas:  Measured electrode potentials
%         sigma_Vmeas: standard deviation of Vmeas
%
% Output:  MIN_CHI: the minimum chi^2 objective function
%
% Function Call:  threeplusthree: Uses the 3-plus-3 paramter optimization method to determine
%                 the moment values of the dipole by using least squares estimation
%**************************************************************************
%% CHI2 as saved on April 21st, 2003

function [min_chi,index] = chi2(Vest, Vmeas1, sigma_Vmeas, n, choose_chi);
C1 =(Vest-Vmeas1)./sigma_Vmeas(1);
C2 = permute(C1,[2 1 3]);
CHI2 = ndfun('mult',C1,C2);
CHI2 = (CHI2)/(n-3);

[min_chi_mat,i_mat] = sort(CHI2);

index = i_mat(choose_chi);
min_chi = min_chi_mat(choose_chi);
```

```
%**********************************************************************
% Tamara Williams
% 01/14/03
% This program takes a cube and fills it with cubes whose size has been
% determined by the user
%
% Input:  CUBE: a 1x3 vector representing the large cube to be filled with
%             smaller cubes
%         vertex: upper left vertex of the outer cube if starting from point at
%                 center of cube
%         scale: size of cubes filling the sphere
%         step: step size used in filling the sphere in x,y,and z
%               directions
% Output:  D_est:  represents the lower, right, front vertex of each cube
%                  formed
%          cube_center:  center of each cube filling the cube
%**********************************************************************
function vertices = cube_fill_cube3(vertex, CUBE, step);

%Set color of the cube for display purposes only
color=[0.5 0.5 0.5];

% Form a cube centered around estimated_dipole
Vert(1,:)=vertex;%-1.5*step;

Vert(5,:)=[vertex(1) vertex(2) vertex(3)+CUBE(3)];
Vert(2,:)=[CUBE(1)+vertex(1) vertex(2) vertex(3)];
Vert(6,:)=[CUBE(1)+vertex(1) vertex(2) CUBE(3)+vertex(3)];
Vert(3,:)=[CUBE(1)+vertex(1) vertex(2)+CUBE(2) vertex(3)];
Vert(7,:)=[CUBE(1)+vertex(1) CUBE(2)+vertex(2) CUBE(3)+vertex(3)];
Vert(4,:)=[vertex(1) CUBE(2)+vertex(2) vertex(3)];
Vert(8,:)=[vertex(1) CUBE(2)+vertex(2) CUBE(3)+vertex(3)];

Vert=[ ...
      Vert(1,:);
   Vert(2,:);
   Vert(3,:);
   Vert(4,:);
   Vert(5,:);
   Vert(6,:);
   Vert(7,:);
   Vert(8,:)];


Faces = [ ...
      1 2 6 5 ;
   2 3 7 6 ;
   3 4 8 7 ;
   4 1 5 8 ;
   1 2 3 4 ;
   5 6 7 8 ];

% Begin filling the large cube at the center
```

```
ii2 = vertex(1):step:vertex(1)+CUBE/2;
ii = vertex(1)-step:-step:vertex(1)-CUBE/2;
ii=fliplr(ii);
ii_all=[ii ii2];

jj2 = vertex(2):step:vertex(2)+CUBE/2;
jj = vertex(2)-step:-step:vertex(2)-CUBE/2;
jj=fliplr(jj);
jj_all=[jj jj2];

kk2 = vertex(3):step:vertex(3)+CUBE/2;
kk = vertex(3)-step:-step:vertex(3)-CUBE/2;
kk=fliplr(kk);
kk_all=[kk kk2];

[shiftX,shiftY,shiftZ]=meshgrid(ii_all, jj_all, kk_all);
vertices=[shiftX(:) shiftY(:) shiftZ(:)];
```

```
function [Px, Py, Pz] = posUnc(Dxyz, Exyz, p_est, Vest, Vmeas)

%%------------------------------------------------------------------------%%
% Function posUnc.m:
% Returns the positional uncertainties of the dipole,Px, Py and Pz.
% Inputs specified by the co-ordinates Dxyz and moment p_est,
% with electrodes at the positions Exyz and the differential error in the
% estimate Vest - Vmeas.
%%------------------------------------------------------------------------%%

%% Initialize matrix A for mapping from dipole parameters to lead voltages
A = zeros(size(Exyz, 1), 6);

delta_par = diag([Dxyz, p_est]);

%% Use 1% deviation for each dipole parameter
delta_par = 0.01 .* delta_par;

vertices = zeros(1, 3);
p_estn = zeros(1, 3);

%% Generate column vectors for matrix A by looking at change in V with
%% incremental changes in the dipole parameters, x.

for dp = 1:6,

   M = [Dxyz, p_est] + delta_par(dp, :);
   vertices = M(1:3);
   p_estn = M(4:6);
   Vestn = moment3(vertices, Exyz, p_estn);
   del_T = Vest - Vestn;
   m = find(delta_par(dp, :));
   m = delta_par(dp, m);
   A(:, dp) = (del_T')/m;
```

```
end

%% Use least-squares approximation, p. 418 of statistics
%% Find unbiased approximation of standard deviation in the error in V

SSE = sum((Vmeas' - Vest).^2);
s_square_V = SSE/(length(Vmeas) - 6 - 1);

%% Generate covariance-variance matrix, and take the diagonal elements for
%% the variance only.
cov_m = inv(A'*A);
s_square_X = (diag(cov_m))' .* s_square_V;
s_X = sqrt(s_square_X);

Px = s_X(1);
Py = s_X(2);
Pz = s_X(3);
```

```
function Vestn = moment3(vertices, Exyz, p_est);

%-------------------------------------------------------------------------%%
%% moment3: estimates the voltages (Vestn) at electrode locations specified by Exyz
%% due to a dipole located at 'vertices' with moment 'p_est'.
%%
%% vertices =  1 x 3 vector
%% Exyz     =  60 by 3 matrix
%% p_est    =  1 x 3 (delta_x, delta_y, delta_z)
%% Vest     =  1 x channels
%-------------------------------------------------------------------------%%
% Use 3 plus 3 parameter optimization to find dipole moments and
% compute new potential.

vertices=repmat(vertices, length(Exyz),1);
mag3=((sum((Exyz-vertices).*(Exyz-vertices),2))).^(-1.5);
mag33=[mag3 mag3 mag3];
A2=(Exyz'-vertices').*mag33';

Vestn = p_est*A2;
```

```
function varargout = newsemd(varargin)

%-------------------------------------------------------------------------%%
% newsemd: controls callback functions for Cardiac Dipole Display GUI.
% Called by pace_or_VT.m after completion of dipole calculations. Reads
% dipole parameters from file 'plotm.txt'.

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

% Last Modified by GUIDE v2.5 10-Jun-2003 23:40:20
%-------------------------------------------------------------------------%%
```

```
global matrix
global choice

if nargin == 0  % LAUNCH GUI

        fig = openfig(mfilename,'reuse');

        % 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


%% Read matrix of locations, magnitudes etc. from file plotm.txt

warning off;

fid = fopen('plotm.txt');
[A, COUNT] = fscanf(fid, '%f', inf);
global nos
nos = COUNT/13;  % nos = number of samples chosen
matrix = zeros(13, nos);
p = 1;
for j = 1: nos
   for i = 1:13
      matrix(i, j) = A(p);
      p = p + 1;
   end
end

%%% Set axes limits for dipole axes, and plot dipoles

r = handles.axes1;
axes(r)
vecx = matrix(1, :);
vecy = matrix(2, :);
vecz = matrix(3, :);

minx = min(vecx);
maxx = max(vecx);
miny = min(vecy);
maxy = max(vecy);
minz = min(vecz);
maxz = max(vecz);

lim = 1;

global xl
global yl
global zl
xl = [minx - lim, maxx + lim];
yl = [miny - lim, maxy + lim];
zl = [minz - lim, maxz + lim];
```

```
set(r, 'XLimMode', 'manual', 'YLimMode', 'manual', 'ZLimMode', 'manual');
set(r, 'XLim', xl, 'YLim', yl, 'ZLim', zl);

showspheres(nos, [1:nos], 1)
hold on

plot3(vecx, vecy, vecz, ':')
hold off

%% Initialize Camera View Angle

r = handles.axes1;
axes(r)
set(r, 'CameraPosition', [20.9229   62.0510   41.7169]);

%% Annotate axes to indicate up, down, right, left

m1 = get(gca, 'XLim');
n1 = get(gca, 'YLim');
o1 = get(gca, 'ZLim');

halfx = (m1(1) + m1(2))/2;
halfy = (n1(1) + n1(2))/2;
halfz = (o1(1) + o1(2))/2;

text(m1(2) + 2, halfy, halfz, 'LEFT');
text(m1(1) - 2, halfy, halfz, 'RIGHT');
text(halfx, halfy, o1(1), 'DOWN');
text(halfx, halfy, o1(2), 'UP');

%%% Set slider values to match input

set(handles.slider1, 'Max', nos);
set(handles.slider1, 'SliderStep', [1/nos, 1/nos]);

%% Plot ECG channel tracing on lower axes

m = handles.ECGaxes;
axes(m);
grid on
channels = matrix(8, 1);
global C1
global C2
C1 = matrix(8, 2);
C2 = matrix(8, 3);
dataline = matrix(9, :);
datafile = dataline(find(dataline));

[e_mat, R, length, r_torso, length_torso] = openSEMD(datafile, channels);
bestchannel = matrix(8, 4);
global chosenECGvec
chosenECGvec = R(:, bestchannel);

global begin
global finish
```

```
if C1 - 20 > 0,
   begin = C1 - 20;
else
   begin = 1;
end

if C2 + 20 < length,
   finish = C2 + 20;
else
   finish = length;
end

samplevalues = [begin:finish];
plot(samplevalues, chosenECGvec([begin :finish]));
hold on;
plot(C1, chosenECGvec(C1), 'k*');
plot(C2, chosenECGvec(C2), 'k*');
axis tight
axis on;
grid on

%% Allow dipole axes to rotate, not ECG chart

set(m, 'hittest', 'off')
axes(r);
cameratoolbar('setmode', 'orbit')

%% Plot auxiliary window for additional dipole information

figure; plotall(C1, C2, datafile, channels)

% INVOKE NAMED SUBFUNCTION OR CALLBACK
elseif ischar(varargin{1})
        try
                if (nargout)
                        [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
                else
                        feval(varargin{:}); % FEVAL switchyard

                end
        catch
                disp(lasterr);
        end

end


%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:h
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%|
```

```
%| The subfunction name is composed using the object's Tag and the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in GUI using
%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and stored in
%| the figure's application data using GUIDATA. A copy of the structure
%| is passed to each callback.  You can store additional information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks.  Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks see
%| the updates. Type "help guihandles" and "help guidata" for more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property to:
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.

% --------------------------------------------------------------------
function varargout = slider1_Callback(h, eventdata, handles, varargin)
%% Moving the slider changes the dipole displayed.

global matrix
global choice
global nos

% Choice = type of dipole display (spheres, wireframes or just dipole vectors)

m = handles.ECGaxes;
axes(m);
set(m, 'CameraPosition', [1480, 0, 17.3])

choice = get(handles.popupmenu1, 'Value');
dipno = get(handles.slider1, 'Value');

%% If being called by zoom in function
if dipno == 0,
   dipno = 1;
end

dipno = round(dipno);
set(handles.slider1, 'Value', dipno);
set(handles.DipoleNumber, 'String', num2str(dipno));
set(handles.slider1, 'Min', 1);

r = handles.axes1;
axes(r)
ret = get(r, 'CameraPosition');
```

```
hold off;

global rhrel
rhrel = max(matrix(6, :));
rh = matrix(6, dipno)/rhrel;

%% Plot both selected dipole and dipole trajectory
imbed_animation(rh, matrix(1,dipno), matrix(2,dipno), matrix(3,dipno), matrix(5, dipno), matrix(4,
dipno), matrix(10, dipno), matrix(11, dipno), matrix(12, dipno), choice);
hold on
showspheres(nos, [1:nos], 1);

vecx = matrix(1, :);
vecy = matrix(2, :);
vecz = matrix(3, :);

plot3(vecx, vecy, vecz, ':')

global xl
global yl
global zl

set(r, 'XLim', xl)
set(r, 'YLim', yl)
set(r, 'ZLim', zl)
set(r, 'CameraPosition', ret);

%% Annotate axes to indicate up, down, right, left

m1 = get(gca, 'XLim');
n1 = get(gca, 'YLim');
o1 = get(gca, 'ZLim');

halfx = (m1(1) + m1(2))/2;
halfy = (n1(1) + n1(2))/2;
halfz = (o1(1) + o1(2))/2;

text(m1(2) + 2, halfy, halfz, 'LEFT');
text(m1(1) - 2, halfy, halfz, 'RIGHT');
text(halfx, halfy, o1(1), 'DOWN');
text(halfx, halfy, o1(2), 'UP');
hold off;

% Display numerical values in the GUI of current dipole selected

c = handles.MagHeart;
set(c, 'String', num2str(matrix(6, dipno)));
d = handles.HOrient;
set(d, 'String', num2str([matrix(5, dipno), matrix(4, dipno)], 3));
e = handles.QualityFit;
set(e, 'String', num2str(matrix(7, dipno)));
f = handles.HeartLoc;
set(f, 'String', sprintf('%1.2f   %1.2f   %1.2f', matrix(1,dipno), matrix(2,dipno), matrix(3,dipno)));
k = handles.PosCert;
set(k, 'String', sprintf('%1.2f   %1.2f   %1.2f', matrix(10,dipno), matrix(11,dipno),
matrix(12,dipno)));
```

```
%% Plot indicator of location of current dipole in time on the ECG axes.

global C1
global C2
global chosenECGvec
global begin
global finish

m = handles.ECGaxes;
axes(m);
set(m, 'hittest', 'off')
hold off;
samplevalues = [begin: finish];
plot(samplevalues, chosenECGvec([begin: finish]));
axis off
hold on;
plot(C1 + dipno -1, chosenECGvec(C1 + dipno - 1), 'ro');
plot(C1, chosenECGvec(C1), 'k*');
plot(C2, chosenECGvec(C2), 'k*');
axis on
axis tight

grid on
cameratoolbar('setmode', 'nomode')
set(m, 'hittest', 'off')
cameratoolbar('setmode', 'orbit')
axes(r);

guidata(gcbo, handles);

% --------------------------------------------------------------------
function varargout = DipoleNumber_Callback(h, eventdata, handles, varargin)

%% Same action as for slider, except that this function is called when the
%% user directly enters the number of the dipole to be displayed.

global matrix;
global choice;
global nos;

choice = get(handles.popupmenu1, 'Value');
dipno = str2num(get(handles.DipoleNumber, 'String'));

previous = get(handles.slider1, 'Value');
r = handles.axes1;

if isnumeric(dipno) & length(dipno) == 1 & dipno >=get(handles.slider1, 'Min') & dipno <=
get(handles.slider1, 'Max')
   set(handles.slider1, 'Value', dipno);

   axes(r);
   ret = get(r, 'CameraPosition');
   hold off;
   global rhrel
   rh = matrix(6, dipno)/rhrel;
```

```
   imbed_animation(rh, matrix(1,dipno), matrix(2,dipno), matrix(3,dipno), matrix(5, dipno),
matrix(4, dipno), matrix(10, dipno), matrix(11, dipno), matrix(12, dipno), choice);
   hold on
   showspheres(nos, [1:nos], 1);

   vecx = matrix(1, :);
   vecy = matrix(2, :);
   vecz = matrix(3, :);

   plot3(vecx, vecy, vecz, ':')

   set(r, 'CameraPosition', ret);
   global xl
   global yl
   global zl
   set(r, 'XLim', xl)
   set(r, 'YLim', yl)
   set(r, 'ZLim', zl)
   hold off;

   %% Annotate axes to indicate up, down, right, left

   m1 = get(gca, 'XLim');
   n1 = get(gca, 'YLim');
   o1 = get(gca, 'ZLim');

   halfx = (m1(1) + m1(2))/2;
   halfy = (n1(1) + n1(2))/2;
   halfz = (o1(1) + o1(2))/2;

   text(m1(2) + 2, halfy, halfz, 'LEFT');
   text(m1(1) - 2, halfy, halfz, 'RIGHT');
   text(halfx, halfy, o1(1), 'DOWN');
   text(halfx, halfy, o1(2), 'UP');


   % Display numerical values in the gui, based on current dipole selected

   c = handles.MagHeart;
   set(c, 'String', num2str(matrix(6, dipno)));
   d = handles.HOrient;
   set(d, 'String', num2str([matrix(5, dipno), matrix(4, dipno)], 3));
   e = handles.QualityFit;
   set(e, 'String', num2str(matrix(7, dipno)));
   f = handles.HeartLoc;
   set(f, 'String', sprintf('%1.2f   %1.2f   %1.2f', matrix(1,dipno), matrix(2,dipno), matrix(3,dipno)));
   k = handles.PosCert;
   set(k, 'String', sprintf('%1.2f   %1.2f   %1.2f', matrix(10,dipno), matrix(11,dipno),
matrix(12,dipno)));

   global C1
         global C2
         global chosenECGvec
   global begin
   global finish
```

```
        m = handles.ECGaxes;
    axes(m);
    set(m, 'hittest', 'off')
        hold off;
        plot([begin: finish], chosenECGvec([begin: finish]));
        hold on;
        plot(C1 + dipno -1, chosenECGvec(C1 + dipno - 1), 'ro');
    plot(C1, chosenECGvec(C1), 'k*');
    plot(C2, chosenECGvec(C2), 'k*');

    axis tight
    grid on
    axis on
    axes(r);

    guidata(gcbo, handles);

else set(handles.DipoleNumber, 'String', num2str(previous));
    axes(r);

end


% --- Executes on button press in zoom

function zoom_Callback(hObject, eventdata, handles)

%% Zooms in on selected dipole

global matrix;
global nos;

% Wait for user to click on dipole of interest
k = waitforbuttonpress;
point2 = get(gca,'CurrentPoint');    % button down detected

% Generate vector of points within axes that are 'below' the point clicked
% by the user, and find dipole closest to any of these.
point1 = zeros(11, 3);
point1(1, :) = point2(1, :);
point1(2, :) = point2(1, :) + 0.1.*(point2(2, :) - point2(1, :));
point1(3, :) = point2(1, :) + 0.2.*(point2(2, :) - point2(1, :));
point1(4, :) = point2(1, :) + 0.3.*(point2(2, :) - point2(1, :));
point1(5, :) = point2(1, :) + 0.4.*(point2(2, :) - point2(1, :));
point1(6, :) = point2(1, :) + 0.5.*(point2(2, :) - point2(1, :));
point1(7, :) = point2(1, :) + 0.6.*(point2(2, :) - point2(1, :));
point1(8, :) = point2(1, :) + 0.7.*(point2(2, :) - point2(1, :));
point1(9, :) = point2(1, :) + 0.8.*(point2(2, :) - point2(1, :));
point1(10, :) = point2(1, :) + 0.9.*(point2(2, :) - point2(1, :));
point1(11, :) = point2(2, :);

vecx = matrix(1, :);
vecy = matrix(2, :);
vecz = matrix(3, :);
```

```
y1 = zeros(1, 11);
i1 = zeros(1, 11);

for index = 1:11,
   points = repmat(point1(index, :), length(vecx), 1);
   place = [vecx' vecy' vecz'];
   dist = sqrt(sum((place - points).^2, 2));,
   [y1(index), i1(index)] = min(dist);
end

% Nearest Dipole
[y1, j1] = min(y1);
i1  = i1(j1);

choose = [vecx(i1) vecy(i1) vecz(i1)];

axis manual

%% Find all dipoles within 4 cm-sided cube of selected dipole.

xb = choose(1) - 2;
xe = choose(1) + 2;
yb = choose(2) - 2;
ye = choose(2) + 2;
zb = choose(3) - 2;
ze = choose(3) + 2;

axis([xb xe yb ye zb ze])

in1 = find(vecx > xb & vecx < xe);
in2 = find(vecy > yb & vecy < ye);
in3 = find(vecz > zb & vecz < ze);

p = [];

%% Plot only nearest dipoles and selected dipole on CONTIGUOUS segment
%% of the trajectory..

for ind = 1:length(in1),
   m_mat = find(in2 == in1(ind));
   if isempty(m_mat),
      continue;
   else
      n_mat = find(in3 == in1(ind));

      if isempty(n_mat),
         continue;
      else
         p = [p, in1(ind)];
      end
   end
end

%% Choose section around selected dipole.

bar = diff(p);
```

```
in4 = find(bar ~= 1); %% dipole index within p corresponding to a 'break'

if length(in4) == 1,    % Only one break --> cut off p
   if i1 > p(in4),
      p_vec = p(in4 + 1: length(p));
   else
      p_vec = p(1: in4 - 1);
   end

elseif isempty(in4), %% No other sections of trajectory are present, p is final

   p_vec = p;

else              %% More than 1 section, must choose which section to keep
   m = p(in4);  %% in4 = indices where diff changes, m = values of p corresponding to these
indices
   in5 = find(m < i1);      %% Find where values are less than selected dipole
   in5 = m(max(in5));    %% Gives you the value at which diff changes just below your chosen
dipole
   in6 = find(m > i1);      %% Find where values are MORE than selected dipole
   in6 = m(min(in6));   %% Gives you the dipole value at which diff changes just above your
dipole

   if isempty(in5),
      s = 1;
   else
      s = find(p == in5);  %% Find the index in p corresponding to lower-break dipole
   end

   if isempty(in6),
      e = length(p);
   else
      e = find(p == in6);  %% Find the index in p corresponding to the upper-break dipole
   end

   p_vec = p(s + 1:e - 1);  %% Select p between the two indices


end

p = p_vec;

hold off

r = handles.axes1;
axes(r);
ret = get(r, 'CameraPosition');

dipno = get(handles.slider1, 'Value');

showspheres(nos, p, 0)
plot3(vecx(p), vecy(p), vecz(p), 'k-')
hold on

% Plot selected dipole if it is within the zoomed-in segment.
```

```
if ~isempty(find(p == dipno)),
   imbed_animation(1, matrix(1,dipno), matrix(2,dipno), matrix(3,dipno), matrix(5, dipno),
matrix(4, dipno), matrix(10, dipno), matrix(11, dipno), matrix(12, dipno), 3);
end

%% Annotate axes to indicate up, down, right, left

m1 = get(gca, 'XLim');
n1 = get(gca, 'YLim');
o1 = get(gca, 'ZLim');

halfx = (m1(1) + m1(2))/2;
halfy = (n1(1) + n1(2))/2;
halfz = (o1(1) + o1(2))/2;

text(m1(2) + 0.5, halfy, halfz, 'LEFT');
text(m1(1) - 0.5, halfy, halfz, 'RIGHT');
text(halfx, halfy, o1(1), 'DOWN');
text(halfx, halfy, o1(2), 'UP');

set(r, 'CameraPosition', ret);
hold off

% --------------------------------------------------------------------
function varargout = selectbutton_Callback(h, eventdata, handles, varargin)
% Select this dipole as the ablation site.

global matrix
global rhrel

m = gcf;
dipno = get(handles.slider1, 'Value');

rh = matrix(6, dipno)/rhrel;
vectorval = matrix(:, dipno);
vectorval = [vectorval', rh];

% Save values of chosen heart vector dipole in file finaldipole.txt

fid = fopen('finaldipole.txt', 'w');
fprintf(fid, '%5.4f \n', vectorval);
fclose(fid);
close all
c=catheterdisp; % Call catheterdisp.m

% --------------------------------------------------------------------
function varargout = figure1_ButtonDownFcn(h, eventdata, handles, varargin)

r = handles.axes1;
m = handles.ECGaxes;
set(m, 'hittest', 'off')
axes(r);
```

```
function plotit = plotall(C1, C2, datafile, channels)

%%------------------------------------------------------------------------%%
% Plotall generates the auxiliary window that appears alongside the Cardiac
% Dipole Display, showing the trend in several dipole parameters over the
% time segment selected by the user for SEMD analysis.
%%------------------------------------------------------------------------%%


fid = fopen('plotm.txt');
[A, COUNT] = fscanf(fid, '%f', inf);
global nos
nos = COUNT/13;  % NOS = NUMBER OF SAMPLES CHOSEN
matrix = zeros(13, nos);
p = 1;
for j = 1: nos
   for i = 1:13
      matrix(i, j) = A(p);
      p = p + 1;
   end
end

%% Calculations required to plot graphs of parameters
%% For plot of distance between adjacent dipoles

dx = matrix(1, 2:size(matrix, 2)) - matrix(1, 1:size(matrix, 2) - 1);
dy = matrix(2, 2:size(matrix, 2)) - matrix(2, 1:size(matrix, 2) - 1);
dz = matrix(3, 2:size(matrix, 2)) - matrix(3, 1:size(matrix, 2) - 1);

dist_vec = (dx.^2 + dy.^2 + dz.^2).^(0.5);
dist_vec = [0, dist_vec];

%% For plot of ECG
[mm, R, mm, mm, mm] = openSEMD(datafile, channels);
bestECG = matrix(8, 4);

%% Plot all parameters

subplot(8, 1, 1), plot(C1:C2, R(C1:C2, bestECG));
ylabel('ECG')
axis tight

subplot(8, 1, 2), plot(C1:C2, matrix(6, :));
ylabel('Amplitude')
axis tight

subplot(8, 1, 3), plot(C1:C2, dist_vec);
ylabel('Distance')
axis tight

subplot(8, 1, 4), plot(C1:C2, matrix(7, :));
ylabel('Chi^2')
axis tight

subplot(8, 1, 5), plot(C1:C2, matrix(13, :));
ylabel('RNMSE')
```

```
axis tight

subplot(8, 1, 6), plot(C1:C2, matrix(10, :));
ylabel('Px')
axis tight

subplot(8, 1, 7), plot(C1:C2, matrix(11, :));
ylabel('Py')
axis tight
```

```
subplot(8, 1, 8), plot(C1:C2, matrix(12, :));
ylabel('Pz')
axis tight

%% Label x axis, and set size of window
xlabel('t (samples)')
set(gcf, 'Position', [232, 45, 650, 640])
```

```
function path = showspheres(nos, vec, holder)

%------------------------------------------------------------
% Showspheres: plots dipole 'dots' for all dipoles in the file
% plotm.txt. These are color-coded to indicate magnitude (red = high
% magnitude, blue = low-magnitude, on a sliding scale).
%
% Shows the trajectory of the dipole over the selected time segment.
%
% nos = number of dipole samples chosen by user to plot
% vec = indices of dipoles to plot within file plotm.txt
% holder = whether to allow next program to write over dipole display.
%------------------------------------------------------------

% Read matrix of locations, magnitudes etc. from file plotmatrix.txt

fid = fopen('plotm.txt');
hold off

[A, COUNT] = fscanf(fid, '%f', inf);
matrix = zeros(13, nos);
p = 1;
for j = 1: nos
   for i = 1:13
      matrix(i, j) = A(p);
      p = p + 1;
   end
end

% Initialize vectors of colours related to magnitudes

redindex = zeros(1, nos);
blueindex = zeros(1, nos);
```

```
% Make basic sphere for a dipole 'dot'
```

```
[X, Y, Z] = sphere(30);
X2 = X./15;
Y2 = Y./15;
Z2 = Z./15;

if holder == 1,
   hold on
end

%% Normalize magnitudes to grade magnitude on a scale from 0 to 1 for color-coding.
```

```
relativeamp = max(matrix(6, :));
redindex= matrix(6, :)/relativeamp;
blueindex = 1 - (matrix(6, :)/relativeamp);

%% Plot dipole trajectory with associated color indicating magnitude

for m = 1:length(vec),
   n = vec(m);
```

```
   surf(X2 + matrix(1, n), Y2 + matrix(2, n), Z2 + matrix(3, n), 'facecolor', [redindex(n), 0,
blueindex(n)], 'edgecolor', [redindex(n), 0, blueindex(n)])
   hold on
end

% Modify plot

axis equal
grid on
box on
rotate3D ON
cameratoolbar('SetMode', 'orbit')
```

```
function showdipole = imbed_animation(rh, xh, yh, zh, ohphi, ohtheta, xcert, ycert, zcert, choice,
rc, xc, yc, zc, ocphi, octheta)

% ------------------------------------------------------------------------------------------
% Creates 3D graphical representation of location and orientation of heart and catheter dipoles

% rh = relative magnitude of heart dipole
% rc = relative magnitude of catheter dipole
% (xh, yh, zh) = co-ordinates of heart dipole
% (xc, yc, zc) = co-ordinates of catheter dipole
% (ohphi, ohtheta) = orientation of heart dipole: angle in X-Y plane, angle from Z-axis (in radians)
% (ocphi, octheta) = orientation of catheter dipole: angle in X-Y plane, angle from Z-axis (in
radians)
% xcert, ycert, zcert = positional uncertainty of heart dipole. Changes
% dimensions of ellipsoid.
% fit = [0, 1] --> measure of goodness of fit. Numeri
% choice = [1, 2, 3] for whether sphere is full surface, wireframe or not present
```

```
% Calls arrow3.m to create arrow representation of dipoles

%-------------------------------------------------------------------------------------------------

%% Set maximum limit on dimensions of ellipsoid

if xcert > 7,
   xcert = 7;
end

if ycert > 7,
   ycert = 7;
end

if zcert > 7,
   zcert = 7;
end

%% Calculate ellipsoid parameters CENTERED at [0, 0, 0]

[X, Y, Z] = ellipsoid(xh, yh, zh, xcert, ycert, zcert, 30);

%% Calculate end co-ordinates of heart dipole vectors

if rh == 1,
   m_fac = 0.5;
else
   m_fac = 1.25;
end

endxh = xh + m_fac*sin(ohtheta) * cos(ohphi);
endyh = yh + m_fac*sin(ohtheta) * sin(ohphi);
endzh = zh + m_fac*cos(ohtheta);

%% Draw arrow vectors with length proportional to settings and color according to fit

hvec = arrow3([xh, yh, zh], [endxh, endyh, endzh], 0);
hold on

%% ------------------------------------------------------- %%
%% Calculate small-sphere parameters for heart and catheter 'points' at base of vectors

[X2, Y2, Z2] = sphere;

Xhp = xh + X2/65;
Yhp = yh + Y2/65;
Zhp = zh + Z2/65;

% Higher the amplitude (rh), the more red the sphere; the lower the amplitude the more blue

   redindex = rh;
   blueindex = 1 - rh;

%% Create catheter and heart dipole ellipsoids, accoording to choice in
%% pull-down menu on Dipole Display Interface
```

```
switch choice
case 1
   pointheart = surf(Xhp, Yhp, Zhp, 'FaceColor', 'k');
   heartsphere = surf(X, Y, Z, 'FaceAlpha', 0.3, 'EdgeAlpha', 0.5, 'FaceColor', [redindex, 0,
blueindex]);

case 2
   pointheart = surf(Xhp, Yhp, Zhp, 'FaceColor', 'k');
   heartsphere = mesh(X, Y, Z, 'FaceColor', 'none', 'EdgeAlpha', 0.3, 'EdgeColor', [redindex, 0,
blueindex]);

case 3
   heartsphere = surf(Xhp, Yhp, Zhp, 'FaceColor', 'k');
end

%% if Ablation Catheter is also to be plotted (otherwise nargin = 10),
%% plot ablation dipole also.

if (nargin == 16),

   endxc = xc + (0.75 * sin(octheta) * cos(ocphi));
   endyc = yc + (0.75 * sin(octheta) * sin(ocphi));
   endzc = zc + (0.75 * cos(octheta));

   cvec = arrow3([xc, yc, zc], [endxc, endyc, endzc], 1);

   Xcp = xc + X2/75;
   Ycp = yc + Y2/75;
   Zcp = zc + Z2/75;

   X2 = xc + 0.1 * X2;
   Y2 = yc + 0.1 * Y2;
   Z2 = zc + 0.1 * Z2;

   switch choice
   case 1
    pointcath = surf(Xcp, Ycp, Zcp, 'FaceColor', 'k');
    cathsphere = surf(X2, Y2, Z2, 'FaceAlpha', 0.3, 'EdgeAlpha', 0.5); %, 'FaceColor', 'y');

   case 2
     pointcath = surf(Xcp, Ycp, Zcp, 'FaceColor', 'k');
     cathsphere = mesh(X2, Y2, Z2,'FaceColor', 'none', 'EdgeAlpha', 0.7); %, 'EdgeColor', 'y');

   case 3
     cathsphere = surf(Xcp, Ycp, Zcp, 'FaceColor', 'm', 'EdgeColor', 'm');
   end

   legend([hvec, cvec], 'Heart', 'Catheter', 1);
end

%% Modify plot characteristics
xlabel('x', 'FontSize', 14)
ylabel('y', 'FontSize', 14)
zlabel('z', 'FontSize', 14)
grid ON
```

box ON

hold off

```
function varargout = catheterdisp(varargin)

%% ------------------------------------------------------------------------%%
% Catheterdisp: controls the Ablation Catheter Display GUI. The
% callback functions below are activated by pressing the corresponding
% button on the interface.
%
% Called by newsemd.m, the Cardiac Dipole Display interface. Catheterdisp
% plots the selected cardiac dipole and the (moving) dipole corresponding
% to the ablation catheter tip.
%
% In this program, ablation catheter movement is simulated using a
% 'for' loop that increments the catheter coordinates every 2.5 seconds
% (the time required for an inverse algorithm calculation).
%% ------------------------------------------------------------------------%%

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

% Last Modified by GUIDE v2.0 24-Feb-2003 14:35:50

if nargin == 0  % LAUNCH GUI

   fig = openfig(mfilename,'new');

   % Use system color scheme for figure:
   set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

   % 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

   %% Open finaldipole.txt, a file written to by newsemd.m containing the
   %% parameters of the dipole selected by the user to represent the site
   %% of arrhythmic origin.

   fid = fopen('finaldipole.txt');
   [A, COUNT] = fscanf(fid, '%f', inf);
   [X, Y, Z] = sphere(25);

   xh = A(1);
   yh = A(2);
   zh = A(3);
   thetah = A(4);
   phih = A(5);
   magh = A(6);
```

```
Qfh = A(7);
xcert = A(10);
ycert = A(11);
zcert = A(12);
rh = A(14);

global M
M = A;

r = handles.axes2;
set(gcf, 'CurrentAxes', r);

%% Plot cardiac dipole ellipsoid and catheter dipole sphere on same
%% axes, make axes tight and annotate for orientation.

global xc
global yc
global zc

xc = 0;
yc = 1;
zc = 2;

imbed_animation(rh, xh, yh, zh, phih, thetah, xcert, ycert, zcert, 2, 1, xc, yc, zc, pi, pi/3);
cameratoolbar('setmode', 'orbit');
axis equal
axis tight

m1 = get(gca, 'XLim');
n1 = get(gca, 'YLim');
o1 = get(gca, 'ZLim');

halfx = (m1(1) + m1(2))/2;
halfy = (n1(1) + n1(2))/2;
halfz = (o1(1) + o1(2))/2;

text(m1(2) + 2, halfy, halfz, 'LEFT');
text(m1(1) - 2, halfy, halfz, 'RIGHT');
text(halfx, halfy, o1(1), 'DOWN');
text(halfx, halfy, o1(2), 'UP');

%% Assign values to numerical display

c = handles.hmag;
set(c, 'String', num2str(magh, 3));
d = handles.cmag;
set(d, 'String', num2str(1, 3));

e = handles.hangle;
set(e, 'String', num2str([phih, thetah], 3));
f = handles.cangle;
set(f, 'String', num2str([pi, pi/3], 3));

g = handles.hloc;
set(g, 'String', sprintf('%1.2f   %1.2f   %1.2f', xh, yh, zh));
k = handles.cloc;
```

```
    set(k, 'String', sprintf('%1.2f   %1.2f   %1.2f', xc, yc, zc));

    guidata(fig, handles);

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

    try
        if (nargout)
            [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        else
            feval(varargin{:}); % FEVAL switchyard
        end
    catch
        disp(lasterr);
    end

end

%% Animate catheter display, using for loop to increment catheter position
global n;
n = 1;
p = 0;

fig = gcf;
handles = guihandles(fig);

fid = fopen('finaldipole.txt');
[A, COUNT] = fscanf(fid, '%f', inf);
[X, Y, Z] = sphere(25);

xh = A(1);
yh = A(2);
zh = A(3);
thetah = A(4);
phih = A(5);
magh = A(6);
Qfh = A(7);
xcert = A(10);
ycert = A(11);
zcert = A(12);
rh = A(14);

global M
M = A;

global xc
global yc
global zc

xdist = (xh - xc)/10;
ydist = (yh - yc)/10;
zdist = (zh - zc)/10;

%% Pause 2.5 seconds between each.
while (p < 10),
    if (get(handles.stopbutton, 'Value') ~= 1),
```

```
    pause(2.5);

    r = handles.axes2;
    set(gcf, 'CurrentAxes', r);
    ret = get(r, 'CameraPosition');
    choice = get(handles.popupmenu1, 'Value');

    xc = xc + xdist;
    yc = yc + ydist;
    zc = zc + zdist;

    imbed_animation(rh, xh, yh, zh, phih, thetah, xcert, ycert, zcert, choice, 1, xc, yc, zc, pi, pi/3);
    set(r, 'CameraPosition', ret);
    axis equal
    axis tight

    % Annotate axes
    m1 = get(gca, 'XLim');
    n1 = get(gca, 'YLim');
    o1 = get(gca, 'ZLim');

    halfx = (m1(1) + m1(2))/2;
    halfy = (n1(1) + n1(2))/2;
    halfz = (o1(1) + o1(2))/2;

    text(m1(2) + 2, halfy, halfz, 'LEFT');
    text(m1(1) - 2, halfy, halfz, 'RIGHT');
    text(halfx, halfy, o1(1), 'DOWN');
    text(halfx, halfy, o1(2), 'UP');

    % Assign values to numerical display
    c = handles.hmag;
    set(c, 'String', num2str(magh, 3));
    d = handles.cmag;
    set(d, 'String', num2str(1, 3));

    e = handles.hangle;
    set(e, 'String', num2str([phih, thetah], 3));
    f = handles.cangle;
    set(f, 'String', num2str([pi, pi/3], 3));

    g = handles.hloc;
    set(g, 'String', sprintf('%1.2f   %1.2f   %1.2f', xh, yh, zh));
    k = handles.cloc;
    set(k, 'String', sprintf('%1.2f   %1.2f   %1.2f', xc, yc, zc));

    guidata(fig, handles);
    p = p+1;

else

    n = 0;
    break

end
```

```
end


% --------------------------------------------------------------------
function varargout = popupmenu1_Callback(h, eventdata, handles, varargin)

%% Allow cardiologist to choose type of dipole display
global M;

choice = get(h, 'Value');

xh = A(1);
yh = A(2);
zh = A(3);
thetah = A(4);
phih = A(5);
magh = A(6);
Qfh = A(7);
xcert = A(10);
ycert = A(11);
zcert = A(12);
rh = A(14);

r = handles.axes2;
set(gcf, 'CurrentAxes', r);
ret = get(r, 'CameraPosition');

global xc
global yc
global zc

imbed_animation(rh, xh, yh, zh, phih, thetah, xcert, ycert, zcert, choice, 1, xc, yc, zc, pi, pi/3);
axis equal
axis tight

m1 = get(gca, 'XLim');
n1 = get(gca, 'YLim');
o1 = get(gca, 'ZLim');

halfx = (m1(1) + m1(2))/2;
halfy = (n1(1) + n1(2))/2;
halfz = (o1(1) + o1(2))/2;

text(m1(2) + 2, halfy, halfz, 'LEFT');
text(m1(1) - 2, halfy, halfz, 'RIGHT');
text(halfx, halfy, o1(1), 'DOWN');
text(halfx, halfy, o1(2), 'UP');

set(r, 'CameraPosition', ret);

% --------------------------------------------------------------------
function varargout = stopbutton_Callback(h, eventdata, handles, varargin)
%% if user presses 'STOP' button, break for loop to end animation.

global n
```

```
n = 0;
```

```
function [L_h,F_h]=arrow3(Start, Stop, Fit, Width, ALen, Base, SLen, BLineStyle);

%-------------------------------------------------------
% arrow         plot 3-D arrow with following properties:
%
%               A-D: SLen                ^ A
%               A-C: ALen               / \
%               A  : Stop point        /   \
%               B  : Start point      /     \
%               E-F: Base            / ..D.. \
%                                 E/..  IC ..\F
%                                    |
%                                    IB
% Input  : - Start points. [X Y Z]
%          - Stop points. [X Y Z]
%          - Arrow length. (default is 0.3 of total length).
%          - Base length.  (default is 1/2 of arrow length).
%          - Short length. (default is like Arrow length).
%          - line width. (default is 1.1).
%          - Certainty. Positional certainty. Decides whether arrow line is solid or dashed.
%          - Fit. Quality of fit (default is '1', decides color of arrow).
%          - line style. (default is 'none').
% Output : - Line handle.
%          - Arrow (filled) handle.

%% Use arrow width 2 -> 5
%-------------------------------------------------------

Length = sqrt((Stop(1)-Start(1)).^2+(Stop(2)-Start(2)).^2 + (Stop(3)-Start(3)).^2 );

if (nargin==8),
  % no default.
elseif (nargin==7),
  BLineStyle = 'none';
elseif (nargin==6),
  SLen      = ALen;
  BLineStyle = 'none';
elseif (nargin==5),
  Base      = 0.5.*ALen;
  SLen      = ALen;
  BLineStyle = 'none';
elseif (nargin==4),
  ALen      = 0.3.*Length;
  SLen      = ALen;
  Base      = 0.5.*ALen;
  BLineStyle = 'none';
elseif (nargin==3),
  Width     = 1.1;
  ALen      = 0.3.*Length;
  Base      = 0.5.*ALen;
  SLen      = ALen;
  BLineStyle = 'none';
elseif (nargin==2),
```

```
  Fit       = 1;
  Width     = 1.1;
  ALen      = 0.3.*Length;
  Base      = 0.5.*ALen;
  SLen      = ALen;
  BLineStyle = 'none';
else
  error('Illegal number of input arguments');
end

if (Fit == 0)
   Color = 'g';
else
   Color = 'r';
end

L_h= plot3([Start(1),Stop(1)],[Start(2),Stop(2)], [Start(3), Stop(3)], Color);
set(L_h,'LineWidth',Width);
hold on

%% End of arrow head
A(1) = Stop(1);
A(2) = Stop(2);
A(3) = Stop(3);

Theta  = atan2(Stop(2)-Start(2),Stop(1)-Start(1));
VTheta = atan2(sqrt((Stop(1)-Start(1)).^2+(Stop(2)-Start(2)).^2), Stop(3) - Start(3));


%% Medial Bottom of arrow head (short length)
D(1) = A(1) - SLen.*cos(Theta).*sin(VTheta);
D(2) = A(2) - SLen.*sin(Theta).*sin(VTheta);
D(3) = A(3) - SLen.*cos(VTheta);

Phi  = atan(0.5.*Base./ALen);

%% Calculate length of side of arrow head
BC   = sqrt(ALen.^2 + (0.5.*Base).^2);

%% Calculate bottom side point of arrow head
E(1) = A(1) - ALen.*cos(Theta).*sin(VTheta) + ((Base/2).*cos(pi/2 - Theta));
E(2) = A(2) - ALen.*sin(Theta).*sin(VTheta) - ((Base/2).*sin(pi/2 - Theta));
E(3) = A(3) - ALen.*cos(VTheta);

%% Calculate top side point of arrow head
F(1) = A(1) - ALen.*cos(Theta).*sin(VTheta) - ((Base/2).*cos(pi/2 - Theta));
F(2) = A(2) - ALen.*sin(Theta).*sin(VTheta) + ((Base/2).*sin(pi/2 - Theta));
F(3) = A(3) - ALen.*cos(VTheta);

F_h=fill3([A(1);E(1);D(1);F(1)],[A(2);E(2);D(2);F(2)], [A(3);E(3);D(3);F(3)],Color);
set(F_h,'LineStyle',BLineStyle);
```

```
function finddist = distance(mint, maxt)

%%-------------------------------------------------------------------%%
```

```
%% Finds maximum distance between any two dipoles within the segment of
%% trajectory between dipoles A and B

%%------------------------------------------------------------------------%%

fid = fopen('plotm.txt');
[A, COUNT] = fscanf(fid, '%f', inf);
nos = COUNT/12;  % NOS = NUMBER OF SAMPLES CHOSEN
matrix = zeros(12, nos);
p = 1;
for j = 1: nos
   for i = 1:12
      matrix(i, j) = A(p);
      p = p + 1;
   end
end

vecx = matrix(1, mint:maxt);
vecy = matrix(2, mint:maxt);
vecz = matrix(3, mint:maxt);
place = [vecx' vecy' vecz'];

find_dist = 0;

for m = 1: (maxt - mint + 1),
   for n = (m+1): (maxt - mint + 1),
      d_p = place(m, :) - place(n, :);
      arf = sqrt(sum(d_p.^2));
      if arf > find_dist,
         find_dist = arf;
         max_m = m;
         max_n = n;
      end
   end
end
```